



К. С. Исупов

Высокопроизводительные вычисления с использованием системы остаточных классов

Аннотация. Система остаточных классов (СОК)—это непозиционная система счисления, являющаяся альтернативой двоичному представлению чисел. В СОК большое целое число представляется в виде набора меньших чисел, являющихся остатками от деления исходной величины на выбранные модули. СОК выполняет сложение, вычитание и умножение с каждым остатком по отдельности. Это приводит к параллельной, свободной от переносов и высокоскоростной компьютерной арифметике для высокопроизводительных вычислений. Однако немодульные операции, требующие оценки величины числа по остаткам, являются сложными для реализации в СОК, так как для них не существует параллельной формы. В вопросах практического использования СОК выполнение немодульных операций занимает центральное место.

Представлен обзор исследований в области разработки и применения на практике методов высокопроизводительных вычислений на основе СОК: Рассмотрены существующие техники выполнения важнейших немодульных операций, таких как обратное преобразование, сравнение чисел, вычисление знака и деление. Акцент сделан на методы, пригодные для произвольных наборов модулей. Показано, каким образом арифметика на основе СОК находит практическое применение в облачных средах, блокчейн-технологиях, вычислениях многократной точности и глубоких нейронных сетях.

Обзор ориентирован на развитие новых направлений исследований, посвященных применению непозиционных систем счисления с параллельной структурой в ресурсоемких приложениях.

Ключевые слова и фразы: система остаточных классов, немодульные операции, высокопроизводительные вычисления, параллельные алгоритмы.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-17-50166.

© К. С. Исупов, 2021

© Вятский государственный университет, 2021

© Программные системы: теория и приложения (дизайн), 2021

 10.25209/2079-3316-2021-12-2-137-192



Введение

Среди различных систем счисления наибольшее распространение получили позиционные b -ичные системы счисления, в частности, двоичная и десятичная системы. В b -ичной системе задается одно фиксированное основание b , и каждой цифре числа присваивается вес b^i , определяемый положением цифры в записи числа. Все цифры числа взаимозависимы, поскольку при выполнении арифметических операций необходимо распространять переносы от младших разрядов к старшим (значимость цифры определяется ее весом). Необходимость распространения переноса приводит к дополнительным накладным расходам, особенно в случае большой разрядности, и это является основным узким местом b -ичных систем счисления.

Иным образом устроены непозиционные системы счисления, в которых вклад каждой цифры в значение числа не зависит от положения цифры. Одной из наиболее известных таких систем является система остаточных классов (СОК) [1–4] которая определяется набором попарно взаимно простых целых чисел, называемых модулями. Большое целое число в СОК представляется в виде набора меньших чисел, являющихся остатками от деления исходной позиционной величины на модули. Особенность в том, что остатки являются взаимно независимыми, т.е. между ними не возникает переносов, что позволяет выполнять сложение, вычитание и умножение с каждым остатком по отдельности. Ввиду этого, обеспечивая возможность реализации высокоскоростной параллельной компьютерной арифметики произвольной разрядности, СОК является перспективным инструментом для высокопроизводительных вычислений.

В последние годы интерес к СОК заметно вырос в связи с активным развитием параллельных архитектур. В настоящее время эта система счисления находит применение во многих ресурсоемких приложениях, например,

- блокчейн [5, 6],
- гомоморфное шифрование [7, 8],
- стохастические вычисления [9],
- высоконадежные облачные среды [10, 11],

- глубокие нейронные сети [12, 13],
- анализ данных компьютерной томографии [14],
- цифровая обработка сигналов [15, 16],
- обработка изображений [17].

Традиционно алгоритмы на основе СОК проектируются для аппаратных реализаций на базе программируемых логических интегральных схем (FPGA) или схем специального назначения (ASIC). В то же время современные массивно-параллельные архитектуры общего назначения, такие как графические процессоры видеокарт (GPU), позволяют эффективно использовать СОК в программном обеспечении. Именно так выполняются операции в асимметричных криптосистемах [18]. На этой же основе реализована арифметика многократной точности на GPU, позволяющая распараллелить вычисления вплоть до уровня отдельных цифр многоразрядных чисел [19, 20].

С другой стороны, в отличие от сложения, вычитания и умножения, такие операции как сравнение, определение знака, масштабирование и деление, называемые немодульными, являются сложными для СОК. Эти операции требуют оценки величины числа по остаткам, и для них не существует параллельной формы. В вопросах практического использования СОК оценка величины числа занимает центральное место. В настоящее время для этого предложено большое количество методов, существенно различающихся по ряду базовых характеристик, определяющих эффективность их применения в реальных приложениях.

Цель настоящей статьи — предоставить литературный обзор исследований, связанных с разработкой и применением на практике методов высокопроизводительных вычислений на основе СОК. Статья имеет следующую структуру. Раздел 1 кратко излагает математические основы и особенности представления чисел в СОК. В разделе 2 исследованы существующие методы выполнения важнейших немодульных операций. В разделе 3 представлены современные ресурсоемкие приложения, для которых использование СОК оказывается полезным. Из рассмотрения исключены такие достаточно хорошо освещенные в литературе приложения, как цифровая обработка сигналов и криптография. Вместо этого обсуждаются применение СОК в облачных хранилищах,

блокчейн-технологиях, вычислениях многократной точности и глубоких нейронных сетях. Статья завершается разделом 4, содержащим заключительные положения.

1. Система остаточных классов

СОК основана на широко известной китайской теореме об остатках (Chinese Remainder Theorem, CRT) [2, 21]. Она утверждает, что, зная наименьшие неотрицательные остатки от деления целого числа X на целые модули m_1, m_2, \dots, m_n , возможно однозначно определить остаток от деления X на произведение этих модулей, при условии, что модули попарно взаимно просты. СОК в отличие от классических b -ичных систем счисления задаётся не одним фиксированным основанием, а набором модулей $\{m_1, m_2, \dots, m_n\}$ таких, что $\gcd(m_i, m_j) = 1$ для всех $i, j \in \{1, 2, \dots, n\}$, $i \neq j$, где $\gcd()$ — наибольший общий делитель. Произведение этих модулей $M = \prod_{i=1}^n m_i$ определяет динамический диапазон СОК. Целое число $X \in [0, M - 1]$ представляется в виде вектора, составленного из наименьших неотрицательных остатков, получаемых при делении X на m_i :

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

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

Отрицательные числа также могут быть представлены в СОК. В общем случае, если X — это целое число со знаком, то диапазон возможных значений X , имеющих уникальное представление в СОК, определяется ограничением $-M \leq 2X < M$.

Над числами в СОК определены базовые операции, которые подразделяются на две группы. К операциям первой группы, которые иногда называются *модульными*, относятся сложение и вычитание чисел без возможности определения знака результата, умножение, а также деление без остатка (когда заранее известно, что делимое кратно делителю). Такие операции выполняются покомпонентно с остатками, т.е. без образования переносов между ними. Пусть числа X , Y и Z представлены в виде (x_1, x_2, \dots, x_n) , (y_1, y_2, \dots, y_n) и (z_1, z_2, \dots, z_n) ,

соответственно. Тогда для всякой модульной операции \circ имеем

$$Z = (|x_1 \circ y_1|_{m_1}, |x_2 \circ y_2|_{m_2}, \dots, |x_n \circ y_n|_{m_n}),$$

т.е. i -я цифра результата в СОК, z_i , определяется только в терминах $|x_i \circ y_i|_{m_i}$ и не зависит ни от какой другой цифры z_j . Это позволяет реализовать свободную от переносов, высокоскоростную (параллельную) компьютерную арифметику и делает СОК привлекательной системой счисления для использования в ресурсоемких приложениях, в особенности связанных с обработкой больших чисел. Это также обеспечивает высокую надежность вычислений, поскольку ошибка в i -й цифре не оказывает влияния на другие цифры и поэтому может быть эффективно локализована и устранена [22].

В свою очередь, для операций второй группы, часто называемых *немодульными*, недостаточно знания значений отдельных остатков и требуется оценка величины чисел: результат такой операции либо вообще не является числом в СОК, либо значение каждой его цифры (остатка) не является только лишь функцией значений соответствующих цифр операндов, а зависит от величины этих операндов. К сожалению, непозиционная структура СОК не позволяет эффективно оценить величину числа по остаткам, и это обстоятельство является основным фактором, сдерживающим широкое распространение СОК в качестве альтернативы b -ичным системам счисления.

2. Методы немодульных вычислений в СОК

В этом разделе рассмотрены методы обратного преобразования, сравнения чисел, определения знака числа, а также деления чисел в СОК. Известно, что на эффективность арифметики в СОК существенно влияет выбор набора модулей, подходящего для решения конкретной задачи [23]. В современных исследованиях часто используются наборы специального вида, состоящие из близких к степени двойки модулей, таких как $\{2^n - 1, 2^n, 2^n + 1\}$, $\{2^{2n}, 2^{2n-1} - 1, 2^{2n-1} + 1\}$ и $\{2^n + 1, 2^n - 1, 2^n, 2^{2n+1} - 1\}$, для некоторого фиксированного параметра n . Определенные математические свойства таких модулей обеспечивают эффективную реализацию многих немодульных операций в СОК, в первую очередь обратное преобразование и сравнение чисел.

С другой стороны, современным приложениям часто требуется большой динамический диапазон вычислений, состоящий из сотен и даже тысяч бит, и единственным приемлемым вариантом для таких приложений является использование наборов модулей произвольного вида. Кроме этого, наборы произвольного вида с близкими по величине модулями являются наиболее экономичными с точки зрения затрат памяти. Поэтому далее рассматриваются методы, пригодные для наборов модулей произвольного вида.

Мы начнем обзор с обратного преобразования, т.е. восстановления целого значения числа по его остаткам, поскольку принципы, лежащие в основе обратного преобразования, используются во многих техниках выполнения других немодульных операций.

2.1. Обратное преобразование

Обратное преобразование является важной частью вычислений в СОК [24]. Базовые методы обратного преобразования основаны на CRT [2, 21] или переходе в промежуточную систему счисления со смешанными основаниями (mixed radix conversion, MRC) [15, 25], а другие существующие техники, как правило, являются их модификациями или комбинациями.

2.1.1. Китайская теорема об остатках (Chinese Remainder Theorem)

Классический метод восстановления целого значения числа по остаткам, получаемый из конструктивного доказательства CRT [26], заключается в вычислении b -ичной суммы по модулю M :

$$(1) \quad X = \left| \sum_{i=1}^n M_i |x_i w_i|_{m_i} \right|_M,$$

где $M_i = M/m_i$, а величины $w_i = |M_i^{-1}|_{m_i}$ известны как мультипликативные (модулярные) инверсии от M_i по модулю m_i , удовлетворяющие $w_i M_i \equiv 1 \pmod{m_i}$. Легко заметить, что $1 \leq w_i < m_i$.

Проблемы при реализации (1) возникают в связи с необходимостью трудоемкой финальной редукции по большому модулю M , т.е. в нахождении остатка от деления вычисленной суммы на M , особенно когда M

состоит из нескольких сотен бит. Поэтому проводимые исследования в основном посвящены разработке алгоритмов, исключающих редукцию по модулю M из процесса вычислений.

2.1.2. Перевод в смешанную систему (*Mixed-Radix Conversion*)

Другой классический метод обратного преобразования не требует модулярной редукции и выражает число X в виде

$$(2) \quad X = \bar{x}_1 W_1 + \bar{x}_2 W_2 + \dots + \bar{x}_n W_n,$$

где W_1, W_2, \dots, W_n — основания смешанной системы счисления:

$$(3) \quad \begin{aligned} W_1 &= 1, \\ W_2 &= m_1, \\ W_3 &= m_1 m_2, \\ &\dots \\ W_n &= m_1 m_2 \dots m_{n-1}. \end{aligned}$$

Гарантируется однозначность представления (2) для всех целых чисел из отрезка $[0, M - 1]$, что указывает на высокую степень сходства СОК и смешанной системы, в основе которых лежит один и тот же набор модулей-оснований.

Краеугольным камнем этого метода является перевод числа из СОК в смешанную систему, т.е. вычисление цифр $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$. Для заданного числа $X = (x_1, x_2, \dots, x_n)$, классический MRC алгоритм [25] последовательно вычисляет все цифры смешанного представления, начиная с наименее значимой:

$$(4) \quad \begin{aligned} \bar{x}_1 &= x_1, \\ \bar{x}_2 &= |(x_2 - \bar{x}_1)c_{12}|_{m_2}, \\ \bar{x}_3 &= |((x_3 - \bar{x}_1)c_{13} - \bar{x}_2)c_{23}|_{m_3}, \\ &\dots \\ \bar{x}_n &= |(\dots(x_n - \bar{x}_1)c_{1n} - \bar{x}_2)c_{2n} - \dots - \bar{x}_{n-1})c_{n-1,n}|_{m_n}, \end{aligned}$$

причем все константы $c_{ij} = |m_i^{-1}|_{m_j}$ вычисляются заранее.

Для вычисления (4) требуется $n(n-1)$ арифметических операций по модулям m_i , а именно, $n(n-1)/2$ вычитаний и столько же умножений. В работе [27] предложен улучшенный алгоритм, позволяющий сократить число умножений до $n-2$. Однако он вводит дополнительные ограничения на выбор модулей СОК, так как должно выполняться условие $|W_i^{-1}|_{m_i} = 1$ для всех $i = 1, 2, \dots, n$.

Хотя каждая следующая цифра \bar{x}_{i+1} зависит от предыдущей цифры \bar{x}_i , процесс (4) может быть частично распараллелен (конвейеризован) [26], как показано в алгоритме 1. В этом алгоритме k — индекс потока, а v — массив в общей памяти, к которому имеют доступ все n потоков. Необходимо отметить, что в конце каждой итерации цикла может потребоваться синхронизация для исключения гонок данных.

АЛГОРИТМ 1. Параллельная (конвейерная) реализация MRC

```

1:  $v_k \leftarrow x_k$ 
2: for  $j \leftarrow 1$  to  $n-1$  do
3:   if  $k > j$  then
4:      $v_k \leftarrow (v_k - v_j)c_{jk} \bmod m_k$ 
5:   end if
6: end for
7:  $\bar{x}_k \leftarrow v_k$ 

```

Таким образом, вычисление смешанного представления выполняется за время $O(n)$, что и является основным недостатком MRC. С другой стороны, в отличие от CRT, MRC алгоритм требует выполнения арифметических операций лишь по небольшим модулям m_i .

В [28] предложена модификация MRC, названная MRC-II, для преобразования числа из СОК в десятичное представление. Алгоритм вычисляет рекуррентное соотношение

$$(5) \quad X_i = \bar{x}_i P_{i-1} + X_{i-1}, \quad 2 \leq i \leq n, \quad n \geq 2,$$

где $P_{i-1} = \prod_{j=1}^{i-1} m_j$ и $X_1 = \bar{x}_1 = x_1$. В конечном итоге, X_n является двоичным представлением числа $X = (x_1, x_2, \dots, x_n)$. Все цифры смешанного представления $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$ вычисляются предварительно и сохраняются в таблицы размера $m_i \log_2 m_i$. Индексом таблицы

является значение $|x_i - X_{i-1}|_{m_i}$ для $i \geq 2$, а содержимое таблицы определяется из уравнения $|x_i - X_{i-1}|_{m_i} = |\bar{x}_i P_{i-1}|_{m_i}$. Хотя МРС-II и имеет линейную сложность по числу модулей СОК, требует хранения значительного объема подстановочных данных и поэтому пригоден для небольших наборов модулей.

2.1.3. *New Chinese Remainder Theorem II*

В [29] предложен оригинальный способ обратного преобразования, названный New CRT II, который основан на принципе «разделяй и властвуй». Пусть $X = (x_1, x_2, \dots, x_n)$ и $t = \lfloor n/2 \rfloor$. Двоичное значение X предлагается вычислять следующим образом:

$$(6) \quad X = X_2 + |k_0(X_1 - X_2)|_{\tilde{M}_1} \tilde{M}_2,$$

где X_1 — целое число, соответствующее остаткам (x_1, x_2, \dots, x_t) относительно модулей $\{m_1, m_2, \dots, m_t\}$ с диапазоном $M_1 = m_1 m_2 \dots m_t$. Аналогично, X_2 — целое число, соответствующее вектору остатков $(x_{t+1}, x_{t+2}, \dots, x_n)$ относительно набора $\{m_{t+1}, m_{t+2}, \dots, m_n\}$ с диапазоном $M_2 = m_{t+1} m_{t+2} \dots m_n$. Константа $k_0 = |M_2^{-1}|_{\tilde{M}_1}$ — мультипликативная инверсия от M_2 по модулю \tilde{M}_1 .

