



# **Implementation of carry save adder in Radix 10 multiplier**

**M.Vijayalakshmi<sup>1</sup>, Ch.Vijayalakshmi<sup>2</sup>, G.Vinatha<sup>3</sup>, B.Ravichandar<sup>4</sup>**

Associate professor, Dept. of ECE, St. Martin's Engineering College, Secunderabad, India<sup>1</sup>

Assistant professor, Dept. of ECE, St. Martin's Engineering College, Secunderabad, India<sup>2</sup>

Assistant professor, Dept. of ECE, St. Martin's Engineering College, Secunderabad, India<sup>3</sup>

Assistant professor, Dept. of ECE, St. Martin's Engineering College, Secunderabad, India<sup>4</sup>

**ABSTRACT:** We present a BCD parallel multiplier having some properties of two different redundant BCD codes to speed up its computation. 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. This encoding has several advantages. The partial products can be recoded to the ODDS representation by just adding a constant factor into the partial product reduction tree. Since the ODDS uses a similar 4-bit binary encoding as non-redundant BCD, conventional binary VLSI circuit techniques, such as binary carry-save adders and compressor trees, can be adapted efficiently to perform decimal operations.

**KEYWORDS:** Parallel multiplication, decimal hardware, overloaded BCD representation, redundant excess-3 code, redundant arithmetic

## **I. INTRODUCTION**

DECIMAL fixed-point and floating-point formats are important in financial, commercial, and user-oriented computing, where conversion and rounding errors that are inherent to floating-point binary representations cannot be tolerated. The new IEEE 754-2008 Standard for Floating-Point Arithmetic, which contains a format and specification for decimal floating-point (DFP) arithmetic has encouraged a significant amount of research in decimal hardware. Since area and power dissipation are critical design factors in state-of-the-art DFPU's, 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 DFPU's. Thus, efficient algorithms for accelerating DFP multiplication should result in regular VLSI layouts that allow an aggressive pipelining. Hardware implementations normally use BCD instead of binary to manipulate decimal fixed-point operands and integer significands of DFP numbers for easy conversion between machine and user representations. BCD encodes a number  $X$  in decimal (non-redundant radix-10) format, with each decimal digit  $X_i \in [0,9]$  represented in a 4-bit binary number system. However, BCD is less efficient for encoding integers than binary, since codes 10 to 15 are unused. Moreover, the implementation of BCD arithmetic has more complications than binary, which lead to area and delay penalties in the resulting arithmetic units. A variety of redundant decimal formats and arithmetics 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 position. It is intended for carry-free accumulation of BCD partial products using rows of BCD digit adders arranged in linear or tree-like configurations.

Decimal signed-digit (SD) representations rely on a redundant digit set  $\{a; \dots; 0; \dots; a\}$ ,  $5 < a < 9$ , to allow decimal carry-free addition. BCD carry-save and signed-digit radix-10 arithmetic offer improvements in performance with respect to nonredundant BCD. However, the resultant VLSI implementations in current technologies of multioperand adder trees may result in more irregular layouts than binary carry-save adders (CSA) and compressor trees. The overloaded BCD (or ODDS—overloaded decimal digit set) representation was proposed to improve decimal multioperand addition, and sequential and parallel decimal multiplications. In this code, each 4-bit binary value represents a redundant radix-10 digit  $X_i \in [0, 15]$ . The ODDS presents interesting properties for a fast and efficient



# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2015

hardware implementation of decimal arithmetic.(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 additional hardware to correct invalid 4-bit combinations. A disadvantage with respect to signed-digit and self-complementing codes, is a slightly more complex implementation of 9's complement operation for negation of operands and subtraction. 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 [-3, 12]. We use a minimally redundant digit set for the recoding of the BCD multiplier digits, the signed-digit radix-10 recoding [30], that is, the recoded signed digits are in the set {-5,-4,-3,-2,-1, 0, 1, 2, 3, 4, 5}. For this digit set, the main issue is to perform the x3 multiple without long carry-propagation (note that x2 and x5 are easy multiples for decimal and that x4 is generated as two consecutive x2 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 positive 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 decimal 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.

