electronics

MDPI

Article

Construction of Residue Number System Using Hardware Efficient Diagonal Function

Maria Valueva [1,*], Georgii Valuev [1], Nataliya Semyonova [2], Pavel Lyakhov [1], Nikolay Chervyakov [1], Dmitry Kaplun [3] and Danil Bogaevskiy [3]

[1] Department of Applied Mathematics and Mathematical Modeling, North-Caucasus Federal University, Stavropol 355009, Russia; elasgreece92@mail.ru (G.V.); ljahov@mail.ru (P.L.); k-fmf-primath@stavsu.ru (N.C.)

[2] Department of Higher Algebra and Geometry, North-Caucasus Federal University, Stavropol 355009, Russia; algebra223@yandex.ru

[3] Department of Automation and Control Processes, St. Petersburg Electrotechnical University "LETI", Saint Petersburg 197376, Russia; dikaplun@etu.ru (D.K.); dvbogaevskiy@etu.ru (D.B.)

* Correspondence: mriya.valueva@mail.ru; Tel.: +7-988-745-8885

Received: 8 March 2019; Accepted: 18 June 2019; Published: 20 June 2019

check for updates

Abstract: The residue number system (RNS) is a non-positional number system that allows one to perform addition and multiplication operations fast and in parallel. However, because the RNS is a non-positional number system, magnitude comparison of numbers in RNS form is impossible, so a division operation and an operation of reverse conversion into a positional form containing magnitude comparison operations are impossible too. Therefore, RNS has disadvantages in that some operations in RNS, such as reverse conversion into positional form, magnitude comparison, and division of numbers are problematic. One of the approaches to solve this problem is using the diagonal function (DF). In this paper, we propose a method of RNS construction with a convenient form of DF, which leads to the calculations modulo 2^n, $2^n - 1$ or $2^n + 1$ and allows us to design efficient hardware implementations. We constructed a hardware simulation of magnitude comparison and reverse conversion into a positional form using RNS with different moduli sets constructed by our proposed method, and used different approaches to perform magnitude comparison and reverse conversion: DF, Chinese remainder theorem (CRT) and CRT with fractional values (CRTf). Hardware modeling was performed on Xilinx Artix 7 xc7a200tfbg484-2 in Vivado 2016.3 and the strategy of synthesis was highly area optimized. The hardware simulation of magnitude comparison shows that, for three moduli, the proposed method allows us to reduce hardware resources by 5.98–49.72% in comparison with known methods. For the four moduli, the proposed method reduces delay by 4.92–21.95% and hardware costs by twice as much by comparison to known methods. A comparison of simulation results from the proposed moduli sets and balanced moduli sets shows that the use of these proposed moduli sets allows up to twice the reduction in circuit delay, although, in several cases, it requires more hardware resources than balanced moduli sets.

Keywords: residue number system (RNS); diagonal function (DF); Chinese remainder theorem (CRT)

1. Introduction

The residue number system (RNS) is a non-positional number system that allows large length numbers to be presented as numbers in independent bits of a small length, which enables computations and the organizing of their parallelisms to be sped up. RNS has several advantages, such as the possibility of faster addition and multiplication compared to all other number systems. Moreover, the use of short numbers in RNS computations can significantly reduce the power consumption of digital devices [1]. It is useful in the synthesis of RNS computational devices with parallel structure, such as field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC). All these attractive features increase interest to RNS in the areas where large amounts of computation are needed. The applications of RNS are digital signal processing [2–4], cryptography [5–7], digital image processing [8], cloud computing [9], Internet of Things [10] and others. In [11], the authors propose a technique to estimate real-valued numbers by means of the Chinese remainder theorem (CRT), employing for this goal a Kroenecker based M-Estimation, to improve robustness. A new method based on the Chinese remainder theorem (CRT) is proposed for absolute position computation in [12]. This has advantages in terms of hardware and flexibility because it does not use memory. The authors of [13] offer to use RNS to improve the performance of the convolutional neural network developed for pattern recognition tasks. Reference [14] describes the method of construction for finite impulse response filers using RNS.

However, the limitations of RNS include some operations such as reverse conversion into positional form, magnitude comparison and division of numbers in RNS [15,16]. These limitations exist because RNS is a non-positional number system, and magnitude comparison of numbers in RNS form is impossible, so the division operation consists of a magnitude comparison operation that is also a problematic operation. Improving the efficiency of the comparison operation in RNS is something that can be used in the development of new approaches to the implementation of other problematic operations in RNS, such as subtraction-based division and the detection of dynamic range overflow. Dynamic range overflow detectors in RNS are widely applied in the design of fault-tolerant systems and secure communication channels [17].

The state-of-the-art in the described problem is as follows. The most common approaches to performing non-modular RNS operations are based on mixed radix conversion (MRC) and the Chinese remainder theorem (CRT) [1,18]. Another class of approaches to perform magnitude comparison in RNS, which is based on the core functions defined from the RNS to the integer [19], was first proposed Akushskii et al. [20]. Recently new alternatives have been developed for the implementation of the non-modular RNS operations problem. These approaches are the use of CRT with fractional values (CRTf) [21] and diagonal function (DF) [22,23]. References [24] and [25] demonstrate that the use of DF has a significant drawback in the necessity to perform modulo sum of quotients (SQ) operations. The authors of these papers show that DF usually does not provide advantages in comparison with MRC and CRT. Therefore, in this paper, we will discuss the issue of constructing RNS with a convenient form of DF, which leads to the calculations modulo 2^n, 2^n-1 or 2^n+1 since the numbers of this form have very effective methods of hardware implementation, as designed in [26–28]. How balanced the moduli set is plays an important role in this method. Table 1 shows samples of known moduli sets.

Table 1. Known balanced moduli sets.

Number of Modules	Moduli Set	Condition	References
3	$\{2^n-1, 2^n, 2^n+1\}$		[29,30]
	$\left\{2^n-1, 2^{n+p}, 2^n+1\right\}$		[31]
	$\left\{2^{2n+p}, 2^{2n}-1, 2^{2n}+1\right\}$	n odd, $p \le \frac{n-5}{2}$	[32]
4	$\left\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1\right\}$	n even	[33]
	$\left\{2^n+1, 2^n-1, 2^n, 2^{n-1}+1\right\}$	n odd	[32]
	$\left\{2^n+1, 2^n-1, 2^n, 2^{n+1}+1\right\}$	n odd	[34]
	$\left\{2^{n+k}, 2^n-1, 2^n+1, 2^{n\pm1}-1\right\}$	n even, $k \in [0, n]$	[35]
5	$\left\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1, 2^{n-1}-1\right\}$	n even	[36]
	$\left\{2^{2n+p}, 2^n-1, 2^n+1, 2^n-2^{\frac{n+1}{2}}+1, 2^n+2^{\frac{n+1}{2}}+1\right\}$	n odd, $p \le \frac{n-5}{2}$	[33]
8	$\left\{2^{n-5}-1, 2^{n-3}-1, 2^{n-3}+1, 2^{n-2}+1, 2^{n-1}-1, 2^{n-1}+1, 2^n, 2^n+1\right\}$	$n=2k, k \ge 4$	[37]

The proposed approach to the construction of RNS can be effective in those applications in which the comparison operation is a significant part of the calculations. One of the examples of such an application is the motion estimation on video, estimated by using high-efficiency video coding (HEVC/H.265) [38]. Another example of a such application is customized signal processing units. For example, the sorting network uses a large number of comparators and is one of the key elements in electronic finance data management systems, digital computers and communication systems [39]. Due to the excessive number of magnitude comparisons required in sorting a large pool of data, the speed of the magnitude comparator determines the overall delay of the sorting process [35].

The rest of the paper is organized as follows. Section 2 discusses RNS issues, represented numbers, and arithmetic operations in RNS. The Section 3 presents the construction of RNS with a convenient form of DF and the results of the hardware simulation of magnitude comparison and reverse conversion into the positional form using CRT, CRTf, and DF. Section 4 discusses the methods of RNS construction presented in this paper and hardware simulation results. The conclusion of the paper is reported in Section 5.

