DOI: 10.15514/ISPRAS-2025-37(3)-7

Исследование специальных наборов модулей системы остаточных классов

[1] *В.В. Луценко, ORCID: 0000-0003-4648-8286 <vvlutcenko@ncfu.ru>*
[2] *М.Д. Кравцов, ORCID: 0009-0001-5098-5034 <micha_kravtsov_olymp@mail.ru>*
[1] *Д.Е. Горлачев, ORCID: 0009-0002-9703-1876 <dimagorlacev12@gmail.com>*
[1] *Н.М. Мирный, ORCID: 0009-0005-2476-494X <mirnyjn@gmail.com>*

[1] *Северо-Кавказский федеральный университет,*
Россия, 355017, г. Ставрополь, ул. Пушкина, д. 1.

[2] *Региональный центр «Сириус 26»,*
Россия, 355017, г. Ставрополь, ул. Артема, д. 6.

Аннотация. В статье рассматриваются современные подходы к повышению производительности вычислительных систем на основе системы остаточных классов. Цель исследования – анализ специальных наборов модулей системы остаточных классов, которые позволяют проводить ключевые вычислительные операции, такие как сложение, обратное преобразование и определение знака, с минимальными затратами. Экспериментальные результаты показали, что базис $\{\mathbf{2^n - 1}, \mathbf{2^n}, \mathbf{2^n + 1}\}$ оказался наиболее эффективным среди трех модульных наборов. Данный базис является перспективным для использования в высокопроизводительных вычислительных системах.

Ключевые слова: система остаточных классов; специальные наборы модулей; Китайская теорема об остатках; функция ядра Акушского.

Для цитирования: Луценко В.В., Кравцов М.Д., Горлачев Д.Е., Мирный Н.М. Исследование специальных наборов модулей системы остаточных классов. Труды ИСП РАН, том 37, вып. 3, 2025 г., стр. 107–120. DOI: 10.15514/ISPRAS–2025–37(3)–7.

Благодарности: Работа выполнена при поддержке Российского научного фонда (проект № 22-71-10046).

Research of Special Sets of Moduli of the Residue Number System

[1] *V.V. Lutsenko, ORCID: 0000-0003-4648-8286 <vvlutcenko@ncfu.ru>*
[2] *M.D. Kravtsov, ORCID: 0009-0001-5098-5034 <micha_kravtsov_olymp@mail.ru>*
[1] *D.E. Gorlacev, ORCID: 0009-0002-9703-1876 < dimagorlacev12@gmail.com >*
[1] *N.M. Mirny, ORCID: 0009-0005-2476-494X < mirnyjn@gmail.com >*

[1] *North-Caucasus Federal University, Stavropol,*
1, Pushkin st., Stavropol, 355017, Russia

[2] *Sirius 26 Regional Centre,*
6, Artema st., Stavropol, 355017, Russia.

Abstract. The article examines modern approaches to enhancing the performance of computing systems based on the residue number system. The objective of the study is to analyze specific sets of residue number system moduli that allow for key computational operations, such as addition, reverse conversion, and sign determination, to be performed with minimal cost. Experimental results showed that the $\{\mathbf{2^n - 1}, \mathbf{2^n}, \mathbf{2^n + 1}\}$ basis was the most efficient among the three moduli sets. This basis is promising for use in high-performance computing systems.

Keywords: residue number system; special sets of moduli; Chinese remainder theorem; Akushsky core functions; non-modular operations.

For citation: Lutsenko V.V., Kravtsov M.D., Gorlacev D.E., Mirny N.M. Research of special sets of moduli of the residue number system. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 3, 2025. pp. 107-120 (in Russian). DOI: 10.15514/ISPRAS-2025-37(3)-7.

Acknowledgements. The research was supported by the Russian Science Foundation Grant No. 22-71-10046, https://rscf.ru/en/project/22-71-10046/.

1. Введение

Развитие алгоритмической базы вычислительных систем требует разработки и внедрения инновационных подходов к организации и выполнению вычислительных задач. Наряду с такими современными технологиями, как квантовые вычисления [1] и ИИ-ускорители [2], параллельная обработка данных остается одной из ключевых стратегий повышения производительности. В данной области основное внимание уделяется модулярным вычислительным структурам, основанным на системе остаточных чисел (СОК). В отличие от традиционных методов, СОК предлагает особые преимущества благодаря своей непозиционной природе и параллельным свойствам, вытекающим из теории чисел и абстрактной алгебры.

Известно, что СОК имеет ряд преимуществ перед обычными позиционными системами счисления при разработке и реализации высокопроизводительных вычислительных приложений, устройств и систем [3]. С момента своего появления в середине 1950-х годов и по сей день арифметика СОК привлекает постоянное внимание исследователей в следующих областях:

1) компьютерные технологии [3],
2) теоретико-числовые методы [4, 5],
3) блокчейн [6],
4) гомоморфное шифрование [7],
5) цифровая обработка сигналов и изображений [8, 9],
6) системы связи [10],
7) высоконадежные облачные среды [11, 12],
8) нейронные сети [13, 14].

Основным преимуществом СОК является возможность разложения чисел большой длины на набор остатков меньшей длины, которые обрабатываются параллельно в независимых модульных каналах. Параллелизм, присущий СОК, позволяет избежать переносов, получаемых при сложении, вычитании и умножении. Данные операции также называют модульными.

Тем не менее, существуют определенные трудности, связанные с применением СОК, которые требуют дополнительного внимания для расширения ее применимости в различных областях, например, в процессорах общего назначения [10]. Наиболее ресурсозатратными операциями в СОК являются операции обратного преобразования, определение знака, сравнения и деления чисел. Данные операции также называют немодульными.

Первоочередная задача при задании СОК – определить набор модулей системы или же, другими словами, базис системы. За более чем полувековой период существования СОК было предложено множество различных наборов модулей [15]. Для классических систем эффективным набор модулей считается набор в форме степеней двойки, который может значительно снизить сложность операций, описанных ранее. Правильный выбор модулей способствует снижению вычислительных затрат и более эффективной реализации алгоритмов на практике.

В данной статье рассмотрены наиболее известные наборы модулей СОК. В частности, уделяется внимание таким критериям, как время выполнения сложения в СОК, а также время обратного преобразования и определения знака числа. Исследование направлено на нахождение оптимальных наборов модулей для различных приложений, что открывает возможности для повышения эффективности современных вычислительных систем.

Статья имеет следующую структуру. В разделе 2 рассматриваются алгоритмы основных модульных и немодульных операций системы остаточных классов. В разделе 3 представлены наборы модулей специального вида. Затем, в разделе 4 проведен анализ их эффективности с использованием алгоритмов из раздела 2. В заключении суммируются полученные результаты.

