electronics

MDPI

Article

A High-Speed Division Algorithm for Modular Numbers Based on the Chinese Remainder Theorem with Fractions and Its Hardware Implementation

Nikolai Chervyakov, Pavel Lyakhov, Mikhail Babenko, Anton Nazarov, Maxim Deryabin *, Irina Lavrinenko and Anton Lavrinenko

Department of Applied Mathematics and Mathematical Modeling, North-Caucasus Federal University, Stavropol 355009, Russia; ncherviakov@ncfu.ru (N.C.); ljahov@mail.ru (P.L.); mgbabenko@ncfu.ru (M.B.); kapitoshking@mail.ru (A.N.); lavrinenko_ir1@mail.ru (I.L.); k-fmf-primath@stavsu.ru (A.L.)

* Correspondence: maderiabin@ncfu.ru; Tel.: +7-919-734-8307

Received: 29 January 2019; Accepted: 21 February 2019; Published: 27 February 2019

Abstract: In this paper, a new simplified iterative division algorithm for modular numbers that is optimized on the basis of the Chinese remainder theorem (CRT) with fractions is developed. It requires less computational resources than the CRT with integers and mixed radix number systems (MRNS). The main idea of the algorithm is (a) to transform the residual representation of the dividend and divisor into a weighted fixed-point code and (b) to find the higher power of 2 in the divisor written in a residue number system (RNS). This information is acquired using the CRT with fractions: higher power is defined by the number of zeros standing before the first significant digit. All intermediate calculations of the algorithm involve the operations of right shift and subtraction, which explains its good performance. Due to the abovementioned techniques, the algorithm has higher speed and consumes less computational resources, thereby being more appropriate for the multidigit division of modular numbers than the algorithms described earlier. The new algorithm suggested in this paper has $O(\log_2 Q)$ iterations, where Q is the quotient. For multidigit numbers, its modular division complexity is $Q(N)$, where N denotes the number of bits in a certain fraction required to restore the number by remainders. Since the number N is written in a weighed system, the subtraction-based comparison runs very fast. Hence, this algorithm might be the best currently available.

Keywords: division algorithm; residue number system; modular arithmetic

1. Introduction

The development of an informational society poses new challenges connected with the problem of multidigit numbers transmission and processing. Important mathematical problems that require such calculations and also considerable computational resources, both in terms of theory and practice, arise in the applied and computational theory of numbers [1,2]. Most of such problems involve integer calculations with the numbers belonging to the large and super large computer ranges, while the results of calculations must be precise without rounding.

A feature of conventional computing devices is limited bit grid, which causes computational complexity for the operations over multidigit numbers.

For calculations with multidigit numbers or calculations with large-range numbers, the residue number systems (RNSs) have clear advantages over the radix number systems. Modern research is focused on the processing of multidigit data in which the values of integer variables considerably exceed the dynamic range of the serially produced computing devices (by 10^3–10^6 times and even more); see Molahosseini et al. [2].

Modular calculations are crucial for applications with multidigit numbers, e.g., cryptography, digital signature, and others [3,4].

For instance, the numbers used in cryptographic systems vary between 2^{600} and 2^{700} to guarantee the high-level security of protected information [5]. During modular processing these numbers are partitioned into small formats (several bits or several tens of bits), which appreciably speeds up their implementation.

Residue number systems attract many researchers as a base for computing devices, and the recent decade has been remarkable for the growing interest in these systems. This fact follows from numerous publications dedicated to RNS usage in digital signal processing, image processing, antinoise coding, cryptographic systems, quantum automata, neurocomputers, systems with the massive parallelism of operations, cloud computing, DNA computing, etc. [1–13].

Residue number systems are remarkable for fast summation, subtraction, and multiplication, which explains major interest in these systems for the applications requiring large volumes of calculations. However, some operations (modulo reduction, comparison and division of numbers) are very complicated in RNSs [14,15]. The development of more efficient algorithms for comparison, sign detection, and division would open up new applications of RNSs [16].

The existing division algorithms for RNSs [17–19] can be classified as the ones based on number comparison and the ones based on number subtraction.

In the comparison-based algorithms [18–24], the quotient is found using the iteration:

$$a' = a - 2^i q_i b$$

where a' and a denote the current and next dividends, respectively; b is the divisor; finally, q_i gives the digit of the quotient. For obtaining q_i , the quotient a has to be compared with $2^i b$.

In the second class of the algorithms [21,25–27], the quotient is calculated using the iterations $a' = a - Q_i b$. The quotient Q_i is generated at each iteration from the complete RNS range, instead of being chosen from a small set.

The integer division algorithm [1] is similar to standard binary division. Yet, the main drawback of this algorithm and its modifications is that each iteration requires numbers comparison.

The algorithm without such drawbacks that was suggested in Szabó and Tanaka [1] replaces the real divisor with the approximate one (an RNS module or the product of several RNS moduli). The algorithm yields correct results under the condition $b \leq \bar{b} < 2b$, where b and $\bar{b}$ are the real and approximate divisors, respectively. Clearly, this condition may be violated for some sets of moduli (e.g., $p_1 = 9$, $p_2 = 11$, $b = 4$, where p_1 and p_2 are the RNS moduli).

Among the shortcomings of the above algorithm, note the need for using mixed radix number systems (MRNSs), scale operation, special logic and approximate divisor calculation tables. As a matter of fact, a series of approaches were proposed for solving the division problem based on the numbers comparison and sign detection methods, which can be classified in the following way: the algorithms [18,19,23] employ the conversions of MRNSs; the algorithms [18,22] formulate the problem in terms of even numbers detection (parity check); and the algorithms [20] involve the base extension method for iterations. However, the proposed algorithms suffer from high execution time and considerable hardware cost due to the usage of MRNS, the Chinese remainder theorem (CRT), and other time-consuming operations.

In Hiasat et al. [27] and Hiasat [24] it introduced a high-speed division algorithm in which the MRNS and CRT in modular numbers division were replaced by the comparison of the high powers of the dividend and divisor. The execution time and hardware cost of this algorithm are smaller in comparison with the other algorithms; nevertheless, it contains redundant stages. In order to speed up current quotient calculation, J.H. Yang [28] suggested a division algorithm based on parity check that finds the quotient twice as fast as the algorithms [24,27]. But the calculations of the high powers of 2 still require much time in RNSs, and these operations are performed at each iteration. In Chung [29] the original algorithm [27] was simplified using division by 2 and efficient quotient search within

a probable range. At each round, this algorithm adopts hard-to-implement parity check, which is its disadvantage.

Most of the listed algorithms contain hard-to-implement operations such as the CRT, scale operation, extension, sign detection, and comparison, which reduce their speed and cause considerable hardware cost of modular numbers division.

There exists a division algorithm in the RNS format that uses the basic set of RNS moduli in combination with an auxiliary module system for storing the remainders of the dividend and divisor. The dividend and divisor in the RNS representation are converted into different RNS representations with different module systems [29]. The usage of two RNS sets leads to higher redundancy, making it necessary to perform direct and inverse transitions from the basic module set to the auxiliary one and back during division. This feature dramatically reduces the speed of calculations. Talahmeh at al. [30] introduced a fast division algorithm based on index transformations over the Galois field $GF(p)$, which is easily implemented using tabular search. However, this algorithm guarantees efficient processing for the data of 6–10 bits and prime moduli only. In the case of larger ranges, the algorithm has low performance because the generator of prime p must be very large to represent integers over the Galois field.

Most of the suggested iterative algorithms involve very many operations at each iteration. As was declared by the authors, the algorithm based on the CRT with fractions [22,26,31] might appear to be best because its time complexity is $O(n\,b)$, where n denotes the number of RNS moduli and b the number of bits in each module under the assumption that the moduli are more or less the same. But this algorithm has some disadvantages as follows: (a) the divisor D is limited by $\frac{P}{16} < D \leq \frac{3P}{16}$, where P is an RNS representation range; (b) each iteration includes several operations such as summation, multiplication, comparison, and parity check; (c) in the end of the algorithm, the quotient has to be converted from the system $\{-1, 0, 1\}$ to the system $\{0, 1\}$, which gives extra computation load during modular numbers division.

In this paper, an alternative modular division algorithm with efficient quotient calculation using the relative values of the dividend and divisor in the fractional representation is introduced. Each iteration of the new algorithm involves the shift and subtraction of the successive intermediate results, which makes hardware implementation more efficient.