2. Materials and Methods

2.1. Background on RNS

Numbers in RNS are represented in the form of relatively prime numbers which are called moduli $\beta = \{m_1, \ldots, m_k\}$, $GCD\left(m_i, m_j\right) = 1$, for $i \neq j$. Any integer number $0 \leq X < M = \prod_{i=1}^{k} m_i$ can be uniquely represented in RNS as a tuple $\{x_1, x_2, \ldots, x_k\}$, where $x_i = |X|_{m_i} = X \bmod m_i$. Operations of addition, subtraction, and multiplication in RNS are defined by the formulas showing the carry-free parallel nature of RNS:

$$A \pm B = \left(|a_1 \pm b_1|_{m_1}, \ldots, |a_n \pm b_n|_{m_n}\right),\ A \times B = \left(|a_1 \times b_1|_{m_1}, \ldots, |a_n \times b_n|_{m_n}\right) \tag{1}$$

The reverse conversion of a number X from residues $\{x_1, x_2, \ldots, x_k\}$ is based on CRT

$$X = \left|\sum_{i=0}^{n} \left|\left|M_i^{-1}\right|_{m_i} x_i\right|_{m_i} M_i\right|_M, \tag{2}$$

where $M_i = M/m_i$, $\gamma_i = \left|M_i^{-1}\right|_{m_i}$ and $\left|M_i^{-1}\right|_{m_i}$ means a multiplicative inverse of M_i modulo m_i.

The DF is defined as

$$D(X) = \left|\sum_{i=1}^{n} k_i x_i\right|_{SQ}, \tag{3}$$

where $SQ = \sum_{i=1}^{n} M_i$ is called the "diagonal modulus" of the RNS and $k_i = \left|-m_i^{-1}\right|_{SQ}$. The principles of applying the DF for reverse conversion and numbers comparison are thoroughly shown in [24] and [25]. Reverse conversion using DF can be implemented by the formula

$$X = \frac{M \cdot D(X) + \sum_{i=1}^{n} x_i M_i}{SQ}, \tag{4}$$

In [25], magnitude comparison is presented using DF. The work uses the magnitude comparison Algorithm 1 of X and Y, presented in [22], which relies on the following properties of the DF.

Algorithm 1. Magnitude comparison using DF [22].

Input: $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_n\}$, $k = \{k_1, \ldots, k_n\}$, SQ;
Variable: D_x, D_y;
Calculations:
$D_x = 0$; $D_y = 0$;
for $i = \overline{0, n-1}$ do
$D_x = |D_x + k_i x_i|_{SQ}$;
$D_y = |D_y + k_i y_i|_{SQ}$;
end for;
if $D_x < D_y$ then
return ("$X < Y$");
else
if $D_x > D_y$ then
return ("$X > Y$");
else
if $x_1 < y_1$ then
return ("$X < Y$");
else
if $x_1 > y_1$ then
return ("$X > Y$");
else
return ("$X = Y$");
end if;
end if;
end if;
end if;

It is obvious that the main obstacle to the development of very-large-scale integration (VLSI) architectures based on the DF is the necessity to perform modulo SQ operations. Below, we show how to construct RNS with a convenient form of DF that leads to modulo 2^n, $2^n - 1$ or $2^n + 1$ computations.

2.2. Construction Methods of RNS with Hardware Efficient DF

The choice of the optimal moduli set is a very important question in RNS theory since it has an impact on performance and the quality of operations. In [26–28], authors perform high-speed architectures of modulo $2^n \pm 1$ adders. The use of moduli 2^n, $2^n - 1$ or $2^n + 1$ allows for there to be an increase in computation performance. In addition, the choice of enough RNS dynamic range is a very important question too. In [40], authors considered the influence of the RNS dynamic range on the quality of image filtering. Therefore, it is necessary to choose optimal moduli sets, so we propose the method of RNS construction with a convenient form of DF.

Let us consider two possible cases.

1. Among the RNS moduli $m_1, m_2, \ldots, m_n$ there is an even one, and the others are odd. Then among $M_1, M_2, \ldots, M_n$ there is an odd one, and the others are even and therefore SQ is odd.
2. All RNS moduli $m_1, m_2, \ldots, m_n$ are odd. Then all $M_1, M_2, \ldots, M_n$ are odd and parity of SQ is the same as the parity of the number of moduli n.

2.2.1. RNS with Even Module

One can suppose that $m_1, m_2, \ldots, m_{n-1}$ are odd and $m_n = 2^\rho(2l_n + 1)$ is even. We will choose m_1, $m_2, \ldots, m_n$ in such a way to satisfy $SQ = 2^k - 1$ or $SQ = 2^k + 1$. We denote $M_0 = m_1 m_2 \ldots m_{n-1}$ and $S_0 = \frac{M_0}{m_1} + \frac{M_0}{m_2} + \ldots + \frac{M_0}{m_{n-1}}$, thus $SQ = S_0 m_n + M_0$. If n is odd then S_0 is even and therefore $S_0 = 2^\omega$ or $S_0 = 2^\omega(2l_0 + 1)$. If n is even then S_0 is odd and $S_0 = 2l_0 + 1$.

If $SQ = 2^k - 1$ then three cases are possible

$$2^k - 1 = M_0 + 2^{\omega} 2^{\rho}(2l_n + 1) \text{ or} \tag{5}$$

$$2^k - 1 = M_0 + 2^{\omega}(2l_0 + 1)2^{\rho}(2l_n + 1) \text{ or} \tag{6}$$

$$2^k - 1 = M_0 + (2l_0 + 1)2^{\rho}(2l_n + 1). \tag{7}$$

Hence

$$2^k = M_0 + 1 + 2^{\omega} 2^{\rho}(2l_n + 1) \text{ or} \tag{8}$$

$$2^k = M_0 + 1 + 2^{\omega}(2l_0 + 1)2^{\rho}(2l_n + 1) \text{ or} \tag{9}$$

$$2^k = M_0 + 1 + (2l_0 + 1)2^{\rho}(2l_n + 1). \tag{10}$$

We choose M_0 in a way that $M_0 = m_1 m_2 \ldots m_{n-1} = 2^t - 1$ is a composite number and $GCD\left(m_i, m_j\right) = 1$ for $i \neq j$. Since among the $2^t - 1$ numbers, there are composite numbers much more than prime numbers, then the choice of M_0 is obviously possible. Therefore

$$2^k = 2^t + 2^{\omega} 2^{\rho}(2l_n + 1) \text{ or} \tag{11}$$

$$2^k = 2^t + 2^{\omega}(2l_0 + 1)2^{\rho}(2l_n + 1) \text{ or} \tag{12}$$

$$2^k = 2^t + (2l_0 + 1)2^{\rho}(2l_n + 1). \tag{13}$$

Hence, since $t < k$ and $t \leq \omega + \rho$ we have

$$2^{k-t} = 1 + 2^{\omega+\rho-t}(2l_n + 1) \text{ or} \tag{14}$$

$$2^{k-t} = 1 + 2^{\omega+\rho-t}(2l_0 + 1)(2l_n + 1) \text{ or} \tag{15}$$

$$2^{k-t} = 1 + 2^{\rho-t}(2l_0 + 1)(2l_n + 1). \tag{16}$$

Suppose that $\omega + \rho - t = 0$ or $\rho - t = 0$. We have

$$2^{k-t} = 1 + (2l_n + 1) \text{ or} \tag{17}$$

$$2^{k-t} = 1 + (2l_n + 1) \text{ or} \tag{18}$$

$$2^{k-t} = 1 + (2l_0 + 1)(2l_n + 1). \tag{19}$$

Hence

$$2l_n + 1 = 2^j - 1, \text{ where } j = 1, 2, 3, \ldots \tag{20}$$

$$\text{or } 2^{k-t} \equiv 1(\mathrm{mod}(2l_0 + 1)) \tag{21}$$

Congruence (21) is solvable due to the fact that $GCD(2, 2l_0 + 1) = 1$. If r is an order of 2 modulo $2l_0 + 1$, then $k - t = rj$, where $j = 1, 2, \ldots, n$. Hence $2l_n + 1 = \frac{2^{rj}-1}{2l_0+1}$. From this, if it is necessary to find $M = m_1 m_2 \ldots m_n$, where m_n is even and $SQ = 2^k - 1$ then proceed as follows.