2. Система остаточных классов

2.1 Основы арифметики в системе остаточных классов

Базисом системы остаточных классов называется множество модулей $\{p_1, p_2, \dots, p_n\}$, где каждый модуль $p_i \geq 2\,(i = 1, 2, \dots, n)$ и $\text{НОД}(p_i, p_j) = 1$. По умолчанию модули упорядочены по возрастанию, то есть $p_1 < p_2 < \dots < p_n$.

Произведение модулей $P = \prod_{i=1}^{n} p_i$ называется динамическим диапазоном системы остаточных классов.

Целое число $X \in [0, P)$ представимо в виде n-мерного вектора, составленного из наименьшего неотрицательный остатка от деления соответствующего числа на p_i:

$$X = (x_1, x_2, \dots, x_n),$$

где $x_i = X \bmod p_i$, что также обозначается как $x_i = |X|_{p_i}$.

Для введения отрицательных чисел необходимо разделить диапазон на два интервала. Пусть P – динамически диапазон системы. Тогда при введении отрицательных чисел число X удовлетворяет следующему соотношению:

$$\begin{cases} \dfrac{-P-1}{2} \leq X \leq \dfrac{P-1}{2}, \text{если } P \text{ нечетное число}, \\ \dfrac{-P}{2} \leq X \leq \dfrac{P}{2} - 1, \text{если } P \text{ четное число}. \end{cases} \quad (1)$$

Рассмотрим СОК с базисом $\{3,4\}$. В этом базисе можно взаимно-однозначно представить числа из полуинтервала $[-6;6)$, так как $P = 3 \cdot 4 = 12$. Если $X = (x_1, x_2, \dots, x_n)$, то отрицательное число $-X = (\overline{x_1}, \overline{x_2}, \dots, \overline{x_n})$, где $\overline{x_i}$ является дополнением x_i до модуля p_i. Для СОК $\{3,4\}$ и числа $X = (1,1)$ получим $-X = (3-1, 4-1) = (2,3)$.

В табл. 1 представлены соответствия чисел из позиционной системы счисления и системы остаточных классов.

Табл. 1. Представление чисел для СОК с базисом $\{3,4\}$.
Table 1. Number representation for RNS with basis $\{3,4\}$.

$-6 \overset{\text{СОК}}{\longrightarrow} (0,2)$	$-5 \overset{\text{СОК}}{\longrightarrow} (1,3)$	$-4 \overset{\text{СОК}}{\longrightarrow} (2,0)$	$-3 \overset{\text{СОК}}{\longrightarrow} (0,1)$
$-2 \overset{\text{СОК}}{\longrightarrow} (1,2)$	$-1 \overset{\text{СОК}}{\longrightarrow} (2,3)$	$0 \overset{\text{СОК}}{\longrightarrow} (0,0)$	$1 \overset{\text{СОК}}{\longrightarrow} (1,1)$
$2 \overset{\text{СОК}}{\longrightarrow} (2,2)$	$3 \overset{\text{СОК}}{\longrightarrow} (0,3)$	$4 \overset{\text{СОК}}{\longrightarrow} (1,0)$	$5 \overset{\text{СОК}}{\longrightarrow} (2,1)$

Предположим, что два числа X и B представлены как $X = (x_1, x_2, \dots, x_n)$ и $B = (b_1, b_2, \dots, b_n)$ в СОК. Используя $\circ \in \{+, -, \times\}$, мы можем выразить арифметику в СОК следующим образом:

$$X \circ B = (r_1, r_2, \dots, r_n), \tag{2}$$

где

$$r_i = |x_i \circ b_i|_{p_i}.$$

В контексте модульных операций, если результат вычитания x_i из b_i отрицательный, то значение r_i определяется как:

$$r_i = p_i + (x_i - b_i).$$

Алгоритм 1 представляет метод выполнения модульных операций в СОК. В примере 1 представлено сложение двух чисел в СОК.

Пример 1 (Сложение в СОК). Сложим два числа $X = -2$ и $B = 3$ в базисе $\{3,4\}$. Их представление в заданном базисе указано в таблице 1. Воспользуемся (2) для сложения:

$$X + B = (|1+0|_3, |2+3|_4) = (1,1).$$

Далее рассмотрим процесс обратного преобразования числа из СОК в позиционную систему счисления.

Алгоритм 1. Выполнение модульных операций в СОК
Input: $\{p_1, p_2, \dots, p_n\}, (x_1, x_2, \dots, x_n), (b_1, b_2, \dots, b_n), \circ \in \{+, -, \times\}$
Output: $(r_1, r_2, \dots, r_n)$
1. **for** $i = 1, i \le n, i++$ **do**:
1.2 $r_i = |x_i \circ b_i|_{p_i}$.
2. **return** $(r_1, r_2, \dots, r_n)$

Алгоритм 1. Метод выполнения модульных операций в СОК.
Algorithm 1. A method for performing modular operations in RNS.

2.2 Китайская теорема об остатках

Если число X задается в виде остатков $x_1, x_2, \dots, x_n$ от деления по модулям $p_1, p_2, \dots, p_n$, число X можно восстановить на основе Китайской теоремы об остатках (КТО) [9]:

$$X = \left| \sum_{i=1}^{n} P_i \cdot x_i \cdot |P_i^{-1}|_{p_i} \right|_P, \tag{3}$$

где $P_i = \frac{P}{p_i}$ и $|P_i^{-1}|_{p_i}$ представляет собой мультипликативную инверсию P_i по модулю p_i и удовлетворяет следующему соотношению $\left||P_i^{-1}|_{p_i} \cdot P_i\right|_{p_i} = 1$.

Алгоритм 2 предназначен для восстановления числа из СОК в позиционную систему счисления.

Алгоритм 2. Обратное преобразование числа с помощью КТО

Input: $\{p_1, p_2, \dots, p_n\}, (x_1, x_2, \dots, x_n), P, P_i, |P_i^{-1}|_{p_i}$ for $i = \overline{1,n}$

Output: X

1. $sums = 0$
2. **for** $i \leq n, i++$ **do**:
2.1 $sums \mathrel{+}= P_i \cdot x_i \cdot |P_i^{-1}|_{p_i}$
3. $X = |sums|_P$
4. **return** X

Алгоритм 2. Обратное преобразование числа с помощью КТО.
Algorithm 2. Inverse number conversion with CRT.

Рассмотрим пример 2 для иллюстрации восстановления числа с помощью КТО.

