УДК 681.3

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[1]

Аннотация. Представлены новый метод и алгоритмы деления модулярных чисел, основанные на процедуре использования относительных величин делимого и делителя к полному диапазону системы остаточных классов. В основе алгоритма модулярного деления используются элементарные операции регистрового сдвига и сложения, что делает его простым и быстродействующим. В настоящее время такой алгоритм считается наиболее быстрым.

Ключевые слова: алгоритм, система остаточных классов, модулярная арифметика, деление.

ВВЕДЕНИЕ

Систему остаточных классов (СОК, RNS — Residue Number System) многие исследователи используют в качестве основы для вычислительной техники. В последнее десятилетие интерес к ней резко возрос, что подтверждается большим числом публикаций, посвященных практическому применению СОК в цифровой обработке сигналов, системах обработки изображений, помехоустойчивом кодировании, криптографических системах, квантовых автоматах, нейрокомпьютерах, системах с массовым параллелизмом операций, облачных вычислениях и др. [1–13]. В работах [14–16] исследованы модели отдельных вычислительных узлов специализированных процессоров, включающих такие элементы СОК, как: входные и выходные преобразователи, модульные сумматоры, схемы определения знака и переполнения, деления и масштабирования, расширения базы СОК, схемы округления, локализации и исправления ошибок и другие, которые могут быть реализованы на классической, нейросетевой или квантовой вычислительной базе.

Арифметико-логическое устройство (АЛУ) в СОК для решения задач с очень большими числами, используемых при шифровании и дешифровании, представлено в [17].

СОК дает преимущества быстрого сложения и умножения по сравнению с остальными системами счисления, что обусловливает большой интерес к этой системе в тех областях, где требуются большие объемы вычислений. Однако некоторые операции, такие как сравнение и деление чисел, весьма сложны в СОК. Разработка более эффективных алгоритмов деления позволит найти новые перспективные области применения данной системы.

Известные алгоритмы деления в СОК можно разделить на две категории: с использованием умножения и с использованием вычитания. Большинство алгоритмов на основе умножения предварительно вычисляют обратную величину делителя, после чего эта величина умножается на делимое. Алгоритмы на основе вычитания используют вычитание кратных делителя из делимого, до тех пор, пока полученный результат не будет меньше делителя. Некоторые известные алгоритмы деления в СОК на основе умножения и функции ядра СОК изложены в [18–20]. Эти алгоритмы используют преобразование в обобщенную позиционную систему счисления (ОПСС, MRC — Mixed Radix Conversion) для нахождения обратной величины делителя и сравнения чисел. Алгоритмы, основанные на ОПСС, медленные и требуют выполнения большого количества арифметических

[1]Работа выполнена при финансовой поддержке Российского Фонда Фундаментальных Исследований, гранты 13-07-00478-а и 14-07-31004-мол-а.

операций. Известен алгоритм деления в формате СОК [21], который не использует двоичной системы счисления или ОПСС во время деления. Он требует дополнительного набора модулей СОК, что ведет к большой избыточности. Большинству существующих алгоритмов присущи различные недостатки, что делает их менее пригодными для решения задачи деления в СОК.

В данной статье предлагается эффективный алгоритм деления в общем случае в СОК с использованием только регистровых сдвигов и суммирований. Улучшенный алгоритм характеризуется следующими свойствами: очень быстр по сравнению с известными алгоритмами, не имеет ограничений на делимое и делитель (кроме равенства нулю делителя), не использует предварительной оценки частного, обратной величины для делителя и операции расширения базы.

Известные алгоритмы деления чисел, представленных в СОК, базируются на абсолютных значениях делимого и делителя. В работе используются не абсолютные значения, а их относительные величины, что позволяет уменьшить вычислительную сложность алгоритмов деления.

МОДИФИКАЦИЯ АЛГОРИТМА ДЕЛЕНИЯ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ НА ОСНОВЕ ПРИБЛИЖЕННОГО МЕТОДА

Рассмотрим модификацию алгоритма основного деления чисел, представленных в СОК в случае, когда и делимое и делитель — произвольные целые числа и делитель не приводится к случаю попарно простого с модулями СОК [22]. Модификация основана на использовании делимого и делителя, представленных в относительных величинах.

В последнее время проявляется значительный интерес к СОК, характеризующейся высоким уровнем естественного параллелизма при выполнении арифметических операций, высокой точностью, надежностью и стойкостью.