1. Choose a composite $M_0 = m_1 m_2 \ldots m_{n-1} = 2^t - 1$.
2. Compute S_0.
3. Consider the possible cases.

   a. If $S_0 = 2^{\omega}$ then $\rho = t - \omega$, $2l_n + 1 = 2^j - 1$, were $GCD\left(2^j - 1, m_i\right) = 1$ for $i = 1, 2, \ldots, n - 1$. $m_n = 2^{t-\omega}\left(2^j - 1\right)$, where $j = 1, 2, 3, \ldots$, $GCD\left(2^j - 1, m_i\right) = 1$, $i = 1, 2, \ldots, n - 1$.
   b. If $S_0 = 2^{\omega}(2l_0 + 1)$ then $\rho = t - \omega$, $2l_n + 1 = \frac{2^{rj}-1}{2l_0+1}$, where $GCD(2l_n + 1, m_i) = 1$, $i = 1, 2, \ldots, n - 1$, and r is order of 2 modulo $2l_0 + 1$.

c. If $S_0 = 2l_0 + 1$ then $\rho = t$, $2l_n + 1 = \frac{2^{rj}-1}{2l_0+1}$, where $GCD(2l_n + 1, m_i) = 1$, $i = 1, 2, \ldots, n-1$, and r is order of 2 modulo $2l_0 + 1$.

Example 1. *Suppose that* $M_0 = m_1 m_2 = 3 \cdot 5 = 2^4 - 1$, $t = 4$. *Then* $S_0 = 3 + 5 = 2^3$, $\omega = 3$. $m_3 = 2^{\rho}(2l_3 + 1)$, $\rho = 4 - 3 = 1$. $2l_3 + 1 = 2^j - 1$, $j = 1, 2, 3, \ldots$, $GCD\big(2^j - 1, 3\big) = 1$, $GCD(2^{\varepsilon j} - 1, 5) = 1$. *Examining a power of two, we have*

$2^1 - 1 = 1$, $2l_3 + 1 = 1$, $m_3 = 2$.
$2^2 - 1 = 3$, $GCD(3, 3) \neq 1$.
$2^3 - 1 = 7$, $2l_3 + 1 = 7$, $m_3 = 14$.
$2^4 - 1 = 15$, $GCD(15, 3) \neq 1$.
$2^5 - 1 = 31$, $2l_3 + 1 = 31$, $m_3 = 62$ *etc.*

Thus, we obtained the following RNS: $\{3, 5, 2\}$, $SQ = 31 = 2^5 - 1$, $\{3, 5, 14\}$, $SQ = 127 = 2^8 - 1$, $\{3, 5, 62\}$, $SQ = 511 = 2^9 - 1$.

Note. For the case $SQ = 2^k + 1$ one needs to take $M_0 = 2^t + 1$. The conclusions obtained are the same as for $SQ = 2^k - 1$.

Example 2. *Suppose that* $M_0 = m_1 m_2 = 3 \cdot 11 = 2^5 + 1$, $t = 5$. *Then*

$S_0 = 3 + 11 = 14 = 2^1 \cdot 7$, $\omega = 1$, $2l_0 + 1 = 7$.
$m_3 = 2^{\rho}(2l_3 + 1)$, $\rho = 5 - 1 = 4$, $2^r \equiv 1(mod7)$, $r = 3j$.
$\frac{2^3-1}{7} = 1$, $2l_3 + 1 = 1$, $m_3 = 2^4 \cdot 1 = 16$.
$\frac{2^6-1}{7} = \frac{63}{7} = 9$, $GCD(9, 3) \neq 1$.
$\frac{2^9-1}{7} = \frac{511}{7} = 73$, $2l_3 + 1 = 73$, $m_3 = 16 \cdot 73 = 1168$ *etc.*

Thus we obtained the following RNS:$\{3, 11, 16\}$, $S = 257 = 2^8 + 1$, $\{3, 11, 1168\}$, $S = 16385 = 2^{14} + 1$.

2.2.2. RNS with Odd Moduli

We only consider the most important practical cases, for example, when RNS contains three, four or five moduli [41].

Case 1. RNS with three moduli. In analogy with the above notations $M_0 = m_1 m_2$, $S_0 = m_1 + m_2$, and $SQ = M_0 + S_0 m_3$. One can verify that $S \equiv 3(\text{mod}4)$. Let us see whether it is possible for the odd m_1 and m_2 to choose such an odd m_3, such that $SQ = 2^k - 1$. If $S = 2^k - 1$ then $2^k = S + 1 = M_0 + 1 + S_0 m_3$. It is clear that $GCD(M_0 + 1, S_0) = 2^{\omega}(2l + 1)$. If $2l + 1 \neq 1$, then the right part of equality

$$2^k = M_0 + 1 + S_0 m_3 \text{ or} \tag{22}$$

divisible by $2l + 1$, and left part of Equality (22) is not divisible by $2l + 1$. This means that for the satisfy Equality (22) it is necessary that

$$GCD(M_0 + 1, S_0) = 2^{\omega}. \tag{23}$$

Under Condition (23) we have

$$2^{k-\omega} = \frac{M_0 + 1}{2^{\omega}} + \frac{S_0}{2^{\omega}} m_3, \tag{24}$$

where $GCD\Big(\frac{M_0+1}{2^{\omega}}, \frac{S_0}{2^{\omega}}\Big) = 1$. If one of the numbers $\frac{M_0+1}{2^{\omega}}$ or $\frac{S_0}{2^{\omega}}$ is even, then (24) is impossible. Thus, for the validity of (24), it is necessary that both numbers $\frac{M_0+1}{2^{\omega}}$ and $\frac{S_0}{2^{\omega}}$ are odd.

Suppose that both conditions are performed.

1. $GCD(M_0 + 1, S_0) = 2^{\omega}$.
2. $\frac{M_0+1}{2^{\omega}}$ and $\frac{S_0}{2^{\omega}}$ are odd.

Let us write (24) as a congruence

$$2^{k-\omega} \equiv \frac{M_0 + 1}{2^{\omega}} \left(\mathrm{mod}\ \frac{S_0}{2^{\omega}}\right). \quad (25)$$

If $\frac{S_0}{2^{\omega}}$ is prime and 2 is a primitive root modulo $\frac{S_0}{2^{\omega}}$ then Congruence (25) will have solutions concerning $k - \omega$ by $\mathrm{mod}\left(\frac{S_0}{2^{\omega}} - 1\right)$. Suppose that ρ is the smallest non-negative solution of Congruence (25). Then

$$k - \omega = \rho + \left(\frac{S_0}{2^{\omega}} - 1\right)t,\ t = 0, 1, 2, \ldots \quad (26)$$

And therefore $2^{\rho+(\frac{S_0}{2^{\omega}}-1)t} = \frac{M_0+1}{2^{\omega}} + \frac{S_0}{2^{\omega}} m_3$. Hence $\frac{S_0}{2^{\omega}} m_3 = 2^{\rho+(\frac{S_0}{2^{\omega}}-1)t} - \frac{M_0+1}{2^{\omega}}$. This means that

$$m_3 = \frac{2^{\rho+(\frac{S_0}{2^{\omega}}-1)t} - \frac{M_0+1}{2^{\omega}}}{\left(\frac{S_0}{2^{\omega}}\right)},\ t = 0, 1, 2, \ldots \quad (27)$$

According to the RNS definition, the number m_3 must be relatively prime with m_1 and m_2.

If the number $\frac{S_0}{2^{\omega}}$ is prime and 2 is not a primitive root modulo $\frac{S_0}{2^{\omega}}$ then Congruence (25) may have no solutions. In addition, Congruence (25) may have no solutions if $\frac{S_0}{2^{\omega}}$ is a composite number. Thus, to construct RNS with three odd moduli and $SQ = 2^k - 1$, four conditions must be fulfilled.

1. $GCD(m_1 m_2 + 1, m_1 + m_2) = 2^{\omega}$.
2. $\frac{m_1 m_2+1}{2^{\omega}}$ is odd.
3. $\frac{m_1+m_2}{2^{\omega}}$ is prime and not equal to 2.
4. 2 is a primitive root modulo $\mathrm{mod}\frac{m_1+m_2}{2^{\omega}}$.

Note that these conditions are not sufficient, since the numbers m_3 found by Formula (27) may not be relatively prime with m_1 or m_2.