## II. REDUNDANT BCD REPRESENTATIONS

The proposed decimal multiplier uses internally a redundant BCD arithmetic to speed up and simplify the implementation. This arithmetic deals with radix-10 ten's complement integers of the form:

$$Z = -S_z * 10^d + \sum_{i=0}^{d-1} Z_i * 10^i,$$

Where d is the number of digits,  $S_z$  is the sign bit, and  $Z_i \in [-e, m-e]$  is the ith digit, with

$$0 \leq l \leq e, 9 + e \leq m \leq 2^4 - 1$$

Parameter e is the excess of the representation and usually takes values 0 (non excess), 3 or 6. The redundancy index p is defined as  $p=m-l+1-r$ , being  $r=10$ . On the other hand, the binary value of the 4-bit vector representation of  $Z_i$  is given by

$$[Z_i] = \sum_{j=0}^3 Z_{i,j} * 2^j,$$

$Z_{i,j}$  being the jth bit of the ith digit. Therefore, the value of digit  $Z_i$  can be obtained by subtracting the excess e of the representation from the binary value of its 4-bit encoding, that is,

$$Z_i = [Z_i] - e,$$

Note that bit-weighted codes such as BCD and ODDS use the 4-bit binary encoding (or BCD encoding. Thus,  $Z_i=[Z_i]$  for operands Z represented in BCD or ODDS.

This binary encoding simplifies the hardware implementation of decimal arithmetic units, since we can make use of state-of-the-art binary logic and binary arithmetic techniques to implement digit operations. In particular, the ODDS representation presents interesting properties (redundancy and binary encoding of its digit set) for a fast and efficient



## International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

**Vol. 4, Issue 11, November 2015**

implementation of multiplier and addition. Moreover, conversions from BCD to the ODDS representation are straightforward, since the digit set of BCD is a subset of the ODDS representation.

In our work we use a SD radix-10 recoding of the BCD multiplier [30], which requires to compute a set of decimal multiples ( $\{-5X \dots 0X, \dots, 5X\}$ ) of the BCD multiplicand. The main issue is to perform the x3 multiple without long carry-propagation.

For input digits of the multiplicand in conventional BCD (i.e., in the range  $[0, 9]$ ,  $e=0$ ,  $p=0$ ), the multiplication by 3 leads to a maximum decimal carry to the next position of 2 and to a maximum value of the interim digit (the result digit before adding the carry from the lower position) of 9. Therefore the resultant maximum digit (after adding the decimal carry and the interim digit) is 11. Thus, the range of the digits after the x3 multiplication is in the range  $[0, 11]$ . Therefore the redundant BCD representations can host the resultant digits with just one decimal carry propagation.

An important issue for this representation is the ten's complement operation. Since after the recoding of the multiplier digits, negative multiplication digits may result, it is necessary to negate (ten's complement) the multiplicand to obtain the negative partial products. This operation is usually done by computing the nine's complement of the multiplicand and adding a one in the proper place on the digit array. The nine's complement of a positive decimal operand is given by

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

The implementation of  $(9 - Z_i)$  leads to a complex implementation, since the  $Z_i$  digits of the multiples generated may take values higher than 9. A simple implementation is obtained by observing that the excess-3 of the nine's complement of an operand is equal to the bit complement of the operand coded in excess-3.

**Table 1 : Nine's complement for the Xs-3 representation**

Digit	Digit			Nine's Complement		
	4-bit Encoding	$Z_i$	$[Z_i]$	4-bit Encoding	$9-Z_i$	$9-[Z_i]$
0000	0000	-3	0	1111	12	15
0001	0001	-2	1	1110	11	14
0010	0010	-1	2	1101	10	13
0011	0011	0	3	1100	9	12
0100	0100	1	4	1011	8	11
0101	0101	2	5	1010	7	10
0110	0110	3	6	1001	6	9
0111	0111	4	7	1000	5	8
1000	1000	5	8	0111	4	7
1001	1001	6	9	0110	3	6
1010	1010	7	10	0101	2	5
1011	1011	8	11	0100	1	4
1100	1100	9	12	0011	0	3
1101	1101	10	13	0010	-1	2
1110	1110	11	14	0001	-2	1
1111	1111	12	15	0000	-3	0

In Table 1 we show how the nine's complement can be performed by simply inverting the bits of a digit  $Z_i$  coded in XS-3. At the decimal digit level, this is due to the fact that:

$$(9 - Z_i) + 3 = 15 - (Z_i + 3)$$

for the ranges  $Z_i \in [-3, 12]$  ( $[Z_i] \in [0, 15]$ ). Therefore to have a simple negation for partial product generation we produce the decimal multiples in an excess-3 code. The negation is performed by simple bit inversion, that corresponds to the excess-3 of the nine's complement of the multiple. Moreover, to simplify the implementation we combine the

# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

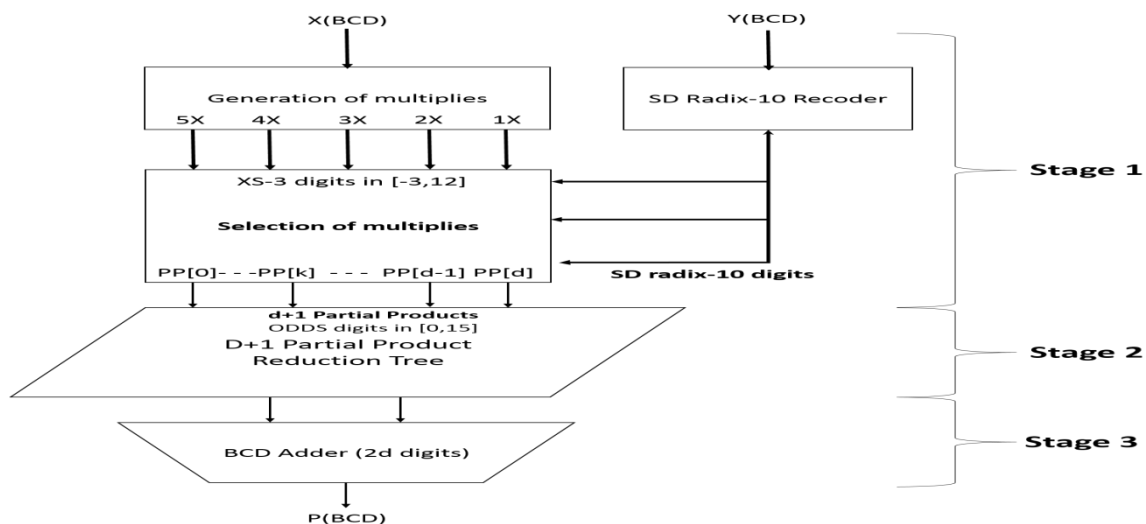
Vol. 4, Issue 11, November 2015

multiple generation stage and the digit increment by 3 (to produce the excess-3) into a single module by using the XS-3 code.

In summary, the main reasons for using the redundant XS-3 code are: (1) to avoid long carry-propagations in the generation of decimal positive multiplicand multiples, (2) to obtain the negative multiples from the corresponding positive ones easily, (3) simple conversion of the partial products generated in XS-3 to the ODDS representation for efficient partial product reduction.

### III. HIGH-LEVEL ARCHITECTURE

The high-level block diagram of the proposed parallel architecture for  $d$ -digit BCD decimal integer and fixed-point multiplication is shown in Fig. 1. This architecture accepts conventional (non-redundant) BCD inputs  $X, Y$ , generates redundant BCD partial products  $PP$ , and computes the BCD product  $P = X \times Y$ . It consists of the following three stages: (1) parallel generation of partial products coded in XS-3, including generation of multiplicand multiples and recoding of the multiplier operand, (2) recoding of partial products from XS-3 to the ODDS representation and subsequent reduction, and (3) final conversion to a non-redundant  $2d$ -digit BCD product.



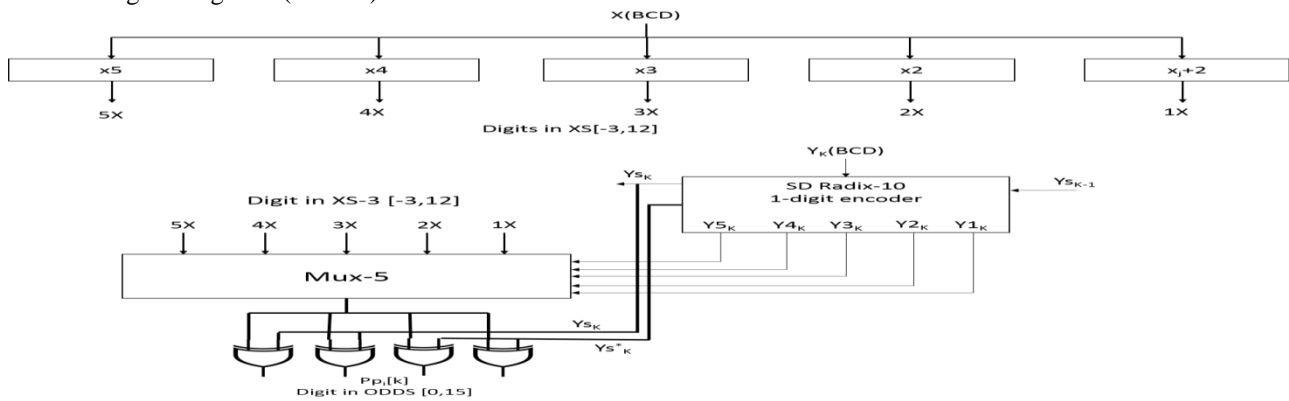
**Stage 1) Decimal partial product generation:** A SD radix-10 recoding of the BCD multiplier has been used. This recoding produces a reduced number of partial products that leads to a significant reduction in the overall multiplier area. Therefore, the recoding of the  $d$ -digit multiplier  $Y$  into SD radix-10 digits  $Y_{bd-1}, \dots, Y_{b0}$ , produces  $d$  partial products  $PP[d-1], \dots, PP[0]$ , one per digit; note that each  $Y_{bk}$  recoded digit is represented in a 6-bit hot-one code to be used as control input of the multiplexers for selecting the proper multiplicand multiple,  $\{-5X \dots, -1X, 0X, 1X, \dots, 5X\}$ . An additional partial product  $PP(d)$  is produced by the most significant multiplier digit after the recoding, so that the total number of partial products generated is  $d+1$ .

