

Restructuring adders for Binary/BCD addition/subtraction under the condition of no sign bit in BCD format

Tara Tavakoli, Nastaran Parvin, Safiyeh Nikkhah Sani and Seyed Reza Talebian

Department of Computer Engineering,
Imam Reza International University, Mashhad, Iran

Abstract

This paper presents a new architecture for binary / BCD addition / subtraction to accelerate the operations and reduce delays in both signed numbers and unsigned numbers. Adding numbers in binary or BCD is one of the most important mathematical operations in digital systems. Various numbers of structures have been proposed to apply the BCD adder and subtraction, and the last structure presented in this article for BCD adder /subtraction of numbers assumes these numbers without any signs. Some articles have been used for some parts of this paper; and these articles, despite their strengths and advantages, contain some problems with their final computing. This paper has solved the problems in the final computing EOP. So, a new structure for addition / subtraction of BCD is presented if the sign bit is not taken into account and work properly.

Key words: adder, subtractor, signbit, condition, Restructuring

1) Introduction

Adding operation is one of the basic operations in all digital systems. Although binary adding is used frequently in many digital systems, BCD adding of digital numbers for is favored in several digital systems, especially in those whose basis of interest is 10. BCD adding consists of more complications comparing to binary adding;

therefore, BCD adders contain a larger area and more complexity than binary adders. [1] [2].

In [3], [4] and [5] all the presented BCD adders use high-speed adder binary structures such as CARRY LOOK AHEAD and CARRY SAVE ADDER to enhance the performance of their circuit. On the other hand, in [6], an adder has been presented and its unique feature is Restructuring, which means the adder can do BCD addition and subtraction in practice. The system presented in this paper assumes that the sign bit in BCD numbers is of no importance, and only in the Restructuring part, [6] has been used that consequently leads to an output signal named EOP that only activates at the end of the computation.

This signal has not had a proper performance and will cause some ambiguities in the future. This paper has considered an addition/subtraction for BCD without any sign and has tried to eliminate the weaknesses of EOP signal and to present an adder/subtractor with the ability of binary/BCD Restructuring, that explains the end of computation without any ambiguity. The structure presented in [6] and its problems will be discussed in the following section.

2) Mathematical background

Definition of BCD numbers:

In this, the equivalent of binary numbers in the basis of interest 10 is directly used.

Binary bits can be given a weight or value due to their location. This approach is in the 1,2,4,8 code of BCD. For example the code 0110 due to the value of bits shows the number of 6 decimal because $0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 = 6$ and also the negative values could be allocated to the decimal code in the figure of -1, -2, 4, 8. In this case of 0110, the number 2 is interpreted and is calculated as follows: $0 \times 8 + 1 \times 4 + 1 \times (-2) + 0 \times (-1) = 2$

Addition BCD numbers:

Adding two decimal numbers in BCD, with the possible carry, choose the previous number with the lower value from the pair; because the number is not more than 9, which means the adding cannot be more than $1+9+9=19$, where the number 1 is the previous carry. If we want to do the same adding in BCD, it will be equivalent to 0000 to 1001, in which the number 1 is the first number, the carry, and the rest of the adding in BCD. Adding two BCD numbers with n digits and no sign would be the same.

Subtraction of BCD numbers:

As discussed earlier, BCD numbers are basically decimal numbers, which only differ in the way of illustration. Therefore, for the subtraction of two BCD numbers named N and M, the second 10's or 9's complement must be found.

Subtraction of BCD numbers:

As discussed earlier, BCD numbers are basically decimal numbers, which only differ in the way of illustration. Therefore, for the subtraction of two BCD numbers named N and M, the second 10's or 9's complement must be found.

3) Explanation of the proposed circuit performance

3-1) The Binary case

If X and Y are two numbers in case of **Signed magnitude n + 1 bit** where

$$X = [X_n X_{n-1} X_{n-2} \dots X_0]$$

$$\text{and } Y = [Y_n Y_{n-1} Y_{n-2} \dots Y_0], \text{ in which } X_n \text{ and } Y_n,$$

then the numbers signed bits are binary and BCD numbers. In this structure we consider some signals called EOP, bin, and op where op identifies the adding and the subtraction operations (the input data op=0 illustrates the adding operation and the input data op=1 illustrates the subtraction). And the variable bin determines the kind of operation which is done based on the binary or BCD; and in the end EOP (END OPERATION) which determines the kind of final operation considering sign bit (where EOP=0 illustrates subtraction and EOP=1 illustrates the adding operation).

