computation

MDPI

Article

Improved Modular Division Implementation with the Akushsky Core Function

Mikhail Babenko [1,2], Andrei Tchernykh [2,3,4,*] and Viktor Kuchukov [1]

1 North-Caucasus Center for Mathematical Research, North-Caucasus Federal University, 355017 Stavropol, Russia; mgbabenko@ncfu.ru (M.B.); vkuchukov@ncfu.ru (V.K.)
2 Institute for System Programming of the Russian Academy of Sciences, 109004 Moscow, Russia
3 CICESE Research Center, Ensenada 22860, Mexico
4 School of Electrical Engineering and Computer Science, South Ural State University, 454080 Chelyabinsk, Russia
* Correspondence: chernykh@cicese.mx

Abstract: The residue number system (RNS) is widely used in different areas due to the efficiency of modular addition and multiplication operations. However, non-modular operations, such as sign and division operations, are computationally complex. A fractional representation based on the Chinese remainder theorem is widely used. In some cases, this method gives an incorrect result associated with round-off calculation errors. In this paper, we optimize the division operation in RNS using the Akushsky core function without critical cores. We show that the proposed method reduces the size of the operands by half and does not require additional restrictions on the divisor as in the division algorithm in RNS based on the approximate method.

Keywords: Akushsky core function; residue number system; RNS; modular division

check for updates

Citation: Babenko, M.; Tchernykh, A.; Kuchukov, V. Improved Modular Division Implementation with the Akushsky Core Function. *Computation* **2022**, *10*, 9. https://doi.org/10.3390/computation10010009

Academic Editor: Demos T. Tsahalis

Received: 19 December 2021
Accepted: 11 January 2022
Published: 13 January 2022

1. Introduction

The results of studies on improving the performance of computing systems show that within the limits of positional number systems, a significant improvement cannot be expected without a considerable increase in the operating frequencies of elements and complications of the hardware of digital computing structures [1]. An important issue is to choose an effective method for encoding numerical information, i.e., selecting a number representation for fast processing. The residue number system (RNS) is used to improve the efficiency of data encryption algorithms [2–4], cloud computing [5–7], digital signal processing [8,9], wireless networks [10], matrix computing [11,12], and artificial neural networks [13,14]. One of the computationally complex operations in RNS is the Euclidean division or remainder division. Reducing the computational complexity of the remainder division algorithm will expand the range of RNS applicability for more efficient use of it in the implementation of numerical methods, etc.

Positional number systems, in which information is presented and processed in modern computing devices, have drawbacks. The main one for the speed limit is the presence of inter-digit transfers. They impose restrictions on the methods of implementing arithmetic operations. Therefore, it seems natural to construct an arithmetic system with no inter-bit transfers, i.e., a non-weighted number system. One of such systems is the RNS, where numbers are represented by the remainders of division by the selected bases of the system, and operations can be performed in parallel on each digit independently.

The development of computing devices based on the RNS began in the 1950s to 1960s of the 20th century. They were implemented in the form of modular coprocessors [15].

If a series of positive integers $p_1, p_2, \ldots, p_n$, called moduli or bases of the system, is given, then the RNS is a system in which a positive integer is represented as a set of remainders obtained by division by the chosen base $X = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, where $\alpha_i = X \; mod \; p_i$ for $i = 1,\ 2,\ \ldots,\ n$ [1].

It is known from number theory that if the moduli p_i are coprime, then the representation of the number $X = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is unique, and X satisfies the condition $X < P$, where $P = p_1 \cdot p_2 \cdot \ldots \cdot p_n$ is a dynamic range of number representation.

For numbers $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ the following expression holds:

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

where $* = \{+, -, \times\}$.

However, despite several advantages, the RNS has the following disadvantages: limited action of this system by the field of positive integers, difficulty in determining the ratios of numbers in magnitude, determining if the result of an operation is out of range, etc.

Operations in the RNS can be divided into two groups: modular, where calculations are performed for each digit independently, and non-modular, which require, to one degree or another, knowledge of the positional characteristics of the number.

In Section 2, we consider the features of division in the RNS, an approximate method based on the Chinese remainder theorem, the Akushsky core function, and also a block diagram of the division algorithm is presented. Section 3 considers examples of the division operation implementation in the RNS in the form of a computing system. Section 4 presents the main results of the work and directions for further research.

2. Features of Division in the Residue Number System

The division is one of the primary arithmetic operations. However, in RNS, the implementation of the modular division is computationally complex. There are methods for performing division with numbers of a specific type, for example, division with zero remainders, scaling, etc. [16].

The well-known division algorithms in the RNS [17–19] are based on comparing and subtracting numbers.

Let a dividend X, a divisor Y, a quotient Q, and a remainder R be given. Then $R = X - Y \cdot Q$, while $R < Y$. Consider the division algorithm based on the sequential approximation of the quotient Q by degrees of the base of the number system, i.e., for a binary system, the process consists in finding $q_i = \{0, 1\}$ such that the equality holds

$$Q = q_n \cdot 2^n + q_{n-1} \cdot 2^{n-1} + \cdots + q_1 \cdot 2^1 + q_0 \cdot 2^0. \tag{1}$$

Substituting (1) into the division formula, we obtain

$$R = X - Y \cdot q_n \cdot 2^n - Y \cdot q_{n-1} \cdot 2^{n-1} - \cdots - Y \cdot q_1 \cdot 2^1 - Y \cdot q_0 \cdot 2^0. \tag{2}$$

Thus, the division process can be reduced to a sequence of subtractions. Let 2^n enter into the representation of the quotient Q, that is, $q_n = 1$, then we denote $\Delta_1 = X - 2^n \cdot Y$, and $\Delta_1 \geq 0$. Substitute Δ_1 in (2).

$$R = \Delta_1 - Y \cdot q_{n-1} \cdot 2^{n-1} - \cdots - Y \cdot q_1 \cdot 2^1 - Y \cdot q_0 \cdot 2^0.$$

Let us continue this process. We denote $\Delta_2 = \Delta_1 - 2^{n-1} \cdot Y$. Since Δ_i is the sum of the remainder of the division and remaining members of the sequence of degrees of the number system are multiplied by the divisor, then $\Delta_i \geq 0$ is always satisfied.

If 2^k is not included in the representation of the quotient Q, i.e., $q_k = 0$, then $\Delta_{n-k-1} = \Delta_{n-k-2} - 2^k \cdot Y < 0$. It is necessary to check the occurrence of 2^{k-1} for which $\Delta_{n-k} = \Delta_{n-k-2} - 2^{k-1} \cdot Y$ is calculated.