**Stage 2) Decimal partial product reduction:** In this stage, the array of  $d+1$  ODDS partial products are reduced to two  $2d$ -digit words ( $A, B$ ). Our proposal relies on a binary carrysave adder tree to perform carry-free additions of the decimal partial products. The array of  $d+1$  ODDS partial products can be viewed as adjacent digit columns of height  $h < d+1$ . 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).

**Stage 3) 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  $P = A+B$ . The proposed architecture is a  $2d$ -digit hybrid parallel prefix/carry-select adder, the BCD Quaternary Tree adder. The sum of input digits  $A_i, B_i$  at each position  $i$  has to be in the range  $[0,18]$  so that at most one decimal carry is propagated to the next position  $i + 1$ . Furthermore, to generate the correct decimal carry, the BCD addition algorithm implemented requires  $A_i + B_i$  to be obtained in excess-6. Several choices are possible. We opt for representing operand  $A$  in BCD excess-6 ( $A_i \in [0, 9], [A_i] = A_i + e, e = 6$ ), and  $B$  coded in BCD ( $B_i \in [0, 9], e = 0$ ).

### IV. DECIMAL PARTIAL PRODUCT GENERATION

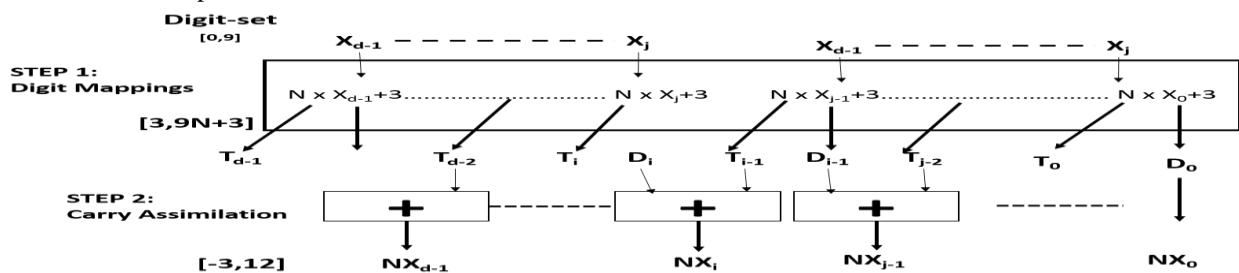
The partial product generation stage comprises the recoding of the multiplier to a SD radix-10 representation, the calculation of the multiplicand multiples in XS-3 code and the generation of the ODDS partial products. The negative multiples are obtained by ten's complementing the positive ones. This 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  $Y_{bk}$  by using a set of 5:1 muxes, as shown in Fig. 2. 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 ( $Y_{sk} = 1$ )



**Fig. 2: SD radix-10 generation of a partial product digit.**

#### Generation of the Multiplicand Multiples

Fig. 3 shows the high-level block diagram of the multiples generation with just one carry propagation. This is performed in two steps:



**Fig. 3: Generation of a decimal multiples  $NX$ .**

1) Digit recoding of the BCD multiplicand digits  $X_i$  into a decimal carry  $0 < T_i < T_{max}$  and a digit  $-3 < D_i < 12 - T_{max}$ , such as  $D_i + 10 * T_i = (N * X_i) + 3$  being  $T_{max}$  the maximum possible value for the decimal carry.



# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2015

2) The decimal carries transferred between adjacent digits are assimilated obtaining the correct 4-bit representation of XS-3 digits  $NX_i$ , that is  $[NX_i] = D_i + T_{i-1}, NX_i \in [0,15](NX_i \in [-3,12])$ ,  
The constraint for  $NX_i$  still allows different implementations for  $NX$ . For a specific implementation, the mappings for  $T_i$  and  $D_i$  have to be selected. Table 2 shows the preferred digit recoding for the multiples  $NX$ . Then, by inverting the bits of the representation of  $NX$ , operation defined at the  $i^{\text{th}}$  digit by  $\overline{NX}_i = 15[NX_i]$ ,  
Replacing the relation between  $NX_i$  and  $[NX_i]$  in the previous expression, it follows that

$$\overline{NX}_i = 15(NX_i + 3) = (9 - NX_i) + 3,$$

## V.MOST-SIGNIFICANT DIGIT ENCODING

The MSD of each  $PP[k]$ ,  $PP[dk]$ , is directly obtained in the ODDS representation. Note that these digits store the carries generated in the computation of the multiplicand multiples and the sign bit of the partial product. For positive partial products we have

$$PP_d[k] = T_{d-1},$$

with  $T_{d-1} \in \{0, 1, 2, 3, 4\}$ . Therefore the two cases can be expressed as

$$PP_d[k] = -10 + (9 - T_{d-1}) = -1 - T_{d-1},$$

$$PP_d[k] = -8 + [PP_d[k]],$$

With

$$[PP_d[k]] = 8 - Y_{sk} + (-1)^{Y_{sk}} T_{d-1},$$

TABLE 2  
Preferred Digit Recoding Mappings for  $NX$  Multiples

$X_i$	1X			2X			3X			4X			5X		
	$X_i+3$	$T_i$	$D_i$	$X_i+3$	$T_i$	$D_i$	$X_i+3$	$T_i$	$D_i$	$X_i+3$	$T_i$	$D_i$	$X_i+3$	$T_i$	$D_i$
0	3	0	3	3	0	3	3	0	3	3	0	3	3	0	3
1	4	0	4	5	0	5	6	0	6	7	0	7	8	0	8
2	5	0	5	7	0	7	9	0	9	11	1	1	13	1	3
3	6	0	6	9	0	9	12	0	12	15	1	5	18	1	8
4	7	0	7	11	1	1	15	1	5	19	1	9	23	2	3
5	8	0	8	13	1	3	18	1	8	23	2	3	28	2	8
6	9	0	9	15	1	5	21	2	1	27	2	7	33	3	3
7	10	0	10	17	1	7	24	2	4	31	2	11	38	3	8
8	11	0	11	19	1	9	27	2	7	35	3	5	43	4	3
9	12	0	12	21	1	11	30	2	10	39	3	9	48	4	8

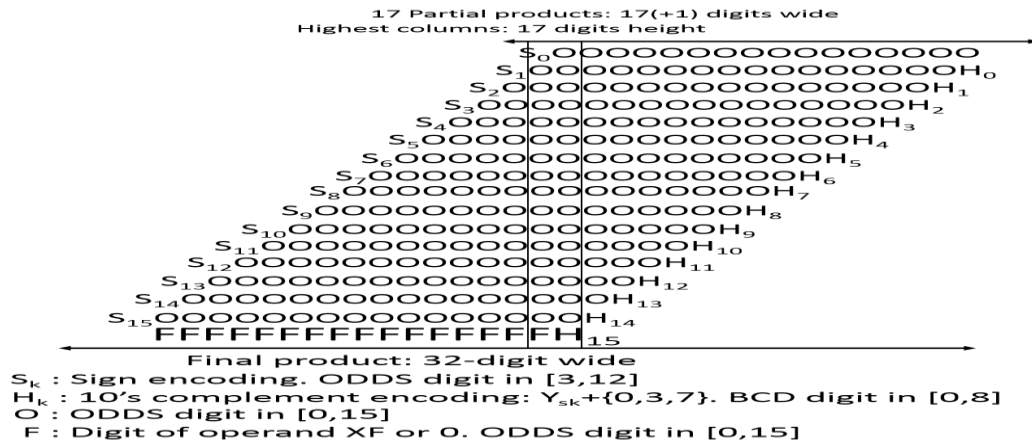
### Correction Term

The pre-computed correction term is given by

$$f_c(d) = -8 * \sum_{k=0}^{d-1} 10^{k+d} - 3 * \left( \sum_{i=0}^{d-1} (i+1)10^i + \sum_{i=0}^{d-2} (d-1-i)10^{1+d} \right)$$

## VI.PRODUCT ARRAY

The below figure illustrates the shape of the partial product array, particularizing for  $d = 16$ . Note that the maximum digit column height is  $d+1$ .