Example 3. *Suppose that* $m_1 = 3$, $m_2 = 7$. *Then* $GCD(3 \cdot 7 + 1, 3 + 7) = GCD(12, 10) = 2$, $\frac{3 \cdot 7+1}{2} = 11$ *is odd,* $\frac{3+7}{2} = 5$ *is prime, 2 is a primitive root modulo 5.*

From the equality $2^{k-1} = 11 + 5m_3$, we obtain a congruence $2^{k-1} \equiv 11(\mathrm{mod} 5)$ which implies $k - 1 = 4t$, $t = 1, 2, \ldots$ or $2^{4t} = 11 + 5m_3$, $m_3 = \frac{2^{4t}-11}{5}$.

Testing of the value t gives the following result:

$t = 1$ gives $m_3 = 1 < 2$,
$t = 2$ gives $m_3 = \frac{256-11}{5} = 49$, $GCD(49, 7) \neq 1$,
$t = 3$ gives $m_3 = \frac{4096-11}{5} = 817$, $GCD(817, 3) = 1$, $GCD(817, 7) = 1$.

So, we get the RNS $\{3, 7, 817\}$ with $SQ = 8191 = 2^{13} - 1$.

Case 2. RNS with 4 moduli. In this case, SQ is even. Consider the problem: for m_1, m_2, m_3 choose m_4 in such a way as to $SQ = 2^k$. If we denote $M_0 = m_1 m_2 m_3$, $S_0 = m_1 m_2 + m_1 m_3 + m_2 m_3$ then $S = M_0 + S_0 m_4$. It is clear that $GCD(M_0, S_0) = 1$. From the equality $2^k = M_0 + S_0 m_4$ follows

$$2^k \equiv M_0(\mathrm{mod} S_0). \quad (28)$$

If S_0 is prime and 2 is a primitive root modulo S_0 then Congruence (28) has a solution on k. Suppose that ρ is the smallest non-negative solution of ongruence (28). Then $k = \rho + (S_0 - 1)t$, $t = 0, 1, 2, \ldots$. It means that $2^{\rho+(S_0-1)t} = M_0 + S_0 m_4$ from which

$$m_4 = \frac{2^{\rho+(S_0-1)t} - M_0}{S_0},\ t = 0, 1, 2, \ldots \tag{29}$$

Since $GCD\left(2^{\rho+(S_0-1)t}, m_1\right) = GCD\left(2^{\rho+(S_0-1)t}, m_2\right) = GCD\left(2^{\rho+(S_0-1)t}, m_3\right) = 1$ then for any $t = 0, 1, 2, \ldots$ it will be obtained that the number m_4 is relatively prime with m_1, m_2 and m_3. If S_0 is a composite number, then Congruence (28) may have no solutions. If S_0 is prime and 2 is not a primitive root modulo S_0 then Congruence (28) may have no solutions too. In other words, to construct RNS with four odd moduli and $SQ = 2^k$, two conditions must be fulfilled.

1. $S_0 = m_1 m_2 + (m_1 + m_2) m_3$ is prime.
2. 2 is a primitive root modulo S_0

Example 4. *Suppose that* $m_1 = 3$*,* $m_2 = 7$*,* $m_3 = 11$*. In this case* $M_0 = m_1 m_2 m_3 = 231$ *and* $S_0 = m_1 m_2 + (m_1 + m_2) m_3 = 131$ *is prime. Two is a primitive root modulo 131. From the equality* $2^k = 231 + 131 m_4$ *follows congruence* $2^k \equiv 100 (mod 131)$*. The least nonnegative solution of this congruence is* 94*, therefore* $k = 94 + 130t$*,* $t = 0, 1, 2, \ldots$

Hence $m_4 = \frac{2^{94+130t} - 231}{131}$, $t = 0, 1, 2, \ldots$ For $t = 0$ we have $m_4 = \frac{2^{94} - 231}{131}$.
We received RNS $\left\{3, 7, 11, \frac{2^{94}-231}{131}\right\}$ with $SQ = 2^{94}$.

Case 3. RNS with 5 moduli. In analogy with the above notations, we denote $M_0 = m_1 m_2 m_4 m_4$ and $S_0 = m_1 m_2 m_3 + m_1 m_2 m_4 + m_1 m_3 m_4 + m_2 m_3 m_4$ and $S = M_0 + S_0 m_5$. One can verify that $S \equiv 1 (\mathrm{mod} 4)$. Let us see whether it is possible for the odd m_1, m_2, m_3 and m_4 choose such an odd m_5 that the $SQ = 2^k + 1$.

If $S = 2^k + 1$ then $2^k = S - 1 = M_0 - 1 + S_0 m_5$. Similar to case 1, the equality

$$2^k = M_0 - 1 + S_0 m_5 \tag{30}$$

is possible if:

1. $GCD(M_0 - 1, S_0) = 2^\omega$;.
2. $\frac{M_0-1}{2^\omega}$ and $\frac{S_0}{2^\omega}$ are odd

Then $2^{k-\omega} = \frac{M_0-1}{2^\omega} + \frac{S_0}{2^\omega} m_5$ or

$$2^{k-\omega} \equiv \frac{M_0 - 1}{2^\omega} \left(\mathrm{mod}\ \frac{S_0}{2^\omega}\right). \tag{31}$$

If $\frac{S_0}{2^\omega}$ is prime and 2 is a primitive root modulo $\frac{S_0}{2^\omega}$ then Congruence (31) has solutions concerning $k - \omega$ modulo $\frac{S_0}{2^\omega} - 1$. Suppose that ρ is the smallest nonnegative such a solution. Then $2^{\rho+(\frac{S_0}{\omega}-1)t} = \frac{M_0-1}{2^\omega} + \frac{S_0}{2^\omega} m_5$ and $t = 0, 1, 2, \ldots$

Hence $m_5 = \frac{2^{\rho+(\frac{S_0}{\omega}-1)t} - \frac{M_0-1}{2^\omega}}{\left(\frac{S_0}{2^\omega}\right)}$ wherein t should be chosen so that m_5 will be relatively prime with m_1, m_2, m_3 and m_4.

Example 5. *Suppose that* $m_1 = 3$*,* $m_2 = 5$*,* $m_3 = 7$ *and* $m_4 = 11$*. Then*

$M_0 = 1155$, $M_0 - 1 = 1154$, $S_0 = 886$.
$GCD(M_0 - 1, S_0) = GCD(1154, 886) = 2$.

$\frac{M_0-1}{2} = 577$ *is odd,* $\frac{S_0}{2} = 443$ *is prime and 2 is a primitive root modulo 443.*

Hence, we can choose such an odd number as m_5 that the following $SQ = 2^k + 1$. From the equality $2^{k-1} = 577 + 443m_5$, we obtain $2^{k-1} \equiv 557(\text{mod}443)$, from which $k - 1 = 53 + 443t$, $t = 0, 1, 2, \ldots$

Hence $m_5 = \frac{2^{53+443t}-577}{443}$, $t = 0, 1, 2, \ldots$ where t needs to be chosen such that $GCD(3, m_5) = GCD(5, m_5) = GCD(7, m_5) = GCD(11, m_5) = 1$.

The smallest $t = 3$ where this condition is performed $t = 3$, therefore $m_5 = \frac{2^{1382}-577}{443}$. We obtain RNS $\left\{3, 5, 7, 11, \frac{2^{1382}-577}{443}\right\}$ with $SQ = 2^{1383} + 1$.

The above methods for constructing RNS with a diagonal function of the 2^n, $2^n - 1$ and $2^n + 1$ forms allow us to develop efficient circuits for comparing numbers and reverse conversion. In the rest of this article, we demonstrate examples of such circuits and show the advantages of their technical characteristics in comparison with the known analogs.

