Designing reverse converter for data transmission systems from two-level RNS to BNS

M V Bergerman, P A Lyakhov, A S Voznesensky, D V Bogaevskiy and D I Kaplun

1Department of Applied Mathematics and Mathematical Modeling, Institute of mathematics of Natural Sciences, North Caucasus Federal University, ul. Pushkina, 1 355009 Stavropol, Russia
2Department of Automation and Control Processes, Saint Petersburg Electrotechnical University “LETI”, ul. Professora Popova, 5 197376 Saint Petersburg, Russia

E-mail: maxx07051997@inbox.ru

Abstract. In this paper, we present a reverse conversion from a two-stage Residue Number System (RNS) to a Binary Number System (BNS) with a special set of level 1 modules \(\{2^{\alpha_1}, 2^{\alpha_2} - 1, \ldots, 2^{\alpha_n} - 1\}\) and level 2 modules \(\{2^{\beta_1}, 2^{\beta_2} - 1, \ldots, 2^{\beta_k} - 1\}\). The proposed method is based on the Chinese Remainder Theorem (CRT) with fractions and using a calculation method that uses constant multiplications to speed up calculations. This article discusses the simulation of FPGA reverse conversion to a two-stage RNS using the proposed method and the standard CRT using adders, their comparison of latency and hardware costs.

1. Introduction

High-tech computer architectures that use parallelism are needed to provide high-speed data transfer and information processing for computer programs. This architecture can be built using Residue Number System (RNS).

RNS is a non-positional number system that divides numbers into residuals and performs arithmetic operations in parallel for each remainder, which leads to faster calculations over existing binary arithmetic. RNS is well suited for applications where the bulk of computing is done by addition, subtraction, and multiplication.

In [1], [2] and [3], transformations from RNS to BNS are considered using special sets of modules to speed up calculations and reduce hardware costs.

We suggest using a two-stage RNS for data transfer, since the introduction of such a RNS allows you to convert data to a small-bit format, which can be effective for low-power systems, for example, for MANET networks [4].

In this paper, we will consider the reverse conversion of a two-stage RNS to BNS. Forward conversion to a two-stage RNS was discussed in [5].

2. Residue number system

In the RNS the numbers are represented in the basis of mutually prime numbers, called moduli \(\beta = \{m_1, \ldots, m_n\}\), \(GCD(m_i, m_j) = 1\), for \(i \neq j\). The product of all modules RNS \(M = \prod_{i=1}^{n} m_i\) call the
dynamic range (DR) of the system. Any integer $0 \leq X < M$ can be uniquely represented in RNS as a vector $\{x_1, x_2, \ldots, x_n\}$, where $x_i = |X|_{m_i} = X mod m_i$.

The operations of addition, subtraction and multiplication in the RNS are determined by the formulas

$$A \pm B = ( |a_1 \pm b_1|_{m_1}, \ldots, |a_n \pm b_n|_{m_n} ),$$  \hspace{1cm} (1)

$$A \times B = ( |a_1 \times b_1|_{m_1}, \ldots, |a_n \times b_n|_{m_n} ).$$  \hspace{1cm} (2)

The conversion of data to and from a two-stage RNS is shown in Figure 1. Next, let's look at the right part of the figure, which shows the reverse conversion from a two-stage RNS to a BNS. One way to do this is to use the Chinese remainder theorem (CRT).

![Figure 1. Converting information using a two-stage RNS.](image)

### 3. Chinese remainder theorem

Calculating the positional representation of a number $X$ in the BNS from its representation in RNS, that is, from its remainder $(x_1, x_2, \ldots, x_n)$ by moduli $\{m_1, \ldots, m_n\}$, can be performed by the formula:

$$X = \left( \sum_{i=1}^{n} |M_i|^{-1}_{m_i} M_i x_i \right)_{M},$$  \hspace{1cm} (3)

where $M_i = \frac{M}{m_i}$ and $|M_i|^{-1}_{m_i}$ is the multiplicative inverse element $M_i$ modulo $m_i$, $i = 1, 2, \ldots, n$. The direct application of formula (1) requires calculating the remainder of the division by the number $M$ with the full range of digits.

