ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 4, No. 1, 2019

LOGICAL ALGORITHMS OF THE ACCELERATED MULTIPLICATION WITH MINIMUM QUANTITY OF NONZERO DIGITS OF THE CONVERTED MULTIPLIERS

Ihor Korol, Ivan Korol

Uzhhorod National University, 46 Pidhirna St, Uzhhorod, Transcarpathia, Ukraine, 88000.
Author's e-mail*:* ivan.korol@uzhnu.edu.ua
Uzhhorod National University, 46, Pidhirna Str., Uzhhorod, Transcarpathia, Ukraine, 88000,
John Paul II Catholic University of Lublin, 14 Al. Racławickie, Lublin, Poland, 20-950.
Author's e-mail: ihor.korol@uzhnu.edu.ua, ihorkorol@kul.lublin.pl

Submitted on 30.06.2019

***Abstract*: The article presents a new algorithm of accelerated multiplication, in which the time of multiplication has been reduced through the decrease in the number of nonzero digits of the multiplier. In this case, the multiplier has been presented in the form of the extended binary code. The article proves the algorithm's efficiency in comparison to previously known methods. The developed algorithm has been implemented using the hardware description language AHDL (Altera Hardware Description Language) in the Logic Development System MAX+PLUS II.**

***Index Terms*: extended binary code, accelerated multiplication device, AHDL, MAX+plus.**

I. INTRODUCTION

As the analysis of the time of execution of certain arithmetic operations in the computer shows, the time spent on multiplication makes up 70–80 % of the total time spent on this operation. Therefore, the development of the accelerated multiplication algorithms, which reduces the execution time, is essential [1–3].

The easiest logical way to accelerate multiplication is to skip adding steps in cases where in binary numeral system the following digit of the multiplier is zero.

The most effective and logical way to accelerate multiplication is to use addition and subtraction operations, also called the method of a sequential transformation of the multiplier numbers. In this case, the multiplier is transformed into the extended binary code (binary quasi-canonical number system) with numbers $\{-1,0,1\}$. Thus, to choose the one that contains the minimum number of non-zero digits {-1, 1} from the possible representations of the multiplier. For this purpose, the group composed of $m \geq 2$ items represented as $0\underbrace{11\ldots1}_{m}0$ is converted into the group represented like $1\underbrace{00\ldots\bar{1}}_{m}0$, where $1\ \bar{1} = -1$. In this case, the number that is multiplied is added to the sum of partial products, if the next digit of the converted multiplier is equal to 1, and subtract or add a complement code of multiplication if the digit of the converted multiplier is equal to $\bar{1}$.

II. KEY METHODS OF MULTIPLICATION PROCESS ACCELERATION

The key methods of accelerated multiplication with the use of the extended binary code are as follows [1, 2, 4]:

accelerated logical multiplication with the recoding of the multiplier by two digits, and by analyzing two digits of the initial multiplier and transposition, starting with the lower-order digit (algorithm A);

accelerated logical multiplication with recoding of the multiplier by two digits and by analyzing three digits of the initial multiplier, starting with the highest digits (algorithm B). Note that algorithm B is also called the modified Booth's algorithm [5].

Note that for the conversion of the multiplier with the corresponding algorithms separate tables were developed (see tables 1, 3). However, the multipliers converted with the help of these algorithms may differ in the quantity of non-zero digits (while the quantity of non-zero digits is not always minimal). This is demonstrated by the example given below.

To obtain a converted multiplier with a minimum quantity of nonzero digits, we have developed a new algorithm:

the conversion of the multiplier is realized by analyzing two digits of the multiplier and the transposition from the lower-order digits. As a result we get one digit of the converted multiplier and a transposition.

The conversion algorithm C is carried out using Table 5. Note that after the conversion of the multiplier, the multiplication algorithm can be implemented both with processing one and two digits, as well as for the digits of both lower-order and higher-order.