2. Approximate Positional Characteristic Calculation for Modular Numbers Based on the CRT with Fractions and Its Application to Modular Division

An RNS is a set of positive and pairwise relatively prime numbers $\{p_1, p_2, \ldots, p_n\}$, which are called moduli or bases. The dynamic range is given by $P = p_1\, p_2 \ldots p_n$. For an unsigned number kh, the additional RNS range has the form $0 \leq kh \leq R - 1$. For the signed numbers, the additional range is defined by $-\left[\frac{R}{2}\right] < x \leq \left[\frac{R-1}{2}\right]$. In this system, any integer a belonging to the range $[0, P-1]$ can be uniquely described by an ordered set of remainders $(\alpha_1, \alpha_2, \ldots, \alpha_n)$. Each remainder α_i has the modulo p_i representation

$$\alpha_i = a \bmod p_i = |a|_{p_i},\ 0 \leq \alpha_i < p_i, i = 1, 2, \ldots, n. \tag{1}$$

Let operation $*$ be arithmetic summation, subtraction or multiplication. The most interesting property of an RNS is that these operations can be converted from the integer representation into the modular operations with different moduli p_i, i.e.,

$$Z = a * b \overset{\text{RNS}}{\rightarrow} \left\{ \begin{array}{c} |z|_{p_1} = |\alpha_1 * \beta_1|_{p_1} \\ |z|_{p_2} = |\alpha_2 * \beta_2|_{p_2} \\ \ldots \\ |z|_{p_n} = |\alpha_n * \beta_n|_{p_n} \end{array} \right\}. \tag{2}$$

Using model Equation (2), the dynamic range is decomposed into parts with a narrower data format so that within them all calculations are performed in parallel. As a result, the complexity of the arithmetic structures decreases accordingly.

The number a in the RNS representation can be restored using the Chinese Remainder Theorem [1,2], i.e.,

$$a = \left| \sum_{i=1}^{n} \frac{P}{p_i} \left| P_i^{-1} \right|_{p_i} \alpha_i \right|_P, \tag{3}$$

where $P = \prod_{i=1}^{n} p_i$; p_i, $i = 1, 2, \ldots, n$, denote the RNS moduli; $\left| P_i^{-1} \right|_{p_i}$ is the multiplicative inversion of P_i on p_i, $P_i = \frac{P}{p_i} = p_1 p_2 \ldots p_{i-1} p_{i+1} \ldots p_n$.

As is well-known, among their drawbacks the RNSs have the implementation complexity of non-modular operations (comparison, division) that are based on given positional characteristics.

The analysis of positional characteristics shows that they can be calculated precisely or approximately. Therefore, the calculation methods of positional characteristics consist of two groups as follows:

- precise calculation methods;
- approximate calculation methods.

The precise calculation methods of positional characteristics were considered in [1,2]. In this paper, an approximate calculation method with appreciably smaller hardware cost and execution time for the operations over the positional codes of decreased digit capacity will be employed.

The approximate calculation method of positional characteristics uses the relative values of modular numbers with respect to the complete range defined by the Chinese Remainder Theorem [2]. This theorem associates with a positional number a its remainder representation $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, where α_i, $i = 1, 2, \ldots, n$, are the least non-negative residues of this number on the RNS moduli $p_1, p_2, \ldots, p_n$.

Assume that the number a has the RNS representation with residues $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Dividing the left- and right-hand sides of Equation (3) by the constant P (the dynamic range) yields the approximate value

$$\left| \frac{a}{P} \right|_1 = \left| \sum_{i=1}^{n} \frac{\left| P_i^{-1} \right|_{p_i}}{p_i} \alpha_i \right|_1 \approx \left| \sum_{i=1}^{n} k_i \alpha_i \right|_1. \tag{4}$$

Here $k_i = \frac{\left| P_i^{-1} \right|_{p_i}}{p_i}$ are the constants of the chosen system; α_i, $i = 1, 2, \ldots, n$, denote the digits of the number in the RNS representation on the moduli p_i. Note that Equation (4) takes values within the interval $[0, 1)$. The final value is obtained by summation and integer part truncation, with the fractional part being retained. The fractional value $F(a) = \left| \frac{a}{P} \right|_1 \in [0, 1)$ contains information about the value of the number and its sign. If $\left| \frac{a}{P} \right|_1 \in \left[0, \frac{1}{2}\right)$, then the number a is positive, and $F(a)$ gives its value divided by P. Otherwise, a is a negative number, and $1 - F(a)$ gives its relative value. Denote by $[F(a)]_{2^{-t}}$ the value of $F(a)$ rounded to 2^{-t} bits. The exact value of $F(a)$ satisfies the inequalities $[F(a)]_{2^{-t}} < F(a) < [F(a)]_{2^{-t}} + 2^{-t}$. The integer part of the number yielded by summing up k_i is neglected, i.e., discarded.

The rounding of $F(a)$ causes inevitable errors. Introduce the notation $\rho = -n + \sum_{i=1}^{n} p_i$. As was demonstrated in [32], $N = \lceil \log_2(P\rho) \rceil$ bits after the decimal point have to be used for rounding $F(a)$ without considerable errors that would affect calculation accuracy. In other words, there exists a bijection between the set of numbers in the RNS representation and the set of numbers rounded to the Nth bit, i.e., $[F(a)]_{2^{-N}}$.

Taking into account the function $[F(a)]_{2^{-N}}$, the sign detection conditions can be written as follows:

$$\begin{array}{l} 1.\ \text{If } a = [F(a)]_{2^{-N}} < \dfrac{1}{2},\ \text{then the number } a \text{ is positive.} \\ 2.\ \text{If } \dfrac{1}{2} \le [F(a)]_{2^{-N}} < 1,\ \text{then the number } a \text{ is negative.} \end{array} \tag{5}$$

Consider the approximate calculation method for comparing numbers in the RNS representation.

Example. Let $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, and $p_4 = 7$ be the system of RNS moduli. Then $P = 2 \cdot 3 \cdot 5 \cdot 7 = 210$, $\rho = 2 + 3 + 5 + 7 - 4 = 13$, $P_1 = \frac{P}{p_1} = 105$, $P_2 = \frac{P}{p_2} = 70$, $P_3 = \frac{P}{p_3} = 42$, and $P_4 = \frac{P}{p_4} = 30$.

The constants k_i for calculating the relative values are

$$k_1 = \frac{\left|\frac{1}{105}\right|_2}{2} = \frac{1}{2},\ k_2 = \frac{\left|\frac{1}{70}\right|_3}{3} = \frac{1}{3},\ k_3 = \frac{\left|\frac{1}{42}\right|_5}{5} = \frac{3}{5},\ k_4 = \frac{\left|\frac{1}{30}\right|_7}{7} = \frac{4}{7}$$

The constants k_i rounded to 12 decimal places are

$$k_1 = 0.100000000000,\ k_2 = 0.010101010101,$$
$$k_3 = 0.100110011001,\ k_4 = 0.100100100100$$

Compare the two numbers $a = 97$ and $b = 96$ in the RNS representation with the moduli p_1, p_2, p_3, and p_4. Note that, for numbers comparison, the critical case is the numbers differing by 1. The RNS representations of the numbers a and b are $a = (1, 1, 2, 6)$ and $b = (0, 0, 1, 5)$, respectively. Their difference is $a - b = (1, 1, 2, 6) - (0, 0, 1, 5) = (1, 1, 1, 1)$. Now, detect the sign of $a - b$. First, find $[F(a - b)]_{2^{-12}} = 0.000000010010$; this value satisfies the first condition of model Equation (5), i.e., $0 < 0.000000010010 < 0.1$. Hence, the natural conclusion is that $a - b > 0$, meaning $a > b$.

3. New Division Algorithm Based on the CRT with Fractions

Consider two numbers—-a dividend $a = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and a divisor $b = (\beta_1, \beta_2, \ldots, \beta_n)$. Let both numbers be represented within the range $[0, P)$, where $P = p_1 \cdot p_2 \cdot \ldots \cdot p_n$ and p_i, $i = 1, 2, \ldots, n$, denote the RNS moduli. For the sake of simplicity, assume that a and b are positive numbers. (The case of negative numbers can be considered by analogy using simple modifications.) The division algorithm calculates the quotient Q and the remainder R so that $a = Q \cdot b + R$, where $0 \le R < b$. The detailed description of this algorithm is given in Appendix A.

Modular division includes two stages. At the first stage, the high power of 2^j is obtained using the binary series approximation of the quotient; at the second stage, the approximation series is refined accordingly. The algorithm yields $Q = (Q_1, Q_2, \ldots, Q_n)$, where $Q_i = \left|\sum_{j \in L} q_j 2^j\right|_{p_i}$; L denotes the set of powers j in the refined quotient approximation series; p_i, $i = 1, 2, \ldots, n$, are the RNS moduli.

