ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 10**, №3(42), с. 81–127

ББК 32.971.32-04:22.192.22
ГРНТИ 50.07.05, 50.09.31
УДК 004.222.3:681.5.07+004.421.4

А. С. Коржавина, В. С. Князьков

Реализация высокоточных вычислений в базисе модулярно-интервальной арифметики

Аннотация. Проблема влияния ошибок округления возникает в большом количестве задач в различных областях знаний, включая вычислительную математику, математическую физику, биохимию, квантовую механику, математическое программирование. Для решения таких задач может потребоваться точность в 100–1000 десятичных цифр. В рамках данного исследования разработаны новые способы представления числовой информации — модулярно-позиционные интервально-логарифмические системы счисления, а также методы выполнения арифметических операций для повышения скорости высокоточных вычислений.

Ключевые слова и фразы: модулярная арифметика, гибридные системы счисления, логарифмическая интервальная характеристика, высокоточные вычисления, длинная арифметика.

Введение

Проблема влияния ошибок округления возникает в большом количестве задач в различных областях знаний, включая вычислительную математику, математическую физику, биохимию, квантовую механику, математическое программирование. Для решения таких задач как голоморфная динамика [1], расчет поверхностей [2], теория чисел [3, 4], исследование гравитационных волн [5], численное интегрирование [6, 7], задачи математической физики [8], задачи линейного и целочисленного программирования [9–13], механика сплошных сред [14], теория мелкой воды [15], квантовая механика [16], преобразования Лапласа

Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта № 18-37-00278 мол_а.

10.25209/2079-3316-2019-10-3-81-127

[17], задачи волнового рассеяния [18], модели Лоренца [19], задачи физики черных дыр [20], моделирование конфигурации макромолекул и метаболизма [21], матричные вычисления [22], может потребоваться точность до 512–2048 бит и более. Наиболее требовательны к точности вычислений задачи экспериментальной математики: для вычислений зачастую требуется точность в несколько тысяч и даже десятков тысяч десятичных цифр [8].

Особое внимание специалистами обращается сегодня на точность, безошибочность, стабильность и воспроизводимость результатов расчетов математических моделей при решении широкого спектра индустриальных и научных задач [23]. При решении таких задач требуется выполнение огромного количества арифметических операций с плавающей точкой, каждая из которых сопровождается погрешностями округления, что приводит к неконтролируемым ошибкам округления и, как следствие, неверному результату. Чтобы получить достоверные результаты при решении ряда задач моделирования и симуляции, необходимо повысить точность представления операндов до 256–2048 бит и более. В связи с этим все большую актуальность приобретает арифметика повышенной точности, позволяющая использовать числа произвольной, многократно превышающей размер машинного слова, точности.

Для решения задач в сверхбольших числовых диапазонах сегодня применяются как программные пакеты высокоточной позиционной арифметики, так и специализированные аппаратные средства. Библиотеки длинной арифметики позволяют работать с числами теоретически неограниченной длины (практически размер ограничен объемом доступной оперативной памяти) [24].

В связи с этим активно проводятся исследования, связанные с модернизацией известных и разработкой новых подходов к численной обработке информации сверхбольшой разрядности для достоверных и высокоточных расчетов. Можно выделить два направления работ по повышению скорости вычислений при выполнении расчетов в сверхбольших числовых диапазонах. К первому направлению относятся работы по модернизации и созданию новых технологий гибридных вычислений и обработки данных — гибридные системы кодирования информации. Примером такого использования является библиотека высокоточной модулярно-позиционной арифметики [25, 26], в которой использованы системы счисления в остаточных классах (СОК) и

вычисления в интервальной арифметике. Другим направлением является разработка специализированных аппартных средств поддержки операций над «сверхбольшими» операндами с использованием ПЛИС (FPGA) и системы на кристалле (СнК) [27–32]. Применение таких спецпроцессоров позволяет сократить время расчетов по сравнению с программными решениями до нескольких десятков раз, но недостатки, присущие позиционной длинной арифметике, сохраняются [31].

Целью данного исследования является разработка способов представления числовой информации и методов выполнения арифметических операций для повышения точности и скорости критичных к ошибкам округления вычислений. Для ее решения необходимо определить способы представления длинных чисел, разработать быстрые методы выполнения основных арифметических операций.

