entropy

MDPI

Article

An Efficient Parallel Reverse Conversion of Residue Code to Mixed-Radix Representation Based on the Chinese Remainder Theorem

Mikhail Selianinau and Yuriy Povstenko *

Department of Mathematics and Computer Sciences, Faculty of Science and Technology, Jan Dlugosz University in Czestochowa, al. Armii Krajowej 13/15, 42-200 Czestochowa, Poland; m.selianinau@ujd.edu.pl
* Correspondence: j.povstenko@ujd.edu.pl; Tel.: +48-343-612-269

Abstract: In this paper, we deal with the critical problems in residue arithmetic. The reverse conversion from a Residue Number System (RNS) to positional notation is a main non-modular operation, and it constitutes a basis of other non-modular procedures used to implement various computational algorithms. We present a novel approach to the parallel reverse conversion from the residue code into a weighted number representation in the Mixed-Radix System (MRS). In our proposed method, the calculation of mixed-radix digits reduces to a parallel summation of the small word-length residues in the independent modular channels corresponding to the primary RNS moduli. The computational complexity of the developed method concerning both required modular addition operations and one-input lookup tables is estimated as $O(k^2/2)$, where k equals the number of used moduli. The time complexity is $O(\lceil \log_2 k \rceil)$ modular clock cycles. In pipeline mode, the throughput rate of the proposed algorithm is one reverse conversion in one modular clock cycle.

Keywords: Residue Number System; modular arithmetic; residue-to-binary conversion; Chinese Remainder Theorem; mixed-radix representation

Citation: Selianinau, M.; Povstenko, Y. An Efficient Parallel Reverse Conversion of Residue Code to Mixed-Radix Representation Based on the Chinese Remainder Theorem. *Entropy* **2022**, *24*, 242. https://doi.org/10.3390/e24020242

Academic Editors: Sergio Cruces, Rubén Martín-Clemente, Andrzej Cichocki and Iván Durán-Díaz

Received: 8 January 2022
Accepted: 3 February 2022
Published: 5 February 2022

1. Introduction

Along with the improvement of computer technology, the development and implementation of new effective approaches to the organization and realization of computational tasks are some of the main ways to increase the data processing speed. At present, high-performance computing is developing extremely rapidly. These reasons lead to qualitatively new requirements imposed on number-theoretic methods and computational algorithms. Practically, all well-known approaches to high-performance computing use certain parallel forms of data representation and processing. In recent decades, special consideration has been given to the so-called modular computational structures. Their arithmetic foundation is the Residue Number System (RNS), whose ideological roots go back to the classic topics of number theory and abstract algebra. The RNS is a non-positional number system with inherent parallelism and occupies a place of particular importance due to its carry-free properties, which provide a high potential for accelerating arithmetic operations.

As is well known, the RNS has some advantages over a conventional Weighted Number System (WNS) in the design and implementation of high-performance computing applications, devices, and systems. From its appearance in the mid-1950s to the present, RNS arithmetic has attracted the constant attention of researchers in computer technology [1,2], number-theoretic methods [3–5], digital signal and image processing [2,5–8], communications systems [5,9], cryptography [2,8,10,11], and other fields [10].

The main advantage of RNS is its unique ability to decompose the large word-length numbers into a set of smaller word-length residues, which are processed in parallel in the independent modular channels. The inherent parallelism of RNS enables avoiding the carry-overs obtained in addition, subtraction, and multiplication, which are usually

time-consuming in the WNS. In this regard, the modularity and carry-free properties make computation fast and efficient. Therefore, the RNS presents one of the most efficient means for increasing data processing speed.

Due to its carry-free property, the residue arithmetic is exceptionally suitable for a broad class of applications in which addition and multiplication are the dominant arithmetic operations. In any case, it has excellent potential for many substantial applications in such areas as digital signal processing, cryptography, distributed information and communication systems, information security systems, fault tolerance, cloud computing, and others. Moreover, these RNS applications may be effectively embedded in processor platforms functioning according to the conventional information-processing approach [2,5,8]. For the reasons mentioned above, residue arithmetic represents an efficient mathematical tool for the high-speed implementation of various computational tasks.

The reverse conversion and base extension are the most critical topics in residue arithmetic. As opposed to conventional WNS, these operations, on a par with other central non-modular procedures such as magnitude comparison, sign determination, overflow detection, general division, scaling, etc., are relatively harder for implementation. They are time consuming and costly due to their more complicated structure compared to modular operations.

As is known, to perform non-modular operations, it is necessary to carry out the binary reconstruction of the integer by its residue code, which in general is hampered by the non-weighted nature of the RNS. This circumstance negates to a substantial extent the main advantages of residue arithmetic.

Therefore, the development of novel approaches and methods for fast number reconstruction by its residue code has significant importance in high-performance computing based on parallel algorithmic structures of RNS, especially for high-speed implementing digital signal processing applications and public-key cryptosystems. That should enable the extensive use of residue arithmetic in many priority areas of science and technology.

In this paper, we present a novel approach to the parallel reverse conversion from the residue code into the mixed-radix representation. In the proposed method, the calculation of mixed-radix digits reduces to a parallel summation of the small word-length residues in the independent modular channels corresponding to the primary RNS moduli.

The paper is structured as follows. Sections 2 and 3 discuss the basic theoretical concepts of the research. Section 4 describes the mathematical background of the proposed reverse conversion method. Sections 5 and 6 present a numerical example and an analysis of the computational cost, respectively. Section 7 provides discussion, and Section 8 concludes the paper.

2. The Basic Concepts of the Residue Arithmetic

The abstract algebra and number theory create the theoretical basis of the residue arithmetic [12,13].

An RNS is defined by an ordered set $\{m_1, m_2, \ldots, m_k\}$ of k pairwise relatively prime moduli, where each modulus $m_i \geq 2$ $(i = 1, 2, \ldots, k)$, and the greatest common divisor of m_i and m_j equals 1, i.e., $\gcd(m_i, m_j) = 1$ for $i \neq j$. For convenience, we assume that the default order of moduli is ascending, i.e., $m_1 < m_2 < \cdots < m_k$.

In the given RNS, it is possible to represent M_k integer numbers, where M_k is the product of all moduli, $M_k = \prod_{i=1}^{k} m_k$. Therefore, the set $\mathbf{Z}_{M_k} = \{0, 1, \ldots, M_k - 1\}$ is usually used as an RNS dynamic range.

Every number $X \in \mathbf{Z}_{M_k}$ has a unique representation in the form of a k-tuple of small integers $(\chi_1, \chi_2, \ldots, \chi_k)$, which is called a residue code, where χ_i is a least non-negative remainder of a division of X by m_i $(i = 1, 2, \ldots, k)$. We can notationally write this relation as $\chi_i = |X|_{m_i}$, where $\chi_i \in \mathbf{Z}_{m_i} = \{0, 1, \ldots, m_i - 1\}$.

