Methods for comparing numbers in non-positional notation of residual classes

Victor Krasnobayev [0000-0001-5192-9918], Alexandr Kuznetsov [0000-0003-2331-6326],

Mihael Zub [0000-0002-7085-797X], Kateryna Kuznetsova [0000-0002-5605-9293]

V. N. Karazin Kharkiv National University, Svobody sq., 4, Kharkiv, 61022, Ukraine

v.a.krasnobaev@gmail.com, kuznetsov@karazin.ua,
mishazub007@gmail.com, kate.kuznetsova.2000@gmail.com

Abstract. Methods for optimizing computations in computer systems and components that are based on the use of arithmetic transformations in the system of residual classes are considered. Methods of comparing numbers in the system of residual classes based on the representation and processing of data without directly converting the compared numbers from a modular code to a positional code and back are investigated. Some methods are based on the principle of obtaining and comparing a unitary single-row code. Based on the proposed methods, algorithms for their implementation have been developed, in accordance with which a class of patentable devices has been developed for performing arithmetic and algebraic comparison of numbers in the system of residual classes.

Keywords. Non-positional number system, class of residues, arithmetic non-positional coding, computer systems and components.

1 Introduction

It is known that the use of non-positional number system in residual classes (RCS) significantly increases the reliability and performance of the computer system (CS) [1-8]. However, the need to determine the positional characteristics of numbers in the RCS reduces the overall effectiveness of the use of modular codes. Existing methods for processing positional data, in particular, methods for comparing numbers into RCS, have significant drawbacks, the main of which is the need to convert numbers from RCS to positional number system and vice versa, which reduces user productivity and reliability of CS [9-12].

In the article there are four methods for comparing numbers into RCS, based on the presentation and processing of data without directly converting the compared numbers from a modular code (an RCS code) into a positional code and backward.

2 Methods for arithmetic comparison of numbers in RCS

Let the RCS be given by ordered ($m_i < m_{i+1}$) mutually n pairs by simple natural numbers (bases) $m_1, m_2, \ldots, m_n$, d let the compared operands be represented as:

$$A = (a_1, a_2, \ldots, a_n),\ B = (b_1, b_2, \ldots, b_n).$$

In the case, it is assumed that the source operands lie in the appropriate intervals:

$$\left[\frac{j_1 M}{m_n}, \frac{(j_1+1)M}{m_n}\right) \text{ and } \left[\frac{j_2 M}{m_n}, \frac{(j_2+1)M}{m_n}\right),$$

where $M = \prod_{i=1}^{n} m_1$, and the interval's number $j_k + 1$ is determined by the well-known expression $j_k = \gamma_n \overline{m}_n (mod\, m_n)$, where the value of m_n is determined from the comparison solution $\overline{m}_n M / m_n \equiv l(mod\, m_n)$. When $j_1 \neq j_2$ the operation of arithmetic comparison can be implemented by comparing the number of intervals, namely: if $j_1 < j_2$, then $A < B$, if $j_1 > j_2$, then $A > B$. When $j_1 = j_2$ is determined by the number $j_2 + 1$ of the interval $\left[\frac{j_3 M}{m_n}, \frac{(j_3+1)M}{m_n}\right)$, in which the number $A - B$ is located. If $0 \leq j_3 < (m_n+1)/2$, then $A < B$, and if $\frac{m_n+1}{2} \leq j_3 < m$, then $A > B$.

The well-known [1] method of arithmetic comparison of numbers with RCS involves the conversion of numbers and type $A^{(H)} = (0, 0, \ldots, \gamma_n)$, which requires $n-1$ clock cycles of the null operation. In addition, it is necessary to make a positional comparison of the ($j_1 + 1$) and ($j_2 + 1$) intervals of the source operands A and B. All of this complicates the comparison algorithm and increases the comparison time of numbers, which leads to the need to develop a comparison method for RCS, which do not require determination of positional characteristics. Consider each of these methods.