Специализированные процессоры на основе арифметики СОК имеют большое значение в высокоскоростных системах обработки данных в режиме реального времени. Операции сложения, вычитания и умножения, называемые модульными операциями, можно реализовать очень быстро, без распространения межразрядных переносов. Немодульные операции деления, сравнения чисел, определения знака и переполнения диапазона остаются сравнительно медленными. Любое улучшение скорости этих медленных алгоритмов значительно улучшает производительность многомодульных АЛУ. Обычно при рассмотрении деления в СОК выделяют три категории: деление с нулевым остатком, масштабирование и деление в общем случае. Проблема деления в общем виде в СОК привлекает внимание многих исследователей для разработки высокопроизводительных многомодульных АЛУ. Известные алгоритмы деления в СОК, основанные на использовании преобразования в ОПСС, масштабировании, округлении, расширении и других операциях, являются медленными и требуют выполнения большого количества арифметических действий. Большинство известных алгоритмов основаны на сравнении делимого с делителем или его удвоенным значением, что представляет определенную сложность [22]. В связи с этим возникает необходимость упростить структуру вычислений при сравнении модулярных чисел. Одним из направлений упрощения операции сравнения модулярных чисел является подход с использованием приближенного метода вычисления позиционной характеристики модулярного числа, позволяющий абсолютно правильно реализовать основные классы процедур принятия решений: проверку равенства (неравенства) двух значений, сравнение двух значений (больше, меньше) и другие, которые обеспечивают решение задач, возникающих при аппаратной или программной реализации вычислений в системе остаточных классов.

Идея приближенного метода вычисления позиционной характеристики модулярного числа и его применения для деления модулярных чисел основана на

использовании относительных величин анализируемых чисел к полному диапазону, определяемому китайской теоремой об остатках, связывающей позиционное число a с его представлением в остатках $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, где α_i — наименьшие неотрицательные вычеты числа, относительно модулей системы остаточных классов $p_1, p_2, \ldots, p_n$ выражением

$$a = \left| \sum_{i=1}^{n} \frac{P}{p_i} |P_i^{-1}|_{p_i} \alpha_i \right|_P . \qquad (1)$$

Здесь $P = \prod_{i=1}^{n} p_i$, p_i — модули СОК, $|P_i^{-1}|$ — мультипликативная инверсия P_i относительно p_i, $P_i = \frac{P}{p_i} = p_1 p_2 \ldots p_{i-1} p_{i+1} \ldots p_n$.

В СОК используются различные наборы модулей для конкретных приложений. Выбор соответствующего набора является одной из наиболее важных проблем в СОК. Набор модулей имеет большое значение для производительности системы, а также оказывает влияние на другие ее части. Каждый набор имеет свои преимущества и недостатки, что обусловливает исследования по их сравнению в заданном динамическом диапазоне $P = p_1 p_2 \ldots p_n$, где p_i — модули СОК. Это позволяет выбрать наиболее эффективные наборы модулей для каждого динамического диапазона. Некоторые из них приводят к более эффективному преобразованию из позиционной системы счисления в СОК, другие эффективнее при арифметической обработке данных.

Основными характеристиками каждого набора модулей являются их количество и тип. Прежде всего необходимо обеспечить покрытие динамического диапазона вычислений. Это можно выполниить либо с большим числом модулей небольшого размера, либо с меньшим числом модулей большого размера. В первом случае увеличивается параллелизм арифметических операций, во втором — снижается сложность обратного преобразования из СОК. Необходимо найти компромисс между этими двумя свойствами.

Кроме того, конкретное приложение для применения СОК также оказывает сильное влияние на выбор модулей. Другими словами, невозможно подобрать единый набор модулей, наилучший для всех приложений. Вопросы важности учета конкретного приложения при выборе модулей СОК рассмотрены в [23–28]. Отметим, что выбор модулей СОК представляет собой самостоятельное исследование, поэтому в данной работе он проведен только в рамках покрытия динамического диапазона вычислений, определенного значениями делимого и делителя.

Рассмотрим метод вычисления относительных значений в СОК. Если разделить левую и правую части выражения (1) на константу P, соответствующую диапазону чисел, то получим приближенное относительное значение

$$\left| \frac{a}{P} \right|_1 = \left| \sum_{i=1}^{n} \frac{|P_i^{-1}|_{p_i}}{p_i} \alpha_i \right|_1 \approx \left| \sum_{i=1}^{n} k_i \alpha_i \right|_1 , \qquad (2)$$

где $k_i = \frac{|P_i^{-1}|_{p_i}}{p_i}$ — константы выбранной системы, а α_i — разряды числа, представленного в СОК по модулям p_i, где $i = 1, 2, \ldots, n$, при этом значение суммы (2) находится в интервале [0,1). Конечный результат суммы определяется после суммирования и отбрасывания целой части числа с сохранением дробной части суммы. Дробная величина $F(a) = \left| \frac{a}{P} \right|_1 \in [0,1)$ содержит как информацию о величине числа, так и о его знаке. Если $\left| \frac{a}{P} \right|_1 \in \left[0, \frac{1}{2} \right)$, то число a поло-

жительное и $F(a)$ равна величине числа a, разделенной на P. В противном случае a — отрицательное число и $1-F(a)$ показывает относительную величину числа a. Округление величины $F(a)$ до 2^{-t} бита обозначим как $[F(a)]_{2^{-t}}$. Точное значение величины $F(a)$ определяется неравенствами $[F(a)]_{2^{-t}} < F(a) < [F(a)]_{2^{-t}} + 2^{-t}$. Целая часть числа, полученная в результате суммирования констант k_i, представляет собой ранг числа, т.е. такую непозиционную характеристику, которая показывает, сколько раз диапазон системы P был превышен при переходе от представления чисел в СОК к его позиционному представлению. При необходимости определение ранга может производиться непосредственно в процессе выполнения операции суммирования констант k_i. Дробную часть можно записать также как $A \bmod 1$, так как $A = \lfloor A \rfloor + A \bmod 1$. Количество разрядов дробной части числа определяется максимально возможной разностью между соседними числами. При точном сравнении, широко используемом при делении чисел, необходимо вычислить значение, которое является эквивалентом преобразования из СОК в позиционную систему счисления.