For modular division optimization, the classical CRT will be replaced by the CRT with fractions. In this case, following model Equation (4), the dividend, divisor, and remainder can be written as the fractions

$$F(a) = \left|\frac{a}{P}\right|_1,\ F(b) = \left|\frac{b}{P}\right|_1,\ F(R) = \left|\frac{R}{P}\right|_1.$$

Using the approximate values, consider the modular division algorithm based on the CRT with fractions.

The algorithm consists of two stages. The first stage is to find the high power j of the divisor by the left shift of $F(b)$ to the zero digits standing before the first significant digit. At the second stage, the general quotient is generated by selecting the powers of 2 that form the partial quotients to be included in the general quotient approximation series. The analysis procedure starts from the highest

power j and ends with the zeroth power of 2, thereby reading out the necessary powers of 2 in the RNS representation. The modular division algorithm based on the CRT with fractions works in the following way. The residues table for the $0(\log_2 P - 1)$ integer powers of 2 is put into memory, and the representations $a = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and $b = (\beta_1, \beta_2, \ldots, \beta_n)$, $(\alpha_i, \beta_i) \bmod p_i$, $i = 1, 2, \ldots, n$, are obtained at the input. Then the quotient is calculated so that $F(a) = (Q \bmod p_1, Q \bmod p_2, \ldots, Q \bmod p_n) F(R) + F(b)$, where $0 \leq F(R) < F(b)$.

The quotient $Q = \left\lfloor \frac{a}{b} \right\rfloor$ is generated at each iteration from the powers of 2 in the RNS representation that are included or excluded depending on the sign of the subtraction chain $\Delta_i - \Delta_{i-1} - \Delta_{i-2} - \ldots - \Delta_m$, $\Delta_i = \Delta_{i-1} - 2^{j-i} q_{j-i} F(b)$. The notations in this formula are the following: j as the highest power of the quotient; q_j as the highest digit of the quotient; i as current iteration; $\Delta_1 = F(a) - F(b)\, q_j\, 2^j \approx [F(a)]_{2^{-t}} - [F(b)]_{2^{-t}} q_j\, 2^j$; $1 \leq i, j < \lfloor \log_2 P \rfloor$; $F(a) = \left| \frac{a}{P} \right|_1$ and $F(b) = \left| \frac{b}{P} \right|_1$ as the fractions; $P = \prod_{i=1}^{n} p_i$ as the complete range; p_i, $i = 1, 2, \ldots, n$, as the RNS moduli; $[F(a, b)]_{2^{-t}} < F(a, b) < [F(a, b)]_{2^{-t}} + 2^{-t}$; $[F(a)]_{2^{-t}}$ and $[F(b)]_{2^{-t}}$ as the values of $F(a)$ and $F(b)$, respectively, rounded to the 2^{-t}th bit (note that the resulting errors do not affect calculation accuracy); Δ_i as the current value and Δ_{i-1} as the successive value, which is defined by the one-position right shift of the divisor multiplied by the corresponding power of 2 (actually, this is equivalent to division by 2 and the subtraction $\Delta_i - \Delta_{i-1}$). Each ith iteration is associated with the ith binary digit in the RNS representation, which are put into memory as the residuals table for the integer powers of 2.

The well-known algorithm needs the dividend and divisor at each iteration. For the new algorithm, the dividend $F(a)$ and the divisor $F(b)$ are required only at the first iteration; all subsequent iterations involve the difference $\Delta_i - \Delta_{i-1}$ because these values contain information about the dividend and divisor. With this method, all quotient approximation iterations are reduced to the subtraction $\Delta_i - \Delta_{i-1}$, and the sign of this difference is used to find the desired partial quotient as the corresponding power of 2 in the RNS representation. As a result, the computational complexity of modular division considerably decreases.

The digits of the quotient are obtained by the $\bmod p_i$ summation of the partial quotients using the sign of the subtraction result. If the sign is positive, the quotient is included (otherwise excluded). In contrast to the well-known algorithms, the new one allows for easy implementation: the time complexity of the iterations is defined by the execution time of shift, subtraction, and summation.

Concerning the advantages of the new algorithm, note that the division procedure does not involve (a) intermediate numeric data in the MRNS representation and (b) difficult-to-implement RNS operations such as comparison, scaling, base extension, and sign detection. These features contribute to higher efficiency of modular division.

The new modular division algorithm for integer numbers $\left[\frac{a}{b} \right]$ has the scheme presented in Appendix B. The hardware implementation of this algorithm is described in detail below.

4. Hardware Implementation of New Modular Division Algorithm

The new modular division algorithm for multidigit numbers includes the following basic parts: quotient sign detection, quotient approximation, and further refinement of the quotient approximation series.

Figure 1 illustrates the hardware implementation of the modular division algorithm, which consists of several units such as converters, summers (summators), multipliers and others. The issues of their optimization were studied in the papers [33–36].

Assume that an RNS contains n moduli and n modular processors. Let m be the number of bits required for the representation of each remainder. For making the hardware implementation complexity analysis of this algorithm simpler, consider the case in which the moduli are more or less the same. Under this hypothesis, the total length of the modular processor bus is $M = n \cdot m$ bits.

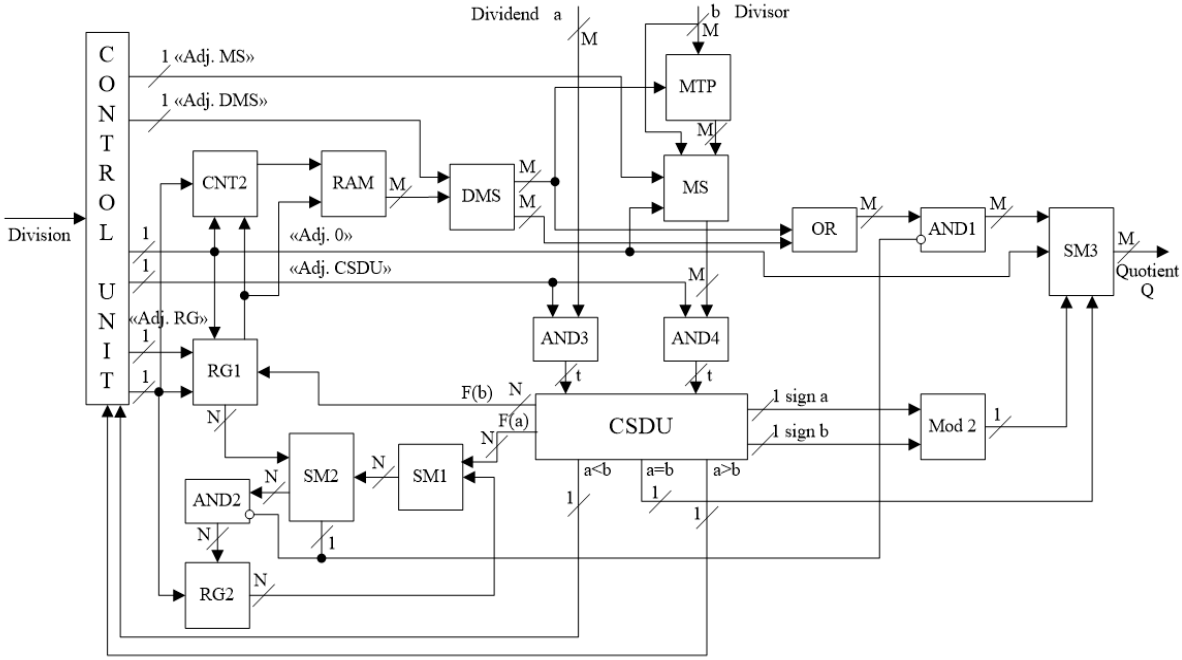

Figure 1. Hardware implementation of new algorithm.

Let the dividend and divisor be arbitrary integers and also let the divisor be not reducible to a pairwise relatively prime number on the RNS moduli.

The hardware implementation scheme in Figure 1 includes buses a and b supplying the dividend and divisor, respectively, and quotient bus Q. Each of buses a , b, and Q has M bits. For division, the one-bit signal is supplied through «Division» bus. Upon receipt of the inputs a and b, the system calculates Q so that $a = Q \cdot b + R$, where $0 \leq R < b$.

