IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
**ISSN: 2320 - 8791**
**www.ijreat.org**

# Design of High Speed Area Optimized Binary Coded Decimal Digit Adder and Multiplier

**[1]Merinmol Mathew [2] Mrs. Georgina Binoy Joseph**

**[1] Electronics and Communication, K.C.G College of Technology, Chennai, Tamil Nadu 600097,India**

**[2] Electronics and Communication, K.C.G College of Technology, Chennai, Tamil Nadu 600097,India**

**Abstract**

Decimal arithmetic is very important for computations in financial and commercial transactions which include banking system, tax calculation, billing etc. Decimal arithmetic algorithm mainly includes decimal addition and multiplication. The main problem in the existing decimal arithmetic is the need for correcting the result as it is in the binary form. This causes the implementation area to be large and increases the delay. In this paper, a high speed area optimized binary coded decimal digit adder and multiplier are designed. The proposed adder and multiplier have improved delay and area due to the employed correction free mechanism and the result thus obtained is in BCD form without adding any correction values. This paper also presents the multilevel optimization of the Boolean expressions. It minimizes the layout area, delay and increases the performance.

Keywords: Decimal arithmetic, BCD, correction free mechanism, multilevel logic optimization

## 1. Introduction

As we are in the era of electronic computing, decimal arithmetic plays a vital role in commercial, financial, internet and industrial control applications. Most of the computing applications are based on binary arithmetic, but the real problem is that binary approximation does not produce accurate result. For example if a telecommunication company approximates a 5% sale tax on an item (such as $0.70 for a telephone call) the yearly loss will be more than $5 million. Binary decimal arithmetic is required to avoid such incorrect approximations. Also, in most of the applications, decimal software runs on custom binary hardware in order to produce precise decimal results, leading to another problem of excessive delays. Software implementation of decimal arithmetic is about 100 to 1000 times slower than the binary implementation in hardware. In a survey of IBM corporation showed that almost 55% of the numeric data columns, used by 51 major organization's databases, were decimal data types and 43.7% were integer types which can be stored as decimals. In order to meet the need for growing evolution of decimal arithmetic, it's necessary to develop efficient algorithms. Decimal digit adders and decimal digit multipliers are the building blocks of any decimal hardware to support decimal arithmetic.

Here we are proposing a high speed and area optimized decimal digit adder and decimal digit multiplier. These designs are described and simulated using verilog hardware description language.

## 2. Related works

In conventional decimal digit adder [7], the output sum is analyzed based on the input carry value. Depending on the input carry if the sum is greater than 9, then it is represented by using digit generate and digit propagate signals. For a 4-bit adder, digit propagate signal is obtained by ANDing Sum [3] and Sum [0]. Digit generate signal is obtained by ANDing the values of Sum [3] and Sum [2]. Then the results are ORed with the ANDed values of Sum [3] and Sum [1] and carry output value.

The conditional speculative adder [3], simplifies the sum correction of the speculative methods. Full binary parallel prefix carry tree configuration is used to improve delay eliminating post-correction from critical path. Quaternary carry tree configurations used in this method Improve area simplifying correction.

I.S. Hwang [6], introduced binary and decimal adder unit. This adder is designed to support both binary and decimal additions. A binary carry look-ahead adder (CLA) is used to add two input operands, which are either binary or decimal numbers. The result of the binary CLA is the correct result for binary inputs, but it

needs to be corrected for decimal inputs. Therefore, in the decimal case, the binary CLA addition result and some of the carries (C[4],C[8],C[12],...)are used to compute the corrected result.

Reduced Delay BCD Adder [1] introduced by Alp Arslan Bayrakci and Ahmet Akkas in order to improve the delay. In this design they used a parallel prefix network is used to reduce the delay of decimal addition. It improves the delay when compared to Hwang's Adder. But the carry network again introduces delay.

Mark A. Erle and Michael J. Schulte [8] introduced Decimal Multiplication via Carry-Save Addition. In that Decimal multiplication performs the computation P = A·B where A is the multiplicand, B is the multiplier, and P is the product. It is assumed that A and B are each n digits and P is maximally 2n digits The most notable shortcomings of this approach are the significant area or delay to generate the eight multiples, and the eight additional registers needed to store the multiples. An alternative to storing all the multiples (called primary multiples or a primary set) is storing a reduced set of multiples.