$$(1) \text{ EOP} = [\text{bin} ((X_n \odot Y_n) \oplus \text{op})] + [\text{bin}' . \text{op}']$$

					$\text{EOP} = [\text{bin} ((X_n \odot Y_n) \oplus \text{op})] + [\text{bin}' . \text{op}']$	
X_n	Y_n	$(X_n \odot Y_n)$	(add) op	(sub) op	Add	Sub
0	0	1	0	1	1	0
0	1	0	0	1	0	1
1	0	0	0	1	0	1
1	1	1	0	1	1	0

Table 1 EOP definition

After the EOP operation was found using EOP equation, the result sign in binary system is calculated using the first operand sign X_i . $e.X_n$ and carry-out. If the final effective operation is adding, the result sign will be equal to the first operand sign $i.e.X_0$. But if the effective operation is subtraction, the result sign will depend on X sign and also the adder circuit carry-out. This shows if $X > Y$ and $\leq Y$, the final result sign for adding will be $S_n = X_n$ if $EOP = 1$ & $bin = 1$ i.e. and the final result sign for subtraction will be $S_n = X_n \oplus (cout)$ if $EOP = 0$ & $bin = 1$. The final result sign in both adder/subtraction in BCD case, considering that sign bit in BCD numbers is of no value, will be zero that is shown in the figure below.

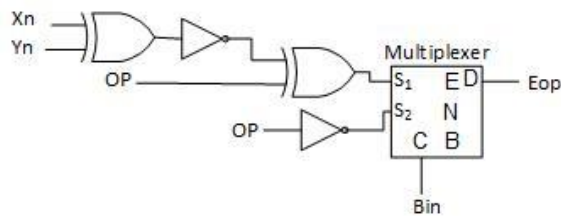


Figure 1: Eop

3-2) The proposed circuit performance in decimal case.

Designing and applying the adder used in [6] and this paper is based on the structure of sklansky [7] adder.

The addition and subtraction of BCD numbers that are considered without any signs will be discussed here and examples for both BCD addition and subtraction will be provided. It must be taken into account that, while adding BCD numbers, if for each carry lever there is another level, the correction block operation will be done; but if there is not any next level, no correction block operation would be necessary; which means the carry is transferred through one

level and it is not transferred from one level to another.

$X = 547$ In BCD Format 010101000111

$Y = 328$ In BCD Format 001100101000

In continue, first the amount of X NEW is built and then the addition of BCD will be done, under the condition mentioned.

$$X_{New} = X + 6$$

$$010101000111 + 011001100110 = 101110101101$$

$$\begin{array}{r} X_{New} \ 101110101101 \\ + \\ Y \ \ \ 001100101000 \\ \hline 111011010101 \\ + \\ 10101010 \\ \hline 100001110101 \end{array}$$

Subtraction is examined in both cases of $X > Y$ and $\leq Y$. The condition here is that carry is transferred from one level to another, and if $> Y$, the subtraction is binary ($Subtraction = X + (-Y)$ where $-Y$ is Y NEW), and in the beginning 1's complement calculates Y, and as mentioned in [6], the subtraction is done. See the process in the example below.

Because in the last level, carry out is produced, the number from the addition is added up with 1, and then the number from the subtraction is added up with 6, to change the result to BCD

$X = 547$ In BCD Format 010101000111

$Y = 328$ In BCD Format 001100101000

Y New = Ones complemet Y

$$\begin{array}{r}
 Y\ New = 110011010111 \\
 X = 010101000111 \\
 + \\
 Y\ New = 110011010111 \\
 \hline
 + \quad 1\ 001000011110 \\
 \hline
 + \quad 1\ 001000011111 \\
 \hline
 + \quad \quad \quad 1010 \\
 \hline
 001000011001
 \end{array}$$

* Because in the last level, carry out is produced, the number from the addition is added up with 1, and then the number from the subtraction is added up with 6, to change the result to BCD.