As a matter of fact, the RNS-to-BNS conversion is one of the most complex operations in terms of hardware implementation. This conversion is performed using the corollaries of the CRT [6].

Since we suggest using level 1 moduli $\{2^{a_1}, 2^{a_2} - 1, \ldots, 2^{a_n} - 1\}$ and level 2 moduli $\{2^{b_1}, 2^{b_2} - 1, \ldots, 2^{b_k} - 1\}$ in a two-level RNS, it is very effective to use constant multiplications to speed up calculations.

### 4. Method for implementing calculation with multiplication by a constant

For multiplying by a constant, partial products can be compressed according to the value of the constant. The use of compression techniques reduces the number of additions [7].

Consider the use of compression techniques for multiplying by a constant modulo $2^a$ and $2^a - 1$. Multiplication by a constant is reduced to addition and bit-shift operations. Formed partial products
equal to 0 can be excluded from further calculations. Therefore, when a constant contains more bits equal to 1 than equal to 0, you need to use the inversion of this constant. This results in the inversion of the second multiplier and the addition of a single term called the correction factor $\Delta_{COR}$.

Consider multiplying a number $X = \sum_{i=0}^{n-1} x_i 2^i$, $g < \alpha$ by a constant $C = \sum_{i=0}^{n-1} c_i 2^i$, $f < \alpha$. For calculations on modulo $2^\alpha$, negative numbers are represented in additional code, that is

$$|(−X) \cdot (−C)|_{2^\alpha} = |\bar{X} \cdot (\bar{C} + 1 - 2f)|_{2^\alpha} + \Delta_{COR}|_{2^\alpha},$$

where the correction factor is calculated as follows:

$$\Delta_{COR} = |(1 - 2^g) \cdot (\bar{C} + 1 - 2f)|_{2^\alpha}. \tag{5}$$

For calculations on modulo $2^\alpha - 1$, negative numbers are represented in the reverse code, i.e.

$$|(−X) \cdot (−C)|_{2^\alpha-1} = |\bar{X} \cdot (\bar{C} + 1 - 2f) + \Delta_{COR}|_{2^\alpha-1}, \tag{6}$$

where the correction coefficient is calculated using the formula:

$$\Delta_{COR} = |(1 - 2^g) \cdot (\bar{C} + 1 - 2f)|_{2^{\alpha-1}}. \tag{7}$$

This way, summands are formed, which are then added using the CSA tree. In the case calculations of modulus $2^\alpha - 1$, the upper carry bit is clipped in the adders, and in the case calculations of modulus $2^\alpha$, the upper carry bit is cycled in the adders.

5. Converting from RNS to BNS

The goal of most reverse transformation studies is to simplify this operation, reducing calculations modulo $M$ to simpler operations. One of these approaches is the CRT modification, which consists in estimating the position of a number on the number line without complex arithmetic operations.

The algorithm yields the most significant bits starting from the $(N + 1)$-th one. Thus,
During the hardware implementation, the division operation in (13) is ignored, since the most significant bits starting from the $(N + 1)$-th one are actually outputted. In the software implementation, this operation is equivalent to the right N-bit shift.

Thus, the RNS-to-BNS conversion algorithm based on formulas (11) and (13) allows eliminating the calculation of the mod $M$ remainder of the size dynamic range by replacing it with multiplication, which is a simpler operation in terms of hardware implementation [8].

Figure 2 shows the architecture of the hardware implementation of the reverse conversion RNS-to-BNS algorithm using CSA and KSA trees, and we will also consider an example of this conversion.

