Date of publication MAR-29, 2023, date of current version OCT-13, 2022.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.22.1.2875

Algorithm for Calculation the Carry and Borrow Signs in Multi-digit Operations in the Parallel Computational Model

ANDRII TERESHCHENKO[1], VALERIY ZADIRAKA[2]

[1]Department of Doctoral Studies V. M. Glushkov Institute of Cybernetics of the National Academy of Sciences, Kyiv, Ukraine (e-mail: teramidi@ukr.net)
[2]Department of Optimization of Numerical Methods V. M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine, Kyiv, Ukraine (e-mail: zvk140@ukr.net)

Corresponding author: Andrii Tereshchenko (e-mail: teramidi@ukr.net).

ABSTRACT The fast algorithm to calculate carry signs and borrow signs for implementation of fast multi-digit operations in the parallel computational model is proposed. The proposed algorithm also makes it possible to predict carry signs in the case of an addition operation and predict borrow signs for a subtraction operation. It is shown how the sign prediction algorithm is implemented in operations in which each parallel processor proceeds the separate group of words into which multi-digit numbers are divided. The iterative calculations of carry signs of grouped words are described. The sign calculation algorithm as component of new modifications of multi-digit addition, subtraction, comparison, the sum of three or more numbers in the parallel computational model is presented. The sign calculation algorithm provides general approach to the implementation of multiplication, division, multiplication by modulo, exponentiation by modulo in the parallel computational model. In the form of a table, a general analysis of the complexity of algorithms and an analysis of the complexity by the number of single-word operations per processor are given.

KEYWORDS multi-digit arithmetic; multi-digit addition; multi-digit subtraction; multi-digit comparison; carry sign; borrow sign; parallel computational model.

I. INTRODUCTION

THE need to solve problems with transcomputational complexity, when mathematical models with million unknowns are calculated, determines the use of modern parallel computing systems. It expands the use of arithmetic with large numbers [1–7]. In the process of implementation, the operation of multiplication by modulo [8] is very dependent on the speed of the addition operation, compared with modulo value, and subtracted in case of excess. The speed of multi-digit multiplication operation [9, 10] determines the speed of cryptographic hardware and software [11–16]. In the sequential computational model, carry and borrow signs are proceeded by the processor automatically. In the parallel computational model, the execution time of addition [17], subtraction, and comparison operations depends on the methods that account for the carry and borrow signs that occur in the corresponding digits (machine words) of a multi-digit operation. Performing multi-digit operations of addition [18], subtraction, comparison, the sum of two or more multi-digit numbers is a "bottleneck" in case of implementing methods and algorithms in the parallel computational model [19, 20].

In 1971 Schmookler and A. Weinberger described *the generation* and *propagation* functions considering the eight-digit adders [21]. Adding the function *kill* C. McGeoch described "carry look-ahead" method to add two *N-bit* integers and "carry save" method to add three integers with logarithmic complexity of executing operation on the circuit [22].

In 1990 R. Floyd and D. Knuth proposed the arithmetic [23] dealing with the operations with the speed of addition operation. The implementation of mathematical operations uses "Addition Machines".

Using interference adders, A.V. Anissimov proposed the method [24] of addition of positive and negative integers of N bits length in $N+K$ steps using K interference transformations. The method is based on the probability analysis of the carry sign appearance.

The methods [21–24] are designed to operate on bits. These methods are efficiently implemented in hardware. Software methods need at least proceeding with words to be efficient.

This work presents the algorithm for calculating the carry signs and the borrow signs, shows the method of using the

proposed algorithm in the case of multi-digit operations of addition, subtraction, comparison, the sum of three or more multi-digit numbers. Defining the algorithm as a separate part allows analyzing the efficiency of individual parts, which affect the overall efficiency, and which are critical for synchronization between processors.

II. PROBLEM STATEMENT AND DEFINITIONS

X, Y, $X(a)$, $a=\overline{0,h-1}$, – bitwise positive integers. Let A be qw-bitwise sum of numbers $X+Y$ excluding carry sign, S be qw-bitwise result of subtraction of $X-Y$ excluding borrow sign, U be $qw\cdot\lceil\log_2 h\rceil$–bitwise sum of values $X(a)$, $a=\overline{0,h-1}$. The sequence $\{X_{q-1},...,X_0\}$ of elements $X_j=\sum_{b=0}^{w-1}(x_{jw+b}\cdot 2^b)$, $j=\overline{0,q-1}$, describes big integer $X=\sum_{j=0}^{q-1}(X_j\cdot 2^{w\cdot j})$, where $X_j=\{x_{jw+w-1},...,x_{jw}\}$, $j=\overline{0,q-1}$, are w-bit words. The sequences $\{Y_{q-1},...,Y_0\}$, $\{A_{q-1},...,A_0\}$, $\{S_{q-1},...,S_0\}$, $\{U_{q-1},...,U_0\}$, $\{X_{q-1}(a),...,X_0(a)\}$, $a=\overline{0,h-1}$, describe big integers Y, A, S, U, $X(a)$, $a=\overline{0,h-1}$,

If one word consists of w bits, then let the following big integers be called q-word numbers: Y, A, S, U, $X(a)$, $a=\overline{0,h-1}$. Accordingly, the number U will be q+1-word number in case of $h<2^w$.