In RNS, any number $X < P$ is unambiguously represented by a set of residues x_i of dividing the number X by relatively prime moduli of the RNS p_i, where $x_i \equiv X \; mod \; p_i$, $P = \prod_{i=1}^{n} p_i$ is the working range of the RNS, $i = \overline{1, n}$. The recovery of the number X

from the RNS to the positional number system can be done, as in the prototype, using the approximate Chinese remainder theorem

$$F(X) = \frac{X}{P} = \left| \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} \cdot x_i \right|_P = \left| \sum_{i=1}^{n} k_i \cdot x_i \right|_1 ,$$

where $k_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i}$, $P_i = P/p_i$, $\left|P_i^{-1}\right|_{p_i}$ is a multiplicative inverse. The application of the approximate method based on the Chinese remainder theorem is considered, in particular, in the patent [20]. However, the k_i coefficient rarely turns out to be a finite fraction. Its rounding leads to accumulation errors.

The sign in the RNS is most often introduced by dividing the range into two parts, then, taking into account the dynamic range P, in the RNS, it is possible to represent the numbers

$-\frac{P-1}{2} \leq X \leq \frac{P-1}{2}$, if P is odd,

$-\frac{P}{2} \leq X \leq \frac{P}{2} - 1$, if P is even.

Then,

X is positive if $0 \leq X \leq \frac{P}{2} - 1$, if P is even, $0 \leq X \leq \frac{P-1}{2}$, if X is odd,

X is negative if $\frac{P}{2} \leq X < P$, if P is even, $\frac{P+1}{2} \leq X < P$, if X is odd.

To perform division according to formula (2), it is necessary to compare RNS numbers and determine their signs.

Since the RNS is a non-weighted number system, then for comparing numbers and determining the sign, i.e., finding the position of the number on the number line, it is necessary to calculate the positional characteristic. An example of a positional characteristic is the Chinese remainder theorem with fractions used in the prototype. Another positional characteristic can be a core function, introduced by I. Ya. Akushsky [21,22]:

$$C(X) = \sum_{i=1}^{n} w_i \frac{X}{p_i}. \tag{3}$$

The numbers w_i, called weights, can be arbitrary. They define each specific core function and can vary depending on the task. The essential property of the core function is that its maximum range can vary and can be significantly less than the P number, depending on the choice of weights. For example, you can use as $C(P)$ some arbitrary value C_P, which has the properties necessary for solving a specific problem. The values of the core function $C(X)$, specified by the weights $w_1, w_2, \ldots, w_n$, under the condition $0 \leq C(X) < C_P$, $X \in [0, P)$, can be calculated using the formula

$$C(X) = \left| \sum_{i=1}^{n} C(B_i) \cdot x_i \right|_{C_P} , \tag{4}$$

where $B_i = P_i \cdot \left|P_i^{-1}\right|_{p_i}$ are the orthogonal bases of RNS. However, in general, the core function does not have the monotonicity required for comparing numbers.

To construct a core function with specified properties, we use the following algorithm (Algorithm 1).

Algorithm 1: Selection of parameters for the core function of a special type for a given set of moduli.

Input: Set of RNS moduli $p_1, p_2, \dots, p_n$. It is required to construct a core function with a module of a special type with $C_P = R(N)$ and non-negative coefficients.
Output: Coefficients $w_1, w_2, \dots, w_n$ of the constructed core function.

```
1. Let $N = log_2 P_n$.
2. For the given value N calculate $\mathrm{w_i^* = \left|R(N) \cdot P_i^{-1}\right|_{p_i}, i = 1,2,\dots,n, and\ C_P^* = P \cdot \sum_{i=1}^{n} \frac{w_i^*}{p_i}}$,
   where $\mathrm{R(N) = 2^N}$.
3. Calculate $Q$ by the formula $Q = \frac{R(N) - C_P^*}{P}$.
4. If $Q < 0$, then $N = N + 1$ and go to step 2. Otherwise, go to step 5.
5. Choose such a $q_i$ that $Q = q_1 + q_2 + \dots + q_n$. Calculate $w_i = q_i \cdot p_i + w_i^*$ for $i = 1, 2, \dots, n$.
6. Check the conditions for the absence of critical cores from below: $C(p_k) = \sum_{i=1}^{k} w_i \frac{p_k}{p_i} \geq 0$
   and the absence of critical cores from above: $\sum_{i=1}^{n} \left(\frac{p_k}{p_i} + 1\right) w_i - w_k > 0$, for all
   $k = 1, 2, \dots, n$. If it does not hold, $N = N + 1$ and go to step 2.
   end.
```

The core function with the given properties is given by expression (4), where

$$C(B_i) = \frac{B_i \cdot C_P}{P} - \frac{w_i}{p_i}, \text{ and } B_i = P_i \cdot \left|P_i^{-1}\right|_P.$$

To compare numbers, let us use the following Algorithm 2.

Algorithm 2: Comparison of numbers represented in the RNS using a core function with non-negative coefficients.

Input: $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$
Output: $X < Y$, $X > Y$ or $X = Y$

```
1. Calculate $C(X)$ and $C(Y)$.
2. Comparison
    2.1. If $C(X) < C(Y)$, then $X < Y$
    2.2. If $C(X) > C(Y)$, then $X > Y$
    2.3. If $C(X) = C(Y)$, then
        2.3.1. if $x_k < y_k$, then $X < Y$
        2.3.2. if $x_k > y_k$, then $X > Y$
        2.3.3. if $x_k = y_k$, then $X = Y$
end.
```

In this case, non-negative coefficients $w_1,\ w_2,\ \cdots,\ w_n$ are taken, and $w_k \neq 0$ is the first non-zero coefficient.

To determine the sign of a number, it is necessary to construct a core function such that $C(X) \leq C(K)$ for positive numbers and $C(X) > C(K)$ for negative numbers, where $K = P/2$ if P is even ($K = (P-1)/2$ if P is odd). K is the middle of the RNS range. Therefore, use Algorithm 2 for X and K.

Let us consider an example of division in RNS based on function (4) and Algorithm 2.

We take $\{11, 13, 17, 19\}$ as RNS. Then $P = 46{,}189$, $P_1 = 4199$, $P_2 = 3553$, $P_3 = 2717$, $P_4 = 2431$, $P_1^{-1} = 7$, $P_2^{-1} = 10$, $P_3^{-1} = 11$, $P_4^{-1} = 18$, $B_1 = 29{,}393$, $B_2 = 35{,}530$, $B_3 = 29{,}887$, $B_4 = 43{,}758$. Using Algorithm 1, we obtain $N = 17$, $w_1 = 16$, $w_2 = 8$, $w_3 = 5$, $w_4 = 9$.

Then, the auxiliary values $C(B_i)$ are equal to $C(B_1) = 83{,}408$, $C(B_2) = 100{,}824$, $C(B_3) = 84{,}811$, $C(B_4) = 124{,}173$.