3. Results

The goal of modeling is a comparison of the methods of implementing the numbers comparison operation and reverse RNS to binary conversion by the proposed methods, a method based on CRT [18] and a method based on CRTf [21]. We use $\{3, 5, 14\}$, $\{7, 9, 124\}$ and $\{5, 29, 93, 313\}$ moduli sets, because their DF has form $2^n - 1$ and 2^n which are low-cost RNS [42]. Figure 1 shows the circuit for numbers comparison in RNS with DF of the form $2^n - 1$. The bit-widths of the RNS moduli $\{m_1, m_2, m_3\}$ are denoted as a_1, a_2, a_3. Multipliers by constants $|X_i \cdot k_i|_{2^n-1}, i = 1, 2, 3$ modulo $2^n - 1$ implement the generation of partial products modulo $2^n - 1$. A modulo $2^n - 1$ compressor is implemented as in [21]. Kogge–Stone adder with end-around carry (KSA-EAC) uses for modulo $2^n - 1$ addition, and it is implemented as in [27]. The circuit for numbers comparison in RNS with DF of the form 2^n has a similar structure to that presented in Figure 1, but it should have four inputs for compared numbers X and Y, since in the theoretical part, we demonstrate that only RNS with four modules can have DF of the form 2^n. In addition, compressors and Kogge-Stone adders (KSAs) must implement modulo 2^n operations that are achieved by simply dropping the carrying of the most significant bit (MSB).

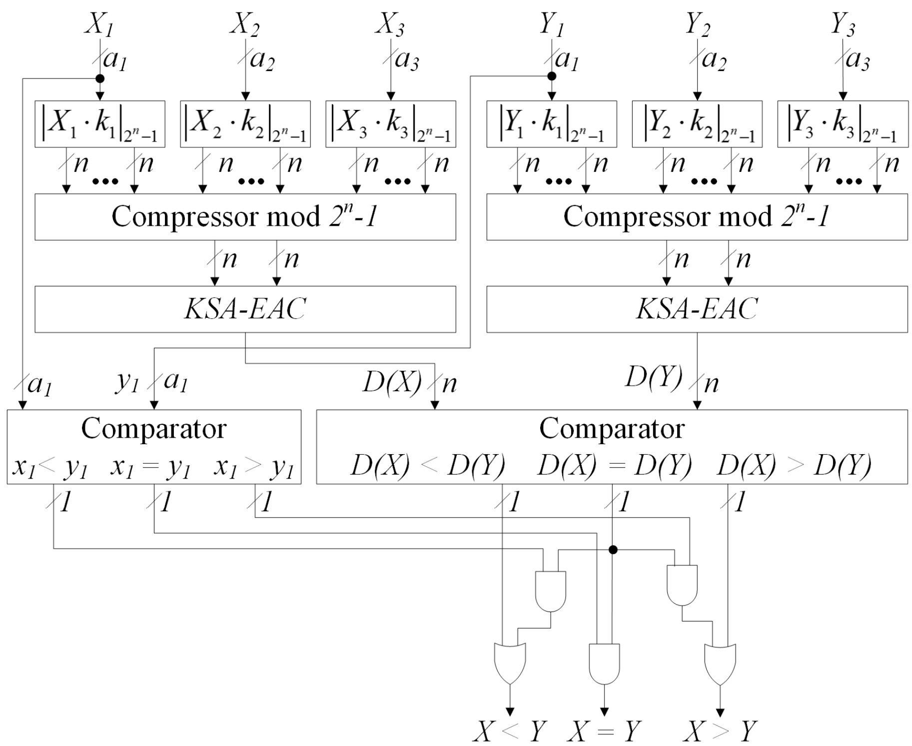

Figure 1. Magnitude comparison circuit using the diagonal function (DF) of the form $2^n - 1$.

Figure 2 shows the reverse conversion circuit for RNS with the DF of the form $2^n - 1$. The bit-widths of the RNS moduli $\{m_1, m_2, m_3\}$ are denoted as a_1, a_2, a_3. Multipliers by constants $|X_i \cdot k_i|_{2^n-1}, i = 1,2,3$ modulo $2^n - 1$, modulo $2^n - 1$ compressor and KSA-EAC blocks are realized as in Figure 1. The rest of the blocks are implemented in standard binary form. The symbol a_R denotes the bit-width of RNS range and symbol a_t denotes the bit-width of the value $M \cdot D(X) + \sum_{i=1}^{n} x_i M_i$. Division by SQ is implemented as multiplication by multiplicative inverse SQ modulo 2^{a_t}. The output of the circuit presented in Figure 2 is a group of a_R most significant bits (MSBs) of the last KSA output. The reverse converter circuit for RNS with a DF of the form 2^n has a similar structure to that presented in Figure 2, with differences similar to the comparator described above.

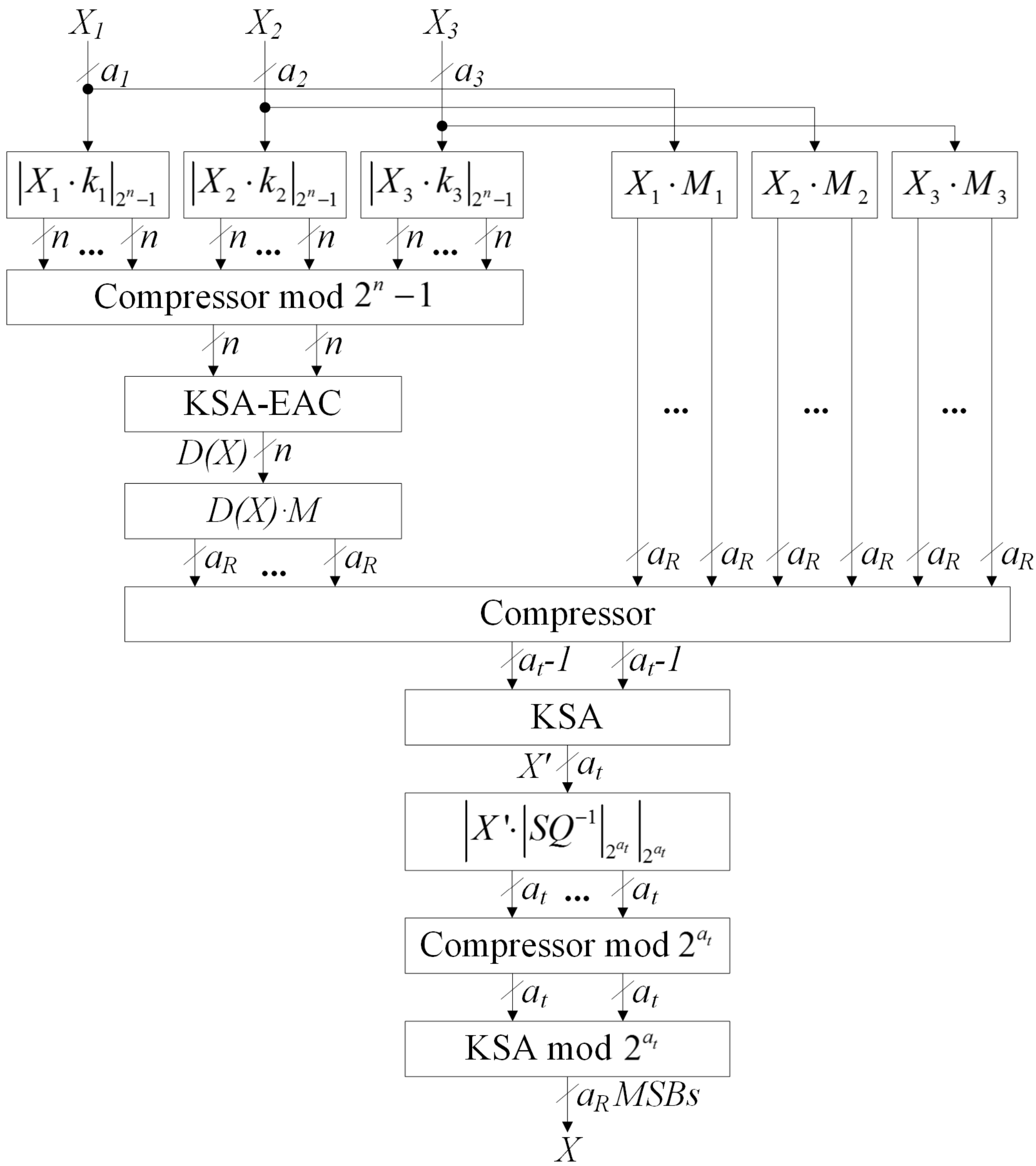

Figure 2. Reverse conversion circuit for the residue number system (RNS) with a DF of the form $2^n - 1$.