Пример 2 (Обратное преобразование числа с помощью КТО). Возьмем СОК с таким же набором модулей $\{3, 4\}$. Преобразуем число $X = (2, 1)$ в позиционную систему счисления. Для этого, найдем значения P_i:

$$P_1 = \frac{P}{p_1} = 4, P_2 = \frac{P}{p_2} = 3.$$

Затем вычислим мультипликативные инверсии $|P_i^{-1}|_{p_i}$:

$$|P_1^{-1}|_{p_1} = 1, |P_2^{-1}|_{p_2} = 3.$$

Имея эти значения, можно вычислить значение X, используя (3) получим:

$$X = |4 \cdot 1 \cdot 2 + 3 \cdot 3 \cdot 1|_{12} = 5.$$

Далее рассмотрим алгоритм определения знака числа в СОК.

2.2 Функция ядра Акушского

С целью снижения вычислительной сложности при определении знака числа в СОК за счет определения позиционных характеристик, в работе [16] была разработана новая функция, известная как функция ядра Акушского (ФЯА). Функция определяется следующим уравнением:

$$C(X) = \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{X}{p_i} \right\rfloor. \quad (4)$$

Функция ядра Акушского может эффективно применяться для определения знака [17], масштабирования [18], общего деления [19], обратного преобразования [20], коррекции ошибок [21].

Веса w_i из уравнения (4) представляют собой константы, определяемые выбором точки интерполяции, они определяют каждую конкретную основную функцию и могут варьироваться в зависимости от конкретной задачи. Более того, веса w_i могут быть в определенной степени произвольными. Алгоритм определения оптимальных весов для ФЯА представлен в [22].

Диапазон функции ядра вычисляется следующим образом:

$$C(P) = C_P = \sum_{i=1}^{n} w_i \cdot P_i \,. \tag{5}$$

Определим так называемые ортогональные базисы:

$$B_i = P_i \cdot |P_i^{-1}|_{p_i}, \tag{6}$$

Тогда функция ядра для ортогональных базисов будет определена как

$$C(B_i) = B_i \cdot \frac{C(P)}{P} - \frac{w_i}{p_i}. \tag{7}$$

Имея число в СОК, а также функции ядра от ортогональных базисов, функция ядра может быть определена следующим образом:

$$C(X) = \left| \sum_{i=1}^{n} x_i \cdot C(B_i) \right|_{C_P} . \tag{8}$$

Используя ФЯА, можно определить знак числа без преобразования числа в позиционную систему счисления. Основываясь на выражениях (1) и (3), получим:

$$\begin{cases} \begin{cases} C(X) < C\left(\frac{P-1}{2}\right) \text{ если } X \text{ положительное} \\ C(X) \geq C\left(\frac{P-1}{2}\right) \text{ если } X \text{ отрицательное} \end{cases} \text{для нечетного } P, \\ \begin{cases} C(X) < C\left(\frac{P}{2}\right) \text{ если } X \text{ положительное} \\ C(X) \geq C\left(\frac{P}{2}\right) \text{ если } X \text{ отрицательное} \end{cases} \text{для четного } P. \end{cases} \tag{9}$$

Алгоритм 3 предназначен для определения знака с использованием функции ядра.

Алгоритм 3. Определение знака числа с помощью функции ядра Акушского

Input: $\{p_1, p_2, \dots, p_n\}, (x_1, x_2, \dots, x_n), P, C\left(\left\lfloor \frac{P}{2} \right\rfloor\right), C(B_i)$ for $i = \overline{1,n}$

Output: S

1. $sum = 0$
2. **for** $i \leq n, i++$ **do**:
2.1 $sum \mathrel{+}= x_i \cdot C(B_i)$
3. $C(X) = |sum|_{C_P}$
4. **if** $C(X) < C\left(\left\lfloor \frac{P}{2} \right\rfloor\right)$:
4.1 $S = 0$
5. **else**:
5.1 $S = 1$
6. **return** S

Алгоритм 3. Определение знака числа с помощью функции ядра Акушского.
Algorithm 3. Determining the sign of a number using the Akushsky core function.

Рассмотрим пример определения знака числа.

Пример 3 (Определение знака числа с помощью функции ядра Акушского). Исследуем СОК с базисом $\{3,4\}$ и весами $w_1 = 1, w_2 = 0$. Предрасчитанные параметры останутся такими же $P = 12, \frac{P}{2} = 6, P_1 = 4, P_2 = 3$ и $\left|{P_1}^{-1}\right|_{p_1} = 1, \left|{P_2}^{-1}\right|_{p_2} = 3$.

Во-первых, найдем B_i:

$$B_1 = P_1 \cdot \left|{P_1}^{-1}\right|_{p_1} = 4, B_2 = P_2 \cdot \left|{P_2}^{-1}\right|_{p_2} = 9.$$

Во-вторых, вычислим $C(P)$ и $C\left(\frac{P}{2}\right)$:

$$C(P) = 1 \cdot 4 + 0 \cdot 3 = 4,$$

$$C\left(\frac{P}{2}\right) = 1 \cdot \left\lfloor\frac{6}{3}\right\rfloor + 0 \cdot \left\lfloor\frac{6}{4}\right\rfloor = 2.$$

Затем найдем $C(B_i)$ используя формулу (7):

$$C(B_1) = 4 \cdot \frac{4}{12} - \frac{1}{3} = 1, C(B_2) = 9 \cdot \frac{4}{12} + \frac{0}{4} = 3.$$

Имея эти значения, можем определить знак числа $X = (2,3)$, используя формулу (15):

$$C(X) = |2 \cdot 1 + 3 \cdot 3|_4 = |11|_4 = 3.$$

Так как $3 > C\left(\frac{P}{2}\right) = 2$, тогда $X = (2,3)$ это отрицательное число, что истинно, так как $X = -1 < 0$.

3. Специальные наборы модулей системы остаточных классов

Выбор набора модулей очень важен для достижения подходящей реализации СОК. Общие наборы модулей, содержащие произвольные числа, неэффективны для реализации, поэтому предпочтительны специальные наборы модулей СОК с популярными модулями вида $2^k - 1, 2^k, 2^k + 1$ [23]. В табл. 2 представлены наиболее распространённые наборы модулей специального вида и их характеристики.

Далее представлен сравнительный анализ наборов модулей из табл. 2. Для набора под номер 25 были подобраны параметры так, чтобы по количество модулей было равно пяти.

4. Сравнительный анализ

Эффективность наборов модулей из табл. 2 была проверена для трех алгоритмов:

- Сложение чисел в СОК (алгоритм 1).
- Обратное преобразование числа с помощью КТО (алгоритм 2).
- Определение знака числа с помощью ФЯА (алгоритм 3).

Алгоритмы реализованы на языке Python. Эксперимент проводился с использованием операционной системы Windows 10 на компьютере с процессором AMD Ryzen 5 3600, оперативной памятью DDR4 16 ГБ 3200 МГц и SSD 512 ГБ.