1. Высокоточные вычисления

К числу задач, для решения которых требуется высокая точность, относятся:

- задачи волнового рассеяния [18] (32–64 дес. цифр);
- моделирование «поведения» сверхновых звезд и черных дыр [20] (32–64 дес. цифр);
- моделирование конфигурации макромолекул и метаболизма [21] (32–64 дес. знаков);
- задачи оптимального управления [33] (60–90 дес. цифр);
- прямое и обратное преобразование Лапласа [17] (60–90 дес. цифр);
- задачи многих тел [34] (32–120 дес. цифр);
- матричные логарифмы [22] (100–1000 дес. цифр);
- интегралы Изинга и Фейнмана, требующиеся для решения задач математической физики [7] (100–1000 дес. знаков);
- модели Лоренца [19] (до 1000 дес. цифр);
- задачи экспериментальной математики, требующие для своего выполнения от нескольких десятков [2, 6] до нескольких тысяч [5] и даже десятков тысяч [4, 8] десятичных цифр.

Наиболее популярными пакетами работы с длинными двоичными числами являются ARPREC, MPFUN90, DDFUN, FMLIB, FMZM90, QD, GMP, MPFR++, NTL, PARI/GP, CLN, HPAlib, Predicates, GQD,

GARPREC, MatLab, Matematica, Maple. Данные программные пакеты поддерживают вычисления в широком диапазоне форматов высокой разрядности, поддерживающих правила вычислений стандарта IEEE 754, а также содержат широкий спектр поддерживаемых математических функций, что дает возможность использования их для решения различных прикладных задач.

Основными средствами повышения точности и достоверности для указанных выше задач являются такие, как:

- форматы double-double и quad-double [**20**, **21**];
- библиотеки длинной арифметики: ARPREC [**6**, **17**, **18**], MPFR [**4**, **8**, **19**], встроенные длинные типы языков c# [**34**] и Pyton [**2**];
- высокоточные расширения математических пакетов MATLAB [**22**, **33**], Matematica [**1**, **5**], Maple [**7**, **15**];
- символьные вычисления в MATLAB [**35**], Maple [**14**, **16**, **36**, **37**], Mathematica [**38**];
- специализированные программы и комплексы для универсальных процессоров, графических процессоров или многоядерных архитектур, поддерживающие безошибочные дробно-рациональные вычисления [**10**];
- алгоритмические решения, различные методы итерационного уточнения [**13**] и контроля ошибок округления [**9**, **11**], а также использование смешанных символьных и численных расчетов [**12**].

Главным недостатком перечисленных выше пакетов является резкое снижение быстродействия по сравнению со стандартной арифметикой в сотни и тысячи раз [**4**, **7**, **8**, **12**, **33**] вследствие необходимости алгоритмической обработки цепочек межзнаковых переносов «длинных» операндов. По сравнению с обычной 64-битной арифметикой стандарта IEEE 754 при переходе к вычислениям в «длинной» арифметике» скорость вычислений снижается в разы и десятки раз. При значительном увеличении разрядности наблюдается квадратичное увеличение времени выполнения. Как следствие, время решения задачи становится неприемлемым для практики.

Помимо перечисленных выше задач, длинная арифметика используется в криптосистемах для решения задач эллиптической криптографии [**31**, **39**, **40**]. Стоит отметить, что зачастую в приложениях криптографии используются системы остаточных классов

для ускорения выполнения операций над длинными целыми числами [41–44]. Разработаны также специализированные криптографические процессоры полностью на основе систем остаточных классов [45]. Стоит отметить, что для решения задач криптографии требуется высокоточная целочисленная арифметика, в то время как для задач моделирования требуется высокоточная вещественная арифметика.

2. Применение недвоичных систем счисления для организации высокоточных вычислений

Как было показано ранее, достаточно широкий спектр прикладных задач не может быть достоверно решен с применением классической технологии вычислений по стандарту IEEE754 без учета его методологических особенностей на уровне конструкций вычислительных программ (что, естественно снижает скорость вычислений). В связи с этим активно проводятся исследовательские и опытно-конструкторские работы по модернизации известных методов, созданию программно-эмулируемых и программно-аппаратных реализаций новых форматов данных и стандартов обработки информации для высокоточных и достоверных расчетов, что особенно актуально для распределённых и параллельных вычислений [46].