Округление величины $F(a)$ неизбежно влечет возникновение погрешности. Обозначим $\rho = -n + \sum_{i=1}^{n} p_i$. В работах [29, 30] показано, что необходимо использовать $N = \lceil \log_2 (P\rho) \rceil$ бит после запятой при округлении величины $F(a)$, чтобы возникающая погрешность не оказывала влияния на точность вычислений. Другими словами, устанавливается взаимно однозначное соответствие между множеством чисел, представленных в СОК, и множеством величин $[F(a)]_{2^{-N}}$.

Перейдем к рассмотрению улучшенной версии алгоритма деления в СОК, приведенного в [22]. Алгоритм деления целых чисел $\frac{a}{b}$ можно описать итеративной схемой, которая выполняется в два этапа. На первом этапе осуществляется поиск старшей степени 2^i при аппроксимации частного двоичным рядом, на втором — уточнение аппроксимирующего ряда. Чтобы получить диапазон, больший, чем P, можно выбрать значение $P' = Pp_{n+1}$, т.е. потребуется расширить базу СОК, добавив дополнительный модуль. Чтобы избежать расширения базы — вычислительно сложной операции, необходимо сравнивать не делимое с промежуточными делителями, а текущие результаты итерации (i) с предыдущими значениями итераций $(i-1)$. Процесс удваивания делителя повторяем до тех пор, пока значение промежуточного делителя при i-й итерации будет меньше, чем при $i-1$-й итерации. Это позволит выполнить условие $0 < b < P-1$.

Алгоритм деления можно описать следующими правилами.

Конструируется некоторое правило φ, которое каждой паре целых положительных чисел a и b ставит в соответствие некоторое положительное число q_i, где i — номер итерации, такое, что $a - bq_i = r_i > 0$, т.е. $a > bq_i$. Тогда деление a на b осуществляется по следующему правилу: согласно операции φ паре чисел a и b ставится в соответствие число $q_1 = q_0$ такое, что $a - bq_1 = r_1 \geq 0$, т.е. $a \geq bq_1$. В качестве q_i примем значения 2^i, которые поместим в память в виде констант $c_i = (2^i \bmod p_1, 2^i \bmod p_2, \ldots, 2^i \bmod p_n)$. При этом $i+1$-я операция не зависит от i-й операции, что позволяет итерации выполнять параллельно. Кроме того, в каждой итерации выполняются только две операции: умножение делителя на константу 2^i и сравнение полученного значения с делимым.

Если $r_1 \leq b_1$, то деление закончено, если $r_1 \geq b_1$, то согласно правилу φ паре чисел (r_1, b) ставится в соответствие q_2 такое, что $a - bq_2 = r_2 \geq 0$, т.е. $a \geq bq_2$. Если $r_2 < b$, то деление завершается, если $r_2 \geq b$, то согласно правилу φ паре чи-

сел (r_2, b) ставится в соответствие q_3 такое, что $a - bq_3 = r_3 \geq 0$, и т.д. Так как последовательное применение операции φ приводит к убывающей последовательности целых чисел $a > r_1 > r_2 > \ldots \geq 0$, алгоритм реализуется за конечное число шагов. Пусть на m-м шаге зафиксирован случай $0 < bq_m$, что означает окончание операции деления. Тогда в итоге получим $a \cong (q_1 + q_2 + \ldots + q_m)b + r_m$, где ряд $q_1 + q_2 + \ldots + q_m$ — аппроксимация частного, которое может содержать лишние q_i. Далее необходимо провести уточнение полученного аппроксимирующего ряда.

Уточнение начнем со старшего q_m. Если $a > bq_m$, то q_m является членом аппроксимирующего ряда образованного частного. Далее выберем $(q_m + q_{m-1})$. Если $a > b(q_m + q_{m-1})$, то q_{m-1} вносится в ряд, иначе, если $a < b(q_m + q_{m-1})$, то q_m исключается из ряда, и т.д. После проверки всех q_i частное определяется оставшимися членами ряда. Тогда искомое частное определяется выражением

$$a = (q_m + q_{m-1} + \ldots + q_i + \ldots + q_1)b + r_m,$$

где

$$q_i = \begin{cases} 1 & \text{при } (q_m + q_{m-1} + \ldots + q_i)b < a, \\ 0 & \text{в противном случае.} \end{cases}$$

Данный алгоритм легко модифицируется в модулярную форму, причем абсолютные значения величин заменены их относительными значениями. Структура предлагаемого алгоритма основана на применении приближенного метода при сравнении чисел, которое выполняется с использованием операции вычитания.

