American Journal of Electrical and Electronic Engineering, 2015, Vol. 3, No. 1, 13-16
Available online at http://pubs.sciepub.com/ajeee/3/1/3

DOI:10.12691/ajeee-3-1-3

An Approach for Fast BCD Addition

Parag K. Lala*

Department of Electrical Engineering, Texas A&M University-Texarkana, Texarkana, USA
*Corresponding author: plala@tamut.edu

Received December 18, 2014; Revised March 03, 2015; Accepted March 12, 2015

Abstract This paper presents a technique for fast addition of multi-digit BCD numbers. The addition of all columns can be performed simultaneously, and the carry values are utilized only in the final stage of the addition. Thus the traditional carry propagation process is drastically reduced, hence speeding up the addition process. The addition technique is used in the summation of partial products generated during a new multiplication approach proposed in the paper resulting in a faster multiplication.

Keywords: *2's complement binary number, overflow, multi-digit BCD addition, partial product*

Cite This Article: Parag K. Lala, "An Approach for Fast BCD Addition." *American Journal of Electrical and Electronic Engineering*, vol. 3, no. 1 (2015): 13-16. doi: 10.12691/ajeee-3-1-3.

1. Introduction

Binary coded decimal (BCD) can represent decimal numbers 0 to 9 in 4-bit binary equivalents. Figure 1 shows *unpacked BCD* representations of decimal digits. Unpacked BCD representation allows only one decimal digit per byte. This means that in an unpacked representation of any BCD number the upper four bits are zero. It is possible to pack two decimal digits into a single byte; this is known as *packed BCD* representation. The value of each 4-bit can range from 0 to 9 although a 4-bit data allows value from 0 to 15. The addition of unpacked or packed BCD numbers follows the general rule of binary addition. If during the addition process an invalid BCD number is produced or a carry is generated then an adjustment is necessary to correct the sum. This is traditionally done by adding 6 to the sum [1].

Decimal Number	BCD
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
8	00001000
9	00001001

Figure 1. Unpacked BCD representation

The use of decimal arithmetic in commercial and financial data processing has been on the rise in recent years [2]. This is in spite of the simplicity and efficiency of binary number based computing systems. In such a system, the decimal data has to be converted to binary first, and after the data processing task has been completed the resulting binary data is reconverted to decimal format. However, the conversion between decimal and binary formats introduces significant delay to the computation task [3]. Furthermore, binary representation of certain decimal fractions require unacceptably high number of bits for accurate representation; an approximate representation of fractions compromises the accuracy of the final result [4]. The BCD representation of decimal fractions avoids errors in representing and calculating such values. The conversion of a BCD number for display is a simple digit by digit mapping that can be done in linear time. Furthermore in the BCD representation of a number, adding a new digit to the number requires just appending a 4-bit data to the BCD number. The disadvantages of BCD compared to typical binary representations are a small increase in the complexity of the circuits needed to implement basic arithmetic operations and less efficient usage of storage.

Several techniques have been proposed in recent years for faster BCD adder design [5,6,7]. This paper presents a new way to design fast BCD adder. Initially all BCD digits are assigned binary values such that half of them receive positive values and the other half get negative values; these values are not the same as the binary patterns used to represent each decimal digit in BCD. The sum of two BCD digits obtained are adjusted (if necessary) based on the overflow bit (if one is generated) and the sign bit of the sum. The next BCD digit in the list of addends is then added to this partial sum; this process is continued till all the BCD digits in list are used. and the composite sum and the associated carry bits are stored.

If multiple columns of BCD digits are to be added, the digits in each individual column are added simultaneously and the resulting sum and carry patterns are added in such a way that the final result as well as intermediate addition results are automatically in BCD format, no additional adjustments are necessary.

A system based on BCD representations of decimal fractions avoids errors representing and calculating such values. The conversion of a BCD number for display is a

simple digit by digit mapping that can be done in linear time. Furthermore in the BCD representation of numbers adding a new 4-bit data to the BCD representation of the original number. The disadvantages of BCD compared to typical binary representations are a small increase in the complexity of the circuits needed to implement basic arithmetic operations and less efficient usage of storage.

This paper presents a new way to design fast BCD adder. Initially all BCD digits are assigned binary values such that half of them receive positive values and the other half get negative values; these values are not the same as the binary patterns used to represent each decimal digit in BCD. The sum of two BCD digits obtained are adjusted (if necessary) based on the overflow bit (if one is generated) and the sign bit of the sum. The next BCD digit in the list of addends is then added to this partial sum; this process is continued till all the BCD digits in list are used. and the composite sum and the associated carry bits are stored. If multiple columns of BCD digits are to be added, the digits in each individual column are added simultaneously and the resulting sum and carry patterns are added in such a way that the final result as well as intermediate addition results are automatically in BCD format, no additional adjustments are necessary.