Для каждого набора модулей (табл. 3) было произведено 10000 запусков по каждой из рассматриваемых операций и замерено среднее время исполнения алгоритмов. Средняя погрешность для операции модульного сложения составила 0,00015 секунд, для алгоритма обратного преобразования 0,0018 и для определения знака числа 0,0015. Результаты представлены в табл. 4.

Первым шагов приведем глобальный анализ результатов. А затем сравним специальные наборы в рамках количества модулей в одном наборе.

Табл.2. Наборы модулей специального вида.
Table 2. Sets of special type moduli.

Номер набора	Ссылка	Набор модулей	Год	Характеристика
1	[23]	$\{2^n-1, 2^n, 2^n+1\}$	1967	Удобное обратное преобразование
2	[24]	$\{2n-1, 2n, 2n+1\}$	1995	Неэффективный
3	[25]	$\{2^{2n}+1, 2^n+1, 2^n-1\}$	1997	Удобное обратное преобразование
4	[26]	$\{2^n-1, 2^n, 2^{n-1}-1\}$	1998	Удобный для арифметики
5	[27]	$\{2^n-1, 2^n, 2^{n+1}-1\}$	1999	Удобный для арифметики
6	[28]	$\{2^n-1, 2^n, 2^{2n+1}-1\}$	2008	Удобный для арифметики
7	[29]	$\{2^n-1, 2^n, 2^{2n}+1\}$	2008	Удобное обратное преобразование
8	[30]	$\{2^\alpha, 2^\beta-1, 2^\beta+1\}$	2008	Гибкий, удобное обратное преобразование
9	[31]	$\{3^n-2, 3^n-1, 3^n\}$	2007	Удобный для арифметики
10	[32]	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$	1999	Удобный для арифметики
11	[33]	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$	2000	Удобный для арифметики
12	[34]	$\{2^n-1, 2^n, 2^n+1, 2^{2n}-1\}$	2003	Удобное обратное преобразование
13	[35]	$\{2^n-1, 2^n+1, 2^n-3, 2^n+3\}$	2004	Сбалансированные модули
14	[36]	$\{2^n-1, 2^n+1, 2^{2n}-2, 2^{2n+1}-3\}$	2008	Большой динамический диапазон
15	[37]	$\{2^n-1, 2^n+1, 2^n, 2^{2n}+1\}$	2009	Удобное обратное преобразование
16	[37]	$\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$	2009	Удобный для арифметики
17	[37]	$\{2^n-1, 2^n+1, 2^{2n}, 2^{2n+1}-1\}$	2009	Большой динамический диапазон, удобный для арифметических вычислений
18	[38]	$\{2^k, 2^n-1, 2^n+1, 2^{n+1}+1\}$	2014	Удобный для арифметики
19	[38]	$\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$	2014	Удобный для арифметики
20	[39]	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$	2005	Удобное преобразование
21	[40]	$\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1, 2^{n+1}-1\}$	2007	Сбалансированный, удобный для арифметики
22	[41]	$\{2^{n/2}-1, 2^n, 2^{n/2}+1, 2^n+1, 2^{2n-1}-1\}$	2009	Несбалансированный
23	[42]	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	2012	Очень большой динамический диапазон, высокий параллелизм
24	[42]	$\{2^n-1, 2^{n+\beta}, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	2012	Гибкий, очень большой динамический диапазон
25	[43]	$\{2^{n+\beta}, 2^n-1, 2^n+1, 2^n-k_1, 2^n+k_1, \dots, 2^n-k_f, 2^n+k_f\}$	2018	Очень большой динамический диапазон, сбалансированные модули

Табл. 3. Наборы модулей, используемые для исследования (Table 3. Moduli sets used for the research).

Номер набора	Размер динамического диапазона, бит			
	8	16	24	32
1	{7, 8, 9}	{63, 64, 65}	{255, 256, 257}	{2047, 2048, 2049}
2	{7, 8, 9}	{43, 44, 45}	{257, 258, 259}	{1629, 1630, 1631}
3	{65, 9, 7}	{257, 17, 15}	{4097, 65, 63}	{65537, 257, 255}
4	{15, 16, 7}	{63, 64, 31}	{511, 512, 255}	{2047, 2048, 1023}
5	{7, 8, 15}	{63, 64, 127}	{255, 256, 511}	{2047, 2048, 4095}
6	{3, 4, 31}	{15, 16, 511}	{63, 64, 8191}	{255, 256, 131071}
7	{7, 8, 65}	{31, 32, 1025}	{127, 128, 16385}	{511, 512, 262145}
8	{16, 7, 9}	{128, 31, 33}	{1024, 127, 129}	{16385, 511, 513}
9	{7, 8, 9}	{79, 80, 81}	{727,728, 729}	{2185, 2186, 2187}
10	{3, 4, 5, 9}	{15, 16, 17, 33}	{63, 64, 65, 129}	{255, 256, 257, 513}
11	{3, 4, 5, 7}	{15, 16, 17, 31}	{63, 64, 65, 127}	{255, 256, 257, 511}
12	{3, 4,5, 17}	{15, 16, 17, 257}	{31, 32, 33, 1025}	{127, 128, 129, 16385}
13	{7, 9, 5, 11}	{31, 33, 29, 35}	{63, 65, 61, 67}	{255, 257, 253, 259}
14	{3, 5, 6, 29}	{7, 9, 10, 125}	{31, 33,18, 2045}	{127, 129, 263, 2765}
15	{3, 5, 8, 9}	{15, 17, 16, 17}	{127, 129, 28, 29}	{2047, 2049, 44, 45}
16	{3, 4, 5, 31}	{15, 16, 17, 511}	{31, 32, 33, 2047}	{127, 128, 129, 32767}
17	{3,5,16,31}	{7, 9, 64, 127}	{15, 17, 256, 511}	{63, 65, 4096, 8191}
18	{4,3,5,9}	{8, 15, 17, 33}	{4, 127, 129, 257}	{16, 511, 513, 1025}
19	{2, 7,9, 5}	{4, 31, 33, 17}	{2, 255, 257, 129}	{8, 1023, 1025, 513}
20	{7, 8,9, 5, 13}	{31, 32, 33, 25, 41}	{127, 128, 129, 113, 145}	{511, 512, 513, 481, 545}
21	{3, 4, 5, 1, 7}	{15, 16, 17, 7, 31}	{31, 32, 33, 15, 63}	{127, 128, 129, 63, 255}
22	{3, 16, 5, 17, 127}	{7, 64, 9, 65, 2047}	{7, 64, 9, 65, 2047}	{15, 256, 17, 257, 32767}
23	{7, 8, 9, 5, 13, 15}	{31, 32, 33, 25, 41, 63}	{31, 32, 33, 25, 41, 65}	{127, 128, 129, 113, 145, 255}
24	{7, 8, 9, 5, 13, 5}	{7, 16, 9, 5, 13, 5}	{31, 32, 33, 25, 41, 17}	{31, 256, 33, 25, 41, 17}
25	{8, 7, 9, 5, 11}	{16, 15, 17, 13, 19}	{32, 31, 33, 29, 35}	{128, 127, 129, 125, 131}

Табл. 4. Результаты исследования, время в секундах (Table 4. Research results, time in seconds).