Известные алгоритмы деления определяют частное на основе итерации $A' = A - QD$, где A и A' — соответственно текущее и следующее делимое, D — делитель, Q_1 — частное, которое генерируется на каждой итерации из полного диапазона СОК, а не выбирается из небольшого множества констант. В предлагаемом алгоритме частное определяется на основе итерации $r_i = A - b2^i$, где A — некоторое делимое, b — делитель, 2^i — член аппроксимирующего ряда частного.

Так как делимое во всех итерациях не меняется, а делитель умножается на константу, предложенный алгоритм позволит уменьшить вычислительную сложность по сравнению с известными аналогами. При итерационном процессе деления в позиционной системе счисления для поиска старшей степени ряда аппроксимации частного и уточнения аппроксимирующего ряда сравниваются делимое с удвоенными делителями или суммой членов ряда. Применение этого принципа для СОК может привести к некорректной работе алгоритма, поскольку при переполнении динамического диапазона промежуточного делителя восстановленное число может выйти за пределы данного диапазона, что вызвано цикличностью СОК. При этом значение промежуточного делителя все еще будет меньше делимого. Это не соответствует действительности, так как на самом деле числа будут превышать диапазон P и алгоритм перейдет в режим «зацикливание». Например, если модули СОК равны $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, то диапазон $P = 2 \cdot 3 \cdot 5 \cdot 7 = 210$. Допустим, при восстановлении получили число $A = 220$. В СОК $A = 220 = (0, 1, 0, 3)$, т.е. $A = 210$ и $A' = 10$ имеют одинаковые представления в системе. Указанная неоднозначность может привести к нарушению работы алгоритма.

Для преодоления этой трудности необходимо в СОК сравнивать результаты текущих значений итераций с предыдущими, что позволяет правильно определить большее или меньшее число. Итак, факт переполнения динамического диапазона в СОК можно использовать для принятия решения «больше – меньше».

На первой итерации происходит сравнение делимого с делителем, на остальных итерациях — сравнение удвоенных значений делителей $q_i b < q_{i+1} b$. На каждой новой итерации происходит сравнение текущего значения с предыдущим. Последовательное применение этих итераций приводит к формированию цепоч-

ки неравенств $bq_1 < bq_2 < \ldots < bq_m > bq_{m+1}$, определяющих необходимое количество требуемых итераций, зависящих от величин делимого и делителя. Таким образом, алгоритм реализуется за конечное число итераций. Пусть на $m+1$-й итерации зафиксирован случай окончания возрастающей последовательности $bq_m > bq_{m+1}$, что соответствует переполнению диапазона СОК, т.е. $bq_{m+1} > P$ и $a < bq_{m+1}$. На этом процесс формирования интерполяции частного двоичным рядом или набором констант в СОК завершается. Итак, процесс аппроксимации частного может осуществляться путем сравнения соседних приближенных делителей.

НОВЫЙ АЛГОРИТМ ОСНОВНОГО ДЕЛЕНИЯ МОДУЛЯРНЫХ ЧИСЕЛ

Предлагаемый алгоритм основного деления модулярных чисел на основе приближенного метода сравнения чисел работает следующим образом. Подав на вход делимое a и делитель b, вычисляем частное Q и остаток R такие, что $a = Qb + R$ и $0 \le R < b$.

Структура алгоритма состоит из следующих блоков.

Блок 1. Вычисление относительных значений делимого и делителя.

Поступившие на вход абсолютные значения делимого a и делителя b переводятся в относительные значения к полному диапазону СОК в соответствии с выражением (2). Полученные относительные значения $F(a) \approx \left|\frac{a}{P}\right|_1$ и $F(b) \approx \left|\frac{b}{P}\right|_1$ представлены в виде двоичной дроби.

Блок 2. Сравнение относительных значений делимого и делителя.

Если $F(a) \le F(b)$, то процесс завершается, если $F(a) > F(b)$, то осуществляется поиск старшей степени двоичной дроби 2^{-i} при аппроксимации частного Q двоичной дробью.

Блок 3. Поиск старшей степени ряда.

Старшая степень делителя $F(b)$ определяется фактом переполнения разрядной сетки, возникающей после i-го умножения на 2 двоичной дроби. В алгоритме умножение выполняется сдвигом влево двоичной дроби до появления переноса старшего значащего разряда в знаковый разряд. Количество сдвигов определяет старшую степень двоичного ряда при аппроксимации частного Q.

Блок 4. Определение членов двоичного ряда, необходимых для образования частного Q.

Формирование членов ряда частного Q начинаем со старшей степени ряда, входящей в аппроксимационный ряд. Остальные члены ряда получаются путем его деления на 2, которое в алгоритме формируется с помощью операций сдвига вправо. Определение членов аппроксимационного ряда, необходимых для образования частного, осуществляется путем анализа полученных значений промежуточных частных, образованных при последовательной их выборке, и суммирования. Члены ряда, суммируемые с промежуточными частными и не превосходящие делимого, включаются в образование частного, а те члены ряда, суммирование которых приведет к превышению делимого, отбрасываются.