Also, modeling was done to compare the proposed moduli sets with balanced RNS moduli sets. The following types of moduli sets were chosen for the simulation: $\{2^n - 1, 2^n, 2^n + 1\}$ [29,30], $\left\{2^n - 1, 2^{n+k}, 2^n + 1\right\}$ [31], $\left\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\right\}$ [34], $\left\{2^n - 1, 2^n + 1, 2^{n\pm1} - 1, 2^{n+k}\right\}$ [35].

All simulated circuits were described in very high speed integrated circuit (VHSIC) hardware description language (VHDL). Hardware modeling was performed on Xilinx Artix 7 xc7a200tfbg484-2 in Vivado 2016.3 and the strategy of synthesis was highly area optimized. The modeling results are presented in Tables 2–4 and show time, hardware costs and the area·delay (A·D) metrics calculated as a product of delay by a number of look up tables (LUTs).

Table 2. Modeling results of the circuit of magnitude comparison.

	Moduli Set	Known Methods		Proposed Method
		CRT [18]	CRTf [21]	
Delay, ns	$\{3, 5, 14\}$	10.961	**7.680**	9.749
	$\{7, 9, 124\}$	16.110	**11.123**	11.830
	$\{5, 29, 93, 313\}$	15.363	**12.611**	11.991
LUTs	$\{3, 5, 14\}$	135	169	**110**
	$\{7, 9, 124\}$	543	329	**273**
	$\{5, 29, 93, 313\}$	1,141	863	**549**
A·D	$\{3, 5, 14\}$	1479	1297	**1072**
	$\{7, 9, 124\}$	8747	3659	**3229**
	$\{5, 29, 93, 313\}$	17529	10883	**6583**
Power, W	$\{3, 5, 14\}$	**4.061**	5.757	4.581
	$\{7, 9, 124\}$	21.751	13.929	**11.840**
	$\{5, 29, 93, 313\}$	40.950	47.128	**24.733**

Table 3. Modeling results of the circuit of reverse RNS to binary conversion.

	Moduli Set	Known Methods		Proposed Method
		CRT [18]	CRTf [21]	
Delay, ns	$\{3, 5, 14\}$	8.181	**8.157**	10.085
	$\{7, 9, 124\}$	15.493	**13.351**	13.531
	$\{5, 29, 93, 313\}$	21.228	**16.814**	17.600
LUTs	$\{3, 5, 14\}$	63	**59**	105
	$\{7, 9, 124\}$	289	358	**285**
	$\{5, 29, 93, 313\}$	997	**920**	1,049
A·D	$\{3, 5, 14\}$	515	**481**	1,058
	$\{7, 9, 124\}$	4,477	4,779	**3,856**
	$\{5, 29, 93, 313\}$	21,164	**15,468**	18,462
Power, W	$\{3, 5, 14\}$	5.946	**5.504**	11.733
	$\{7, 9, 124\}$	**22.226**	39.154	26.789
	$\{5, 29, 93, 313\}$	**65.901**	106.867	117.797

A simulation of magnitude comparison shows that for the $\{3, 5, 14\}$ moduli set, the method using CRTf works 21.22% faster than the proposed method, and 29, 93% faster than the method using CRT. However, the proposed method uses 34.91% fewer hardware resources than CRTf, and 18.52% less than CRT. For $\{7, 9, 124\}$, the circuit, based on CRTf, works 5.98% faster than the circuit, which is based on the proposed method, and 30.96% faster than the circuit which is based on CRT. Furthermore, the circuit, based on the proposed method, uses 17.02% fewer hardware resources than CRTf method and 49.72% less than CRT. For $\{5, 29, 93, 313\}$, the proposed method works 4.92% faster than the method using CRTf, and 21.95% faster than the method using CRT. Moreover, the proposed method uses 36.38% fewer hardware resources than the method using CRTf, and two times fewer resources than the method using CRT. Thus, for the magnitude comparison operation, the proposed method reduces the consumption of hardware resources compared tp known methods. In addition, in the case of using the moduli set $\{5, 29, 93, 313\}$ the proposed method also reduced the delay of the devices. Table 2 also demonstrates the advantages of the proposed method in A·D metrics and power consumption.

Table 4. Modeling results of magnitude comparison and reverse RNS to binary conversion for proposed and balanced moduli sets.

Moduli Set		Ref.	Magnitude Comparison			Reverse Conversion		
			Delay, ns	LUTs	A·D	Delay, ns	LUTs	A·D
$\{2^n-1, 2^n, 2^n+1\}$	$n=3$	[29,30]	12.953	272	3,523	13.486	169	2,279
$\left\{2^n-1, 2^{n+k}, 2^n+1\right\}$	$n=2$, $k=2$	[31]	11.533	150	1,729	11.246	**91**	**1,023**
$\{3, 5, 14\}$,		Proposed	**9.749**	**110**	**1,072**	**10.085**	105	1,058
$\{2^n-1, 2^n, 2^n+1\}$	$n=4$	[29,30]	14.964	275	4,115	15.855	**263**	4,169
$\left\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\right\}$	$n=3$	[34]	16.710	427	7,135	20.217	447	9,036
$\{7, 9, 124\}$		Proposed	**11.830**	**273**	**3,229**	**13.531**	285	**3,856**
$\left\{2^n-1, 2^n+1, 2^{n+1}-1, 2^{n+k}\right\}$	$n=4$, $k=4$	[35]	16.669	**303**	**5,050**	24.163	**572**	**13,821**
$\left\{2^n-1, 2^n+1, 2^{n-1}-1, 2^{n+k}\right\}$	$n=6$, $k=0$	[35]	24.962	1,767	44,107	30.831	1,496	46,123
$\{5, 29, 93, 313\}$		Proposed	**11.991**	549	6,583	**17.600**	1,049	18,462

A simulation of reverse RNS to binary conversion shows that for the $\{3, 5, 14\}$ moduli set method using CRTf works 19.12% faster than the proposed method, and 0, 29% faster than the method using CRT. Moreover, CRTf method uses 43.81% fewer hardware resources than the proposed method and 6.35% less than CRT. For $\{7, 9, 124\}$, circuit, based on CRTf, works 1.33% faster than the circuit based on the proposed method, and 13.82% faster than circuit based on CRT. Furthermore, the circuit based on the proposed method uses 20.39% fewer hardware resources than CRTf method and 1.38% less than CRT. For $\{5, 29, 93, 313\}$, the method using CRTf works 4.47% faster than the proposed method, and 20.79% faster than the method using CRT. Moreover, it uses 12.30% fewer hardware resources than the proposed method and 7.72% fewer resources than the method using CRT. Therefore, the proposed method allows us to reduce hardware resources for the moduli set $\{7, 9, 124\}$ compared to known methods.

For the $\{3, 5, 14\}$ moduli set, the RNS dynamic range is equal to $M = 210$. Due to comparing the performance of the circuit using the proposed moduli set with a circuit using known common moduli sets, two balanced moduli sets were chosen: $\{2^n-1, 2^n, 2^n+1\}$, $n = 3$ [29,30] and $\left\{2^n-1, 2^{n+k}, 2^n+1\right\}$, $n = 2, k = 2$ [26], which are close to this dynamic range. Modeling of the magnitude comparison showed that the circuit using the proposed $\{3, 5, 14\}$ moduli set works 24.73% faster than $\{2^n-1, 2^n, 2^n+1\}$, $n = 3$ moduli set and 15.47% faster than $\left\{2^n-1, 2^{n+k}, 2^n+1\right\}$, $n = 2, k = 2$ moduli set. Moreover, the proposed moduli set uses 2.5 times fewer hardware resources than $\{2^n-1, 2^n, 2^n+1\}$, $n = 3$ moduli set and 26.67% less than $\left\{2^n-1, 2^{n+k}, 2^n+1\right\}$, $n = 2$, $k = 2$ moduli set. Hardware simulation of reverse RNS to binary conversion showed that using the proposed moduli set $\{3, 5, 14\}$ requires 25.22% fewer time costs than the $\{2^n-1, 2^n, 2^n+1\}$, $n = 3$ moduli set and 10.32% less than the $\left\{2^n-1, 2^{n+k}, 2^n+1\right\}$, $n = 2$, $k = 2$ moduli set. Although the proposed moduli set uses 13.33% more hardware resources than $\left\{2^n-1, 2^{n+k}, 2^n+1\right\}$, $n = 2, k = 2$ moduli set, it uses 37.87% less than the $\{2^n-1, 2^n, 2^n+1\}$, $n = 3$ moduli set.