A. ALGORITHM A

The conversion of a multiplier into an extended binary code is performed according to the algorithm A. The denominations are used in Table 1:

$\bar{y}_i \bar{y}_{i+1}$ – converted digits of the multiplier Y;

p_{i+1} – transposition, that already exists (its initial value is 0);

p_i – the new value of the transposition;

$L1X$ – the shift of the multiplied number X one digit to the left;

$(-X)^m_{DK}$ – modified complement code of the multiplied number X;

R2S – the adder's S digits shift two digits to the right;

R2Y – the converted multiplier Yp digits shift two digits to the right.

Example 1. To recode the multiplier Y=00.11011110 using Table 1, starting with the lower-order digits $y_7 y_8$. The initial value of the transposition is $p_8 = 0$. The end value of the transposition is $p_0 = 1$, which is assigned to the higher-order digit of the converted multiplier $Yp_0 = 1$.

The classical logical algorithm for conversion of the multiplier's digits with the analysis of two digits in pairs and the transposition, starting with the lower-order digits gives the result, shown in the Table 2. Thus, a result with a smaller (sometimes with a minimum) quantity of non-zero digits may be obtained via simplification of the combination $\overline{1}\,1 \to 0\,\overline{1}$.

B. ALGORITHM B

Let us analyze the process of the multiplier conversion into an extended binary code following the algorithm B.

In Tabble 2 we observe that the initial multiplier Y=00.11011110 contains 6 non-zero digits. While it is converted according to the algorithm A then the multiplier Y_p contains 5 non-zero digits and the minimized multiplier Y_{min} contains 3 non-zero digits.

Table 3 provides the analysis of three digits at a time $y_i y_{i+1}\, y_{i+2}$ for the given multiplier Y. As a result there appear two new digits $\overline{y}_i \overline{y}_{i+1}$ of the converted multiplier Y_p; Besides, the denominations mentioned above, the following denominations are introduced:

L2S – the adder`s S digits shift two digits to the right;

L2Y – the given multiplier's Y digits shift two digits to the left.

Note that in analyzing the last pair of digits of the multiplier, the value of the neighboring digit on the right side is assumed equal to 0.

Example 2. To recode the multiplier Y=00.11011110 using Table 3**,** starting with the higher-order digits $y_{-1} y_0 y_1 = 00.1$.

From the obtained table it may be seen that the resulting converted multiplier has one digit less than in the previous case, but it is not minimal in the quantity of non-zero digits.

The conversion algorithm B for the multiplier digits Y=00.11011110 with the analysis of three digits at a time, starting with the higher-order digits provides the result given in table 4. Thus, a result with a smaller (sometimes with a minimum) number of non-zero digits may be obtained via simplification of the combination $\overline{1}\,1 \to 0\,\overline{1}$.

Table 1

Conversion of the multiplier according to the algorithm A

Transposition p_{i+1}	Pair of digits under analysis $y_i y_{i+1}$		Converted pair of digits $\overline{y}_i \overline{y}_{i+1}$	Following transposition p_i	Remarks
0	00		00	0	S:=R2S , R2Y
0	01		01	0	S:=S+X , S:=R2S , R2Y
0	10	=>	10	0	S:=S+ $L1X$, S:=R2S , R2Y
0	11		$0\,\overline{1}$	1	S:=S+ $(-X)^m_{DK}$, S:=R2S , R2Y
1	00		01	0	S:=S+X , S:=R2S , R2Y
1	01		10	0	S:=S+ $L1X$, S:=R2S , R2Y
1	10		$0\,\overline{1}$	1	S:=S+ $(-X)^m_{DK}$, S:=R2S , R2Y
1	11		00	1	S:=R2S , R2Y

Table 2

Conversion of the multiplier Y=00.11011110 according to the algorithm A

	Digit number										Quantity of non-zero digits
	-1	0	1	2	3	4	5	6	7	8	
p_i		1		0		1		0		0	
Y	0	0	1	1	0	1	1	1	1	0	6
Y_p	0	1	0	$\overline{1}$	1	0	0	$\overline{1}$	1	0	5
Y_{min}	0	1	0	0	$\overline{1}$	0	0	0	$\overline{1}$	0	3