2. New Technique for BCD Addition

The proposed approach is different from the traditional way of sum generation; each number in a column of decimal digits is replaced first as indicated in Figure 2. Notice that the numbers 1 to 5 are represented by their binary values. The binary representations of the rest of the numbers i.e. 6, 7, 8 and 9 are derived by subtracting 10 from each, and the resulting negative numbers -4, -3, -2 and -1 respectively are replaced by their 2's complement binary representations.

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	1100
7	1101
8	1110
9	1111

Figure 2. Conversion of decimal numbers to signed binary

Before formally presenting the proposed the BCD summation technique let us illustrate the carry and the sum generation by performing the following addition:

6 + 3 + 4 + 7.

The decimal values are first replaced by their binary equivalents using Figure 2.

6	1100
3	0011
4	0100
7	1101

The sum of 6 and 3 is 1111 which is 9 as indicated in Figure 2. Next 4 (0100) is added to 1111 resulting in 1,0011 i.e. 13. Finally, 7 (1101) is added to this partial sum resulting 10,0000 which is 20 in BCD. In general while adding BCD digits based on this approach we need to consider both the carry bit and possible overflow resulting during each stage of the addition process. An overflow occurs if the carry-in to the sign (most significant) bit is different from the carry-out of the sign bit. In the addition example illustrated above no overflow flow was generated at any stage. In most instances, the partial sum generated need to be adjusted based on the carry out bit and/or the overflow bit generated during the partial sum generation process. We need to consider four possible cases:

Case i. Overflow and sign bit of sum is 0 i.e. sum is positive

Case ii. Overflow and sign bit of sum is 1 i.e. sum is negative

Case iii. No overflow and sign bit of sum is 0

Case iv. No overflow and sign bit of sum is 1

Case i can arise only if the first two bits of an addend are 10 and the first two bits of the other addend are also 10 or 11. Since none of the numbers in Figure 1 have 10 as the first bits this situation may arise only during the summation of two numbers at an intermediate stage.

Case ii is possible only when the first two bits of both addends are 01 (e.g. when 4 and 5 are added together, or 4 added to 4 or 5 added to 5).

Case iii is possible only when an addend with first two bits 01 are added to an addend with first two bits 11 (e.g. when 4 and 6 are added together).

Case iv arises only if the first two bits of one addend are 11 and those of the other addend are also 11 or 00.

Algorithm for BCD addition

Step 1. Replace each decimal digit in a column by its binary equivalent using Figure 2.

Step 2. Add the first two binary numbers in a column to generate a partial sum; as indicated previously this may need to be adjusted based on the carry out bit and/or the overflow bit generated during the partial sum generation process.

Choose one of the following options based on the status of the overflow and the sign bit:

Case i. Overflow =Yes, Sum = Positive Add 1010 to the sum and ignore carry resulted from this addition i.e. no changes in the original carry bits are necessary.

Case ii. Overflow = Yes, Sum = Negative

Add 0110 to the result to adjust the sum..

Case iii. Overflow = No, Sum = Positive

No adjustment to the partial sum is needed.

Case iv. Overflow = No, Sum = Negative

Add 1010 to adjust the sum and ignore the carry bit i.e. there no changes to the carry bits that existed before the adjustment.

Step 3. Add the carry bits of each column to the sum bits of its left column. This operation can be carried out simultaneously for all columns.

Step 4. The resulting carry and sum bits are added as in step iv.

Step 5. Repeat step 4 till no carry bits are generated during a summation. The resulting sum is the final result of the addition

To illustrate the algorithm let us perform the addition of the following BCD numbers in a single column

7
8
5
7
3

Let us first add 7 and 8; the corresponding binary numbers as shown in Table 1 are

```
                1101
                1110
               -------
              1,1011        (case iv)
Add 5 to the sum 0101
               --------
             10,0000        (case iii)
Add 7 to the sum 1101
               --------
             10,1101
Add 3 to the sum 0011
               -------
             11,0000        (case iii)
```

Thus the actual sum is 0011 0000 i.e. decimal 30.

As another example let us add the following BCD numbers

929
838
619
788
159
278

The replacement of the digits in each column is as indicated in Figure 1. The individual sum of

```
9  1111   2  0010   9  1111
8  1110   3  0011   8  1110
6  1100   1  0001   9  1111
7  1101   8  1110   8  1110
1  0001   5  0101   9  1111
2  0010   7  1101   8  1110
```

0011 0011 0010 1100 0101 0001

The final sum of the addition is derived as follows:

First col. Second col. Third col

0011 0011 0010 1100 0101 0001

0011 0101 0001 0001

0110

Replacing each 4-bit binary number(shown in bold) by its equivalent decimal value results in sum of 3611 in BCD.