Another design introduced by G. Jaberipur and A. Kaivani [5], used a standard 4×4 unsigned binary multiplier generating an 8-bit binary output, which should be corrected to two BCD digits, with the same arithmetic value. Given that the product value belongs to [0, 81], its most significant bit (weighted $2^7$) is always zero. Let X =$x_3x_2x_1x_0$ and Y = $y_3y_2$ $y_1$ $y_0$ represent the two input BCD digits and p6p5p4 p3 p2 p1p0 is the output (i.e. product) of the standard 4 ×4 multiplier, with p7 = 0 ignored. But in decimal multiplication, because of particularities of using radix 10, which is not a power of 2, one needs to generate BCD partial products to be followed by BCD multi-operand addition. Therefore we need localized reduction trees in order to convert binary to BCD value.

A. Vazquez, E. Antelo and P. Montuschi [2] introduced "A new family of high performance parallel decimal multipliers", which gives an efficient implementation of decimal parallel multiplication by a parallel generation of partial products and the reduction of these partial products using a novel decimal carry–save addition tree.

The decimal addition and multiplication used in the above designs need to go for a correction stage in order to obtain the BCD result. For that an analyzer network is used which introduces delay. So in this project a correction free mechanism is used based upon the Direct Boolean Expression Method. Besides these the partial product reduction in the multiplier can be avoided by using the proposed method and thereby increases the speed.

## 3. Correction free BCD digit adder

Here an optimized correction free BCD digit adder is proposed. The 2 decimal input digits of the BCD adder are A $\epsilon$ {0, 9} and B $\epsilon$ {0, 9} and the decimal carry input is $C_{in}$. We can represent the decimal sum and the decimal carry as as S $\epsilon$ {0, 9} and Cout respectively. The decimal value of A, B, and S can be used to obtain their 8421 BCD representation. In general, we can write A =$a_3a_2a_1a_0$, B = $b_3b_2b_1b_0$, and S =$s_3s_2s_1s_0$, where $a_i$, $b_i$, and $s_i$ € {0, 1} ∀ i € {0, 1, 2, 3}. A and B can be expressed in terms of two integers m = $a_3a_2a_1$ and n = $b_3b_2b_1$ as: A = $2 \times m + a_0$ and B = $2 \times n + b_0$, where $0 \leq m \leq 4$ and $0 \leq n \leq 4$.This implies that the output of the BCD adder can be expressed as

$$\{C_{OUT}, SUM\} = A+B+C_{IN}$$
$$= (2\times m+a_0) + (2\times n+b_0) + C_{in}$$

We can rearrange the above expression for BCD adder output as:

$$\{C_{OUT}, SUM\} = A+B+C_{IN}$$
$$= (2\times m+a_0) + (2\times n+b_0) + C_{in}$$
$$= (2\times n+m) + (a_0+b_0+C_{in})$$

Using the above formula, we have designed the BCD digit adder that consists of two stages: Stage1 and Stage2. Fig: 1 shows the proposed BCD adder. The inputs to Stage1 are m and n. Stage1 generates the partial decimal sum: Z = z3z2z1z00 = 2 × (n + m). It should be observed that this decimal partial sum consists of an even decimal digit (z2z1z0 0) and a decimal carry z3 that can be either 1 or 0 based on the values of m and n.

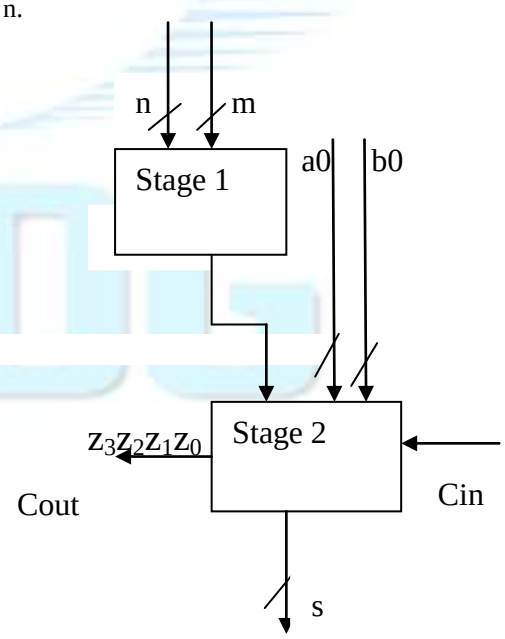


Fig:1 Proposed BCD adder

For example, if the two input decimal digits A and B are 6 = $(0110)_{BCD}$ and 4 = $(0100)_{BCD}$, respectively, then we