At the initial state, the control unit (CU) receives the "Division" signal and then forms the following signals through the one-bit buses (see Figure 1):

- "Adj. 0," for adjusting the zero states of the functional units;
- timing pulses («TP»), for performing control of the registers and counters;
- "Adj. MS," for adjusting the address code of multiplexer («MS») 2:1;
- "Adj. DMS," for adjusting the address code of demultiplexer («DMS») 1:2. Depending on the address code, demultiplexer «DMS»switches the highest power of 2, either simultaneously to multiplier «MTP» and element «OR» or to element «OR» only, whose output is connected to the input of inhibit element I_1. Depending on the address code, multiplexer MS switches to the output of the comparison and sign detection unit (CSDU), or directly to the divisor b, or to the divisor b multiplied by the highest power of 2^k;
- "Adj. RG_1," for adjusting the right shift of reversible register RG_1;
- "Adj. CSDU," for adjusting the blocking of the comparison and sign detection unit (CSDU).

The dividend a and the divisor b in the RNS representation on the chosen moduli are supplied through the M-bit buses to the input of CSDU. And this unit calculates the relative values of the dividend and divisor ($[F(a)]_{2^{-N}}$ and $[F(b)]_{2^{-N}}$) as well as detects their signs and performs their comparison. Through element I_3 the dividend directly comes to the input of CSDU. And the divisor is supplied to the input of CSDU through multiplexer MS, which has the corresponding address at the address input. The CSDU is implemented by Equations (4) and (5) and the model of Example 1. If $[F(a)]_{2^{-N}} = \left|\frac{a}{P}\right|_1 < [F(b)]_{2^{-N}} = \left|\frac{b}{P}\right|_1$, then CSDU forms the signal $b > a$ (the divisor is greater than the dividend), which comes to the input of the control unit. Next, the division unit is adjusted to the initial state, and the quotient $Q = 0$. If $[F(a)]_{2^{-N}} = \left|\frac{a}{P}\right|_1 = [F(b)]_{2^{-N}} = \left|\frac{b}{P}\right|_1$, then CSDU generates the

equality signal of the dividend and divisor ("$a = b$"). This signal is supplied to the input of summer SM_3, where the constant $1 = (1, 1, 1, 1)_{RNS}$ is written. If $[F(a)]_{2^{-N}} = \left|\frac{a}{P}\right|_1 > [F(b)]_{2^{-N}} = \left|\frac{b}{P}\right|_1$, then CSDU forms the signal "$a > b$," and the control unit switches the division unit into the partial quotient approximation mode.

Next, the signs of the dividend a and divisor b are analyzed. The sign of the quotient Q is defined by $sign\, Q = sign\, a \oplus sign\, b = sign\, a\, \overline{sign\, b} + \overline{sign\, a}\, sign\, b$. From the output of CSDU the sign signals of the dividend and divisor ("$sign\ a$" and "$sign\ b$") are supplied through the one-bit buses to the input of element exclusive or «XOR» (addition modulo 2). Using the truth table, this circuit generates the signal "$a = 0$" or "$a = 1$," which is supplied to the input of quotient summer SM_3. If the output signal of element XOR is 0, then the quotient is assigned the positive sign (0); if it is 1, then the quotient is assigned the negative sign (1). Then, through the N-bit buses the values $[F(a)]_{2^{-N}}$ and $[F(b)]_{2^{-N}}$ come to the input of summer SM_1 and the shift register RG_1, respectively.

The value $F(a)$ is converted by SM_1 into the additional code, and the result is supplied to the input of subtractor SM_2 through the N-bit bus. The value $[F(b)]_{2^{-N}}$ comes to the input of reversible shift register RG_1 through the N-bit bus. Using the timing pulses of the control unit, the content of register RG_1 (the value $[F(b)]_{2^{-N}}$) is shifted to the right to the number of zero digits standing before the first significant digit; this number is registered by counter 2 «CNT2». The resulting number of shifts corresponds to the highest power j of 2 in the divisor. The relative values are used to find the highest power of 2 in the quotient approximation series, which is registered by counter CNT2 without iterative calculations. As soon as the high significant digit of the divisor $[F(b)]_{2^{-N}}$ becomes 1 during the shift procedure (similar to number normalization), register RG_1 fixes the state of counter CNT2 through the one-bit bus and enables information read-out from RAM (similarly to stack pointer). Counter CNT2 activates the RAM memory address defining the highest power of 2^j in the quotient approximation series that must be in the quotient $Q = \left[\frac{a}{b}\right]$. In the initial state, buffer register RG_2 has zero value. The quotient approximation mode is completed, and at the RAM output the memory cell is activated that stores the highest power of 2^j in the quotient approximation series. Thus, for approximating the quotient b, the CSDU performs one comparison and one operation $[\log n]$ for obtaining the sum of N-bit numbers, while register RG_1 performs one shift operation to j digits.

Let $[F(b)]_{2^{-t}} = 0.000010100001$; in this case, after shifting the content of register RG_1 the state of counter CNT2 is 0011. This counter activates the memory cell containing the power of $2^3 \overset{RNS}{\rightarrow} \left(\left|2^3\right|_{p_1},\ \left|2^3\right|_{p_2},\ \ldots,\ \left|2^3\right|_{p_n}\right)$.

4.1. Refinement Mode for Quotient Approximation Series with Dividend a and Divisor b

Recall that the highest power of 2^j is obtained in the approximation mode and stored by RAM in the RNS representation. Using the one-bit signal "Adj. DMS," this power comes through the M-bit memory bus to the input of summer SM_3 via elements OR and I_1. After that, through the one-bit bus "Adj. MS" the address code of multiplexer MS is adjusted to the next state; the value $b\,2^j$ is switched at the output of multiplexer MS and then supplied to the input of CSDU through the M-bit bus. This unit calculates $\left[F\left(b\,2^j\right)\right]_{2^{-N}}$ and sends the result to the input of shift register RG_1 through the N-bit bus $F(b)$. The old information in RG_1 is removed. The buses of the dividend a and divisor b are disconnected from CSDU by the signal "Adj. CSDU" coming to the inputs of I_3 and I_4. Using the control signals of the control unit, the content of RG_1 is supplied to an input of SM_2 through the N-bit bus. From the output of summer SM_1 the value $F(a)$ is supplied to the second input of summer SM_2 through the N-bit bus. Next, the signal "Adj. RG_1" switches register RG_1 into the right shift mode.

The divisor multiplied by the highest power of 2 is subtracted from the content of summer SM_2, and the value $[F(a)]_{2^{-N}} - \left[F\left(b\,2^k\right)\right]_{2^{-N}}$ is calculated for detecting the sign of the result. If the sign digit of the subtraction result in summer SM_2 is 0, i.e., $[F\,(a)]_{2^{-N}} > \left[F\left(b\ 2^k\right)\right]_{2^{-N}}$, then 0s are supplied to the inhibit inputs of inhibit elements I_1 and I_2. Through elements OR the inhibit elements I_1 and I_2

pass the highest power of the quotient from the second output of DMS to the input of summer SM_3. The subtraction result in summer SM_2 (i.e., the remainder of the dividend that corresponds to the rest powers of 2) is supplied, through the N-bit bus and inhibit element I_2, first to the input of buffer register RG_2 and then to the input of summer SM_1. The old content of this summer is removed, and the new value is written. Next, the content of register RG_1 is shifted to the right, and the process is repeated as described above. If the sign digit of the subtraction result in summer SM_2 is 1 (i.e., the relative value of the divisor is greater than the dividend), then this 1 comes to the inhibit inputs of elements I_1 and I_2 that inhibit the supply of the corresponding power to summer SM_3 and also the supply of the subtraction result of summer SM_2 to the input of buffer register RG_2. In other words, the register saves the previous value of the subtraction result. At its input summer SM_3 receives only the refined powers of 2 that are the terms of the quotient series. The conversion process ends after analyzing the power of 2^0. Thus, in the refinement mode all redundant terms of the quotient approximation series are iteratively eliminated using uncomplicated transformations.

The approximate quotient is sequentially refined by the subtraction of the divisor first multiplied by the highest power of 2 from the dividend and its further shift and subtraction from the resulting partial remainders during quotient calculation. In comparison with the well-known algorithms, a distinctive feature of the quotient approximation refinement procedure suggested in the new algorithm is the usage of the dividend a and the product $b\,2^j$ (the divisor and the highest power of 2) only at the first iteration. The subsequent iterations are based on an original principle of this paper, namely, on the one-position right shift of $b\,2^j$ at each iteration; in fact, this is equivalent to division by 2 and the subtraction of the results obtained at successive iterations, i.e., $\Delta_j - \Delta_{j-1}$. The principle is unique because each iteration contains two operations—-one-position right shift and subtraction with further sign detection. As a result, the speed of division considerably increases. Each iteration of the well-known algorithms involves multiplication, summation, comparison, and parity check. Assume that each iteration requires t time units; then the total time of each iteration is $4\,t$. Each iteration of the new algorithm consists of one subtraction and one shift, consuming $2\,t$ time units. Thus, the efficiency gain is about 2. The performance analysis of the new algorithm (quotient approximation and refinement) has demonstrated its considerable advantage over the well-known counterparts in terms of modular division time.