Целые числа X_1 и X_2 , в свою очередь, вычисляются посредством применения (6) с разбиением диапазонов \tilde{M}_1 и \tilde{M}_2 на соответствующие поддиапазоны в два раза меньшей длины, и т.д. Рекурсивное разбиение продолжается до тех пор, пока каждый набор модулей не будет содержать только два элемента. Таким образом, обратное преобразование выполняется за $\log_2 n$ шагов с возможностью распараллеливания вычислений на каждом шаге. Максимальный размер операции при этом сокращается с M до \sqrt{M} .

На базе New CRT II разработаны эффективные обратные преобразователи для произвольных наборов модулей [30], а также для наборов вида $\{2^n - 1, 2^n + 1, 2^{2n} - 2, 2^{2n+1} - 3\}$ [31], $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ [32]. Кроме этого, представленная техника недавно использована для обнаружения искаженной информации в оптимизированном механизме хранения данных блокчейна на основе СОК [5] (см. подробнее в подразделе 3.2).

Заметим, что не обязательно использовать рекурсивный подход для снижения размера операции редукции с M до \sqrt{M} . Достаточно ограничиться однократным вычислением (6), а X_1 и X_2 вычислять с использованием обычной версии CRT, т.е. по формуле (1).

2.1.4. *Mixed-Radix Chinese Remainder Theorem*

В работе [33] предложен комбинированный метод преобразования из СОК в двоичную систему счисления — MR CRT, который совмещает достоинства CRT и MRC методов. Согласно MR CRT, восстановление целочисленного значения X выполняется в соответствии с (2), однако для вычисления цифр смешанного представления $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ вместо (4) используются следующие выражения:

$$\begin{aligned}
 \bar{x}_1 &= x_1, \\
 \bar{x}_2 &= |\gamma_1 x_1 + \gamma_2 x_2|_{m_2}, \\
 (7) \quad \bar{x}_3 &= |[(\gamma_1 x_1 + \gamma_2 x_2 + \gamma_3 x_3)/m_2]|_{m_3}, \\
 &\dots \\
 \bar{x}_n &= |[(\gamma_1 x_1 + \gamma_2 x_2 + \dots + \gamma_n x_n)/m_2 m_3 \dots m_{n-1}]|_{m_n}.
 \end{aligned}$$

Константы $\gamma_1, \gamma_2, \dots, \gamma_n$ вычисляются следующим образом:

$$\gamma_i = \begin{cases} (M_1 w_1 - 1)/m_1, & \text{если } i = 1, \\ M_i w_i / m_1, & \text{если } 2 \leq i \leq n, \end{cases}$$

где M_i и w_i (мультипликативные инверсии по модулю) определяются так же, как в формуле (1).

Вычисление естественным образом распараллеливается, поскольку в (7) все \bar{x}_i являются взаимно независимыми (достоинство CRT). При этом не требуется редукция по большому модулю — все операции выполняются по модулям m_i (достоинство MRC). Таким образом, MR CRT является эффективным методом обратного преобразования.

2.1.5. *Диагональные и монотонные функции*

Концепция диагональных функций, предложенная в [34] и развитая в [35], формулирует проблему выполнения немодульных операций

в терминах оценки индексов диагоналей, расчерчивающих n -мерное пространство, образованное модулями СОК. Первоначально предназначенная для сравнения чисел в СОК, данная концепция в дальнейшем была адаптирована для задачи обратного преобразования [36]. Диагональная функция для числа $X = (x_1, x_2, \dots, x_n)$ определяется следующим образом:

$$(8) \quad D(X) = \left| \sum_{i=1}^n x_i k_i \right|_{\text{SQ}},$$

где $\text{SQ} = M_1 + M_2 + \dots + M_n$, $M_i = M/m_i$ и $k_i = |-m_i^{-1}|_{\text{SQ}}$ (т.е. k_i — это аддитивная инверсия мультипликативной инверсии от m_i по модулю SQ). В свою очередь, преобразование X в двоичную систему числения выполняется в соответствии с выражением [36]:

$$X = \frac{x_1 M_1 + x_2 M_2 + \dots + x_n M_n + M \times D(X)}{\text{SQ}},$$

причем числитель делится на модуль SQ без остатка. Таким образом, редукция по модулю M исключается из процесса вычислений, однако требуется деление на большую константу SQ .

Хотя преобразователи, построенные на базе диагональных функций, проигрывают аналогам по быстродействию, имеется возможность снижения аппаратных затрат путем построения совмещенных устройств, выполняющих функции сравнения чисел и обратного преобразования [36].

Монотонные функции [37] являются, по существу, обобщением диагональных функций. Зададим два множества I и J , такие что $I \cup J = \{1, 2, \dots, n\}$, $I \cap J = \emptyset$ и $I \neq \emptyset$, где n — количество модулей СОК. Пусть, как и прежде, $M_i = M/m_i$ и $w_i = |M_i^{-1}|_{m_i}$. Определим следующие константы:

$$M_I = \sum_{i \in I} M_i,$$

$$S_{INV} = \sum_{i \in I} |m_i^{-1}|_{M_I},$$

и вычислим коэффициенты b_i , $i = 1, 2, \dots, n$, такие что

$$b_i = \begin{cases} |-m_i^{-1}|_{M_I}, & \text{если } i \in I, \\ |M_i \cdot S_{INV} \cdot w_i|_{M_I}, & \text{если } i \in J, \end{cases}$$

Для числа $X = (x_1, x_2, \dots, x_n)$ монотонная функция $F_I(X)$ определяется следующим образом:

$$(9) \quad F_I(X) = \sum_{i \in I} \left\lfloor \frac{X}{m_i} \right\rfloor = \left\lfloor \sum_{i=1}^n x_i b_i \right\rfloor_{M_I}.$$

Легко заметить, что диагональная функция $D(X)$ является частным случаем монотонной функции $F_I(X)$, причем если множество J непустое, то с вычислительной точки зрения (9) эффективнее, чем (8), так как $M_I < SQ$.

На базе монотонных функций в [37] предложены алгоритмы сравнения и обратного преобразования, а в [36] разработана архитектура обратного преобразователя для 4-модульной СОК. Общим недостатком диагональных и монотонных функций является необходимость редукции по модулям SQ и M_I , которые являются довольно большими числами, хотя и меньшими, чем M .

2.1.6. Дробная версия CRT (Fractional Representation)

М. А. Soderstrand предложил [38] вычислять дробное представление числа в СОК, которое получается при делении (1) на M :

$$(10) \quad \frac{X}{M} = \left\| \sum_{i=1}^n \frac{|x_i w_i|_{m_i}}{m_i} \right\|_1,$$

где $\| \cdot \|_1$ – дробная часть суммы в интервале $[0, 1)$.

Умножив (10) на M получим двоичное значение числа X . В отличие от (1) для вычисления дробного представления (10), также называемого относительной величиной, не требуется трудоемкой редукции по модулю M ; вместо этого выполняется более простая операция взятия дробной части суммы.

Отбрасываемая целая часть в (10) имеет самостоятельное практическое значение, в частности, в алгоритмах расширения набора модулей (base extension). Действительно, формула (1) может быть переписана в следующем виде:

$$(11) \quad X = \left(\sum_{i=1}^n M_i |x_i w_i|_{m_i} \right) - rM.$$

Разделив обе части уравнения (11) на M , получим:

$$(12) \quad r = \left\lfloor \sum_{i=1}^n \frac{|x_i w_i|_{m_i}}{m_i} \right\rfloor.$$

В свою очередь, зная r , остаток от X по модулю m_e , не входящему в основной набор модулей $\{m_1, m_2, \dots, m_n\}$, вычисляется эффективно:

$$(13) \quad |X|_{m_e} = \left\lfloor \sum_{i=1}^n |M_i |x_i w_i|_{m_i}|_{m_e} - |rM|_{m_e} \right\rfloor_{m_e}.$$

С момента появления оригинального метода вычисления дробного представления в 1983 году, различные его вариации и модификации активно развивались и в том или ином виде применялись для обратного преобразования и других немодульных операций, связанных с оценкой величины чисел в СОК [18, 39–54].

Главная проблема заключается в том, что слагаемые $|x_i w_i|_{m_i}/m_i$ в формулах (10) и (12), являясь в общем случае рациональными числами, не могут быть вычислены точно с использованием арифметики фиксированной разрядности. Соответственно, возникает ошибка округления, которая без должного внимания может привести к некорректному результату немодульной операции. Для решения этой проблемы существует два основных подхода:

- (1) Вычисление многоразрядного (превышающего разрядность M) дробного представления [39, 41, 44, 47, 48, 52–54]
- (2) Учет ошибки округления и вычисление приближенного дробного представления в арифметике стандартной разрядности [18, 40, 43, 45, 46, 49, 51]

Первый подход предполагает оценку числа бит точности (разрядности), с которой необходимо выполнять суммирование, чтобы гарантировать корректность дробного представления для любых входных данных. Основной недостаток — высокие затраты, связанные с поддержкой арифметики многократной точности. В FPGA и ASIC реализациях это приводит к длинным арифметическим схемам, обладающим большой задержкой. В программных реализациях возникает необходимость в эмуляции вычислений многократной точности средствами компиляторов либо специальных арифметических библиотек. При этом время вычислений возрастает существенно.

В рамках второго подхода вычисления выполняются быстро, так как не используется многоразрядная арифметика. Этот подход применим для операций, которые не требуют полного восстановления целого числа, например для сравнения двух чисел в СОК или определения знака алгебраической суммы. Здесь, однако, важно учитывать все факторы, которые могут повлиять на общую погрешность вычислений (чтобы она не оказалась недооцененной). В частности, если используется арифметика с плавающей точкой, что является наиболее естественным вариантом аппроксимации рациональных чисел, то точность результата зависит от многих параметров вычислений, включая порядок инструкций и опции компилятора [55]. При параллельной обработке контролировать порядок выполнения арифметических операций довольно сложно. С другой стороны, применение интервальной арифметики обеспечивает автоматический учет погрешностей округления [40, 45, 49].

2.1.7. Другие методы

Существует множество других исследований, посвященных разработке эффективных методов и реализаций преобразования чисел из СОК в двоичную систему счисления. В их основе лежат различные теоретико-числовые и арифметические концепции, такие как функция ядра (core function) [56], теорема о декомпозиции набора модулей [57], использование избыточного модуля [58] и вычисление функции ранга числа с помощью SRT-алгоритма [59].

J. Chen и R. Yao разработали алгоритм [60], в котором вначале вычисляется b -ичная сумма X_{medium} первых $n - 1$ остатков числа

в СОК, а затем результат обратного преобразования определяется следующим образом:

$$X = \begin{cases} X_{\text{medium}} + k\hat{M}, & \text{если } X_{\text{medium}} + k\hat{M} \leq M, \\ X_{\text{medium}} + k\hat{M} - M, & \text{в противном случае,} \end{cases}$$

где $\hat{M} = m_1 m_2 \cdots m_{n-1}$. Для вычисления k требуется лишь одно вычитание и одно умножение по модулю m_n . При некоторых незначительных ограничениях на выбор модулей использование разработанного алгоритма ведет к существенному снижению аппаратных затрат и времени обратного преобразования по сравнению с [30] и [48].

Недавно Р. Patronik предложил похожую технику [61] для модулей вида $\{m_1, m_2, \dots, m_{n-1}, 2^k\}$, где все m_i — произвольные попарно взаимно простые числа и k — произвольная константа. Преобразователь формирует два слагаемых, X_s и X_c , такие что $0 \leq X_s + X_c < 2M$, где M — полный динамический диапазон СОК с учетом модуля 2^k . Искомая целочисленная величина получается простым выбором между $(X_s + X_c)$ и $(X_s + X_c) - M$. Представленные в [61] результаты синтеза показывают, что разработанная архитектура более эффективна, чем [54] и [60].

В последние годы активно разрабатываются преобразователи для специальных наборов модулей, таких как $\{2^n - 1, 2^n, 2^n + 1\}$ [62], $\{2^n + 1, 2^n - 1, 2^n, 2^{2n+1} - 1\}$ и $\{2^n + 1, 2^n - 1, 2^{2n}, 2^{2n+1} - 1\}$ [63], $\{2^n, 2^n - 1, 2^n, 2^{n+1} - 1\}$, $\{2m - 1, 2m, 2m + 1\}$ и $\{2^n - 3, 2^n - 1, 2^n + 1, 2^n + 3\}$ [64], $\{2^{2n}, 2^{2n-1} - 1, 2^{2n-1} + 1\}$ [65] и других [66]. Модули специального вида обладают полезными арифметическими свойствами, делающими процесс обратного преобразования более эффективным. С дополнительной информацией по методам и архитектурам обратного преобразования читатель может ознакомиться в [1, глава 5].

Хотя обратное преобразование играет важную роль в вопросах практического использования СОК, его влияние на общую производительность может быть снижено, если все промежуточные вычисления будут осуществляться непосредственно в СОК, а в двоичную систему будет переводиться лишь финальный результат. Поэтому далее мы

рассмотрим методы выполнения немодульных операций, не требующие полного восстановления целочисленного представления чисел из остатков.

2.2. Сравнение и вычисление знака

Сравнение чисел в СОК является одной из наиболее важных операций и тесно связано с другими операциями, в частности, с вычислением знака и оценкой переполнения диапазона. Кроме этого, сравнение выступает строительным блоком в более сложных операциях, таких как масштабирование и деление. В связи с этим в литературе предложено большое количество алгоритмов сравнения.

Одним из очевидных вариантов реализации операции сравнения является вычисление коэффициентов (4) смешанных представлений чисел с использованием MRC алгоритма. Цифра \bar{x}_1 в смешанном представлении числа X является наименее значимой, а цифра \bar{x}_n — наиболее значимой, поэтому достаточно сравнить операнды покомпонентно, начиная со старших цифр, как показано в алгоритме 2.

АЛГОРИТМ 2. Сравнение чисел с использованием MRC

```

1: Вычислить  $[\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n] \leftarrow \text{MRC}(X)$ 
2: Вычислить  $[\bar{y}_1, \bar{y}_2, \dots, \bar{y}_n] \leftarrow \text{MRC}(Y)$ 
3: for  $i \leftarrow n$  downto 1 do
4:   if  $\bar{x}_i \neq \bar{y}_i$  then
5:     return  $(\bar{x}_i > \bar{y}_i) ? 1 : -1$ 
6:   end if
7: end for
8: return 0

```

MRC метод также может быть использован для вычисления знака и оценки переполнения суммы двух чисел. Более того, если известно, что операнды имеют одинаковый знак, то их сравнение может быть выполнено в два раза быстрее за счет сравнения цифр смешанного представления разности с заранее вычисленными цифрами константы, соответствующей половине динамического диапазона.

Преимущества MRC метода сравнения является полное отсутствие необходимости в многоразрядных вычислениях: для нахождения

всех коэффициентов смешанного представления требуется только целочисленная арифметика по модулям m_i . Недостатком является последовательная природа MRC процесса, приводящая к минимальной оценке времени выполнения — $O(n)$. Для ресурсоемких приложений, использующих большие наборы модулей, эта оценка часто оказывается неприемлемой. Другим недостатком MRC является необходимость в дополнительных массивах, хранящих цифры x_i и y_i . Этот недостаток становится существенным при реализации алгоритма на системах со сложной иерархией памяти, таких как GPU.

Алгоритм сравнения на базе техники New CRT II, рассмотренной в подразделе 2.1.3, был предложен в [67]. Он позволяет выполнить сравнение двух чисел за $O(\log n)$ шагов, но требует операций по модулю \sqrt{M} , поэтому оказывается непрактичным, когда динамический диапазон СОК состоит из тысяч бит. Кроме этого, процедура *RNS_Compare_size2* содержит большое количество условных операторов, что может стать причиной плохой производительности при реализации алгоритма на SIMD системах.