The core function takes the form

$$C(X) = |83{,}408 \cdot x_1 + 100{,}824 \cdot x_2 + 84{,}811 \cdot x_3 + 124{,}173 \cdot x_4|_{2^{17}}.$$

The middle of the RNS range is $K = 23{,}094$, for which $C(K) = 65{,}517$.

We find the quotient of dividing $X = (5,\ 4,\ 3,\ 5)$ by $Y = (1,\ 10,\ 6,\ 4)$. Let us check the signs of the dividend and the divisor, for which we calculate $C(X)$ and $C(Y)$:

$C(X) = 122{,}770 > 65{,}517 = C(K)$, the number is negative,

$C(Y) = 54 < 65{,}517 = C(K)$, the number is positive.

Since the dividend and divisor are different signs, the result will be negative. For the dividend, we find the opposite value, to perform division over the absolute values. For this, in the RNS, it is necessary to subtract the corresponding remainder from the modulus.

$$-X = (p_1 - x_1,\ p_2 - x_2,\ \ldots,\ p_n - x_n) = (11-5; 13-4; 17-3; 19-5) = (6, 9, 14, 14).$$

We get $|X| = (6, 9, 14, 14)$. Representations of powers "2" in RNS can be calculated in advance, depending on the range of RNS (the highest occurrence power of 2^n is $\lfloor \log_2 K \rfloor$) and stored in memory:

$$\begin{aligned}
&2^{14} = (5,4,13,6),\ 2^{13} = (8,2,15,3),\\
&2^{12} = (4,1,16,11), 2^{11} = (2,7,8,15),\\
&2^{10} = (1,\ 10,4,17),\ 2^{9} = (6,5,2,18),\\
&2^{8} = \ (3,9,1,9), 2^{7} = (7,11,9,14),\\
&2^{6} = \ (9,\ 12,13,7),\ 2^{5} = (10,6,15,13),\\
&2^{4} = \ (5,3,16,16), 2^{3} = (8,8,8,8),\\
&2^{2} = \ (4,4,4,4),\ 2^{1} = (2,2,2,2),\\
&2^{0} = \ (1,1,1,1).
\end{aligned}$$

The highest possible degree of quotient when performing division is equal to the dimension of the dividend. It is necessary to multiply the divisor sequentially by 2^i to find the number for which the values of the core function satisfy the expression $C(|X|) \le C(2^i|Y|)$.

$$C\left(2^6|Y|\right) = 4155 < C(|X|) = 8264 < C\left(2^7|Y|\right) = 8331$$

Using the formula (2), we calculate $\Delta_1 = X - 2^7 \cdot\ Y$:

$$\Delta_1 = (6,9,14,14) - (7,11,9,14)\cdot(1,10,6,4) = (10,3,11,15),\ \Delta_1 < 0.$$

It means that 2^7 is not included in the representation of the quotient Q. Let us check 2^6 by calculating $\Delta_2 = \mathrm{X} - 2^6 \cdot \mathrm{Y}$:

$$\Delta_2 = (6,9,14,14) - (9,\ 12,13,7) \cdot (1,\ 10,\ 6,\ 4) = (8,6,4,5),\ \Delta_2 > 0.$$

It means that 2^6 is included in the representation of the quotient Q. Let us check 2^5, by calculating $\Delta_3 = \Delta_2 - 2^5 \cdot \mathrm{Y}$:

$$\Delta_3 = (8,6,4,5) - (10,6,15,13) \cdot (1,\ 10,\ 6,\ 4) = (9,11,16,10),\ \Delta_3 > 0.$$

It means that 2^5 is included in the representation of the quotient Q. Let us check 2^4, by calculating $\Delta_4 = \Delta_3 - 2^4 \cdot \mathrm{Y}$:

$$\Delta_4 = (9,11,16,10) - (5,3,16,16) \cdot (1,\ 10,\ 6,\ 4) = (4,\ 7,\ 5,\ 3),\ \Delta_4 > 0.$$

It means that 2^4 is included in the representation of the quotient Q. Let us check 2^3 by calculating $\Delta_5 = \Delta_4 - 2^3 \cdot \mathrm{Y}$:

$$\Delta_5 = (4,\ 7,\ 5,\ 3) - (8,8,8,8) \cdot (1,\ 10,\ 6,\ 4) = (7,\ 5,\ 8,\ 9),\ \Delta_5 > 0.$$

It means that 2^3 is included in the representation of the quotient Q. Let us check 2^2 by calculating $\Delta_6 = \Delta_5 - 2^2 \cdot Y$:

$$\Delta_6 = (7,\ 5,\ 8,\ 9) - (4,4,4,4) \cdot (1,\ 10,\ 6,\ 4) = (3,4,1,12),\ \Delta_6 > 0.$$

It means that 2^2 is included in the representation of the quotient Q. Let us check 2^1 by calculating $\Delta_7 = \Delta_6 - 2^1 \cdot Y$:

$$\Delta_7 = (3,4,1,12) - (2,2,2,2) \cdot (1,\ 10,\ 6,\ 4) = (1,10,6,4),\ \Delta_7 > 0.$$

It means that 2^1 is included in the representation of the quotient Q. Let us check 2^0 by calculating $\Delta_8 = \Delta_7 - 2^0 \cdot Y$:

$$\Delta_8 = (1,10,6,4) - (1,1,1,1) \cdot (1,\ 10,\ 6,\ 4) = (0,0,0,0),\ \Delta_7 = 0.$$

It means that 2^0 is included in the representation of the quotient Q.
Therefore,

$$|Q| = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 127.$$

Since the result must be negative, then $Q = -127$. Let us check:

$$-\frac{2921}{23} = -127.$$

3. Implementation of the RNS Division

Let us consider the block diagram of the division of numbers represented in the RNS (Figure 1) [23].