Table 3

Conversion of the multiplier according to the algorithm B

Pair of digits under analysis $y_i y_{i+1}$	Neighboring digit to the right y_{i+2}	Converted pair of digits: $\bar{y}_i \bar{y}_{i+1}$	Remarks
00	0	00	S:=L2S , L2Y
01	0	01	S:=S+X , S:=L2S , L2Y
10	0	$\bar{1}0$	S:=S+ $(-L1X)^m_{DK}$,S:=L2S, L2Y
11	0	$0\bar{1}$	S:=S+ $(-X)^m_{DK}$, S:=L2S , L2Y
00	1	01	S:=S+X , S:=L2S , L2Y
01	1	10	S:=S+L1X, S:=L2S , L2Y
10	1	$0\bar{1}$	S:=S+ $(-X)^m_{DK}$, S:=L2S , L2Y
11	1	00	S:=L2S , L2Y

C.

D. ALGORITHM C

In the proposed algorithm C, the conversion is carried out by analyzing two digits of the multiplier at a time and transposition from the lower-order digits and obtaining a single digit of the transformed multiplier and transposition. Transfer to the lower-order (right) digit is equal to zero. Further on it is predetermined by the Table 5. The last transposition is recorded to the higher-order (in this case to 0) digit of the converted multiplier. Here:

R1S – the adder`s S digits shift one digit to the right;

R1Y – the converted multiplier's Y_p digits shift one digit to the right.

This logical algorithm for conversion of multiplier's digits with the analysis of two digits at a time, starting with the lower-order digits and the transposition, provides the result given in Table 6. It contains a minimal quantity of non-zero digits.

Example 3. To recode the multiplier Y=00.11011110 using Table 5, starting with the lower-order digits $y_7 y_8$. The initial value of the transposition is $p_8 = 0$. The resulting value of the transposition is $p_0 = 1$, which is assigned to the higher-order digit of the converted multiplier, is $Yp_0 = 1$.

From the obtained Table it may be deduced that the converted multiplier has two digits less than in case of the algorithm A, one digit less than while using the algorithm B, and it is the minimum possible for the quantity of non-zero digits for the present multiplier Y.

For control, we provided an illustration in the form of the stepwise conversion of the multiplier according to the algorithm C using the formulas below. Particular attention should be paid to the fact that the transposition $P_{-1} = 0$ may not be calculated.

1st step: k=8, $Y_7Y_8 = 10$, $P_8 = 0 \Rightarrow Yp_8 = 0, P_7 = 0$;

2nd step: k=7, $Y_6Y_7 = 11$, $P_7 = 0 \Rightarrow Yp_7 = \bar{1}, P_7 = 1$;

3rd step: k=6, $Y_5Y_6 = 11$, $P_6 = 1 \Rightarrow Yp_6 = 0, P_5 = 1$;

4th step: k=5, $Y_4Y_5 = 11$, $P_5 = 1 \Rightarrow Yp_5 = 0, P_4 = 1$;

5th step: k=4, $Y_3Y_4 = 01$, $P_4 = 1 \Rightarrow Yp_4 = 0, P_3 = 1$;

6th step: k=3, $Y_2Y_3 = 10$, $P_3 = 1 \Rightarrow Yp_3 = \bar{1}, P_2 = 1$;

7th step: k=2, $Y_1Y_2 = 11$, $P_2 = 1 \Rightarrow Yp_2 = 0, P_1 = 1$;

8th step: k=1, $Y_0Y_1 = 01$, $P_1 = 1 \Rightarrow Yp_1 = 0, P_0 = 1$;

9th step: k=1, $Y_{-1}Y_0 = 00$, $P_0 = 1 \Rightarrow Yp_0 = 1$.

III. IMPLEMENTATION OF THE ACCELERATED MULTIPLICATION ACCORDING TO THE ABOVE MENTIONED ALGORITHMS

The implementation of the accelerated multiplication is illustrated for each of the considered algorithms on the examples.

Example 4. To find the product of multiplication for: X=0.1101, Y=0.1101111.

Solution. The way of obtaining this product using each of the algorithms mentioned above is illustrated.