Блок 5. Вычисление частного $Q = \left\lfloor \frac{a}{b} \right\rfloor$.

Для получения частного от деления a на b суммируем все оставшиеся члены аппроксимационного ряда (блок 4), необходимые для образования частного.

На этом работа алгоритма заканчивается.

Далее приводится детальное описание улучшенного алгоритма деления модулярных чисел. Выполним аппроксимацию частного.

Шаг 1. Вычисляем приближенные значения делимого $F(a)$ и делителя $F(b)$ и сравниваем их. Если $F(a) \le F(b)$, то процесс деления заканчивается и частное $\left\lfloor \frac{a}{b} \right\rfloor$ со-

ответственно равно 0 или 1. Если $F(a) > F(b)$, то осуществляется поиск старшей степени 2^{-N} при аппроксимации частного a двоичным кодом, где N — минимальный значащий разряд двоичной дроби.

Проведем поиск старшей степени двоичной дроби.

Шаг 2. Сдвигаем функцию $[F(b)]_{2^{-N}}$ влево до изменения первого двоичного разряда после запятой. Количество сдвигов определяет старшую степень j, которая регистрируется счетчиком импульсов, соединенных с памятью V.

На этом аппроксимация частного заканчивается. Для уточнения аппроксимационного ряда частного выполним следующие шаги.

Шаг 3. Из памяти выбираем константу 2^j (старшая степень ряда) и умножаем ее на делитель. Величину $2^j F(b)$ сравниваем с делимым $F(a)$ с помощью приближенного метода сравнения чисел в СОК. Константы 2^j, $1 \le j \le \log_2 P$, предварительно записаны в память V, счетчик j и частное Q установлены в «0». Выходы счетчика являются адресными входами памяти V.

Шаг 4. Вычисляем $\Delta_1 = F(a) - F_1(b)$. Если в знаковом разряде Δ_1 значение «1», то соответствующая степень ряда отбрасывается, если значение «0», то в сумматор частного добавляем значение члена ряда с этой степенью, т.е. $2^j \bmod p_i$, $1 \le i \le n$, $0 \le j \le N$.

Шаг 5. Проверяем член ряда со степенью 2^{j-1} путем сдвига вправо и сравнения. Сравниваем Δ_1 и $2^{j-1} b$. Если $\Delta_1 < 2^{j-1} b$, то соответствующая степень ряда отбрасывается; если $\Delta_1 > 2^{j-1} b$, то в сумматор частного добавляем значение члена ряда с этой степенью, т.е. $2^{j-1} \bmod p_i$ и $\Delta_2 = \Delta_1 - 2^{j-1} b$.

Шаг 6. Аналогично проверяем все оставшиеся члены ряда до нулевой степени. Последнее $\Delta_i = R = \Delta_{i-1} - F_{i-1}$, т.е. $0 \le R < b$, будет остатком от деления a на b. Частным Q будет сумма всех 2^i, необходимых для образования частного, которые были накоплены в сумматоре. Работа алгоритма завершается.

Покажем работу модифицированного алгоритма на примере.

Пример. Найти частное $Q = \dfrac{a}{b}$ от деления числа $a = 97$ на число $b = 8$ в СОК с основаниями $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$. Тогда $P = 2 \cdot 3 \cdot 5 \cdot 7 = 210$, $\rho = 2 + 3 + 5 + 7 - 4 = 13$, $P_1 = \dfrac{P}{p_1} = 105$, $P_2 = \dfrac{P}{p_2} = 70$, $P_3 = \dfrac{P}{p_3} = 42$, $P_4 = \dfrac{P}{p_4} = 30$.

Константы k_i, используемые для вычислений относительных величин, равны

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

Для выполнения точных операций с относительными величинами чисел в СОК необходимо использовать $N = \lceil \log_2 (P\rho) \rceil = 12$ знаков после запятой. Константы k_i, округленные до 12 двоичных знаков после запятой, равны

$$k_1 = 0{,}100000000000,\ k_2 = 0{,}010101010101,$$

$$k_3 = 0{,}100110011001,\ k_4 = 0{,}100100100100.$$

Представим в СОК числа a и b:

$$a_{10} = 97 \to (1, 1, 2, 6)_{\text{СОК}},\ b_{10} = 8 \to (0, 2, 3, 1)_{\text{СОК}}.$$

Относительные величины делимого a и делителя b при полной точности вычислений N равны

$$[F(a)]_{2^{-12}} = 0{,}011101011111,\ [F(b)]_{2^{-12}} = 0{,}000010011001.$$

Сдвигая дробную часть делителя b влево, шаг за шагом, определяем, что изменение первого дробного разряда после запятой происходит на четвертом сдвиге. Таким образом, в аппроксимационный ряд могут входить только величины 2^0, 2^1, 2^2 и 2^3, которые в СОК имеют следующее представление:

$$q_0 = 2^0 = (1,1,1,1),\ q_1 = 2^1 = (0,2,2,2),\ q_2 = 2^2 = (0,1,4,4),\ q_3 = 2^3 = (0,2,3,1).$$

Данные величины и будут формировать аппроксимационный ряд частного, который в дальнейшем будет уточнен. Для уточнения аппроксимационного ряда вычитаем из дроби $[F(a)]_{2^{-12}}$ делимого дробь $[F(b)]_{2^{-12}}$ делителя, сдвинутую влево на три разряда (т.е. умноженную на 2^3):

$$\Delta_1 = [F(a)]_{2^{-12}} - 2^3[F(b)]_{2^{-12}} = 0{,}011101011111 - 0{,}010011001 = 0{,}001010010111.$$

Так как $\Delta_1 > 0$, то 2^3 оставляем в аппроксимационном ряду, а величину Δ_1 используем в дальнейших вычислениях.

