applied sciences

MDPI

Article

Multiple Error Correction in Redundant Residue Number Systems: A Modified Modular Projection Method with Maximum Likelihood Decoding

Mikhail Babenko [1,2], Anton Nazarov [1], Maxim Deryabin [3], Nikolay Kucherov [1], Andrei Tchernykh [2,4,5,*], Nguyen Viet Hung [6], Arutyun Avetisyan [2] and Victor Toporkov [7]

1 North Caucasus Center for Mathematical Research, North-Caucasus Federal University, 355017 Stavropol, Russia; mgbabenko@ncfu.ru (M.B.); anazarov@ncfu.ru (A.N.); nkucherov@ncfu.ru (N.K.)
2 Institute for System Programming of the Russian Academy of Sciences, 109004 Moscow, Russia; arut@ispras.ru
3 Computing Platform Lab, Samsung Advanced Institute of Technology, Suwon 16678, Korea; max.deriabin@samsung.com
4 Computer Science Department, CICESE Research Center, Ensenada 22860, Mexico
5 School of Electrical Engineering and Computer Science, South Ural State University, 454080 Chelyabinsk, Russia
6 Department of Information Security, Le Quy Don Technical University, Hanoi 11917, Vietnam; hungnv@lqdtu.edu.vn
7 Computer Science Department, National Research University "MPEI", 111250 Moscow, Russia; ToporkovVV@mpei.ru
* Correspondence: chernykh@cicese.mx; Tel.: +7-906-440-0219

Abstract: Error detection and correction codes based on redundant residue number systems are powerful tools to control and correct arithmetic processing and data transmission errors. Decoding the magnitude and location of a multiple error is a complex computational problem: it requires verifying a huge number of different possible combinations of erroneous residual digit positions in the error localization stage. This paper proposes a modified correcting method based on calculating the approximate weighted characteristics of modular projections. The new procedure for correcting errors and restoring numbers in a weighted number system involves the Chinese Remainder Theorem with fractions. This approach calculates the rank of each modular projection efficiently. The ranks are used to calculate the Hamming distances. The new method speeds up the procedure for correcting multiple errors and restoring numbers in weighted form by an average of 18% compared to state-of-the-art analogs.

Keywords: redundant residue number system; error correction codes; multiple errors; Chinese Remainder Theorem

Citation: Babenko, M.; Nazarov, A.; Deryabin, M.; Kucherov, N.; Tchernykh, A.; Hung, N.V.; Avetisyan, A.; Toporkov, V. Multiple Error Correction in Redundant Residue Number Systems: A Modified Modular Projection Method with Maximum Likelihood Decoding. *Appl. Sci.* **2022**, *12*, 463. https://doi.org/10.3390/app12010463

Academic Editor: Arcangelo Castiglione

Received: 12 November 2021
Accepted: 29 December 2021
Published: 4 January 2022

1. Introduction

Despite significant efforts, both in industry and science, high fault tolerance remains a serious problem in the management of large-scale IT systems.

The consequences of failures, regardless of their causes, can be eliminated using various methods by introducing data redundancy. Erasure codes, replication, and Resilient Distributed Dataset (RDD) [1] are the most important methods for ensuring fault tolerance in modern data storage and processing systems.

Replication is the simplest-to-implement (therefore, most widespread) method of introducing redundancy [2]. However, replication leads to significant overhead costs expressed in additional resources for duplicating data and functionality. In turn, erasure codes [3,4] are devoid of this drawback. Such codes counteract the partial loss of data and are based on a principle of antinoise coding (parity check). Erasure codes save significant costs and energy [3]. However, data recovery after failures is associated with high reconstruction

costs and network traffic. The transition from replication to erasure codes and RDD is complicated by the need to update the hardware–software base and recode the entire volume of stored data. Nevertheless, there are examples of successful adoption of these technologies by Facebook [5] and Microsoft Azure [3].

This paper proposes a new approach to error correction codes based on a redundant residue number system (RRNS) [6,7]. Along with the Reed–Solomon codes [8], the Rabin dispersal algorithm [9], etc., RRNSs provide the functionality implemented by erasure codes and the functionality implemented by RDD algorithms. In addition, an important property of RNSs is the ability to perform arithmetic operations over encoded data. This property can be used in modern distributed data processing systems [10–12] and encrypted data processing schemes [13]. Therefore, RRNSs are a unique and versatile tool for a wide range of applications.

There are two main strategies for correcting errors in RRNSs. The first one involves syndrome decoding [14–16]: erroneous parts of the data are determined by comparing a special numerical syndrome calculated on the obtained data with the reference values. The second strategy is to restore error-free data based on the obtained code [17–19] by calculating the error magnitude and correcting the code during the decoding process. The authors [19] suggested a maximum likelihood decoding (MLD) approach to simplify the corrective decoding method by reducing the enumeration of values during decoding.

This paper proposes a modified method for corrective data decoding in RRNSs using MLD. Unlike the decoding method [19], this approach involves an alternative version of the Chinese Remainder Theorem (CRT) [20–22] for more efficient data restoration due to parallel execution of decoding and error correction. The CRT with fractions allows dividing the decoding process into two independent parts: determining the weighted characteristic of a number (a relative estimate of its magnitude) and decoding itself. The weighted characteristic can be directly used for error correction, and data decoding can be performed in parallel. The proposed modified modular projection method with MLD speeds up the procedure of error correction and number restoration in the weighted number system compared to the original method [19]. At the same time, it consumes many more hardware resources.

This paper is organized as follows: In Section 2, the existing research in the field of error correction in RRNSs is surveyed. A detailed introduction to RNSs, including the notations and preliminaries used below, is given in Section 3. Additionally, this section briefly describes MLD [19]. Section 4 presents the proposed method and analyzes its complexity. Section 5 is devoted to the hardware implementation of the proposed method, comparing it with the existing methods. The outcomes of this paper are summarized in Section 6.

2. Related Works

Error detection and correction codes based on a redundant residue number system (RRNS) are modern and powerful tools to control and correct arithmetic processing and data transmission errors. Algorithms that correct only single errors were considered and improved by several authors [6,23–25]. Decoding the magnitude and location of a single error is significantly less complex than detecting and correcting errors in multiple bits of RRNSs. It requires verifying a huge number of different possible combinations of erroneous residual digit positions in the error localization stage. The localization stage consumes the most time in the procedure for correcting multiple errors in RRNSs. All known methods for detecting and correcting multiple errors in RRNSs can be divided into three large groups: continuous fractions [26,27], syndrome decoding [14–16], and modular projections [17–19]. These groups of methods have advantages and drawbacks.

The methods with continuous fractions are based on the Euclidean recursive algorithm. After localization, erroneous remainders are corrected by expanding the moduli system using error-free remainders. For multiple errors in residual digits of large bit depth, the

recursive Euclidean algorithm needs many iterations and becomes inefficient in hardware implementation.

The paper [16] presented an approach to correcting multiple errors in RRNSs using syndromes. Later, it was improved in [14,15]. According to [15], the residual digits in syndrome decoding are divided into three groups; seven categories of error positions are determined among them. The error magnitudes are distributed in three syndromes, which are used to categorize the errors accurately and localize the erroneous residual digits simultaneously from the six lookup tables. Syndromes can be calculated in parallel, and operations are performed in small moduli. This feature is an undoubted advantage of the approach, providing high-speed execution of the error correction procedure for RRNSs. Among the drawbacks of this method, note a rather strong constraint on the choice of redundant moduli:

$$2m_1m_2 \dots m_k < m_{k+1}m_{k+2} \dots m_n,$$

where k denotes the number of information moduli in an RRNS, and n is the total number of RRNS moduli. A critical shortcoming of methods with syndrome decoding is that the volume of lookup tables sharply grows with an increase in the error multiplicity and the bit width of the residual digits. Searching through large lookup tables is time consuming and negates the benefit of efficiently calculated syndromes. Thus, methods with syndrome decoding are efficient only for small moduli sets.

The method with modular projections was pioneered in [17] and improved and extended to the case of multiple errors in the subsequent works [18,28]. With this approach, erroneous residual digits are localized by reducing the RRNS, deleting the number of remainders equal to the error multiplicity. The value represented in the RRNS falls into the legitimate range after excluding the erroneous residual digits. This process is similar to syndrome decoding. It requires estimating each possible combination of residual digit error locations. Methods with modular projections need no large lookup tables: this is an absolute advantage over methods with syndrome decoding. Other advantages include a weaker constraint on the choice of redundant moduli than in the case of syndrome decoding:

$$m_i < m_{k+1}, m_{k+2}, \dots, m_n, \forall i \leq k,$$

where k denotes the number of information moduli in an RRNS, and n is the total number of RRNS moduli. A critical shortcoming of such methods is the number of modular projections equal to C_n^t, where t denotes the maximum error multiplicity. Obviously, the number of modular projections grows fast with an increase in the error multiplicity, raising the number of computations accordingly.

The paper [19] proposed an essentially different method with modular projections: the number of projections was reduced by changing their construction procedure. The idea is to delete $r = n - k$ remainders instead of t ones (the conventional approach). The algorithm for constructing modular projections and their number was refined in [29]. Since $r > t$, the number of modular projections is significantly reduced, but it becomes necessary to check the Hamming distances according to maximum likelihood decoding (MLD). This approach eliminates the critical shortcoming of the rapidly growing number of projections and seems an optimal choice for correcting multiple errors in modular codes. The ideas and approaches [19] underlie the modified modular projection method with MLD; see Section 4. Using the Chinese Remainder Theorem with fractions [20–22] opens up new opportunities for parallelizing the algorithm and speeding up the procedure for correcting RRNS errors. Section 5 presents the hardware implementations of the original method [19] and the modified modular projection method with MLD on Field-Programmable Gate Array (FPGA). The hardware implementations partially involve the authors' computing modules described in the earlier publications [30,31].

3. Residue Number System and Multiple Error Correction

3.1. Residue Number System

A residue number system (RNS) [7,32,33] is a non-weighted number system. In an RNS, an integer X is represented by a k-dimensional vector $(x_1,\ x_2,\ \ldots\ ,\ x_k)$ of the remainders in dividing it by positive coprimes $(m_1,\ m_2,\ \ldots,\ m_k)$ called the RNS moduli. The magnitude of X belongs to the range $[0,\ M_K - 1]$, where

$$M_K = \prod_{i=1}^{k} m_i,$$

and each remainder x_i is the least nonnegative modulo m_i residue of X:

$$x_i = X \mathrm{mod} m_i \left(\text{equivalently, } |X|_{m_i}\right). \tag{1}$$

Note that when obtaining the RNS representation of X, the modulus remainders are calculated independently and in parallel. The paper [30] considered the most effective methods for calculating the remainder of the division and presented schematic diagrams for their hardware implementation.

The inverse conversion, i.e., restoring the weighted representation of a number from its RNS representation, is a much more difficult task than the direct conversion. The weighted representation of a number (its weighted characteristic) can be found using the Chinese Remainder Theorem (CRT) [7], the CRT with fractions (CRTf) [20–22], or conversion to a mixed radix number system (MRNS) [34,35].

The modified multiple error correction method proposed below partially involves the CRT and CRTf.

3.2. Chinese Remainder Theorem

According to the Chinese Remainder Theorem (CRT), an integer X is calculated from its RNS representation $(x_1,\ x_2,\ \ldots,\ x_k)$ as follows [7]:

$$X = \left| \sum_{i=1}^{k} \frac{M_K}{m_i} \left| \frac{m_i}{M_K} \right|_{m_i} x_i \right|_{M_K} \tag{2}$$

or equivalently,

$$X = \sum_{i=1}^{k} \frac{M_K}{m_i} \left| \frac{m_i}{M_K} \right|_{m_i} x_i - r_X M_K \tag{3}$$

where $\left| \frac{m_i}{M_K} \right|_{m_i}$ denotes the multiplicative inversion of $\left| \frac{M_K}{m_i} \right|_{m_i}$, and r_X is the number rank (a value showing how many times the sum in (3) exceeds the RNS range).

3.3. Chinese Remainder Theorem with Fractions

A number X is converted to a weighted number system (WNS) using the Chinese Remainder Theorem with fractions (CRTf) [4,24,31]:

$$X = F(X) M_K, \tag{4}$$

where

$$F(X) = \left| \sum_{i=1}^{k} k_i^* x_i \right|_1,\ k_i^* = \frac{|m_i / M_K|_{m_i}}{m_i}, \tag{5}$$

and $|\bullet|_1$ denotes taking the fractional part of a number.

The value $F(X)$ is called the approximate weighted characteristic of X in the RNS representation.

We propose using a binary shift of fractions to implement this algorithm on a hardware base not supporting fractions (e.g., FPGA). The accuracy of the constants k_i^* in (5), sufficient

for correctly reconstructing the weighted representation of X, was thoroughly estimated in [20]. A number XX is converted to a WNS using the CRTf with the binary shift as follows:

$$X = \frac{F(X)M_K}{2^N}, \tag{6}$$

where

$$F(X) = \left| \sum_{i=1}^{k} k_i^* x_i \right|_{2^N}, \; k_i^* = \frac{|m_i / M_K|_{m_i}}{m_i} \cdot 2^N \tag{7}$$

and $N = \log_2 \left(M_K \sum_{i=1}^{k} (m_i - 1) \right)$ [4].