Алгоритм сравнения на основе MR CRT (см. подраздел 2.1.4) состоит в вычислении и последующем анализе коэффициентов (7), как описано в [33]. При достаточном количестве ресурсов вычисление может быть выполнено за время $O(\log n)$. Хотя редукция по большому модулю в явном виде исключается, требуется выполнять умножение и сложение многоразрядных констант γ_i , а также деление на $\prod_{i=2}^{j+1} m_i$ для всех j от 1 до $n - 2$. С другой стороны, в этой же работе авторы предложили и реализовали на базе MR CRT эффективный алгоритм сравнения для популярного набора из трех модулей $\{2^n - 1, 2^n, 2^n + 1\}$.

В статье [34] разработан метод сравнения на основе диагональных функций (см. подраздел 2.1.5). Поскольку диагональная функция (8) монотонно возрастает на интервале $[0, M - 1]$, для сравнения чисел $X = (x_1, x_2, \dots, x_n)$ и $Y = (y_1, y_2, \dots, y_n)$ достаточно вычислить и сравнить их диагональные функции $D(X)$ и $D(Y)$. Совпадение значений функций указывает на то, что числа лежат на одной диагонали. В этом случае о результате сравнения можно судить по соотношению любой пары остатков (x_i, y_i) . К сожалению, данный метод требует операций по большому модулю $SQ = \sum_{i=1}^n M/m_i$ и

поэтому непрактичен при программной реализации. С другой стороны, работа [68] ставит под сомнение эффективность его аппаратной реализации. Вместе с тем, диагональные и монотонные функции являются важными математическими концепциями, позволяющими лучше понять специфические свойства целых чисел [69].

В работе [52] предложен метод определения знака числа в СОК с использованием дробной версии CRT (см. подраздел 2.1.6). Приведенный анализ ошибки округления показывает, что для однозначного определения знака дробная часть каждого слагаемого, участвующего в вычислениях, должна содержать по меньшей мере t бит, где

$$\begin{aligned} t &\geq \lceil \log_2 Mn \rceil - 1 && \text{для четного } M, \\ t &\geq \lceil \log_2 Mn \rceil && \text{для нечетного } M. \end{aligned}$$

М. Lu и J.-S. Chiang в рамках разработки алгоритма деления предложили метод на основе контроля четности для сравнения и оценки переполнения в СОК [39]. В соответствии с методом результат сравнения беззнаковых чисел X и Y определяется по следующим правилам, при условии, что набор модулей СОК состоит только из нечетных модулей.

- (1) Пусть X и Y имеют одинаковую четность и $Z = X - Y$. Тогда $X \geq Y$, если Z — четное и $X < Y$, если Z — нечетное.
- (2) Пусть X и Y имеют разную четность и $Z = X - Y$. Тогда $X \geq Y$, если Z — нечетное и $X < Y$, если Z — четное.

Аналогичные правила применяются для чисел со знаками. В [39] предложено два способа определения четности числа по остаткам. Первый основан на использовании подстановочной таблицы, хранящей признак четности для всех чисел из диапазона $[0, M - 1]$. Размер таблицы пропорционален M , поэтому способ подходит лишь для небольших наборов модулей. Второй способ предполагает вычисление признака четности следующим образом:

$$P = LSB(|x_1 w_1|_{m_1}) \oplus \dots \oplus LSB(|x_n w_n|_{m_n}) \oplus LSB(r),$$

где LSB — младший бит числа, w_i — константы из формулы (1), а коэффициент r вычислялся с использованием дробной версии CRT

по формуле (12). Как и в [52], этот способ требует использования многоразрядной арифметики, причем количество бит в дробной части слагаемых $|x_i w_i|_{m_i}/m_i$ должно превышать $\log_2 Mn$.

Также в [39] отмечено, что все необходимые для вычисления r многоразрядные слагаемые могут быть вычислены заранее. Вместе с тем, их суммирование является непростой задачей и, более того, существенный объем подстановочных таблиц может стать проблемой при использовании больших наборов модулей.

В свою очередь, алгоритм сравнения чисел в СОК из [41] оперирует многоразрядными величинами с $\lceil \log_2 M\rho \rceil$ битами в дробной части, где параметр ρ определяется как $\sum_{i=1}^n (m_i - 1)$.

В недавней работе [70] предложен алгоритм сравнения, сочетающий идеи диагональных функций и дробного представления. В алгоритме вычисляется модифицированная диагональная функция

$$\bar{D}(X) = \left\lfloor \sum_{i=1}^n \bar{k}_i x_i \right\rfloor_{2^N},$$

где $\bar{k}_i = \lceil k_i \cdot 2^N / \text{SQ} \rceil$, а константы k_i и SQ определяются так же как в (8). Операции по модулю 2^N выполняются значительно проще и эффективнее, чем операции по модулю SQ, необходимые для вычисления $D(X)$ в оригинальной концепции. Действительно, чтобы найти остаток от деления на 2^N достаточно игнорировать переносы из $(N - 1)$ -го разряда двоичного представления. Сравнение чисел X и Y сводится к сравнению их модифицированных диагональных функций $\bar{D}(X)$ и $\bar{D}(Y)$ в двоичной системе счисления. Однако алгоритм требует вычислений с многоразрядными числами, поскольку размер модуля 2^N определяется следующим образом:

$$N \geq \lceil \log_2(\text{SQ} \cdot (m_n - 1)) \rceil.$$

Все рассмотренные выше методы для сравнения и вычисления знаков чисел в СОК, за исключением MRC метода, хотя и не требуют полного восстановления двоичного представления числа с трудоемкой редукцией по модулю M , но предполагают работу с многоразрядными целыми или дробными величинами, причем разрядность вычислений

часто превышает разрядность произведения модулей M . Это кажется приемлемым для высокопроизводительных аппаратных реализаций, предназначенных для приложений СОК, не требующих больших динамических диапазонов, например, цифровая обработка сигналов.

С другой стороны, это влечет серьезные проблемы при программной реализации, поскольку на вычислительных системах общего назначения многоразрядные (превышающие длину машинного слова) арифметические операции выполняются во много раз медленнее стандартных операций фиксированной разрядности, которые поддерживаются аппаратно. Поэтому далее рассмотрим методы сравнения и вычисления знака в СОК, не требующие использования многоразрядной арифметики.

В [46] предложено использовать округленные дробные числа для определения знака в СОК. Алгоритм включает в себя вычисление приближенного значения $EF_\alpha(X) \approx X/M$ путем округления каждого слагаемого в уравнении (10) до некоторого числа бит, определяемого параметром α . Такие округленные слагаемые вычисляются заранее для всех возможных значений каждого из остатков и сохраняются в подстановочные таблицы, так что нахождение $EF_\alpha(X)$ выполняется очень быстро, поскольку сводится к выборке и простому суммированию заранее вычисленных значений с последующим отбрасыванием дробной части суммы. Суммирование выполняется с округлением до $\beta = \alpha + \lceil \log n \rceil$ бит. Последующее определение знака сводится к анализу $EF_\alpha(X)$.

Метод эффективен с точки зрения скорости расчетов, однако он накладывает ограничения на диапазон вычислений: для однозначного определения знака требуется, чтобы входной аргумент X удовлетворял неравенствам

$$(14) \quad -\left(\frac{1}{2} - 2^{-\alpha}\right)M \leq X \leq \left(\frac{1}{2} - 2^{-\alpha}\right)M.$$

Кроме этого, метод не подходит для больших наборов модулей ввиду отсутствия механизма уточнения $EF_\alpha(X)$ в случае малой величины X . В самом деле, при постоянных α и n ошибка округления (14) ограничена сверху некоторой фиксированной границей, которая может оказаться больше, чем X/M .

В [43] предложен более общий алгоритм вычисления знака, который также основан на округленном дробном представлении и использует только быструю стандартную арифметику с плавающей точкой, а также операции по модулям m_i . Этот алгоритм, названный авторами «Recursive relaxation of the moduli», предоставляет элегантный механизм итерационного уточнения вычислений для небольших входных данных, состоящий в последовательном исключении из рассмотрения модулей $m_n, m_{n-1}, m_{n-2}, \dots$ с соответствующим сокращением динамического диапазона до тех пор, пока на j -й итерации вычисляемая дробная величина $S^{(j)}$ не окажется больше некоторой оценки ошибки округления ε_j . Шаги вычислений показаны в алгоритме 3.

АЛГОРИТМ 3. Recursive relaxation of the moduli [43]

```

1:  $j \leftarrow n + 1$ 
2: repeat
3:    $j \leftarrow j - 1$ 
4:    $S^{(j)} \leftarrow \left\lfloor \sum_{i=1}^j |x_i w_i^{(j)}|_{m_i} / m_i \right\rfloor_1$ 
5: until  $|S^{(j)}| > \varepsilon_j$  or  $j = 0$ 
6: return  $\text{sign}(S^{(j)})$ 

```

В отличие от [46], алгоритм вычисления знака из [43] пригоден для произвольных и больших наборов модулей. Однако он также накладывает ограничение на входные данные:

$$(15) \quad |X| \leq \frac{M}{2}(1 - \varepsilon_n),$$

где ε_n — граница абсолютной ошибки округления.

С одной стороны, ограничения (14) и (15) кажутся мягкими, поскольку не приводят к существенному сокращению диапазона вычислений. Вместе с тем, гарантировать их выполнение на практике — не простая задача, в особенности если аргумент, т.е. число $X = (x_1, x_2, \dots, x_n)$ для которого требуется вычислить знак, сам по себе является результатом предыдущих вычислений в СОК.

Другой метод [40] для сравнения чисел в СОК и выполнения связанных немодульных операций основан на идее, что не обязательно вычислять точное значение X/M , а достаточно иметь интервал

$I(X/M) = [\underline{X/M}, \overline{X/M}]$, такой что $\underline{X/M} \leq X/M \leq \overline{X/M}$. Этот интервал называется интервальной оценкой с плавающей точкой числа X (или просто интервальной оценкой числа X). Границы интервала — числа с плавающей точкой ограниченной разрядности. Вычисление $I(X/M)$ по остаткам числа $X = (x_1, x_2, \dots, x_n)$ выполняется в среднем за время $O(\log n)$ при распараллеливании на n потоков и требует только операций по модулям m_i и стандартных операций с плавающей точкой с направленными округлениями. Соответственно, метод пригоден для современных вычислительных платформ общего назначения, которые поддерживают быструю арифметику IEEE 754. Иллюстрация метода показана на рисунке 1.

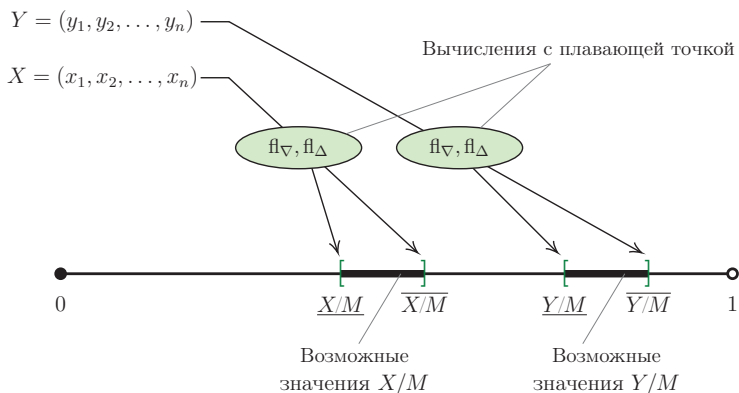


РИСУНОК 1. Использование интервальных оценок для сравнения чисел в СОК: $X < Y$, так как $\overline{X/M} < \underline{Y/M}$

В вычислении $I(X/M)$ при малой величине аргумента X применяется итерационный механизм уточнения, схожий с алгоритмом 3. Существенным отличием является то, что каких-либо ограничений на диапазон изменения аргумента X не накладывается. Другими словами, обеспечивается корректное вычисление $I(X/M)$ для любого $X \in [0, M - 1]$. В некоторых граничных случаях, когда аргумент находится слишком близко к 0 или M , могут потребоваться дополнительные шаги для разрешения неоднозначности. Над интервальными оценками определены арифметические операции, аналогичные операциям с обычными вещественными интервалами. Это позволяет

использовать их не только для сравнения и вычисления знака, но также для оценки переполнения при сложении и умножении чисел в СОК. Кроме того, применение интервальных формул позволяет вычислить $I(X/M)$ за время $O(1)$.

Работы [49] и [71] являются дальнейшим развитием метода интервальных оценок. В [49] предложен улучшенный алгоритм вычисления $I(X/M)$, в котором для разрешения граничных (неоднозначных) случаев применяется MRC-техника, а в основных вычислительных циклах используется попарное суммирование, обеспечивающее лучшую точность по сравнению с классической рекурсивной схемой. Кроме того, в [49] разработаны новые алгоритмы сравнения и деления чисел в СОК на основе интервальных оценок и представлены результаты их реализации на GPU. В [71] опубликована модификация алгоритма вычисления $I(X/M)$, позволяющая снизить число итераций уточняющего цикла в случае малых значений аргумента. Также в [71] представлена реализация параллельной схемы вычисления максимального элемента массива чисел в СОК на GPU. На основе интервальных оценок разработана CUDA библиотека высокопроизводительных вычислений в СОК, поддерживающая наборы модулей с диапазонами до нескольких тысяч бит¹.

В [72] предложен метод определения знака числа и оценки переполнения при сложении и вычитании в СОК, основанный на анализе коэффициента реконструкции \mathcal{R}_C , который является, по существу, тем же самым, что (12). Вычисление \mathcal{R}_C выполняется за логарифмическое время (в среднем) в формате с фиксированной точкой с использованием алгоритма RPPR, который не накладывает ограничений на диапазон допустимых значений аргумента, но требует использования избыточного модуля m_e , взаимно простого со всеми остальными модулями системы. Максимальная длина операндов, участвующих в вычислениях, составляет $\lceil \log_2 n\Phi \rceil$ бит, где Φ — некоторый параметр точности. Результаты экспериментов, представленные в [72], показывают, что на больших динамических диапазонах СОК (от 512 до 5000+ бит) алгоритм RPPR выполняется существенно быстрее, чем вычисление многоразрядного дробного представления. В экспериментах были использованы значения Φ от 10 до 42.

¹Доступна для скачивания по адресу <https://github.com/kisupov/grns>

В таблице 1 представлена сводная информация по рассмотренным алгоритмам.

ТАБЛИЦА 1. Оценки производительности алгоритмов сравнения и вычисления знака в СОК

| | Time | Opsize | Restrict | Redund |
|------|---------------------|---|----------|--------|
| MRC | $O(n)$ | $\text{mod } m_i$ | ○ | ○ |
| [67] | $O(\log n)$ | $\text{mod } \sqrt{M}$ | ○ | ○ |
| [33] | $O(\log n)$ | $\approx \sum_{i=1}^n M/m_i$ | ○ | ○ |
| [34] | $O(\log n)$ | $\text{mod SQ} = \sum_{i=1}^n M/m_i$ | ○ | ○ |
| [52] | $O(\log n)$ | $\log_2 Mn$ бит [‡] | ○ | ○ |
| [39] | $O(\log n)$ | $\log_2 Mn$ бит [‡] | ○ | ○ |
| [46] | $O(\log n)$ | $\alpha + \lceil \log n \rceil$ | ● | ○ |
| [43] | $O(\log n)^\dagger$ | 32/64 бит | ● | ○ |
| [41] | $O(\log n)$ | $\log_2 M \sum_{i=1}^n (m_i - 1)$ бит [‡] | ○ | ○ |
| [40] | $O(\log n)^\dagger$ | 32/64 бит | ○ | ○ |
| [70] | $O(\log n)$ | $\text{mod } 2^N, N \geq \lceil \log_2 \text{SQ}(m_n - 1) \rceil$ | ○ | ○ |
| [72] | $O(\log n)^\dagger$ | $\log_2 n\Phi$ бит [‡] | ○ | ● |

[†] Указана оценка для «среднего» случая, в граничных случаях оценка выше.

[‡] С округлением «вверх» до целого.