2.1 The method of arithmetic comparison of numbers in the RCS (the method of arithmetic parallel subtraction)

We will consider comparable numbers in arbitrary intervals: $\left[jm_i, (j+1)m_i\right)$, where

$$\overline{j = 1, N-1} \left(N = \prod_{\substack{k=l \\ k \neq i}}^{n} m_k \right).$$

In the case, the source operands *A*, *B* are reduced to numbers that are multiples to m_i, by modular subtraction of the following form:

$$A_{m_i} = A - a_i = (a_1^{(i)}, a_2^{(i)}, ..., a_{i-1}^{(i)}, 0, a_{i+1}^{(i)}, ..., a_n^{(i)},$$

$$B_{m_i} = B - b_i = (b_1^{(i)}, b_2^{(i)}, ..., b_{i-1}^{(i)}, 0, b_{i+1}^{(i)}, ..., b_n^{(i)},$$

where

$$a_i = (a_1', a_2', ..., a_i, a_n'),\ b_i = (b_1', b_2', ..., b_i, b_n').$$

Further, by means of a set of constants $0, m_i, 2m_i, ..., (N-1)m_i$, represented by $(n-1)$-y base of the RCS $m_1, m_2, ..., m_{i-1}, m_{i+1}, ..., m_n$, the construction of the so-called single-row code, respectively, in the form

$$K_N^{(n_A)} = \{z_N z_{N-1} ... z_2 z_1\},\ z_{n_A} = 0\ \ (z_1 = 1; 1 = \overline{1, N}, 1 \neq n_A),$$

$$K_N^{(n_B)} = \{z_N' z_{N-1}' ... z_2' z_1'\},\ z_{n_B}' = 0\ \ (z_1' = 1, 1 = \overline{1, N}, 1 \neq n_B).$$

The algorithm for constructing a single-row code in RCS can be represented as follows (1):

$$\begin{cases} A_{m_i} - 0 = z_1, \\ A_{m_i} - m_i = z_2, \\ A_{m_i} - 2m_i = z_3, \\ \dots\dots\dots\dots\dots\dots \\ A_{m_i} - \left(\prod\limits_{\substack{\rho=1 \\ \rho \neq 1}}^{n-1} m_\rho - 1 \right) m_i = z_N, \end{cases} \quad \begin{cases} B_{m_i} - 0 = z_1', \\ B_{m_i} - m_i = z_2', \\ B_{m_i} - 2m_i = z_3', \\ \dots\dots\dots\dots\dots\dots \\ B_{m_i} - \left(\prod\limits_{\substack{\rho=1 \\ \rho \neq 1}}^{n-1} m_\rho - 1 \right) m_i = z_N. \end{cases} \tag{1}$$

In the case, we have:

$$z_{n_A} = 0,\ \text{at}\ A_{m_i} - n_A \cdot m_i = 0;\ \ z_{n_A} = 1,\ \text{at}\ A_{m_i} \neq n_A \cdot m_i;$$

$$z_{n_B}' = 0,\ \text{at}\ B_{m_i} - n_B \cdot m_i = 0;\ \ z_{n_B}' = 1,\ \text{at}\ B_{m_i} \neq n_B \cdot m_i.$$

Geometrically, this number comparison method can be explained as follows. The interval $\left[0, \prod_{i=1}^{n} m_i\right)$ is divided into segments. The source operands A и B, by subtracting the constants of the form $\prod_{\substack{\rho=1 \\ \rho \neq i}}^{n-1} m_\rho$, c and shift to the left edge of their hit interval. This is the equivalent to bringing the compared numbers to numbers, which are compile to comparison m_i of the RCS. On this basis, the accuracy W c and comparison of the operands depends on the size of the base m_i , that is $W = W(m_i)$. However, with the maximum accuracy of the comparison $W_{max} = W(m_{min})$ (for an ordered of the RCS $W_{max} = W(m_1)$ the number of equipment for technical devices realizing the comparison operation in the RCS increases dramatically. Indeed, the number of equipment N_0 of the comparison device in the RCS significantly depends on the number of address N_1, performing the operation of parallel subtraction:

In this way, necessity of ensure a high degree of accuracy of comparison requires a significant amount of equipment, which reduces the efficiency of using existing methods of comparing numbers in the RCS. This situation determines the relevance and importance of finding more effective methods of comparing numbers in the RCS, ensuring a high accuracy of W comparison with a minimum numbers N0 of equipment comparing devices.