Using the fast algorithm for calculating carry and borrow signs, it needs to build fast algorithms for calculating addition $X+Y$, subtraction $X-Y$, comparison $X<Y$, sum $U=\sum_{a=0}^{h-1}X(a)$ in the parallel computational model.

III. CALCULATION CARRY SIGNS IN THE GROUP OF WORDS IN THE PARALLEL COMPUTATIONAL MODEL

A. PRE-CALCULATED NUMBERS M, T, C

Let us introduce $M=\{M_{q-1},...,M_0\}$, where $W=2^w$, $M_j=\langle X_j+Y_j\rangle_W$, $j=\overline{0,q-1}$, to get the addition $X+Y$ of big integers of the length of q words omitting the carry sign between the words.

Next, let us section the big integers X, Y into groups of the length of w words. There will be indices $g-1$, g, $g+1$ to refer to the groups of the length of w words with indices $g-1$, g, $g+1$. For example, the following sequences $\{X_{gw+w-1},...,X_{gw}\}$, $\{Y_{gw+w-1},...,Y_{gw}\}$ determine groups with the index g. For describing the statuses of the group with index g, there are words T_g and C_g of the length of w bits. The phrase "with index" will be omitted for simplicity. T and C of the length of $k=q/w$ words describe the whole status of big integers X, Y.

The number $T=\{T_{k-1},...,T_0\}$ is calculated in the following way:

$$T_g=\sum_{b=0}^{w-1}(t_j\cdot 2^b),\ t_j=\begin{Bmatrix}1,\ if\ M_j=W-1\\0,\ if\ M_j\neq W-1\end{Bmatrix},$$
$$j=gw+b,\ g=\overline{0,k-1},\ k=q/w,\ W=2^w. \quad (1)$$

The number $C=\{C_{k-1},...,C_0\}$ is calculated as follows:

$$C_g=\sum_{b=\begin{Bmatrix}0,\ if\ g>0\\1,\ if\ g=0\end{Bmatrix}}^{w-1}(c_j\cdot 2^b),\ c_j=\begin{Bmatrix}1,\ if\ X_{j-1}+Y_{j-1}\geq W\\0,\ if\ X_{j-1}+Y_{j-1}<W\end{Bmatrix},$$
$$j=gw+b,\ g=\overline{0,k-1},\ k=q/w,\ c_0=0. \quad (2)$$

We will call the carry sign "incoming" in the case of the carry sign "comes" from the previous word and it needs to be considered in the present word. We will call "outgoing" carry sign in the case of transferring the sign to the next word. The "outgoing" carry sign will be considered in the next word.

B. CARRY SIGN ANALYSIS ("PREDICTION") OF THE GROUP OF THE LENGTH OF m WORDS

Let us section into groups of the length of w words the operation of big integers $X+Y$ of the length of q words and obtain $T=\{T_{k-1},...,T_0\}$, $C=\{C_{k-1},...,C_0\}$, $k=q/w$, by formulas (1), (2). The following analysis of T and C helps to "look-ahead" carry signs that occur between groups.

Lemma 1. If C_g and T_g describe the group g of the length of w words $\{X_{gw+w-1},...,X_{gw}\}+\{Y_{gw+w-1},...,Y_{gw}\}$, in accordance with (1), (2), and:

$C_g+T_g\geq W$, $W=2^w$, then the group g "generates" the carry sign to the next group;

$T_g=W-1$, $W=2^w$, then the group g transfers ("propagates") the carry sign to the next group;

$C_g+T_g<W$ and $T_g\neq W-1$, $W=2^w$, then the group g never transfers ("kills") the carry sign to the next.

Let us consider the example shown in Fig. 1. C_g consists of $w=16$ bits and the most significant bit is equal to 1. It means that $X_{gw+w-2}+Y_{gw+w-2}\geq W$. The rest of the bits are not considered. If $w=16$, then $X_{g16+14}+Y_{g16+14}\geq W$.

	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	C_g
	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	T_g
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

Figure 1. Example of "generating" the carry sign by the group g to the next group $g+1$, $C_g + T_g \geq W$

T_g has the highest bit equal to 1 in Fig. 1. It means that $X_{gw+w-1} + Y_{gw+w-1} = W - 1$. If $w = 16$, then $X_{g16+15} + Y_{g16+15} = W - 1$.

C. ITERATIVE CALCULATION OF CARRY SIGNS

Let us consider the example shown in Fig. 2. The value of C_g equal to 1 means that the carry sign comes to the group g from the group $g-1$. The value of T_g equal to max value of the word $W-1$ means that each element of the result of $\{X_{gw+w-1},...,X_{gw}\}+\{Y_{gw+w-1},...,Y_{gw}\}$ in accordance to formula (1) is equal to $W-1$. The condition of values C_g and T_g describes the case of "propagation" of the carry sign.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	C_g
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	T_g
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	C_g

Figure 2. Example of iterative calculation "incoming" carry signs of C_g (in the bottom) based on C_g (on the top)

The simple arithmetic operation $C_g + T_g$ "propagates" the carry sign to the next group of words but it keeps zeros in C_g. Fig. 2 also shows one step of "propagation" of carry sign inside the group of the length of $w = 16$ words. It needs bitwise operation to calculate "incoming" carry signs for every word in the group of words. It is proposed the iterative formula to get carry signs for each word one by one:

$$C_g \leftarrow C_g \vee \left\langle (C_g \wedge T_g) << 1 \right\rangle_W, \; W = 2^w,$$

where "$<<1$" is left shift bits (toward the highest bit) operation. It needs w iterations to calculate all carry signs inside the group of the length of w words.