Вычитаем из Δ_1 дробь $[F(b)]_{2^{-12}}$ делителя, сдвинутую влево на два разряда:

$$\Delta_2 = \Delta_1 - 2^2[F(b)]_{2^{-12}} = 0{,}001010010111 - 0{,}0010011001 = 0{,}000000110011.$$

Поскольку $\Delta_2 > 0$, то 2^2 оставляем в аппроксимационном ряду, а величину Δ_2 используем в дальнейших вычислениях.

Вычитаем из Δ_2 дробь $[F(b)]_{2^{-12}}$ делителя, сдвинутую влево на один разряд:

$$\Delta_3 = \Delta_2 - 2^1[F(b)]_{2^{-12}} = 0{,}000000110011 - 0{,}00010011001 = 1{,}111100000001.$$

Появление единицы в знаковом разряде означает, что $\Delta_3 < 0$, поэтому 2^1 исключаем из аппроксимационного ряда, а Δ_3 в дальнейшем не используем (продолжаем использовать Δ_2).

Вычитаем из Δ_2 дробь $[F(b)]_{2^{-12}}$ делителя (без сдвигов):

$$\Delta_4 = \Delta_2 - 2^{01}[F(b)]_{2^{-12}} = 0{,}000000110011 - 0{,}000010011001 = 1{,}111110011010.$$

Появление единицы в знаковом разряде означает, что $\Delta_4 < 0$, поэтому 2^0 исключаем из аппроксимационного ряда.

На этом процесс уточнения аппроксимационного ряда завершается. Для определения частного необходимо сложить оставшиеся члены аппроксимационного ряда. В данном примере остались следующие члены ряда: $q_3 = 2^3 = (0, 2, 3, 1)$ и $q_2 = 2^2 = (0, 1, 4, 4)$. Тогда частное определяется путем суммирования членов ряда:

$$\left\lfloor \frac{a}{b} \right\rfloor = (0,2,3,1) + (0,1,4,4) = (0,0,2,5) = 12.$$

Для оценки эффективности предложенного алгоритма проанализируем сложность выполнения операций на каждом этапе работы алгоритма.

1. Нахождение дробей $F(a)$ и $F(b)$ по формуле (2) требует выполнения n умножений, которые могут быть реализованы таблично, с использованием LUT таблиц по Np_i бит для каждого модуля СОК. Для суммирования необходимо $\lceil \log_2 n \rceil$ суммирований чисел длины N.

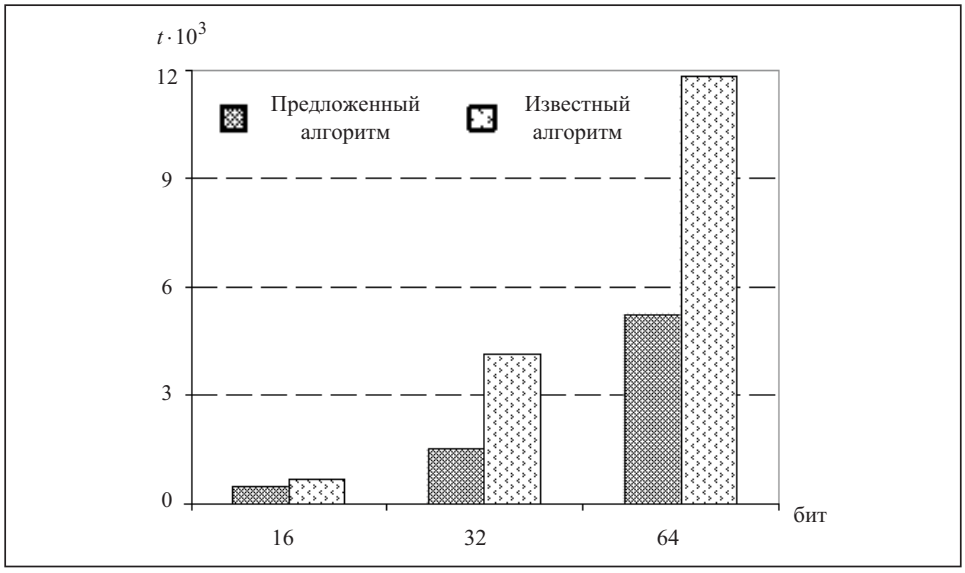

Рис. 1. График времени работы алгоритмов деления в СОК

2. Процедура составления аппроксимационного ряда частного потребует в худшем случае $\lceil \log_2 P \rceil$ операций сдвига.