Выбор оптимального способа представления данных зависит от специфики решаемых задач. Так, в задачах с преобладанием операций сложения и умножения, а также большой размерностью операндов, традиционно используются системы остаточных классов [47–49]. Для задач, не критичных к ошибкам округления и не требующих высокой точности, с преобладанием операций умножения, деления и возведения в степень, целесообразно использовать логарифмическую систему счисления. Для контроля ошибок округления при представлении чисел и выполнении арифметических операций используется интервальная система счисления (интервальная арифметика).

Представляется перспективным использовать для высокоточных вычислений гибридную систему счисления – модулярно-позиционную интервально-логарифмическую систему счисления, сочетающую достоинства систем остаточных классов, интервальных и логарифмических систем счисления. Для представления длинных чисел предлагается использовать специальную гибридную форму с применением модулярно-позиционных систем счисления для кодирования значений чисел и

интервально-логарифмических систем счисления для кодирования позиционной оценки модулярного числа, необходимой при выполнении немодульных операций.

2.1. Интервальная арифметика

Интервальная арифметика — один из способов получить достоверные результаты. В интервальной арифметике вместо чисел, дискретных точек на числовой оси, используется пара чисел, представляющих собой границы закрытых интервалов, или отрезков числовой оси, что позволяет выполнять расчеты с учетом погрешностей в исходных данных, в том числе полученных экспериментально и представленных в ЭВМ с ошибками округления [50, 51]. При этом все вычисления выполняются таким образом, что результирующий интервал гарантированно содержит точный результат вычислений. В связи с этим интервальная арифметика позволяет повысить достоверность вычислений за счет учета в явном виде влияния ошибок округления [50]. Высокий интерес к интервальной арифметике подтверждается введением в 2015 году стандарта интервальной арифметики IEEE-1788 [52]. В качестве определения точности в интервальной арифметике используется радиус интервала операндов [53] или ширину (диаметр) интервала [54].

Любой интервал задается двумя числами — нижней и верхней границей

$$X \xrightarrow{\text{ИСС}} [\underline{x}, \overline{x}],$$

где $\underline{x}$ и $\overline{x}$ обозначают нижнюю и верхнюю границы интервала. Каждый интервал характеризуют такие величины, как середина (центр) интервала

$$\mathbf{mid}(x) = \frac{1}{2}\left(\overline{x} - \underline{x}\right),$$

и ширина (или радиус — половина ширины)

$$\mathbf{wid}(x) = \overline{x} - \underline{x}.$$

Результат сложения, вычитания и умножения интервалов определяется следующим образом

$$x + y = \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right],$$
$$x - y = \left[\underline{x} + \overline{y}, \overline{x} + \underline{y}\right],$$
$$x \cdot y = \left[\min\left(\underline{x} \cdot \underline{y}, \underline{x} \cdot \overline{y}, \overline{x} \cdot \underline{y}, \overline{x} \cdot \overline{y}\right), \max\left(\underline{x} \cdot \underline{y}, \underline{x} \cdot \overline{y}, \overline{x} \cdot \underline{y}, \overline{x} \cdot \overline{y}\right)\right].$$

2.2. Логарифмические системы счисления

В последние несколько десятилетий возрос интерес к логарифмической системе счисления (ЛСС). Это связано, в первую очередь, с тем, что ЛСС дает серьезное преимущество при выполнении операций умножения, деления и возведения в степень над вещественными числами за счет замены их более простыми операциями сложения и вычитания с фиксированной точкой. Помимо упрощения, другим преимуществом является отсутствие ошибок округления, в то время как при выполнении данных операций с плавающей точкой возникает ошибка округления в половину бита. ЛСС успешно применяется в таких приложениях, как системы адаптивного управления [55], нейронные сети [56, 57], системы 3D графики [58], обработка видео [59], аудио [60] и изображений [61], цифровая обработка сигналов [62], амплитудная модуляция сигналов [59], различные цифровые фильтры, например фильтр Калмана, используемые во многих инженерных и эконометрических задачах [55], матричные операции [60], ДНК-вычисления [63], моделирование физических процессов методами Монте-Карло [64], численное решение уравнений [65].

