DOI: 10.15514/ISPRAS-2026-38(5)-1

Algorithm for Reverse Conversion from Hybrid Positional-Residue Number System

[1] A.E. Geryugova, ORCID: 0009-0005-8389-2204 <ajsanatgerugova@gmail.com>
[1] V.V. Lutsenko, ORCID: 0000-0003-4648-8286 <officialvladlutsenko@gmail.com>
[2] A.A. Sinitsyn, ORCID: 0009-0002-7740-4217 <antony@email.su>

[1] North-Caucasus Federal University, Stavropol,
1, Pushkin st., Stavropol, 355017, Russia.

[2] Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. Modern high-performance computing systems require efficient methods for representing and processing numerical information. The residue number system provides a high level of parallelism in performing arithmetic operations; however, it is characterized by high computational complexity of reverse conversion to the positional number system, which limits its practical application. As a compromise approach, the hybrid positional-residue number system is considered, combining the advantages of positional and non-positional representations. In this paper, we propose a reverse conversion algorithm from the hybrid positional-residue number system to the positional number system, based on representing a number as a weighted sum of digits and utilizing a parallel computation structure interpreted as a linear neural network architecture over a finite ring. The proposed approach enables efficient implementation of the summation operation using parallel and pipelined data processing. A theoretical analysis of the computational complexity of the proposed algorithm is performed, and an experimental comparison with classical methods of reverse conversion from the residue number system to the positional number system is carried out. The obtained results show that the proposed method provides a significant reduction in execution time by reducing the number of operations on large-bit numbers and efficiently exploiting parallelism. The proposed approach can be used in the development of hardware and software solutions for problems in cryptography, digital signal processing, high-performance and high-precision computing.

Keywords: residue number system; hybrid positional-residue number system; high performance computing; reverse conversion.

For citation: Geryugova A.E., Lutsenko V.V., Sinitsyn A.A. Algorithm for reverse conversion from hybrid positional-residue number system. Trudy ISP RAN/Proc. ISP RAS, vol. 38 issue 5, 2026, pp. 7-20. DOI: 10.15514/ISPRAS-2026-38(5)-1.

Acknowledgements. The research was supported by the Russian Science Foundation, project no. 25-71-30007.

Алгоритм обратного преобразования из гибридной позиционно-остаточной системы счисления

[1] А.Э. Герюгова, ORCID: 0009-0005-8389-2204 <ajsanatgerugova@gmail.com>
[1] В.В. Луценко, ORCID: 0000-0003-4648-8286 <officialvladlutsenko@gmail.com>
[2] А.А. Синицын, ORCID: 0009-0002-7740-4217 <antony@email.su>

[1] Северо-Кавказский федеральный университет,
Россия, 355017, г. Ставрополь, ул. Пушкина, д. 1.

[2] Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Современные высокопроизводительные вычислительные системы требуют эффективных методов представления и обработки числовой информации. Система остаточных классов обеспечивает высокий уровень параллелизма при выполнении арифметических операций, однако характеризуется высокой вычислительной сложностью обратного преобразования в позиционную систему счисления, что ограничивает её практическое применение. В качестве компромиссного подхода рассматривается гибридная позиционно-остаточная система счисления, сочетающая преимущества позиционных и непозиционных представлений. В работе предлагается алгоритм обратного преобразования из гибридной позиционно-остаточной системы счисления в позиционную систему счисления, основанный на представлении числа в виде взвешенной суммы разрядов и использовании параллельной структуры вычислений, интерпретируемой как линейная нейросетевая архитектура над конечным кольцом. Предложенный подход позволяет эффективно реализовать операцию суммирования с использованием параллельной и конвейерной обработки данных. Проведён теоретический анализ вычислительной сложности предложенного алгоритма, а также выполнено экспериментальное сравнение с классическими методами обратного преобразования из системы остаточных классов в позиционную систему счисления. Полученные результаты показывают, что предложенный метод обеспечивает значительное сокращение времени выполнения за счёт уменьшения количества операций над числами большой разрядности и эффективного использования параллелизма. Предложенный подход может быть использован при разработке аппаратных и программных решений для задач криптографии, цифровой обработки сигналов, высокопроизводительных и высокоточных вычислений.

Ключевые слова: система остаточных классов; гибридная позиционно-остаточная система счисления; высокопроизводительные вычисления; обратное преобразование.

Для цитирования: Герюгова А.Э., Луценко В.В., Синицын А.А. Алгоритм обратного преобразования из гибридной позиционно-остаточной системы счисления. Труды ИСП РАН, том 38, вып. 5, 2026 г., стр. 7–20 (на английском языке). DOI: 10.15514/ISPRAS–2026–38(5)–1.

Благодарности. Исследование выполнено при поддержке Российского научного фонда, проект № 25-71-30007.

1. Introduction

Modern computing systems, digital signal processing, and cryptography fundamentally rely on methods of representing numerical information. Number systems are generally divided into two fundamental classes: positional and non-positional. The most widespread in modern technologies are positional b-ary number systems, in particular binary and decimal, due to their uniformity and the simplicity of algorithms for performing arithmetic operations. However, they have several limitations, such as the propagation of carries during addition and multiplication, which restricts speed in high-performance computing [1].

Non-positional number systems, in which the quantitative value of a digit does not depend on its position in the representation of a number, are structured differently. One of the most common among these is the residue number system (RNS) [2, 3], in which a number is represented by a set of residues from division by pairwise coprime moduli. This system offers several advantages,

including high computational speed, the absence of inter-digit carries, and increased energy efficiency. Operations such as addition, subtraction, and multiplication are performed in parallel and independently for each modulo.

However, a drawback of RNS is the complexity of reverse conversion, i.e., converting a number from its residue representation back to a positional number system (PNS). This is a key problem for so-called non-modular operations such as number comparison [4], sign determination [5], division [6], and overflow detection [7], which require knowledge of the positional representation of the number. Classical reverse conversion methods based on the Chinese Remainder Theorem (CRT) or orthogonal bases [8] are computationally complex and negate the speed gain achieved through parallelism in elementary operations.