In general, the task is interpreted as follows. It is necessary to find $N_0 = min$ at W_{max}, i.e. $N_0(W_{max}) = min$. As shown above $W_{max} = W(m_1)$. In this case, a change in the base $m_i \ (i = \overline{1,n})$ affects only the number N_1 of the equipment of the group of adders. In this case, the problem is correctly formulated as a definition (2)

$$N_1(W_{max}) = min \ . \tag{2}$$

Obviously, with high accuracy equal to the unit length of the interval, $W_{max} = W(m_i = 2)$. However, in this case $N_1(W_{max}) = max$, и and this result does not satisfy the condition (2). On the other hand – $N_1(W_{min}) = min$. Thus, it is necessary to develop such a method of comparison (2). This problem is solved by the method described below.

2.2 The method of arithmetic comparison of numbers in the RCS (the method of parallel subtraction with the comparison of residues)

We introduce an additional operation of comparing the residues a_n and b_n the magnitude of the bases m_n of the RCS. In this case, the result of the comparison of the residuals simultaneously with the result of the comparison of the single-row code $K_N^{(n_A)}$ and $K_N^{(n_B)}$, is determined by the solution of the problem (2), i.e. W_{max} and with the minimum amount of equipment N_{min} is the result of solving the operation of comparing two numbers in RCS. The algorithm for determining the result of an arithmetic comparison operation can be represented as follows:

$$\begin{cases} \text{if } n_A > n_B \text{, then } A > B; \\ \text{if } n_A < n_B \text{, then } A < B; \\ \text{if } n_A = n_B \text{, and at the same time} \\ \begin{cases} a_n = b_n \text{, at } A = B, \\ a_n > b_n \text{, at } A > B, \\ a_n < b_n \text{, at } A < B. \end{cases} \end{cases} \tag{3}$$

The set of relations (3) represents the general algorithm for the implementation of the operation of arithmetic comparison of numbers in the RCS. It is advisable to consider examples of specific performance of the operation of arithmetic comparison of numbers in the RCS. Let the RCS be given by bases, $m_1 = 2,$ $m_2 = 3$ and $m_3 = 5$. Code words are given in Table 1. In Table 2 given constants $a_n\,(b_n)$ presented in a given RCS, and in Table 3, constants of a single-row code are $\Delta \cdot m_n$ $\left(\Delta = 0, \overline{\prod_{i=1}^{n-1} m_i}\right) m_i$ on the basis of the RCS $(i = \overline{1, n-1})$.

Table 1. Table of code words in RCS

A	A in RCS			A	A in RCS		
	$m_1 = 2$	$m_2 = 3$	$m_3 = 5$		$m_1 = 2$	$m_2 = 3$	$m_3 = 5$
0	0	00	000	15	1	00	000
1	1	01	001	16	0	01	001
2	0	10	010	17	1	10	010
3	1	00	011	18	0	00	011
4	0	01	100	19	1	01	100
5	1	10	001	20	0	10	000
6	0	00	001	21	1	00	001
7	1	01	010	22	0	01	010
8	0	10	011	23	1	10	011
9	1	00	100	24	0	00	100
10	0	01	000	25	1	01	000
11	1	10	001	26	0	10	001
12	0	00	010	27	1	00	010
13	1	01	011	28	0	01	011
14	0	10	100	29	1	10	100

Table 2. Table of constants

γ_3	Constants
000	(0, 00, 000)
001	(1, 01, 001)
010	(0, 10, 010)
011	(1, 00, 011)
100	(0, 01, 100)

Example 1. Let the compared numbers be represented as $A_{23} = (1,10,011)$ and $B_{21} = (1,00,001)$. In this case, the values of the constants (Table 2) determine the values $A_{m_n} = A_{23} - a_n = (0,10,000)$, $B_{m_n} = B_{21} - b_n = (01,00,000)$, which corresponds to the shift of the operands A and B to the left edge of the interval [20, 25). Then we use the constants of the single-row code (Table 3), we determine the single-row code for the input numbers in the form:

$$K_N^{(n_A)} = K_6^{(4)} = \{110111\};\ K_N^{(n_B)} = K_6^{(n_B)} = K_6^{(4)} = \{110111\},$$

where

$$N_1 = \prod_{i=1}^{n-1} m_i = 6 .$$

At the same time the comparison result $a_n = 011 > b_n = 001$ is determined in parallel with the ($n = [log_2(m_n - 1)] + 1$)-th bit comparison circuit in time. So, if $n_A = n_B = 4$, in accordance with the above algorithm, we determine that $A_{23} > B_{21}$.