Традиционно в ЛСС операции умножения, деления и возведения в степень просты в реализации, поскольку для их выполнения необходимы соответственно операции сложения, вычитания, умножения с использованием операндов с фиксированной точкой. Наиболее целесообразно использовать ЛСС в приложениях с преобладающим количеством операций умножения, деления и возведения в степень и небольшим количеством аддитивных операций, например, при использовании 32-разрядных логарифмов необходимо, чтобы задача содержала как минимум 65 % операций умножения или 48 % операций деления [66].

Основным недостатком логарифмических систем счисления является высокая сложность выполнения операций сложения и вычитания, увеличивающаяся с ростом дробной части представления логарифмической величины, не позволяющая использовать логарифмические числа большой разрядности: большинство логарифмических вычислительных устройств ограничено разрядностью 32 бита [62]. Использование ЛСС предоставляет существенные преимущества по сравнению с вещественными числами на операциях умножения и деления. Так время выполнения, площадь на кристалле и потребляемая мощность значительно (на порядок) ниже [67] при использовании ЛСС.

В ЛСС вещественное число X представлено следующим образом [68]

$$X \xrightarrow{\text{ЛСС}} \{z, S_X, L_X = \log_b |X|\},$$

где z — признак нуля, S_X — знак. Признак нуля необходим, так как функция логарифма не определена в точке 0. Значение логарифма $L_X = I.F$, где I — целая часть, F — дробная часть.

Вычисление основных арифметических операций в ЛСС представлено в таблице 1, где $s_b(z) = \log_b(1 + b^z)$, $d_b(z) = \log_b(1 - b^z)$.

Таблица 1. Основные арифметические операции в ЛСС

Операция		Логарифм результата L_Z	Знак результата S_Z
Умножение	$X \cdot Y$	$L_X + L_Y$	$S_X \oplus S_Y$
Деление	X/Y	$L_X - L_Y$	$S_X \oplus S_Y$
Возведение в квадрат	X^2	$2 \cdot L_X$	0
Извлечение квадратного корня	$\sqrt{X}, \quad X \geqslant 0$	$\frac{1}{2} \cdot L_X$	0
Вычисление обратной величины	$\frac{1}{X}, \quad X \neq 0$	$-L_X$	S_X
Возведение в степень в общем случае	$X^a, \quad X \geqslant 0$	$a \cdot L_X$	0
Смена знака	$-X$	L_X	$\overline{S_X}$
Сложение	$X + Y$	$L_X + s_b(L_Y - L_X)$	
Вычитание	$X - Y$	$L_X + d_b(L_Y - L_X)$	

2.3. Системы остаточных классов

Система остаточных классов (СОК, модулярная система) — это непозиционная система счисления, в которой длинные целые числа представлены набором независимых цифр, называемых остатками [69–71].

Преимуществом СОК является простота выполнения операций сложения, вычитания, умножения (так называемых модульных операций) по сравнению с арифметикой в позиционных системах счисления, а существенным недостатком — сложность выполнения немодульных операций, таких как операции масштабирования, деления, определения

знака, сравнения и определения переполнения диапазона представления чисел и т.д. Другим существенным преимуществом вычислений с применением СОК является более простая аппаратно-техническая реализация вычислительных устройств и, как следствие, их более низкое энергопотребление по сравнению с устройствами на основе вычислений в позиционных системах счисления. Учитывая особенности организации вычислений в СОК, их применение наиболее эффективно для решения задач, алгоритмизация которых реализована с существенным преобладанием модульных операций, например, задач криптографии [**41**–**44**, **72**, **73**].

Пусть $\mathcal{B} = \{p_1, p_2, ..., p_n\}$ — набор целых, попарно взаимно простых чисел, причем $P = \prod_{i=1}^{n} p_i$, т.е.

$$\mathcal{B} = \{p_1, p_2, ..., p_n\},\ \gcd(p_i, p_j) = 1,\ i \neq j.$$

Представления чисел в СОК основаны на изоморфизме колец вычетов

$$Z_{p_1} \times ... \times Z_{p_n} \simeq Z_{p_1 \times ... \times p_n},$$

где Z_i обозначает кольцо вычетов по модулю p_i.