Let's take a look at the example of subtraction when $X \leq Y$:

X = 328 In BCD Format 001100101000

Y = 547 In BCD Format 010101000111

$$\begin{array}{r}
 Y\ New = Ones complemet Y \\
 Y\ New = 101001111000 \\
 + \\
 X = 001100101000 \\
 \hline
 \quad \quad 110111100000 \\
 + \quad (NOT) \quad 001000011111 \\
 \hline
 \quad \quad \quad 1010 \\
 \hline
 001000011001
 \end{array}$$

* Because in the last level, carry out is not produced, the result number must be inversed (NOT) to calculate the subtraction result in binary, and then again, add it up with 6 to change it to BCD.

4) The proposed architectural design

The proposed architecture includes different parts such as: pre-correct X and Y "the idea of pre-correct is taken from [6]", post-correct and adder, and restructuring,. Each design part is discussed later individually.

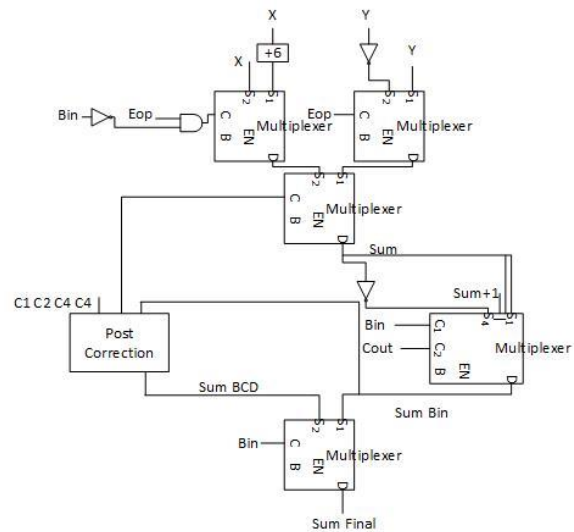


Figure 2: Final architecture

4-1) Designing sign bit logical expression.

Considering the fact that sign bit for BCD numbers is of meaning (sign bit for BCD numbers is considered zero), the proposed circuit for different values of EOP and bin will operate as follows. There are two possible cases for sign bit in binary circuit; one is when EOP=0 , in which subtraction is done and the sign depends on Cout (which means if Cout= 0 , then $X < Y$, and the sign bit is equal to the NOT (inversed) of the X sign; otherwise, it would be Cout=1 that shows $X > Y$ and then the sign bit is equal to the bigger number sign.

In this case, the bin select line in added to mux lines, that if Bin=0, shows the subtraction of BCD. In subtraction of these numbers, only smaller numbers are subtracted from the bigger ones and the sign is disregarded.

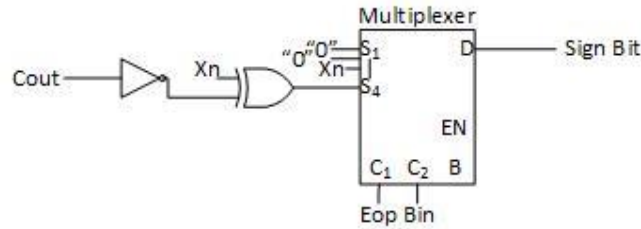


Figure 3: Sign bit architecture

EOP	Bin	EXPLIANTION
0	0	This case shows that the positive number has been taken into account and the subtraction is BCD and the sign bit is considered zero.
0	1	In this case, the subtraction is done on binary numbers and there are two possible cases for sign bit in binary circuit; one is when EOP=0 , in which subtraction is done and the sign depends on Cout (which means if Cout= 0, then X<Y , and the sign bit is equal to the NOT (inversed) of the X sign; otherwise, it would be Cout=1 that shows X>Y and then the sign bit is equal to the bigger number sign.
1	0	This means that the number is BCD, so the sign is of no meaning in that.
1	1	This means that addition is to be done and the Xn sign will be considered for the result.

Table 2 Sign bit definition

4-2) The design of a mux to choose addition/subtraction in the idea of pre-correction, because the binary addition depends on the amount of Cout and Eop, and to choose which one to use according to the situation.

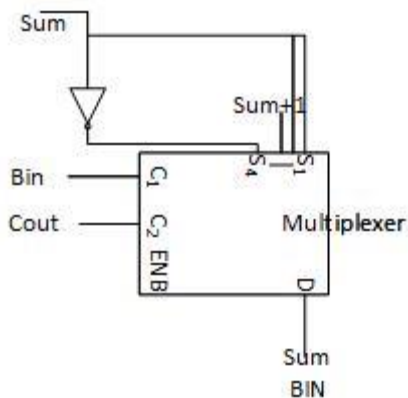


Figure4 : Pre correction adder/subtractor architecture

4-3) The design of the addition/subtraction post-correction circuit (BCD).