4.2. Experimental Performance Analysis: New Modular Division Algorithm Versus Well-Known Algorithms

As is indicated by experiments, the algorithm developed in this paper strongly depends on initial data (the number of RNS moduli and their values) and also on input data (the values of dividend and divisors). Hence, this algorithm is difficult for analytical study.

The new modular division algorithm is similar to the algorithm presented in Hung [25]. However, in comparison with the optimization methods used therein, the operations of multiplication and summation have been replaced by the less demanding shift operations of partial quotients based on comparison. As a result, hardware cost and execution time have been considerably reduced. Besides, the Lu–Chiang algorithm involves the quotient correction operator with sign detection using MRNS, which dramatically decreases its speed.

In this section, the algorithm [25] will be compared with the new modular division algorithm on the same numerical data. For comparison, choose the example considered in Hung [25].

Let the RNS moduli be 5, 7, 9, and 11.

For example, take the dividend $a = 125 = (0, 6, 8, 4)$ and the divisor $b = 14 = (4, 0, 5, 3)$. It is required to obtain the quotient $a = 8 = (3, 1, 8, 8)$ and the remainder $r = 13 = (3, 6, 4, 2)$.

For the sake of illustrative and complete analysis, Tables 1 and 2 provide intermediate data yielded by the algorithm [25] and the new algorithm, respectively, in the course of calculating $\frac{a}{b} = \frac{125}{14}$.

The new modular division algorithm is attractive owing to less operations (see Table 3). Table 3 shows how many operations of different types are consumed by the well-known algorithms and the new algorithm for the example [25] (these data were obtained by experimental study).

Table 1. Calculation of $\frac{125}{14}$ by algorithm [25].

Iteration	Variable	Value	RNS Moduli {5,7,9,11}	Notes
$j=0$	D	14	$(4,0,5,3)$	$P=3465$
	$[P/8]$	433	$(3,6,1,4)$	
	$[P/8]-2D$	405	$(0,6,0,9)$	$EF_\alpha=6/64,\ ES_\alpha=+$
$j=1$	$D=2D$	28	$(3,0,1,6)$	
	$[P/8]-2D$	377	$(2,6,8,3)$	$EF_\alpha=5/64,\ ES_\alpha=+$
$j=2$	$D=2D$	56	$(1,0,2,1)$	
	$[P/8]-2D$	321	$(1,6,6,2)$	$EF_\alpha=4/64,\ ES_\alpha=+$
$j=3$	$D=2D$	112	$(2,0,4,2)$	
	$[P/8]-2D$	209	$(4,6,2,0)$	$EF_\alpha=2/64,\ ES_\alpha=+$
$j=4$	$D=2D$	224	$(4,0,8,4)$	
	$[P/8]-2D$	-15	$(0,6,3,7)$	$EF_\alpha=62/64,\ ES_\alpha=\pm$
$j=5$	$D=2D$	448	$(3,0,7,8)$	
	$[P/8]-2D$	-463	$(2,6,5,10)$	$EF_\alpha=54/64,\ ES_\alpha=-$
Line 3	A	125	$(0,6,8,4)$	
	$A-D$	-323	$(2,6,1,7)$	$EF_\alpha=56/64, ES_\alpha=-$
$i=1$	A	125	$(0,6,8,4)$	$EF_\alpha=1/64,\ ES_\alpha=+$
	$A=2(A-D)$	-646	$(4,5,2,3)$	$EF_\alpha=50/64,\ ES_\alpha=-$
	$Q=2(Q+1)$	2	$(2,2,2,2)$	
$i=2$	$A=2(A+D)$	-396	$(4,3,0,0)$	$EF_\alpha=56/64,\ ES_\alpha=-$
	$Q=2(Q-1)$	2	$(2,2,2,2)$	
$i=3$	$A=2(A+D)$	104	$(4,6,5,5)$	$EF_\alpha=0/64,\ ES_\alpha=+$
	$Q=2(Q-1)$	2	$(2,2,2,2)$	
$i=4$	$A=2(A-D)$	-688	$(2,5,5,5)$	$EF_\alpha=50/64,\ ES_\alpha=-$
	$Q=2(Q+1)$	6	$(1,6,6,6)$	
$i=5$	$A=2(A+D)$	-480	$(0,3,6,4)$	$EF_\alpha=54/64,\ ES_\alpha=-$
	$Q=2(Q-1)$	10	$(0,3,1,10)$	
	$A=A+D$	32	$(3,3,4,1)$	$EF_\alpha=61/64,\ ES_\alpha=\pm,\ S=-$
Line 11	$Q=Q-1$	9	$(4,2,0,9)$	
	$A=A+D$	416	$(1,3,2,9)$	
Line 15	$Q=Q-1$	8	$(3,1,8,8)$	$Quotient=8$
	2^j	32	$(2,4,5,10)$	
Line 16	2^{-j}	758	$(3,2,2,10)$	
	$R=2^{-j}A$	13	$(3,6,4,2)$	$Remainder=13$

Table 2. Calculation of $\frac{125}{14}$ by new algorithm.

Iteration	Variable	Value	RNS Moduli {5,7,9,11}	Notes
Line 1	*a*	125	(0,6,8,4)	P = 3465
	F_a = *F(b)*	1,210,464 × 2^{-25}	-	-
	b	14	(4,0,5,3)	-
	F_b = *F(b)*	135,565 × 2^{-25}	-	*F_a* − *F_b* = 1074899 × 2^{-25}, "+"
j = 0	*F_b* = 2*F_b*	271,130 × 2^{-25}	-	*F_a* – *F_b* = 939,334 × 2^{-25}, "+"
j = 1	*F_b* = 2*F_b*	542,260 × 2^{-25}	-	*F_a* – *F_b* = 668,204 × 2^{-25}, "+"
j = 2	*F_b* = 2*F_b*	1,084,520 × 2^{-25}	-	*F_a* – *F_b* = 125,944 × 2^{-25}, "+"
j =3	*F_b* = 2*F_b*	2,169,040 × 2^{-25}	-	*F_a* – *F_b* = –958,576 × 2^{-25}, "–"
Line 6	*F_b* = *F_b*/2	1,084,520 × 2^{-25}	-	-
Line 7	Δ_1 = *F_a* – *F_b*	125,944 × 2^{-25}	-	-
Line 8	*Q* = 2^j	8	(3,1,8,8)	-
j = 2	*F_b* = *F_b*/2	1,084,520 × 2^{-25}	-	Δ_1 – *F_b* = –958,576 × 2^{-25}, "–"
-	*Q*	8	(3,1,8,8)	-
j = 1	*F_b* = *F_b*/2	542,260 × 2^{-25}	-	Δ_1 – *F_b* = –416316 × 2^{-25}, "–"
-	*Q*	8	(3,1,8,8)	-
j = 0	*F_b* = *F_b*/2	271,130 × 2^{-25}	-	Δ_1 – *F_b* = –145,186 × 2^{-25}, "–"
-	*Q*	Quotient = 8	Quotient = (3,1,8,8)	-
Line 14	R = a – b·Q	Remainder = 13		

Table 3. Comparison of Hung-Parhami algorithms with new algorithm.

Operations	First Hung–Parhami Algorithm	Second Hung–Parhami Algorithm	New Algorithm
Operations in radix number system (summation, subtraction, multiplication)	54	71	18
Modular operations in RNS (modular summation, subtraction, multiplication)	45	53	5
Bit shift operations (left shift, right shift)	6	16	8
Nonmodular operations in RNS (sign detection)	1	1	0

While using the new modular division algorithm, the division specifics of fractions must be considered for avoiding rounding errors.

Actually, the function $[F(a)]_{2^{-N}}$ is an approximate variation of the function $F(a)$. When RNS numbers are replaced by their approximate characteristics, an important issue is the accuracy of the representation $[F(a)]_{2^{-N}}$ that guarantees correct results of the division operation. In accordance with the experimental studies [30,32,37], the values of N that are used for rounding and also for restoring the positional representation of numbers can be insufficient in several cases. This aspect may considerably restrict the performance of the device.