As an alternative to reduce the computational complexity of non-modular operations in RNS, a new number system has been proposed – the hybrid positional-residue number system (HPR) [9], designed for representing large integers and elements of a prime field ($\mathbb{F}_p$). The HPR is based on a synthesis of a positional representation with a mixed radix and RNS digits. This approach can be seen as a compromise between the classical positional number system and RNS. The proposed hybrid system retains internal parallelism in performing addition, subtraction, and multiplication, although this characteristic is somewhat lower than in RNS. At the same time, the presence of positional information in the data representation structure significantly enhances the efficiency of performing non-modular operations, residue calculation, and modular multiplication compared to RNS.

The aim of this study is to develop and analyze a reverse conversion algorithm from the HPR to a positional number system that would minimize the computational cost of this operation and thereby enhance the overall practical applicability of the hybrid system.

To achieve this aim, the following objectives are addressed:

1. To analyze classical reverse conversion methods from RNS to PNS and evaluate their computational complexity.
2. To develop a reverse conversion algorithm from the HPR using a finite ring neural network.
3. To perform a theoretical evaluation of the computational complexity of the proposed algorithm.
4. To conduct an experimental performance analysis (software implementation and testing) of the developed method in comparison with algorithms for RNS and analyze the obtained results.

The paper is structured as follows: Section 2 discusses the residue number system and the hybrid positional-residue number system. Section 3 examines reverse conversion methods from RNS. Then, Section 4 proposes a reverse conversion algorithm from the HPR. Section 5 presents a performance analysis of the proposed methods through computational complexity estimation and software implementation. The conclusion summarizes the obtained results.

2. Mathematical Foundations

2.1 Residue Number System

Definition 1. A basis of the residue number system is a set of moduli $\{p_1, p_2, \dots, p_n\}$, where each modulo $p_i \geq 2\ (i = 1, 2, \dots, n)$ and $gcd(p_i, p_j) = 1$ for $i \neq j$. Here and hereafter, gcd denotes the greatest common divisor. By convention, the moduli are ordered in increasing order: $p_1, p_2, \dots, p_n$.

Definition 2. The product of the moduli $P = \prod_{i=1}^{n} p_i$ is called the dynamic range of the RNS.

Definition 3. An integer $X \in [0, P)$ is represented as an n-dimensional vector of least non-negative residues: $X = (x_1, x_2, \dots, x_n)$, where $x_i \equiv X\ mod p_i$ denoted as $x_i = |X|_{p_i}$.

Consider an RNS with the basis $\{5, 9, 11\}$. The dynamic range is $P = 5 \times 9 \times 11 = 495$. Numbers in the interval $[0, 495)$ are represented uniquely.

Table 1. Results for different diode insertion strategies.

$0 \overset{RNS}{\longrightarrow} (0,0,0)$	$1 \overset{RNS}{\longrightarrow} (1,1,1)$	$2 \overset{RNS}{\longrightarrow} (2,2,2)$	$3 \overset{RNS}{\longrightarrow} (3,3,3)$	$4 \overset{RNS}{\longrightarrow} (4,4,4)$
$5 \overset{RNS}{\longrightarrow} (0,5,5)$	$6 \overset{RNS}{\longrightarrow} (1,6,6)$	$7 \overset{RNS}{\longrightarrow} (2,7,7)$	$8 \overset{RNS}{\longrightarrow} (3,8,8)$	$9 \overset{RNS}{\longrightarrow} (4,0,9)$
$10 \overset{RNS}{\longrightarrow} (0,1,10)$	$11 \overset{RNS}{\longrightarrow} (1,2,0)$	$12 \overset{RNS}{\longrightarrow} (2,3,1)$	$13 \overset{RNS}{\longrightarrow} (3,3,2)$	$14 \overset{RNS}{\longrightarrow} (4,4,3)$
$15 \overset{RNS}{\longrightarrow} (0,6,4)$	$16 \overset{RNS}{\longrightarrow} (1,7,5)$	$17 \overset{RNS}{\longrightarrow} (2,8,6)$	$18 \overset{RNS}{\longrightarrow} (3,0,7)$	$19 \overset{RNS}{\longrightarrow} (4,1,8)$
$20 \overset{RNS}{\longrightarrow} (0,2,9)$	$21 \overset{RNS}{\longrightarrow} (1,3,10)$	$22 \overset{RNS}{\longrightarrow} (2,4,0)$	$23 \overset{RNS}{\longrightarrow} (3,5,1)$	$24 \overset{RNS}{\longrightarrow} (4,6,2)$

The RNS has a characteristic feature – the ability to perform addition, subtraction, and multiplication operations in parallel and independently for each modulo. For numbers $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$, operations are performed in parallel for each modulo:

$$(x_1, x_1, \dots, x_n) \circ (y_1, y_2, \dots, y_n) = \left(|x_1 \circ y_1|_{p_1}, |x_2 \circ y_2|_{p_2}, \dots, |x_k \circ y_k|_{p_n}\right), \quad (1)$$

where $\circ \in \{+, -, \times\}$.

Example 1. Addition of $X = 5$ and $Y = 18$ in the basis $\{5, 9, 11\}$. From Table 1: $X = (0, 5, 5), Y = (3, 0, 7)$

$X + Y = (|0 + 3|_5, |5 + 0|_9, |5 + 7|_{11}) = (3, 5, 1)$.\\The result corresponds to the decimal number 23.

This example clearly demonstrates the advantages of modular arithmetic. The considered dynamic range allows working with numbers up to 4 bits. However, the moduli themselves require only 3 bits for encoding, which enables parallel computations with a smaller bit width for each modulo. This property is particularly valuable in cryptography: instead of processing 2048-bit numbers, as in modern encryption algorithms, one can use 33 parallel RNS channels with 32-bit moduli [10, 11].