The CRTf allows converting RNS representations of numbers to the WNS ones efficiently on any computing platform: all calculations are reduced to operations well implemented in hardware. These operations include addition, multiplication, binary shifts, and discarding the most significant digits of a number [31].

3.4. Redundant Residue Number System

The residual representation of an error-free integer is unique for the range $[0, \; M_K - 1]$. This range is used for detecting and correcting errors in the RNS. Expanding the code space of the residual representation r with additional residual bits, we form an $(n, \; k)$-redundant residue number system (RRNS) with correcting properties.

An RRNS with $n = \; k + \; r$ is based on coprime moduli $(m_1, m_2, \ldots, m_k, m_{k+1}, \ldots, m_n)$ and the full range $M_N = \prod_{i=1}^{n} m_i$. The additional moduli, $m_{k+1}, m_{k+2}, \ldots, m_{k+r}$, are called the redundant moduli of the RRNS. The range $[0, \; M_K - 1]$ corresponding to the first k information moduli from the n-dimensional vector is called the legitimate range. The range $[M_K, \; M_N - 1]$ corresponding to the additional r redundant (control) moduli is called the illegitimate range. The legitimate range is the required computational range of the number system, while the illegitimate range is necessary for error detection, localization, and correction.

When an error occurs, an integer $X = (x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n)$ within the legitimate range is converted to $X' = X + E$, where EE is the error. The value X' falls into the illegitimate range if

1) $m_{k+1}, m_{k+2}, \ldots, m_{k+r} > m_i, \forall i \leq k$,
2) the number of erroneous remainders does not exceed r.

Thus, to detect an error, we should restore the number XX from its RRNS representation $(x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n)$. If the resulting value is smaller than the dynamic range of the RRNS ($X < M_K M$), the number X contains no errors of multiplicity r and below; otherwise, the RRNS representation $(x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n)$ of XX contains 1 or 2 or 3,..., or r erroneous residual digits.

The number of erroneous residual digits detected and corrected is determined by the number of redundant moduli added. In the general case without special restrictions, the $(n, \; k)$-RRNS with $r = \; n - k$ redundant moduli detects r and corrects $t = r/2$ erroneous residual digits [7].

3.5. Modular Projection Method

The modular projection method is used to correct erroneous residual digits of numbers in the RRNS representation. Let an error be detected in a number X' written in the RRNS. The $(n, \; k)$-RRNS allows correcting $t = (n - k)/2$ erroneous residual digits [7]. According to the classical modular projection method, t or fewer errors are corrected in the following steps:

1) Constructing C_n^t modular projections. A modular projection is obtained by deleting t remainders in the original RRNS representation of the number. Deleting different sets of t remainders, we construct different modular projections.

2) Obtaining the weighted representation (weighted characteristic) of each modular projection. The weighted representation of a number is its magnitude in the WNS. The weighted characteristic of a number depends on the inverse RNS-to-WNS conversion method.
3) Comparing the weighted representation (weighted characteristic) of each modular projection with the value (weighted characteristic) of the dynamic range. A modular projection whose weighted representation (weighted characteristic) does not exceed that of the dynamic range contains the correct remainders. Note that there can be C_{n-i}^{t-i} correct modular projections, where i is the number of erroneous digits, $i = 1, \ldots, t$. In this case, the remainders deleted when constructing each correct projection will be erroneous.
4) Calculating the remainders on dividing the correct modular projection's weighted representation by the moduli corresponding to the erroneous remainders. Note that correcting erroneous remainders is not required to decode correct data from the RRNS: the weighted representation of a correct modular projection is the corrected number written in the WNS. When using the weighted characteristic of a modular projection in step 3), it is necessary to perform an additional step for decoding: obtain the weighted representation of the number from its weighted characteristic.

For each projection, calculations are performed in parallel, which significantly reduces the total time to correct errors in RRNS codes. A critical drawback of this method is the fast-growing number of modular projections with an increase in the error multiplicity. An increase in the number of modular projections does not affect the total execution time of the error-correction procedure in RRNS codes due to the parallel computations for each projection. However, it sharply raises hardware costs [18]. This critical drawback can be overcome using the concept of maximum likelihood decoding (MLD) [19].

3.6. Modular Projection Method with MLD

The paper [19] improved the classical modular projection method to reduce the number of projections by changing their construction. When constructing a modular projection, the idea is to delete not r residual bits of the (n, k)-RRNS instead of t ones, where $t = (n-k)/2$ is the maximum multiplicity of an error corrected by the (n, k)-RRNS, and $r = n - k$ is the number of redundant moduli. The algorithm for constructing modular projections was described in detail in [29].

Such an approach to constructing modular projections allows for reducing their number to $C_{n/2}^{t}$ [29]. However, some incorrect projections fall into the legitimate RRNS range. To separate correct projections, we propose using maximum likelihood decoding (MLD) [19]. According to the MLD concept, an additional step is to calculate the Hamming distances between the erroneous number in the RRNS representation and each modular projection. In the RNS, the Hamming distance is defined as the number of distinct corresponding remainders. Let $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ be numbers written in the RNS with bases $m_1, m_2, \ldots, m_n$. Then the Hamming distance is given by

$$h(X, Y) = \sum_{i=1}^{n} f(x_i, y_i), \text{ where } f(a, b) = \begin{cases} 1 \text{ if } a \neq b, \\ 0 \text{ if } a = b. \end{cases} \quad (8)$$

Therefore, we should find the remainders of the moduli deleted when constructing each projection.

Two conditions determine a correct projection:

- The modular projection X_i' in the WNS (its weighted characteristic $F(X_i')$) is smaller than the RRNS dynamic range M_K (its weighted characteristic $F(M_K)$).
- The Hamming distance between the distorted number $X\prime X'$ and the modular projection X_i' in the RRNS does not exceed the value of the maximum multiplicity t of errors corrected by this RRNS.

The following formula combines these conditions:

$$X = X_i' : \begin{cases} F(X_i') < F(M_K) (\text{or } X_i' < M_K), \\ h(X', X_i') \le t, \end{cases} \quad \text{where } t = (n-k)/2. \tag{9}$$

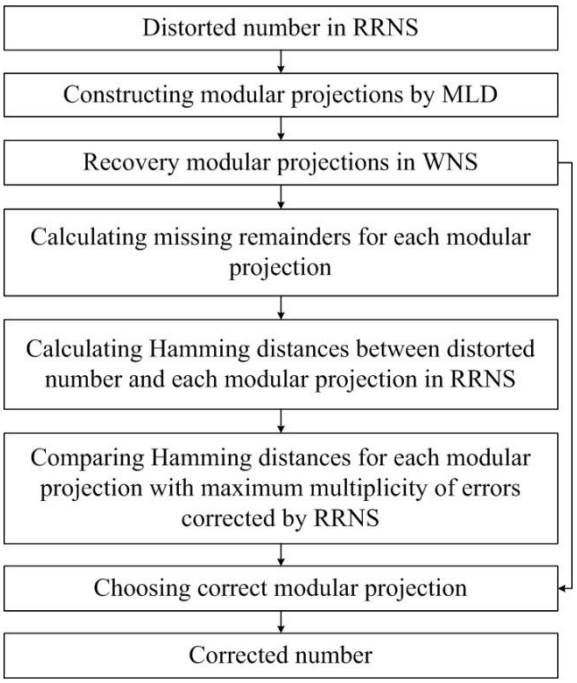
The remainders of the moduli deleted when constructing each projection can be calculated by extending the RNS moduli system [32] or restoring the number in the WNS and calculating the remainders of these moduli. These operations are resource-consuming and significantly increase the total time for correcting errors and restoring the number in the WNS.

4. Modified Modular Projection Method with MLD

Now consider a new method combining the advantages of the CRT, CRTf, and maximum likelihood projections [19]. The proposed modification concerns error localization and restoration of a correct number in the WNS; error detection is identical for the original method [19] and the modified method proposed below. In this regard, the description of error detection is omitted.

According to [19], each projection is restored in the WNS using the CRT. The resulting projections are employed to calculate the values of the missing remainders. Next, the Hamming distances are calculated for each projection, and the correct projection is selected (Figure 1a).

Reconstructing the weighted representation of projections using the CRT has high time delays. First of all, the delay is due to executing operations on a large modulus comparable to the dynamic range of the RRNS.



(**a**)

Figure 1. *Cont.*

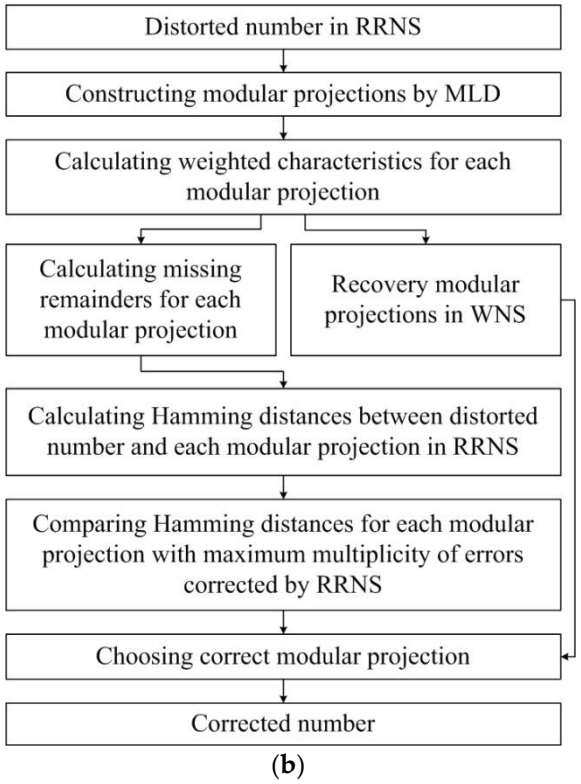


Figure 1. Modular projection methods with MLD: (**a**) original, (**b**) modified.

In the original method [19], the reconstructed weighted representations of modular projections are used, on the one hand, to calculate the missing remainders subsequently and, on the other hand, to obtain the correct number in the WNS. (One of the projections will be the correct number represented in the WNS.) In the modified method, we propose using (a) the ranks of the modular projections to obtain the missing remainders and (b) the approximate weighted characteristics of the CRTf to restore the modular projections in the WNS. This approach allows for calculating the missing remainders and reconstructing projections in the WNS in parallel (Figure 1b), not sequentially, as was proposed in [19].

The efficiency of the proposed modification largely depends on the efficiency of calculating the rank of a number represented in the RRNS. The choice of the CRTf for reconstructing modular projections in the WNS is not accidental: the approximate weighted characteristic $F(X)$ closely relates to the rank r_X of the number X written in the RRNS.

According to (3), the rank r_X is given by

$$r_X = \frac{\sum_{i=1}^{k} \frac{M_K}{m_i} \left| \frac{m_i}{M_K} \right|_{m_i} x_i - X}{M_K} = \frac{\sum_{i=1}^{k} \frac{M_K}{m_i} \left| \frac{m_i}{M_K} \right|_{m_i} x_i}{M_K} - \frac{X}{M_K} = \sum_{i=1}^{k} \frac{|m_i / M_K|_{m_i} x_i}{m_i} - \frac{X}{M_K}.$$

Due to (4) and (5),

$$r_X = \sum_{i=1}^{k} \frac{|m_i / M_K|_{m_i} x_i}{m_i} - \left| \sum_{i=1}^{k} \frac{|m_i / M_K|_{m_i} x_i}{m_i} \right|_1.$$

Recall that $|\bullet|_1$ denotes taking the fractional part of a number. Hence, the rank of X is given by

$$r_X = \left\lfloor \sum_{i=1}^{k} \frac{|m_i / M_K|_{m_i} x_i}{m_i} \right\rfloor = \left\lfloor \sum_{i=1}^{k} k_i^* x_i \right\rfloor, k_i^* = \frac{|m_i / M_K|_{m_i}}{m_i}. \quad (10)$$

Note that $F(X)$ and r_X are the integer and fractional parts of the same value. We introduce the notation

$$F'(X) = \sum_{i=1}^{k} k_i^* x_i, k_i^* = \frac{|m_i / M_K|_{m_i}}{m_i}. \quad (11)$$

Calculating $F\prime(X)$, we simultaneously find the weighted characteristic $F(X)$ (Equation (5)) and the rank r_X of X (Equation (10)).

Similar to the case of the CRTf (Equations (6) and (7)), we propose using a binary shift to implement the calculation of $F'(X)$ on a hardware base not supporting fractions. The accuracy of the constants k_i^* in (11) is estimated by analogy with [20]. Then

$$F'(X) = \sum_{i=1}^{k} k_i^* x_i, k_i^* = \frac{|m_i / M_K|_{m_i}}{m_i} \cdot 2^N, \quad (12)$$

where $N = \log_2 (M_K \sum_{i=1}^{k} (m_i - 1))$ [20].

