electronics

MDPI

Article

RNS Number Comparator Based on a Modified Diagonal Function

Mikhail Babenko [1,2,*], Maxim Deryabin [1], Stanislaw J. Piestrak [3], Piotr Patronik [4], Nikolay Chervyakov [1], Andrei Tchernykh [2,5] and Arutyun Avetisyan [2]

1 Department of Computational Mathematics and Cybernetics, North-Caucasus Federal University, 355017 Stavropol, Russia; maderiabin@ncfu.ru (M.D.); ncherviakov@ncfu.ru (N.C.)
2 Institute for System Programming of the Russian Academy of Sciences, 109004 Moscow, Russia; chernykh@cicese.edu.mx (A.T.); arut@ispras.ru (A.A.)
3 Université de Lorraine, CNRS, IJL, F-54000 Nancy, France; stanislaw.piestrak@univ-lorraine.fr
4 Department of Computer Engineering, Wroclaw University of Technology, 50370 Wrocław, Poland; piotr.patronik@pwr.wroc.pl
5 Computer Science Department, CICESE Research Center, 22860 Ensenada, Mexico
* Correspondence: mgbabenko@ncfu.ru; Tel.: +7-906-440-0219

Received: 1 October 2020; Accepted: 23 October 2020; Published: 27 October 2020

Abstract: Number comparison has long been recognized as one of the most fundamental non-modular arithmetic operations to be executed in a non-positional Residue Number System (RNS). In this paper, a new technique for designing comparators of RNS numbers represented in an arbitrary moduli set is presented. It is based on a newly introduced modified diagonal function, whose strictly monotonic properties make it possible to replace the cumbersome operations of finding the remainder of the division by a large and awkward number with significantly simpler computations involving only a power of 2 modulus. Comparators of numbers represented in sample RNSs composed of varying numbers of moduli and offering different dynamic ranges, designed using various methods, were synthesized for the 65 nm technology. The experimental results suggest that the new circuits enjoy a delay reduction ranging from over 11% to over 75% compared to the fastest circuits designed using existing methods. Moreover, it is achieved using less hardware, the reduction of which reaches over 41%, and is accompanied by significantly reduced power-consumption, which in several cases exceeds 100%. Therefore, it seems that the presented method leads to the design of the most efficient current hardware comparators of numbers represented using a general RNS moduli set.

Keywords: diagonal function; high-speed arithmetic; magnitude comparison; number comparison; Residue Number System (RNS)

1. Introduction

Parallel data processing is one of the most viable approaches to meet steadily growing needs for high-performance computations. Therefore, algorithms and data representations enjoying parallel structures, which facilitate the processing of a large amount of data efficiently, have been an area of active research for many years. One of the promising directions in this field relies on using the Residue Number System (RNS) to represent integers [1,2]. The RNS is a non-positional number system defined by the set of n ($n \geq 2$) pairwise relatively prime positive integers called moduli $\{m_1, m_2, \cdots, m_n\}$. Its dynamic range is equal to the product $M = \prod_{i=1}^{n} m_i$, which allows it to represent all a-bit numbers, where $a = \lfloor \log_2 M \rfloor$. Any non-negative integer X such that $0 \leq X < M$ can be uniquely represented in RNS as $X = \{x_1, x_2, \cdots, x_n\}$, where the ith digit of X in RNS $x_i = |X|_{m_i}$ is the remainder of the integer division of X by the modulus m_i, represented in $a_i = \lceil \log_2 m_i \rceil$ bits.

Indeed, in recent years, an inherent parallelism between RNS and potentially lower power consumption has motivated researchers to consider its use for implementation in hardware of various classes of computations, like digital filtering [3–6], multicarrier modulation schemes with error correction [7], and some cryptographic algorithms [8]. An excellent summary of other RNS applications for Digital Signal Processing (DSP) systems and analysis of various design issues can be found in [9]. Coprocessors with limited sets of instructions for high-speed and low power consumption executed in RNS have been proposed [10,11]. Besides these well-established applications, many other emerging RNS applications have been surveyed [12].

The primary benefit of RNS is the possibility of parallel execution of basic arithmetic operations (addition, subtraction, and multiplication). Unlike positional number representation, the RNS is carry-free, which can simplify the processing of large numbers, replacing it with the execution of parallel arithmetic modular operations on numbers of significantly smaller size. Unfortunately, several non-modular operations are also indispensable, the execution of which in RNS requires interaction between different moduli due to the positional nature of these operations: residue-to-binary (reverse) conversion, magnitude comparison, sign detection, overflow detection, scaling, and division. Among them, magnitude comparison is one of the most fundamental operations, and it, besides being used directly, is also the cornerstone of division, sign detection, overflow detection, etc. Unfortunately, it is a difficult operation in RNS, because a non-positional RNS number representation does not reveal any information about the magnitude of a number, so that special methods involving handling all residue digits must be used.

Several techniques for number comparison in RNS have been studied [1,13–25]. The simplest approach to comparing numbers in RNS relies on first converting them to the positional notation, just to be compared using a simple number comparator [1]. Such a comparator is based on using any reverse (residue-to-binary) converter, which can be built on the basis of the Chinese Remainder Theorem (CRT), the Mixed Radix Conversion (MRC) method, or some variant or a combination of them [1,2,26–28]. Because the reverse converter is available anyway in any RNS-based processor, the only extra hardware cost is that of the ordinary a-bit comparator of numbers, which can be designed, e.g., according to [29] (pp. 45–47). Obviously, a comparison relying on traditional reverse conversion techniques inherits their major drawbacks: the multi-operand addition modulo is a large number (the dynamic range M) for the CRT or lengthy sequential computations for the MRC, either resulting in excessive delay and unnecessarily high power consumption.

One of the most promising approaches to handle non-positional arithmetic operations in RNS that has been proposed relies on computation of some positional characteristics of RNS numbers, according to which it would be possible to determine the magnitudes of numbers and hence their comparison. This idea relies on a hypothesis that computations involving such a positional characteristic can be implemented more efficiently than reverse conversion, due to using simpler arithmetic operations. One is the *core function* introduced in 1977 by Akushskii [13] and used for comparison. Its faster version, making it possible to avoid lengthy iterative computations, relied on introducing a redundant modulus and was proposed in [14]. The other approach uses the so-called *diagonal function*, which is defined as the sum of the quotients of division of the number by all system moduli, introduced in [16] and further developed in [18] and [21]. One of the methods based on the diagonal function, called the Sum of Quotients Technique (SQT), was claimed to be one of the most efficient hardware approaches [18]. Unfortunately, a more accurate performance estimation of the latter, presented recently in [23], revealed that the direct implementation of the comparator using the diagonal function according to [16,18] leads to inefficient circuitry. (Because the design method proposed here is based on some ideas of the diagonal function, while aiming to avoid its drawbacks, it will be detailed in Section 2.) Some other general approaches to RNS number comparison were presented in [15,17,22,26,27]. In [15], a comparison was proposed based on parity checking, provided that the basic moduli set consists of odd moduli only and that the redundant modulus is added. Those proposed in [26,27] make it possible to compute the positional characteristic using CRT without the expensive operation of finding

the remainder of a division. The comparison algorithm based on the new CRT-II, suggested in [17], makes it possible to reduce the maximum size of the modulo addition from M to approximately $\sqrt{M}$, where M is the dynamic range. The method of [22] relies on the approximate calculation of positional numbers according to CRT, whereas that of [24] makes it possible to compare signed numbers, but it also requires sign detection for each compared number. Finally, some comparators have been proposed for RNSs using special bases, e.g., the 4-moduli set composed of two pairs of conjugate moduli $\left\{ 2^n-1,\ 2^n+1,\ 2^{n+1}-1,\ 2^{n+1}+1 \right\}$, as well as the 3-moduli sets $\{ 2^n-1,\ 2^n,\ 2^n+1 \}$ [20] and $\{2^n-1,\ 2^{n+x},\ 2^n+1\}$ [25].