3. An Approach for Fast Multiplication

In this section we present a technique for multiplication. It utilizes the addition technique introduced earlier for the summation of partial products that are generated during the multiplication process, thereby significantly speeding up the multiplication process.

Let us illustrate the multiplication process by multiplying $X = x_{n-1}\ x_{n-2}\ \ldots\ x_0$ by $M = m_{n-1}\ m_{n-2}\ \ldots\ m_0$. The multiplication of X by m_{n-1}, m_{n-2} etc. can be carried out simultaneously. The intermediate product resulting from multiplying X by m_{n-1} is recorded as follows:

$x_{n-1} \times m_{n-1}$ $x_{n-2} \times m_{n-1}$ $x_{n-3} \times m_{n-1}$ ..... $x_0 \times m_{n-1}$

$= r_{n-1}\ s_{n-1}$ $r_{n-2}\ s_{n-2}$ $r_{n-3}\ s_{n-3}$ ..... $r_0 s_0$

digit of the two-digit product term, and $0 \le r_i \le 9$ and $0 \le s_i \le 9$.

Next the first digit of each term is added to the second digit of its previous term as shown below:

$r_{n-1}\ s_{n-1}$ $r_{n-2}\ s_{n-2}$ $r_{n-3}\ s_{n-3}$ $r_{n-4}\ s_{n-4}$ .... $r_1 s_1$ $r_0 s_0$

This process of adding s_{i-1} to r_{i-2} to is continued till each sum is a single digit. The partial product terms are then rearranged as follows and added using the addition technique proposed in section 2.

As an example let us perform 899 × 678. The result of multiplying 899 by 6 is as shown below As an example let us perform 899 × 678.

The result of multiplying 899 by 6 is as shown below

48 54 54

The next step is to add the first digit of each term to the second digit of its previous term as discussed above:

48 54 54

4 13 9 4

5 3 9 4

Similarly the results of multiplying 899 by 7 is

56 63 63

5 12 9 3

6 2 9 3

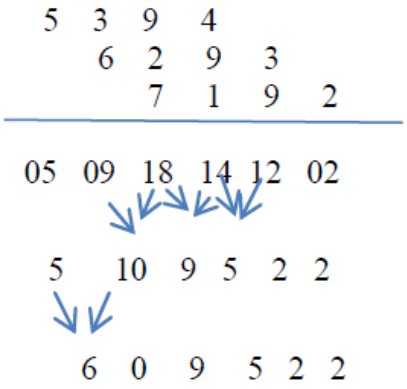
and by 8 is

64 72 72

6 11 9 2

7 1 9 2

These partial products can be derived in parallel, and are then added as shown below



4. Conclusion

This paper presents an approach for fast addition of multi-digit BCD numbers. It assigns both positive and negative binary numbers to the BCD digits. The sum of each column of BCD digits are generated in *parallel;* the sum and the associated carry bits are then added together using minimum number of additional steps to obtain the final sum. The addition of all columns can be performed simultaneously, and the carry values are utilized only in the final stage of the addition. Thus the traditional carry propagation process is drastically reduced, thereby significantly improving the speed of adder operation. It is also shown that the BCD addition approach presented here can be efficiently utilized to add the partial product terms generated during the multiplication process of two multi-digit numbers.

References

[1] P. K. Lala, *Principles of Modern Digital Design*, John Wiley & Sons, 2007.

[2] M.F. Cowlishaw, "Decimal floating-point: algorism for computers",. Proc. 16th IEEE Symposium on Computer Arithmetic, pp. 104-111, June 2003.

[3] W.Buchholz., "Fingers or Fists? (The Choice of Decimal or Binary Representation)", Communications of the ACM, 2 (12), pp. 3-11, December 1959.

[4] T.B. Juang, H.H. Peng, H.L. Kuo, "Parallel and digit-serial implementations of area-efficient 3-operand decimal adders", International Journal of Soft Computing and Engineering (IJSCE), vol. 3, issue 5, pp. 177-182, November 2013.

[5] C.Sundaresan, C.V.S. Chaitanya, P.R. Venkateswaran, S.Bhatt and J. Mohan Kumar, "High speed BCD adder", Proc. 2011 2nd International Congress on Computer Applications and Computer Science, Advances in Intelligent and Soft Computing. Vol. 145, pp. 113-116, 2012.

[6] A. Bayrakci and A. Akkas, "Reduced delay BCD adder", Proc.. IEEE 18th Int. Conf. on Application-specific Systems, Architectures and Processors, (ASAP), pp. 266-271, July 2007.

[7] A. Vazquez and E. Antelo, "A high-performance significant BCD adder with IEEE 754-2008 decimal rounding," Proc. 19th IEEE Symposium on Computer Arithmetic (ARITH-19), pp. 135-144, 2009.