The main advantage of the residue arithmetic over conventional binary arithmetic consists of parallel carrying out addition, subtraction, and multiplication at the level of small word-length residues. The modular operations $\circ \in \{+, -, \times\}$ on integers

$A = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ and $B = (\beta_1, \beta_2, \ldots, \beta_k)$ are performed independently in each modular channel in compliance with the computational rule:

$$A \circ B = (\alpha_1, \alpha_2, \ldots, \alpha_k) \circ (\beta_1, \beta_2, \ldots, \beta_k) =$$

$$= \left(|\alpha_1 \circ \beta_1|_{m_1}, |\alpha_2 \circ \beta_2|_{m_2}, \ldots, |\alpha_k \circ \beta_k|_{m_k} \right), \quad (1)$$

where $\alpha_i = |A|_{m_i}$ and $\beta_i = |B|_{m_i}, i = 1, 2, \ldots, k$.

In other words, the arithmetic operations on long-word operands are decomposed into modular channels with operands that are no larger than the corresponding modulus. Moreover, all the modular channels are entirely independent of each other. The carry-free nature of modular operations (1) is one of the most attractive features of residue arithmetic [1,3,8].

Therefore, compared with the conventional WNS, the RNS simplifies and speeds up the addition and multiplication operations. This fundamental advantage of the residue arithmetic strongly appears in the case of implementing computational procedures, which mainly contain long segments consisting of only sequences of modular arithmetic operations. In this case, the primary moduli set is chosen so that the final results of the computational procedure always belong to the used dynamic range for any allowed values of input operands. At the same time, the intermediate results can even exceed the boundaries of the dynamic range.

Along with the carry-free modular operations, there are also the so-called non-modular operations such as residue-to-binary conversion, base extension, magnitude comparison, sign determination, overflow detection, general division, scaling, etc. These operations are complicated and quite time consuming, and their significant computational complexity limits the applications of the residue arithmetic and restricts its widespread usage for high-speed computing.

To perform the non-modular operations, it is required to consider all residues in the k-tuple $(\chi_1, \chi_2, \ldots, \chi_k)$. Furthermore, it is necessary to determine the integer value of the number by its residue code, which in general is hampered by the non-positional nature of the RNS. The crucial problem of efficient implementation of non-modular operations is constantly receiving considerable attention by modern researchers [2,5,8].

The applicability of residue arithmetic is mainly determined by the computational complexity and feasibility of non-modular operations, which are used as a basis for implementing more complex computational algorithms in RNS. At the same time, the fundamental problem in the residue arithmetic, which unfortunately up to now is yet completely unresolved; it consists of reducing the computational complexity of non-modular operations. Due to a lack of efficient methods and algorithms for non-modular operations implementation, the residue arithmetic is mainly suitable when the modular additions and multiplications make up the bulk of required computations. In this case, the number of used non-modular operations is relatively small. This circumstance bounds the widespread use of the RNS to a narrow class of specific tasks.

3. Reverse Conversion of the Residue Code to Conventional Representation

The root problem of residue arithmetic is that the weighted value of the integer X depends on all the residues $\chi_1, \chi_2, \ldots, \chi_k$. The reconstruction of an integer by its residue code, i.e., the reverse conversion, is one of the most difficult non-modular operations in residue arithmetic. Moreover, this operation underlies all the other non-modular procedures.

Despite the currently extensive studies on residue arithmetic and its applications, there is a need to develop novel efficient approaches and methods of an integer number reconstruction by its residue code. This should enable us the extensive use of residue arithmetic for high-speed computing in many priority fields, first of all, in various digital signal processing and cryptographic applications.

There are two canonical techniques of reverse conversion: the canonical method based on the Chinese Remainder Theorem (CRT) and the residue code conversion to a

weighted representation in the Mixed-Radix System (MRS) [1,2,5,8,14–18]. In general, all other conversion methods represent different variants of these two methods.

Below, we describe the mathematical background of these methods.

3.1. CRT-Base Conversion Method

When the moduli $m_1, m_2, \ldots, m_k$ are pairwise relatively prime, the integer number X and its residue code $(\chi_1, \chi_2, \ldots, \chi_k)$ are related by the equation:

$$X = \left| \sum_{i=1}^{k} M_{i,k} \chi_{i,k} \right|_{M_k}, \tag{2}$$

where $M_{i,k} = M_k / m_i$, $\chi_{i,k} = \left| M_{i,k}^{-1} \chi_i \right|_{m_i}$ is a normalized residue modulo m_i $(i = 1, 2, \ldots, k)$, $\left| Y^{-1} \right|_m$ denotes the multiplicative inverse of an integer Y modulo m.

In essence, Equation (2) represents the CRT [10,19,20].

In the last decades, considerable efforts are directed to reducing the complexity of the CRT implementation and the possibility of its application in high-speed computing [2,5,8,21–23]. The main idea of these methods is to replace the inner multiplications and additions modulo M_k with simpler operations (see (2)).

Consider the CRT-number

$$X_k = \sum_{i=1}^{k} M_{i,k} \chi_{i,k}. \tag{3}$$

As follows from (2), the difference $X_k - X$ is a multiple of M_k. Therefore, the following exact integer equality holds

$$X = X_k - \rho_k(X) M_k. \tag{4}$$

The unique integer number $\rho_k(X)$ is a normalized rank (or, briefly, rank) of the number X [3,4,7].

Equation (4) is called a rank form of the integer X. In essence, the rank $\rho_k(X)$ is a reconstruction coefficient that indicates how many times the dynamic range M_k is exceeded when converting the residue code $(\chi_1, \chi_2, \ldots, \chi_k)$ to the integer X.

In contrast to (2), Equation (4) does not contain a very time-consuming reduction modulo M_k. Therefore, when we have the efficient method for the rank $\rho_k(X)$ computation, the reverse conversion algorithm constructed on the basis of (4) has a substantial lead over the canonical CRT implementation (2).

3.2. MRS-Base Conversion Method

In the MRS defined by a set $\{m_1, m_2, \ldots, m_k\}$ of pairwise relatively prime moduli, the integer $X \in \mathbf{Z}_{M_k}$ is represented by the k-tuple $(x_k, x_{k-1}, \ldots, x_1)$ of mixed-radix digits, resulting in

$$X = x_1 + x_2 M_1 + x_3 M_2 + \cdots + x_k M_{k-1} = \sum_{i=1}^{k} x_i M_{i-1}, \tag{5}$$

where $x_i \in \mathbf{Z}_{m_i}$ $(i = 1, 2, \ldots, k)$ [1,2,8].

It is well known that the MRS surpasses the RNS when performing non-modular operations such as magnitude comparison, sign determination, and overflow detection. Therefore, the mixed-radix representation has received the widest appliance for the implementation of non-modular procedures along with the other generally accepted integral characteristics of the residue code such as the rank of a number, core function, interval index, parity function, diagonal, and quotient functions [3,4,7,24–33].