Номер набора	Размер динамического диапазона, бит											
	8			16			24			32		
	Сложение	КТО	ФЯА	Сложение	КТО	ФЯА	Сложение	КТО	ФЯА	Сложение	КТО	ФЯА
1	1,9E-03	0,2813	0,2837	1,8E-03	0,2758	0,2803	1,8E-03	0,2776	0,2830	1,8E-03	0,2807	0,8182
2	1,8E-03	0,2810	0,2878	1,8E-03	0,2760	0,2793	1,8E-03	0,2768	0,2812	1,9E-03	0,2793	0,8157
3	1,8E-03	0,2795	0,2839	1,8E-03	9,2758	0,2810	1,8E-03	0,2765	0,2793	1,8E-03	0,2801	0,8123
4	1,9E-03	0,2798	0,2826	1,8E-03	0,2759	0,2805	1,8E-03	0,2762	0,2800	1,9E-03	0,2794	0,2866
5	1,8E-03	0,2795	0,2827	1,8E-03	0,2752	0,2769	1,8E-03	0,2756	0,2798	1,9E-03	0,2803	0,8157
6	1,9E-03	0,2804	0,2843	1,8E-03	0,2766	0,2801	1,8E-03	0,2765	0,2805	1,8E-03	0,2814	0,8162
7	1,8E-03	0,2862	0,2929	1,9E-03	0,2854	0,2894	1,8E-03	0,2880	0,2893	1,9E-03	0,2865	0,8326
8	1,8E-03	0,2798	0,2832	1,8E-03	0,2767	0,2791	1,8E-03	0,2761	0,2796	1,8E-03	0,2782	0,8159
9	1,9E-03	0,2893	0,2983	1,8E-03	0,2846	0,2949	1,9E-03	0,2825	0,2862	1,9E-03	0,2829	0,8173
10	2,4E-03	0,3365	0,3388	2,3E-03	0,3300	0,3347	2,3E-03	0,3342	0,3335	2,4E-03	0,3349	1,043
11	2,3E-03	0,3377	0,3406	2,3E-03	0,3308	0,3352	2,3E-03	0,3305	0,3342	2,3E-03	0,3369	1,053
12	2,3E-03	0,3367	0,3389	2,3E-03	0,3309	0,3361	2,3E-03	0,3313	0,3350	2,3E-03	0,3379	1,049
13	2,3E-03	0,3350	0,3391	2,3E-03	0,3310	0,3339	2,3E-03	0,3316	0,3372	2,4E-03	0,3360	0,3423
14	2,4E-03	0,3355	0,3376	2,3E-03	0,3294	0,3352	2,3E-03	0,3310	0,3323	2,3E-03	0,3383	1,057
15	2,3E-03	0,3375	0,3373	2,3E-03	0,3331	0,3345	2,3E-03	0,3320	0,3347	2,3E-03	0,3350	1,045
16	2,3E-03	0,3362	0,3385	2,3E-03	0,3311	0,3357	2,3E-03	0,3317	0,3340	2,4E-03	0,3370	1,048
17	2,4E-03	0,3353	0,3393	2,2E-03	0,3303	0,3320	2,4E-03	0,3308	0,3352	2,3E-03	0,3388	1,046
18	2,3E-03	0,3369	0,3385	2,4E-03	0,3315	0,3335	2,3E-03	0,3324	0,3337	2,3E-03	0,3381	1,047
19	2,3E-03	0,3363	0,3396	2,3E-03	0,3309	0,3367	2,3E-03	0,3303	0,3326	2,4E-03	0,3404	1,057
20	2,8E-03	0,3941	0,3967	2,8E-03	0,3860	0,3905	2,8E-03	0,3908	1,265	2,8E-03	0,3939	1,281
21	2,9E-03	0,3938	0,3977	2,8E-03	0,3863	0,3900	2,8E-03	0,3867	0,3875	2,8E-03	0,3940	1,294
22	2,9E-03	0,3920	0,3957	2,8E-03	0,3894	0,3922	2,8E-03	0,3885	0,3891	2,8E-03	0,3920	1,281
23	3,3E-03	0,4487	0,4502	3,3E-03	0,4429	0,4432	3,3E-03	0,4451	0,4468	3,3E-03	0,4494	1,521
24	3,3E-03	0,4483	0,4522	3,3E-03	0,4413	0,4434	3,3E-03	0,4436	0,4437	3,4E-03	0,4492	1,504
25	2,8E-03	0,3929	0,3963	2,8E-03	0,3856	0,3907	2,8E-03	0,3879	0,3875	2,9E-03	0,3928	1,287

При выполнении операции модульного сложения для различных наборов данных были получены результаты, демонстрирующие высокую степень сходства между собой. Наиболее эффективными для сложения, оказались наборы модулей под номерами 1-9. Рассмотрим результаты сравнения для немодульных операций.

Результаты обратного преобразования с использованием КТО:

Среди 8, 16 и 24 битных, лучший результат показал набор модулей $\{2^n - 1, 2^n, 2^{n+1} - 1\}$, при этом:

- Среди 8 битных наборов он эффективнее набора $\{2^n - 1, 2^n, 2^{n-1} - 1\}$ на 0,1% и в среднем эффективнее остальных базисов на 17%.
- Среди 16 битных наборов он эффективнее набора $\{2^{2n} + 1, 2^n + 1, 2^n - 1\}$ на 0,2% и в среднем эффективнее остальных базисов на 17,1%.
- Среди 24 битных наборов он эффективнее набора $\{2^\alpha, 2^\beta - 1, 2^\beta + 1\}$ на 0,18% и в среднем эффективнее остальных наборов на 17,2%

Среди 32 битных лучший результат у набора $\{2^\alpha, 2^\beta - 1, 2^\beta + 1\}$, при этом он быстрее набора модулей $\{2^n - 1, 2^n, 2^n + 1\}$ на 0,32% и в среднем эффективнее остальных наборов на 17,5%.