In summary, the drawbacks of the previous magnitude comparison algorithms are: the need for using a redundant modulus, restricting the moduli set or time-consuming modulo operations involving large numbers (the size of the dynamic range or close). Here we will show how to extend the idea of diagonal function so that a high-speed and efficient comparator in RNS can be implemented in hardware. The new approach proposed here relies on integrating techniques from [16,26], and it is based on modifying the diagonal function of the numbers represented in RNS. The major advantages of this method are that, in addition to not requiring the computation of a remainder of division, it also leads to computations involving numbers of smaller sizes than in [26].

This paper is organized as follows. Section 2 presents the method of comparison using the SQT based on the diagonal function. Section 3 thoroughly details the theoretical background of the modified diagonal function proposed here, leading to significantly improved performance of the comparator. Performance estimations and comparison against existing circuits are provided in Section 4. Finally, some conclusions and suggestions for future research are given in Section 5.

2. Number Comparison Using the Sum of Quotients Technique (SQT)

In this section, we will present all key ideas related to the SQT method of [16], which will facilitate understanding of our method relying on a modification of the SQT method, which will be presented in Section 3. The main idea of the SQT method relies on the observation that in the finite n-dimensional space determined by the number of moduli n, the integers are ordered along straight lines, which are parallel to the main diagonal of the space. In MRC, each line represents the most significant digit of the number. However, these diagonals can be renumbered in a natural order of integers. In this case, the comparison of two numbers can be done by considering the numbers of the diagonals to which they belong. For fast determination of the diagonal to which a number belongs, a monotonically increasing function called the Sum of Quotients (SQ) was defined:

$$SQ = \sum_{i=1}^{n} M_i \tag{1}$$

where $M_i = M/m_i$. Let us define the following constant:

$$k_i = \left| -\frac{1}{m_i} \right|_{SQ} \tag{2}$$

where $h_i = \left| 1/m_i \right|_{SQ}$ is the multiplicative inverse of m_i mod SQ $(1 < h_i < SQ)$, i.e., such an integer that $|h_i \cdot m_i|_{SQ} = 1$. (Recall that a multiplicative inverse exists provided that m_i and SQ are co-prime, which is indeed the case here.) It was shown that for the set of constants k_i the following congruence holds:

$$|k_i + k_2 + \ldots + k_n|_{SQ} = 0. \tag{3}$$

These notions are essential to defining the diagonal function as

$$D(X) = \left| \sum_{i=1}^{n} k_i \cdot x_i \right|_{SQ} \tag{4}$$

which was shown to be monotonically increasing over a set of integers $0 \leq X < M$. This method is called the Sum of Quotients Technique (SQT), because the following important equality holds:

$$D(X) = \sum_{i=1}^{n} \left\lfloor \frac{X}{m_i} \right\rfloor \tag{5}$$

The comparison of RNS numbers using SQT is summarized in the following algorithm, whose hardware implementation is shown in Figure 1.

Algorithm 1: Comparison of RNS numbers using SQT.

Input: $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_n\}$
Output: "100" if $X < Y$, "010" if $X = Y$, and "001" if $X > Y$.
Step 1. Calculate $D(X)$ and $D(Y)$.
Note: These computations are independent and therefore they can be executed in parallel, provided that two circuits implementing the diagonal function are available.
Step 2. Compare the values of $D(X)$ and $D(Y)$:
1. if $D(X) < D(Y)$ then return "100";
2. if $D(X) > D(Y)$ then return "001";
3. if $D(X) = D(Y)$ then:
3.1. if $x_1 < y_1$ then return "100";
3.2. if $x_1 = y_1$ then return "010";
3.3. if $x_1 > y_1$ then return "001".

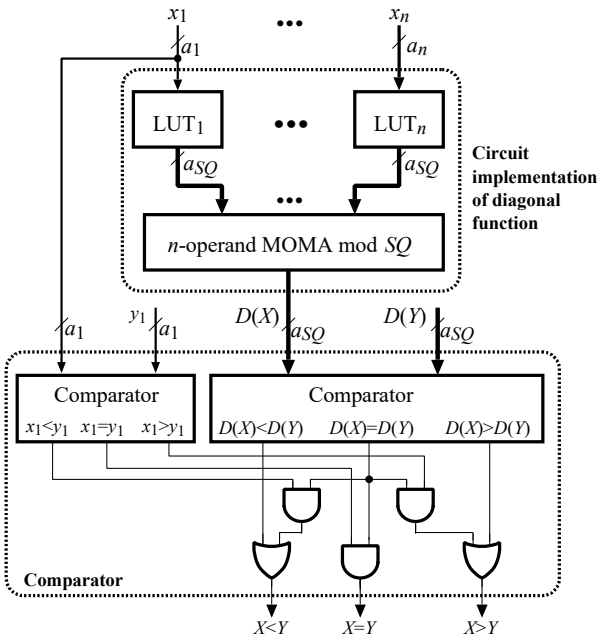

Figure 1. Hardware implementation of the basic number comparison algorithm using the diagonal function [25,28].

The main disadvantage of Algorithm 1 is that the computation of the remainder of the division over the modulus SQ, executed by the n-operand multi-operand modular adder (MOMA) mod SQ, is both hardware and time-consuming. (It will be seen later that for sample moduli sets the difference between the bit sizes of M and SQ could be from 3 to 5-bits.) In [16], it was suggested that in the case of the equality $D(X) = D(Y)$ (the diagonal function is not strictly monotonic), an extra comparison must be executed. However, in [23], it was shown that this additional comparison can actually be done in parallel, so that the only delay penalty is two gate levels (this observation was taken into account in

Figure 1). In the following section, we will show how to modify SQT to replace the MOMA mod SQ with a significantly faster and simpler circuit modulo with a power of 2.

3. Comparison Using the Modified Diagonal Function

Here, we will describe the new method for comparison of RNS numbers based on introducing the modified diagonal function (MDF). It is based on the observation that if all constants k_i are divided by SQ, i.e., similarly as was done for the CRT-based sign detector proposed in [26], then it is possible to move the computations from the residue class $[0,\ SQ-1)$ to the computations in the interval $[0,\ 1)$, so that computations involving integer parts of real numbers are not really needed. In other words, the operation of finding the remainder of the division by SQ is replaced with the more efficient operation of discarding an integer part of a number. However, the major concern with such an approach is its accuracy, because in most cases the fractional numbers cannot be represented exactly using a finite number of bits. Nevertheless, the accurate passing from computations on fraction parts to computations on integers can be done as follows:

1. Multiply each real constant by 2^N, where N is the number of bits of the fraction part, which guarantees sufficient accuracy.
2. For each real number, say Z, calculate $\lceil Z \rceil$, i.e., the smallest integer not less than Z.
3. Execute all computations modulo 2^N (it is sufficient to ignore all carries generated from the $(N-1)$-th position).