IV. PARALLEL ADDITION AND SUBTRACTION

A. PARALLEL CALCULATION OF THE CARRY AND BORROW SIGNS

Let us first consider the algorithm that is used as part of further algorithms for implementing multi-digit operations.

Algorithm 1. Calculation of carry and borrow signs of the result of operations between big integers of the length of w words based on "prediction" of the carry and borrow signs between groups.

Params: Values q, w, k, $W = 2^w$, V, $M = \{M_{q-1},...,M_0\}$, $C = \{C_k,...,C_0\}$.

Output: Updated $C = \{C_k,...,C_0\}$.

Step 1. $T = \{T_{k-1},...,T_0\}$, where $T_g = \sum_{b=0}^{w-1}(t_j \cdot 2^b)$,

$$t_j = \begin{cases} 1, if\ M_j = V \\ 0, if\ M_j \neq V \end{cases}, \; j = gw + b, \; g = \overline{0, q/w - 1}.$$

// "Look-ahead" of the carry and borrow signs

Step 2. For g from 0 to $g < q/w - 1$

Step 3. If $C_g + T_g \geq W$, then $C_{g+1} \leftarrow C_{g+1} \vee 1$

Step 4. End for g.

// Iterative calculation carry and borrow signs

Step 5. For g from to $g < k - 1$

Step 6. For r from 0 to $r < w - 1$

Step 7. $C_g \leftarrow C_g \vee \left\langle (C_g \wedge T_g) << 1 \right\rangle_W$.

Step 8. End for r.

Step 9. End for g.

The feature of Algorithm 1 is that the vector $C = \{C_k,...,C_0\}$ is updated and returned. The input parameter k looks redundant as $k = q/w$, but in the case of implementing multi-digit comparison operation, this parameter will be zero to exclude the execution of the part that corrects the word signs (steps 5–9), and in this way greatly increases the speed of the multi-digit comparison operation of two numbers comparing with the execution time of multi-digit addition and subtraction operations. When the comparison operation of numbers $X < Y$ precedes its subtraction $X - Y$, the parameter k will be non-zero.

B. PARALLEL ADDITION ALGORITHM

Algorithm 2. Calculation of the addition of big numbers $X + Y$ of length of q words with "prediction" of carry signs.

Params: Values q, w, $k = q/w$, $W = 2^w$, $X = \{X_{q-1},...,X_0\}$, $Y = \{Y_{q-1},...,Y_0\}$.

Output: $A = \{A_{q-1},...,A_0\}$; $c = 1$, if $X + Y \geq 2^{qw}$.

Step 1. $M = \{M_{q-1},...,M_0\}$, where

$$M_j = \left\langle X_j + Y_j \right\rangle_W, \; j = \overline{0, q-1}.$$

Step 2. $C = \{0, C_{k-1},...,C_0\}$, where

$$C_g = \sum_{b=\begin{Bmatrix} 0, if\ g>0 \\ 1, if\ g=0 \end{Bmatrix}}^{w-1}(c_j \cdot 2^b), c_j = \begin{cases} 1, if\ X_{j-1} + Y_{j-1} \geq W \\ 0, if\ X_{j-1} + Y_{j-1} < W \end{cases},$$

$$j = gw + b, \; g = \overline{0, k-1}.$$

// "Look-ahead" of the carry signs

Step 3. $C \leftarrow$ Algorithm1$(q, \; w, \; k = q/w, \; W, \; V = W - 1, \; M = \{M_{q-1},...,M_0\}, \; C = \{C_k,...,C_0\})$.

// Calculate the result

Step 4. For g from 0 to $g < k-1$

Step 5. $c_{temp} \leftarrow C_g$.

Step 6. For r from 0 to $r < w-1$

Step 7. $A_{gw+r} \leftarrow \left\langle M_{gw+r} + \left\langle c_{temp} \right\rangle_2 \right\rangle_W$.

Step 8. $c_{temp} \leftarrow c_{temp} >> 1$.

Step 9. End for r.

Step 10. End for g.

Step 11. $c \leftarrow C_k$.

Steps 1, 2, 3, 4–11 are parallelized in case of distributing the calculation.

C. COMPLEXITY OF ADDITION ALGORITHM

Let us first consider the lemma 2 before analyzing the complexity of Algorithm 1.

Lemma 2 [25]. In case of $X_{j-1} < W$, $Y_{j-1} < W$, $X_{j-1} + Y_{j-1} \geq W$, where $W = 2^w$, it is correct that, $(XY < X_{j-1}) \vee (XY < Y_{j-1})$, $XY = \left\langle X_{j-1} + Y_{j-1} \right\rangle_W$.

Comment. The expression $X_{j-1} + Y_{j-1} \geq W$ needs two-word addition and the following two-word comparison. Lemma 2 allows replacing with four single operations: addition, two comparisons, and one logical "or" operation.

Theorem 1. In Algorithms 1 and 2, the number of single operations has the form $O_{1,2}(q) = 14q + 6k$, where big integers consist of $q = kw$ words, the word consists of w bits.