**Figure 2.** Architecture of the reverse conversion RNS-to-BNS algorithm based on a CRTf.
Example 1. Let the level 1 RNS be given by a set of modules \( m_1 = 32, m_2 = 31, m_3 = 15, m_4 = 7 \) and level 2 RNS \( q_1 = 7, q_2 = 3, q_3 = 2 \). We calculate DR of 2 level \( M_0 = 7 \cdot 3 \cdot 2 = 42, \mu = -3 + 7 + 3 + 2 = 9 \). According to the formula (12), we calculate \( N_0 = \lfloor \log_2 (42 \cdot 9) - 1 \rfloor = 8 \). For calculations with integers using the formula (10), we calculate the constants \( k_1 = 220, k_2 = 171, k_3 = 128 \).

Let \( X = ((1,1,0), (1,2,1), (6,0,0), (1,1,1)) \) be a certain number represented in the RNS with the specified modules \( \{32, 31, 15, 7\} = \{7,3,2\}, \{7,3,2\}, \{7,3,2\}, \{7,3,2\} \). Let's restore the positional representation of the first-level modules according to the method used. Then, according to the formula (11),

\[
\overline{Q}_1 = |220 \cdot 1 + 171 \cdot 1 + 128 \cdot 0|_{2^8} = 135, \quad \overline{Q}_2 = |220 \cdot 1 + 171 \cdot 2 + 128 \cdot 1|_{2^8} = 178, \overline{Q}_3 = |220 \cdot 6 + 171 \cdot 0 + 128 \cdot 1|_{2^8} = 40, \quad \overline{Q}_4 = |220 \cdot 1 + 171 \cdot 1 + 128 \cdot 1|_{2^8} = 7.
\]

The positional characteristic \( \overline{Q}_1, \overline{Q}_2, \overline{Q}_3, \overline{Q}_4 \) can be used for restoring the desired number with formula (13):

\[
Q_1 = \lfloor \frac{135 - 42}{2^8} \rfloor = 22, \quad Q_2 = \lfloor \frac{178 - 42}{2^8} \rfloor = 29, \quad Q_3 = \lfloor \frac{40 - 42}{2^8} \rfloor = 6, \quad Q_4 = \lfloor \frac{7 - 42}{2^8} \rfloor = 1.
\]

Similarly, we find the number \( X \). Find \( M_M = 32 \cdot 31 \cdot 15 \cdot 7 = 104160, \mu = -4 + 32 + 31 + 15 + 7 = 81, N_M = \lfloor \log_2 (104160 \cdot 81) - 1 \rfloor = 23 \) and \( k_1 = 1835008, k_2 = 3517804, k_3 = 7829368, \overline{k}_4 = 3595118 \).

Find \( \overline{X} \) and further:

\[
\overline{X} = |22 \cdot 1835008 + 29 \cdot 3517804 + 6 \cdot 7829368 + 1 \cdot 3595118|_{2^{23}} = 19834,
\]

\[
X = \lfloor \frac{19834 \cdot 104160}{2^{23}} \rfloor = 256.
\]

Let's check the solution:

\[
(256 \mod 32, 256 \mod 31, 256 \mod 15, 256 \mod 7) = (22, 29, 6, 1) \equiv ((22 \mod 7, 22 \mod 3, 22 \mod 2), (29 \mod 7, 29 \mod 3, 29 \mod 2), (6 \mod 7, 6 \mod 3, 6 \mod 2), (1 \mod 7, 1 \mod 3, 1 \mod 2)) = ((1,1,0), (1,2,1), (6,0,0), (1,1,1)) - \text{answer match.}
\]

### 6. Modeling

For the simulation, we selected reverse conversion devices from a two-stage RNS-to-BNS with 16 and 32-bit bit sizes. From 3 to 5 modules were selected for the main system (table 1). In the new RNS, the range must be greater than the maximum module in the main RNS.