2.2 Hybrid Positional-Residue Number System

In this section, we define the HPR and describe its properties. The main idea is to create a representation that combines the advantages of the residue number system and the standard positional number system. On the one hand, RNS is efficient for multiplication operations but inefficient for determining the sign of a number, comparing numbers, division, and overflow detection. On the other hand, the positional number system allows faster comparisons, but multiplication in it is more costly.

Definition 4. Let two mutually coprime RNS bases B_a and B_b be given, whose moduli are chosen such that they are pairwise coprime, ensuring the correct operation of the RNS, including the unique representation of numbers within a given range. Together with $P_a = \prod_{i=0}^{n_a-1} p_{a,i}$, a degree $d \in \mathbb{N}^*$, as well as β_{min} and $\beta_{max} \in \mathbb{R}$ such that $\beta_{max} - \beta_{min} \geq 1$ and $\beta_{max} + \beta_{min} \geq 0$. An integer X is represented in the HPR with parameters $(B_a, B_b, \beta_{min}, \beta_{max})$ in a positional system with a high radix P_a as:

$$X_{HPR} = (\langle X_{d-1} \rangle_{a|b}, \dots, \langle X_0 \rangle_{a|b})_{HPR}, \quad (2)$$

where the digits $\langle X_i \rangle_{a|b}$, represented in RNS, satisfy the condition:

$$X = \sum_{i=0}^{d-1} X_i P_a^i, \quad (3)$$

with $\beta_{min} P_a \leq X_i \leq \beta_{max} P_a$.

Fig. 1 illustrates the behavior of basic operations in HPR compared to standard positional systems and RNS. The positional system (top part) has limited internal parallelism due to carries but provides precise magnitude information. RNS (bottom part) exhibits high parallelism, but its non-positional nature complicates comparisons. HPR (middle part) represents a compromise between these two approaches, preserving parallelism at the digit level while providing coarse magnitude information.

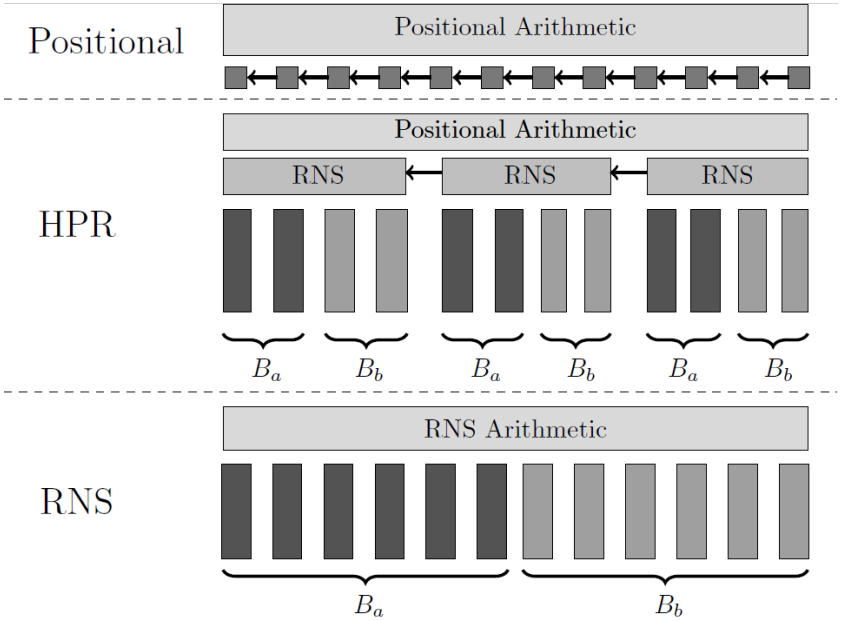

Fig. 1. HPR is considered as a compromise between positional representation and RNS representation.

HPR can be redundant if $\beta_{max} - \beta_{min} > 1$, and supports negative digits when $\beta_{min} < 0$. HPR generalizes both RNS and positional systems. For example, when $d = 1$, $\beta_{min} = 0$, and $\beta_{max} = 1$, HPR coincides with standard RNS. If $B_a = (2^w)$ and $n_b = 0$, HPR becomes a standard positional system with radix 2^w.

Example 5. In HPR, represent the number $X = 200$ with bases $B_a = \{3,5\}, B_b = \{7,11\}$. The number of HPR digits is $d = 2$, $\beta_{min} = 0$, $\beta_{max} = 3$. First, we split into digits with radix $P_a = 15$:

$$X = 13 \cdot 15^1 + 5 \cdot 15^0.$$

We obtain the digits: $X_1 = 13, X_0 = 5$.

We represent each digit in RNS:

$$\langle 13 \rangle_a = (|13|_3, |13|_5) = (1,3),$$
$$\langle 13 \rangle_b = (|13|_7, |13|_{11}) = (6,2),$$
$$\langle 5 \rangle_a = (|5|_3, |5|_5) = (2,0),$$
$$\langle 5 \rangle_b = (|5|_7, |5|_{11}) = (5,5).$$

According to formula (2), the number X is converted to HPR:

$$X_{HPR} = \big((1,3,6,2),(2,0,5,5)\big).$$

Thus, the number X in HPR equals $\big((1,3,6,2),(2,0,5,5)\big)$.

In HPR, operations are performed at two levels: the digit level – using standard positional algorithms, and the residue level – applying RNS algorithms (within digits).

Addition and Subtraction: At the residue level, operations are performed in parallel across all moduli $\langle X_i \rangle_{a|b} = \langle a_i \rangle_{a|b} \pm \langle b_i \rangle_{a|b}$. If the value of digit x_i exceeds P_a, a carry is required, which is handled using redundancy (e.g., as in multi-precision libraries).

Multiplication: At the digit level, any multiplication algorithms can be used – schoolbook, Karatsuba, etc. When multiplying two digits $\langle X_i \rangle_{a|b} = \langle a_i \rangle_{a|b} \times \langle b_i \rangle_{a|b}$, it is necessary to extract the "high part" of X_i, which in RNS requires computing X_i via CRT and dividing by P_a:

$$X_i = Q_i P_a + R_i. \tag{4}$$

The remainder R_i is kept, while the quotient Q_i is carried over to the next digit $X_i + 1$.