Proof. Assume that single operation (bitwise, addition, comparison) is executed equally in terms of CPU time. To find T in step 1 of Algorithm 1, it is needed to execute q operations of comparison $M_j = V$ and q bitwise operations $t_j \cdot 2^b$. In step 1 of Algorithm 1, $2q$ operations must be performed. In step 3 of Algorithm 1, it is needed to perform $C_g + T_g \geq W$ operation on two words to avoid overflow of the result. According to Lemma 2, this operation is simplified with $(CT < C_g) \vee (CT < T_g)$, where $CT = \left\langle C_g + T_g \right\rangle_W$, the calculation of which needs four single operations. Step 3 of Algorithm 1 requires $5k$ operations to take into account $C_{g+1} \leftarrow C_{g+1} \vee 1$,. Operations $3kw = 3q$ are required to perform the loop in step 7 of Algorithm 1. The modulo calculation $M_j = \left\langle X_j + Y_j \right\rangle_W$, $W = 2^w$, in bits matches the machine word. So, the step 1 of Algorithm 2 involves q operations. To find C in step 2 of Algorithm 2, considering Lemma 2 and similarly to step 3 of Algorithm 1, it is needed $5q$ operations. Step 5 of Algorithm 2 is performed k times. To calculate steps 7 and 8 of Algorithm 2, $3kw = 3q$ single operations are executed.

The number of single-word operations for performing algorithms is $2q + 5k + 3q + q + 5q + k + 3q = 14q + 6k$.

Theorem 2. In Algorithms 1 and 2, the number of single operations performed by one processor in case of distribution of calculations among k processors is as follows: $O_{1,2}^{processor}(q) = 14w + 5k + 1$, where $q = kw$ – the length of big integers in words, the word consists of w bits.

Proof. Distributing the calculation among k processors, each processor will perform its operations with an index (variable) g, which eliminates loops at all algorithm steps. Step 1 of Algorithm 2 does not have an index g, but the computations can also be evenly distributed among all processors, so the number of operations performed by one processor will be k times less than the total number of all single-word operations at each step. The exception is step 3 of Algorithm 1, which must be performed as many times as the number of processors used to "propagate" the carry sign from the group with the zero index to the group with the largest index. The step 3 corresponds to the synchronization of signs between processors, each of which processes its own group of words. Therefore, $5k$ is kept in the expression[1]

$$2q/k + (5k) + 3q/k + q/k + 5q/k + \\ + k/k + 3q/k = 14w + 5k + 1$$

D. PARALLEL SUBTRACTION ALGORITHM

Algorithm 3. Calculation of subtraction of big integers $X - Y$ of the length of q words with "prediction" of borrow signs.

Params: Values q, w, $k = q/w$, $W = 2^w$, $X = \{X_{q-1},...,X_0\}$, $Y = \{Y_{q-1},...,Y_0\}$.

Output: $c = 1$, if $X < Y$; $S = \{S_{q-1},...,S_0\}$.

Step 1. $M = \{M_{q-1},...,M_0\}$, where

$M_j = \left\langle X_j - Y_j \right\rangle_W$, $j = \overline{0, q-1}$.

Step 2. $C = \{0, C_{k-1},...,C_0\}$, where

$$C_g = \sum_{b=\left\{\substack{0,\ if\ g>0 \\ 1,\ if\ g=0}\right\}}^{w-1} (c_j \cdot 2^b),\ c_j = \begin{cases} 1, if\ X_{j-1} < Y_{j-1} \\ 0, if\ X_{j-1} \geq Y_{j-1} \end{cases},$$

$j = gw + b$, $g = \overline{0, k-1}$, $k = q/w$.

// "Prediction" of the borrow signs

Step 3. $C \leftarrow$ Algorithm1($q, w, k = q/w$, W, $V = 0$, $M = \{M_{q-1},...,M_0\}$, $C = \{C_k,...,C_0\}$).

// Calculate the result

Step 4. For g from 0 to $g < k-1$

Step 5. $c_{temp} \leftarrow C_g$.

[1] The number of operations marked in parentheses are not parallelized.

Step 6. For r from 0 to $r < w-1$

Step 7. $S_{gw+r} \leftarrow \left\langle M_{gw+r} - \left\langle c_{temp} \right\rangle_2 \right\rangle_W$.

Step 8. $c_{temp} \leftarrow c_{temp} >> 1$.

Step 9. End for r.

Step 10. End for g.

Step 11. $c \leftarrow C_k$.

Algorithms 2 and 3 are almost the same. Small differences are in step 1 to calculate M_j ($M_j = \left\langle X_j + Y_j \right\rangle_W$, $M_j = \left\langle X_j - Y_j \right\rangle_W$), in step 2 for c_j ($c_j = \begin{cases} 1, if\ X_{j-1} + Y_{j-1} \ge W \\ 0, if\ X_{j-1} + Y_{j-1} < W \end{cases}$, $c_j = \begin{cases} 1, if\ X_{j-1} < Y_{j-1} \\ 0, if\ X_{j-1} \ge Y_{j-1} \end{cases}$), in step 3 different parameters ($V = W-1, V = 0$), in step 7 ($A_{gw+r} \leftarrow \left\langle M_{gw+r} + \left\langle c_{temp} \right\rangle_2 \right\rangle_W$, $S_{gw+r} \leftarrow \left\langle M_{gw+r} - \left\langle c_{temp} \right\rangle_2 \right\rangle_W$).