Обозначения столбцов таблицы расшифровываются следующим образом:

Time временная сложность алгоритма;

Opsize наибольшая операция редукции по модулю либо наибольшие целые числа, участвующие в вычислениях, либо разрядность в битах дробной части операндов; значение «32/64 бит» указывает на то, что не требуется работать с длинными числами, выходящими за пределы разрядной сетки вычислительной системы, т.к. вычисления

выполняются в целочисленной арифметике по модулям m_i или в стандартной арифметике с плавающей точкой;

Restrict наличие ограничений, накладываемых на возможные значения аргумента;

Redund необходимость использования избыточных модулей.

2.3. Деление

Для заданных чисел $X = (x_1, x_2, \dots, x_n)$ и $Y = (y_1, y_2, \dots, y_n)$ деление состоит в вычислении $Z = (z_1, z_2, \dots, z_n)$, такого что $Z = \lfloor X/Y \rfloor$. Согласно теореме о делении с нулевым остатком [25], частное Z может быть найдено простым покомпонентным умножением X на мультипликативную инверсию Y , но только если заранее известно, что X кратно Y . В общем же случае деление — одна из наиболее сложных операций в системе остаточных классов. В литературе описаны различные способы ее реализации, которые, как и двоичные алгоритмы деления, можно разделить на два класса: алгоритмы на основе вычитания и алгоритмы на основе умножения, причем большинство существующих техник относятся к первому классу.

В [39] представлен алгоритм деления чисел в СОК со знаками, который вначале вычисляет 2^k , такое что $(2^k \cdot Y) \leq X < (2^{k+1} \cdot Y)$, а затем находит разность между 2^k и истинным частным методом бинарного поиска. Вычисление 2^k выполняется итерационно, и на каждой i -й итерации $(2^i \cdot Y)$ сравнивается с X . Кроме этого, $(2^{i+1} \cdot Y)$ сравнивается с $(M - 1)/2$ для контроля переполнения. Как уже упоминалось выше, для сравнения и оценки переполнения используется техника контроля четности. При большом динамическом диапазоне алгоритм требует многоразрядной арифметики для вычисления (12).

Алгоритм, предложенный в [44], основан на вычислении наиболее значимых ненулевых бит делимого (остатка) и делителя, $j = h(X)$ и $k = h(Y)$, соответственно. Если $j > k$, то к частному прибавляется 2^{j-k-1} и процесс повторяется. Алгоритм не требует вычисления знака и контроля переполнения, а для вычисления j и k предложено две реализации. Реализация I основана на использовании подстановочной таблицы большого размера, из которой выбираются j и k . Реализация II

вычисляет многоразрядные дробные представления чисел, X/M и Y/M , после чего $j = h(X/M)$ и $k = h(Y/M)$. Для исключения трудоемкого деления при вычислении (12), слагаемые $|x_i w_i|_{m_i} / m_i$ вычисляются заранее и сохраняются в n таблиц; адресом для выборки i -го слагаемого является остаток x_i . При большом числе модулей объем таблиц может быть существенным. В [73] предложена оптимизация алгоритма для двух специальных наборов: $\{2^k, 2^k - 1, 2^{k-1} - 1\}$ и $\{2^k + 1, 2^k, 2^k - 1\}$.

В статье [47] показано, что количество итераций алгоритма [44] может быть снижено, если допустить получение временного частного, превышающего истинное частное. Для определения знака делимого на каждой итерации используется техника контроля четности. В свою очередь, в алгоритме деления из [74] значения $h(X)$ и $h(Y)$ вычисляются только на первой итерации, а для дальнейшего уточнения частного используется метод половинного деления.

Алгоритм [75] также основан на анализе дробных представлений делимого и делителя, $F(X) = X/M$ и $F(Y) = Y/M$, которые представлены в формате с фиксированной точкой и вычисляются однократно перед началом итераций. Дальнейший процесс строится на базе $F(X)$ и $F(Y)$ (исходные операнды в нем уже не участвуют) и состоит из двух этапов. На первом этапе $F(Y)$ сдвигается влево до тех пор, пока не превысит $F(X)$. Количество сдвигов определяет потенциальную позицию старшей цифры частного. Второй этап схож с обычным делением в двоичной системе счисления. Здесь посредством повторяющихся вычитаний и сдвигов $F(Y)$ вправо вычисляется серия степеней двойки, формирующих итоговое частное. Таким образом, итерационный процесс деления состоит из сложений в СОК, а также вычитаний и сдвигов двоичных чисел, и поэтому выполняется эффективно. С другой стороны, как и в [41], при большом динамическом диапазоне для вычисления $F(X)$ и $F(Y)$ необходимо использовать многоразрядную арифметику, так как точность дробной части составляет $N = \lceil \log_2 M \sum_{i=1}^n (m_i - 1) \rceil$ бит.

В [46] разработаны алгоритмы деления, общая структура которых аналогична известному алгоритму SRT (Sweeney, Robertson, Tocher) [77]: на каждой итерации основного цикла вычисляется знак остатка и выполняется соответствующая корректировка частного значением

-1, 0 или 1 с последующим его удвоением. Для быстрого вычисления знака в СОК используется приближенная техника на основе вычисления округленного дробного представления $EF_\alpha(X) \approx X/M$ (см. подраздел 2.2). Существенным достоинством алгоритмов является отсутствие многоразрядных операций — все вычисления выполняются с использованием величин ограниченной разрядности.

В [49] предложен алгоритм деления на основе вычисления интервальных оценок делителя и делимого. Пусть $I(X/M) = [\underline{X/M}, \overline{X/M}]$ и $I(Y/M) = [\underline{Y/M}, \overline{Y/M}]$ — интервальные оценки делимого (остатка) X и делителя Y . На каждой итерации алгоритма в арифметике с плавающей точкой вычисляется

$$q \leftarrow \text{fl}_\nabla \left(\underline{X/M} \div \overline{Y/M} \right),$$

где fl_∇ означает, что вычисления внутри скобок выполняются с округлением «вниз» (к минус бесконечности). Поскольку q — число с плавающей точкой, оно имеет вид $q = f \times 2^e$, причем $1 \leq f < 2$. Значение e используется в качестве адреса подстановочной таблицы для выбора очередного приближения частного $Q_i = (|2^e|_{m_1}, |2^e|_{m_2}, \dots, |2^e|_{m_n})$, после чего остаток X корректируется и $I(X/M)$ вычисляется вновь. Итерации продолжаются до тех пор, пока $\underline{X/M} \geq \overline{Y/M}$. Таким образом, деление требует только операций по небольшим модулям m_i и стандартных операций с плавающей точкой. В редких случаях после основного цикла может возникнуть неоднозначность, которая устраняется вызовом MRC алгоритма.

Также в [49] предложена улучшенная версия алгоритма деления, которая позволяет сократить количество вычислений $I(X/M)$ путем анализа каждого ненулевого бита в q . Оба алгоритма были реализованы на GPU и протестированы на больших наборах модулей СОК с диапазонами от 64 до 4096 бит, показав существенное ускорение и экономию памяти по сравнению с многоразрядным алгоритмом из [44]. С другой стороны, как и в любых других алгоритмах на базе вычитания, если делитель значительно меньше делимого, то число итераций может быть большим.

В [76] разработан алгоритм, который вычисляет обратную величину делителя относительно диапазона СОК, используя итерации Ньютона.

Частное получается умножением делимого на обратную величину. Для промежуточных результатов используется расширенный набор модулей СОК, который обеспечивает приблизительно квадрат рабочего диапазона (т.е. разрядность расширенного диапазона в два раза больше разрядности рабочего диапазона), что может стать существенным ограничением. Кроме того, для получения финального результата требуется масштабирование (деление на M).

Алгоритм деления, предложенный в [79], итерационно вычисляет $X_i = X_{i-1} - YZ_i$, $i = 1, 2, 3, \dots$, начиная с $X_0 = X$. Итерации завершаются на r -й итерации, если $X_r = 0$ или $Z_r = 0$, а финальный результат деления вычисляется суммированием Z_i с возможной прибавкой, определяемой значениями Z_r , X_r и X_{r-1} . На каждой итерации для нахождения очередной прибавки к частному Z_i остаток X_{i-1} преобразуется из СОК в систему со смешанными основаниями, что является довольно медленным. Кроме этого, алгоритм требует использования предварительно вычисленных таблиц, содержащих в общей сложности $(m_n - 1)(m_n - 2)/2 + (n - 2)(n - 1)/2$ слов. В [78] представлен улучшенный алгоритм деления, основанный на тех же принципах, но позволяющий снизить объем предварительно вычисляемых данных до $S_{RA} < \sum_{i=1}^n (m_i - 1) + (n - 2)(n - 1)/2$ слов и имеющий на 5% меньшее среднее время выполнения.

В [80] и [81] деление выполняется путем нахождения новых делимого и делителя, которые меньше исходных операндов, но имеют то же соотношение. Целью является получение делителя, равного 1, так как в этом случае делимое будет являться искомым результатом деления. Процесс итерационный и предполагает на каждом шаге умножение делителя на мультипликативную инверсию по модулю M . Недостатком является необходимость в использовании вспомогательной СОК с диапазоном, не меньшим исходного диапазона, для избежания переполнения промежуточных результатов.

В таблице 2 сведены особенности рассмотренных алгоритмов деления, которые могут оказать существенное влияние на эффективность реализации, в первую очередь при использовании больших наборов модулей с динамическими диапазонами из сотен и тысяч бит.

Столбцы таблицы 2 обозначают следующее:

- ROM* объем предварительно вычисляемых данных может быть слишком большим;
- MultiPrec* используется многоразрядная арифметика;
- FlPoint* используется арифметика с плавающей точкой стандартной разрядности (32/64 бит);
- MRC* требуется преобразование в систему со смешанными основаниями (mixed-radix conversion);
- BaseExt* требуется расширение набора модулей СОК, т.е. вычисление остатков по модулям на базе имеющихся остатков по другим модулям;
- Restrict* накладываются дополнительные ограничения на величину делимого или делителя.

ТАБЛИЦА 2. Ключевые особенности алгоритмов деления чисел в СОК

| | ROM | Multiprec | FlPoint | MRC | BaseExt | Restrict |
|-------------------|-----|-----------|---------|-----|---------|----------|
| [39] | ● | ● | ○ | ○ | ○ | ○ |
| [44] [†] | ● | ○ | ○ | ○ | ○ | ○ |
| [44] [‡] | ● | ● | ○ | ○ | ○ | ○ |
| [46]* | ○ | ○ | ● | ● | ○ | ● |
| [47] | ● | ● | ○ | ○ | ○ | ○ |
| [49]* | ○ | ○ | ● | ● | ○ | ○ |
| [74] | ● | ● | ○ | ○ | ○ | ○ |
| [75] | ○ | ● | ○ | ○ | ○ | ○ |
| [76] | ○ | ○ | ○ | ● | ● | ○ |
| [79] | ● | ○ | ○ | ● | ● | ○ |
| [78] | ● | ○ | ○ | ● | ● | ○ |
| [80] | ● | ○ | ○ | ○ | ● | ○ |
| [81] | ○ | ○ | ○ | ○ | ● | ○ |

[†] Реализация I.

[‡] Реализация II.

* MRC требуется только в отдельных (неоднозначных) случаях.

3. Использование СОК в высокопроизводительных приложениях

В этом разделе представлено несколько современных приложений, в которых применяется система остаточных классов. Мы не рассматриваем такую классическую область, как цифровая обработка сигналов, поскольку данной теме посвящено множество работ (см, например, [15, 16]). Другим известным применением СОК является криптография [7, 8, 18, 82, 83], которую однако мы также исключили из рассмотрения, поскольку существуют современные обширные обзоры по данной теме [84, 85].

3.1. Долгосрочная доступность и конфиденциальность данных в облачном хранилище

Ежедневно во всем мире генерируются колоссальные объемы данных, требующие эффективного и безопасного хранения. Часто для этих целей используется облачное хранилище, состоящее из совокупности распределенных в сети серверов, управляемых хостинговой компанией (провайдером) и предоставляемых клиенту как единый логический пул с возможностью доступа через веб-интерфейс или API. Важнейшими задачами, стоящими перед сервисами облачного хранения, являются обеспечение долгосрочной доступности и конфиденциальности данных пользователей. Для решения этих задач M. Villari и др. [10] предложили механизм, основанный на использовании избыточной СОК с $p + r$ модулями: модули $m_1 \dots m_p$ определяют рабочий диапазон системы, а модули $m_{p+1} \dots m_{p+r}$ определяют избыточный диапазон. Механизм работает следующим образом.

Предварительно зашифрованные данные (файлы) пользователя посредством преобразования в СОК разбиваются на $p + r$ сегментов, каждый из которых ассоциирован со своим модулем m_i . Далее эти сегменты кодируются в BASE-64, записываются в специальные XML-оболочки и загружаются различным провайдерам облачных хранилищ, причем, по возможности каждому провайдеру загружается только один сегмент данных. Сводная информация по загруженным сегментам записывается в отдельный метафайл определенной структуры, который

либо хранится у пользователя, либо после определенных преобразований также загружается в облачное хранилище. Для получения данных сегменты загружаются из облачных хранилищ и декодируются из BASE-64, после чего восстанавливаются с использованием CRT формулы (1).

Описанный механизм распределенного хранения позволяет достичь двух важных целей:

- (1) Высокая конфиденциальность, так как каждый провайдер обладает только i -й частью данных, соответствующих одному модулю СОК, и поэтому не сможет восстановить их.
- (2) Долгосрочная доступность, так как даже если r сегментов данных по каким-либо причинам с течением времени окажутся недоступными, оставшихся p сегментов будет достаточно для однозначного CRT декодирования.

Таким образом, коэффициент избыточности СОК определяет максимальное число узлов (серверов), выход из строя которых не препятствует успешному восстановлению данных пользователя.

В [11] предложен улучшенный механизм распределенного хранения, в котором шифрование с использованием симметричного алгоритма (например, AES) применяется не к исходным данным, а по отдельности к каждому из $p + r$ сегментов, т.е. уже после преобразования исходных данных в СОК. При этом ключи, необходимые для дальнейшего расшифрования, остаются у пользователя и недоступны для поставщиков облачных сервисов.

3.2. Оптимизированная схема хранения в сети блокчейна

Блокчейн — это распределенная база данных (реестр), представляющая собой выстроенную определенным образом непрерывную цепочку блоков с информацией, каждый из которых содержит свою собственную хеш-сумму и хеш-сумму предыдущего блока. Для обеспечения согласованности данных и их защиты от неправомерного изменения, копии цепочек блоков хранятся на множестве независимых друг от друга узлов сети. Первой известной реализацией блокчейна стала пиринговая платежная система Bitcoin, появившаяся в октябре 2008 года. С тех пор блокчейн-технологии привлекают к себе все больше внимания и

находят применение не только в банковском секторе, инвестициях и биржах, но также в медицине, образовании и государственном секторе.

Одним из главных узких мест блокчейна является огромный объем данных, которые необходимо хранить на каждом узле сети. Например, по состоянию на конец марта 2021 г., объем сети Bitcoin составлял более 335 ГБ. Поэтому оптимизация хранения данных без ущерба децентрализации и без изменения общей структуры цепочки блоков является важной проблемой технологии блокчейн.

Н. Меi и др. предложили [6] оптимизированный механизм хранения данных блокчейна, функционирующий на базе распределенной СОК с набором модулей $\{m_1, m_2, \dots, m_n\}$. При подключении к сети новый узел выбирает один модуль m_i и хранит данные транзакций только по этому модулю, т.е. разные узлы хранят одни и те же данные, но по разным модулям. Полагая, что полные данные составляют k бит, а размер модуля — b бит, степень сжатия составляет b/k . Поскольку операции сложения и умножения в СОК выполняются независимо по разным модулям, каждый узел обновляет данные транзакций только по своему локальному модулю и взаимодействие между узлами не требуется. В свою очередь, для проверки достоверности данных транзакции узел собирает недостающие остатки по всем модулям с других узлов и восстанавливает их с использованием метода New CRT II, который рассматривался в разделе 2.1.3.

Помимо оптимизированного механизма хранения, в [6] предложена гибридная схема организации узлов, обеспечивающая консенсус в сети блокчейна (рисунок 2).