3. Reverse Conversion Methods from RNS to PNS

3.1 Chinese Remainder Theorem

The classical method for recovering the integer value of a number from its residues, obtained from the constructive proof of the CRT [12], consists of computing the b-ary sum modulo P:

$$X = \left| \sum_{i=1}^{n} P_i |x_i w_i|_{p_i} \right|_P, \tag{5}$$

where $P_i = \frac{P}{p_i}$, and the values $w_i = |P_i^{-1}|_{p_i}$ are known as multiplicative (modular) inverses of P_i modulo p_i, satisfying $w_i P_i \equiv 1 \, mod \, p_i$. It is easy to see that $1 \le w_i < p_i$.

Problems in implementing (5) arise due to the need for a labor-intensive final reduction modulo the large modulo P, i.e., finding the remainder of the computed sum upon division by P, especially when P consists of several hundred bits. Therefore, ongoing research is mainly devoted to developing algorithms that eliminate reduction modulo P from the computation process.

To illustrate reverse conversion using CRT, let us consider the following Example 6.

Example 6. In an RNS with the basis $\{5, 9, 11\}$recover the number $X = (3,5,1)$. First, we find P_i:

$$P_1 = \frac{P}{p_1} = 99;\ P_2 = \frac{P}{p_2} = 55;\ P_3 = \frac{P}{p_3} = 45.$$

Next, we find the values w_i:

$$w_1 = 4;\ w_2 = 1;\ w_3 = 1.$$

Then, using (5), we obtain:

$$X = |99 \cdot |3 \cdot 4|_5 + 55 \cdot |5 \cdot 1|_9 + 45 \cdot |1 \cdot 1|_{11}|_{495} = |518|_{495} = 23.$$

Thus $X = (3,5,1) = 23$.

3.2 Mixed-Radix Conversion

Another classical reverse conversion method does not require modular reduction and expresses the number X in the form:

$$X = \overline{x_1} \cdot W_1 + \overline{x_2} \cdot W_2 + \ldots + \overline{x_n} \cdot W_n, \tag{6}$$

where $W_1, W_2, \ldots, W_n$ are the bases of the mixed-radix system:

$$\begin{aligned} W_1 &= 1, \\ W_2 &= p_1, \\ W_3 &= p_1 p_2, \\ &\ldots \\ W_n &= p_1 p_2 \ldots p_n. \end{aligned} \tag{7}$$

The uniqueness of representation (6) is guaranteed for all integers in the interval $[0, P-1]$, which indicates a high degree of similarity between RNS and the mixed-radix system, both of which are based on the same set of modulo bases.

The main problem of this method is converting a number from RNS to the mixed-radix system, i.e., computing the digits $\overline{x_1}, \overline{x_2}, \dots, \overline{x_n}$. For a given number $X = (x_1, x_2, \dots, x_n)$, the classical MRC algorithm [13] sequentially computes all digits of the mixed-radix representation, starting from the least significant:

$$\begin{gathered} \overline{x_1} = x_1, \\ \overline{x_2} = |(x_2 - \overline{x_1})c_{12}|_{p_2}, \\ \overline{x_3} = \left|\left((x_3 - \overline{x_1})c_{13} - \overline{x_2}\right)c_{23}\right|_{p_3}, \\ \dots \\ \overline{x_n} = \left|\left((\dots(x_n - \overline{x_1})c_{1n} - \overline{x_2})c_{2n} - \ \dots - \overline{x_{n-1}}\right)c_{n-1,n}\right|_{p_n}. \end{gathered} \tag{8}$$

Computing (7) requires arithmetic operations modulo p, namely $\frac{n(n-1)}{2}$ subtractions and the same number of multiplications. An improved algorithm proposed in [14] reduces the number of multiplications to $n-2$. However, it introduces additional constraints on the choice of RNS moduli, since the condition $|P_i^{-1}|_{p_I} = 1$ must hold for all $i = 1, 2, \dots, n$.

Although each subsequent digit $\overline{x_i} + 1$ depends on the previous digit $\overline{x_i}$, process (6) can be partially parallelized (pipelined) [12], as shown in Algorithm 2. In this algorithm, k is the stream index, and v is an array in shared memory accessible to all n streams. It should be noted that synchronization may be required at the end of each loop iteration to eliminate data races.

To illustrate reverse conversion using mixed-radix conversion, let us consider the following Example 7.

Example 7. Continuing with the modulo set $\{5, 9, 11\}$ and the number $X = (3, 5, 1)$, we find the mixed-radix bases W_i:

$$W_1 = 1;\ W_2 = 5;\ W_3 = 45.$$

Next, let us calculate the mixed-radix digits:

$$\overline{x_1} = 3;\ \overline{x_2} = |(5-3)2|_9 = |4|_9 = 4;\ \overline{x_3} = \left|\left((1-3)9 - 4\right)5\right|_{11} = |110|_{11} = 0.$$

Then, according to (6), we have:

$$X = 3 \cdot 1 + 4 \cdot 5 + 0 \cdot 45 = 23.$$

Hence $X = (3, 5, 1) = 23$.

3.3 New Chinese Remainder Theorem II

In [15], an original reverse conversion method called New CRT II was proposed, which is based on the divide-and-conquer principle. Let $X = (x_1, x_2, \dots, x_n)$ and $t = \left\lfloor \frac{n}{2} \right\rfloor$. The binary value of X is proposed to be computed as follows:

$$X = X_2 + |k_0(X_1 - X_2)|_{\widetilde{P_1}}\widetilde{P_2}, \tag{9}$$

where X_1 is the integer corresponding to the residues $(x_1, x_2, \dots, x_t)$ with respect to the moduli $\{p_1, p_2, \dots, p_t\}$ with dynamic range $P_1 = p_1 \cdot p_2 \cdot \dots \cdot p_t$. Similarly, X_2 is the integer corresponding to the residue vector $(x_{t+1}, x_{t+2}, \dots, x_n)$ with respect to the set $\{p_{t+1}, p_{t+2}, \dots, p_n\}$ with dynamic range $\widetilde{P_2} = p_{t+1} \cdot p_{t+2} \cdot \dots \cdot p_n$. The constant $k_0 = |P_2^{-1}|_{\widetilde{P_1}}$ is the multiplicative inverse of P_2 modulo $\widetilde{P_1}$.