The RNS-to-MRS reverse conversion establishes an association between the residue code $(\chi_1, \chi_2, \ldots, \chi_k)$ of the number X and its mixed-radix representation $(x_k, x_{k-1}, \ldots, x_1)$. The mixed-radix digits x_i $(i = 1, 2, \ldots, k)$ in (5) are computed according to the following calculation relations [1]:

$$x_1 = \chi_1,$$

$$x_2 = \left|(\chi_2 - x_1)\left|m_1^{-1}\right|_{m_2}\right|_{m_2},$$

$$x_3 = \left|\left((\chi_3 - x_1)\left|m_1^{-1}\right|_{m_3} - x_2\right)\left|m_2^{-1}\right|_{m_3}\right|_{m_3},$$

$$\ldots$$

$$x_k = \left|\left(\ldots\left((\chi_k - x_1)\left|m_1^{-1}\right|_{m_k} - x_2\right)\left|m_2^{-1}\right|_{m_k} - \cdots - x_{k-1}\right)\left|m_{k-1}^{-1}\right|_{m_k}\right|_{m_k}.$$

This sequential calculation procedure called a chained algorithm can be written in the general form

$$x_i = \left|X^{(i)}\right|_{m_i}, \tag{6}$$

where

$$X^{(i)} = \begin{cases} X, \text{if } i = 1, \\ \left(X^{(i-1)} - x_{i-1}\right)m_{i-1}^{-1}, \text{if } i = 2, 3, \ldots, k. \end{cases} \tag{7}$$

From (6) and (7), it follows that the considered computational process requires two modular operations: subtraction and multiplication by the multiplicative inverse. Thus, the most crucial advantage of this algorithm is its high modularity. However, its strictly sequential nature prevents general use for the construction of appropriate high-performance parallel computing procedures.

4. A Novel CRT-Base RNS-to-MRS Reverse Conversion Method

Now, we describe a proposed new method for calculating mixed-radix digits $x_1, x_2, \ldots, x_k$ of the number X by its residue code $(\chi_1, \chi_2, \ldots, \chi_k)$.

Consider the CRT-number X_k. According to (3), we have

$$X_k = \sum_{i=1}^{k-1} M_{i,k-1} m_k \chi_{i,k} + M_{k-1}\chi_{k,k}. \tag{8}$$

By Euclid's Division Lemma, the integer $m_k\chi_{i,k}$ can be written as

$$m_k\chi_{i,k} = \chi_{i,k-1} + \left\lfloor \frac{m_k\chi_{i,k}}{m_i} \right\rfloor m_i, \tag{9}$$

where

$$\chi_{i,k-1} = \left|m_k\chi_{i,k}\right|_{m_i} = \left|m_k\left|M_{i,k}^{-1}\chi_i\right|_{m_i}\right|_{m_i} = \left|m_k M_{i,k}^{-1}\chi_i\right|_{m_i} = \left|M_{i,k-1}^{-1}\chi_i\right|_{m_i},$$

$\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

Substituting (9) into (8), we obtain

$$X_k = X_{k-1} + M_{k-1}S_k(X), \tag{10}$$

where

$$X_{k-1} = \sum_{i=1}^{k-1} M_{i,k-1}\chi_{i,k-1}, \tag{11}$$

$$S_k(X) = \sum_{i=1}^{k} R_{i,k}(\chi_i), \tag{12}$$

$$R_{i,k}(\chi_i) = \left\lfloor \frac{m_k \chi_{i,k}}{m_i} \right\rfloor \ (i = 1, 2, \ldots, k). \tag{13}$$

Taking into account (9), we have

$$R_{i,k}(\chi_i) = \frac{m_k \chi_{i,k} - \chi_{i,k-1}}{m_i}.$$

Since $R_{i,k}(\chi_i) \in \mathbf{Z}_{m_k}$, we can reduce the right side of equality modulo m_k. Hence, the residue $R_{i,k}(\chi_i)$ can be calculated as

$$R_{i,k}(\chi_i) = \left| -\frac{\chi_{i,k-1}}{m_i} \right|_{m_k} = \left| -\frac{\left| M_{i,k-1}^{-1} \chi_i \right|_{m_i}}{m_i} \right|_{m_k} \ (i = 1, 2, \ldots, k-1). \tag{14}$$

At the same time, from (13) it follows that

$$R_{k,k}(\chi_k) = \chi_{k,k} = \left| M_{k,k}^{-1} \chi_k \right|_{m_k} = \left| M_{k-1}^{-1} \chi_k \right|_{m_k}. \tag{15}$$

Similarly, taking into account Equations (10)–(13), the numbers X_i $(i = k-1, k-2, \ldots, 1)$ can be written by turns as

$$X_{k-1} = X_{k-2} + M_{k-2} S_{k-1}(X),$$

$$X_{k-2} = X_{k-3} + M_{k-3} S_{k-2}(X),$$

$$\ldots$$

$$X_2 = X_1 + M_1 S_2(X),$$

$$X_1 = M_0 S_1(X),$$

where $M_0 = 1$, $S_1(X) = \chi_1$, the integers $S_l(X)$ $(l = 2, 3, \ldots, k)$ are calculated according to (12)–(15) in the case when the index k is replaced by l.

Finally, substituting the above equations for X_l $(l = k-1, k-2, \ldots, 1)$ by turns into (10), we obtain

$$X_k = \sum_{i=1}^{k} M_{l-1} S_l(X). \tag{16}$$

At the same time, according to Euclid's Division Lemma, we have

$$S_l(X) = R_l(X) + m_l Q_l(X), \tag{17}$$

where $R_l(X) = |S_l(X)|_{m_l}$ and $Q_l(X) = \lfloor S_l(X)/m_i \rfloor$ are the remainder and quotient of the division $S_l(X)$ by the modulus m_l, respectively.

Therefore, taking into account (12), when the index k is replaced by l, the integers $R_l(X)$ and $Q_l(X)$ can be computed as

$$R_l(X) = \left| \sum_{i=1}^{l} R_{i,l}(\chi_i) \right|_{m_l}, \tag{18}$$

$$Q_l(X) = \left\lfloor \frac{1}{m_l} \sum_{i=1}^{l} R_{i,l}(\chi_i) \right\rfloor. \tag{19}$$

From (19), it follows that $Q_l(X)$ equals the number of occurred overflows when calculating the sum $R_l(X)$ of residues $R_{1,l}(\chi_1), R_{2,l}(\chi_2), \ldots, R_{l,l}(\chi_l)$ modulo m_l $(l = 2, 3, \ldots, k)$.

Note that $R_1(X) = \chi_1$ and $Q_1(X) = 0$ since $S_1(X) = \chi_1$.

Substituting (17) into (16), we obtain

$$X_k = X_k^{(R)} + X_{k-1}^{(Q)} + M_k Q_k(X), \tag{20}$$

where

$$X_k^{(R)} = \sum_{l=1}^{k} M_{l-1} R_l(X), \tag{21}$$

$$X_{k-1}^{(Q)} = \sum_{l=1}^{k-1} M_l Q_l(X). \tag{22}$$

Let us draw attention to Equations (21) and (22). It is evident that the number $X_k^{(R)}$ is represented by the k-tuple $\left(x_k^{(R)}, x_{k-1}^{(R)}, \ldots, x_1^{(R)}\right)$ of mixed-radix digits, where $x_l^{(R)} = R_l(X)$, $l = 1, 2, \ldots, k$ (see Equation (5)). At the same time, $x_l^{(R)} \in \mathbf{Z}_{m_l}$ and $X_k^{(R)} \leq M_k - 1$.