Согласно этой схеме, вся сеть узлов логически разбивается на три подсети. Первые две подсети, образованные узлами по модулям m_1 и m_2 , имеют небольшой масштаб. Консенсус в этих подсетях обеспечивается алгоритмом Raft [86].

Третья подсеть формируется всеми другими узлами и синхронизируется на основе метода New CRT II. Зная, что данные по модулям m_1 и m_2 являются достоверными, New CRT II позволяет рекурсивно обнаружить некорректные остатки по оставшимся модулям m_3, m_4, \dots, m_n , тем самым выявить узлы, которые ненамеренно (вследствие сбоя) или намеренно (по злому умыслу) предоставляют недостоверные данные.

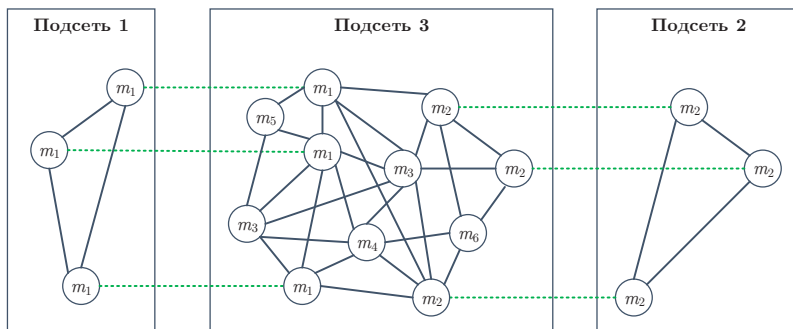


РИСУНОК 2. Отказоустойчивая схема сети блокчейна на основе СОК (адаптация рисунка из [6])

3.3. Вычисления многократной точности

Арифметика с плавающей точкой является основой научных вычислений. Форматы чисел одинарной и двойной точности IEEE 754 (binary32 и binary64, соответственно) обеспечивают высокую производительность на современных компьютерах, но вместе с тем приводят к ошибке округления почти в каждой арифметической операции. Для многих типовых задач эти ошибки не препятствуют получению корректного результата. Вместе с тем, в связи с быстрым ростом масштаба вычислений, возникает все больше крупных задач, для которых точности стандартных форматов IEEE 754 оказывается недостаточно. В этом случае распространенным подходом является арифметика многократной точности [87], позволяющая выполнять операции с числами, разрядность которых превышает стандартные форматы, а в общем случае ограничена лишь объемом доступной памяти вычислительной системы.

Традиционный формат чисел многократной точности предполагает представление мантиссы в виде массива взвешенных цифр по фиксированному основанию, обычно равному некоторой степени двойки. Цифры мантиссы являются числами машинной точности [88]. Одно из узких мест алгоритмов, основанных на таком представлении — необходимость распространения переносов между цифрами мантиссы, приводящая к сильному ветвлению и медленной работе алгоритмов.

В этом отношении привлекательной альтернативой является СОК, в которой между остатками числа не возникает переносов.

В работе [20] использован формат представления чисел с плавающей точкой многократной точности на основе СОК, изображенный на рисунке 3.

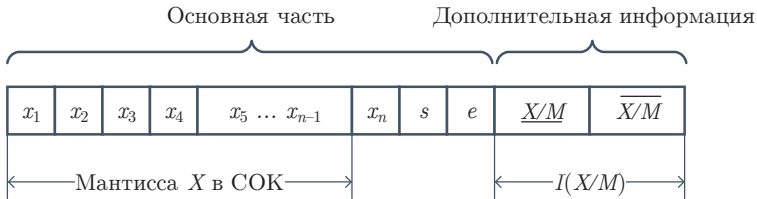


РИСУНОК 3. Формат с плавающей точкой многократной точности

Число x представляется в виде объекта, состоящего из знака s , мантиссы X , целочисленного порядка (экспоненты) e , а также дополнительной информации о величине мантиссы — ее интервальной оценке $I(X/M) = [\underline{X/M}, \overline{X/M}]$.

Знак интерпретируется так же, как и в двоичном дополнительном коде: знак равен единице, когда x положительное и нулю, когда оно отрицательное. Мантисса выражает абсолютное значение числа x и представлена в СОК остатками x_1, x_2, \dots, x_n относительно набора модулей $\{m_1, m_2, \dots, m_n\}$. Остатки — обычные целые числа в дополнительном коде.

Значение числа можно вычислить, используя CRT формулу (1):

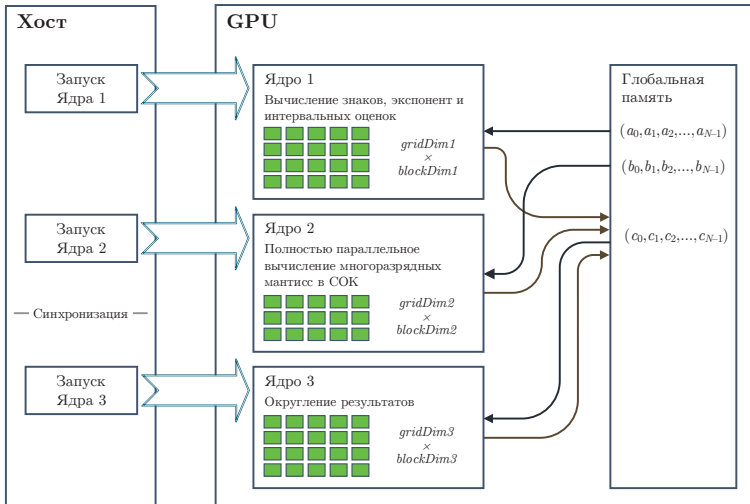
$$x = (-1)^s \times \left| \sum_{i=1}^n M_i |x_i w_i|_{m_i} \right|_M \times 2^e.$$

Для быстрой оценки величины мантиссы при выполнении немодульных операций в формат числа дополнительно включена ее интервальная оценка $I(X/M)$, рассмотренная в подразделе 2.2. Интервальная оценка вычисляется по остаткам (x_1, x_2, \dots, x_n) при преобразовании числа в многоразрядный формат. В свою очередь, при выполнении какой-либо арифметической операции интервальная оценка результата вычисляется с использованием формул интервальной арифметики за время $O(1)$, но также при необходимости может быть заново вычислена по остаткам

(x_1, x_2, \dots, x_n) . Границы $\underline{X/M}$ и $\overline{X/M}$ являются числами с плавающей точкой стандартной разрядности, но с расширенным диапазоном экспоненты, что предотвращает потерю значимости в случае большой величины M (при $M < 2^{1000}$ для хранения границ достаточно стандартного формата binary64).

В [20] представлены алгоритмы выполнения основных арифметических операций в представленном числовом формате. Точность этих операций с учетом округления составляет $\approx \lfloor \log_2 \sqrt{M} \rfloor - 1$ бит. Таким образом, желаемая точность может быть достигнута выбором соответствующего набора модулей.

Также в [20] предложены оптимизированные алгоритмы для покомпонентной обработки векторов многократной точности с полностью параллельным вычислением всех цифр многоразрядных мантисс, ориентированные на GPU. В этих алгоритмах арифметические операции разбиваются на части, каждая из которых выполняется как отдельное CUDA ядро, как показано на рисунке 4. Впоследствии алгоритмы были



Рисунком 4. Параллельная покомпонентная обработка векторов многократной точности на GPU (адаптация рисунка из [20])

расширены и адаптированы для вычислений с матрицами, и на их основе разработана библиотека базовых подпрограмм линейной алгебры многократной точности для GPU — MPRES-BLAS [19]. Эксперименты показали, что во многих случаях подпрограммы из MPRES-BLAS выполняются быстрее при одинаковой точности, чем аналогичные реализации, построенные на базе существующих позиционных библиотек многократной точности для CPU и GPU.

3.4. Глубокие нейронные сети

Глубокое обучение является одной из наиболее ярких технологий настоящего времени и применяется во многих сферах человеческой деятельности, связанных с обработкой больших объемов данных, включая поиск новых лекарственных препаратов, анализ медицинских изображений, прогнозирование качества сна, обнаружение финансового мошенничества и борьбу с отмыванием денег, предоставление таргетированной рекламы, распознавание речи, восстановление изображений и обработку естественного языка.

Однако применение глубоких нейронных сетей требует высокой пропускной способности памяти, так как вначале необходимо выполнить загрузку всех обученных весов, а затем сохранить результаты работы сети. Интенсивный обмен данными между процессором и памятью приводит к затратам времени и повышенному энергопотреблению и зачастую оказывается основным узким местом, особенно в условиях применения нейронных сетей в мобильных устройствах. Одним из многообещающих подходов к решению данной проблемы является концепция «вычислений в памяти» (processing-in-memory), предполагающая выполнение операций с данными непосредственно в памяти, т.е. без их загрузки в регистры процессора.

В работе [12] авторы использовали СОК для реализации глубокой нейронной сети RNSnet, работающей непосредственно в памяти устройства. RNSnet упрощает основные операции, выполняемые в сети, сводя их к сложению/вычитанию и выборке из памяти. В представленной реализации все входные данные и обученные веса преобразуются в СОК; функциональность каждого нейрона внутреннего слоя также реализована в СОК и не предполагает какого-либо обратного преобразования.

Использование набора модулей вида $\{2^t - 1, 2^t, 2^t + 1\}$ позволило реализовать непосредственно в памяти все необходимые арифметические операции в RNSnet, в том числе умножение входов и весов нейронов. Умножение выполняется посредством нескольких сложений, вычитаний и выборок предварительно вычисленных значений. Сеть RNSnet поддерживает линейные и нелинейные функции активации, причем нелинейные функции аппроксимируются с использованием нескольких первых членов разложения в ряд Тейлора. Для слоя пулинга и некоторых функций активации, например ReLU, требуется выполнение операции сравнения. Поэтому авторы реализовали эффективный алгоритм сравнения чисел в СОК, использующий свойства модулей $2^t \pm 1$.

Ускоритель RNSnet реализован на языке описания аппаратуры SystemVerilog и синтезирован с использованием компилятора Synopsys Design по 45-нм технологии. Представленные в [12] экспериментальные результаты показывают, что использование СОК позволяет гибко настроить разрядность вычислений под конкретную задачу, решаемую нейронной сетью. Это обеспечивает более высокую производительность и низкое энергопотребление, а также существенную экономию памяти. На некоторых задачах RNSnet показала ускорение до 35 раз и снижение энергопотребления до 145 раз по сравнению с GPU реализациями глубоких нейронных сетей.

4. Заключение

Система остаточных классов — перспективная система счисления, популярность которой возрастает с развитием параллельных вычислительных архитектур. Независимость вычислений с остатками для операций сложения, вычитания и умножения делает эту систему привлекательным инструментом для многих приложений, критичных к скорости вычислений. С другой стороны, использование СОК на двоичных компьютерах требует поддержки эффективного обратного преобразования, а также сравнения, вычисления знака, деления и ряда других операций, которые являются сложными в силу непозиционной структуры системы.

Недостатком классического метода обратного преобразования является необходимость редукции по большому модулю M . Методы на основе техники New CRT II, диагональных и монотонных функций, хотя и не исключают полностью операцию редукции по модулю, но позволяют снизить ее размер до \sqrt{M} , SQ и M_I , соответственно. Все эти методы требуют выполнения $O(\log n)$ операций при параллельной реализации. Метод на основе перехода к системе счисления по смешанным основаниям (MRC) исключает полностью операцию редукции, однако является медленным при большом размере набора модулей в силу своего последовательного характера, обладая сложностью $O(n)$ при конвейерной реализации.

В связи с этим наиболее эффективным представляется метод на основе техники MR CRT, который не требует редукции по большому модулю и обладает сложностью $O(\log n)$. С другой стороны, если поддерживаются быстрые операции с дробными числами (в форматах с фиксированной или с плавающей точкой), то перспективными также выглядят методы, основанные на дробной версии CRT. В этих методах редукция по модулю M заменяется редукцией по модулю 1, которая сводится к простому отбрасыванию целой части вычисленной суммы. Однако из-за ошибок округления эти методы требуют вычислений с дробными величинами, разрядность которых превышает M .

Операции сравнения, определения знака и контроля переполнения диапазона играют ключевую роль при реализации высокопроизводительных алгоритмов в СОК на вычислительных системах общего назначения, таких как CPU и GPU. Ввиду ограниченной разрядной сетки, поддержка многоразрядной арифметики в таких системах реализуется посредством программной эмуляции и оказывается крайне затратной. Поэтому важно иметь возможность сравнения величин и выполнения связанных операций в СОК с использованием только небольших арифметических операций, ограниченных разрядной сеткой. Подходящим вариантом в данном контексте является MRC метод, требующий только операций по модулям m_i для вычисления всех цифр смешанных представлений чисел. В свою очередь, почти все современные системы общего назначения поддерживают быстродействующую арифметику с плавающей точкой IEEE 754. Поэтому в качестве более эффективной альтернативы MRC может быть рекомендован

метод, основанный на вычислении интервальной оценки для дробного представления числа в СОК.











Производительность алгоритмов деления в СОК определяется соотношением между делимым и делителем, поскольку большинство алгоритмов деления позволяют за одну итерацию вычислить одну либо несколько цифр частного Q и, таким образом, обладают сложностью $O(\log Q)$. Среди различных вариантов деления популярны алгоритмы, основанные на вычислении и анализе старших (наиболее значимых) разрядов делимого и делителя. Для определения наиболее значимых разрядов числа X в СОК может быть использовано его дробное представление X/M , причем в аппаратных реализациях одно и то же устройство, вычисляющее X/M , может быть использовано как для деления, так и для обратного преобразования. В свою очередь, как уже упоминалось выше, в программных реализациях использование многоразрядных дробных представлений неэффективно, поэтому предпочтительны алгоритмы деления, вычисляющие приближенное (округленное) дробное представление. В любом случае операция деления является одной из наиболее трудоемких для СОК, поэтому следует по возможности избегать множественных делений, например, выполняя определенные модификации алгоритма.

Последние исследования показывают, что эффективное использование СОК уже не ограничивается задачами цифровой обработки сигналов и криптографии. Высоконадежные облачные хранилища, блокчейн-технологии, вычисления многократной точности и глубокие нейронные сети представляют далеко не полный перечень областей, в которых применение параллельной арифметики на основе СОК является целесообразным. Поэтому важно продолжать исследования, направленные на создание эффективных аппаратных и программных реализаций, позволяющих преодолеть основные узкие места этой непозиционной системы счисления.