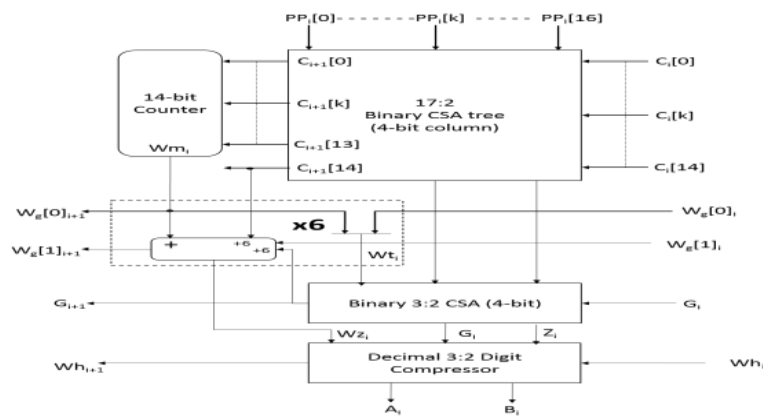


**Fig. : Decimal partial product array generated for d =16**

**VII. 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 (see Fig. 4), and  $h < d+1$ .

The below figure shows the high-level architecture of a column of the PPR tree (the  $i^{th}$  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 Fig. 5) reduces the h input digits and ncin input carry bits, transferred from the previous column of the binary CSA tree, to two digits,  $S_i, C_i$ , with weight  $10^i$ . Moreover, a group of ncout carry outputs are generated and transferred to the next digit column of the PPR tree. Roughly, the number of carries to the next column is  $ncout = h-2$ .



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

The digit columns of the binary CSA tree are implemented efficiently using 4-bit 3:2, 4:2 and higher order compressors made of full adders. These compressors take advantage of the delay difference of the inputs and of the sum and carry outputs of the full adders, allowing significant delay reductions. Thus, there is a difference between the value of the carry outs generated at the i-column and the value of the carries transferred to the (i+1)-column. This difference, T, is computed in the sum correction block of every digit column and added to the partial product sum (S, C) in the decimal CSA.

# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2015

$$W_i = \sum_{k=0}^{n_{count}-1} c_{i+1}[k] ,$$

the contribution of the column i to the sum correction term T is given by

$$W_i * 16 - W_i * 10 = W_i * 6 ,$$

Therefore, the sum correction is given by

$$T = \sum_{i=0}^{2d-1} (W_i * 6 * 10^i) = 6 * \sum_{i=0}^{2d-1} W_i * 10^i ,$$

Consequently, the sum correction block evaluates  $W_i * 6$ . This module is composed of a m-bit binary counter and a x6 operator. A straightforward implementation would use  $m = n_{cout}$  and a decomposition of the x6 operator into x5 and x1 (both without long carry propagations), and then a four to two decimal reduction to add the correction to the PPR tree result.

## VIII.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 implements

$$[A_i] + B_i + 0 \text{ and } [A] + B_i + 1 ,$$

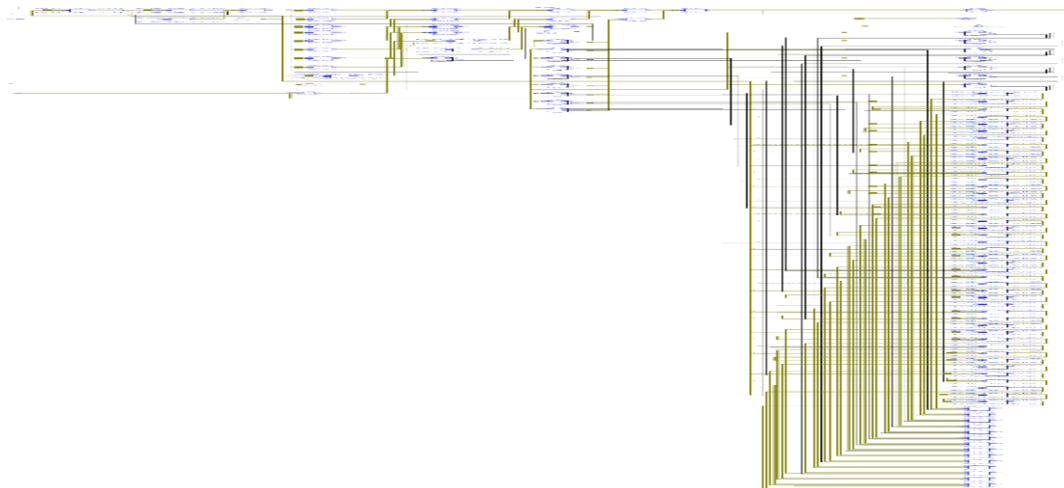
These conditional sums correspond to each one of the carry input values. If the conditional carry out from a digit is one, the digit adder performs a -6 subtraction. The selection of the appropriate conditional BCD digit sums is implemented with a final level of 2 : 1 multiplexers.