Bearing in mind that $Q_1(X) = 0$, the number $X_{k-1}^{(Q)}$ can be written as

$$X_{k-1}^{(Q)} = \sum_{l=1}^{k-1} M_{l-1} Q_l'(X), \tag{23}$$

where $Q_1'(X) = 0$, $Q_2'(X) = Q_1(X) = 0$, and $Q_l'(X) = Q_{l-1}(X)$ for $l \geq 3$. Therefore, taking into account (19), the integer $Q_l'(X)$ can be calculated as

$$Q_l'(X) = \left\lfloor \frac{1}{m_{l-1}} \sum_{i=1}^{l-1} R_{i,l-1}(\chi_i) \right\rfloor \ (l = 3, 4, \ldots, k). \tag{24}$$

Hence, $Q_l'(X) < l - 1$ since $R_{i,l-1}(\chi_i) \leq m_{l-1} - 1$.

Thus, the integer $X_{k-1}^{(Q)}$ (see Equations (23) and (5)) can be represented by a k-tuple $\left(x_k^{(Q)}, x_{k-1}^{(Q)}, \ldots, x_1^{(Q)}\right)$ of mixed-radix digits under the condition that $x_l^{(Q)} \in \mathbf{Z}_{m_l} (l = 1, 2, \ldots, k)$, where $x_1^{(Q)} = x_2^{(Q)} = 0, x_l^{(Q)} = Q_l'(X)$ for $l > 2$. Consequently, that entails the fulfillment of the condition $\mathbf{Z}_{l-1} \subset \mathbf{Z}_{m_l}$, which leads to inequality

$$m_l \geq l - 1 \ (l = 1, 2, \ldots, k). \tag{25}$$

Thus, when the moduli set $\{m_1, m_2, \ldots, m_k\}$ meets the conditions (25), we have that $X_{k-1}^{(Q)} < M_k$.

Note that the integer $X_{k-1}^{(Q)}$ is a multiple of the number $M_2 = m_1 m_2$ because of $x_1^{(Q)} = x_2^{(Q)} = 0$ (see Equation (5)).

Now, let us return to Equation (20). According to Euclid's Division Lemma, the sum of two mixed-radix numbers $X_k^{(R)}$ and $X_{k-1}^{(Q)}$ results in

$$X_k^{(R)} + X_{k-1}^{(Q)} = \left| X_k^{(R)} + X_{k-1}^{(Q)} \right|_{M_k} + M_k \left\lfloor \frac{X_k^{(R)} + X_{k-1}^{(Q)}}{M_k} \right\rfloor. \tag{26}$$

Hence, substituting (26) into (20), we obtain

$$X_k = \left|X_k^{(R)} + X_{k-1}^{(Q)}\right|_{M_k} + M_k\left(Q_k(X) + \left\lfloor \frac{X_k^{(R)} + X_{k-1}^{(Q)}}{M_k} \right\rfloor\right). \tag{27}$$

Taking into account the rank form of the number X (4), from (27) we have

$$X = \left|X_k^{(R)} + X_{k-1}^{(Q)}\right|_{M_k}. \tag{28}$$

From (28), it follows that the mixed-radix representation of the number X, i.e., k-tuple $(x_k, x_{k-1}, \ldots, x_1)$, can be calculated as a result of the addition of two mixed-radix numbers $X_k^{(R)} = \left(x_k^{(R)}, x_{k-1}^{(R)}, \ldots, x_1^{(R)}\right)$ and $X_{k-1}^{(Q)} = \left(x_k^{(Q)}, x_{k-1}^{(Q)}, \ldots, x_1^{(Q)}\right)$ (see (21) and (23)) in the basis $\{m_1, m_2, \ldots, m_k\}$. Note that $x_1^{(R)} = \chi_1$, $x_1^{(Q)} = x_2^{(Q)} = 0$. At the same time, the digits $x_2^{(R)}, x_3^{(R)}, \ldots, x_k^{(R)}$ and $x_3^{(Q)}, x_4^{(Q)}, \ldots, x_k^{(Q)}$ are calculated as the sum of the residues $R_{1,l}(\chi_1), R_{2,l}(\chi_2), \ldots, R_{l,l}(\chi_l)$ modulo m_l along with the counting of occurred overflows according to (18) and (24) ($l = 2, 3, \ldots, k$).

Therefore, the mixed-radix digits $x_l^{(R)}$ and $x_l^{(Q)}$ are computed as

$$x_1^{(R)} = \chi_1, \;\; x_l^{(R)} = \left|\sum_{i=1}^{l} R_{i,l}(\chi_i)\right|_{m_l} \;\; (l = 2, 3, \ldots, k), \tag{29}$$

$$x_1^{(Q)} = x_2^{(Q)} = 0, \;\; x_l^{(Q)} = \left\lfloor \frac{1}{m_{l-1}} \sum_{i=1}^{l-1} R_{i,l-1}(\chi_i) \right\rfloor \;\; (l = 3, 4, \ldots, k), \tag{30}$$

where

$$R_{i,l}(\chi_i) = \left| -\frac{\left|M_{i,l-1}^{-1}\chi_i\right|_{m_i}}{m_i} \right|_{m_l} \;\; (i \neq l), \tag{31}$$

$$R_{l,l}(\chi_l) = \left|M_{l-1}^{-1}\chi_l\right|_{m_l} \;\; (l = 2, 3, \ldots, k). \tag{32}$$

Furthermore, in the MRS with the bases $m_1, m_2, \ldots, m_k$, we calculate the sum of two numbers $X_k^{(R)}$ and $X_{k-1}^{(Q)}$. As a result, we obtain the mixed-radix representation $(x_k, x_{k-1}, \ldots, x_1)$ of the number X.

Table 1 given below presents the pre-calculation components (see Equations (31) and (32)). It should be recalled that $\langle R_{1,1}(\chi_1)\rangle = \chi_1$. The abbreviation LUT means lookup table. The bit-length of residues is $b_l = \lceil \log_2 m_l \rceil$ ($l = 1, 2, \ldots, k$). Here, and further, $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

Table 1. The pre-calculation components.

Input Residue	Number and Skope of LUTs	Output Residue Set
χ_1	$k-1$, $2^{b_1} \times b_l$ ($l = 2, 3, \ldots, k$)	$\langle R_{1,2}(\chi_1), R_{1,3}(\chi_1), \ldots, R_{1,k}(\chi_1)\rangle$
χ_2	$k-1$, $2^{b_2} \times b_l$ ($l = 2, 3, \ldots, k$)	$\langle R_{2,2}(\chi_2), R_{2,3}(\chi_2), \ldots, R_{2,k}(\chi_2)\rangle$
$\ldots$	$\ldots$	$\ldots$
χ_{k-1}	2, $2^{b_{k-1}} \times b_l$ ($l = k-1, k$)	$\langle R_{k-1,k-1}(\chi_{k-1}), R_{k-1,k}(\chi_{k-1})\rangle$
χ_k	1, $2^{b_k} \times b_k$	$\langle R_{k,k}(\chi_k)\rangle$

Table 2 presents the results of calculations in the modular channels according to Equations (29) and (30). It should be reminded that in the first modular channel corresponding to the modulus m_1, the calculations are not carried out, so $x_1^{(R)} = \chi_1$ and $x_2^{(Q)} = 0$.

Table 2. The results of calculations in the modular channels.