The value $F'(X)$ is especially convenient for the hardware implementation of the error correction procedure for modular codes. Really, the N least significant bits of the binary representation of $F\prime(X)$ calculated by Equation (12) will be the shifted weighted characteristic $F(X)$ (Equation (7)):

$$F(X) = F\prime(X)_{[N-1..0]}, \quad (13)$$

where $N = \log_2 (M_K \sum_{i=1}^{k} (m_i - 1))$ [20].

The other (most significant) bits will be equal to the rank r_X of X:

$$r_X = F\prime(X)_{[Length(F'(X))-1..N]}, \quad (14)$$

where $N = \log_2 (M_K \sum_{i=1}^{k} (m_i - 1))$ [20].

After obtaining the shifted approximate weighted characteristic $F(X)$ by Equation (13), the standard CRTf Equation (6) is used for restoration. We discuss in detail the calculation of the missing remainders using the rank r_X.

This operation extends the system of RNS moduli [7] and can be performed efficiently by efficiently calculating the rank of a number.

According to (3),

$$X = \sum_{i=1}^{k} \frac{M_K}{m_i} \left| \frac{m_i}{M_K} \right|_{m_i} x_i - r_X M_K.$$

Introducing the compact notation $B_i = \frac{M_K}{m_i} \left| \frac{m_i}{M_K} \right|_{m_i}$, we write

$$X = \sum_{i=1}^{k} B_i x_i - r_X M_K, \; B_i = \frac{M_K}{m_i} \left| \frac{m_i}{M_K} \right|_{m_i}, \; i = 1, \ldots k. \quad (15)$$

Due to (15), for each modular projection, the missing remainders on dividing the number X by the moduli m_j deleted when constructing the projection are given by

$$|X|_{m_j} = \left| \sum_{i=1}^{k} B_i x_i - r_X M_K \right|_{m_j}.$$

Performing trivial transformations, we obtain

$$|X|_{m_j} = \left| \left| \sum_{i=1}^{k} B_i x_i \right|_{m_j} + |r_X \cdot (-M_K)|_{m_j} \right|_{m_j}$$

and, finally,

$$|X|_{m_j} = \left| \left| \sum_{i=1}^{k} |B_i|_{m_j} x_i \right|_{m_j} + \left| r_X \cdot |-M_K|_{m_j} \right|_{m_j} \right|_{m_j}. \quad (16)$$

Note that for each modular projection, the values $|B_i|_{m_j}$ and $|-M_K|_{m_j}$ differ. However, these values are completely determined by the set of RRNS moduli. Therefore, they are calculated in advance and stored in the memory of the computing device.

Thus, the missing remainders are calculated using multiplication by a constant and addition on a small modulus comparable to the RRNS moduli. Both operations are well implemented in hardware. The key feature of the proposed approach is the ability to calculate the first term $\left| \sum_{i=1}^{k} |B_i|_{m_j} x_i \right|_{m_j}$ in the sum (16) independently and in parallel with the second one $\left| r_X \cdot |-M_K|_{m_j} \right|_{m_j}$. We denote them by $X_{m_j}^{(1)}$ and $X_{m_j}^{(2)}$, respectively:

$$X_{m_j}^{(1)} = \left| \sum_{i=1}^{k} |B_i|_{m_j} x_i \right|_{m_j}. \quad (17)$$

$$X_{m_j}^{(2)} = \left| r_X \cdot |-M_K|_{m_j} \right|_{m_j}. \quad (18)$$

Then Equation (16) becomes

$$|X|_{m_j} = \left| X_{m_j}^{(1)} + X_{m_j}^{(2)} \right|_{m_j}. \quad (19)$$

Since the values $X_{m_j}^{(1)}$ are calculated in parallel to $F(X)$ and $X_{m_j}^{(2)}$, the total execution time of the procedure for correcting errors in modular codes and restoring the correct number in the WNS is reduced.

An example demonstrating an approach to extending the system of RNS moduli using the number rank is presented in Appendix A (See, Example A1).

Another example demonstrating the proposed approach to error correction and restoration of the correct number in the WNS is presented in Appendix B (See, Example A2).

The proposed modified modular projection method with MLD for correcting multiple errors in modular codes allows for achieving a high level of parallelism and significantly reducing the total execution time of the multiple error correction procedure.

The proposed method requires calculation of the rank r_X (Equation (14)) for base extension. CRT-I and CRT-II are not directly applicable to do this, while function $F(X)$, used in the article, gives the advantage to calculate the relative size of both the value X and r_X simultaneously.

5. Hardware Implementation of Modified Modular Projection Method with MLD

The original [19] and modified modular projection methods for correcting multiple errors in modular codes (Section 4) have the identical error detection stage. For both methods equally, the implementation variability of this stage lies in choosing an appropriate algorithm for calculating the weighted characteristic of the corrected number. The hardware implementation and comparative analysis of such algorithms were presented in [31]. Let us compare the efficiency of error localization and correct data restoration in a WNS.

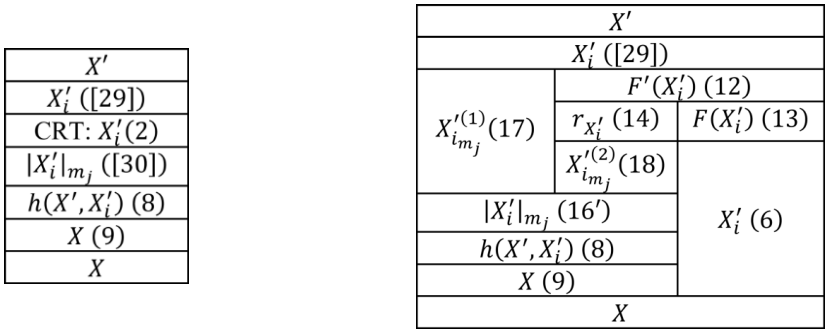
Using various computing devices for implementing methods and algorithms is often dictated by the need to adapt them to the architectural features of a given hardware base. Support of calculations in the ring of integers is a characteristic feature of FPGA computing devices that increases speed but severely restricts the class of executed tasks.

The original method [19] can be implemented on FPGA without any restrictions. The modified method contains operations with fractions and, therefore, requires adaptation when implemented on FPGA. Equations (12)–(14) are used for passing to integer calculations. This transition is illustrated by an example in Appendix C (See, Example A3).

The result of this example coincides with that of an example in Appendix B. Therefore, the transition to integer calculations by Equations (12)–(14) is correct.

The efficiency of the original [19] and modified methods for correcting multiple errors in modular codes was compared by their implementation in the Very High SpeedIntegrated Circuits Hardware Description Language (VHDL) using Xilinx Vivado 2019.1, a software tool for the design and analysis of HDL structures. For testing, we used a Kintex-7 FPGA (core Xilinx xc7k70tfbg676-2) of a sufficient area without DSP blocks.

This software–hardware base has hardware parallelization at the level of blocks implementing individual procedures and functions and, moreover, at the level of operations within these blocks. Figure 2 shows the schemes for parallelizing the compared methods at the block level: vertical lines separate parallel executed blocks, whereas horizontal lines indicate the synchronization of upper blocks. The schemes in Figure 2 specify the corresponding schemes in Figure 1, considering their hardware implementation.



(**a**) (**b**)

Figure 2. Scheme of parallelizing modular projection methods with MLD: (**a**) original, (**b**) modified.

The original method [19] contains no parallel blocks (Figure 2a); however, calculations for each modular projection are performed in parallel within each block, except for the first and the last ones. In the modified method, calculations for each modular projection are also performed in parallel within each block, except for the first and last ones; in addition, there is parallelism at the block level (Figure 2b).

Let us consider the hardware implementation of each block in Figure 2.

Original modular projection method with MLD [19] is presented in Appendix D.

Modified Modular Projection Method with MLD

Block $X\prime$. When an error is detected, this block receives at the input n remainders representing the distorted number $X\prime$ in the (n, k)-RRNS.

Block X'_i. This block constructs modular projections. It is implemented in $C^t_{n/2}$ parallel computational threads, where $t = (n-k)/2$. Each thread corresponds to a modular projection and receives at the input k remainders of the distorted number $X\prime$ (the outputs of the block $X\prime$) obtained by the modular projection algorithm [29].

Block $F\prime(X'_i)$. This block calculates the extended weighted characteristics of modular projections X'_i in the WNS. According to (12), it is implemented by k parallel multiplications of the remainders (the outputs of the block X'_i) by constants modulo M and a k-operand adder [31].

Block $r_{X'_i}$. This block calculates the ranks for each modular projection X'_i. It is implemented by extracting the most significant bits with numbers not less than N (14) from the extended weighted characteristic (the output of the block $F'(X'_i)$).

Block $F(X'_i)$. This block calculates weighted characteristics for each modular projection X'_i. It is implemented by extracting the least significant bits with numbers less than N (13) from the extended weighted characteristic (the output of the block $F'(X'_i)$).

Block $X'^{(1)}_{i_{m_j}}$. This block calculates the missing remainders (the first term) for each modular projection X'_i. It is implemented by k parallel multiplications of the remainders (the outputs of the block X'_i) by constants modulo M and a k-operand modulo adder [31]; see Equation (17).

Block $X'^{(2)}_{i_{m_j}}$. This block calculates the missing remainders (the second term) for each modular projection X'_i. It is implemented by multiplying the rank (the output of the block $r_{X'_i}$) by a constant modulo M [31]; see Equation (18).

Block $\left|X'_i\right|_{m_j}$. This block calculates the missing remainders for each modular projection X'_i. It is implemented by modulo adding the first term (the output of the block $X'^{(1)}_{i_{m_j}}$) to the second term (the output of the block $X'^{(2)}_{i_{m_j}}$) [31]; see Equation (19).

Block X'_i (6). This block calculates the modular projections X'_i in the WNS. It is implemented by multiplying the weighted characteristic of the modular projection (the output of the block $F\left(X'_i\right)$) by the dynamic range M_K. Subsequently, the least significant bits with numbers less than N are discarded; see Equation (6).

Block $h\left(X\prime,\ X'_i\right)$. This block calculates Hamming distances between the distorted number $X\prime$ and each modular projection X'_i. It is implemented by $(n-k)$ parallel comparisons of the corresponding remainders of the distorted number $X\prime$ (the outputs of the block $X\prime$) and the modular projection X'_i (the outputs of the block $\left|X'_i\right|_{m_j}$). Subsequently, the mismatches are counted by an n-operand adder.

Block XX (9). This block chooses a correct modular projection. It is implemented by the conjunction of two comparisons: the modular projection X'_i in the WNS (the output of the block CRT: X'_i) is smaller than the dynamic range M_K, and the Hamming distance (the output of the block $h\left(X\prime,\ X'_i\right)$) is not greater than the maximum multiplicity t of errors corrected by the $(n,\ k)$-RRNS. The results for each thread (conjunctions for different modular projections) are glued together into a $C^t_{n/2}$-bit number, and the number of the first nonzero bit corresponds to the correct modular projection.

Block XX. This block outputs the correct number X in the WNS. It is implemented by a $C^t_{n/2}$-input multiplexer with one control input. The common inputs are the modular projections X'_i in the WNS (the outputs of the block CRT: X'_i). The control input is the number of the correct modular projection (the output of the block X (9)).

In the course of comparative analysis, we obtained data on the time and hardware costs for correcting single and double errors in modular codes. We used the $(2,4)$-RRNS (single error) and $(2,6)$-RRNS (double error) for numbers of different bit widths (4 bits, 8 bits, 16 bits, 24 bits, 32 bits, 48 bits, and 64 bits). The moduli sets, the time and hardware costs for correcting single and double errors and restoring the number in the WNS using the original modular projection method with MLD [19], and the modified one proposed in this paper are presented in Appendix E (See, Tables A7 and A8).

Figure 3 shows the operating times of the methods depending on the bit width of the numbers under correction for a single error (see, Figure 3a) and double errors (see, Figure 3b).

According to Figure 3, the modified modular projection method with MLD proposed in this dynamic is faster than the original method [19]. Based on the simulation results on FPGA (see Appendix D), we conclude that the proposed method speeds up the procedure of error correction and number restoration in the WNS by an average of 1.23 times (18%) compared to the original approach [19]. On average, the modified modular projection method with MLD requires 1.55 times (35%) more resources for single errors and 1.76 times (43%) more resources for double errors.