C. COMPLEXITY OF SUBTRACTION ALGORITHM

Theorem 3. In Algorithms 1 and 3, the number of single operations is as follows: $O_{1,3}(q) = 11q + 6k$, where big integers consist of $q = kw$ words, the word consists of w bits.

Theorem 4. In Algorithms 1 and 3, the number of single operations performed by one processor when distributing calculations among k processors looks as follows: $O_{1,3}^{processor}(q) = 11w + 5k + 1$, where $q = kw$ – the length of big integers in words, the word consists of w bits.

The proofs of theorems 3 and 4 take place by analogy with theorems 1 and 2, considering that the calculation c_j requires one-word comparison operation in step 2 of Algorithm 3.

V. PARALLEL COMPARISON

The comparison operation is a slightly more complicated operation than the subtraction operation, since two subtraction operations must be performed to determine the smaller number from the numbers X and Y. If $X - Y < 0$, then $X < Y$, otherwise, if $Y - X < 0$, then $Y < X$, otherwise X and Y are the same. The equivalent of the comparison operation is the operation that returns three values: –1, 0, +1 for cases $X < Y$, $X = Y$, $X > Y$. The comparison can be simplified. For this, one subtraction operation to determine the occurrence of a borrow sign can be replaced with an operation of checking the equivalence of numbers.

A. PARALLEL COMPARISION ALGORITHM

Algorithm 4. Comparison of big integers X, Y, which consist of q words, with "look-ahead" of borrow signs.

Params: Values $X = \{X_{q-1},...,X_0\}$, $Y = \{Y_{q-1},...,Y_0\}$, $k = q/w$, $W = 2^w$; $EQ_g = W-1$, $C_g = 0$, $g = \overline{0,k}$.

Output: $CF = 1$, if $X < Y$; $EQ = 1$, if $X = Y$.

Step 1. $M = \{M_{q-1},...,M_0\}$, where $M_j = \left\langle X_j - Y_j \right\rangle_W$, $j = \overline{0,q-1}$.

Step 2. $C = \{C_{k-1},...,C_0\}$, $C_g = \sum_{b=\left\{\begin{smallmatrix} 0, if\ g>0 \\ 1, if\ g=0 \end{smallmatrix}\right\}}^{w-1} (c_j \cdot 2^b)$,

$c_j = \begin{cases} 1, if\ X_{j-1} < Y_{j-1} \\ 0, if\ X_{j-1} \ge Y_{j-1} \end{cases}$, $j = gw + b$, $g = \overline{0,k-1}$.

// "Look-ahead" of borrow signs

Step 3. $C \leftarrow$ Algorithms1(q, w, $k = 0$, W, $V = 0$, $M = \{M_{q-1},...,M_0\}$, $C = \{C_k,...,C_0\}$).

// "Look-ahead" of the equivalent of groups of words

Step 4. For g from 0 to $g < k$

Step 5. $EQ_{g+1} \leftarrow EQ_{g+1} \wedge EQ_g \wedge T_g$.

Step 6. End for g.

Step7. $CF = \begin{cases} 1, if\ C_k = 1 \\ 0, if\ C_k = 0 \end{cases}$, $EQ = \begin{cases} 1, if EQ_k = W-1 \\ 0, if EQ_k \ne 0 \end{cases}$

Although $T_g = \sum_{b=0}^{w-1} (t_j \cdot 2^b)$ defined and computed in the middle of Algorithm 1, for simplicity we assume that Algorithms 1 and 4 share the same memory space. Note that in step 3 of Algorithm 4, there is a parameter $k = 0$ to exclude the loop of correcting the word-wise carry signs (steps 5–9 of Algorithm 1), which is the reserve of optimization.

B. COMPLEXITY OF COMPARISION ALGORITHM

Theorem 5. In Algorithms 1 and 4, the number of single operations looks as follows: $O_{1,4}(q) = 8q + 7k + 2$, where $q = kw$ – the length of big integers in words, the word consists of w.

Proof. Assume that single operation (bitwise, addition, comparison) is executed equally in terms of CPU time. By analogy with theorem 1, steps 1–4 of algorithm take $5k$ single-word operations. Steps 5–9 are not performed because the parameter $k = 0$. Step 1 of Algorithm 4 requires q operations. The modulo calculation operations $M_j = \left\langle X_j + Y_j \right\rangle_W$, $W = 2^w$, in bits matches the machine word. To find C in step 2 of Algorithm 4, q comparison operations and q bitwise operations are necessary. Step 5 of Algorithm 4 is performed k times and requires $2k$ operations. Step 7 requires 2 one-word operations to calculate.

The total number of single-word operations for performing algorithms is $2q + 5k + q + 2q + 2k + 2 = 5q + 7k + 2$.

Theorem 6. In Algorithms 1 and 4, the number of single operations performed by one processor in case of distribution of calculations among k processors is as follows: $O_{1,4}^{процесор}(q) = 5w + 5k + 4$, where $q = kw$ – the length of big integers in words, the word consists of w bits.