For the proposed $\{7, 9, 124\}$ moduli set, the dynamic RNS range is equal to $M = 7812$. For this dynamic range, two known balanced moduli sets were chosen:$\{2^n-1, 2^n, 2^n+1\}$, $n = 4$ [29,30] and $\left\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\right\}$, $n = 3$ [34]. Modeling of magnitude comparison showed that circuit using proposed $\{7, 9, 124\}$ moduli set works 20.94% faster than the $\{2^n-1, 2^n, 2^n+1\}$, $n = 4$ moduli set and 29.20% faster than the $\left\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\right\}$, $n = 3$ moduli set. In addition, the proposed moduli set uses 0.73% fewer hardware resources than the $\{2^n-1, 2^n, 2^n+1\}$, $n = 4$ moduli set and 36.06% less than the $\left\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\right\}$, $n = 3$ moduli set. Hardware simulation of reverse RNS to binary conversion showed that using the proposed moduli set $\{7, 9, 124\}$ requires 14.66% less time cost than the $\{2^n-1, 2^n, 2^n+1\}$, $n = 4$ moduli set and 33.07% less than the $\{2^n-1, 2^n, 2^n+1\}$, $n = 4$ moduli set.

Although the proposed moduli set uses 7.72% more hardware resources than the $\{2^n-1, 2^n, 2^n+1\}$, $n=4$ moduli set, it uses 36.24% less than the $\{2^n-1, 2^n, 2^n+1\}$, $n=4$ moduli set.

For the proposed $\{5, 29, 93, 313\}$ moduli set, the dynamic range of RNS was equal to $M = 4220805$. For this dynamic range, two known balanced moduli sets were chosen: $\left\{2^n-1, 2^n+1, 2^{n+1}-1, 2^{n+k}\right\}$, $n=4$, $k=4$ and $\left\{2^n-1, 2^n+1, 2^{n-1}-1, 2^{n+k}\right\}$, $n=6$, $k=0$ [33]. Modeling of the magnitude comparison showed that the circuit using the proposed $\{5, 29, 93, 313\}$ moduli set works 28.06% faster than the $\left\{2^n-1, 2^n+1, 2^{n+1}-1, 2^{n+k}\right\}$, $n=4$, $k=4$ moduli set and 2 times faster than the $\left\{2^n-1, 2^n+1, 2^{n-1}-1, 2^{n+k}\right\}$, $n=6$, $k=0$ moduli set. Although the proposed moduli set uses 44.81% more hardware resources than the $\left\{2^n-1, 2^n+1, 2^{n+1}-1, 2^{n+k}\right\}$, $n=4$, $k=4$ moduli set, it uses 3 times less than the $\left\{2^n-1, 2^n+1, 2^{n-1}-1, 2^{n+k}\right\}$, $n=6$, $k=0$ moduli set. Hardware simulation of reverse RNS to binary conversion showed that the using of the proposed moduli set $\{5, 29, 93, 313\}$ requires 27.16% fewer time costs than the $\left\{2^n-1, 2^n+1, 2^{n+1}-1, 2^{n+k}\right\}$, $n=4$, $k=4$ moduli set and 42.91% less than the $\left\{2^n-1, 2^n+1, 2^{n-1}-1, 2^{n+k}\right\}$, $n=6$, $k=0$ moduli set. Although the proposed moduli set uses 45.47% more hardware resources than the $\left\{2^n-1, 2^n+1, 2^{n+1}-1, 2^{n+k}\right\}$, $n=4$, $k=4$ moduli set, it uses 29.88% less than the $\left\{2^n-1, 2^n+1, 2^{n-1}-1, 2^{n+k}\right\}$, $n=6$, $k=0$ moduli set.

Thus, in comparison to known balanced moduli sets, the proposed moduli sets reduce the delay of magnitude comparison and reverse conversion in devices. In case of operation magnitude comparison, using the proposed moduli sets, $\{3, 5, 14\}$ and $\{7, 9, 124\}$, reduces the use of hardware resources in devices.

The experimentally obtained results showed that the approach developed in this paper allows us to improve two problem operations in the RNS: the comparison of numbers and reverse conversion. The proposed devices for such operations can be used in those applications of the RNS for which these operations are the most important, for example, in video processing systems, sorting networks, etc.

4. Discussion

The results obtained in Section 3 are summarized in Table 5. The main conclusion we can assume is that the RNS construction with all cases $SQ = 2^n$, $SQ = 2^n - 1$, $SQ = 2^n + 1$ is principally possible. The cases $SQ = 2^n - 1$ and $SQ = 2^n + 1$ for RNS with one even module are easiest for practical implementation. All cases for RNS with all odd moduli are more complicated. However, among these cases, there is one particularly attractive option. As we have demonstrated, there is the possibility of RNS constructing with $SQ = 2^n$. This case requires the use of four odd RNS moduli.

Table 5. The possibility of RNS constructing with a given sum of quotients (SQ) form.

Type and Number of RNS Moduli		Form of SQ		
		SQ=2^n−1	SQ=2^n	SQ=2^n+1
one even module		exist	not exist	exist
all moduli are odd	3 moduli	exist	not exist	not exist
	4 moduli	not exist	exist	not exist
	5 moduli	not exist	not exist	exist

According to the proposed construction method of RNS with a convenient form of DF, moduli sets with three and four moduli were chosen: $\{3, 5, 14\}$, $\{7, 9, 124\}$ and $\{5, 29, 93, 913\}$. We performed the hardware simulation of magnitude comparison and reverse RNS to binary conversion using RNS with the presented moduli sets and using different approaches to perform the non-modulo comparison operation: the proposed method, method [18], and method [21]. The hardware simulation results of magnitude comparison show that, for three moduli, the use of the proposed method reduces hardware resources, and the use of method [21] reduces circuit delay. For four moduli, the proposed method reduces both time and hardware costs. The modeling of reverse RNS to binary conversion shows that method [21] works faster and requires fewer hardware resources than the others considered methods. Comparison of the simulation results of proposed moduli sets and balanced moduli sets shows that the use of the proposed moduli sets reduces circuit delay, although, in several cases, it required more hardware resources than balanced moduli sets.

5. Conclusions

The paper concerns the problem of RNS construction with convenient forms of DF. We propose several methods of RNS construction with SQ forms 2^n, $2^n - 1$ and $2^n + 1$. The use of these forms of moduli allow for developing efficient methods of hardware implementation. We performed hardware implementation of magnitude comparison and reverse RNS to binary conversion using the proposed method, method [18] method [21]. A comparison of the implementation results shows that using the proposed method is effective for magnitude comparison operation, but for the reverse RNS to binary conversion operation, method [21] performs better modeling results than the proposed method and method [18]. In addition, according to the simulation results, the proposed moduli sets reduce circuit delay in comparison with balanced moduli sets, although, in several cases, require more hardware resources than balanced moduli sets.

The proposed method allows more efficient and problematic operations in RNS, such as sign detection, number comparison, and division, to be performed. It can be used in the development of video processing systems and customized signal processing units using RNS.

Author Contributions: Conceptualization, M.V.; Data curation, N.C.; Formal analysis, D.B.; Investigation, P.L., G.V. and D.K.; Methodology, N.S.; Project administration, P.L.; Resources, D.K. and D.B.; Software, P.L., G.V. and N.S.; Validation, M.V., N.C., P.L. and D.K.; Visualization, G.V., N.S., D.B.; Writing—original draft, M.V. and N.C.; Writing—review & editing, D.K., and P.L.

Funding: The research is supported by the grant of the Russian Science Foundation (Project No. 19-19-00566).