Modular Channel	Input Data	Output Data
m_2	$\langle R_{1,2}(\chi_1),\ R_{2,2}(\chi_2)\rangle$	$x_2^{(R)},\ x_3^{(Q)}$
m_3	$\langle R_{1,3}(\chi_1),\ R_{2,3}(\chi_2),\ R_{3,3}(\chi_3)\rangle$	$x_3^{(R)},\ x_4^{(Q)}$
...	...	...
m_{k-1}	$\langle R_{1,k-1}(\chi_1),\ R_{2,k-1}(\chi_2),\ \ldots,\ R_{k-1,k-1}(\chi_{k-1})\rangle$	$x_{k-1}^{(R)},\ x_k^{(Q)}$
m_k	$\langle R_{1,k}(\chi_1),\ R_{2,k}(\chi_2),\ \ldots,\ R_{k,k}(\chi_k)\rangle$	$x_k^{(R)}$

The stated above allows us to formulate the following substantial theorem.

Theorem 1. (About RNS-to-MRS reverse conversion).

Let an arbitrary RNS be defined by an ascending-ordered set of k pairwise relatively prime moduli $m_1, m_2, \ldots, m_k$ $(m_l \geq l-1,\ l = 1,2,\ldots,k,\ k \geq 2)$, *and let the residue code* $(\chi_1, \chi_2, \ldots, \chi_k)$ *of the number* $X \in \mathbf{Z}_{M_k}$ *be given. Then, the mixed-radix representation* $(x_k, x_{k-1}, \ldots, x_1)$ *of the number X can be computed as a result of the summation of two mixed-radix numbers, namely, the appropriate number* $X_k^{(R)} = \left(x_k^{(R)}, x_{k-1}^{(R)}, \ldots, x_1^{(R)}\right)$ *and the correction number* $X_{k-1}^{(Q)} = \left(x_k^{(Q)}, x_{k-1}^{(Q)}, \ldots, x_1^{(Q)}\right)$, *where the digits* $x_l^{(R)}$ *and* $x_l^{(Q)}$ $(l = 1,2,\ldots,k)$ *are calculated according to (29) and (30), respectively, taking into account (31) and (32).*

5. A Numerical Example of the Proposed Conversion Method

The main idea of the proposed approach to reverse conversion is illustrated below by a simple numerical example. For convenience, we consider a four-moduli RNS.

Example 1. *Let the RNS moduli-set be* $\{m_1, m_2, m_3, m_4\} = \{5,7,9,11\}$. *Suppose that we wish to calculate the digits of the mixed-radix representation* (x_4, x_3, x_2, x_1) *of the given number X by its residue code* $(\chi_1, \chi_2, \chi_3, \chi_4) = (3,6,4,2)$.

Step 1. *The calculation of the primitive constants in a given RNS.*

$M_4 = 3465,\ M_3 = 315,\ M_2 = 35,\ M_1 = 5,\ M_0 = 1,$

$M_{1,4} = 693,\ M_{2,4} = 495,\ M_{3,4} = 385,\ M_{4,4} = 315,$

$\left|M_{1,4}^{-1}\right|_{m_1} = 2,\ \left|M_{2,4}^{-1}\right|_{m_2} = 3,\ \left|M_{3,4}^{-1}\right|_{m_3} = 4,\ \left|M_{4,4}^{-1}\right|_{m_4} = 8,$

$\left|m_1^{-1}\right|_{m_4} = 9,\ \left|m_2^{-1}\right|_{m_4} = 8,\ \left|m_3^{-1}\right|_{m_4} = 5,\ \left|M_3^{-1}\right|_{m_4} = 8,$

$M_{1,3} = 63,\ M_{2,3} = 45,\ M_{3,3} = 35,$

$\left|M_{1,3}^{-1}\right|_{m_1} = 2,\ \left|M_{2,3}^{-1}\right|_{m_2} = 5,\ \left|M_{3,3}^{-1}\right|_{m_3} = 8,$

$\left|m_1^{-1}\right|_{m_3} = 2,\ \left|m_2^{-1}\right|_{m_3} = 4,\ \left|M_2^{-1}\right|_{m_3} = 8,$

$M_{1,2} = 7,\ M_{2,2} = 5,$

$\left|M_{1,2}^{-1}\right|_{m_1} = 3,\ \left|M_{2,2}^{-1}\right|_{m_2} = 3,$

$\left|m_1^{-1}\right|_{m_2} = 3,\ \left|M_1^{-1}\right|_{m_2} = 3.$

Step 2. *The calculation of the residue sets* $\langle R_{1,l}(\chi_1), R_{2,l}(\chi_2), \ldots, R_{l,l}(\chi_l)\rangle$ *according to (31) and (32)* $(l = 1,2,3,4)$.

We obtain

$R_{1,1}(\chi_1) = \chi_1 = 3,$

$R_{1,2}(\chi_1) = |-|1 \cdot 3|_5 \cdot 3|_7 = 5,$

$R_{2,2}(\chi_2) = |3 \cdot 6|_7 = 4,$

$R_{1,3}(\chi_1) = |-|3 \cdot 3|_5 \cdot 2|_9 = 1,$

$R_{2,3}(\chi_2) = |-|3 \cdot 6|_7 \cdot 4|_9 = 2,$

$R_{3,3}(\chi_3) = |8 \cdot 4|_9 = 5,$

$R_{1,4}(\chi_1) = |-|2 \cdot 3|_5 \cdot 9|_{11} = 2,$

$R_{2,4}(\chi_2) = |-|5 \cdot 6|_7 \cdot 8|_{11} = 6,$
$R_{3,4}(\chi_3) = |-|8 \cdot 4|_9 \cdot 5|_{11} = 8,$
$R_{4,4}(\chi_4) = |8 \cdot 2|_{11} = 5.$
As a result, the following sets of residues occur
$\langle R_{1,1}(\chi_1)\rangle = \langle 3\rangle,$
$\langle R_{1,2}(\chi_1),\ R_{2,2}(\chi_2)\rangle = \langle 5,4\rangle,$
$\langle R_{1,3}(\chi_1),\ R_{2,3}(\chi_2),\ R_{3,3}(\chi_3)\rangle = \langle 1,2,5\rangle,$
$\langle R_{1,4}(\chi_1),\ R_{2,4}(\chi_2),\ R_{3,4}(\chi_3),\ R_{4,4}(\chi_4)\rangle = \langle 2,6,8,5\rangle.$

Step 3. *The summation of the residues* $R_{1,l}(\chi_1),\ R_{2,l}(\chi_2),\ldots,R_{l,l}(\chi_l)$ *modulo* m_l *along with the counting of occurring overflows according to (18) and (19), respectively* $(l = 2,3,4)$.

Recall that $R_1(X) = R_{1,1}(\chi_1) = 3$, *and* $Q_1(X) = 0$. *We have*
$R_2(X) = |5+4|_7 = |9|_7 = 2,$
$R_3(X) = |1+2+5|_9 = |8|_9 = 8,$
$R_4(X) = |2+6+8+5|_{11} = |21|_{11} = 10,$
$Q_2(X) = \lfloor (5+4)/7 \rfloor = \lfloor 9/7 \rfloor = 1,$
$Q_3(X) = \lfloor (1+2+5)/9 \rfloor = \lfloor 8/9 \rfloor = 0,$
$Q_4(X) = \lfloor (2+6+8+5)/11 \rfloor = \lfloor 21/11 \rfloor = 1.$

Therefore, the mixed-radix representations of the numbers $X_4^{(R)}$ *and* $X_3^{(Q)}$ *(see (21) and (23)) are computed:*
$\left(x_4^{(R)}, x_3^{(R)}, x_2^{(R)}, x_1^{(R)}\right) = (R_4(X), R_3(X), R_2(X), R_1(X)) = (10,8,2,3),$
$\left(x_4^{(Q)}, x_3^{(Q)}, x_2^{(Q)}, x_1^{(Q)}\right) = (Q_3(X), Q_2(X), 0, 0) = (0,1,0,0).$

Step 4. *The calculation of the mixed-radix digits* (x_4, x_3, x_2, x_1).