The integers X_1 and X_2 are in turn computed by applying (9) with the partitioning of the ranges $\widetilde{P_1}$ and $\widetilde{P_2}$ into corresponding subranges of half the length, and so on. The recursive partitioning continues until each set of moduli contains only two elements. Thus, reverse conversion is performed in $log_2 n$ steps with the possibility of parallelizing computations at each step. The maximum operation size is reduced from P to $\sqrt{P}$.

Based on New CRT II, efficient reverse converters have been developed for arbitrary modulo sets [16], as well as for sets of the form $\{2^n - 1, 2^n + 1, 2^{2n} - 2, 2^{2n+1} - 3\}$ [17] and

$\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ [18]. Furthermore, this technique has recently been used to detect corrupted information in an optimized RNS-based blockchain data storage mechanism [19].

To illustrate reverse conversion using the New Chinese Remainder Theorem, let us consider the following Example 8.

Example 8. In an RNS with the basis $\{5, 9, 11\}$, recover the number $X = (3, 5, 1)$ considered in Example 4. First, we find $\widetilde{P_i}$:

$$\begin{gathered} \widetilde{P_1} = p_1 = 5, \\ \widetilde{P_2} = p_2 \cdot p_3 = 99. \end{gathered}$$

Determine the constant k_0:

$$k_0 = |9^{-1}|_5 = 4.$$

Then, according to (9), we have:

$$X = 23 + |4(3 - 23)|_5 99 = 23 + 0 = 23.$$

Thus $X = (3, 5, 1) = 23$.

3.4 Approximate Chinese Remainder Theorem

In [21], the use of the residue number system for representing floating-point numbers is proposed. To obtain this result, we divide (6) by P:

$$\frac{X}{P} = \left| \sum_{i=1}^{n} X_a \cdot \frac{|P_1^{-1}|_{p_i}}{p_i} \right|_1 = \left| \sum_{i=1}^{n} X_a \cdot k_i \right|_1. \tag{10}$$

The values $k_i = \frac{|P_i^{-1}|_{p_i}}{p_i}$ are constants of the chosen system, and equation (10) yields a result in the interval $[0, 1)$. In this context, the process of determining the remainder modulo the large modulus is replaced by simply discarding the integer part, which is a straightforward operation to implement. To obtain the exact value, the fractional part is multiplied by P.

To illustrate reverse conversion using the approximate method, consider the following Example 9.

Example 9. Given the modulus set {5,9,11} and the number $X = (3, 5, 1)$. We find the constants k_i:

$$k_1 = \frac{4}{5},\ k_2 = \frac{1}{9},\ k_3 = \frac{1}{11}.$$

Then, using formula (10), we easily find:

$$\frac{X}{P} = \left| 3 \cdot \frac{4}{5} + 5 \cdot \frac{1}{9} + 1 \cdot \frac{1}{11} \right|_1 = \left| \frac{1508}{495} \right|_1 = \left| 3\frac{23}{495} \right|_1 = \frac{23}{495},$$

then

$$X = (3, 5, 1) = 23.$$

Thus, $X = (3, 5, 1) = 23$.

4. Reverse Conversion Algorithm from HPR to PNS

Using the digits X_i in positional form that satisfy the condition $\beta_{min} P_a \le X_i \le \beta_{max} P_a$ together with formula (3), the number X in the standard positional number system is computed as the weighted sum $X = \sum_{i=0}^{d-1} X_i P_a^i$. Thus, the reverse conversion problem at this stage reduces solely to evaluating this polynomial, which represents a conventional arithmetic multiply-accumulate operation.

However, for large values of d and P_a, direct sequential computation of the sum can be computationally expensive, especially when a hardware implementation with high throughput is required. To perform this operation efficiently, we propose using a finite ring neural network.

The finite ring neural network, proposed by Zhang [21], enables the computation of the weighted sum in a parallel and pipelined manner. The network architecture is organized as follows: each input neuron takes the digit X_i and scales it by the corresponding weight coefficient P_a^i. Unlike traditional

neural networks, where weights are adjusted during training, in this case the weights P_a^i are deterministic constants determined by the radix of the hybrid system. Neurons in the second layer perform pairwise summation of the weighted values, and the output neuron yields the final result X.

The use of a neural network interpretation in this case is structural in nature and allows describing the computational process in terms of layers and connections, which is convenient for designing hardware implementations. At the same time, the proposed approach is essentially a special case of a linear neural network without nonlinear activation functions.

Owing to the absence of feedback connections and the linear nature of the transformation, the network computes the sum in a single clock cycle, provided that all d neurons in the first layer operate in parallel. In this case, the bit width of the intermediate sums must be sufficient to represent the maximum value of X, which does not exceed $P = P_a^d$. This approach fully preserves the parallelism advantages for which the hybrid positional-residue number system was developed and allows reverse conversion to be performed in minimal time.

Algorithm 1 formalizes the described computation, and Fig. 2 illustrates the architecture of the finite ring neural network that implements the summation for the case where all digits X_i are already known.

Input: $(\langle X_{d-1}\rangle_{a|b}, \dots, \langle X_0\rangle_{a|b})_{HPR}$, $\{p_1, p_2, \dots, p_n\}, d, P_a$
Output: X
1. $sum = 0$
2. **for** $i = 0$ to $d - 1$ do:
2.1. $sum = sum + X_i \cdot P_a^i$
3. **return** $X = sum$

Algorithm 1. Reverse Conversion from HPR to PNS Using a Finite Ring Neural Network.

