Design of Software Approach for Speeding up Addition Arithmetic Operation

Imad Matti Bakko
Al Mamoon University College, Computer Scince Department/Baghdad
Email:emad_matti@yahoo.com

Received On: 17/9/2012 & Accepted On: 10/1/2013

ABSTRACT

This paper presents a new method to perform arithmetic addition operation on numbers in a faster way in comparison with the exist one on computers.

The proposed method builds a new architecture for the Adder Circuit in the CPU, which does not perform any carry operations. In fact, there is no need for a waiting time to perform carrying bits from low order positions to high order positions when adding two numbers.

The new method is successfully tested with many different examples.

Keywords: Shift, Rotate and Add operations, Carry concept, Adder Circuit, Clock Cycles.

تصميم طريقة برمجية لتسريع عملية الجمع الحسابية

الخلاصة

يقدم هذا البحث طريقة جديدة لتنفيذ عملية الجمع الحسابية على الإعداد بصورة أسرع مقارنة بما هو معتمد عليه حاليا في الحاسبات الالكترونية.

تقترح هذه الطريقة بناء معمارية دائرة الجامع (adder circuit) في المعالج المركزي , بحيث لا وجود فيها لعملية التحميل (carry), حيث لايعد هنالك حاجة إلى وقت للانتظار (waiting time) عند تنفيذ التحميل (carry bit) من المرتبة السابقة للعدد إلى المرتبة اللاحقة له وذلك عند جمع عددين.

تم اختبار تفاصيل الطريقة الجديدة بنجاح على امثلة عديدة مختلفة.

https://doi.org/10.30684/etj.31.3B.11
2412-0758/University of Technology-Iraq, Baghdad, Iraq

INTRODUCTION

To implement the add micro operation in computer, we need registers that hold data, and digital components that perform the arithmetic addition. Figure (1), shows block diagram, which accepts two binary digits on it's inputs, and produces two binary digits on it's outputs, a sum bit, and a carry bit[1].

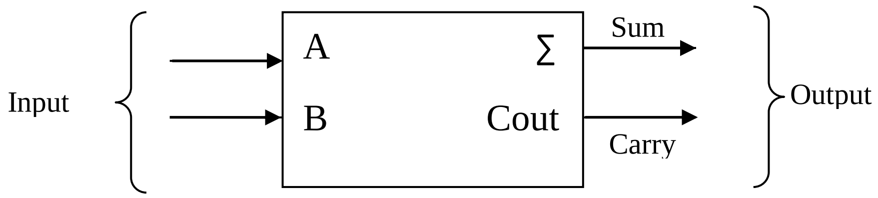

Figure (1) block diagram of the add micro operation.

The digital circuit that generates the arithmetic sum of two binary numbers of any lengths is called binary adder.

The binary adder is constructed with full-adder circuits connected in a cascade, with the output carry from one full-adder connected to the input carry of the next full-adder. Figure (2), shows the inter connections of four full-adders (FA) to provide a 4-bit binary adder [2], [3].

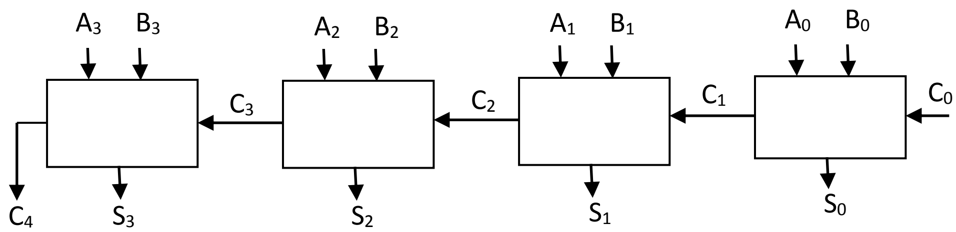

Figure (2) 4–bit binary adder.

The augends bits of A and the addend bits of B are designated by subscript numbers from right to the left, with subscript 0 denoting the low-order bit. The carries are connected in a chain through the full-adders.

The input carry to the binary adder is C_0 and the output carry is C_4. The S outputs of the full-adders generate the required sum bits [1][2].

Since the output carry from each full-adder (FA) is the input carry of the next-high-order full-adder, hence to generate the sum S1 for example it depends on the carry C_1 generated from the previous full-adder (FA) and so forth. This situation does not speed up the add micro operation, since there is a waiting time to generate a carry bit as an input to the next full-adder.