Это означает, что число в СОК представлено кортежем чисел

$$X \xrightarrow{\text{СОК}} \langle x_1, x_2, \ldots, x_n \rangle,$$

где $x_i = X \mod p_i \equiv |X|_{p_i}$ – i-й остаток от деления числа X по i-му модулю p_i:

$$x_i = |X|_{p_i} = X - \left\lfloor \frac{X}{p_i} \right\rfloor \cdot p_i, i = 1, 2, \ldots, n,$$

где $\left\lfloor \frac{X}{p_i} \right\rfloor$ — целая часть частного $\frac{X}{p_i}$, $\mathcal{B} = \{p_1, p_2, ..., p_n\}$ — набор оснований или базис СОК.

Изоморфизм колец также сохраняется и в отношении основных арифметических операций: сложения, вычитания и умножения.

Пусть даны два числа $X \in [0, P)$ и $Y \in [0, P)$, обозначим $* \in \{+, -, \times\}$, тогда

$$|X * Y|_P \leftrightarrow \langle |X_1 * Y_1|_{p_1}, ..., |X_n * Y_n|_{p_n} \rangle.$$

Важнейшими преимуществами СОК являются отсутствие межразрядных переносов и возможность параллельной обработки разрядов.

Применение СОК обеспечивает такие преимущества, как повышенную производительность и надежность, простоту аппаратной реализации, контроль и восстановление ошибок при вычислениях.

Несмотря на ранее отмеченные преимущества применения СОК, их существенными недостатком является отсутствие «быстрых» методов и приемлемых для практики алгоритмов выполнения «не табличным» способом немодульных операций без предварительного перехода из модулярной системы счисления в иную. К таким операциям относятся: операций сравнения операндов, операция деления, операции масштабирования данных, логические операции над данными, в том числе операции сдвигов. Эти особенности сужают область эффективного применения технологии обработки данных в СОК до уровня оперирования данными малой разрядности и только в формате целых чисел.

Для решения проблемы реализации немодульных операций над данными и обработки данных при использовании СОК перспективным решением является переход на гибридные технологии вычислений и обработки данных с применением смешанных систем кодирования данных. В связи с этим использование различных вариантов применения гибридного подхода к вычислениям и обработке данных представляется целесообразным с точки зрения обеспечения высокой скорости вычислений.

3. Модулярный интервально-логарифмический формат представления данных

В данной работе предлагается объединенить преимущества СОК, ЛСС и интервальной арифметики: для высокоточных вычислений в сверхбольших числовых диапазонах предлагается новая гибридная модулярно-позиционная интервально логарифмическая форма представления чисел, в которой мантисса представлена в СОК, что позволяет представлять числа произвольной разрядности, а приближенная оценка абсолютной величины мантиссы — в интервально-логарифмическом формате, позволяющем выполнять немодульные операции сравнения, а также быстрого определения коэффициентов масштабирования.

3.1. Интервально-логарифмический формат представления вещественных чисел

Интервальная арифметика позволяет повысить достоверность вычислений за счет учета в явном виде влияния ошибок округления [50, 51]. Ширина интервала результата при выполнении арифметических операций в интервальной арифметике является не только критерием точности, но и критерием достоверности [74], поэтому снижение влияния погрешностей направленных округлений при выполнении арифметических операций, то есть снижение скорости расширения интервала при вычислениях, является одной из важных задач. Совмещение преимуществ логарифмической и интервальной арифметик может существенно увеличить точность и достоверность компьютерных вычислений.

Существенными преимуществами интервальной логарифмической арифметики по сравнению с позиционной являются, во-первых, более высокая точность, во-вторых, простота выполнения мультипликативных операций, включая операцию масштабирования.

Положительное число в интервально-логарифмическом представлении выглядит следующим образом:

$$X \xrightarrow{\text{ИЛСС}} \{\underline{L_X} = \underline{\log_b |X|}; \overline{L_X} = \overline{\log_b |X|}\}, \tag{1}$$

где $\log_b$ — логарифм числа по основанию b, вычисленный с округлением к минус и плюс бесконечности соответственно, X — модуль числа, представленный в позиционной системе счисления.