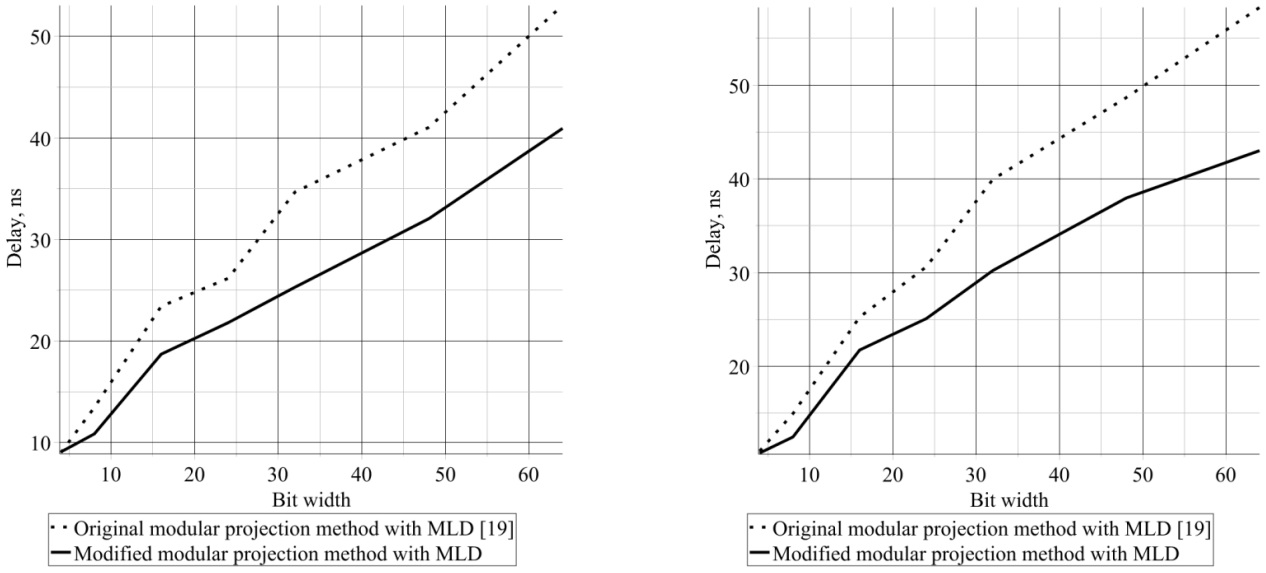


Figure 3. Time to correct errors and restore numbers in WNS: (**a**) single error, (**b**) double error.

6. Conclusions

Recovery weighted values of modular projections has high time complexity due to computations on a large modulus. The paper proposes a method based on the use of approximate positional characteristics and ranks of modular projections. The use of the Chinese Remainder Theorem with fractions in the procedure for correcting errors and recovery numbers in the weighted number system makes it possible to efficiently calculate the rank of each of the modular projections. The ranks of modular projections are employed to calculate the Hamming distances. This approach allows for parallelizing computations at the levels of operations and computing blocks.

The simulation results on FPGA have demonstrated that the proposed modified modular projection method with MLD speeds up the procedure of error correction and number restoration in the weighted number system by an average of 18% compared to the original method [19]. At the same time, it consumes many more hardware resources.

However, in further study, we are going to research the moduli set selection and the possibility of applying the proposed method for error correction codes with the RNS special moduli sets.

Author Contributions: Conceptualization, M.B., A.N., M.D., N.K., A.T., N.V.H., A.A. and V.T.; methodology, M.B., A.N., M.D., N.K., A.T., N.V.H., A.A. and V.T.; validation, M.B., A.N., M.D., N.K., A.T., N.V.H., A.A. and V.T.; formal analysis, M.B., A.N., M.D., N.K., A.T., N.V.H., A.A. and V.T.; investigation, M.B., A.N., M.D., N.K., A.T., N.V.H., A.A. and V.T.; writing—original draft preparation, M.B., A.N., M.D., N.K., A.T., N.V.H., A.A. and V.T.; writing—review and editing, M.B., A.N., M.D., N.K., A.T., N.V.H., A.A. and V.T.; supervision, M.B., A.N., M.D., N.K., A.T., N.V.H., A.A. and V.T. The main part of this work was made when Maxim Deryabin was a Senior Researcher at North-Caucasus Federal University. All authors have read and agreed to the published version of the manuscript.

Funding: This work was carried out at the North Caucasus Center for Mathematical Research within agreement no. 075-02-2021-1749 with the Ministry of Science and Higher Education of the Russian Federation. The reported study was funded by RFBR, project number 20-37-70023; Russian Federation President Grant MK-24.2020.9; and SP-1685.2019.5.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Appendix A

Example A1. *An RNS is given by a set of moduli* $m_1 = 3$, $m_2 = 5$, and $m_3 = 11$. *It is required to find the remainder on dividing the number* $X = (1, 4, 6)$ *in this RNS by the modulus* $m_4 = 7$. *For ease of further consideration, note that* $X = 94$ *in the WNS.*

Appendix A.1. Base Extension. Pre-Computation Stage

We obtain the constants for the moduli set $\{3, 5, 7\}$.
$M_K = m_1 m_2 m_3 = 3 \cdot 5 \cdot 11 = 165$.
We calculate the constants B_i *by Equation* (15):

$$B_1 = \frac{M_K}{m_1}\left|\frac{m_1}{M_K}\right|_{m_1} = \frac{165}{3}\left|\frac{3}{165}\right|_3 = 55 \cdot 1 = 55,$$

$$B_2 = \frac{M_K}{m_2}\left|\frac{m_2}{M_K}\right|_{m_2} = \frac{165}{5}\left|\frac{5}{165}\right|_5 = 33 \cdot 2 = 66,$$

$$B_3 = \frac{M_K}{m_3}\left|\frac{m_3}{M_K}\right|_{m_3} = \frac{165}{11}\left|\frac{11}{165}\right|_{11} = 15 \cdot 3 = 45.$$

Next, we calculate the constants for Equation (16):

$$|B_1|_{m_4} = |55|_7 = 6,\ |B_2|_{m_4} = |66|_7 = 3,\ |B_3|_{m_4} = |45|_7 = 3,\ |-M_K|_{m_4} = |-165|_7 = 3.$$

Finally, we calculate the constants necessary for finding the rank r_X of X. *According to (10),*

$$k_1^* = \frac{|m_1/M_K|_{m_1}}{m_1} = \frac{|3/165|_3}{3} = \frac{1}{3},$$

$$k_2^* = \frac{|m_2/M_K|_{m_2}}{m_2} = \frac{|5/165|_5}{5} = \frac{2}{5},$$

$$k_3^* = \frac{|m_3/M_K|_{m_3}}{m_3} = \frac{|11/165|_{11}}{11} = \frac{3}{11}.$$

Appendix A.2. Base Extension. Computation Stage

Step 1. We obtain $X_{m_4}^{(1)}$ *by Equation (17):*

$$X_{m_4}^{(1)} = \left|\sum_{i=1}^{3} |B_i|_{m_4} x_i\right|_{m_4} = |6 \cdot 1 + 3 \cdot 4 + 3 \cdot 6|_7 = 1.$$

Calculating the first term in Equation (16), we calculate the rank r_X *in parallel. According to (10),*

$$r_X = \sum_{i=1}^{3} k_i^* x_i = \frac{1}{3} \cdot 1 + \frac{2}{5} \cdot 4 + \frac{3}{11} \cdot 6 = 3\frac{94}{165} = 3.$$

Using Equation (18), we find the second term $X_{m_4}^{(2)}$ *in Equation (16):*

$$X_{m_4}^{(2)} = \left|r_X \cdot |-M_K|_{m_4}\right|_{m_4} = |3 \cdot 3|_7 = 2.$$

Step 2. We calculate the desired remainder by Equation (19):

$$|X|_{m_4} = \left|X_{m_4}^{(1)} + X_{m_4}^{(2)}\right|_{m_4} = |1 + 2|_7 = 3.$$

This value matches the remainder on dividing $X = 94$ by the modulus $m_4 = 7$.
End of example.

Appendix B

Example A2. *An (n, k)-RRNS, where $n = 6$ and $k = 2$, is given by a set of moduli $\{5, 7, 8, 9, 11, 13\}$. It is required to find erroneous digits in the number $X\prime = (0, 3, 7, 6, 1, 2)$ and correct them (if any). For ease of further consideration, note that the correct number is $X = 15 = (0, 1, 7, 6, 4, 2)$, (The number $X\prime$ contains a double error in the 2nd and 5th remainders).*

The $(6, 2)$-RRNS can correct $t = (n - k)/2 = r/2 = 2$ errors.

Appendix B.1. Error Detection. Pre-Computation Stage

To detect an error, we have to restore the number $X\prime$ in the WNS and compare it with the dynamic range M_K or calculate the weighted characteristic $F(X\prime)$ and compare it with the corresponding weighted characteristic $F(M_K)$ of the dynamic range. To reduce calculations, we detect errors using weighted characteristics yielded by the CRTf. We calculate all necessary constants that can be pre-computed, saving them in memory (See, Table A1).

We calculate the dynamic and full ranges of this RRNS:

$$M_K = 5{\cdot}7 = 35,\ M_N = 5{\cdot}7{\cdot}8{\cdot}9{\cdot}11{\cdot}13 = 360360.$$

The constants for error detection (Equation (5)) are

$$k_1^* = \frac{|m_1/M_N|_{m_1}}{m_1} = \frac{|5/360360|_5}{5} = \frac{3}{5},$$

$$k_2^* = \frac{|m_2/M_N|_{m_2}}{m_2} = \frac{|7/360360|_7}{7} = \frac{4}{7},$$

$$k_3^* = \frac{|m_3/M_N|_{m_3}}{m_3} = \frac{|8/360360|_8}{8} = \frac{5}{8},$$

$$k_4^* = \frac{|m_4/M_N|_{m_4}}{m_4} = \frac{|9/360360|_9}{9} = \frac{8}{9},$$

$$k_5^* = \frac{|m_5/M_N|_{m_5}}{m_5} = \frac{|11/360360|_{11}}{11} = \frac{6}{11},$$

$$k_6^* = \frac{|m_6/M_N|_{m_6}}{m_6} = \frac{|13/360360|_{13}}{13} = \frac{10}{13}.$$

According to Equation (4),

$$F(M_K) = \frac{M_K}{M_N} = \frac{35}{360360} = \frac{1}{10296}.$$

Table A1. Pre-computed constants for error detection using CRTf.

$\{m_i\}$	$\{k_i^*\}$	$F(M_K)$
{5, 7, 8, 9, 11, 13}	{3/5, 4/7, 5/8, 8/9, 6/11, 10/13}	1/10296

Appendix B.2. Error Detection. Computation Stage

According to Equation (5),

$$F(X\prime) = \left|\sum_{i=1}^{6} k_i^* x_i'\right|_1 = \left|\tfrac{3}{5}\cdot 0 + \tfrac{4}{7}\cdot 3 + \tfrac{5}{8}\cdot 7 + \tfrac{8}{9}\cdot 6 + \tfrac{6}{11}\cdot 1 + \tfrac{10}{13}\cdot 2\right|_1 = \\ = \left|\tfrac{324481}{24024}\right|_1 = \tfrac{12169}{24024} \geq \tfrac{1}{10296} = F(M_K).$$

Since $F(X\prime) \geq F(M_K)$, the number $X\prime$ contains an error.

Appendix B.3. Error Localization and Correction. Pre-Computation Stage

To correct errors, we use the modified modular projection method with MLD. The projections are constructed according to the algorithm [29]:

$$X'_{1,2} = (x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6),$$

$$X'_{3,4} = (x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6),$$

$$X'_{5,6} = (x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6).$$

We calculate constants for each modular projection, saving them in memory (See, Table A2).

Table A2. Pre-computed constants for each modular projection.