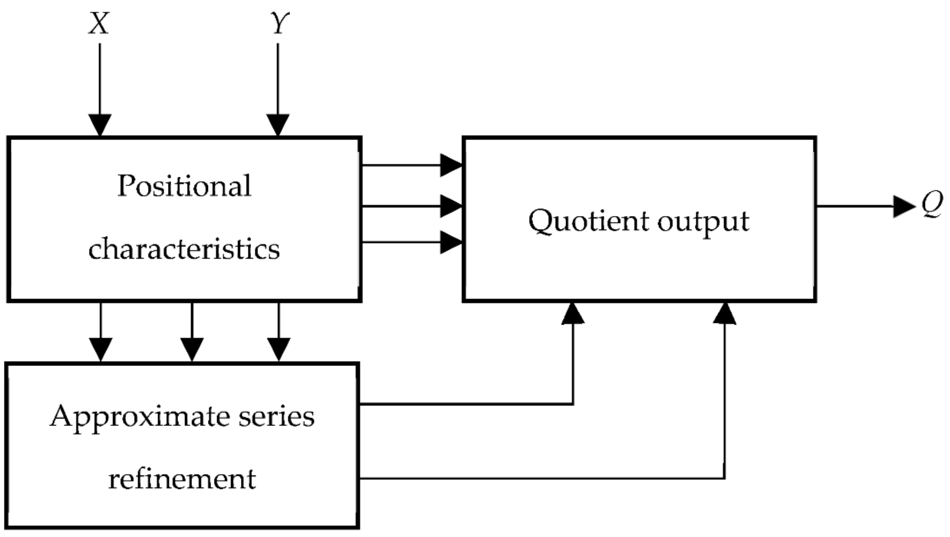

Figure 1. General block diagram for calculating division.

Figure 1 shows the general block diagram of the division calculation, which contains the input of the dividend X, input of the divisor Y, block for calculating positional characteristics, block for refining the approximation series, block for the derivation of the quotient, and output of the quotient Q. The inputs of the dividend X and the divisor Y are connected to the first and second input blocks for calculating positional characteristics.

From the first output of the block for calculating positional characteristics, the sign value of the result is fed to the first input of the quotient output block.

From the second output of the block for calculating positional characteristics, the signal "$|X| < |Y|$" is sent to the second input of the quotient output block, i.e., the result of the division is 0.

From the third output of the block for calculating positional characteristics, the signal "$|X| = |Y|$" is sent to the third input of the quotient output block, i.e., the division result is ± 1, depending on the signs of the input numbers.

From the fourth and sixth outputs of the block for calculating positional characteristics, the absolute values of the dividend $|X|$ and divisor $|Y|$ are sent.

From the fifth output of the block for calculating positional characteristics, the value of the core function of the absolute value of the dividend $C(|X|)$ is sent to the second input of the block for refining the approximation series.

The signal of the end of enumeration of powers "2" included in the representation of the quotient Q is received from the first output of the block for refining the approximation series to the fourth input of the quotient output unit.

The first output of the quotient output block is the quotient output.

Figure 2 shows a block diagram for calculating positional characteristics, which contains inverters of the dividend and divider, blocks for multiplying by constants and for addition, blocks for determining the sign, dividend and divisor multiplexers, an XOR element, and a comparison block.

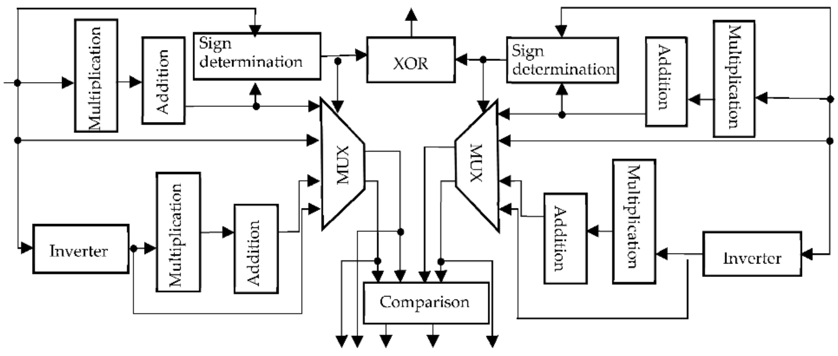

Figure 2. Positional characteristics calculation block diagram.

At the inputs of the dividend and the divisor, the values of the dividend X and the divisor Y are sent, represented in RNS as $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$.

In inverters of the dividend and divisor, the opposite values $-X$ and $-Y$ are calculated, correspondingly.

In the RNS, to obtain the opposite value $-X$, it is necessary to subtract the corresponding remainder from the modulus $(p_1 - x_1, p_2 - x_2, \ldots, p_n - x_n)$. Then, the numbers X and $-X$, Y and $-Y$ in the blocks of multiplication by constants are multiplied by the constants of the values of the core function of the RNS orthogonal bases $C(B_i)$, i.e., in each block, there is a parallel multiplication of the residues by $C(B_i)$ according to the formula (4).

The values of the products from the multiplication blocks by the constants are fed to the inputs of the addition blocks, the output of which is the lowest N bits of the sum, which corresponds to finding the remainder modulo 2^N in formula (4). N is determined in advance by Algorithm 1 while constructing the core function.

The values of the dividend and the divisor from the inputs of the dividend and the divisor, respectively, are sent to the first inputs of the sign determination blocks. The values of the core function $C(X)$ and $C(Y)$ from the outputs of the addition blocks are sent to the second inputs of the sign determination blocks. In the blocks for determining the sign, the values of the core function and the remainders are compared on one of the bases with the middle of the RNS range K according to Algorithm 2.

The signs of X and Y from the outputs of the blocks for determining the sign are sent to the inputs of the XOR element, as well as to the control inputs of the corresponding multiplexers. The XOR element output is the first output of the positional characteristic calculation block.

The first and second inputs of the multiplexer of the dividend receive the value of the core function $C(X)$ from the output of the addition block and the value of the dividend X from the input of the dividend.

The third and fourth information inputs of the multiplexer receive the value of the core function $C(-X)$ from the output of the addition block and the value $-X$ from the output of the inverter.

The first output of the multiplexer is connected to the second input of the comparison unit and the fifth output of the positional characteristics calculating unit and transfers the value of the core function from the absolute value of the dividend $C(|X|)$. The second output of the multiplexer is connected to the first input of the comparison unit and is the fourth output of the unit for calculating positional characteristics. It transmits the absolute value of the dividend $|X|$.

The first and second inputs of the divider multiplexer receive the value of the core function $C(Y)$ from the output of the addition block, and the value of the divider Y from the input of the divider. The third and fourth information inputs of the multiplexer receive the value of the core function $C(-Y)$ from the output of the addition block, and the value $-Y$ from the output of the inverter. The first output of the multiplexer is connected to the third input of the comparison unit. It transfers the value of the core function from the absolute value of the divisor $C(|Y|)$. The second output of the multiplexer is connected to the fourth input of the comparison unit, is the sixth output of the positional characteristics calculating unit and transmits the absolute value of the divider $|Y|$.

The comparison block is based on Algorithm 2. It compares the absolute values of the dividend $|X|$ and divisor $|Y|$ with $C(|X|)$ and $C(|Y|)$ values, respectively. It sends to the first output of the comparison unit a signal in the case of "$|X| < |Y|$", which is fed to the second output of the positional characteristics calculating unit. It sends a signal to the second output of the comparison unit in the case of "$|X| = |Y|$", which is fed to the third output of the block for calculating positional characteristics.