Table 4

Conversion of the multiplier Y=00.11011110 according to the algorithm B

	Digit number											Quantity of non-zero digits
	-1	0	1	2	3	4	5	6	7	8		
p_i			1		0		1		1		0	
Y	0	0	1	1	0	1	1	1	1	0		6
Y_p	0	1	0	$\bar{1}$	1	0	0	0	$\bar{1}$	0		4
Y_{min}	0	1	0	0	$\bar{1}$	0	0	0	$\bar{1}$	0		3

Table 5

Conversion of the multiplier according to the algorithm C

No.	Y_{k-1}	Y_k	P_k		Y_{pk} ??	P_{k-1}	Remarks
0	0	0	0		0	0	S:=R1S , R1Y
1	0	0	1		1	0	S:=S+X , S:=R1S , R1Y
2	0	1	0		1	0	S:=S+X , S:=R1S , R1Y
3	0	1	1	$\Rightarrow$	0	1	S:=R1S , R1Y
4	1	0	0		0	0	S:=R1S , R1Y
5	1	0	1		$\bar{1}$	1	S:=S+ $(-X)^m_{DK}$, S:=R1S , R1Y
7	1	1	1		0	1	S:=R1S , R1Y

Table 6

Conversion of the multiplier Y=00.11011110 according to the algorithm C

	Digit number										Quantity of non-zero digits
	-1	0	1	2	3	4	5	6	7	8	
p_i		1	1	1	1	1	1	1	0	0	
Y	0	0	1	1	0	1	1	1	1	0	6
Y_p	0	1	0	0	$\bar{1}$	0	0	0	$\bar{1}$	0	3

Table 7

Implementation of the accelerated multiplication according to the algorithm A

Y_p	Remarks	↓	P_{sign}	S												
	PS		0	0	0	0	0	0								
10	+L1X		0	1	1	0	1	0								
	S		0	1	1	0	1	0								
	R2S→		0	0	0	1	1	0	1	0						
$0\bar{1}$	+Xdm		1	1	0	0	1	1								
	S		1	1	1	0	0	1	1	0						
	R2S→		1	1	1	1	1	0	0	1	1	0				
10	+L1X		0	1	1	0	1	0								
	S	1	0	1	1	0	0	0	0	1	1	0				
	R2S→		0	0	0	1	1	0	0	0	0	1	1	0		
$0\bar{1}$	+Xdm		1	1	0	0	1	1								
	S		1	1	1	0	0	1	0	0	0	1	1	0		
	R2S→		1	1	1	1	1	0	0	1	0	0	0	1	1	0
01	+X		0	0	1	1	0	1								
		1	0	0	1	0	1	1	0	1	0	0	0	1	1	0

E. ALGORITHM A

Let us create a modified complement code Xdm = = 11.0011 of the number X=-00.1101 and L1X=01.101 that is a shifted number on one digit to the left of the number X=0.1101. Using Table 1 for the multiplier Y=0.1101111 we get the converted multiplier: Y_p=01. $0\bar{1}$ 10 $0\bar{1}$ 10. To fulfill the multiplication we use Table 7, wherein the columns P_{sign} there are entered significant digits, and in the columns marked by the symbol ↓ there are entered the number 1 of transposition, which are being neglected.

F. ALGORITHM B

According to Table 3 for the multiplier Y=0.1101111 the converted multiplier is Y_p=01. 0 $\bar{1}$ 10 0 $\bar{1}$ 10. While finding a solution for the specified product, we use the numbers: RX=00.000000001101 – for the preservation of X; L1RX=00.000000011010 – for the preservation of X shifted one digit to the left; RXdm=11.111111110011 – for the preservation of the complement code; L1RXdm= 11.111111100110 – for the preservation of the complement code shifted one digit to the left. For multiplication, we use Table 8.

G. ALGORITHM C

While implementing the accelerated multiplication we use the following numbers: RX=001101 – for the preservation of X; RXdm=110011 – for preservation of the complement code and the table of conversion of the multiplier into extended binary code.