Список литературы












- [1] P. V. Ananda Mohan. *Residue Number Systems: Theory and Applications*, Birkhäuser, Basel, 2016, ISBN 978-3-319-41385-3, x+351 pp.  [↑](#) 138, 151
- [2] A. Omondi, B. Premkumar. *Residue Number Systems: Theory And*

- Implementation*, Imperial College Press, London, UK, 2007, ISBN 978-1-86094-866-4. [↑_{138,140,142}](#)
- [3] A. S. Molahosseini, L. Sousa. “Introduction to residue number system: structure and teaching methodology”, *Embedded Systems Design with Special Arithmetic and Number Systems*, eds. A. S. Molahosseini, L. S. de Sousa, C. Chang, Springer, Cham, 2017, ISBN 978-3-319-49742-6, pp. 3–17. [doi](#) [↑₁₃₈](#)
- [4] И. Я. Акушинский, Д. И. Юдицкий. *Машинная арифметика в остаточных классах*, Сов. радио, М., 1968, 439 с. [↑₁₃₈](#)
- [5] Z. Guo, Z. Gao, H. Mei, M. Zhao, J. Yang. “Design and optimization for storage mechanism of the public blockchain based on redundant residual number system”, *IEEE Access*, **7** (2019), pp. 98546–98554. [doi](#) [↑_{138,145}](#)
- [6] H. Mei, Z. Gao, Z. Guo, M. Zhao, J. Yang. “Storage mechanism optimization in blockchain system based on residual number system”, *IEEE Access*, **7** (2019), pp. 114539–114546. [doi](#) [↑_{138,168,169}](#)
- [7] A. Qaisar Ahmad Al Badawi, Y. Polyakov, K. M. M. Aung, B. Veeravalli, K. Rohloff. “Implementation and performance evaluation of RNS variants of the BFV homomorphic encryption scheme”, *IEEE Transactions on Emerging Topics in Computing*, **9:2** (2019), pp. 941–956. [doi](#) [↑_{138,166}](#)
- [8] J. Bajard, J. Eynard, M. A. Hasan, V. Zucca. “A full RNS variant of FV like somewhat homomorphic encryption schemes”, *Selected Areas in Cryptography*, SAC 2016, Lecture Notes in Computer Science, vol. **10532**, eds. R. Avanzi, H. Heys, Springer International Publishing, Cham, 2017, ISBN 978-3-319-69453-5, pp. 423–442. [doi](#) [↑_{138,166}](#)
- [9] K. Givaki, R. Hojabr, M. H. Najafi, A. Khonsari, M. H. Gholamrezayi, S. Gorgin, D. Rahmati. “Using residue number systems to accelerate deterministic bit-stream multiplication”, *Proc. 2019 IEEE 30th International Conference on Application-Specific Systems, Architectures and Processors*, ASAP (15–17 July 2019, New York, NY, USA), 2019, pp. 40. [doi](#) [↑₁₃₈](#)
- [10] M. Villari, A. Celesti, F. Tusa, A. Puliafito. “Data reliability in multi-provider cloud storage service with RRNS”, *Advances in Service-Oriented and Cloud Computing*, Communications in Computer and Information Science, vol. **393**, eds. C. Canal, M. Villari, Springer, Berlin–Heidelberg, 2013, ISBN 978-3-642-45364-9, pp. 83–93. [doi](#) [↑_{138,166}](#)
- [11] A. Celesti, M. Fazio, M. Villari, A. Puliafito. “Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems”, *Journal of Network and Computer Applications*, **59** (2016), pp. 208–218. [doi](#) [↑_{138,167}](#)
- [12] S. Salamat, M. Imani, S. Gupta, T. Rosing. “RNSnet: in-Memory neural network acceleration using residue number system”, *Proc. 2018 IEEE International Conference on Rebooting Computing*, ICRC, 7–9 Nov. 2018, McLean, VA, USA, pp. 1–12. [doi](#) [↑_{139,172,173}](#)














- [13] N. Samimi, M. Kamal, A. Afzali-Kusha, M. Pedram. “Res-DNN: A residue number system-Based DNN accelerator unit”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, **67**:2 (2020), pp. 658–671.  [↑₁₃₉](#)
- [14] O. L. Usman, R. C. Muniyandi. “CryptoDL: predicting dyslexia biomarkers from encrypted neuroimaging dataset using energy-efficient residue number system and deep convolutional neural network”, *Symmetry*, **12**:5 (2020), 836, 24 pp.  [↑₁₃₉](#)
- [15] P. V. Ananda Mohan. “RNS-Based arithmetic circuits and applications”, *Arithmetic Circuits for DSP Applications*, Ch. 6, eds. P. K. Meher, T. Stouraitis, John Wiley and Sons, Ltd, 2017, ISBN 9781119206804, pp. 186–236.  [↑_{139,142,166}](#)
- [16] G. C. Cardarilli, A. Nannarelli, M. Re. “RNS applications in digital signal processing”, *Embedded Systems Design with Special Arithmetic and Number Systems*, eds. A. S. Molahosseini, L. S. de Sousa, C. Chang, Springer, Cham, 2017, ISBN 978-3-319-49742-6, pp. 181–215.  [↑_{139,166}](#)
- [17] D. Younes, P. Steffan. “Efficient image processing application using residue number system”, *Proc. 20th International Conference Mixed Design of Integrated Circuits and Systems*, MIXDES 2013, 20–22 June 2013, Gdynia, Poland, pp. 468–472. [↑₁₃₉](#)
- [18] E. Ochoa-Jiménez, L. Rivera-Zamarripa, N. Cruz-Cortés, F. Rodríguez-Henríquez. “Implementation of RSA signatures on GPU and CPU architectures”, *IEEE Access*, **8** (2020), pp. 9928–9941.  [↑_{139,149,166}](#)
- [19] K. Isupov, V. Knyazkov. “Multiple-Precision BLAS library for graphics processing units”, *RuSCDays 2020: Supercomputing*, Communications in Computer and Information Science, vol. **1331**, Springer International Publishing, Cham, 2020, ISBN 978-3-030-64616-5, pp. 37–49.  [↑_{139,172}](#)
- [20] K. Isupov, V. Knyazkov, A. Kuvaev. “Design and implementation of multiple-precision BLAS level 1 functions for graphics processing units”, *Journal of Parallel and Distributed Computing*, **140** (2020), pp. 25–36.  [↑_{139,170,171}](#)
- [21] V. Shoup. *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press, Cambridge, UK, 2005, ISBN 9781139165464.  [↑_{140,142}](#)
- [22] V. T. Goh, M. U. Siddiqi. “Multiple error detection and correction based on redundant residue number systems”, *IEEE Transactions on Communications*, **56**:3 (2008), pp. 325–330.  [↑₁₄₁](#)
- [23] L. Sousa. “Nonconventional computer arithmetic circuits, systems and applications”, *IEEE Circuits and Systems Magazine*, **21**:1 (2021), pp. 6–40.  [↑₁₄₁](#)






- [24] C. Chang, A. S. Molahosseini, A. A. E. Zarandi, T. F. Tay. “Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications”, *IEEE Circuits and Systems Magazine*, **15**:4 (2015), pp. 26–44. [doi](#) [↑]₁₄₂
- [25] N. S. Szabo, R. I. Tanaka. *Residue Arithmetic and its Application to Computer Technology*, McGraw-Hill, New York, USA, 1967, ISBN 978-0070626591, 236 pp. [↑]_{142,143,161}
- [26] D. E. Knuth. *The Art of Computer Programming. V. 2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, USA, 1997, ISBN 0201896842, 784 pp. [↑]_{142,144}
- [27] H. M. Yassine, W. R. Moore. “Improved mixed-radix conversion for residue number system architectures”, *IEE Proceedings G (Circuits, Devices and Systems)*, **138**:1 (1991), pp. 120–124. [doi](#) [↑]₁₄₄
- [28] M. Akkal, P. Siy. “A new mixed radix conversion algorithm MRC-II”, *Journal of Systems Architecture*, **53**:9 (2007), pp. 577–586. [doi](#) [↑]₁₄₄
- [29] Y. Wang. “New Chinese remainder theorems”, *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers. V. 1* (1–4 Nov. 1998, Pacific Grove, CA, USA), 1998, ISBN 0-7803-5148-7, pp. 165–171. [doi](#) [↑]₁₄₅
- [30] Y. Wang. “Residue-to-binary converters based on new Chinese remainder theorems”, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, **47**:3 (2000), pp. 197–205. [doi](#) [↑]_{145,151}
- [31] W. Zhang, P. Siy. “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*, **178**:1 (2008), pp. 264–279. [doi](#) [↑]₁₄₅
- [32] A. S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, S. Timarchi. “Efficient reverse converter designs for the new 4-Moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ and $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n} + 1\}$ based on new CRTs”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, **57**:4 (2010), pp. 823–835. [doi](#) [↑]₁₄₅
- [33] S. Bi, W. Gross. “The mixed-Radix Chinese remainder theorem and its applications to residue comparison”, *IEEE Transactions on Computers*, **57**:12 (2008), pp. 1624–1632. [doi](#) [↑]_{146,153,160}
- [34] G. Dimauro, S. Impedovo, G. Pirlo. “A new technique for fast number comparison in the residue number system”, *IEEE Transactions on Computers*, **42**:5 (1993), pp. 608–612. [doi](#) [↑]_{146,153,160}
- [35] G. Dimauro, S. Impedovo, G. Pirlo, A. Salzo. “RNS architectures for the implementation of the ‘diagonal function’”, *Information Processing Letters*, **73**:5–6 (2000), pp. 189–198. [doi](#) [↑]₁₄₆

- [36] P. V. Ananda Mohan. “RNS to binary conversion using diagonal function and Pirlo and Impedovo monotonic function”, *Circuits, Systems, and Signal Processing*, **35**:3 (2016), pp. 1063–1076. doi ↑_{147,148}
- [37] G. Pirlo, D. Impedovo. “A new class of monotone functions of the residue number system”, *International Journal of Mathematical Models and Methods in Applied Sciences*, **7**:9 (2013), pp. 802–809. URL ↑_{147,148}
- [38] M. Soderstrand, C. Vernia, J. Chang. “An improved residue number system digital-to-analog converter”, *IEEE Transactions on Circuits and Systems*, **30**:12 (1983), pp. 903–907. doi ↑₁₄₈
- [39] M. Lu, J. Chiang. “A novel division algorithm for the residue number system”, *IEEE Transactions on Computers*, **41**:8 (1992), pp. 1026–1032. doi ↑_{149,154,155,160,161,165}
- [40] K. Isupov, V. Knyazkov. “Interval estimation of relative values in residue number system”, *Journal of Circuits, Systems and Computers*, **27**:1 (2018), pp. 1850004. doi ↑_{149,150,157,160}
- [41] Н. И. Червяков, М. Г. Бабенко, П. А. Ляхов, И. Н. Лавриненко. «Приближенный метод сравнения модулярных чисел и его применение для деления чисел в системе остаточных классов», *Кибернетика и системный анализ*, **50**:6 (2014), с. 176–186. ↑_{149,155,160,162}
- [42] Jin Yul Kim, Kyu Ho Park, Hwang Soo Lee. “Efficient residue-to-binary conversion technique with rounding error compensation”, *IEEE Transactions on Circuits and Systems*, **38**:3 (1991), pp. 315–317. doi ↑₁₄₉
- [43] H. Brönnimann, I. Z. Emiris, V. Y. Pan, S. Pion. “Sign determination in residue number systems”, *Theoretical Computer Science*, **210**:1 (1999), pp. 173–197. doi ↑_{149,157,160}
- [44] A. A. Hiasat, H. S. Abdel-Aty-Zohdy. “Design and implementation of an RNS division algorithm”, *Proc. 13th IEEE Symposium on Computer Arithmetic* (6–9 July 1997, Asilomar, CA, USA), 1997, ISBN 0-8186-7846-1, pp. 240–249. doi ↑_{149,161,162,163,165}
- [45] А. С. Коржавина, В. С. Князьков. «Реализация высокоточных вычислений в базисе модулярно-интервальной арифметики», *Программные системы: теория и приложения*, **10**:3(42) (2019), с. 81–127. doi ↑_{149,150}
- [46] C. Y. Hung, B. Parhami. “An approximate sign detection method for residue numbers and its application to RNS division”, *Computers & Mathematics with Applications*, **27**:4 (1994), pp. 23–35. doi ↑_{149,156,157,160,162,165}
- [47] J. Yang, C. Chang, C. Chen. “A high-speed division algorithm in residue number system using parity-checking technique”, *International Journal of Computer Mathematics*, **81**:6 (2004), pp. 775–780. doi ↑_{149,162,165}

- [48] R. Conway, J. Nelson. “New CRT-based RNS converter using restricted moduli set”, *IEEE Transactions on Computers*, **52**:5 (2003), pp. 572–578.  [↑_{149,151}](#)
- [49] K. Isupov. “Using floating-point intervals for non-modular computations in residue number system”, *IEEE Access*, **8** (2020), pp. 58603–58619.  [↑_{149,150,159,163,165}](#)
- [50] S. Arthireena, G. Shanmugavadivel. “Efficient sign detection using parallel prefix adder”, *Proc. 2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering*, ICEICE, IEEE, 27–28 April 2017, Karur, India, pp. 1–5.  [↑₁₄₉](#)
- [51] S. Kawamura, M. Koike, F. Sano, A. Shimbo. “Cox-Rower architecture for fast parallel Montgomery multiplication”, *Advances in Cryptology, EUROCRYPT 2000, Lecture Notes in Computer Science*, vol. **1807**, ed. B. Preneel, Springer, Berlin–Heidelberg, 2000, ISBN 978-3-540-45539-4, pp. 523–538.  [↑₁₄₉](#)
- [52] T. V. Vu. “Efficient implementations of the Chinese remainder theorem for sign detection and residue decoding”, *IEEE Transactions on Computers*, **C-34**:7 (1985), pp. 646–651.  [↑_{149,154,155,160}](#)
- [53] S. J. Meehan, S. D. O’Neil, J. J. Vaccaro. “An universal input and output RNS converter”, *IEEE Transactions on Circuits and Systems*, **37**:6 (1990), pp. 799–803.  [↑₁₄₉](#)
- [54] N. Chervyakov, A. S. Molahosseini, P. A. Lyakhov, M. G. Babenko, M. A. Deryabin. “Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem”, *International Journal of Computer Mathematics*, **94**:9 (2017), pp. 1833–1849.  [↑_{149,151}](#)
- [55] N. Whitehead, A. Fit-Florea. *Precision & performance: floating point and IEEE 754 compliance for NVIDIA GPUs*, 2011 (Accessed 20 March 2021), 7 pp.  [↑₁₅₀](#)
- [56] N. Burgess. “Scaled and unscaled residue number system to binary conversion techniques using the core function”, *Proc. 13th IEEE Symposium on Computer Arithmetic*, 6–9 July 1997, Asilomar, CA, USA, ISBN 0-8186-7846-1, pp. 250–257.  [↑₁₅₀](#)
- [57] Shaoqiang Bi, Wei Wang, A. Al-Khalili. “New modulo decomposed residue-to-binary algorithm for general moduli sets”, *Proc. 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing. V. V*, 17–21 May 2004, Montreal, QC, Canada, ISBN 0-7803-8484-9, pp. 141–144.  [↑₁₅₀](#)
- [58] A. Omondi. “Fast residue-to-binary conversion using base extension and the Chinese remainder theorem”, *Journal of Circuits, Systems and Computers*, **16**:3 (2007), pp. 379–388.  [↑₁₅₀](#)

- [59] N. Burgess. “Efficient RNS to binary conversion using high-radix SRT division”, *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers*. V. 2, 1–4 Nov. 1998, Pacific Grove, CA, USA, 1998, ISBN 0-7803-5148-7, pp. 1240–1243. doi ↑₁₅₀
- [60] J. Chen, R. Yao. “Efficient CRT-based residue-to-binary converter for the arbitrary moduli set”, *Science China Information Sciences*, **54**:1 (2011), pp. 70–78. doi ↑_{150,151}
- [61] P. Patronik. “On reverse converters for arbitrary multi-moduli RNS”, *Integration*, **75** (2020), pp. 158–167. doi ↑₁₅₁
- [62] A. Hiasat. “A residue-to-binary converter with an adjustable structure for an extended RNS three-moduli set”, *Journal of Circuits, Systems and Computers*, **28**:08 (2019), 1950126. doi ↑₁₅₁
- [63] L. Sousa, S. Antao. “MRC-based RNS reverse converters for the four-moduli sets $\{2^n + 1, 2^n - 1, 2^n, 2^{2n+1} - 1\}$ and $\{2^n + 1, 2^n - 1, 2^{2n}, 2^{2n+1} - 1\}$ ”, *IEEE Transactions on Circuits and Systems II: Express Briefs*, **59**:4 (2012), pp. 244–248. doi ↑₁₅₁
- [64] P. V. Ananda Mohan. “Reverse conversion using core function, CRT and mixed radix conversion”, *Circuits, Systems, and Signal Processing*, **36**:7 (2017), pp. 2847–2874. doi ↑₁₅₁
- [65] E. K. Bankas, K. A. Gbolagade. “An efficient VLSI design of residue to binary converter circuit for a new moduli set $\{2^{2n}, 2^{2n-1} - 1, 2^{2n-1} + 1\}$ ”, *Proc. 2019 IEEE 4th International Conference on Integrated Circuits and Microsystems*, ICICM (25–27 Oct. 2019, Beijing, China), pp. 24–28. doi ↑₁₅₁
- [66] A. S. Molahosseini, A. A. E. Zarandi, P. Martins, L. Sousa. “A multifunctional unit for designing efficient RNS-Based datapaths”, *IEEE Access*, **5** (2017), pp. 25972–25986. doi ↑₁₅₁
- [67] Y. Wang, X. Song, M. Aboulhamid. “A new algorithm for RNS magnitude comparison based on New Chinese remainder theorem II”, *Proc. Ninth Great Lakes Symposium on VLSI* (4–6 March 1999, Ypsilanti, MI, USA), ISBN 0-7695-0104-4, pp. 362–365. doi ↑_{153,160}
- [68] S. J. Piestrak. “A note on RNS architectures for the implementation of the diagonal function”, *Information Processing Letters*, **115**:4 (2015), pp. 453–457. doi ↑₁₅₄
- [69] G. Pirlo. “Non-modular operations of the residue number system: functions for computing”, *Embedded Systems Design with Special Arithmetic and Number Systems*, eds. A. S. Molahosseini, L. S. de Sousa, C. Chang, Springer, Cham, 2017, ISBN 978-3-319-49742-6, pp. 49–64. doi ↑₁₅₄
- [70] M. Babenko, M. Deryabin, S. J. Piestrak, P. Patronik, N. Chervyakov, A. Tchernykh, A. Avetisyan. “RNS number comparator based on a modified diagonal function”, *Electronics*, **9**:11 (2020), 1784, 16 pp. doi ↑_{155,160}