Figure 3 shows the block for refining the approximation series. It contains the storage register of the reduced, storage register of the divisor, modulo multiplying register, storage register of degrees "2", power counting unit, demultiplexer, multiplexer, modulo subtraction block, multiplication by the constants blocks, addition blocks, block for determining the sign, and logical element AND.

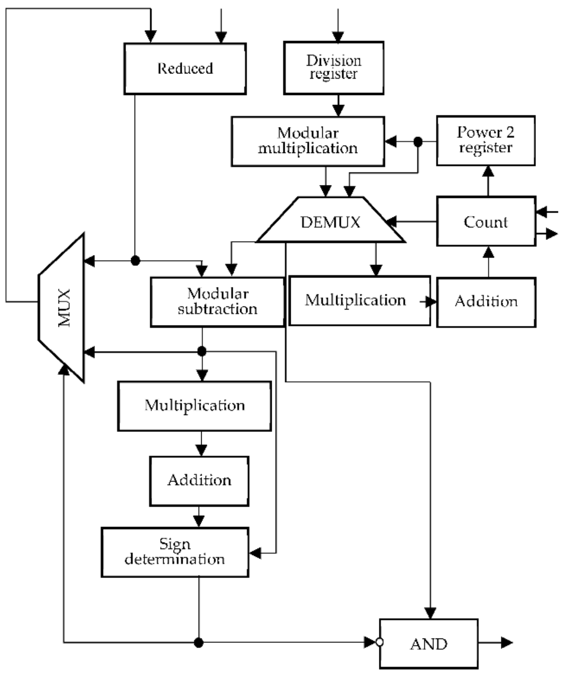

Figure 3. Refinement of the approximation series block diagram.

The value $|X|$ from the first input of the refinement block of the approximation series is received at the first input of the storage register of the reduced one. The input of the divider storage register is the third input of the approximation series refinement block. It transmits the value $|Y|$ to the first input of the modulo multiplication unit, where the divider is multiplied by the power of "2", presented in the RNS, which come from the first output of the storage register of the powers of "2".

Additionally, degrees "2" from the first output of the storage register are fed to the second input of the demultiplexer. The first input receives the value of the product from the output of the multiplying unit.

The degree counting unit determines the degree "2", for which the product $2^i \cdot |Y| \geq |X|$ and then it counts down the degrees to check the occurrence of the power "2" in the representation of the quotient Q. To determine the maximum degree "2", the power count unit supplies the first output connected to the first input of the power storage register "2" with the address values starting from 1, while the maximum degree is equal to $\lfloor \log_2 K \rfloor$, the storage register of degrees "2" supplies the first output of degrees "2" presented in the RNS.

The second output, connected to the control input of the demultiplexer, is supplied with the value of the operating mode: direct or countdown degrees.

In the direct counting case, the value of the product from the output of the modulo multiplication unit is fed to the third output of the demultiplexer. It is connected to the input of the multiplication unit by a constant where the values of the core function from the orthogonal bases of the RNS $C(B_i)$ are multiplied and then added in the addition block.

The least significant N bits of the result are fed to the second input of the power counting block, where it is compared with the value of the dividend core function, which is fed to the first input of the power counting block from the second input of the approximation series refinement block.

In the countdown case, the value of the product from the output of the modulo multiplying unit is fed to the first output of the demultiplexer, to the second output of which the value of the power "2" is supplied. At the end of the countdown, a signal about the end of counting is sent to the third output of the power counting unit.

In the subtraction unit, subtraction is performed according to formula (2). The first input is fed from the output of the storage register of the reduced product, and the second one is fed from the first output of the demultiplexer.

The result of subtraction is fed to the input of the multiplying unit by constants, from where it is fed through the addition unit to the first input of the sign determination unit, the second input of which is connected to the output of the modulo subtraction unit, which is also connected to the second information input of the multiplexer.

The output of the sign determination unit is connected through the inverter to the first output of the AND logic element, the second input of which receives degrees "2" from the second output of the demultiplexer, and to the control input of the multiplexer, the first input of which is connected to the output of the storage register of the decreasing one, and the output of the multiplexer is connected to the second to the input of the storage register of the decrement. The output of the AND gate is the second output of the approximation series refinement block.

Figure 4 shows a quotient output block containing a modulo addition block, demultiplexer, inverter, storage register "1" in the RNS, storage register "-1" in the RNS, private multiplexer, unit multiplexer, quotient selection multiplexer, and the AND logic gate.

Degrees "2" from the fifth input of the quotient output unit are fed to the first input of the modulo addition unit. Its output is connected to the output of the demultiplexer.

Depending on the end-of-count signal, it connects to the demultiplexer control input from the fourth input of the quotient output unit, feeds the result to the second input of the block addition modulo or to the second input of the quotient multiplexer and to the first input of the quotient multiplexer through the inverter.

The signal of Q sign from the first input of the quotient output unit is fed to the control inputs of the quotient multiplexer and unit multiplexer. Its first and second information inputs receive signals from the outputs of the storage register "1" in the RNS and the storage register "-1" in the RNS, respectively.

The outputs of the quotient multiplexer and unit multiplexer are connected to the first and second inputs of the private selection multiplexer. Its control input receives the signal "$|X| = |Y|$" from the third input of the private output block. The output of the private selection multiplexer is connected to the first input of the AND gate. The signal "$|X| < |Y|$" is supplied through the inverter from the second input of the private output. The output of the AND gate is the output of the quotient Q.

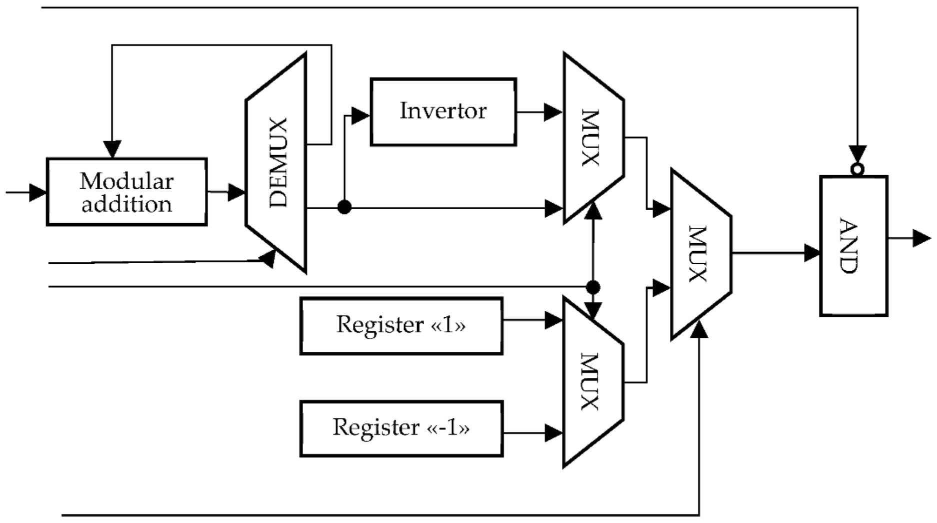

Figure 4. Quotient output block diagram.