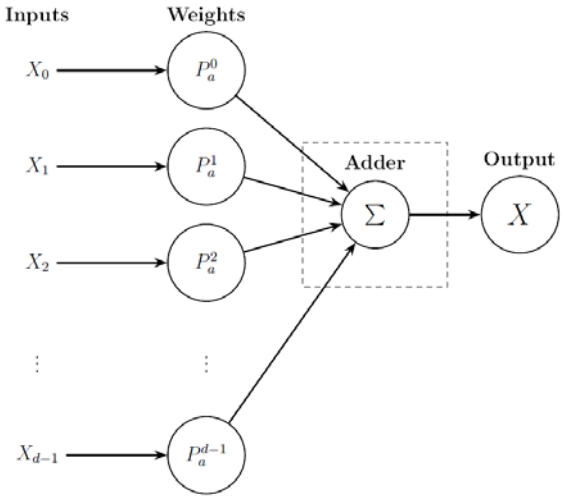

Fig. 2. Representation of reverse conversion in HPR based on a finite ring neural network.

In the case where the digits X_i are unknown, the reverse conversion of a number from HPR to the positional number system is performed in two stages. For each digit $\langle X_i\rangle_{a|b} = \left(x_{i,1}, \dots, x_{i,n_a}, x_{i,n_a+1}, \dots, x_{i,n_a+n_b}\right)$, one of the standard reverse conversion methods for RNS can be applied, for example CRT:

$$X_i = \left|\sum_{j=1}^{n_a+n_b} \left|x_{i,j} \cdot P_{a|b,j}^{-1}\right|_{p_j} \cdot P_{a|b,j}\right|_{P_a}, \quad (11)$$

where $P_{a|b} = P_a \cdot P_b, P_{a|b,j} = \frac{P_{a|b}}{p_j}$.

After recovering all digits X_i, the number X is computed using (3). If the HPR employs redundancy (e.g., β_{max}), digit normalization may be required before conversion. Furthermore, for large d and n, computations are optimized through parallel application of the approximate CRT and fast multiplication algorithms (e.g., the Karatsuba method).

To illustrate reverse conversion in HPR, consider the following example.

Example 10. Given the basis system $B_a = \{3,5\}, B_b = \{7,11\}$ and the HPR number $X_{HPR} = \big((1,3,6,2),(2,0,5,5)\big)$. The number of HPR digits is $d = 2, \beta_{min} = 0, \beta_{max} = 3$. The dynamic range is $P_a = 45$. We recover the digits using formula (15):

$$(1,3) = |5 \cdot |1 \cdot 2|_3 + 3 \cdot |3 \cdot 2|_5|_{15} = ⍰13⍰_a,$$
$$(6,2) = |11 \cdot |6 \cdot 2|_7 + 7 \cdot |2 \cdot 8|_{11}|_{77} = ⍰13⍰_b$$
$$(2,0) = |5 \cdot |2 \cdot 2|_3 + 3 \cdot |0 \cdot 2|_5|_{15} = ⍰5⍰_a,$$
$$(5,5) = |11 \cdot |5 \cdot 2|_7 + 7 \cdot |5 \cdot 8|_{11}|_{77} = ⍰5⍰_b.$$

Assembling the number using (3):

$$X = 13 \cdot 15 + 5 \cdot 15^0 = 200.$$

Therefore $X = (3,5) = 200$.

In the case where the digits X_i are unknown, the application of a finite ring neural network is also possible. However, in the subsequent performance analysis of the algorithms, we assume knowledge of the digits X_i, since otherwise the advantages of HPR are effectively nullified in comparison to RNS.

5. Performance Analysis

5.1 Asymptotic Computational Complexity Analysis

This section presents an evaluation of the time efficiency of reverse conversion algorithms from RNS to PNS and the reverse conversion algorithm from HPR to PNS based on a finite ring neural network. Two approaches are used for the analysis: asymptotic complexity and experimental evaluation of time efficiency with increasing input data size.

To properly compare the efficiency of reverse conversion methods, we introduce the following notation. Let n be the number of modular bases (channels) in the residue number system, m be the maximum modulus value in the basis (characterizing the bit width of processing), and d be the number of digits in the hybrid positional-residue number system representation.

Table 2 presents complexity estimates for the five considered methods: classical CRT, Mixed-Radix Conversion (MRC), New CRT II (NCRT), Approximate CRT (ACRT), and the proposed HPR-based method.

The performed asymptotic complexity evaluation allows us to draw the following conclusions. The MRC method exhibits a quadratic dependence on the number of moduli, $O(n^2 \cdot log^2 m)$, which makes it unsuitable for high-performance systems with a large number of channels. Methods based on CRT and ACRT demonstrate a linear dependence on n and a polylogarithmic dependence on m, providing acceptable performance for a wide range of problems. NCRT, owing to its divide-and-conquer recursive scheme, reduces the dependence on n to logarithmic while retaining a quadratic dependence on $log\, m$.

Table 2. Complexity evaluation of reverse conversion methods.

Method	Operations	Complexity Estimate	Overall Complexity
CRT	Multiplication	$O(n \cdot log\ m)$	$O(n \cdot log^2 m)$
	Addition	$O(log\ nm)$	
MRC	Modulo division	$O(log\ nm \cdot log\ m)$	$O(n^2 \cdot log^2 m)$
	Multiplication	$O(n^2 \cdot log^2 m)$	
	Addition	$O(n^2 \cdot log\ m)$	
NCRT	Recursion	$O(log\ n)$	$O(log^2 m\ \cdot \log n)$
	Multiplication	$O(log^2 m\ \cdot\ log\ n)$	
	Addition	$O(log\ m\ \cdot\ log\ n)$	
ACRT	Modulo division	$O(log\ m)$	$O(n \cdot log^2 m)$
	Division	$O(1)$	
	Multiplication	$O(n \cdot log^2 m)$	
	Addition	$O(n \cdot log\ m)$	
HPR	Multiplication	$O(d)$	$O(d \cdot log\ d)$
	Addition	$O(d)$	
	Exponentiation	$O(d \cdot log\ d)$	

The best asymptotic efficiency is demonstrated by the reverse conversion method based on HPR. Its complexity $O(d \cdot log\ d)$ depends solely on the number of digits d in the hybrid representation and is independent of the number of modular channels n. Under the condition that $d \ll n$ (which is typical for representing large numbers), this method provides a significant performance gain compared to classical RNS conversion algorithms. Thus, the application of HPR followed by reverse conversion based on a finite ring neural network enables efficient scaling of the computing system as the bit width of the processed data increases.