3. Для уточнения ряда частного потребуется в худшем случае $\lceil \log_2 P \rceil$ операций сдвига и $\lceil \log_2 P \rceil$ операций вычитания чисел длины N.

4. Суммирование членов ряда при окончательном получении значения частного может быть организовано параллельно с предыдущими пунктами и не требует дополнительного времени на выполнение.

Таким образом, для реализации алгоритма требуется $N\lceil \log_2 n \rceil + (N+2)\lceil \log_2 P \rceil$ тактов вычислительной системы и $N(p_1 + p_2 + \ldots + p_n)$ бит памяти.

Алгоритм, изложенный в работе [22], требует выполнения в худшем случае $2\lceil \log_2 P \rceil$ операций сравнения в СОК. Наилучший метод сравнения чисел в СОК требует применения преобразования числа в ОПСС [27]. Для каждой операции преобразования необходимо выполнить одно табличное умножение и суммирование n чисел ширины $\max \{\log_2 p_i\}, 1 \le i \le n$, которое в силу особенностей метода ОПСС не может быть организовано параллельно [11]. Таким образом, вычислительная сложность известного алгоритма составляет $2n\lceil \log_2 P \rceil \max \{\log_2 p_i\}$.

Сравнение алгоритмов деления в СОК приведено на рис. 1. Была смоделирована работа предложенного алгоритма и алгоритма из [22] в СОК с диапазонами 16, 32 и 64 бит. В качестве критерия сравнения использовано количество тактов вычислительной системы, необходимых для выполнения алгоритмов. Результаты моделирования показывают, что предложенный алгоритм работает быстрее. Для СОК с диапазоном 16 бит время работы сокращается примерно в 1,4 раза, с диапазоном 32 бит — в 2,6 раз, с диапазоном 64 бит — в 2,3 раза. Данный результат свидетельствует о том, что разработанный алгоритм превосходит известные аналоги по скорости и является лучшим на сегодняшний день.

ЗАКЛЮЧЕНИЕ

Изложенный в статье алгоритм позволяет выполнять операцию деления в СОК быстрее, чем при использовании известных подходов. Это объясняется тем, что структура алгоритма содержит простые операции: при формировании

дробей-эквивалентов используются операции сложения и умножения, а при непосредственном нахождении частного — операции сдвига и вычитания. Переход к представлению чисел в СОК в виде отношения к диапазону позволил избежать необходимости применения в алгоритме таких операций, как нахождение остатка числа по модулю и перевод числа в ОПСС. Указанные операции весьма затратны в СОК, поэтому приведенный алгоритм, не использующий их, работает быстрее известных аналогов.

Улучшение скорости выполнения деления чисел позволит расширить применение СОК на практике. Предложенный подход может быть успешно использован в приложениях, требующих выполнения не только операций сложения, вычитания, умножения, но и деления. Поиск новых областей применения СОК на практике продолжает оставаться актуальной задачей, поэтому изложенный в работе подход может расширить прикладную значимость модулярной арифметики.

Следует отметить, что современное состояние теории СОК таково, что нахождение улучшенных вариантов алгоритмов выполнения немодульных операций является важной и актуальной задачей. Наряду с делением необходима разработка быстродействующих алгоритмов определения знака числа, сравнения чисел, обнаружения и локализации ошибки и др. В настоящее время ведется активный поиск оптимальных наборов модулей специального вида. Однако данный подход является достаточно узким и однобоким, в то время как разработка новых теоретических принципов СОК может дать хороший результат на практике. Создание принципиально новых алгоритмов выполнения проблемных операций в СОК будет способствовать развитию данной области математики.

СПИСОК ЛИТЕРАТУРЫ

1. Molahosseini A.S., Sorouri S., Zarandi A.A.E. Research challenges in next-generation residue number system architectures // Computer Science & Education (ICCSE), 7th Intern. Conf. — 2012. — P. 1658–1661.
2. Yatskiv V., Su Jun, Yatskiv N., Sachenko A., Osolinskiy O. Multilevel method of data coding in WSN // Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), IEEE 6th Intern. Conf. — 2011. — P. 863–866.
3. Su Jun, Zhengbing Hu. Method and dedicated processor for image coding based on residue number system // Modern Problems of Radio Engineering Telecommunications and Computer Science (TCSET), Intern. Conf. — 2012. — P. 406–407.
4. Су Цзунь. Многоуровневый метод кодирования данных в беспроводных сенсорных сетях // Электротехн. и компьютер. системы. — 2011. — № 4. — С. 213–218.
5. Ляхов П.А., Червяков Н.И. Реализация многоканальных фильтров в системе остаточных классов // Инфокоммуникац. технологии. — 2011. — **9**, № 2. — С. 4–7.
6. Chervyakov N.I., Veligosha A.V., Tyncherov K.T., Velikikh S.A. Use of modular coding for high-speed digital filter design // Cybernetics and Systems Analysis. — 1998. — N 34. — P. 254–260.
7. Zheng XD., Xu J., Li W. Parallel DNA arithmetic operation based on n-moduli set // Appl. Math. and Comput. — 2009. — **212**, N 1. — P. 177–184.
8. Gomathisankaran M., Tyagi A., Namuduri K. HORNS: A homomorphic encryption scheme for cloud computing using residue number system // Inform. Sci. and Systems (CISS), 45th Ann. Conf. — 2011. — P. 1–5.
9. Alia G., Martinelli E. NEUROM: a ROM based RNS digital neuron // Neural Networks. — 2005. — **18**. — P. 179–189.
10. Червяков Н.И., Сахнюк П.А., Шапошников А.В., Макоха А.Н. Нейрокомпьютеры в остаточных классах. — М.: Радиотехника, 2003. — 272 с.
11. Червяков Н.И., Сахнюк П.А., Шапошников А.В., Рядов С.А. Модулярные параллельные вычислительные структуры нейропроцессорных систем. — М.: Физматлит, 2003. — 288 с.