Proof. Calculating on k processors, step 1 of Algorithm 1 and steps 1 and 2 of Algorithm 4 can be evenly distributed among all processors, so the number of operations performed by one processor will be k times less. Step 3 of Algorithm 1 and step 5 of Algorithm 4 need to be performed as many times as there are processors involved to "propagate" the borrow sign from the group with the zero index to the group with the largest index, which corresponds to the synchronization of signs between the processors, each of which processes its own group of words. Therefore, $5k$ is kept. Step 7 of Algorithm 4 cannot be parallelized, so term 2 is also kept in the expression[2]

$$2q/k+(5k)+q/k+2q/k+2k/k+(2)= \\ =5w+5k+4$$

VI. PARALLEL SUM OF THREE OR MORE MULTI-DIGIT NUMBERS

Let us find the sum G of h big integers $G_j=\sum_{a=0}^{h-1}X_j(a)$, $j=\overline{0,q-1}$, $h<2^w$. Considering the carry sign of addition operation each element of G_j, $j=\overline{0,q-1}$, is longer than one word. Let us try to build the calculation to stick to formulas (1) and (2) as much as possible. For this purpose, we introduce q+1-word big integer H, where $H_{j+1}\leftarrow\lfloor G_j/W\rfloor$, $W=2^w$, $j=\overline{q-1,0}$, $H_0\leftarrow 0$, and then $M=\{M_{q-1},...,M_0\}$ is based on new G, H, $M_j=\left\langle\left\langle G_j\right\rangle_W+H_j\right\rangle_W$, $W=2^w$, $j=\overline{0,q-1}$.

The calculation $T=\{T_{k-1},...,T_0\}$ is not changed, where $T_g=\sum_{b=0}^{w-1}(t_j\cdot 2^b)$, $t_j=\left\{\begin{matrix}1,\ if\ M_j=W-1\\0,\ if\ M_j\neq W-1\end{matrix}\right\}$, $j=gw+b$, $g=\overline{0,k-1}$, $k=q/w$, $W=2^w$.

The calculation $C=\{C_{k-1},...,C_0\}$ is a little bit different in the conditions:

$$C_g=\sum_{b=\left\{\begin{matrix}0,\ if\ g>0\\1,\ if\ g=0\end{matrix}\right\}}^{w-1}(c_j\cdot 2^b),\ c_j=\left\{\begin{matrix}1,\ if\ H_{j-1}+\left\langle G_{j-1}\right\rangle_W\geq W\\0,\ if\ H_{j-1}+\left\langle G_{j-1}\right\rangle_W<W\end{matrix}\right.$$

$$j=gw+b,\ g=\overline{0,k-1},\ k=q/w,\ c_0=0. \quad (3)$$

A. PARALLEL ALGORITHM TO SUM BIG INTEGERS

By analogy to the "method with the saving of carry signs" [22], which operates on bits, there is algorithm 5, which operates on words.

Algorithm 5. Calculating the sum of h big integers $X(a)$, $a=\overline{0,h-1}$, of the length of q words with "saving" and "predicting" carry signs.

Params: Values q, w, h, $k=q/w$, $W=2^w$, $X(a)$, $a=\overline{0,h-1}$.

Output: $U=\{U_q,...,U_0\}$.

Step 1. $G=\{G_{q-1},...,G_0\}$, where $G_j=\sum_{a=0}^{h-1}X_j(a)$, $j=\overline{0,q-1}$.

Step 2. $H=\{H_q,...,H_1,0\}$, where $H_{j+1}\leftarrow\lfloor G_j/W\rfloor$, $j=\overline{0,q-1}$.

Step 3. $M=\{M_{q-1},...,M_0\}$, where $M_j=\left\langle G_j+H_j\right\rangle_W$, $j=\overline{0,q-1}$.

Step 5. $C=\{0,C_{k-1},...,C_0\}$, where

$$C_g=\sum_{b=\left\{\begin{matrix}0,\ if\ g>0\\1,\ if\ g=0\end{matrix}\right\}}^{w-1}(c_j\cdot 2^b),\ c_j=\left\{\begin{matrix}1,\ if\ H_{j-1}+\left\langle G_{j-1}\right\rangle_W\geq W\\0,\ \text{if}\ H_{j-1}+\left\langle G_{j-1}\right\rangle_W<W\end{matrix}\right.$$

$j=gw+b$, $g=\overline{0,k-1}$, $k=q/w$.

// "Look-ahead" of the carry signs

Step 6. $C\leftarrow$Algorithm1(q, w, $k=q/w$, W, $V=W-1$, $M=\{M_{q-1},...,M_0\}$, $C=\{C_k,...,C_0\}$).

// Calculation of the result

Step 7. $U_n\leftarrow H_n+C_k$.

Step 8. For g from 0 to $g<k-1$

Step 9. $c_{\text{temp}}\leftarrow C_g$.

Step 10. For r from 0 to $r<w-1$

Step 11. $U_{gw+r}\leftarrow\left\langle M_{gw+r}+\left\langle c_{\text{temp}}\right\rangle_2\right\rangle_W$.

Step 12. $c_{\text{temp}}\leftarrow c_{\text{temp}}>>1$.

Step 13. End for r.

Step 14. End for g.

B. COMPLEXITY OF THE ALGORITHM OF THE SUM OF NUMBERS

Theorem 7. In Algorithms 1 and 5, the number of single operations looks as follows: $O_{1,5}(q)=2qh+14q+6k+1$, where big integers consist of $q=kw$ words, h – the number of big integers, the word consists of w bits.