В качестве эксперимента провели серию аддитивных и мультипликативных операций с использованием программной модели (процессор Intel® Core™ i3 560, 3.33 ГГц, ОС Windows Server 2008 R2 Standard). Сравнивались результаты вычислений с использованием 32-разрядных логарифмических интервалов в формате с фиксированной точкой и 32-разрядных чисел с плавающей точкой стандарта IEEE 754. Поскольку сравнивать ширину вещественного и непосредственно логарифмического интервала не корректно, результат выполнения в логарифмическом интервале восстанавливался до вещественного значения.

По сравнению с позиционной интервальной арифметикой применение логарифмической интервальной арифметики обеспечивает более высокую точность вычислений при выполнении «очень длинных»

итерационных расчетов (рисунки 1*а*,1*б*) — диаметр интервала результата расширяется при выполнении серии операций суммирования в ≈ 5 раз медленнее, а умножения в 50–100 раз медленнее, что естественно повышает точность выполняемых вычислений и позволяет контролировать вычислительный процесс аналогично изложенному для модулярно-позиционно-интервальных вычислений в [26].

(*а*) при итерационном умножении

(*б*) при итерационном суммировании

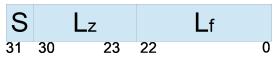
Рисунок 1. Расширение интервала

3.2. Перевод вещественного числа в интервально-логарифмический формат

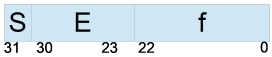
Вещественное число в стандарте IEEE-754 [75] представлено следующим образом (рисунок 2*а*):

$$X = (-1)^S \times 1.f \times 2^e,$$

где S — знак числа, f — беззнаковая дробная часть, $M = 1.f$ — мантисса, e — порядок числа.



(*а*) в формате с плавающей точкой (IEEE Single)



(*б*) в логарифмическом формате с фиксированной точкой (Log 8.23)

Рисунок 2. Представление 32-разрядных чисел

Вещественное число в логарифмическом формате представлено следующим образом (рисунок 2*б*):

$$X = (-1)^S \times b^{L_X},$$

где S — знак числа, L_X — логарифм числа по основанию b. При этом L_X представлено в формате с фиксированной запятой, L_z — целая часть, L_f — дробная часть.

Величина логарифма числа, представленного в формате с плавающей точкой $A = (-1)^S \times 1.f \times 2^{E-E_0}$, где S — знак числа, $1.f$ — нормализованная мантисса числа $M = 1.f \in [1; 2)$, $e = E - E0$ — порядок числа, E_0 — смещение порядка, вычисляется следующим образом:

$$\log_b(X) = \log_b(M \cdot 2^{E-E_0}) = \log_b(M) + (E - E_0) \cdot \log_b 2.$$

Таким образом, основная сложность в вычислении логарифма числа заключается в вычислении значения логарифма мантиссы M числа: $\log_b(M) = \log_b(1 + f)$.

Определим интервальную логарифмическую характеристику мантиссы числа как целочисленный интервал (отрезок) $[L_l, L_h]$, нижняя и верхняя границы которого представлены r-разрядными двоичными числами без знака с фиксированной точкой и вычисляются следующим образом:

$$L_l = \lfloor 2^r \cdot \log_b M \rfloor,$$
$$L_h = L_l + 1,$$

где $\lfloor\ \rfloor$ — целая часть выражения, $\log_b M$ — значение логарифма числа, вычисленное с точностью r двоичных разрядов.

По стандарту чисел с плавающей точкой IEEE 754 мантисса принадлежит диапазону $M = 1.f \in [1; 2)$. Для вычисления интервально-логарифмического числа воспользуемся способом вычисления двоичного логарифма, предложенного фирмой Intel для процессоров

семейства Itanium [76]. Представим мантиссу M числа следующим образом:

$$M = \frac{1 + (1 + M_{\mu_i} + M_{\phi_i}) \cdot r_{\mu_i} - 1}{r_{\mu_i}}, \tag{2}$$

где $M = 1 + M_{\mu_i} + M_{\phi_i}$, μ_i — определяется старшими i разрядами дробной части мантиссы, ϕ_i — определяется младшими $t - 1 - i$ разрядами дробной части мантиссы, $r_{\mu_i} = \dfrac{1}{1 + M_{\mu_i}}$. Формирование чисел μ_i, ϕ_i изображено на рисунке 3.