- [71] K. Isupov. “High-performance computation in residue number system using floating-point arithmetic”, *Computation*, **9**:2 (2021), 9, 15 pp.  [↑₁₅₉](#)
- [72] D. S. Phatak, S. D. Houston. “New distributed algorithms for fast sign detection in residue number systems (RNS)”, *Journal of Parallel and Distributed Computing*, **97** (2016), pp. 78–95.  [↑_{159,160}](#)
- [73] A. A. Hiasat, H. Abdel-Aty-Zohdy. “Semi-Custom VLSI design and implementation of a new efficient RNS division algorithm”, *The Computer Journal*, **42**:3 (1999), pp. 232–240.  [↑₁₆₂](#)
- [74] C. Chang, J. Yang. “A division algorithm using bisection method in residue number system”, *International Journal of Computer, Consumer and Control (IJ3C)*, **2**:1 (2013), pp. 59–66.  [↑_{162,165}](#)
- [75] N. Chervyakov, P. Lyakhov, M. Babenko, A. Nazarov, M. Deryabin, I. Lavrinenko, A. Lavrinenko. “A high-speed division algorithm for modular numbers based on the Chinese remainder theorem with fractions and its hardware implementation”, *Electronics*, **8**:3 (2019), 261, 17 pp.  [↑_{162,165}](#)
- [76] M. A. Hitz, E. Kaltofen. “Integer division in residue number systems”, *IEEE Transactions on Computers*, **44**:8 (1995), pp. 983–989.  [↑_{163,165}](#)
- [77] M. Mccann, N. Pippenger. “SRT division algorithms as dynamical systems”, *Proc. 2003 16th IEEE Symposium on Computer Arithmetic* (15–18 June 2003, Santiago de Compostela, Spain), ISBN 0-7695-1894-X, pp. 46–53.  [↑₁₆₂](#)
- [78] W. A. Chren Jr.. “A new residue number system division algorithm”, *Computers & Mathematics with Applications*, **19**:7 (1990), pp. 13–29.  [↑_{164,165}](#)
- [79] D. K. Banerji, T. Cheung, V. Ganesan. “A high-speed division method in residue arithmetic”, 5th IEEE Symposium on Computer Arithmetic (18-19 May 1981, Ann Arbor, Michigan, USA), 1981, pp. 158–164.  [↑_{164,165}](#)
- [80] D. Gamberger. “New approach to integer division in residue number systems”, *Proc. 10th IEEE Symposium on Computer Arithmetic* (26–28 June 1991, Grenoble, France), ISBN 0-8186-9151-4, pp. 84–91.  [↑_{164,165}](#)
- [81] C. Chang, Y. Lai. “A division algorithm for residue numbers”, *Appl. Math. Comput.*, **172**:1 (2006), pp. 368–378.  [↑_{164,165}](#)
- [82] N. Mukhtar, L. Papachristodoulou, A. P. Fournaris, L. Batina, Y. Kong. *Machine-learning assisted side-channel attacks on RNS-based elliptic curve implementations using hybrid feature engineering*, Cryptology ePrint Archive, Report 2020/1065, 2020, 31 pp.  [↑₁₆₆](#)
- [83] S. Kawamura, Y. Komano, H. Shimizu, T. Yonemura. “RNS Montgomery reduction algorithms using quadratic residuosity”, *Journal of Cryptographic Engineering*, **9**:4 (2019), pp. 313–331.  [↑₁₆₆](#)

- [84] D. Schoinianakis. “Residue arithmetic systems in cryptography: a survey on modern security applications”, *Journal of Cryptographic Engineering*, **10**:3 (2020), pp. 249–267.  [↑₁₆₆](#)
- [85] P. Martins, L. Sousa. “The role of non-positional arithmetic on efficient emerging cryptographic algorithms”, *IEEE Access*, **8** (2020), pp. 59533–59549.  [↑₁₆₆](#)
- [86] D. Ongaro, J. Ousterhout. “In search of an understandable consensus algorithm”, *Proceedings of USENIX ATC’14: 2014 USENIX Annual Technical Conference* (June 19–20, 2014, Philadelphia, PA, USA), ISBN 978-1-931971-10-2, pp. 305–320.  [↑₁₆₈](#)
- [87] D. Bailey, J. Borwein. “High-precision arithmetic in mathematical physics”, *Mathematics*, **3**:2 (2015), pp. 337–367.  [↑₁₆₉](#)
- [88] R. Brent, P. Zimmermann. *Modern Computer Arithmetic*, Cambridge University Press, Cambridge, 2010, ISBN 9780511921698.  [↑₁₆₉](#)

Поступила в редакцию 02.04.2021
 Переработана 22.05.2021
 Опубликовано 28.06.2021

Рекомендовал к публикации *проф. Н.Н. Непейвода*

Пример ссылки на эту публикацию:

К. С. Исупов. «Высокопроизводительные вычисления с использованием системы остаточных классов». *Программные системы: теория и приложения*, 2021, **12**:2(49), с. 137–192.  [10.25209/2079-3316-2021-12-2-137-192](#)
 http://psta.psiras.ru/read/psta2021_2_137-192.pdf

Об авторе:



Константин Сергеевич Исупов

Кандидат технических наук, доцент кафедры электронных вычислительных машин Вятского государственного университета. Область научных интересов: высокоточные вычисления, система остаточных классов, компьютерная арифметика, параллельные алгоритмы, программирование для графических процессоров, GPGPU.



0000-0003-0239-0404

e-mail: ks_isupov@vyatsu.ru

CSCSTI 50.33.04

UDC 004.222+004.272

Konstantin S. Isupov. *An overview of high-performance computing using the residue number system.*

ABSTRACT. A residue number system (RNS) is a non-positional number system, an alternative to a binary representation of numbers. In RNS, a large integer is represented as a set of smaller numbers, which are the remainders (“residues”) of dividing its original value by some moduli. An exciting feature of the RNS is that addition, subtraction, and multiplication with each residue are performed independently, which provides parallel, carry-free, and high-speed computer arithmetic. On the other hand, non-modular operations that require estimating the magnitude of a number by its residues are challenging to implement in RNS since there is no parallel form for them.

This paper provides an overview of research on the implementation and practical application of high-performance computational techniques for RNS. More specifically, the paper addresses the following two aspects:

- Existing techniques for performing the most critical non-modular operations: reverse conversion, magnitude comparison, sign identification, and division;
- How RNS arithmetic finds practical application in cloud environments, blockchain technologies, multiple-precision computation, and deep neural networks.

The research was aimed at the wider use of non-positional number systems in resource-intensive applications.

Key words and phrases: system of residual classes, non-modular operations, high performance computing, parallel algorithms.


2020 *Mathematics Subject Classification:* 68W10; 68W35, 68M07

The reported study was funded by RFBR, project number 20-17-50166.

© K. S. ISUPOV, 2021

© VYATKA STATE UNIVERSITY, 2021

© PROGRAM SYSTEMS: THEORY AND APPLICATIONS (DESIGN), 2021

 10.25209/2079-3316-2021-12-2-137-192
















References

- [1] P. V. Ananda Mohan. *Residue Number Systems: Theory and Applications*, Birkhäuser, Basel, 2016, ISBN 978-3-319-41385-3, x+351 pp. [doi](#)↑_{138,151}
- [2] A. Omondi, B. Premkumar. *Residue Number Systems: Theory And Implementation*, Imperial College Press, London, UK, 2007, ISBN 978-1-86094-866-4. [doi](#)↑_{138,140,142}
- [3] A. S. Molahosseini, L. Sousa. “Introduction to residue number system: structure and teaching methodology”, *Embedded Systems Design with Special Arithmetic and Number Systems*, eds. A. S. Molahosseini, L. S. de Sousa, C. Chang, Springer, Cham, 2017, ISBN 978-3-319-49742-6, pp. 3–17. [doi](#)↑₁₃₈
- [4] I. Ya. Akushinskiy, D. I. Yuditskiy. *Computer Arithmetic in Residue Classes*, Sov. radio, M., 1968 (in Russian), 439 pp. [doi](#)↑₁₃₈
- [5] Z. Guo, Z. Gao, H. Mei, M. Zhao, J. Yang. “Design and optimization for storage mechanism of the public blockchain based on redundant residual number system”, *IEEE Access*, **7** (2019), pp. 98546–98554. [doi](#)↑_{138,145}
- [6] H. Mei, Z. Gao, Z. Guo, M. Zhao, J. Yang. “Storage mechanism optimization in blockchain system based on residual number system”, *IEEE Access*, **7** (2019), pp. 114539–114546. [doi](#)↑_{138,168,169}
- [7] A. Qaisar Ahmad Al Badawi, Y. Polyakov, K. M. M. Aung, B. Veeravalli, K. Rohloff. “Implementation and performance evaluation of RNS variants of the BFV homomorphic encryption scheme”, *IEEE Transactions on Emerging Topics in Computing*, **9:2** (2019), pp. 941–956. [doi](#)↑_{138,166}
- [8] J. Bajard, J. Eynard, M. A. Hasan, V. Zucca. “A full RNS variant of FV like somewhat homomorphic encryption schemes”, *Selected Areas in Cryptography, SAC 2016, Lecture Notes in Computer Science*, vol. **10532**, eds. R. Avanzi, H. Heys, Springer International Publishing, Cham, 2017, ISBN 978-3-319-69453-5, pp. 423–442. [doi](#)↑_{138,166}
- [9] K. Givaki, R. Hojabr, M. H. Najafi, A. Khonsari, M. H. Gholamrezayi, S. Gorgin, D. Rahmati. “Using residue number systems to accelerate deterministic bit-stream multiplication”, *Proc. 2019 IEEE 30th International Conference on Application-Specific Systems, Architectures and Processors, ASAP* (15–17 July 2019, New York, NY, USA), 2019, pp. 40. [doi](#)↑₁₃₈
- [10] M. Villari, A. Celesti, F. Tusa, A. Puliafito. “Data reliability in multi-provider cloud storage service with RRNS”, *Advances in Service-Oriented and Cloud Computing, Communications in Computer and Information Science*, vol. **393**, eds. C. Canal, M. Villari, Springer, Berlin–Heidelberg, 2013, ISBN 978-3-642-45364-9, pp. 83–93. [doi](#)↑_{138,166}
- [11] A. Celesti, M. Fazio, M. Villari, A. Puliafito. “Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems”, *Journal of Network and Computer Applications*, **59** (2016), pp. 208–218. [doi](#)↑_{138,167}
- [12] S. Salamat, M. Imani, S. Gupta, T. Rosing. “RNSnet: in-Memory neural network acceleration using residue number system”, *Proc. 2018 IEEE International Conference on Rebooting Computing, ICRC*, 7–9 Nov. 2018, McLean, VA, USA, pp. 1–12. [doi](#)↑_{139,172,173}







- [13] N. Samimi, M. Kamal, A. Afzali-Kusha, M. Pedram. “Res-DNN: A residue number system-Based DNN accelerator unit”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, **67**:2 (2020), pp. 658–671. [doi](#)↑₁₃₉
- [14] O. L. Usman, R. C. Muniyandi. “CryptoDL: predicting dyslexia biomarkers from encrypted neuroimaging dataset using energy-efficient residue number system and deep convolutional neural network”, *Symmetry*, **12**:5 (2020), 836, 24 pp. [doi](#)↑₁₃₉
- [15] P. V. Ananda Mohan. “RNS-Based arithmetic circuits and applications”, *Arithmetic Circuits for DSP Applications*, Ch. 6, eds. P. K. Meher, T. Stouraitis, John Wiley and Sons, Ltd, 2017, ISBN 9781119206804, pp. 186–236. [doi](#)↑_{139, 142, 166}
- [16] G. C. Cardarilli, A. Nannarelli, M. Re. “RNS applications in digital signal processing”, *Embedded Systems Design with Special Arithmetic and Number Systems*, eds. A. S. Molahosseini, L. S. de Sousa, C. Chang, Springer, Cham, 2017, ISBN 978-3-319-49742-6, pp. 181–215. [doi](#)↑_{139, 166}
- [17] D. Younes, P. Steffan. “Efficient image processing application using residue number system”, *Proc. 20th International Conference Mixed Design of Integrated Circuits and Systems*, MIXDES 2013, 20–22 June 2013, Gdynia, Poland, pp. 468–472. ↑₁₃₉
- [18] E. Ochoa-Jiménez, L. Rivera-Zamarripa, N. Cruz-Cortés, F. Rodríguez-Henríquez. “Implementation of RSA signatures on GPU and CPU architectures”, *IEEE Access*, **8** (2020), pp. 9928–9941. [doi](#)↑_{139, 149, 166}
- [19] K. Isupov, V. Knyazkov. “Multiple-Precision BLAS library for graphics processing units”, *RuSCDays 2020: Supercomputing*, Communications in Computer and Information Science, vol. **1331**, Springer International Publishing, Cham, 2020, ISBN 978-3-030-64616-5, pp. 37–49. [doi](#)↑_{139, 172}
- [20] K. Isupov, V. Knyazkov, A. Kuvaev. “Design and implementation of multiple-precision BLAS level 1 functions for graphics processing units”, *Journal of Parallel and Distributed Computing*, **140** (2020), pp. 25–36. [doi](#)↑_{139, 170, 171}
- [21] V. Shoup. *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press, Cambridge, UK, 2005, ISBN 9781139165464. [doi](#)↑_{140, 142}
- [22] V. T. Goh, M. U. Siddiqi. “Multiple error detection and correction based on redundant residue number systems”, *IEEE Transactions on Communications*, **56**:3 (2008), pp. 325–330. [doi](#)↑₁₄₁
- [23] L. Sousa. “Nonconventional computer arithmetic circuits, systems and applications”, *IEEE Circuits and Systems Magazine*, **21**:1 (2021), pp. 6–40. [doi](#)↑₁₄₁
- [24] C. Chang, A. S. Molahosseini, A. A. E. Zarandi, T. F. Tay. “Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications”, *IEEE Circuits and Systems Magazine*, **15**:4 (2015), pp. 26–44. [doi](#)↑₁₄₂
- [25] N. S. Szabo, R. I. Tanaka. *Residue Arithmetic and its Application to Computer Technology*, McGraw-Hill, New York, USA, 1967, ISBN 978-0070626591, 236 pp. ↑_{142, 143, 161}
- [26] D. E. Knuth. *The Art of Computer Programming. V. 2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, USA, 1997, ISBN 0201896842, 784 pp. ↑_{142, 144}