Note: The only limitation for the above conversion could occur when SQ divides 2^N (i.e., SQ is a power of two), because in this case, the method suggested reduces to the original one. Nevertheless, because in most cases SQ does not divide 2^N, we will therefore henceforth consider only this case. To determine the smallest N which guarantees sufficient accuracy, we proceed as follows.

First, notice that the constants can be recalculated as

$$\overline{k_i} = \left\lceil \frac{k_i \cdot 2^N}{SQ} \right\rceil = \frac{k_i \cdot 2^N}{SQ} + R_i,\ 1 \le i \le n \tag{6}$$

where $R_i = \left\lceil \frac{k_i 2^N}{SQ} \right\rceil - \frac{k_i 2^N}{SQ}$ and $R_i \in [0,\ 1)$. Because according to Equality (3) the sum $\sum_{i=1}^{n} k_i$ is divisible by SQ, it implies that $R = \sum_{i=1}^{n} R_i$ is an integer. Furthermore, because $0 < R_i < 1$ then $1 \le R < n$.

Now we can define the following positional characteristic of a number, which will be called the modified diagonal function (MDF):

$$\overline{D}(X) = \left| \sum_{i=1}^{n} \overline{k_i} \cdot x_i \right|_{2^N} \tag{7}$$

Theorem 1. *Let $m_n$be the largest modulus of the moduli set. If $N \ge \left\lceil \log_2(SQ \cdot (m_n - 1)) \right\rceil$ then the MDF $\overline{D}(X)$ is strictly increasing for any $0 \le X < M$, i.e., for any $0 \le X_i < X_j \le M-1$ we have $\overline{D}(X_i) < \overline{D}\left(X_j\right)$.*

Proof. First, we find the value of $D(X-1)$ for any $0 < X < M$. Because for any $1 \le\ i \le\ n$

$$\left\lceil \frac{X-1}{m_i} \right\rceil = \begin{cases} \left\lceil \frac{X}{m_i} \right\rceil & if\ x_i \neq 0 \\ \left\lceil \frac{X}{m_i} \right\rceil - 1 & if\ x_i = 0 \end{cases} \tag{8}$$

therefore, according to Equality (5), we have

$$D(X-1) = \sum_{i=1}^{n} \left\lceil \frac{X}{m_i} \right\rceil - \sum_{i=1}^{n} z(x_i) = D(X) - \sum_{i=1}^{n} z(x_i) \tag{9}$$

where

$$z(x_i) = \begin{cases} 0 & if\ x_i \neq 0 \\ 1 & if\ x_i = 0 \end{cases}$$

Consider

$$\widetilde{D}(X) = \frac{D(X)}{SQ} = \left| \sum_{i=1}^{n} \frac{k_i}{SQ} \cdot x_i \right|_1 \tag{10}$$

where $|Z|_1$ denotes discarding of an integer part of Z. Obviously, because $\widetilde{D}(X) \geq \widetilde{D}(X-1)$, $\widetilde{D}(X-1)$ can be determined using Equality (10):

$$\widetilde{D}(X-1) = \frac{D(X)}{SQ} - \frac{1}{SQ} \cdot \sum_{i=1}^{n} z(x_i) = \widetilde{D}(X) - \frac{1}{SQ} \cdot \sum_{i=1}^{n} z(x_i) \tag{11}$$

Now we will determine the properties of the functions $\widetilde{D}(X)$ and $\widetilde{D}(X-1)$. By applying the notation introduced earlier, we obtain

$$\overline{D}(X) = \left| \sum_{i=1}^{n} \left(\frac{k_i \cdot 2^N}{SQ} + R_i \right) \cdot x_i \right|_{2^N} = \left| \sum_{i=1}^{n} \frac{k_i \cdot 2^N}{SQ} x_i + \sum_{i=1}^{n} R_i \cdot x_i \right|_{2^N} = \left| 2^N \cdot \sum_{i=1}^{n} \frac{k_i}{SQ} x_i + 2^N \cdot \left| \sum_{i=1}^{n} \frac{k_i}{SQ} x_i \right|_1 + \sum_{i=1}^{n} R_i \cdot x_i \right|_{2^N} \tag{12}$$

which leads to

$$\overline{D}(X) = \left| 2^N \cdot \widetilde{D}(X) + \sum_{i=1}^{n} R_i \cdot x_i \right|_{2^N} \tag{13}$$

Now according to Equality (11) and by taking into account that in RNS $X - 1 = \left\{ |x_1 - 1|_{m_1},\ |x_2 - 1|_{m2}, \ldots, |x_n - 1|_{m_n} \right\}$, we obtain

$$\overline{D}(X-1) = \left| 2^N \cdot \widetilde{D}(X-1) + \sum_{i=1}^{n} R_i \cdot |x_i - 1|_{m_i} \right|_{2^N} = \left| 2^N \widetilde{D}(X) - \frac{2^N}{SQ} \cdot \sum_{i=1}^{n} z(x_i) + \sum_{i=1}^{n} R_i \cdot |x_i - 1|_{m_i} \right|_{2^N} \tag{14}$$

Because for any $0 \leq i \leq n$

$$|x_i - 1|_{m_i} = \begin{cases} x_i - 1 & if\ x_i \neq 0 \\ m_i - 1 & if\ x_i = 0 \end{cases}$$

in Equality (14) we have

$$\sum_{i=1}^{n} R_i \cdot |x_i - 1|_{m_i} = \sum_{i=1}^{n} R_i \cdot x_i - R + \sum_{i=1}^{n} z(x_i) \cdot R_i \cdot m_i \tag{15}$$

which hence becomes

$$\overline{D}(X-1) = \left| 2^N \cdot \widetilde{D}(X) + \sum_{i=1}^{n} R_i \cdot x_i - \frac{2^N}{SQ} \cdot \sum_{i=1}^{n} z(x_i) - R + \sum_{i=1}^{n} z(x_i) \cdot R_i \cdot m_i \right|_{2^N} \tag{16}$$

From the formulas derived above, it is obvious that $\overline{D}(X) - \overline{D}(X-1)$ is the additional term of the expression equal to

$$\overline{D}(X) - \overline{D}(X-1) = \left| -\frac{2^N}{SQ} \cdot \sum_{i=1}^{n} z(x_i) - R + \sum_{i=1}^{n} z(x_i) \cdot R_i \cdot m_i \right|_{2^N} \tag{17}$$

By considering that

$$2^N \cdot \widetilde{D}(X) + \sum_{i=1}^{n} R_i \cdot x_i - \frac{2^N}{SQ} \cdot \sum_{i=1}^{n} z(x_i) - R + \sum_{i=1}^{n} z(x_i) \cdot R_i \cdot m_i = 2^N \cdot \widetilde{D}(X-1) + \sum_{i=1}^{n} R_i \cdot |x_i - 1|_{m_i} > 0, \tag{18}$$

we obtain

$$2^N \cdot \widetilde{D}(X) + \sum_{i=1}^{n} R_i \cdot x_i > \frac{2^N}{SQ} \cdot \sum_{i=1}^{n} z(x_i) + R - \sum_{i=1}^{n} z(x_i) \cdot R_i \cdot m_i \tag{19}$$

For the function $\widetilde{D}(X)$ to be strictly increasing, it is necessary to satisfy the two following conditions.