To design the carry prefix tree 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.

## IX.SYNTHESIS RESULTS

### SCHEMATIC DIAGRAM:

A **schematic**, or **schematic diagram**, is a representation of the elements of a system using abstract, graphic symbols rather than realistic pictures. A **schematic** usually omits all details that are not relevant to the information the **schematic** is intended to convey, and may add unrealistic elements that aid comprehension.





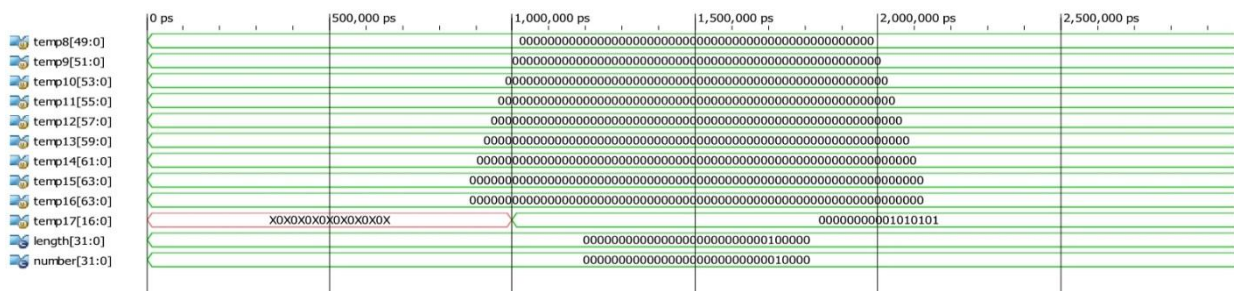
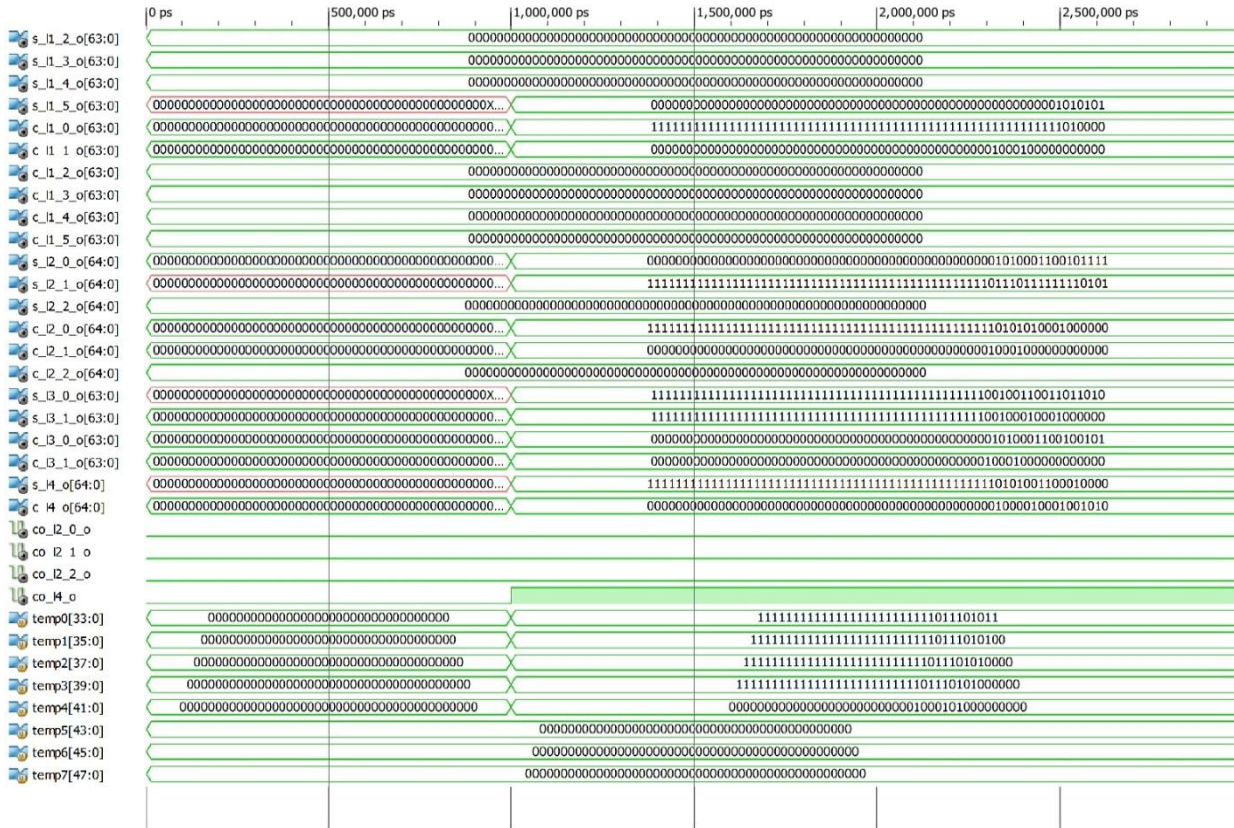




# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2015





# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2015

## IX.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 precomputed 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.

## REFERENCES

- [1] A. Aswal, M. G. Perumal, and G. N. S. Prasanna, —On basic financial decimal operations on binary machines,| IEEE Trans. Comput., vol. 61, no. 8, pp. 1084–1096, Aug. 2012.
- [2] M. F. Cowlshaw, E. M. Schwarz, R. M. Smith, and C. F. Webb, —A decimal floating-point specification,| in Proc. 15th IEEE Symp.Comput. Arithmetic, Jun. 2001, pp. 147–154.
- [3] M. F. Cowlshaw, —Decimal floating-point: Algorithm for computers,| in Proc. 16th IEEE Symp. Comput. Arithmetic, Jul. 2003, pp. 104–111.
- [4] S. Carlough and E. Schwarz, —Power6 decimal divide,| in Proc. 18th IEEE Symp. Appl.-Specific Syst., Arch., Process., Jul. 2007, pp. 128–133.
- [5] S. Carlough, S. Mueller, A. Collura, and M. Kroener, —The IBM zEnterprise-196 decimal floating point accelerator,| in Proc. 20th IEEE Symp. Comput. Arithmetic, Jul. 2011, pp. 139–146.
- [6] L. Dadda, —Multioperand parallel decimal adder: A mixed binary and BCD approach,| IEEE Trans. Comput., vol. 56, no. 10, pp. 1320–1328, Oct. 2007.
- [7] L. Dadda and A. Nannarelli, —A variant of a Radix-10 combinational multiplier,| in Proc. IEEE Int. Symp. Circuits Syst., May 2008, pp. 3370–3373.
- [8] L. Eisen, J. W. Ward, H.-W. Tast, N. Mading, J. Leenstra, S. M. Mueller, C. Jacobi, J. Preiss, E. M. Schwarz, and S. R. Carlough, —IBM POWER6 accelerators: VMX and DFU,| IBM J. Res. Dev., vol. 51, no. 6, pp. 663–684, Nov. 2007.
- [9] M. A. Erle and M. J. Schulte, —Decimal multiplication via carrysave addition,| in Proc. IEEE Int. Conf Appl.- Specific Syst., Arch., Process., Jun. 2003, pp. 348–358.
- [10] M. A. Erle, E. M. Schwarz, and M. J. Schulte, —Decimal multiplication with efficient partial product generation,| in Proc. 17th IEEE Symp.Comput. Arithmetic, Jun. 2005, pp. 21–28.
- [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.

## BIOGRAPHY



M.VIJAYALAKSHMI received the degree in Electronics & Communication Engineering from GPREC, Kurnool in 2000, Masters in Electronics & Communication Engineering from JNTU, Hyderabad in 2009. Presently, she is working as Associate Professor in ECE department in St. Martin's Engineering College, Secunderabad.



CH.VIJAYALAKSHMI received the Electronics Engineering P.G degree from JNTU, KAKINADA in 2012. Since 2008, She has been working as assistant professor in the Department of Electronics And Communication Engineering for TECA, Anumaralapudi, Guntur District, Andhrapradesh, India. She is currently with St. Martin's engineering college, Dulapally, Secunderabad, India. She received P.G.DIPLOMAIN FINANCIAL MANAGEMENT from IGNOU, NEWDELHI in 2014.



ISSN (Print) : 2320 – 3765  
ISSN (Online): 2278 – 8875

## International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 11, November 2015**



G.VINATHA completed Graduation from Kshatriya College of Engineering, JNTUH and she achieved her Master's degree from the same college. She worked as Assistant Professor in different Colleges and presently working as Associate Professor in St. Martin's Engineering College.



RAVI CHANDER B completed his M.Tech in 2013 from scient institute of technology affiliated to JNTUH and he is working as an assistant professor in ECE department from 2 years .