The addition of two numbers $X_4^{(R)} = (10,8,2,3)$ *and* $X_3^{(Q)} = (0,1,0,0)$ *according to (28) gives the mixed-radix representation* $(0,0,2,3)$ *of the number* X.

Let us now verify the obtained result. According to (5), we have

$$X = (0,0,2,3) = 0 \cdot 315 + 0 \cdot 35 + 2 \cdot 5 + 3 = 13.$$

This result holds because the residue code of the integer number $X = 13$ *is* $3,6,4,2$, *since* $|13|_5 = 3$, $|13|_7 = 6$, $|13|_9 = 4$, $|13|_{11} = 2$. *Thus, this result coincides with the condition of the example.*

6. The Computational Cost of the Reverse Conversion Method

As it follows from the results mentioned above, the calculation of the mixed-radix digits $x_1, x_2, \ldots, x_k$ reduces to the independent and parallel summation of small residues $R_{1,l}(\chi_1), R_{2,l}(\chi_2), \ldots, R_{l,l}(\chi_l)$ modulo m_l in lth modular channel $(l = 1,2,\ldots,k)$, taking into account the number of the overflows occuring during the modular addition operations (see (29)–(32)).

Let us evaluate the time required to perform the parallel reverse conversion.

First, we consider the calculation of mixed-radix digits of the numbers $X_k^{(R)} = \left(x_k^{(R)}, x_{k-1}^{(R)}, \ldots, x_1^{(R)}\right)$ and $X_{k-1}^{(Q)} = \left(x_k^{(Q)}, x_{k-1}^{(Q)}, \ldots, x_1^{(Q)}\right)$ (see (29) and (30)). As can be seen, there are no modular addition operations in the first modular channel corresponding to the modulus m_1. In the second channel, we have only one addition operation modulo m_2. Furthermore, two additions modulo m_3 are performed in the third channel and so on. Thus, in the lth modular channel, we have $l-1$ additions modulo m_l $(l = 2,3,\ldots,k)$. These calculations are easily parallelized and pipelined. Therefore, the required computation time for calculating digits $x_l^{(R)}$ and $x_l^{(Q)}$ is $T_l = \lceil \log_2 l \rceil$ modular clock cycles.

Thus, the time for obtaining the mixed-radix representations of the numbers $X_k^{(R)}$ and $X_{k-1}^{(Q)}$ is determined by the time in the kth modular channel and equals $T_k = \lceil \log_2 k \rceil$ modular clock cycles.

The summation of $X_k^{(R)}$ and $X_{k-1}^{(Q)}$ on the bases $\{m_1, m_2, \ldots, m_k\}$ involves two additional modular clock cycles taking into account the inter-digit carries. Therefore, the

execution time of the reverse conversion equals $T_{conv} = T_k + 2$ modular clock cycles. Thus, the overall time is $t_{conv} = T_{conv} t_{mod}$, where t_{mod} denotes the modular clock cycle time. At the same time, when pipelined, the throughput rate of the proposed conversion method is one conversion in one modular clock cycle.

Consider now the evaluation of the required computational cost. Due to the small word-length of residues in the k-tuple $(\chi_1, \chi_2, \ldots, \chi_k)$, the pre-computation and lookup table techniques are suitable for reverse conversion implementation. So, we can use one-input lookup tables depending on the residues word-length in each modular channel.

At the beginning stage of the reverse conversion, in the lth channel corresponding to the modulus m_l, the number of lookup tables required to store the residue set $\langle R_{1,l}(\chi_1), R_{2,l}(\chi_2), \ldots, R_{l,l}(\chi_l) \rangle$ equals $N_{lut}(l) = l$. At the same time, the word length of recorded residues is $b_l = \lceil \log_2 m_l \rceil$ bits $(l = 2, 3, \ldots, k)$. In the first modular channel, $N_{lut}(1) = 0$ since $S_1(X) = \chi_1$.

Then, the overall number of one-input lookup tables in all modular channels is equal to

$$N_{lut} = \sum_{l=2}^{k} N_{lut}(l) = \frac{k^2 + k - 2}{2}.$$

The summation of the residues $R_{1,l}(\chi_1), R_{2,l}(\chi_2), \ldots, R_{l,l}(\chi_l)$ modulo m_l requires $N_{add}(l) = l - 1$ modular addition operations $(l = 2, 3, \ldots, k)$. At the same time, all independent calculations are realized in parallel in corresponding modular channels.

Taking into account that $x_1^{(Q)} = x_2^{(Q)} = 0$, the summation of two numbers $X_k^{(R)} = \left(x_k^{(R)}, x_{k-1}^{(R)}, \ldots, x_1^{(R)}\right)$ and $X_{k-1}^{(Q)} = \left(x_k^{(Q)}, x_{k-1}^{(Q)}, \ldots, x_1^{(Q)}\right)$ on the final stage of the reverse conversion requires $2(k-2)$ modular addition operations.

Hence, the overall number of modular addition operations in all modular channels is equal to

$$N_{add} = \sum_{l=2}^{k} N_{add}(l) + 2(k-2) = \frac{k^2 + 3k - 8}{2}.$$

When pipelined, the throughput rate of the proposed method is one reverse conversion in one modular clock cycle.

7. Discussion

As it follows from [1], the calculation of the mixed-radix digits $x_1, x_2, \ldots, x_k$ (see (6) and (7)) requires $k - 1$ both addition and multiplication operations; in this case, the overall conversion time is $k(k-1)/2 \cdot (t_{add} + t_{mul})$, where t_{add} and t_{mul} denote an execution time of addition/subtraction and multiplication, respectively. The computational cost of the pipelined implementation of this algorithm is $k(k-1)/2$, both multiplication and addition operations, while the conversion time is $(k-1)(t_{add} + t_{mul})$. The main drawback of this method is its strictly sequential nature.

The parallel conversion method circumscribed in [16] uses the additional lookup tables. At the same time, $k(k+1)/2$ lookup tables and $k(k+1)/2$ adders are required. The conversion time is $t_{lut} + (k-1)t_{add}$ due to the need to generate the inter-digit carries when performing addition operations. As noted in [34], the method proposed in [16] does not allow obtaining the claimed depth of $O(\log_2 k)$ in terms of RNS processing elements. In this regard, an improved method was proposed by adding extra $k(k+1)/2$ multipliers to hardware resources used in [16]. The implementation time is $t_{lut} + t_{mul} + (2\log_2 k + 1)t_{add}$. Hence, the time complexity of this conversion algorithm is $O(\log_2 k)$.