12. Червяков Н.И., Евдокимов А.А., Галушкин А.И., Лавриненко И.Н., Лавриненко А.В. Применение искусственных нейронных сетей и системы остаточных классов в криптографии. — М.: Физматлит, 2012. — 280 с.

13. Терещенко А.Н. Оптимизация метода Монтгомери за счет использования однословных умножений по однословному модулю // Компьютер. математика. — 2010. — № 1. — С. 73–82.

14. Червяков Н.И. Организация арифметических расширителей в микропроцессорных системах, базирующихся на множественном представлении информации // Упр. системы и машины. — 1987. — № 1. — С. 26–29.

15. Червяков Н.И. Методы и принципы построения модулярных нейрокомпьютеров // 50 лет модулярной арифметике: Сб. тр. Юбил. Междунар. науч.-техн. конф. — М.: ОАО «Ангстрем», МИЭТ. — 2006. — С. 239–249.

16. Червяков Н.И. Реализация высокоэффективной модулярной цифровой обработки сигналов на основе программируемых логических интегральных схем // Нейрокомпьютеры: разработка, применение. — 2006. — № 10. — С. 24–36.

17. Pat. WO 2013/176852 A1. Residue number arithmetic logic unit / E. Olsen. — Publ. 28.11.13.

18. Червяков Н.И., Лавриненко И.Н. Модулярные методы и алгоритмы деления на основе спуска Ферма и итераций Ньютона // Инфокоммуникац. технологии. — 2009. — **7**, № 4. — С. 9–12.

19. Червяков Н.И., Лавриненко И.Н., Лавриненко С.В., Мезенцева О.С. Методы и алгоритмы округления, масштабирования и деления модулярных чисел // 50 лет модулярной арифметике: Сб. тр. Юбил. Междунар. науч.-техн. конф. — М.: ОАО «Ангстрем», МИЭТ. — 2006. — С. 291–310.

20. Червяков Н.И. Методы, алгоритмы и техническая реализация основных проблемных операций, выполняемых в системе остаточных классов // Инфокоммуникац. технологии. — 2011. — **9**, № 4. — С. 4–12.

21. Chin-Chen Chang, Yeu-Pong Lai. A division algorithm for residue numbers // Appl. Math. and Comput. — 2006. — **172**, N 1. — P. 368–378.

22. Синьков М.В., Синькова Т.В., Федоренко А.В., Чапор А.А. Нетрадиционная система остаточных классов и ее основоположник И.Я. Акушский. — www.icfcst.kiev.ua/Symposium/Proceedings2/Sinkov.rtf.

23. Онищенко С.М. Применение гиперкомплексных чисел в теории инерциальной навигации. Автономные системы. — Киев: Наук. думка, 1983. — 208 с.

24. Пухов Г.Е., Евдокимов В.Ф., Синьков М.В. Разрядно-аналоговые вычислительные системы. — М.: Сов. радио, 1978. — 256 с.

25. Younes D., Steffan P. A comparative study on different moduli sets in residue number system // Intern. Conf. on Computer Systems and Industrial Informatics (ICCSII). — 2012. — P. 1–6.

26. Younes D., Steffan P. Efficient image processing application using residue number system // Proc. of the 20th Intern. Conf. on Mixed Design of Integrated Circuits and Systems (MIXDES). — 2013. — P. 468–472.

27. Кнут Д. Искусство программирования. — М.: Вильямс, 2003. — Т. 2. — 832 с.

28. Omondi A., Premkumar B. Residue number systems: Theory and implementation. — London: Imperial College Press, 2007. — 296 p.

29. Червяков Н.И., Ляхов П.А. Метод определения знака числа в системе остаточных классов на основе приближенных вычислений // Нейрокомпьютеры: разработка, применение. — 2012. — № 12. — С. 56–64.

30. Червяков Н.И., Бабенко М.Г., Ляхов П.А., Лавриненко И.Н. Приближенный метод определения знака числа в системе остаточных классов и его техническая реализация // Науч.-техн. ведомости С.-Петербург. гос. политех. ун-та. Информатика. Телекоммуникации. Управление. — 2013. — № 176. — С. 131–141.

Поступила 01.08.2013