4. Discussion

This section presents a description of the example for verification of obtained results, their interpretation, as well as the conclusions that can be drawn.

Let us consider an example for the RNS $\{p_1, p_2, p_3, p_4\} = \{11, 13, 17, 19\}$. According to Algorithm 1, the internal parameters $N = 17$, $w_1 = 16$, $w_2 = 8$, $w_3 = 5$, and $w_4 = 9$ are calculated. The constant multiplication blocks multiply each of the four remainders of the number by $C(B_1) = 83,408$, $C(B_2) = 100,824$, $C(B_3) = 84,811$, and $C(B_4) = 124,173$, respectively. Addition blocks add the obtained products and output the least significant N bits of the number. Thus, pairs of blocks of multiplication by constants with addition blocks implement the formula

$$C(X) = |83,408 \cdot x_1 + 100,824 \cdot x_2 + 84,811 \cdot x_3 + 124,173 \cdot x_4|_{2^{17}}.$$

The inverters find the opposite value for the number represented in the RNS by subtracting the corresponding remainder from the modulus.

$X = (5, 4, 3, 5)$ is fed to the input of the dividend, which, after calculating the core function by blocks of multiplication by constants and addition, feeds the value $C(X) = 122,770$ to the second input of the sign determination block, to the first input of which $X = (5, 4, 3, 5)$ is sent from the input of the dividend.

In the block for determining the sign, a comparison is made with $K = 23,094$ and $C(K) = 65,517$ according to Algorithm 2. The value of the sign of X is 1 (negative) and it is fed to the control input of the dividend multiplexer, and to the first input of the XOR element. Additionally, the value $X = (5, 4, 3, 5)$ is fed to the inverter, the result of which is $-X = (p_1 - x_1, p_2 - x_2, \ldots, p_n - x_n) = (6, 9, 14, 14)$.

After calculating the core function by the constant multiplication and addition blocks, $C(-X) = 8264$ is obtained. $C(X) = 122,770$ is fed to the first information input of the multiplexer. $X = (5, 4, 3, 5)$ is fed to the second information input. $C(-X) = 8264$ is fed to the third information input. $-X = (6, 9, 14, 14)$ is fed to the fourth information input. Since the dividend is negative, $C(|X|) = 8264$ is fed to the first output of the multiplexer of the dividend and $|X| = (6, 9, 14, 14)$ is fed to the second output.

At the same time, $Y = (1, 10, 6, 4)$ is fed to the input of the divider, which, after calculating the core function by the blocks of multiplication by constants and addition, feeds the value $C(Y) = 54$ to the second input of the sign determining block, the first input of which is $Y = (1, 10, 6, 4)$ from the input of the divider. In the block for determining the sign, a comparison is made with $K = 23,094$ and $C(K) = 65,517$ according to Algorithm 2.

The value of the sign of Y is 0 (positive), and it is fed to the control input of the divider multiplexer and to the second input of the XOR element. In addition, the value $Y = (1, 10, 6, 4)$ is fed to the inverter. The result is $-Y = (10, 3, 11, 15)$. After calculating the core function by the constant multiplication and addition blocks, $C(-Y) = 130,980$ is obtained. $C(Y) = 54$ is fed to the first information input of the multiplexer, $Y = (1, 10, 6, 4)$ is fed to the second information input, $C(-Y) = 130,980$ is fed to the third information input, and $-Y = (10, 3, 11, 15)$ is fed to the fourth information input. Since the divisor is positive, $C(|Y|) = 54$ is fed to the first output of the multiplexer, and $|Y| = (1, 10, 6, 4)$ is sent to the second output.

Since the signs of the dividend X and the divisor Y are different, signal 1 is sent to the output of the XOR element, i.e., the result is negative. The comparison block is based on Algorithm 2. $C(|X|) = 8264$ and $|X| = (6, 9, 14, 14)$ are compared with $C(|Y|) = 54$ and $|Y| = (1, 10, 6, 4)$ coming from the dividend and divisor multiplexers. Since $C(|X|) = 8264 > C(|Y|) = 54$, then $|X| > |Y|$, "$|X| < |Y|$", and "$|X| = |Y|$", 0 is sent.

Thus, the sign of result 1 is fed to the first output of the block for calculating positional characteristics. The zeros are fed to the second and third outputs, which means that the conditions "$|X| < |Y|$" and "$|X| = |Y|$" are not met. At the fourth, fifth, and sixth outputs, respectively, there are the values $|X| = (6, 9, 14, 14)$, $C(|X|) = 8264$ and $|Y| = (1, 10, 6, 4)$.

$|Y| = (1, 10, 6, 4)$ is fed to the input of the divider storage register. The degree counting unit delivers to the first output the address of the power "2^1" represented in the RNS, which is stored in the degree storage register "2".

The value (2,2,2,2) is fed to the second input of the modulo multiplication block, the first input of which is (1,10,6,4). The result (2,7,12,8), under the action of the signal at the control input of the demultiplexer, is fed to the multiplication and addition blocks. The value of the core function $C(2|Y|) = 116$ is calculated. In the degree counting unit, this value is compared with $C(|X|) = 8264$, which is fed to the first input. Since $116 < 8264$, the degree counting continues. This countdown continues until 2^7, for which $C\left(2^7|Y|\right) = 8331$. After that, the degree counting unit goes into the countdown state and sends a corresponding signal to the control input of the demultiplexer.

The degree counting unit delivers to the first output the address of the power "2^7" represented in the RNS, which is stored in the degree storage register "2". The value (7,11,9,14) is fed to the second input of the modulo block, the first input of which is (1,10,6,4).

The result (7,6,3,18), under the action of a signal at the control input of the demultiplexer, is fed to the second input of the subtractor modulo, to the first input of which $|X| = (6, 9, 14, 14)$ is fed.

The result $\Delta_1 = (10, 3, 11, 15)$ is fed to the blocks of multiplication by a constant and addition, in which the value of the core function $C(\Delta_1) = 130,980$ is calculated. It is fed to the first input of the block for determining the sign, to the second input of which the value Δ_1 is sent, since the number $C(\Delta_1) > C(K)$, then $\Delta_1 < 0$ and 2^7 is not included in the representation of the quotient Q.