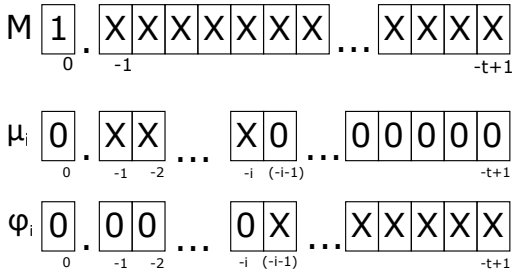


Рисунок 3. Формирование чисел μ_i, ϕ_i при вычислении логарифма мантиссы вещественного числа

Следует отметить, что значение логарифма числа по любому основанию можно получить имея значение натурального логарифма числа, поэтому все дальнейшие вычисления приводятся для натурального логарифма.

Представим мантиссу числа следующим образом:

$$M = \frac{1 + M \cdot y_i - 1}{y_i}. \tag{3}$$

Значение натурального логарифма мантиссы числа, представленной формулой (3), может быть вычислено следующим образом:

$$\ln M = \ln(1 + z) - \ln y_i, \tag{4}$$

где $z = M \cdot y_i - 1 = \dfrac{\phi_i}{2^t + \mu_i 2^i}$, $y_i = \dfrac{2^i}{2^i + \mu_i}$. Поскольку величина z достаточно мала, то $\ln(1 + z) \approx z$ при $z \ll 1$. Тогда формулу (4) можно переписать следующим образом:

$$\ln M = z - \ln \frac{2^i}{2^i + \mu_i} = z + \ln \frac{2^i + \mu_i}{2^i}, \tag{5}$$

где значения $y_i = \dfrac{2^i}{2^i + \mu_i}$, $\ln \dfrac{2^i + \mu_i}{2^i}$ вычислены заранее с точностью r бит и хранятся в подстановочных таблицах, $z = M \cdot y_i - 1$.

Прием, использованный в формуле (2) можно повторить, разбив μ_i на две группы цифр μ_i и μ_j (рисунок 4). Тогда можно использовать 2 таблицы меньшего объема. Все вычисления будем производить в целых числах, подразумевая соотвествующие масштабирующие коэффициенты.

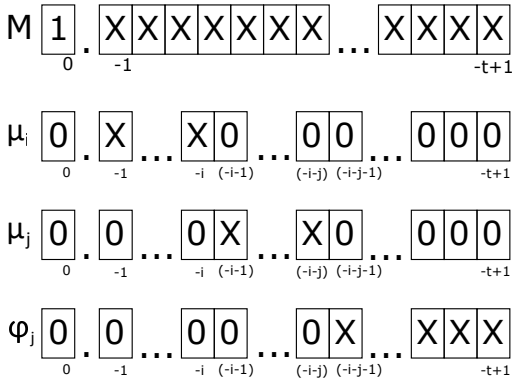


Рисунок 4. Формирование чисел μ_i, μ_j, ϕ_i при вычислении логарифма мантиссы вещественного числа

Мантисса числа равна

$$M = M^* \cdot 2^{-t} = 1 + \mu_i \cdot 2^{-i} + \mu_j \cdot 2^{-i-j} + \phi_j \cdot 2^{-t}$$

или

$$M = 1 + \frac{\mu_i}{2^i} + \frac{\mu_j}{2^{i+j}} + \frac{\phi_j}{2^t}.$$

Представим мантиссу числа следующим образом

$$M = \frac{1 + M \cdot y_i \cdot y_j - 1}{y_i \cdot y_j}, \tag{6}$$

тогда значение натурального логарифма мантиссы числа, представленной формулой (6) может быть вычислено следующим образом:

$$\ln M = \ln(1 + z) - \ln y_i - \ln y_j, \tag{7}$$

где $z = M \cdot y_i \cdot y_j - 1$, $y_i = \dfrac{2^i}{2^i + \mu_i}$, $y_j = \dfrac{2^{i+j}}{2^{i+j} + \mu_i}$. Поскольку величина z достаточно мала, то $\ln(1 + z) \approx z$ при $z \ll 1$.