- [27] H. M. Yassine, W. R. Moore. “Improved mixed-radix conversion for residue number system architectures”, *IEEE Proceedings G (Circuits, Devices and Systems)*, **138**:1 (1991), pp. 120–124. [doi](#)[↑]₁₄₄
- [28] M. Akkal, P. Siy. “A new mixed radix conversion algorithm MRC-II”, *Journal of Systems Architecture*, **53**:9 (2007), pp. 577–586. [doi](#)[↑]₁₄₄
- [29] Y. Wang. “New Chinese remainder theorems”, *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers*. V. 1 (1–4 Nov. 1998, Pacific Grove, CA, USA), 1998, ISBN 0-7803-5148-7, pp. 165–171. [doi](#)[↑]₁₄₅
- [30] Y. Wang. “Residue-to-binary converters based on new Chinese remainder theorems”, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, **47**:3 (2000), pp. 197–205. [doi](#)[↑]_{145,151}
- [31] W. Zhang, P. Siy. “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*, **178**:1 (2008), pp. 264–279. [doi](#)[↑]₁₄₅
- [32] A. S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, S. Timarchi. “Efficient reverse converter designs for the new 4-Moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ and $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n} + 1\}$ based on new CRTs”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, **57**:4 (2010), pp. 823–835. [doi](#)[↑]₁₄₅
- [33] S. Bi, W. Gross. “The mixed-Radix Chinese remainder theorem and its applications to residue comparison”, *IEEE Transactions on Computers*, **57**:12 (2008), pp. 1624–1632. [doi](#)[↑]_{146,153,160}
- [34] G. Dimauro, S. Impedovo, G. Pirlo. “A new technique for fast number comparison in the residue number system”, *IEEE Transactions on Computers*, **42**:5 (1993), pp. 608–612. [doi](#)[↑]_{146,153,160}
- [35] G. Dimauro, S. Impedovo, G. Pirlo, A. Salzo. “RNS architectures for the implementation of the ‘diagonal function’”, *Information Processing Letters*, **73**:5–6 (2000), pp. 189–198. [doi](#)[↑]₁₄₆
- [36] P. V. Ananda Mohan. “RNS to binary conversion using diagonal function and Pirlo and Impedovo monotonic function”, *Circuits, Systems, and Signal Processing*, **35**:3 (2016), pp. 1063–1076. [doi](#)[↑]_{147,148}
- [37] G. Pirlo, D. Impedovo. “A new class of monotone functions of the residue number system”, *International Journal of Mathematical Models and Methods in Applied Sciences*, **7**:9 (2013), pp. 802–809. [URL](#)[↑]_{147,148}
- [38] M. Soderstrand, C. Vernia, J. Chang. “An improved residue number system digital-to-analog converter”, *IEEE Transactions on Circuits and Systems*, **30**:12 (1983), pp. 903–907. [doi](#)[↑]₁₄₈
- [39] M. Lu, J. Chiang. “A novel division algorithm for the residue number system”, *IEEE Transactions on Computers*, **41**:8 (1992), pp. 1026–1032. [doi](#)[↑]_{149,154,155,160,161,165}
- [40] K. Isupov, V. Knyazkov. “Interval estimation of relative values in residue number system”, *Journal of Circuits, Systems and Computers*, **27**:1 (2018), pp. 1850004. [doi](#)[↑]_{149,150,157,160}
- [41] N. I. Chervyakov, M. G. Babenko, P. A. Lyakhov, I. N. Lavrinenko. “An approximate method for comparing modular numbers and its application to the division of

- numbers in residue number systems”, *Cybernetics and Systems Analysis*, **50**:6 (2014), pp. 977–984. [doi](#)[↑][149,155,160,162](#)
- [42] Jin Yul Kim, Kyu Ho Park, Hwang Soo Lee. “Efficient residue-to-binary conversion technique with rounding error compensation”, *IEEE Transactions on Circuits and Systems*, **38**:3 (1991), pp. 315–317. [doi](#)[↑][149](#)
- [43] H. Brönnimann, I. Z. Emiris, V. Y. Pan, S. Pion. “Sign determination in residue number systems”, *Theoretical Computer Science*, **210**:1 (1999), pp. 173–197. [doi](#)[↑][149,157,160](#)
- [44] A. A. Hiasat, H. S. Abdel-Aty-Zohdy. “Design and implementation of an RNS division algorithm”, *Proc. 13th IEEE Symposium on Computer Arithmetic* (6–9 July 1997, Asilomar, CA, USA), 1997, ISBN 0-8186-7846-1, pp. 240–249. [doi](#)[↑][149,161,162,163,165](#)
- [45] A. S. Korzhavina, V. S. Knyaz’kov. “High-precision computations using residue-interval arithmetic on FPGAs”, *Program Systems: Theory and Applications*, **10**:3(42) (2019), pp. 81–127 (in Russian). [doi](#)[↑][149,150](#)
- [46] C. Y. Hung, B. Parhami. “An approximate sign detection method for residue numbers and its application to RNS division”, *Computers & Mathematics with Applications*, **27**:4 (1994), pp. 23–35. [doi](#)[↑][149,156,157,160,162,165](#)
- [47] J. Yang, C. Chang, C. Chen. “A high-speed division algorithm in residue number system using parity-checking technique”, *International Journal of Computer Mathematics*, **81**:6 (2004), pp. 775–780. [doi](#)[↑][149,162,165](#)
- [48] R. Conway, J. Nelson. “New CRT-based RNS converter using restricted moduli set”, *IEEE Transactions on Computers*, **52**:5 (2003), pp. 572–578. [doi](#)[↑][149,151](#)
- [49] K. Isupov. “Using floating-point intervals for non-modular computations in residue number system”, *IEEE Access*, **8** (2020), pp. 58603–58619. [doi](#)[↑][149,150,159,163,165](#)
- [50] S. Arthireena, G. Shanmugavadivel. “Efficient sign detection using parallel prefix adder”, *Proc. 2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering*, ICEICE, IEEE, 27–28 April 2017, Karur, India, pp. 1–5. [doi](#)[↑][149](#)
- [51] S. Kawamura, M. Koike, F. Sano, A. Shimbo. “Cox-Rower architecture for fast parallel Montgomery multiplication”, *Advances in Cryptology*, EUROCRYPT 2000, Lecture Notes in Computer Science, vol. **1807**, ed. B. Preneel, Springer, Berlin–Heidelberg, 2000, ISBN 978-3-540-45539-4, pp. 523–538. [doi](#)[↑][149](#)
- [52] T. V. Vu. “Efficient implementations of the Chinese remainder theorem for sign detection and residue decoding”, *IEEE Transactions on Computers*, **C-34**:7 (1985), pp. 646–651. [doi](#)[↑][149,154,155,160](#)
- [53] S. J. Meehan, S. D. O’Neil, J. J. Vaccaro. “An universal input and output RNS converter”, *IEEE Transactions on Circuits and Systems*, **37**:6 (1990), pp. 799–803. [doi](#)[↑][149](#)
- [54] N. Chervyakov, A. S. Molahosseini, P. A. Lyakhov, M. G. Babenko, M. A. Deryabin. “Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem”, *International Journal of Computer Mathematics*, **94**:9 (2017), pp. 1833–1849. [doi](#)[↑][149,151](#)


- [55] N. Whitehead, A. Fit-Florea. *Precision & performance: floating point and IEEE 754 compliance for NVIDIA GPUs*, 2011 (Accessed 20 March 2021), 7 pp. [URL](#)  [↑₁₅₀](#)
- [56] N. Burgess. “Scaled and unscaled residue number system to binary conversion techniques using the core function”, *Proc. 13th IEEE Symposium on Computer Arithmetic*, 6–9 July 1997, Asilomar, CA, USA, ISBN 0-8186-7846-1, pp. 250–257. [DOI](#)  [↑₁₅₀](#)
- [57] Shaoqiang Bi, Wei Wang, A. Al-Khalili. “New modulo decomposed residue-to-binary algorithm for general moduli sets”, *Proc. 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. V. V, 17–21 May 2004, Montreal, QC, Canada, ISBN 0-7803-8484-9, pp. 141–144. [DOI](#)  [↑₁₅₀](#)
- [58] A. Omondi. “Fast residue-to-binary conversion using base extension and the Chinese remainder theorem”, *Journal of Circuits, Systems and Computers*, **16**:3 (2007), pp. 379–388. [DOI](#)  [↑₁₅₀](#)
- [59] N. Burgess. “Efficient RNS to binary conversion using high-radix SRT division”, *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers*. V. 2, 1–4 Nov. 1998, Pacific Grove, CA, USA, 1998, ISBN 0-7803-5148-7, pp. 1240–1243. [DOI](#)  [↑₁₅₀](#)
- [60] J. Chen, R. Yao. “Efficient CRT-based residue-to-binary converter for the arbitrary moduli set”, *Science China Information Sciences*, **54**:1 (2011), pp. 70–78. [DOI](#)  [↑_{150,151}](#)
- [61] P. Patronik. “On reverse converters for arbitrary multi-moduli RNS”, *Integration*, **75** (2020), pp. 158–167. [DOI](#)  [↑₁₅₁](#)
- [62] A. Hiasat. “A residue-to-binary converter with an adjustable structure for an extended RNS three-moduli set”, *Journal of Circuits, Systems and Computers*, **28**:08 (2019), 1950126. [DOI](#)  [↑₁₅₁](#)
- [63] L. Sousa, S. Antao. “MRC-based RNS reverse converters for the four-moduli sets $\{2^n + 1, 2^n - 1, 2^n, 2^{2n+1} - 1\}$ and $\{2^n + 1, 2^n - 1, 2^{2n}, 2^{2n+1} - 1\}$ ”, *IEEE Transactions on Circuits and Systems II: Express Briefs*, **59**:4 (2012), pp. 244–248. [DOI](#)  [↑₁₅₁](#)
- [64] P. V. Ananda Mohan. “Reverse conversion using core function, CRT and mixed radix conversion”, *Circuits, Systems, and Signal Processing*, **36**:7 (2017), pp. 2847–2874. [DOI](#)  [↑₁₅₁](#)
- [65] E. K. Bankas, K. A. Gbolagade. “An efficient VLSI design of residue to binary converter circuit for a new moduli set $\{2^{2n}, 2^{2n-1} - 1, 2^{2n-1} + 1\}$ ”, *Proc. 2019 IEEE 4th International Conference on Integrated Circuits and Microsystems, ICICM (25–27 Oct. 2019, Beijing, China)*, pp. 24–28. [DOI](#)  [↑₁₅₁](#)
- [66] A. S. Molahosseini, A. A. E. Zarandi, P. Martins, L. Sousa. “A multifunctional unit for designing efficient RNS-Based datapaths”, *IEEE Access*, **5** (2017), pp. 25972–25986. [DOI](#)  [↑₁₅₁](#)
- [67] Y. Wang, X. Song, M. Aboulhamid. “A new algorithm for RNS magnitude comparison based on New Chinese remainder theorem II”, *Proc. Ninth Great Lakes Symposium on VLSI (4–6 March 1999, Ypsilanti, MI, USA)*, ISBN 0-7695-0104-4, pp. 362–365. [DOI](#)  [↑_{153,160}](#)

- [68] S. J. Pietrak. “A note on RNS architectures for the implementation of the diagonal function”, *Information Processing Letters*, **115**:4 (2015), pp. 453–457. [doi](#)[↑]₁₅₄
- [69] G. Pirlo. “Non-modular operations of the residue number system: functions for computing”, *Embedded Systems Design with Special Arithmetic and Number Systems*, eds. A. S. Molahosseini, L. S. de Sousa, C. Chang, Springer, Cham, 2017, ISBN 978-3-319-49742-6, pp. 49–64. [doi](#)[↑]₁₅₄
- [70] M. Babenko, M. Deryabin, S. J. Pietrak, P. Patronik, N. Chervyakov, A. Tchernykh, A. Avetisyan. “RNS number comparator based on a modified diagonal function”, *Electronics*, **9**:11 (2020), 1784, 16 pp. [doi](#)[↑]_{155,160}
- [71] K. Isupov. “High-performance computation in residue number system using floating-point arithmetic”, *Computation*, **9**:2 (2021), 9, 15 pp. [doi](#)[↑]₁₅₉
- [72] D. S. Phatak, S. D. Houston. “New distributed algorithms for fast sign detection in residue number systems (RNS)”, *Journal of Parallel and Distributed Computing*, **97** (2016), pp. 78–95. [doi](#)[↑]_{159,160}
- [73] A. A. Hiasat, H. Abdel-Aty-Zohdy. “Semi-Custom VLSI design and implementation of a new efficient RNS division algorithm”, *The Computer Journal*, **42**:3 (1999), pp. 232–240. [doi](#)[↑]₁₆₂
- [74] C. Chang, J. Yang. “A division algorithm using bisection method in residue number system”, *International Journal of Computer, Consumer and Control (IJ3C)*, **2**:1 (2013), pp. 59–66. [URL](#)[↑]_{162,165}
- [75] N. Chervyakov, P. Lyakhov, M. Babenko, A. Nazarov, M. Deryabin, I. Lavrinenko, A. Lavrinenko. “A high-speed division algorithm for modular numbers based on the Chinese remainder theorem with fractions and its hardware implementation”, *Electronics*, **8**:3 (2019), 261, 17 pp. [doi](#)[↑]_{162,165}
- [76] M. A. Hitz, E. Kaltofen. “Integer division in residue number systems”, *IEEE Transactions on Computers*, **44**:8 (1995), pp. 983–989. [doi](#)[↑]_{163,165}
- [77] M. Mccann, N. Pippenger. “SRT division algorithms as dynamical systems”, *Proc. 2003 16th IEEE Symposium on Computer Arithmetic* (15–18 June 2003, Santiago de Compostela, Spain), ISBN 0-7695-1894-X, pp. 46–53. [doi](#)[↑]₁₆₂
- [78] W. A. Chren Jr.. “A new residue number system division algorithm”, *Computers & Mathematics with Applications*, **19**:7 (1990), pp. 13–29. [doi](#)[↑]_{164,165}
- [79] D. K. Banerji, T. Cheung, V. Ganesan. “A high-speed division method in residue arithmetic”, 5th IEEE Symposium on Computer Arithmetic (18-19 May 1981, Ann Arbor, Michigan, USA), 1981, pp. 158–164. [URL](#)[↑]_{164,165}
- [80] D. Gamberger. “New approach to integer division in residue number systems”, *Proc. 10th IEEE Symposium on Computer Arithmetic* (26–28 June 1991, Grenoble, France), ISBN 0-8186-9151-4, pp. 84–91. [doi](#)[↑]_{164,165}
- [81] C. Chang, Y. Lai. “A division algorithm for residue numbers”, *Appl. Math. Comput.*, **172**:1 (2006), pp. 368–378. [doi](#)[↑]_{164,165}
- [82] N. Mukhtar, L. Papachristodoulou, A. P. Fournaris, L. Batina, Y. Kong. *Machine-learning assisted side-channel attacks on RNS-based elliptic curve implementations using hybrid feature engineering*, Cryptology ePrint Archive, Report 2020/1065, 2020, 31 pp. [URL](#)[↑]₁₆₆

- [83] S. Kawamura, Y. Komano, H. Shimizu, T. Yonemura. “RNS Montgomery reduction algorithms using quadratic residuosity”, *Journal of Cryptographic Engineering*, **9:4** (2019), pp. 313–331.  [↑₁₆₆](#)
- [84] D. Schoinianakis. “Residue arithmetic systems in cryptography: a survey on modern security applications”, *Journal of Cryptographic Engineering*, **10:3** (2020), pp. 249–267.  [↑₁₆₆](#)
- [85] P. Martins, L. Sousa. “The role of non-positional arithmetic on efficient emerging cryptographic algorithms”, *IEEE Access*, **8** (2020), pp. 59533–59549.  [↑₁₆₆](#)
- [86] D. Ongaro, J. Ousterhout. “In search of an understandable consensus algorithm”, *Proceedings of USENIX ATC’14: 2014 USENIX Annual Technical Conference* (June 19–20, 2014, Philadelphia, PA, USA), ISBN 978-1-931971-10-2, pp. 305–320.  [↑₁₆₈](#)
- [87] D. Bailey, J. Borwein. “High-precision arithmetic in mathematical physics”, *Mathematics*, **3:2** (2015), pp. 337–367.  [↑₁₆₉](#)
- [88] R. Brent, P. Zimmermann. *Modern Computer Arithmetic*, Cambridge University Press, Cambridge, 2010, ISBN 9780511921698.  [↑₁₆₉](#)

Sample citation of this publication:

Konstantin S. Isupov. “An overview of high-performance computing using the residue number system”. *Program Systems: Theory and Applications*, 2021, **12:2**(49), pp. 137–192. (*In Russian*).  [10.25209/2079-3316-2021-12-2-137-192](#)

 http://psta.psir.ru/read/psta2021_2_137-192.pdf