have: m = $(011)_2$ = 3 , n = $(010)_2$ = 2 and Z= z3z2z1z0 0 = 2 × (n + m) = 2 × (3 + 2) = $(12)_{10}$.This means that the decimal carry z3 is 1 and the even decimal digit z2 z1 z0 0 = $(2)_{10}$ = $(0010)_{BCD}$. Since the produced decimal digit is always even, only z3z2z1z0 are forwarded to Stage2 o the BCD digit adder. For finding the Boolean expression, first we have to make a truth table for different combinations of the input starting from(0000) to (1001).

Table 1: Examples from the truth table to demonstrate how the first stage output is calculated

| a0a1a2a3 | b0b1b2b2 | m | n | 2 (m +n) | z3z2z1z0 |
|---|---|---|---|---|---|
| 0000 | 0000 | 0 | 0 | 0 | 0000 |
| 0000 | 0001 | 0 | 0 | 0 | 0000 |
| 0000 | 0010 | 0 | 1 | 2 | 0001 |
| 0000 | 0011 | 0 | 1 | 2 | 0001 |
| 0000 | 0100 | 0 | 2 | 4 | 0010 |

After getting the truth table we have to derive the boolean expressions. Here we use Quine Mccluskey method for truth table simplification because it gives more systematic method of minimizing expressions of larger number of variables.

Based on this approach, the values of z3, z2, z1, and z0 are computed for all possible combinations of A and B and the following optimized boolean equations are derived for Stage1:

z3=a2a1b2+a2a1'b3+a2b2b1+a3b2b1'+a1b3+a3b1+a3b3;

z2=a3'a2'a1'b3+a3b3'b2'b1'+a2a1'b2b1'+a2a1b2'b1+a2'a1b2b1;

z1=a3'a2'a1'b2b1+a2b3'b2'b1'+a3'a2'b2b1'+a2a1'b2'b1+a2'a1b2'b1+a2a1b3+ a3b2b1+a3b3;

z0=a3'a2'a1'b2b1+a1b3'b2'b1'+a3'a1'b2'b1+a2'a1b2b1'+a2a1b2b1+a2 a1'b3+a3b2b1'+a3b3;

The outputs of Stage1 along with a0, b0, and Cin are given as input to Stage2. In order to design Stage2, the values of Cout, s3, s2, s1, and s0 have been calculated for all possible combinations of z3, z2, z1, z0, a0, b0, and Cin and optimized boolean equations for Stage2 have been derived. To illustrate this procedure, the various possible combinations are presented in Table II. For example, if the values of z3, z2, z1, and z0 are 1, 0, 1, and 1, respectively, then this means that the value of 2 × (m + n) that is to be added to the value of (a0 + b0 + Cin) is $(16)_{10}$,since a0, b0, and Cin in the example are 1, 1, and 1 respectively. The value of A+B = 2 × (n +m) + (a0 + b0 + Cin) is $(19)_{10}$. Therefore, the computed values for Cout, s3, s2, s1, and s0 are 1, 1, 0, 0, and 1, respectively. Based on this, Stage2 generates the decimal sum output and the decimal carry output.It should be emphasized that our proposed BCD digit adder does not require any corrections to the results and the results are computed with only two stages.

Table 2: Examples from the truth table to demonstrate how the final stage output is calculated

| z3z2z1z0 | z | a0 | b0 | Cin | cout | s3s2s1s0 |
|---|---|---|---|---|---|---|
| 0000 | 0 | 0 | 0 | 0 | 0 | 0000 |
| 0000 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 0001 | 2 | 0 | 1 | 0 | 0 | 0001 |
| 0001 | 2 | 0 | 1 | 1 | 0 | 0010 |

The truth table obtained above has to be simplified in the similar way as we did for computing the Boolean expression for the first stage output. And thus we derive the Boolean expression for the corresponding sum and output carry.