		Modular Projection		
		$X'_{1,2}$={0, 3}	$X'_{3,4}$={7, 6}	$X'_{5,6}$={1, 2}
1	Number	$i = 1,\ 2$ $j = 3,\ldots,6$	$i = 3,\ 4$ $j = 1,\ 2,\ 5,\ 6$	$i = 5,\ 6$ $j = 1,\ldots,\ 4$
2	$\{m_i\}$	{5, 7}	{8, 9}	{11, 13}
3	M_K	35	72	143
4	$F(M_K),\ (M_K = 35)$	1	35/72	35/143
5	$\{k_i^*\}$	{3/5, 3/7}	{1/8, 8/9}	{6/11, 6/13}
6	$[\{\lvert B_i\rvert_{m_j}\}]$	[{5, 7}, {3, 6}, {10, 4}, {8, 2}]	[{4, 4}, {2, 1}, {9, 9}, {9, 12}]	[{3, 1}, {1, 3}, {6, 2}, {6, 3}]
7	$\{\lvert -M_K\rvert_{m_j}\}$	{5, 1, 9, 4}	{3, 5, 5, 6}	{2, 4, 1, 1}

The modular projection $X'_{1,2}$: $\{m_i\} = \{m_1,\ m_2\} = \{5,\ 7\}$, $M_K = 5{\cdot}7 = 35$.
The weighted characteristic of the dynamic range M_K for the projection is obtained by Equation (4):

$$F(35) = 35/M_K = 35/35 = 1.$$

The constants for Equation (18) are

$$|-M_K|_{m_3} = |-35|_8 = 5,\ |-M_K|_{m_4} = |-35|_9 = 1,$$

$$|-M_K|_{m_5} = |-35|_{11} = 9,\ |-M_K|_{m_6} = |-35|_{13} = 4.$$

According to Equation (15),

$$B_1 = \frac{M_K}{m_1}\left|\frac{m_1}{M_K}\right|_{m_1} = \frac{35}{5}\left|\frac{5}{35}\right|_5 = 7\cdot 3 = 21,$$

$$B_2 = \frac{M_K}{m_2}\left|\frac{m_2}{M_K}\right|_{m_2} = \frac{35}{7}\left|\frac{7}{35}\right|_7 = 5\cdot 3 = 15.$$

The constants for Equation (17) are

$$|B_1|_{m_3} = |21|_8 = 5,\ |B_1|_{m_4} = |21|_9 = 3,\ |B_1|_{m_5} = |21|_{11} = 10,\ |B_1|_{m_6} = |21|_{13} = 8,$$

$$|B_2|_{m_3} = |15|_8 = 7,\ |B_2|_{m_4} = |15|_9 = 6,\ |B_2|_{m_5} = |15|_{11} = 4,\ |B_2|_{m_6} = |15|_{13} = 2.$$

According to (11),

$$k_1^* = \frac{|m_1/M_K|_{m_1}}{m_1} = \frac{|5/35|_5}{5} = \frac{3}{5},$$

$$k_2^* = \frac{|m_2/M_K|_{m_2}}{m_2} = \frac{|7/35|_7}{7} = \frac{3}{7}.$$

The modular projection $X'_{3,4}: \{m_i\} = \{m_3, m_4\} = \{8, 9\}$, $M_K = 8{\cdot}9 = 72$.
The weighted characteristic of the dynamic range M_K *for the projection is obtained by Equation (4):*

$$F(35) = 35/M_K = 35/72.$$

The constants for Equation (18) are

$$|-M_K|_{m_1} = |-72|_5 = 3,\ |-M_K|_{m_2} = |-72|_7 = 5,$$

$$|-M_K|_{m_5} = |-72|_{11} = 5,\ |-M_K|_{m_6} = |-72|_{13} = 6.$$

According to Equation (15),

$$B_3 = \frac{M_K}{m_3}\left|\frac{m_3}{M_K}\right|_{m_3} = \frac{72}{8}\left|\frac{8}{72}\right|_8 = 9 \cdot 1 = 9,$$

$$B_4 = \frac{M_K}{m_4}\left|\frac{m_4}{M_K}\right|_{m_4} = \frac{72}{9}\left|\frac{9}{72}\right|_9 = 8 \cdot 8 = 64.$$

The constants for Equation (17) are

$$|B_3|_{m_1} = |9|_5 = 4,\ |B_3|_{m_2} = |9|_7 = 2,\ |B_3|_{m_5} = |9|_{11} = 9,\ |B_3|_{m_6} = |9|_{13} = 9,$$

$$|B_4|_{m_1} = |64|_5 = 4,\ |B_4|_{m_2} = |64|_7 = 1,\ |B_4|_{m_5} = |64|_{11} = 9,\ |B_4|_{m_6} = |64|_{13} = 12.$$

According to (11),

$$k_3^* = \frac{|m_3/M_K|_{m_3}}{m_3} = \frac{|8/72|_8}{8} = \frac{1}{8},$$

$$k_4^* = \frac{|m_4/M_K|_{m_4}}{m_4} = \frac{|9/72|_9}{9} = \frac{8}{9}.$$

The modular projection $X'_{5,6}: \{m_i\} = \{m_5, m_6\} = \{11, 13\}$, $M_K = 11{\cdot}13 = 143$.
The weighted characteristic of the dynamic range M_K *for the projection is obtained by Equation (4):*

$$F(35) = 35/M_K = 35/143.$$

The constants for Equation (18) are

$$|-M_K|_{m_1} = |-143|_5 = 2,\ |-M_K|_{m_2} = |-143|_7 = 4,$$

$$|-M_K|_{m_3} = |-143|_8 = 1,\ |-M_K|_{m_4} = |-143|_9 = 1.$$

According to Equation (15),

$$B_5 = \frac{M_K}{m_5}\left|\frac{m_5}{M_K}\right|_{m_5} = \frac{143}{11}\left|\frac{11}{143}\right|_{11} = 13 \cdot 6 = 78,$$

$$B_6 = \frac{M_K}{m_6}\left|\frac{m_6}{M_K}\right|_{m_6} = \frac{143}{13}\left|\frac{13}{143}\right|_{13} = 11 \cdot 6 = 66.$$

The constants for Equation (17) are

$$|B_5|_{m_1} = |78|_5 = 3,\ |B_5|_{m_2} = |78|_7 = 1,\ |B_5|_{m_3} = |78|_8 = 6,\ |B_5|_{m_4} = |78|_9 = 6,$$

$$|B_6|_{m_1} = |66|_5 = 1,\ |B_6|_{m_2} = |66|_7 = 3,\ |B_6|_{m_3} = |66|_8 = 2,\ |B_6|_{m_4} = |66|_9 = 3.$$

According to (11),

$$k_5^* = \frac{|m_5/M_K|_{m_5}}{m_5} = \frac{|11/143|_{11}}{11} = \frac{6}{11},$$

$$k_6^* = \frac{|m_6/M_K|_{m_6}}{m_6} = \frac{|13/143|_{13}}{13} = \frac{6}{13}.$$

Appendix B.4. Error Localization and Correction. Computation Stage

Step 1. We calculate the extended weighted characteristic for each modular projection (Equation (11)) using the pre-computed constants from Table A2. The weighted characteristic and rank of each modular projection are the fractional and integer parts, respectively, of the extended weighted characteristic.

The modular projection $X'_{1,2}$*:*

$$F\prime\left(X'_{1,2}\right) = \frac{3}{5} \cdot 0 + \frac{3}{7} \cdot 3 = 1\frac{2}{7},$$

$$F\left(X'_{1,2}\right) = \left|F\prime\left(X'_{1,2}\right)\right|_1 = \left|1\frac{2}{7}\right|_1 = \frac{2}{7},\ r_{X'_{1,2}} = F\prime\left(X'_{1,2}\right) = 1\frac{2}{7} = 1.$$

The modular projection $X'_{3,4}$*:*

$$F\prime\left(X'_{3,4}\right) = \frac{1}{8} \cdot 7 + \frac{8}{9} \cdot 6 = 6\frac{5}{24},$$

$$F\left(X'_{3,4}\right) = \left|F\prime\left(X'_{3,4}\right)\right|_1 = \left|6\frac{5}{24}\right|_1 = \frac{5}{24},\ r_{X'_{3,4}} = F\prime\left(X'_{3,4}\right) = 6\frac{5}{24} = 6.$$

The modular projection $X'_{5,6}$*:*

$$F\prime\left(X'_{5,6}\right) = \frac{6}{11} \cdot 1 + \frac{6}{13} \cdot 2 = 1\frac{67}{143},$$

$$F\left(X'_{5,6}\right) = \left|F\prime\left(X'_{5,6}\right)\right|_1 = \left|1\frac{67}{143}\right|_1 = \frac{67}{143},\ r_{X'_{5,6}} = F\prime\left(X'_{5,6}\right) = 1\frac{67}{143} = 1.$$

Step 2. We calculate the second terms for finding the missing remainders for each modular projection (Equation (18)) using the constants from Table A2.

The modular projection $X'_{1,2}$*:*

$$X'^{(2)}_{1,2m_3} = |1 \cdot 5|_8 = 5,\ X'^{(2)}_{1,2m_4} = |1 \cdot 1|_9 = 1,$$

$$X'^{(2)}_{1,2m_5} = |1 \cdot 9|_{11} = 9,\ X'^{(2)}_{1,2m_6} = |1 \cdot 4|_{13} = 4.$$

The modular projection $X'_{3,4}$*:*

$$X'^{(2)}_{3,4m_1} = |6 \cdot 3|_5 = 3,\ X'^{(2)}_{3,4m_2} = |6 \cdot 5|_7 = 2,$$

$$X'^{(2)}_{3,4m_5} = |6 \cdot 5|_{11} = 8,\ X'^{(2)}_{3,4m_6} = |6 \cdot 6|_{13} = 10.$$

The modular projection $X'_{5,6}$*:*

$$X'^{(2)}_{5,6m_1} = |1 \cdot 2|_5 = 2,\ X'^{(2)}_{5,6m_2} = |1 \cdot 4|_7 = 4,$$

$$X'^{(2)}_{5,6m_3} = |1 \cdot 1|_8 = 1,\ X'^{(2)}_{5,6m_4} = |1 \cdot 1|_9 = 1.$$

We obtain the weighted representations of modular projections (Equation (4)) in parallel with calculating the second terms.

The modular projection $X'_{1,2}$*:*

$$X'_{1,2} = F\left(X'_{1,2}\right) M_K = \frac{2}{7} \cdot 35 = 10.$$

The modular projection $X'_{3,4}$*:*

$$X'_{3,4} = F\left(X'_{3,4}\right) M_K = \frac{5}{24} \cdot 72 = 15.$$

The modular projection $X'_{5,6}$*:*

$$X'_{5,6} = F\left(X'_{5,6}\right) M_K = \frac{67}{143} \cdot 143 = 67.$$

Step (1–2). We calculate the first terms for finding the missing remainders for each modular projection (Equation (17)) using the constants from Table A2. This is done in parallel with Steps 1–2.

The modular projection $X'_{1,2}$*:*

$$X'^{(1)}_{1,2m_3} = |5 \cdot 0 + 7 \cdot 3|_8 = 5,\ X'^{(1)}_{1,2m_4} = |3 \cdot 0 + 6 \cdot 3|_9 = 0,$$

$$X'^{(1)}_{1,2m_5} = |10 \cdot 0 + 4 \cdot 3|_{11} = 1,\ X'^{(1)}_{1,2m_6} = |8 \cdot 0 + 2 \cdot 3|_{13} = 6.$$

The modular projection $X'_{3,4}$*:*

$$X'^{(1)}_{3,4m_1} = |4 \cdot 7 + 4 \cdot 6|_5 = 2,\ X'^{(1)}_{3,4m_2} = |2 \cdot 7 + 1 \cdot 6|_7 = 6,$$

$$X'^{(1)}_{3,4m_5} = |9 \cdot 7 + 9 \cdot 6|_{11} = 7,\ X'^{(1)}_{3,4m_6} = |9 \cdot 7 + 12 \cdot 6|_{13} = 5.$$

The modular projection $X'_{5,6}$*:*

$$X'^{(1)}_{5,6m_1} = |3 \cdot 1 + 1 \cdot 2|_5 = 0,\ X'^{(1)}_{5,6m_2} = |1 \cdot 1 + 3 \cdot 2|_7 = 0,$$

$$X'^{(1)}_{5,6m_3} = |6 \cdot 1 + 2 \cdot 2|_8 = 2,\ X'^{(1)}_{5,6m_4} = |6 \cdot 1 + 3 \cdot 2|_9 = 3.$$

Step 3. We calculate the missing remainders for each modular projection (Equation (19)).

The modular projection $X'_{1,2}$*:*

$$\left|X'_{1,2}\right|_{m_3} = |5 + 5|_8 = 2,\ \left|X'_{1,2}\right|_{m_4} = |0 + 1|_9 = 1,$$

$$\left|X'_{1,2}\right|_{m_5} = |1 + 9|_{11} = 10,\ \left|X'_{1,2}\right|_{m_6} = |6 + 4|_{13} = 10.$$

The modular projection $X'_{3,4}$*:*

$$\left|X'_{3,4}\right|_{m_1} = |2 + 3|_5 = 0,\ \left|X'_{3,4}\right|_{m_2} = |6 + 2|_7 = 1,$$

$$\left|X'_{3,4}\right|_{m_5} = |7 + 8|_{11} = 4,\ \left|X'_{3,4}\right|_{m_6} = |5 + 10|_{13} = 2.$$

The modular projection $X'_{5,6}$:

$$\left|X'_{5,6}\right|_{m_1} = |0+2|_5 = 2,\ \left|X'_{5,6}\right|_{m_2} = |0+4|_7 = 4,$$

$$\left|X'_{5,6}\right|_{m_3} = |2+1|_8 = 3,\ \left|X'_{5,6}\right|_{m_4} = |3+1|_9 = 4.$$

Step 4. We calculate the Hamming distances between each modular projection in the RRNS and the distorted number (Equation (8)):

$$h\left(X', X'_{1,2}\right) = h\begin{pmatrix} (0,3,7,6,1,2) \\ (0,3,2,1,10,10) \end{pmatrix} = 4,$$

$$h\left(X', X'_{3,4}\right) = h\begin{pmatrix} (0,3,7,6,1,2) \\ (0,1,7,6,4,2) \end{pmatrix} = 2,$$

$$h\left(X', X'_{5,6}\right) = h\begin{pmatrix} (0,3,7,6,1,2) \\ (2,4,3,4,1,2) \end{pmatrix} = 4.$$

Step 5. We choose a correct projection (See, Table A3).

Table A3. Computed modular projections and their characteristics.