5.2 Experimental Performance Evaluation

To validate the theoretical findings presented in Section 5.1, we conduct a comparative analysis of the reverse conversion algorithms described in Sections 3 and 4. The goal of the experiment is to identify the most efficient method in terms of time costs under various input parameters.

All algorithms are implemented in the Python programming language. The experiments are conducted on the Windows 10 Home Edition operating system running on a computer with the following specifications: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.11 GHz processor, 8 GB DDR4 RAM at 1196 MHz, and a 512 GB SSD.

The experimental design is as follows. For reverse conversion methods from RNS to PNS, five modulo sets are used, with the number of bases ranging from 8 to 32 and the dynamic range bit width ranging from 128 to 1024 bits. For the reverse conversion method based on HPR, five sets are also used; however, the number of moduli in these sets ranges from 2 to 4 with the same output number bit width. The number of digits d is equal to 2 in all experiments.

To obtain reliable results, ten thousand iterations are performed for each method and each modulus set, minimizing measurement error. The CRT method is implemented using the built-in *crt* function

from the SymPy library. The remaining algorithms are implemented manually. The obtained results are summarized in Table 3; execution times are given in microseconds.

Table 3. Execution time of reverse conversion methods for different numbers of moduli and different dynamic range bit widths (μs).

Method	n	Bit width (bits)	Reverse conversion time		
			Max	Mean	Min
CRT	8	128	35.4	13.4918	12.8
	12	256	44.0	18.9534	18.1
	16	512	78.8	31.5677	29.7
	20	768	94.6	40.7049	38.8
	32	1024	114.5	63.9214	61.0
MRC	8	128	78.4	16.1222	15.4
	12	256	116.2001	28.9054	27.5
	16	512	167.9	62.1103	58.3
	20	768	247.5	109.1785	101.3
	32	1024	463.0	162.0373	150.5
NCRT	8	128	132.7	63.671	60.0
	12	256	382.0	156.9521	152.4
	16	512	703.5	327.7717	316.0
	20	768	1158.7	568.9794	554.4
	32	1024	1941.5	948.9353	921.9
ACRT	8	128	26.4	13.7	6.9
	12	256	52.5	13.8	6.7
	16	512	87.3	14.0	6.7
	20	768	133.2	14.5	6.5
	32	1024	192.1	16.0	6.6
HPR	4	128	36.9	0.9	0.5
	4	256	38.7	1.7	1.4
	4	512	34.8	1.6	1.1
	8	768	47.7	1.9	1.2
	8	1024	48.1	2.1	1.2

Analysis of the obtained data allows us to make the following observations. The MRC method exhibits the highest time costs, especially as the number of moduli n increases, which is consistent with its quadratic asymptotic complexity. NCRT, despite its recursive scheme, also shows relatively high execution time, lagging behind classical CRT.

CRT-based and ACRT show comparable results for small n; however, ACRT advantageously demonstrates stable average execution time (13.7–16.0 μs) regardless of increasing bit width.

The best results across all experimental scenarios are demonstrated by the HPR-based reverse conversion method. The average conversion time for this method ranges from 0.9 to 2.1 μs depending on bit width and number of moduli, with an overall average of 1.64 μs.

Quantitative comparison shows that the HPR method outperforms ACRT by an average of 88.6% and classical CRT by 95.1%. This significant gain is explained by the fact that in HPR, the main computational cost is shifted to the digit recovery stage X_i, which, under the assumption of known digits (as in the conducted experiment), reduces to linear summation, whereas classical methods require modular operations on large numbers.

Thus, the experimental results fully confirm the theoretical conclusions of Section 5.1 and demonstrate the high efficiency of the proposed approach for practical applications requiring fast reverse conversion from modular to positional representation.

5. Conclusion

In this paper, a reverse conversion algorithm from the HPR to the positional number system has been proposed and investigated. The main idea of the method consists of representing the conversion process as a weighted sum of HPR digits and implementing this operation using a structure based on a finite ring neural network, which enables efficient parallelization of computations.

A comparative analysis of the proposed approach with classical reverse conversion methods from the residue number system has been performed, including the CRT, MRC, NCRT and ACRT. The theoretical evaluation has shown that the computational complexity of the proposed method is determined by the number of digits in the hybrid representation and does not directly depend on the number of moduli, which provides better scalability when operating with large bit-width numbers.

The results of the experimental study confirm the efficiency of the proposed algorithm. It has been shown that the HPR-based method demonstrates lower execution time compared to classical reverse conversion algorithms from RNS to PNS under the considered conditions. The achieved gain is due to the reduction in the number of resource-intensive operations on large numbers and the possibility of parallel summation.

Nevertheless, it should be noted that the efficiency of the proposed approach largely depends on the method of obtaining the HPR digits. If their recovery is required, additional computational costs may affect the overall performance of the method.

Future research will be directed toward the application of HPR for high-precision computing.

References

[1]. Goldberg D. What every computer scientist should know about floating-point arithmetic. ACM Computing Surveys, 1991, vol. 23, no. 1, pp. 5-48.

[2]. Ananda Mohan P.V. Residue Number Systems: Theory and Applications. Birkhäuser, Basel, 2016, 351 p. ISBN 978-3-319-41385-3.