Cout=z2 cin b0 +z2 cin a0+z2 a0 b0+z3;

s3=z2 cin' b0'+z2 cin'a0'+z1 z0 cin b0+z1 z0 cin a0+z1 z0 a0 b0+z2 a0' b0';

s2=z1cin'b0'+z1cin'a0'+z1a0'b0'+z1'z0cinb0+z1'z0cina0+z1'z0a0b0+z0'z1;

s1=z2' z0'cin b0+z2' z0'cin a0 z2'z0'a0 b0+z0 a0' b0'+z0 cin'b0'+z0 cin'a0';

s0=cin a0'b0'+cin'a0 b0'+cin'a0'b0+cin a0 b0;

## 4. Direct Boolean Expression BCD Digit Multiplier

In the proposed direct Boolean expression BCD digit multiplier, the direct functionality is performed by using the simplified Boolean expression. In this case, the two operands are two decimal digits A = a3a2a1a0 and B = b3b2b1b0 and the output P = A × B is 8 bit p7p6p5p4p3p2p1p0 (two BCD digits). The input given is 8 bits wide; the number of possible combinations in the truth table is $2^8$ = 256. Out of these 256 combinations only 100 combinations are valid and the rest are invalid.

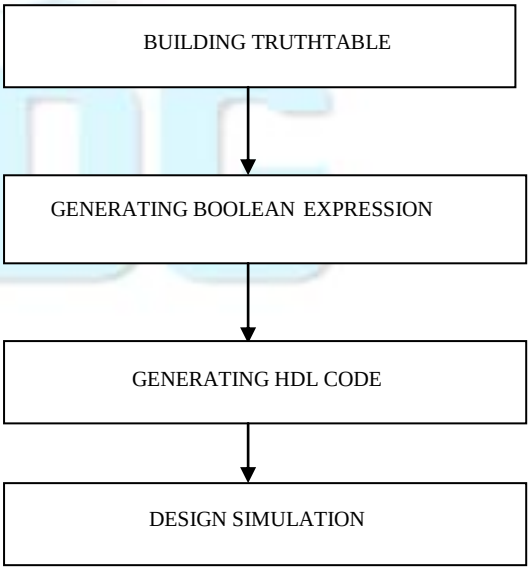


Fig.2 Design flow

Table 3: Examples from the Truth table of BCD multiplier

| Inputs | | Outputs | |
|---|---|---|---|
| a3a2a1a0 | b3b2b1b0 | p7p6p5p4 | p3p2p1p0 |
| 0000 | 0001 | 0000 | 0000 |
| 0010 | 0110 | 0001 | 0010 |
| 0100 | 1001 | 0011 | 0110 |
| 1010 | 1001 | xxxx | Xxxx |

The truth table obtained above is then used for finding the Boolean expression. Here we have done two level logic optimization. And we will get the following Boolean Expressions:

p0 = a0 b0;
p1 = a2 a1 a0 b3 + a2 a1' a0' b3 + a3 a0' b3 b0 + a3 b2 b1 b0 + a3 a0 b3 b0' + a3 b2 b1'b0' + a3' a2' a1' a0 b1 + a2 a1 a0' b2 b1 + a1 b3' b2' b1' b0 + a2' a1 a0' b3 b0' + a2 a1 a0' b1 b0' + a2 a1 b2 b1 b0' + a1 a0' b2 b1 b0' + a3 a0' b2' b1 b0' + a2 a1' a0' b2' b1 b0 + a2' a1 a0 b2 b1' b0' + a2 a1' a0' b2 b1' b0' + a2' a1' a0 b1 b0 + a1 a0 b2' b1' b0 + a2' a1 a0' b2' b1 b0 + a2' a1 a0 b2' b1 b0';

p2 = a2' a1 b2' b1 b0' + a2 b3' b2' b1' b0 + a3' a1' a0 b2 b0 + a2' a1' a0 b2 b0' + a2 a0' b2' b1' b0 + a3 a0' b3 b0' + a2' a1 b3 b0' + a3 a0' b2' b1 + a1 a0 b3 b0' + a3 a0' b1 b0 + a2 a0' b2 b0' + a2' a1 a0 b3 + a3 b2' b1 b0 + a2 a1' a0 b0 + a0 b2 b1' b0 + a1 a0 b2' b1 b0' + a2' a1 a0' b1 b0;