Consider the error of the division operation. For given numbers a and b in the RNS representation, the exact quotient $Q = a/b$ is approximated by the partial quotient

$$Q* = \frac{[F(a)]_{2^{-N}}}{[F(b)]_{2^{-N}}}.$$

In the quotient calculation problem, the algorithm outputs the integer part of the number $Q*$, which corresponds to the integer part of the exact quotient Q. The absolute error of the quotient is bounded by

$$\Delta Q* \leq \frac{[F(a)]_{2^{-N}} \cdot \Delta F(a) + [F(b)]_{2^{-N}} \cdot \Delta F(b)}{([F(b)]_{2^{-N}})^2}, \quad (6)$$

where $\Delta F(a)$ and $\Delta F(b)$ denote the calculation errors of the functions $[F(a)]_{2^{-N}}$ and $[F(b)]_{2^{-N}}$, respectively. Clearly, the absolute error of the quotient is growing as the dividend increases. However, special role is played by the divisor. In the current problem, the inequality $[F(a)]_{2^{-N}} < 1$ always holds; hence, the denominator of the right-hand side of Equation (6) decreases faster than the numerator, and the error is rapidly growing for sufficiently small b (see the graph in Figure 2). The relative error is demonstrated in Figure 3.

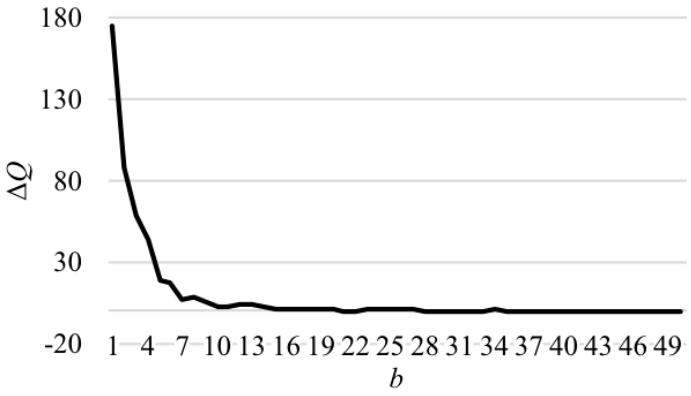

Figure 2. Absolute error for approximate division of $P - 1$ by b with RNS moduli set {5, 7, 9, 11} and $N = 17$.

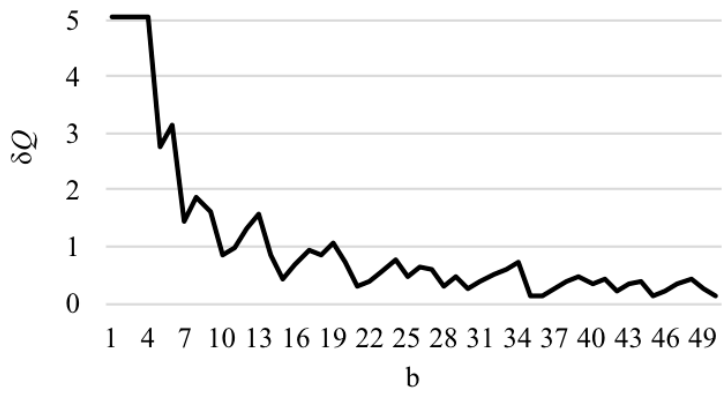

Figure 3. Relative error for approximate division of $P-1$ by b with RNS moduli set {5, 7, 9, 11} and $N = 17$.

Using Equation (6), estimate the value of N that guarantees exact division. Note that the function $F(a)$ can be calculated by the equation

$$F(a) = \frac{a}{P} = \left|\sum_{i=1}^{n} k_i \alpha_i\right|_1 = \sum_{i=1}^{n} k_i \alpha_i - r_a,$$

where r_a gives the rank of number a. By analogy with this formula, the rounded value of the function $[F(a)]_{2^{-N}}$ is obtained using the constants k_i^* rounded to N decimal points, i.e.,

$$[F(a)]_{2^{-N}} = \left|\sum_{i=1}^{n} k_i^* \alpha_i\right|_1 = \sum_{i=1}^{n} k_i^* \alpha_i - r_a.$$

For each k_i^*, the inequality $\Delta k_i^* \leq 2^{-N}$ holds. This yields the following estimate for the calculation error of $[F(a)]_{2^{-N}}$:

$$\Delta F(a) \leq 2^{-N} \sum_{i=1}^{n} \alpha_i \leq 2^{-N} \rho,$$

where $\rho = p_1 + p_2 + \ldots + p_n - n$.

Furthermore, the value $[F(a)]_{2^{-N}}$ converges to the exact value $F(a)$ as N is increased. If the constants k_i are rounded down, then the relationship

$$[F(a)]_{2^{-N}} \leq F(a)$$

holds for all a from the RNS range.

Taking this inequality into account, Equation (1) can be rewritten as

$$\begin{aligned} \Delta Q* \leq \frac{[F(a)]_{2^{-N}} \cdot \Delta F(a) + [F(b)]_{2^{-N}} \cdot \Delta F(b)}{\left([F(b)]_{2^{-N}}\right)^2} \leq 2^{-N} \rho \frac{F(a) + F(b)}{\left([F(b)]_{2^{-N}}\right)^2} \approx \\ \approx 2^{-N} \rho \frac{F(a) + F(b)}{(F(b))^2} = 2^{-n} \rho P \frac{a+b}{b^2}. \end{aligned} \tag{7}$$

In view of the earlier notes, consider the worst case causing the largest error of calculations. Choose $a = P - 1$ as the largest possible dividend and $a = 1$ as the smallest possible divisor. Using Equation (7),

$$\Delta Q* \leq 2^{-N} \rho P \frac{P - 1 + 1}{1} = 2^{-N} \rho P^2$$

Now, require that the error $\Delta Q*$ does not exceed a given threshold $\varepsilon < 1$. Then

$$\Delta Q* \leq 2^{-N}\rho P^2 \leq \varepsilon.$$

Solving this inequality in N yields

$$N \geq \log_2 \frac{\rho P^2}{\varepsilon}$$

The integer parts of the approximate and exact quotients coincide if $\varepsilon = 0.005$. Using experimental studies, it was established that the threshold $\varepsilon = 0.5$ is sufficient for exact calculations. As a result, the final bound takes the form

$$N \geq \log_2 2\rho P^2 \tag{8}$$

The right-hand side of Equation (8) is greater than the lower bound in the inequality

$$N \geq \log_2 \rho P \tag{9}$$

which is required for the exact number restoration in the algorithm. Table 4 shows the distribution of modular division errors with this bound for different RNS moduli. In addition to the general-form moduli, some sets of special form are also considered. Note that the share of faulty divisions varies from 0.5% to 14.3%. In accordance with Table 4, bound Equation (8) can be applied for exact division in RNS without any restrictions. Note that this approach and division in RNSs are difficult for theoretical study.

Table 4. Distribution of division errors with insufficient rounding accuracy of N for different sets of RNS moduli.

Set of RNS Moduli	Number of Digits for Exact Restoration N, Bits (Equation (9))	Range of Inadmissible Divisors	Share of Faulty Divisions, %	Number of Digits for Exact Division N, Bits (Equation (8))	Amount of Wrong Divisions, %
2, 3, 5, 7	12	1–14	6.67	19	0
2, 3, 5, 7, 11	16	1–154	6.71	26	0
5, 7, 9, 11	17	1–165	4.76	27	0
2, 3, 5, 7, 11, 13	21	1–858	2.86	34	0
2, 3, 5, 7, 11, 13, 17	25	1–31,907	6.25	42	0
2, 3, 5, 7, 11, 13, 17, 19	30	1–510,510	5.26	51	0
2, 3, 5, 7, 11, 13, 17, 19, 23	35	1–8,262,699	3.70	60	0
Moduli $2^n - 1$, 2^n, $2^n + 1$, $2^{2n+1} - 1$					
$n = 2$	17	1–81	4.41	25	0
$n = 3$	24	1–5819	9.09	35	0
$n = 4$	31	1–297,840	14.29	45	0
$n = 5$	38	1–779,193	1.16	55	0
Moduli $2^n - 1$, $2^n + 1$, 2^{2n}, $2^{2n} + 1$					
$n = 2$	18	1–80	1.99	27	0
$n = 3$	26	1–3360	1.28	39	0
$n = 4$	34	1–82,240	0.49	51	0
$n = 5$	42	1–4,364,800	0.41	63	0

In their paper, Hung and Parhami [25] imposed a strict constraint on the choice of the divisor: it was recommended to use any divisors from the range $\frac{P}{16} < b < \frac{3P}{16}$, which considerably restricts the method's applicability. The new approach suggested in this paper adopts bound Equation (8) for the number of digits without any constraints on the divisor.