[3]. Molahosseini A.S., Sousa L. Introduction to residue number system: structure and teaching methodology. In: Embedded Systems Design with Special Arithmetic and Number Systems. Springer International Publishing, Cham, 2017, pp. 3-17.

[4]. Xiao H., et al. Algorithms for comparison in residue number systems. In Proc. of the 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA). IEEE, 2016, pp. 1-6.

[5]. Brönnimann H., et al. Sign determination in residue number systems. Theoretical Computer Science, 1999, vol. 210, no. 1, pp. 173-197.

[6]. Lu M., Chiang J.S. A novel division algorithm for the residue number system. IEEE Transactions on Computers, 1992, vol. 41, no. 8, pp. 1026-1032.

[7]. Szabo N.S., Tanaka R.I. Residue Arithmetic and Its Application to Computer Technology. McGraw-Hill, New York, 1967, 236 p.

[8]. Исупов К. С. Высокопроизводительные вычисления с использованием системы остаточных классов. Программные системы: теория и приложения, 2021, т. 12, № 2 (49), стр. 137-192. / Isupov K.S. High-performance computing using residue number system. Program Systems: Theory and Applications, 2021, vol. 12, issue 2 (49), pp. 137-192 (in Russian).

[9]. Bigou K., Tisserand A. Hybrid position-residues number system. In Proc. of the 23rd IEEE Symposium on Computer Arithmetic (ARITH). IEEE, 2016, pp. 126-133.

[10]. Bajard J.-C., Didier L.-S., Kornerup P. An RNS Montgomery modular multiplication algorithm. IEEE Transactions on Computers, 1998, vol. 47, no. 7, pp. 766-776.

[11]. Nozaki H., Motoyama M., Shimbo A., Kawamura S. Implementation of RSA algorithm based on RNS Montgomery multiplication. In Proc. of the International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2001, pp. 364-376.

[12]. Knuth D.E. The Art of Computer Programming. Vol. 2: Seminumerical Algorithms, 3rd ed. Addison-Wesley, 1997, 784 p. ISBN 0201896842.

[13]. Акушский И.Я., Юдицкий Д.И. Машинная арифметика в остаточных классах. М., Советское радио, 1968, 440 с. / Akushsky I. Ya., Yuditsky D. I. Computer arithmetic in residual classes. Moscow, Soviet Radio, 1968, 440 p. (in Russian).

[14]. Yassine H.M., Moore W.R. Improved mixed-radix conversion for residue number system architectures. IEE Proceedings G: Circuits, Devices and Systems, 1991, vol. 138, no. 1, pp. 120-124.

[15]. Wang Y. New Chinese remainder theorems. In Proc. of the Thirty-Second Asilomar Conference on Signals, Systems and Computers. IEEE, 1998, vol. 1, pp. 165-171.

[16]. Wang Y. Residue-to-binary converters based on new Chinese remainder theorems. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 2000, vol. 47, no. 3, pp. 197-205.

[17]. Zhang W., Siy P. An efficient design of residue to binary converter for four moduli set $\{2^n-1, 2^n+1, 2^{2n}-2, 2^{2n+1}-3\}$ based on new CRT II. Information Sciences, 2008, vol. 178, no. 1, pp. 264–279.

[18]. Molahosseini A.S., Navi K., Dadkhah C., Kavehei O., Timarchi S. Efficient reverse converter designs for the new 4-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1,\ 2^n+1, 2^{2n}, 2^{2n}+1\}$ based on new CRTs. IEEE Transactions on Circuits and Systems I: Regular Papers, 2010, vol. 57, no. 4, pp. 823--835.

[19]. Guo Z., Gao Z., Mei H., Zhao M., Yang J. Design and optimization for storage mechanism of the public blockchain based on redundant residual number system. IEEE Access, 2019, vol. 7, pp. 98546-98554.

[20]. Soderstrand M., Vernia C., Chang J.H. An improved residue number system digital-to-analog converter. IEEE Transactions on Circuits and Systems, 1983, vol. 30, pp. 903-907.

[21]. Zhang D., Jullien G.A., Miller W.C. A neural-like network approach to finite ring computations. IEEE Transactions on Circuits and Systems, 1990, vol. 37, no. 8, pp. 1048-1052.

Информация об авторах / Information about authors

Айсанат Эдуардновна ГЕРЮГОВА – студент кафедры математического анализа, алгебры и геометрии факультета математики и компьютерных наук имени профессора Н.И. Червякова ФГАОУ ВПО «Северо-Кавказский федеральный университет». Сфера научных интересов: система остаточных классов, математика и компьютерные науки.

Aisanat Eduardnovna GERYUGOVA – student at the Department of Mathematical Analysis, Algebra and Geometry, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov, North Caucasus Federal University. Her research interests include residue number systems, mathematics and computer science.

Владислав Вячеславович ЛУЦЕНКО – ассистент кафедры вычислительной математики и кибернетики факультета математики и компьютерных наук имени профессора Н.И. Червякова ФГАОУ ВПО «Северо-Кавказский федеральный университет». Сфера научных интересов: высокопроизводительные вычисления, система остаточных классов, умный город, нейронные сети, интернет вещей.

Vladislav Vyacheslavovich LUTSENKO – Assistant Professor, Department of Computational Mathematics and Cybernetics, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov, North Caucasus Federal University. Research interests: high-performance computing, residue number system, smart city, neural networks, Internet of Things.

Антон Алексеевич СИНИЦЫН – аспирант третьего года обучения по специальности 2.3.5 «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» Института системного программирования им. В.П. Иванникова Российской академии наук. Сфера научных интересов: гомоморфное шифрование, схемы гомоморфного шифрования на основе системы остаточных классов, оптимизация модулярной арифметики.

Anton Alekseevich SINITSYN – third-year postgraduate student in specialty 2.3.5 "Mathematical and Software Support of Computing Systems, Complexes, and Computer Networks" at the Ivannikov Institute for System Programming of the Russian Academy of Sciences. Research interests: homomorphic encryption, homomorphic encryption schemes based on the residue number system, optimization of modular arithmetic.