p3 = a3' a2' a1' a0 b3 + a2' a1 a0' b3 b0 + a3 b3' b2' b1' b0 + a2 a1 a0' b3 b0' + a3 a0' b2 b1 b0' + a3 a0 b2' b1 b0' + a2 a1 a0 b2 b1 b0 + a2 a1' a0' b2 b1 b0 + a2' a1 a0 b2' b1 b0 + a2 a1 a0' b2' b1 b0 + a2' a1 a0 b2 b1 b0' + a2 a1' a0' b2' b1 b0' + a2 a1 a0 b2 b1' b0' + a2' a1 a0' b2 b1' b0';

p4 = a2' a1 a0' b3 + a2 a1' a0' b3 + a2 a1' a0 b1 + a3 a0' b3 b0 + a1 b2 b1' b0 + a3 a0 b3 b0' + a3 b2' b1 b0' + a3 b2 b1' b0' + a3 a0' b2 b1 b0 + a2 a1 a0 b3 b0' + a2 a1' a0' b2 b1' b0' + a1 a0' b3 b0 + a3 a0 b1 b0' + a2' a1 a0' b2 b1 + a2 a0' b2' b1 b0 + a2' a1 a0 b2 b0' + a1 a0' b2 b1 b0' + a2 a1 b2' b1 b0';

p5 = a3 a0' b3 + a3 b3 b0' + a2' a1 a0 b3 + a2 a1' a0' b3 + a1 a0 b3 b0 + a3 a0 b1 b0 + a3 b2' b1 b0 + a3 b2 b1' b0' + a2' a1 a0 b2 b1 b0 + a2 a1 a0 b2' b1 b0 + a2 a1' b2 b1 + a2 a0 b2 b1' + a2 b2 b1' b0 + a2 a1 a0' b2 b0';

p6= a2 a1 b3 + a2 a0 b3 + a3 a0' b3 + a3 b2 b1 + a3 b2 b0 + a3 b3 b0' + a2 a1 a0 b2 b1 + a2 a1 b2 b1 b0;

p7 = a3 a0 b3 b0;

The two level optimization we done here will increase the area and delay. Therefore in order to reduce that we have to go for multilevel optimization. The advantages of the multilevel logic optimization includes minimizing overall layout area and critical path delay time, maximizing the testability of the synthesized logic and providing a complete set of test vectors as a by-product of the optimization.

## 5. Multilevel Logic Optimization

Multilevel logic can be described by a set $\chi$ of completely specified Boolean functions. Each Boolean function $f \in \chi$ , maps one or more input and intermediate signals to an output or a new intermediate signal. If a circuit is represented as a Boolean network, each node has a Boolean variable and a Boolean expression associated with it. There is a directed edge to a node g from a node f if the expression associated with node g contains in it the variable associated with *f* in either true or complemented form. A circuit is also viewed as a set of gates. Each gate has one or more input pins and one output pin.

Here we are applying this logic to the proposed correction free decimal digit adder* and decimal digit multiplier* thereby minimizing the boolean expression. For that we need to factorize each Boolean expression and have to substitute values for the common terms present in all the expressions. For example, in the case of adder the Boolean expression for z3 can be written as:

z3=m6g1+m4a1'+m5b1'+m8+m9+m3;
Where m6=a2b2, g1=a1+b1;
m4=a2b3, m5=a3b2;
m8=a1b3, m9=a3+b1;
m3=a3b3

Similarly all other expressions can be written by assigning values to the common terms. The same procedures have to be followed for decimal digit multiplier.

## 6. Results and Comparisons

From the table given below it is clear that the synthesis results of the correction free adder and multiplier after multilevel logic optimization has comparatively smaller area.