Table 3. Table of constants

$(0 \div N-1) \cdot m_n$	RCS		Number of zero position
	$m_1 = 2$	$m_2 = 3$	
$0 \cdot m_3 = 0$	0	00	1
$1 \cdot m_3 = 5$	1	10	2
$2 \cdot m_3 = 10$	0	01	3
$3 \cdot m_3 = 15$	1	00	4
$4 \cdot m_3 = 20$	0	10	5
$5 \cdot m_3 = 25$	1	01	6

Example 2. Let the compared numbers be represented as $A_{23} = (1,10,011)$, $B_3 = (1,00,011)$. In this case, the following differences are determined from the values of the constants (Table 2):

$$A_{m_n} = A_{23} - a_n = (0,10,00) \text{ and } B_{m_n} = B_3 - b_n = (1,00,000),$$

which corresponds to the shift of the operand A_{23} to the left edge of the interval [20, 25), and the operand B_3 to the left edge of the interval [0, 5). Next, using the constants of the single-row code (Table 3), we determine the single-row code for the considered input operands A_{23} , and B_3:

$$K_N^{(n_A)} = K_6^{(5)} = \{101111\},\ K_N^{(n_B)} = K_6^{(1)} = \{111110\}.$$

So, if $n_A = 5 > n_B = 1$, in accordance with the above algorithm, we determine that $A_{23} > B_3$.

2.3 The arithmetical method of comparison of numbers in the RCS (the method of comparison with a constant)

Consider the method of implementing the operation of arithmetic comparison of numbers in the RCS. The essence of this method is that not the operands A and B, are directly compared, but the quantities $\chi = (A - B) \mod M = (\gamma_1, \gamma_2, ..., \gamma_n)$ and m_1.

In this case, the value is determined:

$$\chi_{m_1} = \chi - \gamma_1 = (0, \gamma_2', \gamma_3', ..., \gamma_n'),$$

where constants $\gamma_1 = (\gamma_1, \gamma_2', \gamma_3', ..., \gamma_n')$ and $\gamma_1 = (a_1 - b_1) \mod m_1$ are represented in a given RCS.

Then the general algorithm for comparison of the operands is presented in the form:

$$\begin{cases} A > B, \text{ at } \chi_{m_1} \le m_1; \\ A < B, \text{ at } \chi_{m_1} > m_1; \\ A = B, \text{ at } \chi_{m_1} = 0. \end{cases} \tag{4}$$

By dialing constants

$$0, m_1, 2m_1, ..., (N-1) \cdot m_1 \left(N = \prod_{i=1}^{n} m_i \right),$$

which were represented in the RCS with bases $m_2, m_3, ..., m_n$, the construction of a single-row code in the form of:

$$K_n^{(n_\chi)} = \{ z_N z_{N-1} ... z_2 z_1 \},$$

where

$$\begin{cases} \chi_{m_1} - 0 = z_1, \\ \chi_{m_1} - m_1 = z_2, \\ \chi_{m_1} - 2m_1 = z_3, \\ . \; . \; . \; . \; . \; . \; . \; . \; . \\ \chi_{m_1} - (N-1)m_1 = z_n. \end{cases}$$

In addition

$$z_{n_\chi} = 0, \text{ for } \chi_{m_1} - n_\chi m_1 = 0,$$

$$z_{n_\chi} = 1, \text{ for } \chi_{m_1} - n_\chi m_1 \neq 0 .$$

The first module m_1 RCS is also represented by a single-row code of length N binary digits, in which the second place on the right will be zero $(m_1 - 1 \cdot m_1 = 0)$, and on the rest - one, i.e. the single-row unitary code will be represented as: $K_n^{(2)} = \{11...101\}$.

Further, by known methods, in accordance with algorithm (4), operands A and B, represented by a single-row code, are compared.

Example 3. For the above RCS we consider an example of the implementation of this method. Let the numbers be equal $A = (0,01,010)$ and $B = (1,10,011)$. We define the value:

$$\chi = (A - B) \bmod M = \left[(0-1) \bmod 2, (01-10) \bmod 3, (010-011) \bmod 5\right] =$$
$$= (1,10,100).$$

By value $\gamma_1 = 1$ we define a constant in the form $\gamma_1 = (1,01,001)$ (Table 4). Then we perform the operation:

$$\chi_{m_1} = \chi - \gamma = (00,01,011).$$

The operand χ_{m_1}, which is a multiple of the module $m_1 = 2$ value, goes to the first inputs of the corresponding adders, the second inputs of which receive the corresponding constants (Table 5).