		Modular Projection		
		$X'_{1,2}$=10	$X'_{3,4}$=15	$X'_{5,6}$=67
1	$F(X'_i)$	2/7	5/24	67/143
2	$F(M_K),\ (M_K = 35)$	1	35/72	35/143
3	(x'_i)	(0,3,2,1,10,10)	(0,1,7,6,4,2)	(2,4,3,4,1,2)
4	$(x\prime)$	(0,3,7,6,1,2)	(0,3,7,6,1,2)	(0,3,7,6,1,2)
5	$h(X\prime,\ X'_i)$	4	2	4
6	$F(X'_i) < F(M_K),\ (M_K = 35)$	Yes	Yes	No
7	$h(X\prime,\ X'_i) \le t,\ (t = 2)$	No	Yes	No

Both correctness conditions (9) are satisfied for the modular projection $X'_{3,4}$ *only. Therefore, the corrected number is* $X = X'_{3,4} = 15 = (0,\ 1,\ 7,\ 6,\ 4,\ 2)$*, and the proposed error correction method works properly.*

End of example.

Appendix C

Example A3. *An* $(n,\ k)$*-RRNS, where* $n = 6$ *and* $k = 2$*, is given by a set of moduli* $\{5,\ 7,\ 8,\ 9,\ 11,\ 13\}$*. It is required to find erroneous digits in the number* $X\prime = (0,\ 3,\ 7,\ 6,\ 1,\ 2)$ *and correct them (if any). For ease of further considerations, note that the correct number is* $X = 15 = (0,\ 1,\ 7,\ 6,\ 4,\ 2)$*. (The number* $X\prime$*contains a double error in the 2nd and 5th remainders.)*

The$(6,\ 2)$*-RRNS can correct* $t = (n-k)/2 = r/2 = 2$ *errors.*

Appendix C.1. Error Detection. Pre-Computation Stage

To detect an error, we have to restore the number $X\prime$ *in the WNS and compare it with the dynamic range* M_K *, or calculate the weighted characteristic* $F(X\prime)$ *and compare it with the corresponding weighted characteristic* $F(MK)$ *of the dynamic range. To reduce calculations, we detect errors using weighted characteristics yielded by the CRTf. We calculate all necessary constants that can be precomputed, saving them in memory (See, Table A4). In the memory of the computing device, the constants are stored in the binary representation. In this example, we use the decimal representation for clarity.*

We calculate the dynamic and full ranges of this RRNS:

$$M_K = 5{\cdot}7 = 35_{(10)} = 100011_{(2)},$$

$$M_N = 5{\cdot}7{\cdot}8{\cdot}9{\cdot}11{\cdot}13 = 360360_{(10)} = 1010111111110101000_{(2)}.$$

The constants for error detection (Equation (5)) are

$$N = \log_2\left(M_K \sum_{i=1}^{6}(m_i - 1)\right) = \log_2\left(35 \cdot (4+6+7+8+10+12)\right) = 25,$$

$$k_1^* = \frac{|m_1/M_N|_{m_1}}{m_1} \cdot 2^N = \frac{|5/360360|_5}{5} \cdot 2^{25} = \frac{3}{5} \cdot 2^{25} = 20132660_{(10)} = \\ = 1001100110011001100110100_{(2)},$$

$$k_2^* = \frac{|m_2/M_N|_{m_2}}{m_2} \cdot 2^N = \frac{|7/360360|_7}{7} \cdot 2^{25} = \frac{4}{7} \cdot 2^{25} = 19173962_{(10)} = \\ = 1001001001001001001001010_{(2)},$$

$$k_3^* = \frac{|m_3/M_N|_{m_3}}{m_3} \cdot 2^N = \frac{|8/360360|_8}{8} \cdot 2^{25} = \frac{5}{8} \cdot 2^{25} = 20971520_{(10)} = \\ = 1010000000000000000000000_{(2)},$$

$$k_4^* = \frac{|m_4/M_N|_{m_4}}{m_4} \cdot 2^N = \frac{|9/360360|_9}{9} \cdot 2^{25} = \frac{8}{9} \cdot 2^{25} = 29826162_{(10)} = \\ = 1110001110001110001110010_{(2)},$$

$$k_5^* = \frac{|m_5/M_N|_{m_5}}{m_5} \cdot 2^N = \frac{|11/360360|_{11}}{11} \cdot 2^{25} = \frac{6}{11} \cdot 2^{25} = 18302418_{(10)} = \\ = 1000101110100010111010010_{(2)},$$

$$k_6^* = \frac{|m_6/M_N|_{m_6}}{m_6} \cdot 2^N = \frac{|13/360360|_{13}}{13} \cdot 2^{25} = \frac{10}{13} \cdot 2^{25} = 25811102_{(10)} = \\ = 1100010011101100010011110_{(2)}.$$

The weighted characteristic of the dynamic range M_K is obtained by Equation (6):

$$F(M_K) = \frac{M_K}{M_N} \cdot 2^N = \frac{35}{360360} \cdot 2^{25} = 3259_{(10)} = 110010111011_{(2)}.$$

Table A4. Pre-computed constants for error detection using CRTf.

$\{m_i\}$	$\{k_i^*\}$	$F(M_K)$
{5, 7, 8, 9, 11, 13}	{20132660, 19173962, 20971520, 29826162, 18302418, 25811102}	3259

Appendix C.2. Error Detection. Computation Stage

According to Equation (7),

$$F(X\prime) = \left|\sum_{i=1}^{6} k_i^* x_i'\right|_{2^{25}} = |20132660 \cdot 0 + 19173962 \cdot 3 + 20971520 \cdot 7 + 29826162 \cdot 6 + \\ + 18302418 \cdot 1 + 25811102 \cdot 2|_{33554432} = 16996504_{(10)} = \\ = 1000000110101100010011000_{(2)}.$$

$$F(X\prime) = 16996504 \geq 3259 = F(M_K).$$

Since $F(X\prime) \geq F(M_K)$, the number $X\prime$ contains an error.

Appendix C.3. Error Localization and Correction. Pre-Computation Stage

To correct errors, we use the modified modular projection method with MLD. The projections are constructed according to the algorithm [29]:

$$X'_{1,2} = (x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6),$$

$$X'_{3,4} = (x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6),$$

$$X'_{5,6} = (x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6).$$

We calculate constants for each modular projection, saving them in memory (Table A5). In the memory of the computing device, the constants are stored in the binary representation. In this example, we use the decimal representation for clarity.

The modular projection $X'_{1,2}:\ \{m_i\} = \{m_1,\ m_2\} = \{5,\ 7\}$,

$$M_K = 5{\cdot}7 = 35_{(10)} = 100011_{(2)},$$

$$N = \lceil \log_2 \left(\mathrm{M_K} \sum_{\mathrm{i}=1}^{2} (\mathrm{m_i} - 1) \right) \rceil = \lceil \log_2 (35 \cdot (4+6)) \rceil = 9.$$

The weighted characteristic of the dynamic range M_K *for the projection is obtained by Equation (6):*

$$F(35) = \lceil \frac{35}{M_K} \cdot 2^N \rceil = \lceil \frac{35}{35} \cdot 2^9 \rceil = 512_{(10)} = 1000000000_{(2)}.$$

The constants for Equation (18) are

$$|-M_K|_{m_3} = |-35|_8 = 5_{(10)} = 101_{(2)},\ |-M_K|_{m_4} = |-35|_9 = 1_{(10)} = 1_{(2)},$$

$$|-M_K|_{m_5} = |-35|_{11} = 9_{(10)} = 1001_{(2)},\ |-M_K|_{m_6} = |-35|_{13} = 4_{(10)} = 100_{(2)}.$$

According to Equation (15),

$$B_1 = \frac{M_K}{m_1} \left| \frac{m_1}{M_K} \right|_{m_1} = \frac{35}{5} \left| \frac{5}{35} \right|_5 = 7 \cdot 3 = 21,$$

$$B_2 = \frac{M_K}{m_2} \left| \frac{m_2}{M_K} \right|_{m_2} = \frac{35}{7} \left| \frac{7}{35} \right|_7 = 5 \cdot 3 = 15.$$

The constants for Equation (17) are

$$|B_1|_{m_3} = |21|_8 = 5_{(10)} = 101_{(2)},\ |B_2|_{m_3} = |15|_8 = 7_{(10)} = 111_{(2)},$$

$$|B_1|_{m_4} = |21|_9 = 3_{(10)} = 11_{(2)},\ |B_2|_{m_4} = |15|_9 = 6_{(10)} = 110_{(2)},$$

$$|B_1|_{m_5} = |21|_{11} = 10_{(10)} = 1010_{(2)},\ |B_2|_{m_5} = |15|_{11} = 4_{(10)} = 100_{(2)},$$

$$|B_1|_{m_6} = |21|_{13} = 8_{(10)} = 1000_{(2)},\ |B_2|_{m_6} = |15|_{13} = 2_{(10)} = 10_{(2)}.$$

According to (12),

$$k_1^* = \lceil \frac{|m_1 / M_K|_{m_1}}{m_1} \cdot 2^N \rceil = \lceil \frac{|5/35|_5}{5} \cdot 2^9 \rceil = \lceil \frac{3}{5} \cdot 2^9 \rceil = 308_{(10)} = 100110100_{(2)},$$

$$k_2^* = \lceil \frac{|m_2 / M_K|_{m_2}}{m_2} \cdot 2^N \rceil = \lceil \frac{|7/35|_7}{7} \cdot 2^9 \rceil = \lceil \frac{3}{7} \cdot 2^9 \rceil = 220_{(10)} = 11011100_{(2)}.$$

The modular projection $X'_{3,4}:\ \{m_i\} = \{m_3,\ m_4\} = \{8,\ 9\}$,

$$M_K = 8{\cdot}9 = 72,$$

$$N = \lceil \log_2 \left(M_K \sum_{i=1}^{2} (m_i - 1) \right) \rceil = \lceil \log_2 (35 \cdot (7+8)) \rceil = 11.$$

The weighted characteristic of the dynamic range M_K for the projection is obtained by Equation (6):

$$F(35) = \lceil \frac{35}{M_K} \cdot 2^N \rceil = \lceil \frac{35}{72} \cdot 2^{11} \rceil = 996_{(10)} = 1111100100_{(2)}.$$

The constants for Equation (18) are

$$|-M_K|_{m_1} = |-72|_5 = 3_{(10)} = 11_{(2)},\ |-M_K|_{m_2} = |-72|_7 = 5_{(10)} = 101_{(2)},$$

$$|-M_K|_{m_5} = |-72|_{11} = 5_{(10)} = 101_{(2)},\ |-M_K|_{m_6} = |-72|_{13} = 6_{(10)} = 110_{(2)}.$$

According to Equation (15),

$$B_3 = \frac{M_K}{m_3} \left| \frac{m_3}{M_K} \right|_{m_3} = \frac{72}{8} \left| \frac{8}{72} \right|_8 = 9 \cdot 1 = 9,$$

$$B_4 = \frac{M_K}{m_4} \left| \frac{m_4}{M_K} \right|_{m_4} = \frac{72}{9} \left| \frac{9}{72} \right|_9 = 8 \cdot 8 = 64.$$

The constants for Equation (17) are

$$|B_3|_{m_1} = |9|_5 = 4_{(10)} = 100_{(2)},\ |B_4|_{m_1} = |64|_5 = 4_{(10)} = 100_{(2)},$$

$$|B_3|_{m_2} = |9|_7 = 2_{(10)} = 10_{(2)},\ |B_4|_{m_2} = |64|_7 = 1_{(10)} = 1_{(2)},$$

$$|B_3|_{m_5} = |9|_{11} = 9_{(10)} = 1001_{(2)},\ |B_4|_{m_5} = |64|_{11} = 9_{(10)} = 1001_{(2)},$$

$$|B_3|_{m_6} = |9|_{13} = 9_{(10)} = 1001_{(2)},\ |B_4|_{m_6} = |64|_{13} = 12_{(10)} = 1100_{(2)}.$$

According to (12),

$$k_3^* = \lceil \frac{|m_3/M_K|_{m_3}}{m_3} \cdot 2^N \rceil = \lceil \frac{|8/72|_8}{8} \cdot 2^{11} \rceil = \lceil \frac{1}{8} \cdot 2^{11} \rceil = 256_{(10)} = 100000000_{(2)},$$

$$k_4^* = \lceil \frac{|m_4/M_K|_{m_4}}{m_4} \cdot 2^N \rceil = \lceil \frac{|9/72|_9}{9} \cdot 2^{11} \rceil = \lceil \frac{8}{9} \cdot 2^{11} \rceil = 1821_{(10)} = 11100011101_{(2)}.$$

The modular projection $X'_{5,6}$: $\{m_i\} = \{m_5, m_6\} = \{11, 13\}$,

$$M_K = 11 \cdot 13 = 143,$$

$$N = \lceil \log_2 (M_K \sum_{i=1}^{2} (m_i - 1)) \rceil = \lceil \log_2 (35 \cdot (10 + 12)) \rceil = 12.$$