Condition 1. $2^N \cdot \widetilde{D}(X) + \sum_{i=1}^{n} R_i \cdot x_i < 2^N$

This inequality makes it possible to pass from computation of the remainder of the division to the computation mod 2^N for $\overline{D}(X)$ in Equality (13), and hence in Equality (16) as well.

Condition 2. $\frac{2^N}{SQ} \cdot \sum_{i=1}^{n} z(x_i) + R - \sum_{i=1}^{n} z(x_i) \cdot R_i \cdot m_i > 0$

If this inequality holds and both Condition 1 and Inequality (19) are satisfied, it implies that the value of $\overline{D}(X)$ calculated by Equality (13) is larger than the value of $\overline{D}(X-1)$ calculated by Equality (16).

Whether any of these two conditions is satisfied, it depends on N. Now we will show how to determine the smallest N, for which both Conditions 1 and 2 hold. As the function $\widetilde{D}(X)$ is monotonic, hence $\widetilde{D}(X) \leq \widetilde{D}(X-1)$. Thus, according to Equality (2)

$$\widetilde{D}(M-1) = \left| \sum_{i=1}^{n} \frac{k_1}{SQ} \cdot (m_i - 1) \right|_1 = 1 - \frac{n}{SQ} \tag{20}$$

Additionally, because $0 < R_i < 1$ and $\max\limits_{1 \leq i \leq n} m_i = m_n$, therefore

$$\sum_{i=1}^{n} z(x_i) \cdot R_i \cdot x_i < n \cdot (m_n - 1) \tag{21}$$

Because Condition 1 leads to the inequality

$$2^N \cdot \left(1 - \frac{n}{SQ}\right) + n \cdot (m_n - 1) < 2^N \tag{22}$$

therefore

$$1 - \frac{n}{SQ} + \frac{n \cdot (m_n - 1)}{2^N} < 1 \tag{23}$$

which in turn leads to $2^N > SQ \cdot (m_n - 1)$, and finally to

$$N > \log_2(SQ \cdot (m_n - 1)) \tag{24}$$

From Condition 1, and assuming that Inequality (24) holds, we have

$$\overline{D}(X) - \overline{D}(X-1) = \frac{2^N}{SQ} \cdot \sum_{i=1}^{n} z(x_i) + R - \sum_{i=1}^{n} z(x_i) \cdot R_i \cdot m_i \tag{25}$$

Now consider the inequality

$$\frac{2^N}{SQ} \cdot \sum_{i=1}^{n} z(x_i) + R - \sum_{i=1}^{n} z(x_i) \cdot R_i \cdot m_i > 0 \tag{26}$$

If the inequality

$$\frac{2^N}{SQ} > m_i \cdot R_i - R \tag{27}$$

holds for $1 \leq i \leq n$, then Inequality (26) is true for X with any number of zeros in its RNS representation. Therefore, we can estimate N from Inequality (27). Recalling that $0 \leq R_i < 1$, $1 \leq R < n$, and $\max\limits_{1 \leq i \leq n} m_i = m_n$, therefore

$$m_i \cdot R_i - R < m_n - 1, \tag{28}$$

which implies that Inequality (27) holds if

$$\frac{2^N}{SQ} > m_n - 1, \tag{29}$$

To estimate N, we calculate the logarithm of the last inequality

$$N > \log_2(SQ \cdot (m_n - 1)), \tag{30}$$

which is identical to Inequality (24). Therefore, if Inequality (24) holds, then both Conditions 1 and 2 are satisfied, which concludes the proof. □

Inequality (24) can be considered as the condition that guarantees strict monotonicity of the MDF $\overline{D}(X)$: if it holds, to compare two RNS numbers X and Y, it suffices to compare the values of $\overline{D}(X)$ and $\overline{D}(Y)$. The above considerations can be formally summarized as the following algorithm.

Algorithm 2: Comparison of RNS numbers using MDF.

Input: $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_n\}$
Output: "100" if $X < Y$, "010" if $X = Y$, and "001" if $X > Y$.
Step 1. Calculate $\overline{D}(X)$ and $\overline{D}(Y)$.
Note: These computations are independent and therefore they can be executed in parallel, provided that two circuits implementing the MDF are available.
Step 2. Compare the values of $\overline{D}(X)$ and $\overline{D}(Y)$.
1. if $\overline{D}(X) < \overline{D}(Y)$ then return "100"
2. if $\overline{D}(X) > \overline{D}(Y)$ then return "001"
3. if $\overline{D}(X) = \overline{D}(Y)$ then return "010"

In summary, the diagonal function $D(X)$ of [16,18] is monotonic for $0 \leq X < M$, whereas the MDF $\overline{D}(X)$ proposed here is strictly monotonic over this set (which makes it possible to compare numbers directly).

The differences between $D(X)$ and $\overline{D}(X)$ are illustrated for a sample 3-moduli RNS {5, 11, 17}. Figure 2 shows the diagrams of the values of the functions $D(X)$ and $\overline{D}(X)$ for the first 15 values of X, which clearly reflect their monotonic properties and demonstrate their differences.

Example 1. *Consider a sample set of* $n = 3$ *moduli* $m_1 = 13$, $m_2 = 15$, *nad* $m_3 = 17$ *whose dynamic range is* $M = 3315$. *Here, we have* $M_1 = \frac{M}{m_1} = 255$, $M_2 = \frac{M}{m_2} = 221$, $M_3 = \frac{M}{m_3} = 195$, *which implies* $SQ = M_1 + M_2 + M_3 = 255 + 221 + 195 = 671$, *so that* $N = \left\lceil \log_2((m_n - 1) \cdot SQ) \right\rceil = 14$ *and the four constants*

$$\bar{k}_1 = 2^{14} \cdot \frac{|-1/13|_{671}}{671} = 6300,\ \bar{k}_2 = 2^{14} \cdot \frac{|-1/15|_{671}}{671} = 12,014,\ \bar{k}_3 = 2^{14} \cdot \frac{|-1/17|_{671}}{671} = 14,456$$

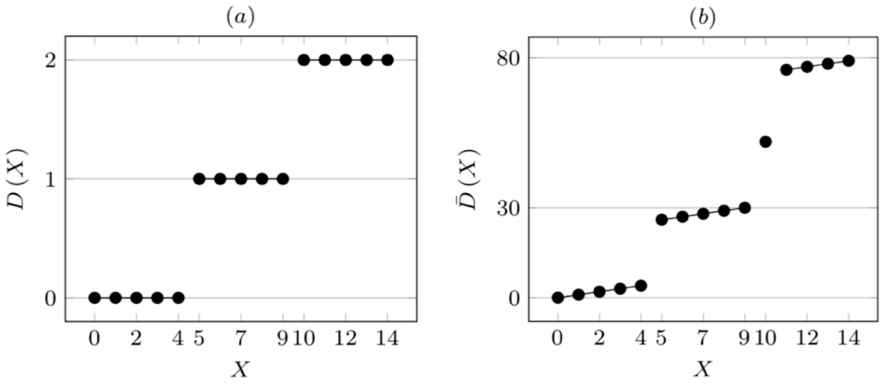

Figure 2. Values of the functions for $0 \leq X \leq 14$: (**a**) $D(X)$ and (**b**) $\overline{D}(X)$.