THE PROPOSED METHOD FOR ADDING TWO4-BITS BINARY NUMBERS

$$\text{Let } A = \overset{2^2}{(a_2},\ \overset{2^1}{a_1},\ \overset{20}{a_0})_2, \text{ and } B = \overset{2^2}{(b_2},\ \overset{2^1}{b_1},\ \overset{2^0}{b_0})_2,$$

Be two binary numbers.

We put the digits of the number A in 3 registers, say A_1, A_2, and A_3 in the following manner:-

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	0	0	0	0	0	a_0	$\leftarrow A_1$
0	0	0	0	0	0	a_1	0	$\leftarrow A_2$
0	0	0	0	0	a_2	0	0	$\leftarrow A_3$

i.e. the digit a_0 in the position 2^0 of the number A will be in the position 2^0 of register A_1.

The digit a_1 in the position 2^1 of the number A will be in the position 2^1 of register A_2.

The digit a_2 in the position 2^2 of the number A will be in the position 2^2 of register A_3.

We put also the digits of the number B in register say R as follows:-

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	$\leftarrow$ R

For the sum of the number A and B, we use a register say S, and we put 0 in all positions of it, as follows:-

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	0	0	0	0	0	0	$\leftarrow$ S

Now, to add the numbers A and B into the sum register S, the method suggest the truth table which is shown in Table (1).

Table (1) Suggested Truth Table for Adder Circuit.

+	0	1
0	0	1
1	1	Either 1 or 0 (without carry) depending on the Algorithm presented in this paper.

Algorithm for adding two 4-bits numbers.

Input : Two 4-bit numbers
Output : Sum of 4-bit numbers
1. START.
2. i ⟵ 0.
3. **DO**
If the digit in the position 2^i of register A_1 is not equal to 1, **then**
begin {1}
  if the digit in the position 2^i of register R is not equal to 0, **then**
  begin {2}
    if the digit in the position 2^i of register A2 is equal to 0, **then**
      begin {3}
      if the digit in the position 2^i of register A3 is not equal to 0, then
        begin {4}
        a. shift to the left the digit in the position 2^i to the position 2^{i+1} of register A3.
        b. put 0 instead of 1 in the position 2^i of register R, and A3.
        end {4}
      end {3}
    else

      begin{5}
        a. shift to the left the digit in the position 2^i to the position 2^{i+1} of register A2.
        b. put 0 instead of 1 in the position 2^i of register R, and A2.
      end {5}
  end {2}
end {1}
else
  begin {6}
    if the digit in the position 2^i of register R is not equal to zero **then**
      begin {7}
      a. Shift to the left the digit in the position 2^i to the position 2^{i+1} of register A1.
      b. put 0 instead of 1 in the position 2^i of register R, and A1.
      end {7}
  else
    if the digits in the position 2^i of registers A_2 and A_3 are not equal to 0 **then**

begin {8}

a. put 0 in the position 2^i of registers A_2 and A_3.

b. shift to the left the digit in the position 2^i to the position 2^{i+1} of register A_1.

c. put 0 in the position 2^i of register A1.

end {8}

end {6}

Add the digits in the position 2^i of registers A_1, A_2, A_3, and R.

put the sum in the position 2^i of register S.

i = i + 1.

WHILE (i < 3)

4. Add the digits in the position 2^3 of registers A_1, A_2, A_3, and R.

5. Put the sum in the position 2^3 of register S.

6. STOP.

Note1: The number in the register S will be the result of adding the numbers A and B.

Example for Adding Two 4-Bits Numbers According to Proposed Algorithm

Suppose the number $A=(011)_2$ is added to the number $B=(110)_2$, without performing the carry operations. the following steps can be followed:

1. Put the digits of the number A in 3 registers, say register A_1, register A_2, and register A_3, in the following manner (since number A consists of 3 digits):

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2°	
0	0	0	0	0	0	0	1	⟵ Register A1
0	0	0	0	0	0	1	0	⟵ Register A2
0	0	0	0	0	0	0	0	⟵ Register A3

The digit in the position 2° of A will be in the position 2° of register A_1.
The digit in the position 2^1 of A will be in the position 2^1 of register A_2,
And the digit in the position 2^2 of A will be in the position 2^2 of register A^3.