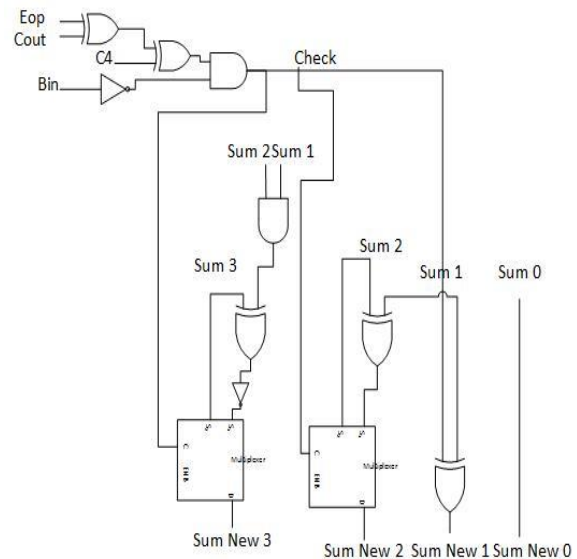


Figure5 : Post correction BCD architecture

5) Simulation:

The XILINX software is often used in all of the simulations, and also here, the proposed circuit architecture is compared to the presented architecture in [6], and the results

in case of Delay and Logic Utilization are provided below.

	[6]	MY ARCHITECTURES
DELAY	36.619 ns	35.665ns
Logic Utilization	115	110

Table 3 Result definition

6) Conclusion

What was discussed in this paper was the design of a BCD/binary adder and subtraction with an ability of reconfiguration, assuming the sign bit in BCD numbers is of no meaning. Assuming that and without using any extra operation, the correct answer could be found and the problems faced with during this time from the article number [6] were solved and improved.

7) References

[1]M. J. Adiletta and V. C. Lamere. BCD AdderCircuit.Digital Equipment Corporation, US patent 4805131, pages 1 – 18, Jul 1989
 [2] Computer Arithmetic Algorithms, Israel Koren; Pub A K Peters, 2002
 [3]Sreehari Veeramachaneni, Kirthi Krishna M, Prateek G V, Subroto S, Bharat S, M.B. Srinivas, A Novel Carry-look ahead approach to an Unified BCD and Binary Adder/Subtractor,10639667/08\$25.00©2008 IEEE
 [4]Osama Al-Khaleel_,Mohammad Al-Khaleel§, Zakaria Al-Qudah§,Christos A. Papachristou¶, Khaldoon Mhaidat_, and Francis G. Wolff¶_ Jordan University of Science and Technology, Irbid, Jordan, Fast Binary/Decimal Adder/Subtractor with a Novel Correction-Free BCD Addition, 978-1-4577-1846-5/11/\$26.00 ©2011 IEEE
 [5]Kuan Jen Lin, Ju Lin Shih, Tsz Hao Lin and Yu Mei Wang, A Parallel Decimal Adder with Carry]Correction during Binary

Accumulation,978-1-4673-08595/12/\$31.00 ©2012 IEEE
 [6]SyedErshadAhmed,SreehariVeeramancharani,MoorthyMuthukrishnanN,M.BSrinivas, Reconfigurable Adders for Binary/BCD addition/Subtraction,978-1-4577-1610-2/11/\$26.00 ©2011 IEEE
 [7] J. Sklansky, "Conditional-sum addition logic," IRE Trans.Electronic Computers,vol. EC-9, pages 226-231, June 1960
 [8] M.S .Schmookler and A.W. Weinderger, "High Speed,Decimal Addition", IEEE Transactions on. Computers,vol. C-20, pp. 862-867, August 1971.
 [9] U. Grupe, "Decimal Adder," Vereinigte Flugtechnische WerkeFokker gmbH, US patent 3935438, pp. 1–11, Jan 1976.
 [10] M. F. Cowlshaw, "Decimal Floating-Point: Algorism for Computers," Proc. 16th IEEE Symp. Computer Arithmetic (ARITH '03), pp. 104-111, 2003.