Let us compare three integers

$$X = 950 \overset{RNS}{\rightarrow} \{1,\ 5,\ 15\} Y = 951 \overset{RNS}{\rightarrow} \{2,\ 6,\ 16\} Z = 952 \overset{RNS}{\rightarrow} \{3,\ 7,\ 0\}$$

for which we have

$$\begin{aligned} \overline{D}(X) &= |1 \cdot 6300 + 5 \cdot 12,014 + 15 \cdot 14,456|_{2^{14}} = 4682 \\ \overline{D}(Y) &= |2 \cdot 6300 + 6 \cdot 12,014 + 16 \cdot 14,456|_{2^{14}} = 4684 \\ \overline{D}(Z) &= |3 \cdot 6300 + 7 \cdot 12,014 + 0 \cdot 14,456|_{2^{14}} = 4694 \end{aligned}$$

Recall that according to Theorem 1, the function $\overline{D}(X)$ is strictly monotonic for any $0 \leq X < M$. Indeed, it is seen that: (i) $\overline{D}(X) < \overline{D}(Y)$ implies $X < Y$ and (ii) $\overline{D}(Z) > \overline{D}(Y)$ implies $Z > Y$.

Figure 3 shows the general scheme of the new comparator that implements Algorithm 2 (here $b_i = \min\left\{\left\lfloor \log_2(m_i \cdot \overline{k}_i) \right\rfloor,\ N\right\}$). Obviously, it is necessary to add more extra hardware than its simple counterpart using any reverse converter followed by the a-bit comparator, because only the latter small circuit must be added to the RNS-based processor. In our circuit, both the circuit computing $\overline{D}(X)$ and the N-bit comparator must be added. Nevertheless, the new comparator has two potential major advantages over the latter: (i) higher speed, because of the delay of the n-operand MOMA mod 2^N is certainly significantly smaller than of the n-operand MOMA mod M [30,31] in the CRT-based version of the reverse converter or of its MRC-based version (a slightly larger size of the operands handled by the final N-bit comparator ($N = a + \left\lfloor \log_2 n \right\rfloor$ vs. a) has little impact on the area or delay of the final N-bit comparator); and (ii) lower power consumption, because of the significantly smaller circuitry involved in performing the comparison. Either claim will be confirmed by performance estimations obtained for ASIC implementations of various basic general RNS number comparators, presented in the next section.

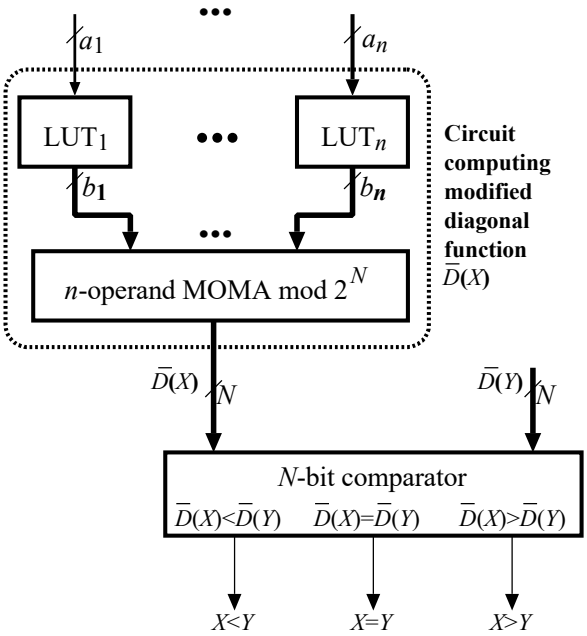

Figure 3. Hardware implementation of the new comparator built using the MDF.

4. Performance Estimations

In this section, we first present an approximate evaluation of the performance of hardware implementations of the new general RNS comparators and their three best-known counterparts, and then we provide more accurate estimations for ASIC implementations of all circuits considered.

Suppose that all n RNS moduli are l-bit numbers. Then, the basic parameters of operands involved in modulo operations handled by four general RNS number comparators can be summarized as listed in Table 1. First, recall that all these circuits have a number of steps growing logarithmically in the function of the number of moduli n, i.e., they all have $O(\log n)$ delay. Second, notice that the following inequalities hold: $\sqrt{M} < SQ < M < 2^N$ and $a_{\sqrt{M}} < a_{SQ} < a < N$. The three circuits based on the CRT, SQT, and MDF have a similar structure, composed of the n-operand modulo adder, with the major difference made by the modulus. Because neither M nor SQ is a power of 2 and $a > a_{SQ}$, then it seems that the SQT-based circuit should involve less hardware and be faster than the CRT-based one. On the other hand, because the MDF-based circuit proposed here uses the n-operand adder modulo a power of 2 (2^N), it enjoys the major advantage of all arithmetic circuits mod 2^N: significantly smaller delay and exceptional hardware efficiency compared to all its counterparts modulo any odd modulus involving cumbersome and lengthy operations of finding the remainder of the division by a large and awkward number M or SQ. The simplicity and the speed gained by the latter outweighs the minor delay/area differences due to a slightly larger final comparator of N-bit vs. a-bit and a_{SQ}-bit numbers. As for the comparator based on the CRT-II of [17] it executes $\lceil \log_2 n \rceil$ iterative steps on operands of growing size and involving computations modulo a size growing up to about $\sqrt{M}$. On one hand, $\sqrt{M}$ is not only the smallest of the moduli involved in computations by all comparators considered, but it is also involved only in the final stage of iterative computations, which suggests that it would result in some advantages. On the other hand, the estimation of delay/area performance of this circuit is difficult, because each iterative step involves modulo computations which, despite being executed on relatively small size moduli are nevertheless time-consuming and executed serially.

Table 1. Summary of basic parameters of four RNS comparison methods for general moduli sets.

Method	Operands		Modulus
	Size [bits]	Number	
CRT	$a \le n \cdot l$	n	M
SQT	$a_{SQ} \le (n-1) \cdot l + \lceil \log_2 n \rceil$	n	SQ
CRT-II †	$a_{\sqrt{M}} \le \lceil (n \cdot l)/2 \rceil$	$\lceil n/2 \rceil$	$\sqrt{M}$
MDF	$N \le n \cdot l + \lceil \log_2 n \rceil$	n	2^N

CRT—the direct method based on CRT; SQT—the method by Dimauro et al. [16]; CRT II—the method by Wang et al. [17]; MDF—the new method proposed here; † only the largest sizes of operands and moduli are indicated.