"1" is sent to the output of the block for determining the sign, which arrives to the multiplexer control input, overwriting the value $|X|$ in the decrement storage register. Additionally, 1 from the output of the sign determination unit is fed to the inverted first input of the AND gate, zeroing the value of the power "2^7" supplied from the second output of the demultiplexer.

Further, the degree counting unit feeds to the first output the address of the degree "2^6", represented in the RNS, which is stored in the degree storage register "2". The value (9,12,13,7) is fed to the second input of the modulo block, the first input of which is (1,10,6,4). The result (9,3,10,9), under the action of a signal at the control input of the demultiplexer, is fed to the second input of the subtractor modulo, to the first input of which $|X| = (6, 9, 14, 14)$ is fed. The result $\Delta_2 = (8, 6, 4, 5)$ is fed to the blocks of multiplication by a constant and addition, in which the value of the core function $C(\Delta_2) < C(K)$ is calculated, which is fed to the first input of the block for determining the sign, to the second input of which comes the value of Δ_2, since the number $C(\Delta_2) < C(K)$, then $\Delta_2 > 0$ and 2^6 is included in the representation of the quotient Q.

At the output of the block for determining the sign, 0 is sent, which is fed to the control input of the multiplexer, recording the value of Δ_2 in the storage register to be reduced. In addition, 0 from the output of the sign determining block is fed to the inverted first input of the AND gate, passing the value of the power "2^6" supplied from the second output of the demultiplexer to the second output of the approximation series refinement block.

The rest of the degrees are checked in the same way. Finally, the degree counting unit sends to the first output the address of the "2^0" represented in the RNS, which is stored in the degree storage register "2". The value (1,1,1,1) is fed to the second input of the modulo block, the first input of which is (1,10,6,4). The result (1,10,6,4) under the action of the signal at the control input of the demultiplexer is fed to the second input of the subtractor modulo. The first input is sent with $\Delta_7 = (1, 10, 6, 4)$ from the output storage register decreasing.

The result $\Delta_8 = (0, 0, 0, 0)$ is fed to the blocks of multiplication by a constant and addition, where the value of the core function $C(\Delta_8) = 0$ is calculated, which is fed to the first input of the block for determining the sign. The second input receives the value Δ_8, since the number $C(\Delta_8) < C(K)$. Then $\Delta_8 \geq 0$ and 2^0 are included in the representation of the quotient Q.

"0" is sent to the output of the block for determining the sign, which is fed to control input of the multiplexer, writing the value of Δ_8 in the storage register to be reduced. In addition, "0" from the output of the sign determining block is fed to the inverted first input of the AND element, passing the value of the power "2^0" supplied from the second output of the demultiplexer to the second output of the approximation series refinement block. The end of the counting signal is sent to the third output of the power counting block.

The final quotient is formed in the block of the output of the quotient. The first input of the modulus addition block sequentially receives the degrees "2" included in the representation of the quotient Q. The second input of the modulus addition block receives the sum of the previously obtained degrees.

The number "2^7" after logical multiplication with zero is equal to (0,0,0,0), entering the first input, is added with (0,0,0,0). Then

$2^6 = (9, 12, 13, 7)$ is added with $(0, 0, 0, 0)$.

$2^5 = (10, 6, 15, 13)$ is added with $2^6 = (9, 12, 13, 7)$,

$2^4 = (5, 3, 16, 16)$ is added with $2^6 + 2^5 = (8, 5, 11, 1)$,

$2^3 = (8, 8, 8, 8)$ is added with $2^6 + 2^5 + 2^4 = (2, 8, 10, 17)$,

$2^2 = (4, 4, 4, 4)$ is added with $2^6 + 2^5 + 2^4 + 2^3 = (10, 3, 1, 6)$,

$2^1 = (2, 2, 2, 2)$ is added with $2^6 + 2^5 + 2^4 + 2^3 + 2^2 = (3, 7, 5, 10)$,

$2^0 = (1, 1, 1, 1)$ is added with $2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 = (5, 9, 7, 12)$

The result $(6, 10, 8, 13)$ under the effect of the signal of the end of the counting of the fourth input of the quotient output block to the control input of the demultiplexer is fed to the input of the inverter, where the opposite value $(5, 3, 9, 6)$ is located, arriving at the first input of the multiplexer quotient, the second input of which receives $(6, 10, 8, 13)$ from the second output of the demultiplexer.

Since the control inputs of the private multiplexer and the unit multiplexer receive a signal that the result is negative, the value $(5, 3, 9, 6)$ from the inverter and $(10, 12, 16, 18)$ from the storage register "-1" in the RNS. Under the action of the signal "$|X| = |Y|$" (in this case 0) from the third input of the quotient output unit to the control input of the quotient selection multiplexer, the value $(5, 3, 9, 6)$ is sent to the output from the inverter and since the signal "$|X| < |Y|$" (in this case, 0) from the second input of the quotient output block is inverted by the second input of the AND gate, then the quotient output is supplied with the value $(5, 3, 9, 6)$, which corresponds to the value -127.

5. Comparative Analysis

The proposed implementation of the Euclidean division algorithm reduces the size of the operands by half compared to the algorithm from [17] by using the Akushsky core function instead of the approximate method. On the other hand, using the Akushsky core function without critical cores allows reducing the depth compared to the algorithms from [18,24–26] (see Table 1). By depth, we mean a number of the RNS processor elements, circuit for arithmetic or Boolean operations, such as addition, multiplication, modulo, etc.

Table 1. Main properties of the studied Euclidean division algorithms, where a is the size of modulo RNS in bits.

Ref.	Dividend	Divisor	Size of Coefficients	Depth	RNS Processor Elements
[17]	$[0, P)$	$(0, P)$	$2a \cdot n$	$\log P$	n
[18]	$[0, P)$	$(0, P)$	a	$a + 2n + 4$	n
[24]	$[0, P)$	$(0, P)$	a	$\log n \log P$	n^2
				$\log n \log\log P + \log\log P$	$a \cdot n^2$
[25]	$[0, P)$	$(0, P)$	$a \cdot n + \log n$	$2a \cdot n \log n + 3a \log n$	n^2
[26]	$[0, P)$	$(0, P)$	$a \cdot n$	$2 \log P$	n
new	$[0, P)$	$(0, P)$	$a \cdot n$	$\log P$	n

Akushsky core function is a generalization of the Pirlo and Impedovo function. Both Akushsky's core function and approximate method possess similar arithmetic options and used for similar application areas, for instance, Euclidean division algorithms. However, Akushsky's core function avoids computational errors arising due to rounding operations.

6. Conclusions

We propose an enhanced modular division implementation. It has an improved accuracy and performance with minimal hardware requirements. The proposed method reduces the size of the operands by half in comparison with the RNS division algorithm based on the approximate method. The field programmable gate arrays (FPGAs) implementation can be used both as a separate device and as a coprocessor to perform non-modular operations.