Table 4: Area for decimal digit adder and decimal digit multiplier

| **Decimal digit adder** | **Area in number of LUT's** |
|---|---|
| Proposed decimal digit adder(3) | 11 |
| Multilevel logic optimized adder* | 10 |
| Conditional speculative adder [3] | 19 |
| **Decimal digit multiplier** | **Area in number of LUT's** |
| Proposed decimal digit multiplier (4) | 29 |
| Multilevel logic optimized multiplier* | 28 |
| BCD digit multiplier [2] | 32 |

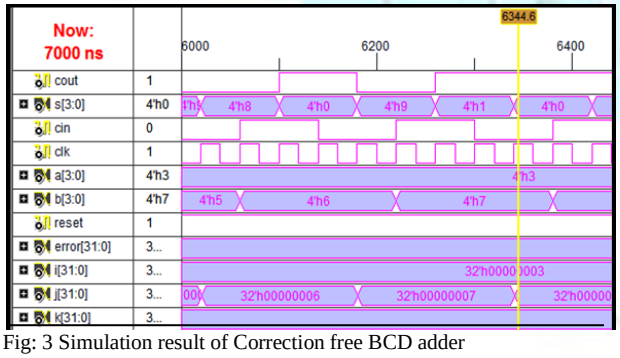


Fig: 3 Simulation result of Correction free BCD adder

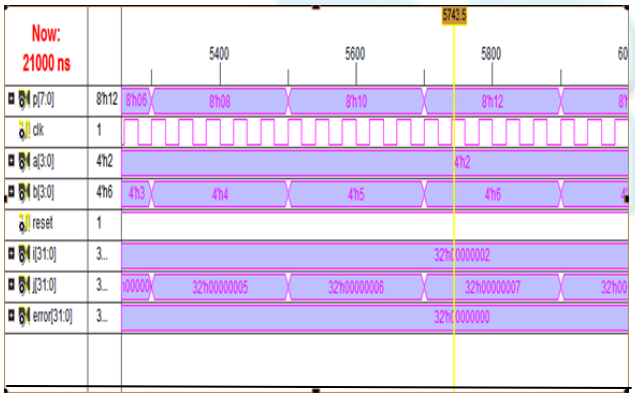


Fig: 4 Simulation result of Correction free BCD multiplier

## 7. Conclusion

In this paper, direct Boolean expression binary coded decimal digit adder and multiplier will produce the output in the BCD form. As a result a correction free BCD digit adder and multiplier is obtained when compared with the existing system which needs an analyzer circuit for determining the whether the output value is greater than 9.Here the Boolean expression is obtained using two-level logic optimization is modified to multilevel logic optimization for reducing the area and delay. The design is synthesized, verified and tested for correct functionality using verilog coding and simulation.

## References

[1] Alp Arslan Bayrakci and Ahmet Akkas "Reduced Delay BCD Adder" *IEEE International Conf. on Application-specific Systems, Architectures and Processors (ASAP 2007).* Volume-Issue: 9-11, Page(s):266-271 July 2007

[2] A. Vazquez, E. Antelo, and P. Montuschi, "A new family of high performance parallel decimal multipliers," in *IEEE Symposium on Computer Arithmetic*, Washington, DC,ARITH '07, pp. 195–204, IEEE Computer Society,2007

[3] A. V´azquez and E. Antelo, "Conditional speculative decimal addition, "Nancy, France, 2006, pp. 47–57.

[4] E. M. Schwarz, "Decimal multiplication with efficient partial product generation," in *IEEE Symposium on Computer Arithmetic*, Washington, DC, USA, ARITH '05, pp. 21–28, IEEE Computer Society,2005

[5 ]G. Jaberipur and A. Kaivani, "Binary-coded decimal digit multipliers,"*IET Computers and Digital Techniques*, vol. 1, no. 4, pp. 377–381, 2007

[6] I.S. Hwang, "High Speed Binary and Decimal Arithmetic Unit", United States Patent 2007.

[7] M. M. Mano. Digital Design, pages 129–131, Prentice Hall, third edition, 2002.

[8] M. A. Erle and M. J. Schulte"Decimal multiplication via carry-save Addition*". IEEE Int'l Conference on Application Specific system Architectures and Processors*, pages 348–358, June 2003.

[9] Osama Al-Khaleel, Mohammad Al-Khaleel, Zakaria Al-Qudah "Fast Binary/Decimal Adder/Subtractor with a Novel Correction-Free BCD Addition" *in IEEE Conference on Computer arithmetic*, 2011