To obtain more accurate complexity estimations, we synthesized all four comparators described above for various RNS moduli sets, which are grouped into two classes, listed in Table 2. Class 1 consists of 4-moduli sets; each set composed of moduli of the same size p. Varying the size of moduli $p \in \{5, 7, 9, 11, 13\}$ makes it possible to observe comparators' performance in the function of the dynamic range M, which grows only with the size of the moduli but not with their number (which remains constant). Class 2 consists of moduli of the same size (we chose $p = 7$ bits), whose number n varies from 3 to 8, allowing to observe comparators' performance in the function of the number of moduli. All sets of selected moduli consist of the largest existing pairwise prime moduli for a given n.

Table 2. Sample sets of moduli and their characteristics.

Class	n	Moduli Set	Size of Moduli [Bits]	Size of M [Bits]	Size of SQ [Bits]	N
1	4	27, 29, 31, 32	5	20	17	22
		123, 125, 127, 128	7	28	23	30
		507, 509, 511, 512	9	36	29	38
		2043, 2045, 2047, 2048	11	44	35	46
		8187, 8189, 8191, 8192	13	52	41	54
2	3	125, 127, 128	7	21	16	23
	4	123, 125, 127, 128	7	28	23	30
	5	121, 123, 125, 127, 128	7	35	31	38
	6	119, 121, 123, 125, 127, 128	7	42	38	45
	7	113, 119, 121, 123, 125, 127, 128	7	49	45	52
	8	109, 113, 119, 121, 123, 125, 127, 128	7	56	52	59

The circuits were described in parametrized structural VHDL following identical coding guidelines and synthesized following the similar layout of module hierarchy and primitive components like adders. The additions and multiplications were implemented with register-transfer level (RTL) operators and selection of their architectures was left to be done by the synthesis tool. We performed logic synthesis of the comparators for a range of target moduli sets using Cadence RTL Compiler v. 8.1 and an industrial 65 nm low-power library (STM CMOS065LP). For each design and moduli set, the minimum delay was found, which we assumed to be the smallest delay target when the synthesis was still able to achieve a non-negative timing slack. The cell area and total power (including dynamic and leakage components) reported by the synthesis tool were given an area and power figures.

The complexity characteristics obtained are detailed in Tables 3–8 and visualized in Figures 4 and 5. It can be seen that the delay of the new comparator proposed here grows equally slowly (almost linearly) while increasing the dynamic range DR or the number of moduli n. It seems that it results directly from the possibility of replacing cumbersome operations of finding the remainder of the division by a large and awkward number with significantly simpler multi-operand additions mod 2^N. The synthesis results suggest that the new comparator proposed here is faster than all known similar circuits for all sample moduli sets considered, with delay reduction ranging from over 11% to

over 75% compared to the fastest circuit designed using existing methods. Only the basic CRT-based implementation introduces delay slightly larger but only in a few cases. The largest delay comes with the introduction of the comparator based on the CRT-II of [17].

Table 3. Delay [ps] for 4-moduli sets composed of p] for 4-moduli sets composed of p-bit moduli, $5 \leq p \leq 13$, p odd (Class 1).

p	*DR* [bits]	3	4	5	6	Reduction [%]		
		CRT	SQT	CRT-II	New	$\frac{3-6}{6}$	$\frac{4-6}{6}$	$\frac{5-6}{6}$
5	20	3187	2954	5401	2100	51.76	40.67	157.19
7	28	3993	3626	5903	2427	64.52	49.40	143.22
9	36	3132	4209	6624	2583	21.25	62.95	156.45
11	44	3430	5371	6844	2698	27.13	99.07	153.67
13	52	4099	5719	7248	2810	45.87	103.52	157.94

Table 4. Area [μm^2] for 4-moduli sets composed of p-bit moduli, $5 \leq p \leq 13$, p odd (Class 1).

p	*DR* [bits]	3	4	5	6	Reduction [%]		
		CRT	SQT	CRT-II	New	$\frac{3-6}{6}$	$\frac{4-6}{6}$	$\frac{5-6}{6}$
5	20	11537	16193	14398	11109	3.85	45.76	29.61
7	28	21732	23159	20827	15213	42.85	52.23	36.90
9	36	23073	31970	28910	21036	9.68	51.98	37.43
11	44	29733	54727	36486	26901	10.53	103.44	35.63
13	52	47798	67636	46237	32197	48.45	110.07	43.61

Table 5. Power [μW] for 4-moduli sets composed of p-bit moduli, $5 \leq p \leq 13$, p odd (Class 1).

p	*DR* [bits]	3	4	5	6	Reduction [%]		
		CRT	SQT	CRT-II	New	$\frac{3-6}{6}$	$\frac{4-6}{6}$	$\frac{5-6}{6}$
5	20	3108	3939	5784	2015	52.24	95.48	187.05
7	28	6850	6915	10068	3011	127.50	129.66	234.37
9	36	7721	11950	16371	4352	77.41	174.59	276.17
11	44	11338	25169	22576	6150	84.36	309.25	267.09
13	52	20588	32301	30468	7380	178.97	337.68	312.85

Table 6. Delay [ps] for various n-moduli sets, $3 \leq n \leq 8$] for various n-moduli sets, $3 \leq n \leq 8$ (Class 2).

n	*DR* [bits]	3	4	5	6	Reduction [%]		
		CRT	SQT	CRT-II	New	$\frac{3-6}{6}$	$\frac{4-6}{6}$	$\frac{5-6}{6}$
3	21	2465	3388	3831	1972	25.00	71.81	94.27
4	28	3993	3626	5903	2427	64.52	49.40	143.22
5	35	3479	5099	7861	3127	11.26	63.06	151.39
6	42	3682	6102	10950	3289	11.95	85.53	232.93
7	49	5941	6210	11243	3377	75.93	83.89	232.93
8	56	6115	5877	13521	3623	68.78	62.21	273.20

Table 7. Area [μm^2] for various n-moduli sets, $3 \leq n \leq 8$ (Class 2).

n	*DR* [bits]	3	4	5	6	Reduction [%]		
		CRT	SQT	CRT-II	New	$\frac{3-6}{6}$	$\frac{4-6}{6}$	$\frac{5-6}{6}$
3	21	9814	11281	7370	9690	1.28	16.42	−23.94
4	28	21732	23159	20827	15213	42.85	52.23	36.90
5	35	26130	36701	31836	28150	−7.18	30.38	13.09
6	42	43209	68508	51542	41463	4.21	65.23	24.31
7	49	93113	98583	72575	63926	45.66	54.21	13.53
8	56	107450	129063	103950	76654	40.18	68.37	35.61

Table 8. Power [μW] for various n-moduli sets, $3 \leq n \leq 8$ (Class 2).

n	*DR* [bits]	3	4	5	6	Reduction [%]		
		CRT	SQT	CRT-II	New	$\frac{3-6}{6}$	$\frac{4-6}{6}$	$\frac{5-6}{6}$
3	21	2750	3474	2063	1688	62.91	105.81	22.22
4	28	6850	6915	10068	3011	127.50	129.66	234.37
5	35	8523	14188	18247	7748	10.00	83.12	135.51
6	42	15461	31885	36415	12584	22.86	153.38	189.38
7	49	45114	43488	56798	19053	136.78	128.25	198.11
8	56	43278	54785	90474	26095	65.85	109.94	246.71