In [15], the mixed-radix conversion is realized by the cascaded scheme of lookup tables and adders. The computational cost for the sequential implementation is $k(k-2)/4$ double-size lookup tables and $k(k-2)/4$ adders, while the conversion time equals $(k/2) \cdot (t_{lut} + t_{add})$. When pipelined, the throughput rate is determined by the time equals

$t_{lut} + t_{add}$. This method works well when the used moduli do not have a very large word-length, since the size of lookup tables increases significantly with a word-length growth.

The paper [17] presents the parallel reverse conversion method, which uses the lookup table technique and requires no arithmetic or logical units. As reported, this algorithm is better than the ones presented in [15,16]. It is based on solving $k(k-1)/2$ linear Diophantine Equations and requires $k(k-1)/2$ lookup tables of size $m_i \times m_j$, while a conversion time is $(k-1)t_{lut}$. When pipelined, its effective conversion rate is one conversion per t_{lut}. So, this method is attractive for DSP implementation. However, it is not suitable for implementing cryptographic applications because of the enormous size of the required lookup tables, especially when processing large numbers.

In the paper [9], the reverse conversion method is based on modular reduction by a modified canonic CRT algorithm. This enables minimizing the bit-width of intermediate data processing. The lookup tables translate the b_i-bit input residues $(i = 1, 2, \ldots, k)$ into b_{out}-bit output integers, where $b_i = \lceil \log_2 m_i \rceil$, $b_{out} = \left\lceil \frac{1}{2} \log_2 \left(\sum_{i=1}^{k} b_i \right) \right\rceil$, and k is the number of RNS moduli. As a result, the modular reduction of the modified k-tuple of b_{out}-bits integers is carried out over a ring of size $2b_{out}$ such that only the b_{out} least significant bits of the binary representation are maintained. In this case, all the b_{out}-bit outputs in the modified k-tuple are added together by adder tree without regard to overflow, propagating the b_{out} least significant bits to the output. The reverse conversion requires k lookup tables and $k-1$ adders. The scope of used lookup tables is $2^b \times 2^{b_{out}}$, $b \in \{b_1, b_2, \ldots, b_k\}$. The overall conversion time is $t_{lut} + \lceil \log_2 k \rceil \, t_{add}$.

Some reverse conversion methods use the special moduli sets with a limited number of moduli, such as $m = 2^n + d$ $(d \in \{-1, 0, 1\})$ [2,8,35–40]. Their main drawback consists in a small number of the selected moduli, typically from three to five. These moduli sets are suitable for the efficient implementations of DSP algorithms but completely not applicable for large numbers processing widely used in cryptography. For example, to represent 1024-bit word-length cryptographic numbers using four RNS moduli, each modular channel must have residues of 256-bit length, which is not qualified for high-performance computing.

Table 3 compares the results across multiple techniques of the reverse conversion. Here, we use the following abbreviations: LUT—lookup table, ADD–adder, MUL—multiplier. The bit length $b \in \{b_1, b_2, \ldots, b_k\}$, $b_l = \lceil \log_2 m_l \rceil$ $(l = 1, 2, \ldots, k)$.

Table 3. RNS-to-MRS reverse conversion methods.

Method	Number and Scope of LUTs	ADD	MUL	Conversion Time
[1], sequential	–	$k-1$	$k-1$	$\frac{k(k-1)}{2}(t_{mul} + t_{add})$
[1], sequential, pipelined	$\frac{k(k-1)}{2}$; $2^{(b+1)} \times b$	$\frac{k(k-1)}{2}$	–	$(k-1)(t_{lut} + t_{add})$
[16], parallel	$\frac{k(k+1)}{2}$; $2^b \times b$	$\frac{k(k+1)}{2}$	–	$t_{lut} + (k-1)t_{add}$
[34], parallel	$\frac{k(k+1)}{2}$; $2^b \times b$	$\frac{k(k+1)}{2}$	$\frac{k(k+1)}{2}$	$t_{lut} + t_{mul} + (2\log_2 k + 1)t_{add}$
[15], sequential	$\frac{k(k-2)}{4}$; $2^{2b} \times 2b$	$\frac{k(k-1)}{4}$	–	$\frac{k}{2}(t_{lut} + t_{add})$
[15], parallel	$\frac{k(k-2)}{4} + k - 1$; $2^{2b} \times 2b$	$\frac{k(k+2)}{4} - 3$	–	$t_{lut} + \frac{k}{2}t_{add}$
[17], parallel	$\frac{k(k-1)}{2}$; $2^{2b} \times b$	–	–	$(k-1)t_{lut}$
[9]	k; $2^b \times 2^{\lceil \frac{1}{2}\log_2(kb) \rceil}$	$k-1$	–	$t_{lut} + (\lceil \log_2 k \rceil)t_{add}$
Our method, parallel	$\frac{k^2+k-2}{2}$; $2^b \times b$	$\frac{k^2+3k-8}{2}$	–	$(\lceil \log_2 k \rceil + 2)t_{mod}$

As seen from above, the proposed parallel reverse conversion method has time complexity of the order $O(\lceil \log_2 k \rceil)$. In pipelined mode, it enables the high throughput rate and has one reverse conversion in one modular clock cycle. At the same time, the computational complexity is of the order of $O(k^2/2)$ in terms of the number of both required arithmetic operations and one-input lookup tables.

8. Conclusions

In this paper, a novel approach to parallel reverse conversion of the residue code $(\chi_1, \chi_2, \ldots, \chi_k)$ of the number X to mixed-radix representation $(x_k, x_{k-1}, \ldots, x_1)$ is described.

The calculation of the mixed-radix digits $(x_k, x_{k-1}, \ldots, x_1)$ is reduced to a parallel summation of the small word-length residues $R_{1,l}(\chi_1)$, $R_{2,l}(\chi_2)$, …, $R_{l,l}(\chi_l)$ modulo m_l in lth modular channel $(l = 1, 2, \ldots, k)$, taking into account the number of the overflows occuring during the modular addition operations. These modular operations are performed fast and independently in each modular channel and easily pipelined.

The computational cost of the proposed reverse conversion method is presented. In all modular channels, the general number of modular addition operations is equal to $N_{add} = \left(k^2 + 3k - 8\right)/2$. At the same time, the summary number of reqiured one-input lookup tables makes up $N_{lut} = \left(k^2 + k - 2\right)/2$.

The execution time of the reverse conversion equals $T_{conv} = \lceil \log_2 k \rceil + 2$ modular clock cycles. At the same time, when pipelined, the throughput rate of the proposed conversion method is one conversion in one modular clock cycle.

The proposed parallel reverse conversion method coincides with the development vector of modern high-performance computing using residue arithmetic. It can find a widespread application for implementing a broad class of tasks in various areas of science and technology, first of all, in digital signal processing and cryptography.