Результаты в определении знака с использованием ФЯА:

- Для 8 битных базисов лучший результат у набора $\{2^n - 1, 2^n, 2^{n-1} - 1\}$, при этом он эффективнее набора $\{2^n - 1, 2^n, 2^{n+1} - 1\}$ на 0,03% и в среднем эффективнее остальных наборов на 17%.
- Среди 16 битных базисов лучший результат у набора $\{2^n - 1, 2^n, 2^{n+1} - 1\}$, при этом он лучше второго по оптимальности набора $\{2^\alpha, 2^\beta - 1, 2^\beta + 1\}$ на 0,79% и в среднем эффективнее остальных наборов на 17,5%.
- Среди 24 битных базисов лучший результат у набора $\{2^{2n} + 1, 2^n + 1, 2^n - 1\}$, при этом он эффективнее набора $\{2^\alpha, 2^\beta - 1, 2^\beta + 1\}$ на 0,1% и в среднем эффективнее показателей остальных наборов на 24,8%.
- Среди 32 битных базисов лучший результат результаты у набора $\{2^n - 1, 2^n, 2^{n-1} - 1\}$, он эффективнее набора $\{2^n - 1, 2^n + 1, 2^n - 3, 2^n + 3\}$ на 19,43% и в среднем эффективнее остальных наборов на 71,9%.

Для оценки эффективности выполнения операций в различных разрядностях наборов данных были вычислены суммы временных затрат для каждого типа набора, в рамках каждой операции. Полученные результаты представлены в табл. 5, которая позволяет провести сравнительный анализ и сформировать итоговый рейтинг производительности.

Исходя из данных приведенных в табл. 5 можно сделать следующие выводы.

Для трёх модульных базисов:

- В сложении лучшие результаты показали наборы $\{2^\alpha, 2^\beta - 1, 2^\beta + 1\}$ и $\{2^{2n} + 1, 2^n + 1, 2^n - 1\}$**,** они в среднем быстрее других трех модульных наборов на 1.69%.
- В операции обратного преобразования лучший результат показал набор $\{2^n - 1, 2^n, 2^{n+1} - 1\}$**,** он быстрее других трех модульных наборов на 0.78%.
- В определении знака с лучший результат показал набор $\{2^n - 1, 2^n, 2^n + 1\}$**,** он быстрее остальных трех модульных наборов на 42.51%.

Для четырех модульных базисов:

- В сложении лучшие результаты показали наборы $\{2^n - 1, 2^n + 1, 2^n, 2^{2n} + 1\}$, $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} - 1\}$ и $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$**,** они в среднем быстрее других четырех модульных наборов на 0.87%.
- В операции обратного преобразования лучший результат показал набор $\{2^n - 1, 2^n + 1, 2^n - 3, 2^n + 3\}$**,** он быстрее других четырёх модульных наборов на 0.19%.

- В определении знака с лучший результат показал набор $\{2^n-1, 2^n+1, 2^n-3, 2^n+3\}$, он быстрее других четырех модульных наборов на 46.87%.

Для пяти модульных базисов:

- В сложении лучшие результаты показали наборы $\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$, они в среднем быстрее других пяти модульных наборов на 6.54%.
- В операции обратного преобразования лучший результат показал набор $\{2^{n+\beta}, 2^n-1, 2^n+1, 2^n-k_1, 2^n+k_1, \dots, 2^n-k_f, 2^n+k_f\}$, он быстрее других пяти модульных наборов на 4.91%.
- В определении знака с лучший результат показал набор $\{2^{n/2}-1, 2^n, 2^{n/2}+1, 2^n+1, 2^{2n-1}-1\}$, он быстрее других пяти модульных наборов на 11.38%.

Табл.5. Рейтинг наборов модулей специального вида.
Table 5. Ranking of special type moduli sets.

Количество модулей	Место	Модули	Сумма времени, сек.		
			Сложение	КТО	ФЯА
3	1	$\{2^n-1, 2^n, 2^n+1\}$	0,0074	1,1113	**1,1297**
	2	$\{2^n-1, 2^n, 2^{n+1}-1\}$	0,0073	**1,1106**	1,6551
	3	$\{2^{2n}+1, 2^n+1, 2^n-1\}$	**0,0072**	1,1119	1,6565
	4	$\{2^\alpha, 2^\beta-1, 2^\beta+1\}$	**0,0072**	1,1108	1,6578
	5	$\{2^n-1, 2^n, 2^{2n+1}-1\}$	0,0073	1,1149	1,6611
	6	$\{2n-1, 2n, 2n+1\}$	0,0073	1,1131	1,6640
	7	$\{2^n-1, 2^n, 2^n+1\}$	0,0073	1,1154	1,6652
	8	$\{3^n-2, 3^n-1, 3^n\}$	0,0075	1,1393	1,6967
	9	$\{2^n-1, 2^n, 2^{2n}+1\}$	0,0074	1,1461	1,7042
4	1	$\{2^n-1, 2^n+1, 2^n-3, 2^n+3\}$	0,0093	**1,3336**	**1,3525**
	2	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$	0,0094	1,3356	2,0504
	3	$\{2^n-1, 2^n+1, 2^{2n}, 2^{2n+1}-1\}$	0,0093	1,3352	2,0528
	4	$\{2^n-1, 2^n+1, 2^n, 2^{2n}+1\}$	**0,0092**	1,3376	2,0511
	5	$\{2^k, 2^n-1, 2^n+1, 2^{n+1}+1\}$	0,0093	1,3389	2,0528
	6	$\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$	0,0093	1,3360	2,0560
	7	$\{2^n-1, 2^n, 2^n+1, 2^{2n}-1\}$	**0,0092**	1,3368	2,0591
	8	$\{2^n-1, 2^n+1, 2^{2n}-2, 2^{2n+1}-3\}$	0,0093	1,3342	2,0616
	9	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$	**0,0092**	1,3359	2,0625
	10	$\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$	0,0093	1,3379	2,0662
5	1	$\{2^{n+\beta}, 2^n-1, 2^n+1, 2^n-k_1, 2^n+k_1, \dots, 2^n-k_f, 2^n+k_f\}$	0,0113	1,5619	**2,4577**
	2	$\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1, 2^{n+1}-1\}$	0,0113	**1,5592**	2,4613
	3	$\{2^n-1, 2^{n+\beta}, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	0,0113	1,5608	2,4690
	4	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	0,0133	1,7824	2,8432
	5	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$	0,0132	1,7861	2,8612
	6	$\{2^{n+\beta}, 2^n-1, 2^n+1, 2^n-k_1, 2^n+k_1, \dots, 2^n-k_f, 2^n+k_f\}$	**0,0112**	1,5648	3,3329

5. Заключение

В данной статье были исследованы известные наборы модулей СОК и их производительность по ключевым операциям: сложение, обратное преобразование и определение знака числа. Результаты показывают, что эффективность различных наборов может сильно меняться при

выполнении немодульных операций, что оказывает значительное влияние на производительность, особенно в случае более высоких разрядностей.