Table 4. Constants

γ_1	Constants
00	(0, 00, 000)
01	(1, 01, 001)

Table 5. Table of constants number of zero position

$(0 \div N-1) \cdot m_1$ $\Delta \cdot 2$	RCS		Number of zero position
	$m_2 = 3$	$m_3 = 5$	
0 $m_1 = 0$	00	000	1
1 $m_1 = 2$	10	010	2
2 $m_1 = 4$	01	100	3
3 $m_1 = 6$	00	001	4
4 $m_1 = 8$	10	011	5
5 $m_1 = 10$	01	000	6
6 $m_1 = 12$	00	010	7
7 $m_1 = 14$	10	100	8
8 $m_1 = 16$	01	001	9
9 $m_1 = 18$	00	011	10
10 $m_1 = 20$	10	000	11
11 $m_1 = 22$	01	010	12
12 $m_1 = 24$	00	100	13
13 $m_1 = 26$	10	001	14
14 $m_1 = 28$	01	011	15

Since $\chi_{m_1} - 14_{m_i} = 0$, then the single-row code will take the form:

$$K_n^{(n_\chi)} = K_{15}^{(14)} = \{011111111111111\}.$$

In accordance with algorithm (4), we determine that $A < B$.

The advantage of the considered method is to ensure maximum accuracy of comparison with an acceptable amount of equipment for its implementation.

2.4 The method of algebraic comparison of numbers in the RCS

It is easy to go from the implementation of the operation of arithmetic comparison of numbers to the algebraic comparison of numbers. In this case, the compared numbers A and B have one additional significant digit, i.e. the number is accompanied by a indication $\Omega_A(\Omega_B)$ of the sign $signA(signB)$, where:

$$\Omega_A(\Omega_B) = \begin{cases} 0, \text{ if } A(B) \geq 0, \\ 1, \text{ if } A(B) < 0. \end{cases}$$

In this way, the compared numbers are presented in the form:

$$A' = (\Omega_A; A) = \{\Omega_A; (a_1, a_2, ..., a_n)\}, \text{ and } B' = (\Omega_B; B) = \{\Omega_B; (b_1, b_2, ..., b_n)\},$$

and the method of comparing numbers A' and B' is determined by the set of operations (5).

We will conduct a comparative analysis of the implementation time of the comparison operation of two numbers A and B or the proposed method and the most well-known. The essence of the known method is to convert the numbers A and B from the system of residual classes to the positional number system $A_{ПСС}$, $B_{ПСС}$ and the further comparison of the operands of A_{PNS} и B_{PNS}. Moreover, the transfer of numbers from RCS to PNS is made in accordance with the expression:

$$A_{PNS} = \left| \sum^{n} a_i B_i \right|_M ,$$

where $a_i = [\, A_{PNS} \,/\, m_i \,]\, m_i$, where B_i – is the orthogonal basis over the mi base of the RCS.

$$\begin{cases} \text{if } n_A > n_B \begin{cases} \Omega_A = 0,\ \Omega_B = 0, \text{ then } A' > B', \\ \Omega_A = 0,\ \Omega_B = 1, \text{ then } A' > B, \\ \Omega_A = 1,\ \Omega_B = 0, \text{ then } A' < B', \\ \Omega_A = 1,\ \Omega_B = 1, \text{ then } A' > B'; \end{cases} \\ \text{if } n_A < n_B \begin{cases} \Omega_A = 0,\ \Omega_B = 0, \text{ then } A' < B', \\ \Omega_A = 0,\ \Omega_B = 1, \text{ then } A' > B', \\ \Omega_A = 1,\ \Omega_B = 0, \text{ then } A' < B', \\ \Omega_A = 1,\ \Omega_B = 1, \text{ then } A' > B'; \end{cases} \\ \text{if } n_A = n_B \begin{cases} a_n = b_n \begin{cases} \Omega_A = 0,\ \Omega_B = 0, \text{ then } A' = B', \\ \Omega_A = 0,\ \Omega_B = 1, \text{ then } A' > B', \\ \Omega_A = 1,\ \Omega_B = 0, \text{ then } A' < B', \\ \Omega_A = 1,\ \Omega_B = 1, \text{ then } A' = B'; \end{cases} \\ a_n > b_n \begin{cases} \Omega_A = 0,\ \Omega_B = 0, \text{ then } A' > B', \\ \Omega_A = 0,\ \Omega_B = 1, \text{ then } A' > B', \\ \Omega_A = 1,\ \Omega_B = 0, \text{ then } A' < B', \\ \Omega_A = 1,\ \Omega_B = 1, \text{ then } A' < B'; \end{cases} \\ a_n < b_n \begin{cases} \Omega_A = 0,\ \Omega_B = 0, \text{ then } A' < B', \\ \Omega_A = 0,\ \Omega_B = 1, \text{ then } A' > B', \\ \Omega_A = 1,\ \Omega_B = 0, \text{ then } A' < B', \\ \Omega_A = 1, \Omega_B = 1, \text{ then } A' > B'. \end{cases} \end{cases} \end{cases} \quad (5)$$