We use Akushsky core function $C(X) = |\sum_{i=1}^{n} C(B_i) \cdot x_i|_{C_P}$ with no critical cores due to its monotonicity, which allows accurately comparing numbers and determining the sign of the number.

Our method improves the accuracy of calculating the division of numbers and determining the sign of the RNS numbers by avoiding rounding errors arising when the approximate method based on the Chinese remainder theorem is used.

Author Contributions: M.B., A.T. and V.K. contributed to the manuscript equally. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Education and Science of the Russian Federation (Project 075-15-2020-915).

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.

References

1. Akushsky, I.Y. *Machine Arithmetic in Residue Number System*; Akushsky, I.Y., Yuditsky, D.I., Eds.; Soviet Radio: Moscow, Russia, 1968; p. 440.
2. Bajard, J.-C.; Imbert, L. A full RNS implementation of RSA. *IEEE Trans. Comput.* **2004**, *53*, 769–774. [CrossRef]
3. Babenko, M.; Tchernykh, A.; Chervyakov, N.; Kuchukov, V.; Miranda-López, V.; Rivera-Rodriguez, R.; Du, Z.; Talbi, E.-G. Positional Characteristics for Efficient Number Comparison over the Homomorphic Encryption. *Program. Comput. Softw.* **2019**, *45*, 532–543. [CrossRef]
4. Asif, S.; Hossain, M.S.; Kong, Y.; Abdul, W. A Fully RNS based ECC Processor. *Integration* **2018**, *61*, 138–149. [CrossRef]
5. Tchernykh, A.; Babenko, M.; Chervyakov, N.; Miranda-López, V.; Kuchukov, V.; Cortés-Mendoza, J.M.; Deryabin, M.; Kucherov, N.; Radchenko, G.; Avetisyan, A. AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage. *Int. J. Approx. Reason.* **2018**, *102*, 60–73. [CrossRef]
6. 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]
7. Gomathisankaran, M.; Tyagi, A.; Namuduri, K. HORNS: A homomorphic encryption scheme for Cloud Computing using Residue Number System. In Proceedings of the 2011 45th Annual Conference on Information Sciences and Systems, Baltimore, MD, USA, 23–25 March 2011; pp. 1–5.
8. Di Claudio, E.D.; Piazza, F.; Orlandi, G. Fast combinatorial RNS processors for DSP applications. *IEEE Trans. Comput.* **1995**, *44*, 624–633. [CrossRef]
9. Jyothi, G.N.; Sanapala, K.; Vijayalakshmi, A. ASIC implementation of distributed arithmetic based FIR filter using RNS for high speed DSP systems. *Int. J. Speech Technol.* **2020**, *23*, 259–264. [CrossRef]
10. Chang, C.-H.; Molahosseini, A.S.; Zarandi, A.A.E.; Tay, T.F. Residue Number Systems: A New Paradigm to Datapath Optimization for Low-Power and High-Performance Digital Signal Processing Applications. *IEEE Circuits Syst. Mag.* **2015**, *15*, 26–44. [CrossRef]
11. Isupov, K. High-Performance Computation in Residue Number System Using Floating-Point Arithmetic. *Computation* **2021**, *9*, 9. [CrossRef]
12. Isupov, K.; Knyazkov, V.; Kuvaev, A. Design and implementation of multiple-precision BLAS Level 1 functions for graphics processing units. *J. Parallel Distrib. Comput.* **2020**, *140*, 25–36. [CrossRef]
13. Nakahara, H.; Sasao, T. A High-speed Low-power Deep Neural Network on an FPGA based on the Nested RNS: Applied to an Object Detector. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5.
14. Roohi, A.; Taheri, M.; Angizi, S.; Fan, D. RNSiM: Efficient Deep Neural Network Accelerator Using Residue Number Systems. In Proceedings of the 2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD), Munich, Germany, 1–4 November 2021; pp. 1–9.
15. Mohan, P.V.A. *Residue Number Systems*; Springer International Publishing: Cham, Switzerland, 2016; ISBN 978-3-319-41383-9. [CrossRef]
16. 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]
17. Chervyakov, N.; Lyakhov, .P.; Babenko, M.; Lavrinenko, I.; Deryabin, M.; Lavrinenko, A.; Nazarov, A.; Valueva, M.; Voznesensky, A.; Kaplun, D. A Division Algorithm in a Redundant Residue Number System Using Fractions. *Appl. Sci.* **2020**, *10*, 695. [CrossRef]
18. Chang, C.C.; Lai, Y.P. A division algorithm for residue numbers. *Appl. Math. Comput.* **2006**, *172*, 368–378. [CrossRef]
19. Chiang, J.S.; Lu, M. A general division algorithm for residue number systems. In Proceedings of the IEEE Symposium on Computer Arithmetic, Grenoble, France, 26–28 June 1991.
20. Chervyakov, N.I.; Babenko, M.G.; Kuchukov, V.A.; Deryabin, M.A.; Lavrinenko, I.N.; Lavrinenko, A.V. Russian Federation, IPC G06F7/72. Modular Number Division Device: No. 2016146626. Applicant and Patentee Federal State Autonomous Educational Institution of Higher Education "North Caucasus Federal University". Patent No. 2628179, 15 August 2017.
21. Burgess, N. Scaling an RNS number using the core function. In Proceedings of the 2003 16th IEEE Symposium on Computer Arithmetic, Santiago de Compostela, Spain, 15–18 June 2003; pp. 262–269. [CrossRef]
22. Pirlo, G.; Impedovo, D. A new class of monotone functions of the residue number system. *Int. J. Math. Models Methods Appl. Sci.* **2013**, *7*, 803–809.
23. Hiasat, A.A.; Abdel-Aty-Zohdy, H.S. Design and implementation of an RNS division algorithm. In Proceedings of the 13th IEEE Sympsoium on Computer Arithmetic, Asilomar, CA, USA, 6–9 July 1997; pp. 240–249. [CrossRef]
24. Hitz, M.A.; Kaltofen, E. Integer division in residue number systems. *IEEE Trans. Comput.* **1995**, *44*, 983–989. [CrossRef]
25. Bajard, J.-C.; Didier, L.-S.; Muller, J.-M. A new Euclidean division algorithm for residue number systems. In Proceedings of the International Conference on Application Specific Systems, Architectures and Processors: ASAP '96, Chicago, IL, USA, 19–21 August 1996; pp. 45–54. [CrossRef]
26. Lu, M.; Chiang, J.-S. A novel division algorithm for the residue number system. *IEEE Trans. Comput.* **1992**, *41*, 1026–1032. [CrossRef]