Экспериментальные результаты показали, что базис $\{2^n - 1, 2^n, 2^n + 1\}$ оказался наиболее эффективным среди трех модульных наборов. Это делает его оптимальным выбором для современных вычислительных систем. Дальнейшие исследования будут направлены на алгоритм генерации модулей, способный задавать набор очень большой размерности, но при этом сохранять эффективность.

Список литературы / References

[1]. Horowitz M., Grumbling E. Quantum Computing: Progress and Prospects. 2019.

[2]. Reuther A., Michaleas P., Jones M., Gadepally V., Samsi S., Kepner J. AI Accelerator Survey and Trends. In Proceedings of the 2021 IEEE High Performance Extreme Computing Conference (HPEC); IEEE, 2021; pp. 1–9.

[3]. Mohan P.V.A. Residue Number Systems; Springer International Publishing: Cham, 2016, ISBN 978-3-319-41383-9.

[4]. Амербаев В. М. Теоретические основы машинной арифметики. АлмаАта: Наука, 1976. 324 с. / Amerbayev V.M. Theoretical Foundations of Machine Arithmetic. Alma-Ata, Science 1976, 324 (in Russian).

[5]. Krasnobayev V., Kuznetsov A., Yanko A., Kuznetsova T. The Analysis of the Methods of Data Diagnostic in a Residue Number System. Computer Modeling and Intelligent Systems 2020, 2608, 594–609, doi:10.32782/cmis/2608-46.

[6]. Guo Z., Gao Z., Mei H., Zhao M., Yang J. Design and Optimization for Storage Mechanism of the Public Blockchain Based on Redundant Residual Number System. IEEE Access 2019, vol. 7, pp. 98546–98554, doi:10.1109/ACCESS.2019.2930125.

[7]. Su Y., Yang B.-L., Yang C., Zhao S.-Y. ReMCA: A Reconfigurable Multi-Core Architecture for Full RNS Variant of BFV Homomorphic Evaluation. IEEE Transactions on Circuits and Systems I: Regular Papers 2022, vol. 69, pp. 2857–2870, doi:10.1109/TCSI.2022.3163970.

[8]. Cardarilli G.C., Nunzio L.D., Fazzolari R., Nannarelli A., Petricca M., Re M. Design Space Exploration Based Methodology for Residue Number System Digital Filters Implementation. IEEE Transactions on Emerging Topics in Computing 2022, vol. 10, pp. 186–198, doi:10.1109/TETC.2020.2997067.

[9]. 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. International Journal of Computer Mathematics 2017, vol. 94, pp. 1833–1849, doi:10.1080/00207160.2016.1247439.

[10]. Omondi A.R., Premkumar A.B. Residue Number Systems: Theory And Implementation; World Scientific, 2007, ISBN 978-1-908979-11-7.

[11]. Chervyakov N., Babenko M., Tchernykh A., Kucherov N., Miranda-López V., Cortés-Mendoza J.M. AR-RRNS: Configurable Reliable Distributed Data Storage Systems for Internet of Things to Ensure Security. Future Generation Computer Systems 2019, vol. 92, pp. 1080–1092, doi:10.1016/j.future.2017.09.061.

[12]. Lutsenko V., Zgonnikov M. Fault Tolerant System for Data Storage, Transmission and Processing in Fog Computing Using Artificial Neural Networks. In Current Problems of Applied Mathematics and Computer Systems. Lecture Notes in Networks and Systems. Springer Nature Switzerland: Cham, 2024, vol. 1044, pp. 199–212, ISBN 978-3-031-64009-4.

[13]. Valueva M.V., Nagornov N.N., Lyakhov P.A., Valuev G.V., Chervyakov N.I. Application of the Residue Number System to Reduce Hardware Costs of the Convolutional Neural Network Implementation. Mathematics and Computers in Simulation 2020, vol. 177, pp. 232–243, doi:10.1016/j.matcom.2020.04.031.

[14]. Roohi A., Taheri M., Angizi S., Fan D. RNSiM: Efficient Deep Neural Network Accelerator Using Residue Number Systems. In Proceedings of the 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), November 2021, pp. 1–9.

[15]. Molahosseini A.S., Navi K. Study of the Reverse Converters for the Large Dynamic Range Four-Moduli Sets; InTech Press, 2011.

[16]. Акушский И.Я., Юдицкий Д.И. Машинная арифметика в остаточных классах. М., Советское радио, 1968, 440 с. / Akushsky I. Ya., Yuditsky D. I. Computer arithmetic in residual classes. Moscow, Soviet Radio, 1968, 440 p. (in Russian).

[17]. Shiriaev E., Kucherov N., Babenko M., Nazarov A. Fast Operation of Determining the Sign of a Number in RNS Using the Akushsky Core Function. Computation, 2023, vol. 11, issue 124, doi:10.3390/computation11070124.

[18]. Burgess N. Scaling an RNS Number Using the Core Function. In Proceedings of the 16th IEEE Symposium on Computer Arithmetic, 2003. Proceedings.; IEEE Comput. Soc: Santiago de Compostela, Spain, 2003, pp. 262–269.

[19]. Луценко В.В., Бабенко М.Г., Черных А.Н., Лапина М.А. Оптимизация алгоритма деления чисел в системе остаточных классов на основе функции ядра Акушского. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 157–168. / Lutsenko V.V., Babenko M.G., Tchernykh A.N., Lapina, M.A. Optimization of a Number Division Algorithm in the Residue Number System Based on the Akushsky Core Function. Proceedings of the Institute for System Programming of the RAS, 2023, vol. 35, issue 5, pp. 157–168. (in Russian), DOI: 10.15514/ISPRAS–2023–35(5)–11.

[20]. Lutsenko V., Babenko M. High Speed Method of Conversion Numbers from Residue Number System to Positional Notation.

[21]. Lutsenko V., Zgonnikov M. Investigation of Neural Network Methods for Error Detection and Correction in the Residue Number System. In AISMA-2024: International Workshop on Advanced Information Security Management and Applications. Lecture Notes in Networks and Systems; Springer Nature Switzerland: Cham, 2024, vol. 863, pp. 194–206 ISBN 978-3-031-72170-0.

[22]. Shiriaev E., Kucherov N., Babenko M., Lutsenko V., Al-Galda S. Algorithm for Determining the Optimal Weights for the Akushsky Core Function with an Approximate Rank. Applied Sciences, 2023, vol. 13, 10495.

[23]. Szabo N.S., Tanaka R.I. Residue Arithmetic and Its Applications to Computer Technology, 1967.

[24]. Premkumar A.B. An RNS to Binary Converter in a Three Moduli Set with Common Factors. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 1995, vol. 42, pp. 298–301.

[25]. Pourbigharaz F. A Signed-Digit Architecture for Residue to Binary Transformation. IEEE Transactions on Computers, 1997, vol. 46, pp. 1146–1150.