For the known and proposed methods of comparing numbers, the implementation time is determined by the corresponding mathematical relations

$$T_{RCS}^{(1)} = t^{(11)} + t^{(12)} + t^{(13)} + t^{(14)}, \tag{6}$$

$$T_{RCS}^{(2)} = t^{(21)} + t^{(22)} + t^{(23)}, \tag{7}$$

where

- $t^{(11)}$ – is the implementation time of the multiplication operation for the maximum m_n by the magnitude of the RCS module (the implementation time of the multiplication operation of two-digit binary numbers $K = [log_2(m_n - 1)] + 1$);
- $t^{(12)}$ – is the time $(n-1)$ of the th addition of two numbers of the type $a_i B_i + a_{i+1} B_{i+1} (i = \overline{1, n-1})$;
- $t^{(13)}$ – is the time for determining the deduction of the number of A_{PNS} (B_{PNS});
- $t^{(14)}$ – is the comparison time of the positional operands of the A_{PNS} and B_{PNS};
- $t^{(21)}$ – is the time of implementation of the operation of subtraction in the RCS $A_{COK} - a_i$;
- $t^{(22)}$ – is the time of implementation of the operation of subtraction in the RCS $A_{m_i} - k \cdot m_i$;
- $t^{(23)}$ – is the time of comparison of two positional N-bit single-row unitary codes of two corresponding numbers.

It is known that the time of addition of t_c and t_y multiplication of two operands in the PNS is determined by the following relations:

$$t_c = \tau(2\rho - 1) \text{ and } t_y = 2\tau\rho^2 ,0$$

where ρ is the bit width of the processed operands; τ – is the time of "shift" of one binary digit. In this case, the time of "triggering" of the logical element AND (OR) is determined by the expression: $t_{AND} \approx t_{OR} \approx \tau / 2$, and $t^{(23)} \approx 6t_n$ and expression

$$t^{(13)} \approx 2\tau \left\{ \left[log_2 \left(\prod_{i=1}^{n} m_i - 1 \right) \right] + 1 \right\}^2 .$$

Taking into account the above, relations (6) and (7), respectively, are presented in the form:

$$T_{RCS}^{(1)} = 2\tau k^2 + (n-1)\tau(2k-1) + 2\tau \left\{ \left[log_2 \left(\prod_{i=1}^{n} m_i - 1 \right) \right] + 1 \right\}^2 + 3\tau ; \tag{8}$$

$$T_{RCS}^{(2)} = 2\tau + 2\tau + 3\tau . \tag{9}$$

According to with expressions (8) and (9), values are calculated $T^{(1)}_{RCS}$, $T^{(2)}_{RCS}$ (table 6) for various l – byte bit CS grids $(l = \overline{1,4})$. From Table 6 it can be seen that with increasing length of the CS discharge grid, which is typical of the current trend in the development of systems and tools for processing digital information, the effectiveness of applying the proposed methods for comparing numbers, as compared to existing ones, increases.

Table 6. Comparative analysis of the implementation time of the comparison operation

T	l			
	1	2	3	4
$T^{(1)}_{RCS}$	324	870	1916	3334
$T^{(2)}_{RCS}$	6	6	6	6

On the basis of the proposed methods, algorithms for their implementation have been developed, in accordance with which a class of patentable devices has been developed for performing arithmetic and algebraic comparisons of numbers in an RCS [13-15]. Prospective direction of a further research is the argumentation of practical recommendations concerning a realization of the introduced method and the ways of its use in different mechanisms of an information security of telecommunications networks and systems [16-31].