2. put the digits of the number B in register R as follows:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	0	0	0	1	1	0	⟵ register R

3. put 0 in all positions of register S (for the sum) as follows:-

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	0	0	0	0	0	0	⟵ register S

4. Since the digit in the position $2^0$of register A_1 is equal to 1, and the digit in the position $2^0$of register R is equal to 0, and the digits in the positions 2^0 of registers A_2 and A_3 are equal to 0, we add the digits in the position 2^0 of registers A_1, A_2, A_3, and R. then we put the sum in the position 2^0 of register S, which will be 1, as follows:-

2^3 2^2 2^1 2^0

0	0	0	**1**	← register S

5. Since the digit in the position 2^1 of register A_1 is equal to 0, and the digit in the position 2^1 of register R is equal to 1, and the digit in the position 2^1 of register A_2 is equal to 1, we shift to the left the digit in the position 2^1 of register A_2 to the position 2^2 of it.
Then put 0 in the position 2^1 of register R.
And take the sum of the digits in the position 2^1 of registers A_1, A_2, A_3, and R.
Put the sum in the position 2^1 of register S, as follows:-

2^3	2^2	2^1	2^0	
0	0	**0**	**1**	← register S

6. Since the digit in the position 2^2 of register A_1 is equal to 0, and the digit in the position 2^2 of register R is equal to 1, and the digit in the position 2^2 of register A_2 is equal to 1, we shift to the left the digit in the position 2^2 of register A_2 to the position 2^3 of it.
Then put 0 instead of 1 in the position 2^2 of register R.
Take the sum of the digits in the position 2^2 of registers A_1, A_2, A_3, and R.
Put the sum in the position 2^2 of register S, as follows:-

2^3	2^2	2^1	2^0	
0	**0**	**0**	**1**	← register S

7. Add the digits in the position 2^3 of registers A_1, A_2, A_3, and R.
Put the sum in the position 2^3 of register S, as follows:-

2^3	2^2	2^1	2^0	
1	**0**	**0**	**1**	← register S

The number in the register S is the result of adding the numbers A and B, which is the same result of adding A and B with carry.

Testing the Proposed add method with many other test data:-

In this subsection, 9 different samples for adding two binary numbers A and B, are introduced.

The results of applying the suggested add method are the same results of adding them with carry.

Test data 1

					2^3	2^2	2^1	2^0		
A=	0	1	1	1	0	0	0	1	←	A_1
					0	0	1	0	←	A_2
B=	0	0	1	1 +	0	1	0	0	←	A_3
					0	0	1	1	←	R
S=	1	0	1	0	1	0	1	0	←	S

With using carry. **Without using carry** according to Algorithm.

Test data 2

A=	0	1	1	1	
B=	0	1	0	1	+
S=	1	1	0	0	

Test data 3

A=	0	1	1	1	
B=	0	1	1	1	+
S=	1	1	1	0	

Test data 4

A=	**0**	**1**	**1**	**1**	
B=	**0**	**1**	**1**	**0**	+
S=	**1**	**1**	**0**	**1**	

Test data 5

A=	**0**	**1**	**1**	**0**	
B=	**0**	**1**	**1**	**1**	+
S=	**1**	**1**	**0**	**1**	

Test data 6

A=	**0**	**1**	**0**	**0**	
B=	**0**	**0**	**1**	**1**	+
S=	**0**	**1**	**1**	**1**	

Test data 7

A=	**0**	**1**	**1**	**1**	
B=	**0**	**1**	**0**	**0**	+
S=	**1**	**0**	**1**	**1**	

Test data 8

A=	0	1	0	1	
B=	0	1	0	1	+
S=	1	0	1	0	

Test data 9

A=	0	1	0	1	
B=	0	1	1	0	+
S=	1	0	1	1	

ADD OPERATIONS VS. SHIFT OPERATION AND ROTATE OPERATIONS

In 8085 microprocessor:

The 8085 microprocessor does not provide a shift instruction: however, it does provide two forms of rotate in two directions:

1. RLC Rotate left instruction.
2. RRC Rotate right instruction.
3. RAL Rotate left instruction through carry.
4. RAR Rotate right instruction through carry.

To compare the delay time between add operations, from one side and the rotate operations, from the other side, we must know the time required for each instruction.