Author Contributions: Conceptualization, M.S.; investigation, Y.P. ; methodology, M.S.; writing—original draft preparation, M.S.; writing—review and editing, Y.P. All authors have read and improved the final version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Szabo, N.S.; Tanaka, R.I. *Residue Arithmetic and its Application to Computer Technology*; McGraw-Hill: New York, NY, USA, 1967.
2. Molahosseini, A.S.; de Sousa, L.S.; Chang, C.H. (Eds.) *Embedded Systems Design with Special Arithmetic and Number Systems*; Springer: Cham, Switzerland, 2017.
3. Akushskii, I.Y.; Juditskii, D.I. *Machine Arithmetic in Residue Classes*; Soviet Radio: Moscow, Russia, 1968. (In Russian)
4. Amerbayev, V.M. *Theoretical Foundations of Machine Arithmetic*; Nauka: Alma-Ata, Kazakhstan, 1976. (In Russian)
5. Omondi, A.R.; Premkumar, B. *Residue Number Systems: Theory and Implementation*; Imperial College Press: London, UK, 2007.
6. Soderstrand, M.A.; Jenkins, W.K.; Jullien, G.A.; Taylor, F.J. (Eds.) *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*; IEEE Press: New York, NY, USA, 1986.
7. Chernyavsky, A.F.; Danilevich, V.V.; Kolyada, A.A.; Selyaninov, M.Y. *High-Speed Methods, and Systems of Digital Information Processing*; Belarusian State University: Minsk, Belarus, 1996. (In Russian)
8. Ananda Mohan, P.V. *Residue Number Systems. Theory and Applications*; Springer: Cham, Switzerland, 2016.
9. Michaels, A.J. A maximal entropy digital chaotic circuit. In Proceedings of the 2011 IEEE International Symposium of Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, 15–18 May 2011; pp. 717–720.
10. Ding, C.; Pei, D.; Salomaa, A. *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*; World Scientific: Singapore, 1996.
11. Omondi, A.R. *Cryptography Arithmetic: Algorithms and Hardware Architectures*; Springer: Cham, Switzerland, 2020.
12. Burton, D.M. *Elementary Number Theory*, 7th ed.; McGraw-Hill: New York, NY, USA, 2011.
13. Hardy, G.H.; Wright, E.M. *An Introduction to the Theory of Numbers*, 6th ed.; Oxford University Press: London, UK, 2008.
14. Akkal, M.; Siy, P. A new mixed radix conversion algorithm MRC-II. *J. Syst. Archit.* **2007**, *53*, 577–586. [CrossRef]
15. Chakraborti, N.B.; Soundararajan, J.S.; Reddy, A.L.N. An implementation of mixed-radix conversion for residue number applications. *IEEE Trans. Comput.* **1986**, *35*, 762–764. [CrossRef]

16. Huang, C.H. Fully parallel mixed-radix conversion algorithm for residue number applications. *IEEE Trans. Comput.* **1983**, *32*, 398–402. [CrossRef]
17. Miller, D.F.; McCormick, W.S. An arithmetic free parallel mixed-radix conversion algorithm. *IEEE Trans. Circuits Syst. II* **1998**, *45*, 158–162. [CrossRef]
18. Yassine, H.M.; Moore, W.R. Improved mixed-radix conversion for residue number architectures. *IEE Proc. G - Circuits Devices Syst.* **1991**, *138*, 120–124. [CrossRef]
19. Knuth, D.E. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed.; Addison-Wesley: Boston, MA, USA, 1998.
20. Shoup, V. *A Computational Introduction to Number Theory and Algebra*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2005.
21. Phatak, D.S.; Houston, S.D. New distributed algorithms for fast sign detection in residue number systems (RNS). *J. Parallel Distrib. Comput.* **2016**, *97*, 78–95. [CrossRef]
22. Shenoy, M.A.P.; Kumaresan, R. A fast and accurate RNS scaling technique for high speed signal processing. *IEEE Trans. Acoust. Speech Signal Process.* **1989**, *37*, 929–937. [CrossRef]
23. Vu, T.V. Efficient implementations of the Chinese Remainder Theorem for sign detection and residue decoding. *IEEE Trans. Comput.* **1985**, *34*, 646–651.
24. Miller, D.D.; Altschul, R.E.; King, J.R.; Polky, J.N. Analysis of the residue class core function of Akushskii, Burcev, and Pak. In *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*; IEEE Press: Piscataway, NJ, USA, 1986; pp. 390–401.
25. Gonnella, J. The application of core functions to residue number system. *IEEE Trans. Signal Process.* **1991**, *39*, 69–75. [CrossRef]
26. Abtahi, M. Core function of an RNS number with no ambiguity. *Comput. Math. Appl.* **2005**, *50*, 459–470. [CrossRef]
27. Kong, Y.; Asif, S.; Khan, M.A.U. Modular multiplication using the core function in the residue number system. *Appl. Algebra Eng. Commun. Comput.* **2016**, *27*, 1–16. [CrossRef]
28. Kolyada, A.A.; Selyaninov, M.Y. Generation of integral characteristics of symmetric-range residue codes. *Cybern. Syst. Anal.* **1986**, *22*, 431–437. [CrossRef]
29. Selianinau, M. An efficient implementation of the CRT algorithm based on an interval-index characteristic and minimum-redundancy residue code. *Int. J. Comput. Meth.* **2020**, *17*, 2050004. [CrossRef]
30. Lu, M.; Chiang, J.-S. A novel division algorithm for the residue number system. *IEEE Trans. Comput.* **1992**, *41*, 1026–1032. [CrossRef]
31. Dimauro, G.; Impedovo, S.; Modugno, R.; Pirlo, G.; Stefanelli, R. Residue-to-binary conversion by the "quotient function". *IEEE Trans. Circuits Syst. II Analog Digital Signal Process.* **2003**, *50*, 488–493. [CrossRef]
32. Dimauro, G.; Impedovo, S.; Pirlo, G.; Salzo, A. RNS architectures for the implementation of the 'diagonal function'. *Inf. Process. Lett.* **2000**, *73*, 189–198. [CrossRef]
33. Pirlo, G.; Impedovo, D. A new class of monotone functions of the residue number system. *Int. J. Math. Models Meth. Appl. Sci.* **2013**, *7*, 802–809.
34. Hitz, M.A.; Kaltofen, E. Integer division in residue number systems. *IEEE Trans. Comput.* **1995**, *44*, 983–989. [CrossRef]
35. Bergerman, M.V.; Lyakhov, P.A.; Voznesensky, A.S.; Bogaevskiy, D.V.; Kaplun, D.I. Designing reverse converter for data transmission systems from two-level RNS to BNS. *J. Phys. Conf. Ser.* **2020**, *1658*, 012005. [CrossRef]
36. Daphni, S.; Vijula Grace, K.S. A review analysis of reverse converter based on RNS in signal processing. *Int. J. Sci. Technol. Res.* **2020**, *9*, 1686–1689.
37. Sousa, L.; Paludo, R.; Martins, P.; Pettenghi, H. Towards the integration of reverse converters into the RNS channels. *IEEE Trans. Comput.* **2020**, *69*, 342–348. [CrossRef]
38. Mojahed, M.; Molahosseini, A.S.; Zarandi, A.A.E. A multifunctional unit for reverse conversion and sign detection based on the 5-moduli set. *Comp. Sci.* **2021**, *22*, 101–121. [CrossRef]
39. Salifu, A. New reverse conversion for four-moduli set and five-moduli set. *J. Comp. Commun.* **2021**, *9*, 57–66. [CrossRef]
40. Taghizadeghankalantari, M.; TaghipourEivazi, S. Design of efficient reverse converters for Residue Number System. *J. Circuits Syst. Comp.* **2021**, *30*, 2150141. [CrossRef]