Proof. Assume that single operation (bitwise, addition, comparison) is executed equally in terms of CPU time. Algorithm 5 differs from Algorithm 2 in the additional steps 1 and 2, in which additional numbers H and G are calculated to save the carry signs. It needs $2qh$ operations in step 1 to

[2] The number of operations marked in parentheses are not parallelized.

"save" all carry signs. In step 2, the operation $\lfloor G_j / W \rfloor$ extracts the high word, and in step 3, the modulo operation $M_j = \langle X_j + Y_j \rangle_W$, $W = 2^w$, in bits matches the machine word, so step 2 does not require arithmetic operations, and step 3 requires q single-word additions. Finally, there is the following expression:

$$2q + 5k + 3q + 2qh + q + 5q + 1 + k + 3q = \\ = 2qh + 14q + 6k + 1$$

Theorem 8. In Algorithms 1 and 5, the number of single operations performed by one processor during the distribution of calculations among k processors looks as follows: $O_{1,5}^{processor}(q) = 2wh + 14w + 5k + 2$, where h – the number of big integers, big integers consist of $q = kw$ words, the word consists of w bits.

Proof. Calculating on k processors, step 1 of Algorithm 1 and steps 1–5 of Algorithm 5 can be evenly distributed among all processors, so the number of operations performed by one processor will be k times less. Step 3 of Algorithm 1 must be performed as many times as there are processors involved in "propagating" the carry signs from the group with the zero index to the group with the largest index, which corresponds to the synchronization of signs between the processors, each of which processes its own group of words. Therefore, $5k$ is kept. Step 7 of Algorithm 5 cannot be parallelized, so term 1 is also kept in the expression

$$2q/k + (5k) + 3q/k + \\ + 2qh/k + q/k + 5q/k + (1) + k/k + 3q/k = \\ = 2wh + 14w + 5k + 2$$

VII. IMPELEMENTATIONS

Using the sign calculation algorithm, the described algorithms of addition, subtraction, comparison, the sum of three or more multi-digit numbers are included in the mathematical library which is developed and used in the Department of optimization of numerical methods of V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine. The described algorithms are used as components of multi-digit operations of multiplication, multiplication by modulo, exponentiation by modulo implemented in the parallel computational model.

The mathematical library is developed using technologies C# (user interface), C++ (low level calculations), OpenCL (GPU level) and implemented in Security Service of Ukraine.

VIII. CONCLUSIONS

This paper presents the fast algorithm to calculate carry (or borrow) signs in the parallel computational model. As shown the algorithm could be used to build fast multi-digit operations of addition, subtraction, comparison, the sum of three or more multi-digit numbers in parallel computational model. The paper analyzes the overall complexity of modified multi-digit operations [25] and complexity in the parallel computational model for one processor that uses vector operations of length 16. The analysis of parts that can be parallelized is studied in the form of theorems 2, 4, 6, 8. The summary analysis of the complexity is provided in the form of a table. Complexity analysis has shown that in the case when the number of involved processors increases proportionally to the length of number, then the number of single-word operations performed by a single parallel processor is almost unchanged. The algorithm allows building more complicated multi-digit operations, such as multiplication, division, multiplication by modulo, exponentiation by modulo in the parallel computational model.

Table 1. Priori estimates of the overall complexity based on the number of single-word operations when implementing multi-digit operations in the parallel computational model, where big integers consist of $q = kw$ words, h – the number of big integers, the word consists of w bits

Multi-digit operation	Overall complexity	Number of operations for one processor	The number of operations for one processor using vector operations of the length $w = 16$
Addition	14*q*+6*k*	14*w*+5*k*+1	5*k*+225
Substraction	11*q*+6*k*	11*w*+5*k*+1	5*k*+177
Comparison	5*q*+7*k*+2	5*w*+5*k*+4	5*k*+84
Sum of numbers	2*qh*+14*q*+6*k*+1	2*wh*+14w+5*k*+2	5*k*+226+32*h*

References

[1] R. K. Richards, *Arithmetic Operations in Digital Computers*. New York: Van Nostrand, 1955.

[2] O. L. MacSorley, "High-speed arithmetic in binary computers," *Proc. IRE*, vol. 49, Jan., pp. 67–91, 1961. https://doi.org/10.1109/JRPROC.1961.287779.

[3] M. Flynn, S. Oberman, *Advanced Computer Arithmetic Design*. Wiley, 2001, 344 p.

[4] J. E. Robertson, *Theory of Computer Arithmetic Employed in the Design of the New Computer at the University of Illinois*, Urbana: Digital Computer Lab., University of Illinois, June 1960.

[5] I. Koren, *Computer Arithmetic Algorithms*, AK Peters Ltd, 2001, 296 p.

[6] V. Kumar, et al., "A unified architecture for BCD and binary adder/subtractor," *Proceedings of the 14th Euromicro Conference on Digital System Design. Architectures, Methods and Tools. (DSD 2011)*, Oulu, pp. 426–429, 2011. https://doi.org/10.1109/DSD.2011.58.