On the other hand, the algorithms [25] and the new algorithm are comparable in terms of hardware cost and execution time for the MRNS operations (summation, subtraction, multiplication) and bit

shift operations (right and left shifts). Really, the new algorithm requires less operations of these types, but the operands have higher digit capacity.

5. Conclusions

The new algorithm described in this paper speeds up the modular division procedure in the RNS representation in comparison with the well-known counterparts. This fact can be explained by the rather simple structure of the algorithm containing uncomplicated operations, namely, summation and shift (for quotient approximation) as well as shift and subtraction (for quotient refinement). Being based on the CRT with fractions, the new algorithm does not include such operations as modular remainder calculation and number conversion into the MRNS representation. In comparison with the well-known RNS division algorithms, the new modular division algorithm has several considerable advantages. First, the division procedure involves no additional constraints on the dividend and divisor, such as representation range constraints. The only requirement of the new algorithm is that both parameters belong to the RNS range. Second, the new algorithm does not include any non-modular operation. Furthermore, the new algorithm uses less modular operations (modular summation, subtraction, multiplication) than some other RNS division algorithms. The new algorithm considerably differs from the abovementioned ones. It is unique in the sense that iterations contain shifts and subtractions. In comparison with the existing analogs, this algorithm appreciably decreases hardware cost and execution time for modular division.

The developed division algorithm can be used to design arithmetic-logic RNS devices and also to design problem-oriented RNS processors for digital signal processing, cryptography, etc. These new RNS applications will promote further development of this field of computational mathematics.

Author Contributions: Conceptualization, N.C. and P.L.; methodology, N.C. and P.L.; software, A.N.; validation, P.L. and M.D.; formal analysis, P.L., A.N. and M.D.; investigation, N.C.; resources, I.L.; data curation, A.L.; writing—original draft preparation, N.C. and P.L; writing—review and editing, M.B. and M.D.; visualization, P.L., A.N. and M.D.; supervision, N.C.; project administration, M.B.; funding acquisition, N.C., P.L., M.B. and M.D.

Funding: This research was funded by Russian Federation State task No. 2.6035.2017, the Russian Foundation for Basic Research (RFBR), grants numbers 18-07-00109 A and 19-07-00130 A, and the Council on grants of the President of the Russian Federation, grants numbers SP-2245.2018.5 and MK-6294.2018.9.

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

Appendix A

This appendix presents the modular integer division algorithm of a number a by a number b, which yields the quotient Q and also the remainder R.

The algorithm is as follows.

1. $F\_a = F(a)$, $F\_b = F(b)$
2. Set $j = 0$, $Q = 0$
3. If ($F\_b \leq F\_a$) then
4. $F\_b = 2F\_b$
5. While ($F\_b \leq F\_a$) do $j = j + 1$, $F\_b = 2F\_b$
6. $F\_b = F\_b/2$
7. $\Delta_1 = F\_a - F\_b$
8. $Q = 2^j$
9. For $I = j - 1, j - 2, \ldots, 0$ do begin
10. $F\_b = F\_b/2$
11. If ($\Delta - F\_b \geq 0$) then $Q = Q + 2^i$, $\Delta = \Delta - F\_b$
12. end
13. end if
14. $R = a - b \cdot Q$.

The detailed description of this algorithm is given below.

Line 1. Calculate the positional characteristics $F(a)$ and $F(b)$ of the dividend and divisor, respectively, with required accuracy.

Line 2. Initialize the index of iterations $j = 0$ and the quotient $Q = 0$.

Line 3. Check the condition $F\_b \leq F\_a$: if it holds, continue the division algorithm; otherwise go to line 14.

Line 4. Perform the left shift of the positional characteristic of the divisor to one binary digit.

Line 5. While $F\_b \leq F\_a$, increase $j = j + 1$ and perform the left shift of the positional characteristic of the divisor.

Line 6. Perform the right shift of the positional characteristic of the divisor to one binary digit.

Line 7. Calculate Δ as the difference between the positional characteristic F_a of the dividend and the positional characteristic F_b of the divisor that is shifted by j binary digits to the left.

Line 8. Increase the quotient Q by 2^j.

Line 9. Start the refinement procedure of the approximation series. For each i from $j - 1$ to 0, do the following operations:

Line 10. Perform the right shift of the positional characteristic of the divisor to one binary digit.

Line 11. If $\Delta - F\_b \geq 0$, then increase the quotient by 2^i and calculate the next value $\Delta = \Delta - F\_b$.

Line 12. Finish the refinement procedure of the approximation series.

Line 13. Close the condition checked in line 3.

Line 14. Calculate the remainder R.

Appendix B

This appendix presents the new algorithm for modular division of numbers $\left[\frac{a}{b}\right]$ that involves the relative values $F(a)$ and $F(b)$ in the CRT representation with fractions. A certain rule ψ is constructed to reduce each pair of numbers a, b to the fractions $F(a)$, $F(b)$, $F(b) \neq 0$; then there exists a collection Q, $\Delta = F(R)$ such that $F(a) = F(b) \cdot Q + F(R)$ and $0 \leq F(R) < F(b)$. The correctness of this algorithm can be argued as follows. Using the operation ψ, a pair of numbers $F(a)$, $F(b)$ is assigned the highest power j in $q_j 2^j$ extracted from memory such that, if $F(a) - F(b) q_j 2^j > 0$, then $\Delta_j \to F(a) - F(b) q_j 2^j$. If $\Delta_j < 0$, then division ends because $F(a) < F(b)$. If $\Delta_j \geq 0$, then $q_j = 1$ and $q_j 2^j$ is the desired partial quotient to be included in the general quotient. The highest power of $2^j = \left(2^j \bmod p_1, 2^j \bmod p_2, \ldots, 2^j \bmod p_n\right)$ is a summand of the general quotient. The analysis process starts from the highest power of 2 and ends with zero power. Next, in accordance with the operation ψ, the pair of numbers $\left(\Delta_j, \Delta_{j-1}\right)$ is assigned $q_{j-1} 2^{j-1}$ by the right shift of Δ_j (which is equivalent to division by 2). As a result, $\Delta_{j-1} = F(a) - \left(q_j 2^j + q_{j-1} 2^{j-1}\right) F(b)$ and $q_{j-1} = \begin{cases} 0 \text{ for } \Delta_{j-1} < 0, \\ 1 \text{ for } \Delta_{j-1} \geq 0. \end{cases}$

Depending on the value of q_{j-1}, the second summand is included (if $q_{j-1} = 1$) or excluded from further analysis (otherwise, as a redundant term for the quotient approximation series).

The subsequent iterations take into account only the necessary powers of 2 in parentheses. Therefore, the partial quotient is included or excluded from the general quotient using the above condition. The iterative process continues until the zero power of 2 is reached at step 0.

Consequently, the quotient is formed from the set of the partial powers of 2 that satisfy $q_j = 1$ in the RNS representation on module p_i. Let the inequality $0 \leq F(a) - \left(q_j 2^j + \ldots + q_0 2^0\right) = F(R) < F(b)$ hold at step 0. In this case, the final result is $F(a) = \left(\sum_{j \in L} q_j 2^j\right) F(b) + F(R)$, where L denotes the set of the necessary powers j of 2 in the general quotient.

At each iteration, the corresponding power of 2 is either eliminated or used as the partial quotient that must be in the general quotient. Since $q_j 2^j$ is described by the residues table for the integer powers of 2, the partial quotient has the form $Q = (Q_1, Q_2, \ldots, Q_n)$, where $Q_i = \left| \sum_{j \in L} q_j 2^j \right|_{p_i}$, $i = 1, 2, \ldots, n$; $L = 1, 2, \ldots, [\log_2 Q]$.

To summarize, the algorithm consists of two stages as follows. At the first stage, the left shift of $F(b)$ is used to find the high power of 2. The second stage is intended to analyze the successive iterations:

1. $\Delta_1 = F(a) - q_j 2^j \cdot F(b)$; if $\Delta_1 > 0$, then $2^j = \left(|2^j|_{p_1}, |2^j|_{p_2}, \ldots, |2^j|_{p_n}\right)$ is included in the general quotient; if $\Delta_1 < 0$, then 2^j is excluded.
2. $\Delta_2 = \Delta_1 - q_{j-1} 2^{j-1} \cdot F(b)$; if $\Delta_2 > 0$, then $q_{j-1} = 1$; otherwise $q_{j-1} = 0$.
3. ...
4. $\Delta_m = \Delta_{m-1} - q_0 2^0 \cdot F(b)$; if $\Delta_m > 0$, then $q_m = 1$; otherwise $q_m = 0$.