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

References

1. Akkal, M.; Siy, P. A new mixed radix conversion algorithm MRC-II. *J. Syst. Archit.* **2007**, *53*, 577–586. [CrossRef]
2. Ramirez, J.; Garcia, A.; Lopez-Buedo, S.; Lloris, A. RNS-enabled digital signal processor design. *Electron. Lett.* **2002**, *38*, 266–268. [CrossRef]
3. Chang, C.; 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]
4. 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]
5. Esmaeildoust, M.; Schinianakis, D.; Javashi, H.; Stouraitis, T.; Navi, K. Efficient RNS implementation of elliptic curve point multiplication GF(p). *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2013**, *21*, 1545–1549. [CrossRef]
6. Bajard, J.-C.; Imbert, L. A full RNS implementation of RSA. *IEEE Trans. Comput.* **2004**, *53*, 769–774. [CrossRef]
7. Sousa, L.; Antao, S.; Martins, P. Combining residue arithmetic to design efficient cryptographic circuits and systems. *IEEE Circuits Syst. Mag.* **2016**, *16*, 6–32. [CrossRef]
8. Chervyakov, N.I.; Lyakhov, P.A.; Babenko, M.G. Digital filtering of images in a residue number system using finite-field wavelets. *Autom. Control Comput. Sci.* **2014**, *48*, 180–189. [CrossRef]
9. Kar, A.; Sur, K.; Godara, S.; Basak, S.; Mukherjee, D.; Sukla, A.S.; Das, R.; Choudhury, R. Security in cloud storage: An enhanced technique of data storage in cloud using RNS. In Proceedings of the IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 20–22 October 2016; pp. 1–4.
10. Navi, K.; Esmaeildoust, M.; Molahosseini, A.S. A general reverse converter architecture with low complexity and high performance. *IEICE Trans. Inf. Syst.* **2011**, *94*, 264–273. [CrossRef]

11. Milanezi Junior, J.; da Costa, J.P.C.L.; Römer, F.; Miranda, R.K.; Marinho, M.A.M.; Del Galdo, G. M-estimator based Chinese remainder theorem with few remainders using a kroenecker product based mapping vector. *Digit. Signal Process.* **2019**, *87*, 60–74. [CrossRef]
12. Cardarilli, G.C.; Di Nunzio, L.; Fazzolari, R.; Gerardi, L.; Re, M.; Campolo, G.; Cascone, D. A new electric encoder position estimator based on the Chinese Remainder Theorem for the CMG performance improvements. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), New York, NY, USA, 28–31 May 2017; pp. 1–4.
13. Chervyakov, N.I.; Lyakhov, P.A.; Valueva, M.V. Increasing of convolutional neural network performance using residue number system. In Proceedings of the 2017 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), New York, NY, USA, 18–22 September 2017; pp. 135–140.
14. Cardarilli, G.C.; Del Re, A.; Nannarelli, A.; Re, M. Impact of RNS coding overhead on FIR filters performance. In Proceedings of the 2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers, New York, NY, USA, 4–7 November 2007; pp. 1426–1429.
15. Chang, C.; Lee, W.; Liu, Y.; Goi, B.; Phan, R.C.-W. Signature gateway: Offloading signature generation to IoT gateway accelerated by GPU. *IEEE Internet Things J.* **2019**, *6*, 4448–4461. [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. Selvam, R.; Tyagi, A. Power side channel resistance of RNS secure logic. In Proceedings of the 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), New York, NY, USA, 6–10 January 2018; pp. 143–148.
18. Mohan, P.V.A. *Residue Number Systems: Theory and Applications*; Birkhauser: Basel, Switzerland, 2016; ISBN 9783319413853.
19. Gonnella, J. The application of core functions to residue number system. *IEEE Trans. Signal Process.* **1991**, *39*, 69–75. [CrossRef]
20. Akushskii, I.J.; Burcev, V.M.; Pak, I.T. A new positional characteristic of nonpositional codes and its applications. *Coding Theory Optim. Complex Syst.* **1977**, 8–16.
21. Matos, R.; Paludo, R.; Chervyakov, N.; Lyakhov, P.A.; Pettenghi, H. Efficient implementation of modular multiplication by constants applied to RNS reverse converters. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
22. Dimauro, G.; Impedovo, S.; Pirlo, G. A new technique for fast number comparison in the residue number system. *IEEE Trans. Comput.* **1993**, *42*, 608–612. [CrossRef]
23. Dimauro, G.; Impedovo, S.; Pirlo, G.; Salzo, A. RNS architectures for the implementation of the 'diagonal function'. *Inf. Process. Lett.* **2000**, *73*, 189–198. [CrossRef]
24. Mohan, P.V.A. RNS to binary conversion using diagonal function and pirlo and impedovo monotonic function. *Circuits Syst. Signal Process.* **2015**, *35*, 1–14. [CrossRef]
25. Piestrak, S.J. A note on RNS architectures for the implementation of the diagonal function. *Inf. Process. Lett.* **2015**, *115*, 453–457. [CrossRef]
26. Kalampoukas, L.; Nikolos, D.; Efstathiou, C.; Vergos, H.T.; Kalamatianos, J. High-speed parallel-prefix modulo $2^n - 1$ Adders. *IEEE Trans. Comput.* **2000**, *49*, 673–680. [CrossRef]
27. Efstathiou, C.; Vergos, H.T.; Nikolos, D. Fast parallel-prefix Modulo $2^n + 1$ Adders. *IEEE Trans. Comput.* **2004**, *53*, 1211–1216. [CrossRef]
28. Vergos, H.T.; Dimitrakopoulos, G. On Modulo $2^n + 1$ Adder design. *IEEE Trans. Comput.* **2012**, *61*, 173–186. [CrossRef]
29. Chaves, R.; Sousa, L. Improving residue number system multiplication with more balanced moduli sets and enhanced modular arithmetic structures. *IET Comput. Digit. Tech.* **2007**, *1*, 472. [CrossRef]
30. Jaberipur, G.; Nejati, S. Balanced minimal latency RNS addition for moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. In Proceedings of the 18th International Conference on Systems, Signals and Image Processing, Sarajevo, Bosnia and Herzegovina, 16–18 June 2011; pp. 1–7.
31. 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]
32. 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]

33. Hiasat, A. A reverse converter and sign detectors for an extended RNS five-moduli set. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *64*, 111–121. [CrossRef]
34. Mohan, P.V.A.; Premkumar, A.B. RNS-to-Binary converters for two four-moduli sets $\{2^n + 1, 2^n - 1, 2^n, 2^{n-1} - 1\}$ and $\{2^n + 1, 2^n - 1, 2^n, 2^{n+1} + 1\}$. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2007**, *54*, 1245–1254. [CrossRef]
35. Kumar, S.; Chang, C.-H.; Tay, T.F. New algorithm for signed integer comparison in $\{2^{n+k}, 2^n - 1, 2^n + 1, 2^{n\pm1} - 1\}$ and its efficient hardware implementation. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *64*, 1481–1493. [CrossRef]
36. Kumar, S.; Chang, C.-H. A scaling-assisted signed integer comparator for the balanced five-moduli set RNS $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1, 2^{n-1} - 1\}$. *IEEE Trans. Very Large Scale Integr. Syst.* **2017**, *25*, 3521–3533. [CrossRef]
37. Skavantzos, A.; Abdallah, M.; Stouraitis, T.; Schinianakis, D. Design of a balanced 8-modulus RNS. In Proceedings of the 2009 16th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2009), New York, NY, USA, 13–16 December 2009; pp. 61–64.
38. 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.* **2019**, *29*, 572–581. [CrossRef]
39. Cheng, S.W. A high-speed magnitude comparator with small transistor count. In Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems, 2003 (ICECS 2003), New York, NY, USA, 14–17 December 2003; pp. 1168–1171.
40. Chervyakov, N.I.; Lyakhov, P.A.; Kalita, D.I.; Shulzhenko, K.S. Effect of RNS dynamic range on grayscale images filtering. In Proceedings of the XV International Symposium Problems of Redundancy in Information and Control Systems (REDUNDANCY), St. Petersburg, Russia, 26–29 September 2016; pp. 33–37.
41. Molahosseini, A.S.; Sorouri, S.; Zarandi, A.A.E. Research challenges in next-generation residue number system architectures. In Proceedings of the IEEE 7th International Conference on Computer Science & Education (ICCSE), Melbourne, VIC, Australia, 14–17 July 2012; pp. 1658–1661.
42. Parhami, B. *Computer Arithmetic: Algorithms and Hardware Designs*; Oxford University Press: Oxford, UK, 2010; ISBN 9780195328486.