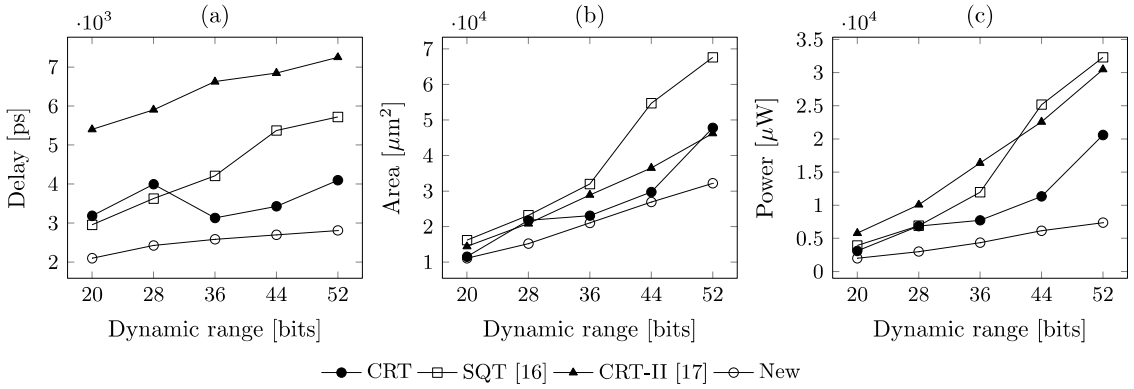

Figure 4. Complexity characteristics of RNS comparators for Class 1 moduli sets.

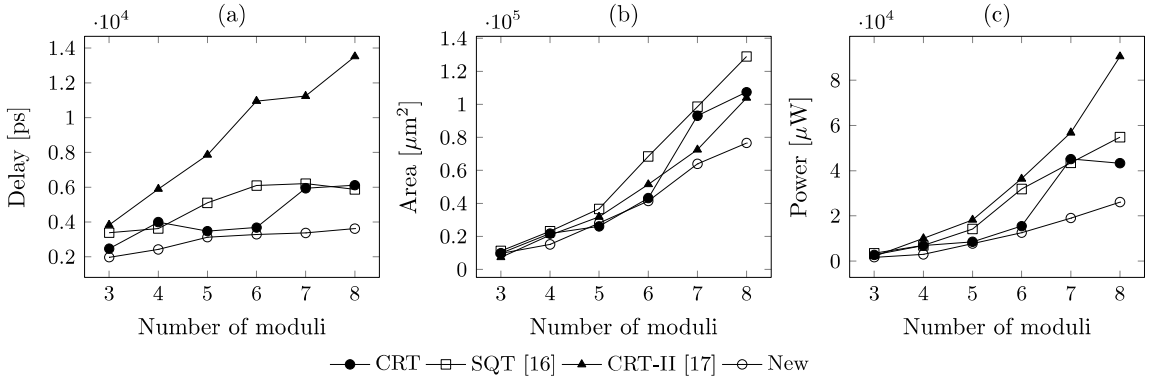

Figure 5. Complexity characteristics of RNS comparators for Class 2 moduli sets.

Moreover, the speed advantage of the new comparators was achieved using less hardware resources, with only two exceptions. For large dynamic ranges, hardware reduction is significant, as it can exceed 40% compared to the least complex existing designs. For all cases considered, the SQT-based method of [25] consumes more hardware resources than any other method. Hardware complexity of the basic CRT-based comparator deserves some special comments, because most of it is the reverse

converter, which is used anyway as a stand-alone circuit. Therefore, it should not be considered a contributor to the overall hardware complexity.

Finally, power-consumption seems the major advantage of the new comparators, as its reduction ranges from over 50% to over 178% for Class 1 moduli sets and from over 10% to over 130% for Class 2 moduli sets. Moreover, it was achieved using circuits which are faster for all cases considered. In this context, using specifically designed comparators instead of the CRT-based comparators (which actually require including the least amount of extra hardware: only the final comparator of a-bit numbers), could be of some practical interest. This is because once the reverse converter is activated just for the purpose of comparing numbers, it could be extremely power-consuming, as can be seen from the data listed in Tables 5 and 8, as well as shown in Figures 4c and 5c.

5. Conclusions

This paper proposes a new general approach to the comparison of the numbers represented in Residue Number System (RNS). It is based on a newly introduced concept of the modified diagonal function, which serves as a theoretical basis to develop a significantly faster and more efficient comparison algorithm. It made it possible to introduce a new positional characteristic of an RNS number which is strictly monotonic so that it makes it possible to precisely reflect a relative positioning of numbers. Now, unlike in existing algorithms, computations involving cumbersome operations of finding the remainder of the division by a large and awkward number are replaced with significantly simpler computations involving only a power of 2 modulus. The newly proposed comparator and its most efficient known counterparts applicable for arbitrary RNS moduli sets, designed using various methods for several sample moduli sets, were synthesized for the 65 nm technology. Performance estimations obtained suggest that the new circuits enjoy delay reduction ranging from over 11% to over 75%, compared to the fastest circuits designed using existing methods. Moreover, it is achieved using less hardware, the reduction can even reach over 41%, and accompanied by significantly reduced power-consumption which in several cases exceeds 100%. Therefore, it seems that the presented method leads to the design of what is currently the most efficient hardware comparators of numbers represented using a general RNS moduli set. The magnitude comparison of RNS numbers, besides being used directly (like in some implementations of recent cryptographic algorithms using RNS), is also essential for the implementation of other RNS non-modular operations like division, sign detection, and overflow detection. Future research will include extensions of the approach proposed to handle other difficult non-modular RNS operations like sign and overflow detection.

Author Contributions: Formal analysis, M.B., M.D. and S.J.P.; Funding acquisition, M.B. and M.D.; Investigation, M.B., M.D. and S.J.P.; Methodology, S.J.P. and A.T.; Project administration, M.B. and A.A.; Software, M.D. and P.P.; Supervision, N.C. and A.A.; Validation, M.B. and A.T.; Writing—Original draft, M.D., S.J.P. and A.T.; Writing—Review & editing, M.B., S.J.P. and A.T. All authors have read and agreed to the published version of the manuscript.

Funding: The reported study was funded by RFBR, project number 20-37-70023 and project NCFU.