3 Conclusion

In this paper, a method is proposed for comparing numbers in a non-positional number system of remainder classes, which is based on the principle of obtaining and comparing a unitary single-row code.

The developed method can be used to improve advanced computer systems and their components. In particular, its practical use allows to increase the performance of computer calculations and the reliability of information systems.

References

1. Shu, S., Wang, Y., Wang, Y.: A research of architecture-based reliability with fault propagation for software-intensive systems. In: 2016 Annual Reliability and Maintainability Symposium (RAMS), Tucson, AZ, 2016, pp. 1–6. (2016) doi:10.1109/RAMS.2016.7447984
2. Gokhale, S.S., Lyu, M.R., Trivedi, K.S.: Reliability simulation of component-based software systems. In: Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No.98TB100257), Paderborn, Germany, 1998, pp. 192–201. (1998) doi:10.1109/ISSRE.1998.730882

3. Tiwari, A., Tomko, K.: Enhanced Reliability of Finite State Machines in FPGA Through Efficient Fault Detection and Correction. IEEE Transaction on Reliability, vol. 54(3), pp. 459–467. (2005) doi:10.1109/TR.2005.853438
4. Singh, C., Sprintson, A.: Reliability assurance of cyber-physical power systems. In: IEEE PES General Meeting, Providence, RI, 2010, pp. 1–6. (2010) doi:10.1109/PES.2010.5590189
5. Krasnobayev, V.A., Koshman, S.A., Mavrina, V.A.: A Method for Increasing the Reliability of Verification of Data Represented in a Residue Number System. Cybernetics and Systems Analysis, 2014, vol. 50(6), pp. 969–976. (2014) doi:10.1007/s10559-014-9688-3
6. Reddy, M., Nalini, N.: FT2R2Cloud: Fault tolerance using time-out and retransmission of requests for cloud applications. In: 2014 International Conference on Advances in Electronics Computers and Communications, Bangalore, 2014, pp. 1–4. (2014) doi:10.1109/ICAECC.2014.7002396
7. Braun, C., Wunderlich, H.: Algorithm-based fault tolerance for many-core architectures. In: 2010 15th IEEE European Test Symposium, Praha, 2010, pp. 253–253. (2010) doi:10.1109/ETSYM.2010.5512738
8. Radu, M.: Reliability and fault tolerance analysis of FPGA platforms. In: IEEE Long Island Systems, Applications and Technology (LISAT) Conference 2014, Farmingdale, NY, 2014, pp. 1–4. (2014) doi:10.1109/LISAT.2014.6845211
9. Akushsky, I., Yuditsky, D.: Machine arithmetic in residual classes. Moscow, Sov. Radio (in Russian) (1968)
10. Torgashov, V.: System of residual classes and reliability of digital computers. Moscow, Sov. Radio (in Russian) (1973)
11. Krasnobayev, V., Kuznetsov, A., Koshman, S., Moroz, S.: Improved Method of Determining the Alternative Set of Numbers in Residue Number System. In: Recent Developments in Data Science and Intelligent Analysis of Information. ICDSIAI 2018. Advances in Intelligent Systems and Computing, Springer, Cham, 2018, vol. 836, pp. 319–328. (2019) doi:10.1007/978-3-319-97885-7_31
12. Krasnobayev, V.A., Yanko, A.S., Koshman, S.A.: A Method for arithmetic comparison of data represented in a residue number system. Cybernetics and Systems Analysis, 2016, vol. 52(1), pp. 145–150. (2016) doi:10.1007/s10559-016-9809-2
13. Patent for utility model No. 73379, Ukraine, G 06F 7/04. Device for the popular danich, which is presented in the classroom extra. No. u 2012 01885. Appl. 02/20/2012. Publ. 09/25/2012, Bull. No. 18
14. Patent for utility model No. 73384, Ukraine, G 06F 7/04. Device for the same number of numbers in the classroom too. No. u 2012 01916. Appl. 02/20/2012. Publ. 09/25/2012, Bull. No. 18
15. Patent for utility model No. 73407, Ukraine, G 06F 7/04. Device for algebraic porous numbers with class numbers No. U 2012 02150. Appl. February 24, 2012. Publ. 09/25/2012, Bull. No. 18
16. Gorbenko, I.D., Zamula, A.A., Semenko, A.E., Morozov, V.L.: Method for complex improvement of characteristics of orthogonal ensembles based on multiplicative combining of signals of different classes. Telecommunications and Radio Engineering, 2017, vol. 76(18), pp. 1581–1594. (2017) doi:10.1615/TelecomRadEng.v76.i18.10
17. Si-ru, Z.: Application of Improved Genetic Algorithm in Optimization Computation. In: 2009 Fifth International Conference on Natural Computation, Tianjin, 2009, pp. 144–148. (2009) doi:10.1109/ICNC.2009.393
18. Gorbenko, I., Zamula, A., Morozov, V.: Methods for implementing communications in info-communication systems based on signal structures with specified properties. In: 2017