The correctness of this method is verified by trivial transformations. Then $\Delta_m = F(a) - F(b) \cdot \left(2^j + 2^{j-1} + \ldots + 2^0\right)$, and hence $F(a) = F(b) \cdot \left(2^j + 2^{j-1} + \ldots + 2^0\right) + \Delta_m$. Here $0 \leq \Delta_m < F(R)$ and j denotes each of the powers of 2 that are included in the quotient approximation series.

References

1. Szabó, N.S.; Tanaka, R.I. Residue arithmetic and its applications to computer technology. *SIAM Rev.* **1967**, *11*, 103–104.
2. Molahosseini, A.; Sousa, L.D.; Chang, C. *Embedded Systems DESIGN with Special Arithmetic and Number Systems*; Springer International Publishing: Cham, Switzerland, 2017.
3. Asif, S.; Andersson, O.; Rodrigues, J.; Kong, Y. 65-nm CMOS low-energy RNS modular multiplier for elliptic-curve cryptography. *IET Comput. Digit. Tech.* **2018**, *12*, 62–67. [CrossRef]
4. Vayalil, N.C.; Paul, M.; Kong, Y. A residue number system hardware design of fast-search variable-motion-estimation accelerator for HEVC/H.265. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *29*. [CrossRef]
5. Alia, G.; Martinelli, E. NEUROM: A ROM based RNS digital neuron. *Neural Netw.* **2005**, *18*, 179–189. [CrossRef] [PubMed]
6. 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. [CrossRef]
7. Zheng, X.; Xu, J.; Li, W. Parallel DNA arithmetic operation based on n-moduli set. *Appl. Math. Comput.* **2009**, *212*, 177–184. [CrossRef]
8. Jun, S.; Hu, Z. Method and dedicated processor for image coding based on residue number system. In Proceedings of the Modern Problems of Radio Engineering Telecommunications and Computer Science (TCSET), Lviv-Slavske, Ukraine, 21–24 February 2012; pp. 406–407.
9. Mohan, P.V.A. *Residue Number Systems*; Springer International Publishing: Cham, Switzerland, 2016.
10. Molahosseini, A.S.; Sorouri, S.; Zarandi, A.A.E. Research challenges in next-generation residue number system architectures. In Proceedings of the ICCSE 2012—Proceedings of 2012 7th International Conference on Computer Science and Education, Melbourne, VIC, Australia, 14–17 July 2012; pp. 1658–1661. [CrossRef]
11. Chervyakov, N.I.; Lyakhov, P.A.; Babenko, M.G.; Garyanina, A.I.; Lavrinenko, I.N.; Lavrinenko, A.V.; Deryabin, M.A. An efficient method of error correction in fault-tolerant modular neurocomputers. *Neurocomputing* **2016**, *205*, 32–44. [CrossRef]
12. Chervyakov, N.I.; Molahosseini, A.S.; Lyakhov, P.A.; Babenko, M.G.; Deryabin, M.A. Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem. *Int. J. Comput. Math.* **2017**, *94*, 1833–1849. [CrossRef]
13. Kaplun, D.; Butusov, D.; Ostrovskii, V.; Veligosha, A.; Gulvanskii, V.; Kaplun, D.; Butusov, D.; Ostrovskii, V.; Veligosha, A.; Gulvanskii, V. Optimization of the FIR filter structure in finite residue field algebra. *Electronics* **2018**, *7*, 372. [CrossRef]
14. Hiasat, A. Efficient RNS scalers for the extended three-moduli set $(2^{n}-1, 2^{n+p}, 2^{n}+1) $. *IEEE Trans. Comput.* **2017**, *66*, 1253–1260. [CrossRef]
15. Kumar, S.; Chang, C.-H.; Tay, T.F. New algorithm for signed integer comparison in $\{2^{n+k},2^{n}-1,2^{n}+1,2^{n\pm 1}-1\}$ and its efficient hardware implementation. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *64*, 1481–1493. [CrossRef]

16. Nakahara, H.; Nakanishi, H.; Iwai, K.; Sasao, T. An FFT circuit for a spectrometer of a radio telescope using the nested RNS including the constant division. *ACM SIGARCH Comput. Archit. News* **2017**, *44*, 44–49. [CrossRef]
17. Mrabet, A.; El-Mrabet, N.; Bouallegue, B.; Mesnager, S.; Machhout, M. An efficient and scalable modular inversion/division for public key cryptosystems. In Proceedings of the 2017 International Conference on Engineering & MIS (ICEMIS), Monastir, Tunisia, 8–10 May 2017; pp. 1–6. [CrossRef]
18. Chren, W.A. A new residue number system division algorithm. *Comput. Math. Appl.* **1990**, *19*, 13–29. [CrossRef]
19. 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]
20. Chiang, J.-S.; Lu, M. A general division algorithm for residue number systems. In Proceedings of the 10th IEEE Symposium on Computer Arithmetic, Grenoble, France, 26–28 June 1991; pp. 76–83. [CrossRef]
21. Gamberger, D. New approach to integer division in residue number systems. In Proceedings of the 10th IEEE Symposium on Computer Arithmetic, Grenoble, France, 26–28 June 1991; pp. 84–91. [CrossRef]
22. Lu, M.; Chiang, J.-S. A novel division algorithm for the residue number system. *IEEE Trans. Comput.* **1992**, *41*, 1026–1032. [CrossRef]
23. Bajard, J.; Rico, F. How to improve division in residue number systems. In Proceedings of the 16th IMACS World Congress, Lausanne, Switzerland, 21–25 August 2000; pp. 110–121.
24. Hiasat, A.A. Semi-Custom VLSI Design and Implementation of a New Efficient RNS Division Algorithm. *Comput. J.* **1999**, *42*, 232–240. [CrossRef]
25. Hung, C.Y.; Parhami, B. Fast RNS division algorithms for fixed divisors with application to RSA encryption. *Inf. Process. Lett.* **1994**, *51*, 163–169. [CrossRef]
26. Hung, C.Y.; Parhami, B. An approximate sign detection method for residue numbers and its application to RNS division. *Comput. Math. Appl.* **1994**, *27*, 23–35. [CrossRef]
27. Hiasat, A.A.; Abdel-Aty-Zohdy, H.S. Design and implementation of an RNS division algorithm. In Proceedings of the Proceedings 13th IEEE Sympsoium on Computer Arithmetic, Asilomar, CA, USA, 6–9 July 1997; pp. 240–249. [CrossRef]
28. Yang, J.-H.; Chang, C.-C.; Chen, C.-Y. A high-speed division algorithm in residue number system using parity-checking technique. *Int. J. Comput. Math.* **2004**, *81*, 775–780. [CrossRef]
29. Chang, C.-C.; Yang, J.-H. A Division algorithm using bisection method in Residue Number System. *Int. J. Comput.* **2013**, *2*, 59–66.
30. Talahmeh, S.; Siy, P. Arithmetic division in RNS using Galois Field GF(p). *Comput. Math. Appl.* **2000**, *39*, 227–238. [CrossRef]
31. Chang, C.-C.; Lai, Y.-P. A division algorithm for residue numbers. *Appl. Math. Comput.* **2006**, *172*, 368–378. [CrossRef]
32. 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]
33. Patronik, P.; Piestrak, S.J. Design of reverse converters for the new RNS moduli set $\{2^n+1, 2^n-1, 2^n, 2^{n-1}+1\}$ (n odd). *IEEE Trans. Circuits Syst. I Regul. Pap.* **2014**, *61*, 3436–3449. [CrossRef]
34. Tay, T.F.; Chang, C.-H.; Sousa, L. Base transformation with injective residue mapping for dynamic range reduction in RNS. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2015**, *62*, 2248–2259. [CrossRef]
35. Vun, C.H.; Premkumar, A.B.; Zhang, W. A new RNS based DA approach for inner product computation. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2013**, *60*, 2139–2152. [CrossRef]
36. Kouretas, I.; Paliouras, V. A low-complexity high-radix RNS multiplier. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2009**, *56*, 2449–2462. [CrossRef]
37. Younes, D.; Steffan, P. A comparative study on different moduli sets in residue number system. In Proceedings of the 2012 International Conference on Computer Systems and Industrial Informatics, Sharjah, UAE, 18–20 December 2012; pp. 1–6. [CrossRef]