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

Abbreviations and Symbols

CRT	Chinese Remainder Theorem
DR	Dynamic Range
MDF	Modified Diagonal Function
MOMA	Multi-Operand Modular Adder
MRC	Mixed Radix Conversion
RNS	Residue Number System
SQ	Sum of Quotients
SQT	Sum of Quotients Technique
a	the number of bits to represent M

a_i	the number of bits to represent x_i
a_{SQ}	the number of bits to represent SQ
$D(X)$	the diagonal function
$\overline{D}(X)$	the modified diagonal function
$h_i = \left\|1/m_i\right\|_{SQ}$	the multiplicative inverse of m_i mod SQ
$\{m_1, m_2, \cdots, m_n\}$	RNS moduli set
n	the number of moduli
N	the number of bits of the fraction part
x_i	the residue modulo (mod) m_i
$\{x_1, x_2, \cdots, x_n\}$	RNS representation of an integer X

References

1. Szabo, N.; Tanaka, R. *Residue Arithmetic and Its Applications to Computer Technology*; McGraw Hill: New York, NY, USA, 1967.
2. Mohan, P.V.A. *Residue Number Systems*; Birkhauser: Basel, Switzerland, 2016; ISBN 978-3-319-41383-9.
3. Nannarelli, A.; Cardarilli, G.C.; Re, M. Power-delay tradeoffs in residue number system. In Proceedings of the 2003 International Symposium on Circuits and Systems, 2003, ISCAS '03, Bangkok, Thailand, 25–28 May 2003; Volume 5, pp. V-413–V-416.
4. Cardarilli, G.C.; Del Re, A.; Nannarelli, A.; Re, M. Low-power implementation of polyphase filters in Quadratic Residue Number System. In Proceedings of the 2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512), Vancouver, BC, Canada, 23–26 May 2004; p. II-725.
5. Patronik, P.; Berezowski, K.; Piestrak, S.J.; Biernat, J.; Shrivastava, A. Fast and energy-efficient constant-coefficient FIR filters using residue number system. In Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design, Fukuoka, Japan, 1–3 August 2011; pp. 385–390.
6. Chervyakov, N.I.; Lyakhov, P.A.; Babenko, M.G. Digital filtering of images in a residue number system using finite-field wavelets. *Autom. Control Comput. Sci.* **2014**, *48*, 180–189. [CrossRef]
7. Keller, T.; Liew, T.H. Lajos Hanzo Adaptive redundant residue number system coded multicarrier modulation. *IEEE J. Sel. Areas Commun.* **2000**, *18*, 2292–2301. [CrossRef]
8. Posch, K.C.; Posch, R. Residue number systems: A key to parallelism in public key cryptography. In Proceedings of the Fourth IEEE Symposium on Parallel and Distributed Processing, Arlington, TX, USA, 1–4 December 1992; pp. 432–435.
9. Albicocco, P.; Cardarilli, G.C.; Nannarelli, A.; Re, M. Twenty years of research on RNS for DSP: Lessons learned and future perspectives. In Proceedings of the 2014 International Symposium on Integrated Circuits (ISIC), Singapore, 10–12 December 2014; pp. 436–439.
10. Chokshi, R.; Berezowski, K.S.; Shrivastava, A.; Piestrak, S.J. Exploiting residue number system for power-efficient digital signal processing in embedded processors. In *Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems—CASES '09*; ACM Press: New York, NY, USA, 2009; p. 19.
11. Patronik, P.; Piestrak, S.J. Hardware/software approach to designing low-power RNS-enhanced arithmetic units. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *64*, 1031–1039. [CrossRef]
12. Chang, C.-H.; Molahosseini, A.S.; Zarandi, A.A.E.; Tay, T.F. Residue number systems: A paradigm to datapath optimization for low-power and high-performance digital signal processing applications. *IEEE Circuits Syst. Mag.* **2015**, *15*, 26–44. [CrossRef]
13. Akushskii, I.J.; Burcev, V.M.; Pak, I.T. A new positional characteristic of non-positional codes and its application. In *Coding Theory and the Optimization of Complex Systems*; Nauka: Alma-Ata, Russia, 1977.
14. 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: New York, NY, USA, 1986; pp. 390–401.
15. Lu, M.; Chiang, J.-S. A novel division algorithm for the residue number system. *IEEE Trans. Comput.* **1992**, *41*, 1026–1032. [CrossRef]
16. Dimauro, G.; Impedovo, S.; Pirlo, G. A new technique for fast number comparison in the residue number system. *IEEE Trans. Comput.* **1993**, *42*, 608–612. [CrossRef]

17. Wang, Y.; Song, X.; Aboulhamid, M. A new algorithm for RNS magnitude comparison based on New Chinese Remainder Theorem II. In Proceedings of the Proceedings Ninth Great Lakes Symposium on VLSI, Ypsilanti, MI, USA, 4–6 March 1999; pp. 362–365.
18. 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]
19. Sousa, L. Efficient method for magnitude comparison in RNS based on two pairs of conjugate moduli. In Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH'07), Montpellier, France, 25–27 June 2007; pp. 240–250.
20. Bi, S.; Gross, W.J. The Mixed-Radix Chinese Remainder Theorem and Its Applications to Residue Comparison. *IEEE Trans. Comput.* **2008**, *57*, 1624–1632. [CrossRef]
21. Pirlo, G.; Impedovo, D. A new class of monotone functions of the residue number system. *Int. J. Math. Model. Methods Appl. Sci.* **2013**, *7*, 803–809.
22. Chervyakov, N.I.; Babenko, M.G.; Lyakhov, P.A.; Lavrinenko, I.N. An Approximate Method for Comparing Modular Numbers and its Application to the Division of Numbers in Residue Number Systems. *Cybern. Syst. Anal.* **2014**, *50*, 977–984. [CrossRef]
23. Piestrak, S.J. A note on RNS architectures for the implementation of the diagonal function. *Inf. Process. Lett.* **2015**, *115*, 453–457. [CrossRef]
24. Chervyakov, N.I.; Molahosseini, A.S.; Lyakhov, P.A.; Babenko, M.G.; Lavrinenko, I.N.; Lavrinenko, A.V. Comparison of modular numbers based on the Chinese remainder theorem with fractional values. *Autom. Control Comput. Sci.* **2015**, *49*, 354–365. [CrossRef]
25. Sousa, L.; Martins, P. Sign Detection and Number Comparison on RNS 3-Moduli Sets $\{2^n - 1, 2^{n+x}, 2^n + 1\}$. *Circuits Syst. Signal Process.* **2017**, *36*, 1224–1246. [CrossRef]
26. Vu, T.V. Efficient Implementations of the Chinese Remainder Theorem for Sign Detection and Residue Decoding. *IEEE Trans. Comput.* **1985**, *C–34*, 646–651. [CrossRef]
27. Hung, C.Y.; Parhami, B. Error analysis of approximate Chinese-remainder-theorem decoding. *IEEE Trans. Comput.* **1995**, *44*, 1344–1348. [CrossRef]
28. Akkal, M.; Siy, P. A new Mixed Radix Conversion algorithm MRC-II. *J. Syst. Archit.* **2007**, *53*, 577–586. [CrossRef]
29. Hwang, K. *Computer Arithmetic Principles, Architecture, and Design*; Wiley: New York, NY, USA, 1979.
30. Piestrak, S.J. Design of residue generators and multioperand modular adders using carry-save adders. *IEEE Trans. Comput.* **1994**, *43*, 68–77. [CrossRef]
31. Piestrak, S.J. Design of high-speed residue-to-binary number system converter based on Chinese Remainder Theorem. In Proceedings of the 1994 IEEE International Conference on Computer Design: VLSI in Computers and Processors, Cambridge, MA, USA, 10–12 October 1994; pp. 508–511.