| Range, bits | Number of modules | Modules \( m_i \) | Modules \( \mu_i \) |
|-------------|-------------------|-------------------|-------------------|
| 16          | 3                 | \((2^6, 2^6 - 1, 2^5 - 1)\) | \((2^3 - 1, 2^2 - 1, 2^2 - 1)\) |
|             | 4                 | \((2^5, 2^5 - 1, 2^4 - 1, 2^3 - 1)\) | \((2^3 - 1, 2^2 - 1, 2^2 - 1)\) |
| 32          | 3                 | \((2^{12}, 2^{11} - 1, 2^{10} - 1)\) | \((2^5, 2^5, 2^3 - 1, 2^2 - 1)\) |
|             | 4                 | \((2^9, 2^9 - 1, 2^8 - 1, 2^7 - 1)\) | \((2^4 - 1, 2^3, 2^3 - 1)\) |
|             | 5                 | \((2^{10}, 2^{10} - 1, 2^9 - 1, 2^8 - 1, 2^7 - 1, 2^5 - 1, 2^3 - 1)\) | \((2^5 - 1, 2^3, 2^3 - 1)\) |

Hardware modeling was carried out in the environment of Xilinx ISE Design Suite 14.7 with the use of the VHDL language. Devices with 16- and 32-bit ranges were implemented on the Xilinx xc7k480t-3ffg901 board. The simulation results are presented in table 2.

| Range, bits | CRT | Proposed method |
|-------------|-----|-----------------|
| 16          |     |                 |
| 32          |     |                 |
Summarizing the results, we can conclude that the proposed method for fast calculation and the lowest hardware costs wins in all parameters. If in numbers, then at 16 bits for delay, the proposed method is on average 2 times faster than using CRT, and 2 times faster for hardware costs. At 32 bits, our method is also faster by 2 times than CRT, and loses by hardware costs – up to 40%.

7. Conclusion
The paper demonstrates the reverse conversion from two-stage RNS to BNS in data transmission systems. The results of hardware modeling showed that using modules of the form $m_3$ for the main RNS and modules of the form $q_k$ for the new RNS allows you to get a significant gain in the speed of calculation. More specifically, the results of modeling on the 16- and 32-bit ranges showed that the use of the proposed method gives a gain in speed, on average, by 2 times, depending on the bit rate. In hardware costs, when the bit rate increases, the proposed method will lose.

Acknowledgments
This work was supported by the Russian Science Foundation, project no. 19-19-00566.

References
[1] Hiasat A 2017 An Efficient Reverse Converter for the Three-Moduli Set (2n+1 − 1, 2n, 2n − 1) IEEE Transactions on Circuits and Systems II: Express Briefs 64 8 962–966
[2] Hiasat A 2019 A Reverse Converter for Three-Moduli Set (2^k, 2^n−1, 2^n+1) IEEE Jordan Int. Jt. Conf. Electr. Eng. Inf. Technol. 548–553
[3] Pettenghi H, De Matos R and Molahosseini A 2016 RNS reverse converters for moduli sets with dynamic ranges of 9n-bit VII Latin American Symposium on Circuits and Systems (LASCAS) 143–146
[4] Doss S, Nayyar A, Suseendran et al. 2018 APD-JFAD: Accurate Prevention and Detection of Jelly Fish Attack in MANET IEEE Access 6 56954–56965
[5] Bergerman M V, Lyakhov P A, Chervyakov N I, Kaplun D I and Bogaevskiy D V 2019 Designing forward converters for data transmission systems in two-level RNS Journal of Physics: Conference Series 1352 012005
[6] Omondi A and Premkumar B 2007 Residue Number Systems: Theory and Implementation (Imperial College Press) p 296
[7] Chervyakov N I, Lyakhov P A and Valueva M V 2017 Increasing of Convolutional Neural Network Performance Using Residue Number System International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON) 135–140
[8] Valueva M V, Nagornov N N, Lyakhov P A, Valuev G V and Chervyakov N I 2020 Application of the residue number system to reduce hardware costs of the convolutional neural network implementation Mathematics and Computers in Simulation 177 232–243