The weighted characteristic of the dynamic range M_K for the projection is obtained by Equation (6):

$$F(35) = \lceil \frac{35}{M_K} \cdot 2^N \rceil = \lceil \frac{35}{143} \cdot 2^{12} \rceil = 1003_{(10)} = 1111101011_{(2)}.$$

The constants for Equation (18) are

$$|-M_K|_{m_1} = |-143|_5 = 2_{(10)} = 10_{(2)},\ |-M_K|_{m_2} = |-143|_7 = 4_{(10)} = 100_{(2)},$$

$$|-M_K|_{m_3} = |-143|_8 = 1_{(10)} = 1_{(2)},\ |-M_K|_{m_4} = |-143|_9 = 1_{(10)} = 1_{(2)}.$$

According to Equation (15),

$$B_5 = \frac{M_K}{m_5} \left| \frac{m_5}{M_K} \right|_{m_5} = \frac{143}{11} \left| \frac{11}{143} \right|_{11} = 13 \cdot 6 = 78,$$

$$B_6 = \frac{M_K}{m_6}\left|\frac{m_6}{M_K}\right|_{m_6} = \frac{143}{13}\left|\frac{13}{143}\right|_{13} = 11 \cdot 6 = 66.$$

The constants for Equation (17) are

$$|B_5|_{m_1} = |78|_5 = 3_{(10)} = 11_{(2)},\ |B_6|_{m_1} = |66|_5 = 1_{(10)} = 1_{(2)},$$
$$|B_5|_{m_2} = |78|_7 = 1_{(10)} = 1_{(2)},\ |B_6|_{m_2} = |66|_7 = 3_{(10)} = 11_{(2)},$$
$$|B_5|_{m_3} = |78|_8 = 6_{(10)} = 110_{(2)},\ |B_6|_{m_3} = |66|_8 = 2_{(10)} = 10_{(2)},$$
$$|B_5|_{m_4} = |78|_9 = 6_{(10)} = 110_{(2)},\ |B_6|_{m_4} = |66|_9 = 3_{(10)} = 11_{(2)}.$$

According to (12),

$$k_5^* = \lceil \frac{|m_5/M_K|_{m_5}}{m_5} \cdot 2^N \rceil = \lceil \frac{|11/143|_{11}}{11} \cdot 2^{12} \rceil = \lceil \frac{6}{11} \cdot 2^{12} \rceil = 2235_{(10)} = 100010111011_{(2)},$$

$$k_6^* = \lceil \frac{|m_6/M_K|_{m_6}}{m_6} \cdot 2^N \rceil = \lceil \frac{|13/143|_{13}}{13} \cdot 2^{12} \rceil = \lceil \frac{6}{13} \cdot 2^{11} \rceil = 1891_{(10)} = 11101100011_{(2)}$$

Appendix C.4. Error Localization and Correction. Computation Stage

Step 1. We calculate the extended weighted characteristic for each modular projection (Equation (12)) using the pre-computed constants from Table A5. The positional characteristic and rank of each modular projection are given by Equations (13) and (14), respectively.

Table A5. Pre-computed constants for each modular projection.

		Modular Projection		
		$X'_{1,2}$={0, 3}	$X'_{3,4}$={7, 6}	$X'_{5,6}$={1, 2}
1	Number	$i = 1, \ldots, 2$ $j = 3, \ldots, 6$	$i = 3, 4$ $j = 1, 2, 5, 6$	$i = 5, 6,$ $j = 1, \ldots, 4$
2	$\{m_i\}$	{5, 7}	{8, 9}	{11, 13}
3	N	9	11	12
4	M_K	35	72	143
5	$F(M_K)$, $(M_K = 35)$	512	996	1003
6	$\{k_i^*\}$	{308, 220}	{256, 1821}	{2235, 1891}
7	$[\{\lvert B_i\rvert_{m_j}\}]$	[{5, 7}, {3, 6}, {10, 4}, {8, 2}]	[{4, 4}, {2, 1}, {9, 9}, {9, 12}]	[{3, 1}, {1, 3}, {6, 2}, {6, 3}]
8	$\{\lvert -M_K\rvert_{m_j}\}$	{5, 1, 9, 4}	{3, 5, 5, 6}	{2, 4, 1, 1}

The modular projection $X'_{1,2}$*:*

$$F'\left(X'_{1,2}\right) = 308 \cdot 0 + 220 \cdot 3 = 660_{(10)} = 1010010100_{(2)},$$

$$F\left(X'_{1,2}\right) = F'(X'_{1,2})_{[N-1..0]} = 1010010100_{[8..0]} = 010010100_{(2)} = 148_{(10)},$$
$$r_{X'_{1,2}} = F'(X'_{1,2})_{[Length(F'(X'_{1,2}))-1..N]} = 1010010100_{[9..9]} = 1_{(2)} = 1_{(10)}.$$

The modular projection $X'_{3,4}$*:*

$$F'(X'_{3,4}) = 256 \cdot 7 + 1821 \cdot 6 = 12718_{(10)} = 11000110101110_{(2)},$$

$$F(X'_{3,4}) = F'(X'_{3,4})_{[N-1..0]} = 11000110101110_{[10..0]} = 00110101110_{(2)} = 430_{(10)},$$

$$r_{X'_{3,4}} = F'(X'_{3,4})_{[Length(F'(X'_{3,4}))-1..N]} = 11000110101110_{[13..11]} = 110_{(2)} = 6_{(10)}$$

The modular projection $X'_{5,6}$*:*

$$F'(X'_{5,6}) = 2235 \cdot 1 + 1891 \cdot 2 = 6017_{(10)} = 1011110000001_{(2)},$$

$$F(X'_{5,6}) = F'(X'_{5,6})_{[N-1..0]} = 1011110000001_{[11..0]} = 011110000001_{(2)} = 1921_{(10)},$$

$$r_{X'_{5,6}} = F'(X'_{5,6})_{[Length(F'(X'_{5,6}))-1..N]} = 1011110000001_{[12..12]} = 1_{(2)} = 1_{(10)}.$$

Step 2. We calculate the second terms for finding the missing remainders for each modular projection (Equation (18)) using the constants from Table A5.

The modular projection $X'_{1,2}$*:*

$$X'^{(2)}_{1,2m_3} = |1 \cdot 5|_8 = 5_{(10)} = 101_{(2)},\ X'^{(2)}_{1,2m_4} = |1 \cdot 1|_9 = 1_{(10)} = 1_{(2)},$$

$$X'^{(2)}_{1,2m_5} = |1 \cdot 9|_{11} = 9_{(10)} = 1001_{(2)},\ X'^{(2)}_{1,2m_6} = |1 \cdot 4|_{13} = 4_{(10)} = 100_{(2)}.$$

The modular projection $X'_{3,4}$*:*

$$X'^{(2)}_{3,4m_1} = |6 \cdot 3|_5 = 3_{(10)} = 11_{(2)},\ X'^{(2)}_{3,4m_2} = |6 \cdot 5|_7 = 2_{(10)} = 10_{(2)},$$

$$X'^{(2)}_{3,4m_5} = |6 \cdot 5|_{11} = 8_{(10)} = 1000_{(2)},\ X'^{(2)}_{3,4m_6} = |6 \cdot 6|_{13} = 10_{(10)} = 1010_{(2)}.$$

The modular projection $X'_{5,6}$*:*

$$X'^{(2)}_{5,6m_1} = |1 \cdot 2|_5 = 2_{(10)} = 10_{(2)},\ X'^{(2)}_{5,6m_2} = |1 \cdot 4|_7 = 4_{(10)} = 100_{(2)},$$

$$X'^{(2)}_{5,6m_3} = |1 \cdot 1|_8 = 1_{(10)} = 1_{(2)},\ X'^{(2)}_{5,6m_4} = |1 \cdot 1|_9 = 1_{(10)} = 1_{(2)}.$$

We obtain the weighted representations of modular projections (Equation (6)) in parallel with calculating the second terms. Note that in Equation (6), division by 2^N*, followed by rounding down, is implemented by simply discarding the last N digits in the binary representation.*

The modular projection $X'_{1,2}$*:*

$$X'_{1,2} = \lceil \frac{F\left(X'_{1,2}\right) M_K}{2^N} \rceil = \lceil \frac{148 \cdot 35}{2^9} \rceil = 10.$$

The modular projection $X'_{3,4}$*:*

$$X'_{3,4} = \lceil \frac{F\left(X'_{3,4}\right) M_K}{2^N} \rceil = \lceil \frac{430 \cdot 72}{2^{11}} \rceil = 15.$$

The modular projection $X'_{5,6}$*:*

$$X'_{5,6} = \lceil \frac{F\left(X'_{5,6}\right) M_K}{2^N} \rceil = \lceil \frac{1921 \cdot 143}{2^{12}} \rceil = 67.$$

Steps (1–2). We calculate the first terms for finding the missing remainders for each modular projection (Equation (17)) using the constants from Table A5. This is done in parallel with Steps 1–2.

The modular projection $X'_{1,2}$*:*

$$X'^{(1)}_{1,2m_3} = |5 \cdot 0 + 7 \cdot 3|_8 = 5_{(10)} = 101_{(2)},\ X'^{(1)}_{1,2m_4} = |3 \cdot 0 + 6 \cdot 3|_9 = 0_{(10)} = 0_{(2)},$$

$$X'^{(1)}_{1,2m_5} = |10 \cdot 0 + 4 \cdot 3|_{11} = 1_{(10)} = 1_{(2)},\ X'^{(1)}_{1,2m_6} = |8 \cdot 0 + 2 \cdot 3|_{13} = 6_{(10)} = 110_{(2)}.$$

The modular projection $X'_{3,4}$*:*

$$X'^{(1)}_{3,4m_1} = |4 \cdot 7 + 4 \cdot 6|_5 = 2_{(10)} = 10_{(2)},\ X'^{(1)}_{3,4m_2} = |2 \cdot 7 + 1 \cdot 6|_7 = 6_{(10)} = 110_{(2)},$$

$$X'^{(1)}_{3,4m_5} = |9 \cdot 7 + 9 \cdot 6|_{11} = 7_{(10)} = 111_{(2)},\ X'^{(1)}_{3,4m_6} = |9 \cdot 7 + 12 \cdot 6|_{13} = 5_{(10)} = 101_{(2)}.$$

The modular projection $X'_{5,6}$*:*

$$X'^{(1)}_{5,6m_1} = |3 \cdot 1 + 1 \cdot 2|_5 = 0_{(10)} = 0_{(2)},\ X'^{(1)}_{5,6m_2} = |1 \cdot 1 + 3 \cdot 2|_7 = 0_{(10)} = 0_{(2)},$$

$$X'^{(1)}_{5,6m_3} = |6 \cdot 1 + 2 \cdot 2|_8 = 2_{(10)} = 10_{(2)},\ X'^{(1)}_{5,6m_4} = |6 \cdot 1 + 3 \cdot 2|_9 = 3_{(10)} = 11_{(2)}.$$

Step 3. We calculate the missing remainders for each modular projection (Equation (19)).
The modular projection $X'_{1,2}$*:*

$$\left|X'_{1,2}\right|_{m_3} = |5 + 5|_8 = 2_{(10)} = 10_{(2)},\ \left|X'_{1,2}\right|_{m_4} = |0 + 1|_9 = 1_{(10)} = 1_{(2)},$$

$$\left|X'_{1,2}\right|_{m_5} = |1 + 9|_{11} = 10_{(10)} = 1010_{(2)},\ \left|X'_{1,2}\right|_{m_6} = |6 + 4|_{13} = 10_{(10)} = 1010_{(2)}.$$

The modular projection $X'_{3,4}$*:*

$$\left|X'_{3,4}\right|_{m_1} = |2 + 3|_5 = 0_{(10)} = 0_{(2)},\ \left|X'_{3,4}\right|_{m_2} = |6 + 2|_7 = 1_{(10)} = 1_{(2)},$$

$$\left|X'_{3,4}\right|_{m_5} = |7 + 8|_{11} = 4_{(10)} = 100_{(2)},\ \left|X'_{3,4}\right|_{m_6} = |5 + 10|_{13} = 2_{(10)} = 10_{(2)}.$$

The modular projection $X'_{5,6}$*:*

$$\left|X'_{5,6}\right|_{m_1} = |0 + 2|_5 = 2_{(10)} = 10_{(2)},\ \left|X'_{5,6}\right|_{m_2} = |0 + 4|_7 = 4_{(10)} = 100_{(2)},$$

$$\left|X'_{5,6}\right|_{m_3} = |2 + 1|_8 = 3_{(10)} = 11_{(2)},\ \left|X'_{5,6}\right|_{m_4} = |3 + 1|_9 = 4_{(10)} = 100_{(2)}.$$

Step 4. We calculate the Hamming distances between each modular projection in the RRNS and the distorted number (Equation (8)):

$$h(X', X'_{1,2}) = h\begin{pmatrix} (0,3,7,6,1,2) \\ (0,3,2,1,10,10) \end{pmatrix} = 4,$$

$$h(X', X'_{3,4}) = h\begin{pmatrix} (0,3,7,6,1,2) \\ (0,1,7,6,4,2) \end{pmatrix} = 2,$$

$$h(X', X'_{5,6}) = h\begin{pmatrix} (0,3,7,6,1,2) \\ (2,4,3,4,1,2) \end{pmatrix} = 4.$$

Step 5. We choose a correct projection (See, Table A6).