4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkov, 2017, pp. 101–104. (2017) doi:10.1109/INFOCOMMST.2017.8246359
19. Kuznetsov, A., Kiyan, A., Uvarova, A., Serhiienko, R., Smirnov, V.: New Code Based Fuzzy Extractor for Biometric Cryptography. In: 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 2018, pp. 119–124. (2018) doi:10.1109/INFOCOMMST.2018.8632040
20. Gorbenko, I., Kuznetsov, A., Tymchenko, V., Gorbenko, Y., Kachko, O.: Experimental Studies of the Modern Symmetric Stream Ciphers. In: 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 2018, pp. 125–128. (2018) doi:10.1109/INFOCOMMST.2018.8632058
21. Eberhart, R.C., Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), Seoul, South Korea, 2001, pp. 94–100. (2001) doi:10.1109/CEC.2001.934376O
22. Ivanov, O., Ruzhentsev, V., Oliynykov, R.: Comparison of Modern Network Attacks on TLS Protocol. In: 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 2018, pp. 565–570. (2018) doi:10.1109/INFOCOMMST.2018.8632026
23. Kandemir, M., Choudhary, A., Ramanujam, J., Kandaswamy, M.A.: A unified framework for optimizing locality, parallelism, and communication in out-of-core computations. IEEE Transactions on Parallel and Distributed Systems, 2000, vol. 11(7), pp. 648–668. (2000) doi:10.1109/71.877759
24. Pousset, Y., Vauzelle, R., Combeau, P.: Optimising the computation time of radio coverage predictions for microcellular mobile systems. IEEE Proceedings - Microwaves, Antennas and Propagation, 2003, vol. 150(5), pp. 360–364. (2003) doi:10.1049/ip-map:20030543
25. Xing, C., Wang, H.: Optimized similarity computation for nearest neighbor choosing. In: Proceedings of 2012 2nd International Conference on Computer Science and Network Technology, Changchun, 2012, pp. 2011–2014. (2012) doi:10.1109/ICCSNT.2012.6526313
26. Munetomo, M., Bando, S.: A scalable infrastructure of interactive evolutionary computation to evolve services online with data. In: 2013 IEEE International Conference on Big Data, Silicon Valley, CA, 2013, pp. 28–28. (2013) doi:10.1109/BigData.2013.6691793
27. Kavun, S.: Conceptual fundamentals of a theory of mathematical interpretation. Int. J. Computing Science and Mathematics, 2015, vol. 6(2), pp. 107–121. (2015) doi:10.1504/IJCSM.2015.069459
28. Kavun, S.: Indicative-geometric method for estimation of any business entity. Int. J. Data Analysis Techniques and Strategies, 2016, vol. 8(2), pp. 87–107. (2016) doi: 10.1504/IJDATS.2016.077486
29. Kovtun, V., Kavun, S., Zyma, O.: Co-Z Divisor Addition Formulae in Homogeneous Representation in Jacobian of Genus 2 Hyperelliptic Curves over Binary Fields. International Journal Biomedical Soft Computing and Human Sciences, 2012, vol. 17(2), pp. 37–43. (2012) doi:10.24466/ijbschs.17.2_37
30. Zamula, A., Kavun, S.: Complex systems modeling with intelligent control elements. Int. J. Model. Simul. Sci. Comput., 2017, vol. 08(01). (2017) doi:10.1142/S179396231750009X

31. Kavun, S., Zamula, A., Mikheev, I.: Calculation of expense for local computer networks. In: Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), 2017 4th International, Kharkiv, Ukraine, 2017, pp. 146–151. (2017) doi:10.1109/INFOCOMMST.2017.8246369