Table 8

Implementation of the accelerated multiplication according to the algorithm B

Y_p	Remarks	↓	P_{sign}	S												
`	PS		0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	+RX		0	0	0	0	0	0	0	0	0	0	1	1	0	1
	S		0	0	0	0	0	0	0	0	0	0	1	1	0	1
	←L2S		0	0	0	0	0	0	0	0	1	1	0	1	0	0
$0\bar{1}$	+RXdm		1	1	1	1	1	1	1	1	1	1	0	0	1	1
	S	1	0	0	0	0	0	0	0	1	0	0	1	1	1
	←L2S		0	0	0	0	0	1	0	0	1	1	1	0	0
10	+L1RX		0	0	0	0	0	0	0	0	1	1	0	1	0
	S		0	0	0	0	0	1	0	1	1	0	1	1	0
	←L2S		0	0	0	1	0	1	1	0	1	1	0	0	0
00	←L2S		0	0	1	0	1	1	0	1	1	0	0	0	0	0
$\bar{1}0$	+L1RXdm		1	1	1	1	1	1	1	1	1	0	0	1	1	0
	Z=	1	0	0	1	0	1	1	0	1	0	0	0	1	1	0

Table 9

Implementation of accelerated multiplication according to the algorithm C

Y_p	Remarks	↓	P_{sign}		S											
	PS		0	0	0	0	0	0								
0	R1S→		0	0	0	0	0	0	0							
$\bar{1}$	+RXdm		1	1	0	0	1	1								
	S		1	1	0	0	1	1	0							
0	R1S→		1	1	1	0	0	1	1	0						
0	R1S→		1	1	1	1	0	0	1	1	0					
0	R1S→		1	1	1	1	1	0	0	1	1	0				
0	R1S→		1	1	1	1	1	1	0	0	1	1	0			
$\bar{1}$	+RXdm		1	1	0	0	1	1								
	S	1	1	1	0	0	1	0	0	0	1	1	0			
0	R1S→		1	1	1	0	0	1	0	0	0	1	1	0		
0	R1S→		1	1	1	1	0	0	1	0	0	0	1	1	0	
0	R2S→		1	1	1	1	1	0	0	1	0	0	0	1	1	0
1	+RX		0	0	1	1	0	1								
	S	1	0	0	1	0	1	1	0	1	0	0	0	1	1	0

Table 10

Modification of the algorithm C

Y_p	Remarks	↓	P sign		S											
	PS		0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	+RX		0	0	0	0	0	0	0	0	0	0	1	1	0	1
	S		0	0	0	0	0	0	0	0	0	0	1	1	0	1
00	←L4S		0	0	0	0	0	0	1	1	0	1	0	0	0	0
$\bar{1}0$	L1RXdm		1	1	1	1	1	1	1	1	1	0	0	1	1	0
	S	1	0	0	0	0	0	0	1	0	1	1	0	1	1	0
00	←L4S		0	0	1	0	1	1	0	1	1	0	0	0	0	0
$\bar{1}0$	L1RXdm		1	1	1	1	1	1	1	1	1	0	0	1	1	0
	Z=S	1	0	0	1	0	1	1	0	1	0	0	0	1	1	0

To fulfill multiplication, we use Table 9. **Example 5**. Use the converted multiplier according to the third algorithm of the accelerated multiplication, to find the product of numbers: X=0.1101, Y=0.1101111 by multiplication with the processing of two multiplication digits at a time, starting with the higher-order digits.

Solution. The implementation process of this algorithm is illustrated by Table 10.