Table A6. Computed modular projections and their characteristics.

		Modular Projection		
		$X'_{1,2}$=10	$X'_{3,4}$=15	$X'_{5,6}$=67
1	$F(X'_i)$	148	430	1921
2	$F(M_K)$, $(M_K = 35)$	512	996	1003
3	(x'_i)	(0,3,2,1,10,10)	(0,1,7,6,4,2)	(2,4,3,4,1,2)
4	$(x\prime)$	(0,3,7,6,1,2)	(0,3,7,6,1,2)	(0,3,7,6,1,2)
5	$h(X\prime, X'_i)$	4	2	4
6	$F(X'_i) < F(M_K)$, $(M_K = 35)$	Yes	Yes	No
7	$h(X\prime, X'_i) \leq t$, $(t = 2)$	No	Yes	No

Both correctness conditions (9) are satisfied for the modular projection $X'_{3,4}$ only. Therefore, the corrected number is $X = X'_{3,4} = 15 = (0, 1, 7, 6, 4, 2)$, and the proposed error correction method works properly.

End of example.

Appendix D

Original Modular Projection Method with MLD

Block $X\prime X'$. When an error is detected, this block receives at the input n remainders representing the distorted number $X\prime$ in the (n, k)-RRNS.

Block $X'_i X'$. This block constructs modular projections. It is implemented in $C^t_{n/2}$ parallel computational threads, where $t = (n-k)/2$. Each thread corresponds to a modular projection and receives at the input k remainders of the distorted number $X\prime$ (the outputs of the block $X\prime$) obtained by the modular projection algorithm [29].

Block CRT: X'_i. This block calculates the modular projections X'_i in the WNS. According to the CRT, it is implemented by k parallel multiplications of the remainders (the outputs of the block X'_i) by constants modulo M and a k-operand modulo adder [31].

Block $\left|X'_i\right|_{m_j}$. This block calculates the missing remainders for each modular projection X'_i. It is implemented by $(n-k)$ parallel computations of the remainders of the division of the number X'_i (the output of the block CTR: X'_i) by the missing moduli [30].

Block $h(X\prime, X'_i)$. This block calculates Hamming distances between the distorted number $X\prime$ and each modular projection X'_i. It is implemented by $(n-k)$ parallel comparisons of the corresponding remainders of the distorted number $X\prime$ (the outputs of the block $X\prime$) and the modular projection X'_i (the outputs of the block $\left|X'_i\right|_{m_j}$). Subsequently, the mismatches are counted by an n-operand adder.

Block X (9). This block chooses a correct modular projection. It is implemented by the conjunction of two comparisons: the modular projection X'_i in the WNS (the output of the block CRT: X'_i) is smaller than the dynamic range M_K, and the Hamming distance (the output of the block $h(X\prime, X'_i)$) is not greater than the maximum multiplicity t of errors corrected by the (n, k)-RRNS. The results for each thread (conjunctions for different modular projections) are glued together into a $C^t_{n/2}$-bit number, and the number of the first nonzero bit corresponds to the correct modular projection.

Block X. This block outputs the correct number X in the WNS. It is implemented by a $C^t_{n/2}$-input multiplexer with one control input. The common inputs are the modular projections X'_i in the WNS (the outputs of the block CRT: X'_i). The control input is the number of the correct modular projection (the output of the block X (9)).

Appendix E

Table A7. Hardware and time costs for correcting single errors and restoring numbers in WNS.

	Bit Width	(2,4)-RRNS	Original Modular Projection Method with MLD [19]		Modified Modular Projection Method with MLD	
			LUT	Delay, ns	LUT	Delay, ns
1	4	{5, 7, 8, 9}	45	9.012	84	9.079
2	8	{16, 17, 19, 21}	108	13.459	161	10.853
3	16	{256, 257, 259, 261}	323	23.428	480	18.709
4	24	{4096, 4097, 4099, 4101}	601	26.173	996	21.803
5	32	{65536, 65537, 65539, 65541}	998	34.677	1526	25.305
6	48	{16777216, 16777217, 16777219, 16777221}	1932	41.042	2698	32.047
7	64	{4294967296, 4294967297, 4294967299, 4294967301}	3304	53.05	4606	40.942

Table A8. Hardware and time costs for correcting double errors and restoring numbers in WNS.

	Bit Width	(2,6)-RRNS	Original Modular Projection Method with MLD [19]		Modified Modular Projection Method with MLD	
			LUT	Delay, ns	LUT	Delay, ns
1	4	{5, 7, 8, 9, 11, 13}	145	10.961	251	10.751
2	8	{16, 17, 19, 21, 23, 25}	277	14.963	464	12.467
3	16	{256, 257, 259, 261, 263, 265}	811	25.261	1414	21.744
4	24	{4096, 4097, 4099,4101, 4103, 4105}	1612	30.616	3154	25.092
5	32	{65536, 65537, 65539,65541, 65543, 65545}	2633	39.965	4960	30.23
6	48	{16777216,16777217,16777219,16777221, 16777223, 16777225}	5309	48.673	9048	37.962
7	64	{4294967296, 4294967297, 4294967299, 4294967301, 4294967303, 4294967305}	9102	58.294	14709	43.015

References

1. Zaharia, M.; Chowdhury, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Spark: Cluster computing with working sets. *HotCloud* **2010**, *10*, 95.
2. Li, W.; Yang, Y.; Yuan, D. A Novel Cost-Effective Dynamic Data Replication Strategy for Reliability in Cloud Data Centres. In Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, Sydney, NSW, Australia, 12–14 December 2011; pp. 496–502.
3. Huang, C.; Simitci, H.; Xu, Y.; Ogus, A.; Calder, B.; Gopalan, P.; Li, J.; Yekhanin, S. Erasure coding in windows azure storage. In Proceedings of the 2012 USENIX Annual Technical Conference (USENIXATC 12), Boston, MA, USA, 13–15 June 2012; pp. 15–26.
4. Sathiamoorthy, M.; Asteris, M.; Papailiopoulos, D.; Dimakis, A.G.; Vadali, R.; Chen, S.; Borthakur, D. XORing Elephants: Novel Erasure Codes for Big Data. *arXiv* **2013**, arXiv:1301.3791. [CrossRef]
5. Muralidhar, S.; Lloyd, W.; Roy, S.; Hill, C.; Lin, E.; Liu, W.; Pan, S.; Shankar, S.; Sivakumar, V.; Tang, L.; et al. f4: Facebook's Warm BLOB Storage System. In Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), Broomfield, CO, USA, 6–8 October 2014; pp. 383–398.
6. Goh, V.T.; Tinauli, M.; Siddiqi, M.U. A novel error correction scheme based on the Chinese remainder theorem. In Proceedings of the Ninth International Conference on Communications Systems, ICCS 2004, Singapore, 7 September 2004; pp. 461–465.
7. Ding, C.; Pei, D.; Salomaa, A. *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*; World Scientific: Singapore, 1996; 213p.
8. Wicker, S.B.; Bhargava, V.K. *Reed-Solomon Codes and Their Applications*; John Wiley & Sons: New York, NY, USA, 1999; 336p.
9. Rabin, M.O. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM* **1989**, *36*, 335–348. [CrossRef]
10. Kshemkalyani, A.D.; Singhal, M. *Distributed Computing: Principles, Algorithms, and Systems*; Cambridge University Press: Cambridge, UK, 2011; 756p.
11. Tel, G. *Introduction to Distributed Algorithms*; Cambridge University Press: Cambridge, UK, 2000; 596p.
12. Van Steen, M.; Tanenbaum, A. Distributed systems principles and paradigms. *Network* **2002**, *2*, 28.
13. Gentry, C.; Halevi, S.; Smart, N.P. Fully Homomorphic Encryption with Polylog Overhead. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, 15–19 April 2012; pp. 465–482.
14. Sun, J.-D.; Krishna, H. A coding theory approach to error control in redundant residue number systems. II. Multiple error detection and correction. *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.* **1992**, *39*, 18–34. [CrossRef]

15. Tay, T.F.; Chang, C.-H. A new algorithm for single residue digit error correction in Redundant Residue Number System. In Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne, Australia, 1–5 June 2014; pp. 1748–1751.
16. Yau, S.S.-S.; Li, Y.-C. Error Correction in Redundant Residue Number Systems. *IEEE Trans. Comput.* **1973**, *100*, 5–11. [CrossRef]
17. Barsi, F.; Maestrini, P. Error Correcting Properties of Redundant Residue Number Systems. *IEEE Trans. Comput.* **1973**, *100*, 307–315. [CrossRef]
18. Chervyakov, N.I.; Lyakhov, P.A.; Babenko, M.G.; Lavrinenko, I.N.; Lavrinenko, A.V.; Nazarov, A.S. The architecture of a fault-tolerant modular neurocomputer based on modular number projections. *Neurocomputing* **2018**, *272*, 96–107. [CrossRef]
19. Goh, V.T.; Siddiqi, M.U. Multiple error detection and correction based on redundant residue number systems. *IEEE Trans. Commun.* **2008**, *56*, 325–330. [CrossRef]
20. Chervyakov, N.I.; Molahosseini, A.S.; Lyakhov, P.A.; Babenko, M.G.; Deryabin, M.A. Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem. *Int. J. Comput. Math.* **2017**, *94*, 1833–1849. [CrossRef]
21. Soderstrand, M.; Vernia, C.; Chang, J.-H. An improved residue number system digital-to-analog converter. *IEEE Trans. Circuits Syst.* **1983**, *30*, 903–907. [CrossRef]
22. Van Vu, T. Efficient Implementations of the Chinese Remainder Theorem for Sign Detection and Residue Decoding. *IEEE Trans. Comput.* **1985**, *100*, 646–651. [CrossRef]
23. Tang, Y.; Boutillon, E.; Jego, C.; Jezequel, M. A new single-error correction scheme based on self-diagnosis residue number arithmetic. In Proceedings of the 2010 Conference on Design and Architectures for Signal and Image Processing (DASIP), Edinburgh, UK, 26–28 October 2010; pp. 27–33.
24. Tchernykh, A.; Babenko, M.; Chervyakov, N.; Miranda-Lopez, V.; Avetisyan, A.; Drozdov, A.Y.; Rivera-Rodriguez, R.; Radchenko, G.; Du, Z. Scalable Data Storage Design for Nonstationary IoT Environment with Adaptive Security and Reliability. *IEEE Internet Things J.* **2020**, *7*, 10171–10188. [CrossRef]
25. Yin, P.; Li, L. A new algorithm for single error correction in RRNS. In Proceedings of the 2013 International Conference on Communications, Circuits and Systems (ICCCAS), Chengdu, China, 15–17 November 2013; pp. 178–181.
26. Goldreich, O.; Ron, D.; Sudan, M. Chinese remaindering with errors. In Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, Atlanta, Georgia, 1–4 May 1999; pp. 225–234.
27. Mandelbaum, D. On a class of arithmetic codes and a decoding algorithm (Corresp.). *IEEE Trans. Inf. Theory* **1976**, *22*, 85–88. [CrossRef]
28. Jenkins, W.K.; Altman, E.J. Self-checking properties of residue number error checkers based on mixed radix conversion. *IEEE Trans. Circuits Syst.* **1988**, *35*, 159–167. [CrossRef]
29. Nazarov, A.; Babenko, M.; Tchernykh, A.; Pulido-Gaytand, B.; Cortés-Mendozad, J.M.; Vashchenko, I. Algorithm for constructing modular projections for correcting multiple errors based on a redundant residue number system using maximum likelihood decoding. *Program. Comput. Softw.* **2021**, *47*, 839–848. [CrossRef]
30. Nazarov, A.; Babenko, M.; Golimblevskaia, E. Efficient Hardware Implementation of forward Conversion WNS-RNS on FPGA. In Proceedings of the 2020 International Conference Engineering and Telecommunication (En & T), Dolgoprudny, Russia, 25–26 November 2020; pp. 1–4. [CrossRef]
31. Nazarov, A.; Babenko, M.; Golimblevskaia, E. Hardware Implementation of the Reverse Conversion RNS-WNS on FPGA. In Proceedings of the 2020 International Conference Engineering and Telecommunication (En & T), Dolgoprudny, Russia, 25–26 November 2020; pp. 1–5. [CrossRef]
32. Ananda Mohan, P.V. *Residue Number Systems: Theory and Applications*; Birghauser Math: Basel, Switzerland, 2016. [CrossRef]
33. Omondi, A.R.; Premkumar, A.B. *Residue Number Systems: Theory and Implementation*; World Scientific: Singapore, 2007.
34. Menezes, A.J.; Van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 2018.
35. Nazarov, A.; Chervyakov, N.; Tchernykh, A.; Babenko, M. Reliability improvement of information systems by residue number system code. *Int. J. Comb. Optim. Probl. Inform.* **2018**, *9*, 81.