Table (2) lists some of the 8085 add instructions set in comparison with rotate instructions, along with delay information [3].

Mnemonic	Cycles	SDK-85
ADD M	7	2.28
ADC M	7	2.28
ADI	7	2.28
ACI	7	2.28
DAD	10	3.26
RLC	4	1.30
RRC	4	1.30
RAL	4	1.30
RRC	4	1.30

It’s obvious from a Table (2) that the rotate instructions require less number of clock cycles in comparison with the add instructions. That is the reason behind the suggestion of the method mentioned, which include shifting and a kind of add operation, which does not include any carry concept.

In 8086, 8088 and other:-

These microprocessors provide a set of shift instructions and a set of rotate instructions which position or move numbers to left or right within register or memory location [4].

The set of shift and rotate instructions are shown in Table (3).

Table (3) 8086 shift and rotate instructions.

ABBRIVIATION	INSTRUCTION
SHL	Shift Logical Left
SAL	Shift Arithmetic Left
SHR	Shift Logical Right
SAR	Shift Arithmetic Right
RCL	Rotate Left Through Carry
ROL	Rotate Left
RCR	Rotate Right Through Carry
ROR	Rotate Right

Table (4), illustrates the differences in the delay time (in clocks) between the add operations from one side and the shift and rotate operations from the other side [4].

Table (4) 8086, 8088 and others microprocessors and their instructions times.

Format	Microprocessor	Clocks
ADC reg , reg	8086, 8088	3 , 3 respectively
ADC mem , reg	8086, 8088, 80286, 80386, 80486	16+ea , 24 respectively 7 , 7 , 3 respectively
ADC reg , mem	8086, 8088, 80286, 80386	9+ea , 13 +ea respectively 7 , 6 respectively
SAL reg , 1	8086	2
SHL reg , 1	8088	2
SAL mem , 1	8086	15 + ea
SHL mem , 1	8088	23 + ea
ROL reg , 1	8086,8088	2 , 2 respectively
RCL	8086,8088	2 , 2 respectively

This method presents a new design of the adder circuit, since the shift instructions (rotate instructions in 8085) is simple, cheap, fast, and does not cost any waiting time [5], [6], [7], [8].

CONCLUSIONS

The proposed approach is successfully implemented and tested. But some points can be inferred:

1- The add operation is built on the carry concept by hardware means, while the proposed algorithm is built by software means.

2- Since there is always a waiting time to add the carry bit from low-order position to high-order position , hence the add with carry operation will be slower in comparison with the proposed add operation.
3- The idea of proposed algorithm is to exchange carry operation with (move or shift) operation to reduce the execution time.

4- Its recommended to develop a general algorithm that extends the numbers A and B as follows:-
$A = (a_7 , . . . , a_2 , a_1 , a_0)_2$ and $B = (b_7 , . . . , b_2 , b_1 , b_0)_2$.

5- Its recommended to develop a general algorithms to perform all other arithmetic operations.

REFERENCES

[1]. Morris Mano, "Digital Fundamentals ", Prentice Hall, International Education, Eighth Edition, 2003.
[2]. Morris Mano and Charles R. Kime, M. "LOGIC and COMPUTER DESIGN FUNDAMENTALS ", Pearson International Edition, Pearson Prentice Hall, fourth Edition, Printed in Singapore, 2008.
[3]. Ramesh S. Gankar, "Microprocessor Architecture, programming, and Application with the 8085", Fifth Edition, person Education, Inc, 2002.
[4]. Barry B. Brey, "The Intel Microprocessor Architecture, Programming, and Interfacing", Pearson International Edition , printed in the United States of America , Pearson Prentice-Hall, Eighth Education , 2009.
[5]. " LOOKAHEAD ADDERS - CARRY [PDF]"
http://writphotec.com/mano4/Supplements/Carrylookahead_supp4.pdf
www.google.com
[6]. Fast Addition Carry Lookahead" , "Harvey Mudd College"
http://writphotec.com/mano4/Supplements/Carrylookahead_supp4.pdf
www.yahoo.com
[7]. " Carry-lookahead Adder ", WIKIPEDIA the Free Encyclopedia.
WWW.YAHOO.COM
[8]**. "** Carry- save Adder **",**Wikipedia , the free Encyclopedia.
www.yahoo.com