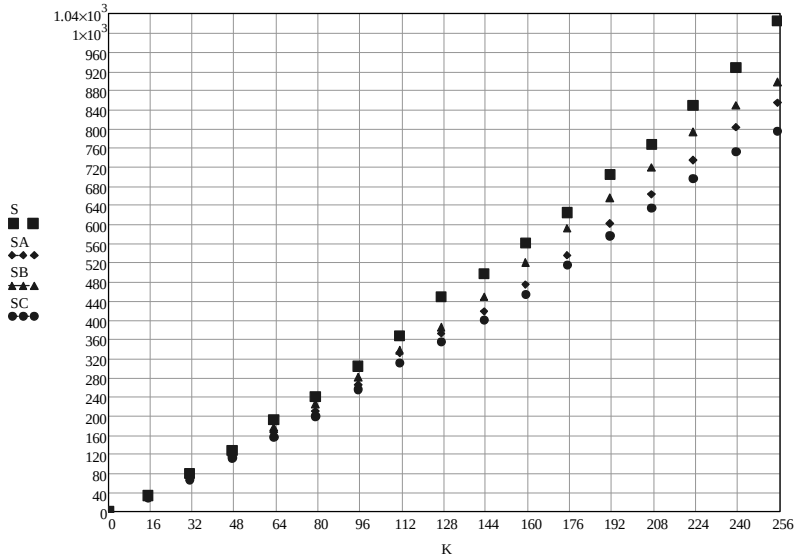

Fig. 1. Comparison of the number of non-zero digits converted by different methods

As it may be seen, the latest implementation of the accelerated multiplication requires the smallest, in comparison with other implementations, quantity of additions and displacements, which results in the reduction of the time spent for the transaction.

IV. COMPARISON OF THE METHODS EFFECTIVENESS

We will calculate the quantity of non-zero digits in the unmodified multiplier and multipliers converted according to other three methods, for all numbers from 1 to 11111111 = 255. The calculation results are shown in Figure 1 with step K = 16, where S is the quantity of non-zero digits in the unconverted multiplier; SA, SB, SC – are the quantities of the non-zero digits in the unconverted multiplier gained as a result of the methods A, B, C correspondingly. In general, all numbers from 1 to 11111111 = 255 contain 1024 non-zero digits, and those converted by methods A, B, C contain 853, 896 and 796 non-zero digits, respectively. It is obvious, the smaller is the quantity of non-zero digits, the less time is spent on the implementation of operations. From Figure 1, it may be deduced that the algorithm C proposed by us in this article is the most effective.

REFERENCES

[1] Melnyk A. O. Computer Architecture, Lutsk, 2008. – 470 p. (in Ukrainian).
[2] Melnyk A. O., Melnyk V. A. Personal computers: architecture, design, application, Lviv, 2013. – 516 p.
[3] Knuth, Donald E. The Art of Computer Programming, 3rd ed. Reading , MA: Addison-Wesley, 1998. – 762 p.
[4] Korniichuk V. I., Tarasenko V. P., Tarasenko-Kliatchenko O. V. Basics of Computer Arithmetic, Kyiv, 2006. – 164 p. (in Ukrainian).
[5] Tsmots I. G. Parallel algorithms and matrix VLSI structures of multiplication devices for real-time computer systems. Infornation Technologies and Systems. Lviv, 2004. Vol. 7. N 1, pp. 5–16.

Ihor I. Korol is a Vice-Rector for Academic Policy and Research at Uzhhorod National University. He received the Master's degree in mathematics and applied mathematics at Lomonosov Moscow State University in 1992. In 1996 he obtained his Ph.D. in Mathematics (Differential Equations), and in 2011 he obtained his D.Sc. degree at Taras Shevchenko National University in Kyiv. Since 1993 to 2011 he has been Senior Lecturer, Associate professor, professor of the Department of Differential Equations at Uzhhorod National University. His work resulted in 87 publications. He has also been a visiting professor at the Institute of Mathematics of Pomeranian University in Słupsk. Currently he is a visiting professor at the Faculty of Mathematics, Informatics and Landscape Architecture of John Paul II Catholic University of Lublin.

Ivan Yu. Korol is an Associate professor of the Department of Computer Systems and Networks at Uzhhorod National University.

He received the Master's degree in applied mathematics at Uzhhorod National University in 1964. In 1973 he obtained his Ph.D. in Computational Mathematics at Ivan Franko National University in Lviv. His work resulted in 82 publications. Since 1965 to 1991 he has been a Senior Lecturer, Associate professor of the Department of Computational Mathematics. Since 1991 to 2018 he has been a Head of the Department of Computer Systems and Networks at Uzhhorod National University.