[26]. Hiasat A.A., Abdel-Aty-Zohdy S.H. Residue-to-Binary Arithmetic Converter for the Moduli Set $2^k, 2^k - 1, 2^{k-1} - 1$. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 1998, 45, 204–209.

[27]. Mathew J., Radhakrishnan D., Srikanthan T. Fast Residue-to-Binary Converter Architectures. In Proceedings of the 42nd Midwest Symposium on Circuits and Systems (Cat. No. 99CH36356). IEEE, 1999; vol. 2, pp. 1090–1093.

[28]. Molahosseini A.S., Navi K., Rafsanjani M.K. A New Residue to Binary Converter Based on Mixed-Radix Conversion. In Proceedings of the 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008, pp. 1–6.

[29]. Hariri A., Navi K., Rastegar R. A New High Dynamic Range Moduli Set with Efficient Reverse Converter. Computers & mathematics with applications 2008, vol. 55, pp. 660–668.

[30]. Molahosseini A.S., Navi K., Hashemipour O., Jalali A. An Efficient Architecture for Designing Reverse Converters Based on a General Three-Moduli Set. journal of Systems Architecture, 2008, vol. 54, pp. 929–934.

[31]. Hosseinzad M., Navi K. A New Moduli Set for Residue Number System in Ternary Valued Logic. Journal of Applied Sciences, 2007, vol. 7, pp. 3729–3735.

[32]. Bhardwaj M., Srikanthan T., Clarke C.T., Center M.D., Microelectronics S. Reverse Converter for the 4-Moduli Superset $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$. In Proceedings of the Proceedings of the IEEE Conference on Computer Arithmetic; Citeseer, 1999, vol. 14, pp. 168–175.

[33]. Vinod A.P., Premkumar A.B. A memoryless reverse converter for the 4-moduli superset $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$. J circuit syst comp, 2000, vol. 10, pp. 85–99, doi:10.1142/S0218126600000044.

[34]. Cao B., Chang C.-H., Srikanthan T. An Efficient Reverse Converter for the 4-Moduli Set $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} - 1\}$ Based on the New Chinese Remainder Theorem. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 2003, vol. 50, pp. 1296–1303.

[35]. Sheu M.-H., Lin S.-H., Chen C., Yang S.-W. An Efficient VLSI Design for a Residue to Binary Converter for General Balance Moduli $\{2^n - 1, 2^n + 1, 2^n - 3, 2^n + 3\}$. IEEE Transactions on Circuits and Systems II: Express Briefs, 2004, vol. 51, pp. 152–155.

[36]. Zhang W., Siy P. An Efficient Design of Residue to Binary Converter for Four Moduli Set $(2^n - 1, 2^n + 1, 2^{2n} - 2, 2^{2n+1} - 3)$ Based on New CRT II. Information Sciences, 2008, vol. 178, pp. 264–279.

[37]. Molahosseini A.S., Navi K., Dadkhah C., Kavehei O., Timarchi S. Efficient Reverse Converter Designs for the New 4-Moduli Sets $\{2^n-1, 2^n+1, 2^n, 2^{2n}+1\}$ and $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n+1}-1\}$ Based on New CRTs. IEEE Transactions on Circuits and Systems I: Regular Papers 2009, vol. 57, pp. 823–835.

[38]. Patronik P., Piestrak S.J. Design of Reverse Converters for General RNS Moduli Sets $\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$ and $\{2^k, 2^n-1, 2^n+1, 2^{n+1}+1\}$ (n Even). IEEE Transactions on Circuits and Systems I: Regular Papers, 2014, vol. 61, pp. 1687–1700.

[39]. Hiasat A.A. VLSI Implementation of New Arithmetic Residue to Binary Decoders. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2005, vol. 13, pp. 153–158.

[40]. Cao B., Chang C.-H., Srikanthan T. A Residue-to-Binary Converter for a New Five-Moduli Set. IEEE Transactions on Circuits and Systems I: Regular Papers, 2007, vol. 54, pp. 1041–1049.

[41]. Molahosseini A.S., Dadkhah C., Navi K. A. New Five-Moduli Set for Efficient Hardware Implementation of the Reverse Converter. IEICE Electronics Express, 2009, vol. 6, pp. 1006–1012.

[42]. Pettenghi H., Chaves R., Sousa L. RNS Reverse Converters for Moduli Sets with Dynamic Ranges up to (8n+1) -Bit. IEEE Transactions on Circuits and Systems I: Regular Papers, 2012, vol. 60, pp. 1487–1500.

[43]. Pettenghi H., Paludo R., Matos R., Lyakhov P.A. Efficient RNS Reverse Converters for Moduli Sets with Dynamic Ranges Up to (10n+1)(10N+1)-Bit. Circuits, Systems, and Signal Processing, 2018, vol. 37, pp. 5178–5196.

Информация об авторах / Information about authors

Владислав Вячеславович ЛУЦЕНКО – аспирант, кафедры вычислительной математики и кибернетики факультета математики и компьютерных наук имени профессора Н.И. Червякова ФГАОУ ВПО «Северо-Кавказский федеральный университет». Сфера научных интересов: высокопроизводительные вычисления, система остаточных классов, умный город, нейронные сети, интернет вещей.

Vladislav Vyacheslavovich LUTSENKO – postgraduate student, Department of Computational Mathematics and Cybernetics, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov, North Caucasus Federal University. Research interests: high-performance computing, residue number system, smart city, neural networks, Internet of Things.

Михаил Дмитриевич КРАВЦОВ является учеником регионального центра выявления, поддержки и развития способностей и талантов детей и молодежи Ставропольского края «Сириус 26». Сфера научных интересов: высокопроизводительные вычисления, система остаточных классов, компьютерные науки.

Mikhail Dmitrievich KRAVTSOV is a student of the regional center for identification, support and development of abilities and talents of children and youth of the Stavropol Territory Sirius 26. Area of scientific interests: high-performance computing, residue number system, computer science.

Дмитрий Евгеньевич ГОРЛАЧЕВ – студент Северо-Кавказского Федерального университета, факультет математики и компьютерных наук имени профессора Н.И. Червякова. Сфера научных интересов: математика и компьютерные науки.

Dmitriy Evgenievich GORLACHEV is a student of the North Caucasus Federal University, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov. Area of scientific interests: mathematics and computer sciences.

Никита Михайлович МИРНЫЙ – студент Северо-Кавказского Федерального университета, факультет математики и компьютерных наук имени профессора Н.И. Червякова. Сфера научных интересов: математика и компьютерные науки.

Nikita Mikhailovich MIRNY is a student of the North Caucasus Federal University, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov. Area of scientific interests: mathematics and computer sciences.