[7] V. S. Knyazkov, K. S. Isupov, "Parallell multiple-precision arithmetic based on residue number system," *Program Syst. Theor. Appl.*, vol. 7, issue 28, pp. 61–97, 2016. https://doi.org/10.25209/2079-3316-2016-7-1-61-97.

[8] P. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 4, no. 170, pp. 519–521, 1985. https://doi.org/10.1090/S0025-5718-1985-0777282-X.

[9] G. Jaberipur, A. Kaivani, "Improving the speed of parallel decimal multiplication," *IEEE Transactions on Computers*, vol. 58, issue 11, pp. 1539–1552, 2009. https://doi.org/10.1109/TC.2009.110.

[10] A. Schonhage and V. Strassen, "Schnelle Multiplication grosser Zahlen," *Computing*, vol. 7, no. 3–4, pp. 281–292, 1971. https://doi.org/10.1007/BF02242355.
[11] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, pp. 472–492, 1976. https://doi.org/10.1109/TIT.1976.1055638.
[12] B. Schneier, *Applied Cryptography.* John Wiley & Sons, New York, 1996. 467 p.
[13] T. Elgamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inform. Theory*, vol. IT-31, no. 4, pp. 469–472, 1985. https://doi.org/10.1109/TIT.1985.1057074.
[14] S. R. Dusse and B. S. Kalisky, "A cryptographic library for the Motorola DSP 56000," *Advances in Cryptology*, Eurocrypt'90, Lect. Notes Comput. Sci., pp. 230–244, 1990. https://doi.org/10.1007/3-540-46877-3_21.
[15] B. Arazi, "On primality testing using purely divisionless operations," The *Computer Journal*, vol. 37, no. 3, pp. 219–222, 1994. https://doi.org/10.1093/comjnl/37.3.219.
[16] S. Kumaravel and R. Marimuthu, "VLSI implementation of high performance RSA algorithm using vedic mathematics," *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, Sivakasi, India, 2007, pp. 126-128. https://doi.org/10.1109/ICCIMA.2007.238.
[17] Y. Bassil, A. Barbar, "Sequential and parallel algorithms for the addition of big-integer numbers," *International Journal of Computational Science*, vol. 4, no. 1, pp. 52–69, 2010.
[18] L. Dadda, "Multioperand parallel decimal adder: A mixed binary and BCD approach," *IEEE Transactions on Computers*, vol. 56, no. 10, pp. 1320–1328, 2007. https://doi.org/10.1109/TC.2007.1067.
[19] M. Véstias, N. Horácio, "Improving the area of fast parallel decimal multipliers," *Microprocessors and Microsystems*, vol. 61, pp. 96–107, 2018. https://doi.org/10.1016/j.micpro.2018.05.015.
[20] A. K. Cherri, "Optical carry-free addition and borrow-free subtraction based on redundant signed-digit numbers," *Proceedings of the IEEE 1993 National Aerospace and Electronics Conference-NAECON 1993*, pp. 1094-1099 vol.2, 1993. https://doi.org/10.1109/NAECON.1993.290789.
[21] M. S. Schmookler and A. Weinberger, "High speed decimal addition," *IEEE Transactions on Computers*, vol. C-20, no. 8, pp. 862–866, 1971. https://doi.org/10.1109/T-C.1971.223362.
[22] C. McGeoch, "Parallel addition," *The American Mathematical Monthly*, vol. 100, issue 9, pp. 867–871, 1993. https://doi.org/10.2307/2324666.
[23] R. Floyd, D. Knuth, "Addition machines," *SIAM Journal on Computing*, vol. 19, issue 2, pp. 329–340, 1990. https://doi.org/10.1137/0219022.
[24] A. V. Anisimov, "Carryless addition," *Cybernetics and Systems Analysis*, vol. 32, pp. 153–163, 1996. https://doi.org/10.1007/BF02366527.
[25] V. K. Zadiraka, A. M. Tereshchenko, "Calculating the sum of multidigit values in the parallel computational model," *Cybernetics and Systems Analysis*, vol. 58, pp. 473–480, 2022. https://doi.org/10.1007/s10559-022-00478-7.

***ANDRII TERESHCHENKO,** graduated Sumy State University in 1995. Since 1995 working in information technologies. 2005–2016 APL software developer at the Kyiv office of the company SimCorp (Denmark). Since 2016, C# (.NET) software developer of the company "Infopulse Ukraine". Defended thesis "The fast proceedings algorithms of multi-digit arithmetic" in 2010. Specialist in computational mathematics, orthogonal transformations of Fourier, Walsh. Author of 22 scientific works.*

VALERIY ZADIRAKA, *graduated the Faculty of Mechanics and Mathematics of T.G. Shevchenko Kyiv State University and since then has been working at the V.M.Glushkov Institute of Cybernetics of NAS of Ukraine. 1981-2017, a professor of the Department of Computational Mathematics, since 2001 professor of the Department of Automated Information Processing Systems and Management of Igor Sikorsky Kyiv Polytechnic Institute. Academician of the National Academy of Sciences of Ukraine (2015), doctor of physical and mathematical sciences (1981), professor (1992), head of the numerical methods optimization department of V. M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine (1988). Specialist in computational mathematics, theory of Fourier integrals, digital signal processing, cybersecurity. Author of more than 380 scientific works, including 25 books, student books and scientific manuals. Supervisor for 6 doctors and 19 candidates of sciences.*