Тогда формулу (7) можно переписать следующим образом:

(8)
$$\ln M = z - \ln\frac{2^i}{2^i+\mu_i} - \ln\frac{2^{i+j}}{2^{i+j}+\mu_i} = z + \ln\frac{2^i+\mu_i}{2^i} + \ln\frac{2^{i+j}+\mu_i}{2^{i+j}}.$$

Пусть значения $y_i = \dfrac{2^i}{2^i+\mu_i}$, $\ln\dfrac{2^i+\mu_i}{2^i}$, $y_j = \dfrac{2^{i+j}}{2^{i+j}+\mu_i}$, $\ln\dfrac{2^{i+j}+\mu_i}{2^{i+j}}$ вычислены заранее с точностью r бит и хранятся в подстановочных таблицах в виде целых чисел:

$$\nu_i = \left\lfloor \frac{2^i}{2^i+\mu_i}\cdot 2^r \right\rfloor;\ \nu_j = \left\lfloor \frac{2^{i+j}}{2^{i+j}+\mu_j}\cdot 2^r \right\rfloor;$$
$$l_{\mu_i} = \left\lfloor \ln\frac{2^i+\mu_i}{2^i}\cdot 2^r \right\rfloor;\ l_{\mu_j} = \left\lfloor \ln\frac{2^{i+j}+\mu_j}{2^{i+j}}\cdot 2^r \right\rfloor.$$

Нижняя и верхняя границы ζ_1,ζ_2 целочисленного интервальнго значения величины z равны

$$\zeta_1 = \left\lfloor \frac{\phi_i - 2^{t-2\cdot i-j}\cdot\mu_i\cdot\mu_j}{2^t}\cdot\frac{\nu_i}{2^r}\cdot\frac{\nu_j}{2^r}\cdot 2^r \right\rfloor,$$
$$\zeta_2 = \left\lfloor \frac{\phi_i - 2^{t-2\cdot i-j}\cdot\mu_i\cdot\mu_j}{2^t}\cdot\frac{\nu_i+1}{2^r}\cdot\frac{\nu_j+1}{2^r}\cdot 2^r \right\rfloor.$$

Тогда верхняя и нижняя границы логарифмического интервала вычисляются следующим образом:

$$\omega_l = (\zeta_1 - 1) + l_{\mu_i} + l_{\mu_j} = \zeta_1 + l_{\mu_i} + l_{\mu_j} - 1,$$
$$\omega_h = (\zeta_1 + 1) + (l_{\mu_i} + 1) + (l_{\mu_j} + 1) = \omega_l + 4.$$

В данных формулах -1 и $+1$ используются вместо операции направленного округления к минус и плюс бесконечности соответственно по правилам интервальной арифметики. Таким образом, диаметр интервально-логарифмической величины постоянен и равен 4 ulp (unit in the last place — значение младшего разряда).

Таким образом, итоговое значение логарифмического интервала

$$L_l(X) = \frac{1}{\ln b}\downarrow\cdot\frac{\omega_l}{2^r} + (E - E_0)\cdot\underline{\log_b 2},$$
$$L_h(X) = \frac{1}{\ln b}\uparrow\cdot\frac{\omega_h}{2^r} + (E - E_0)\cdot\overline{\log_b 2}.$$

3.2.1. *Результаты экспериментальных исследований*

На основании разработанного способа перевода вещественного числа в интервально-логарифмическое представление была разработана программная модель преобразователя. В ходе серии вычислительных экспериментом путем полного перебора всех значений нормализованных мантисс чисел с плавающей точкой двойной точности стандарта IEEE 754 были определены необходимые значения количества разрядов, отводимых для чисел μ_i, μ_j в разработанном способе, а также значения количества разрядов, отводимых для чисел μ_i в способе Intel [**76**], гарантирующие вычисление логарифмического интервала необходимой точности; под точностью в данном случае подразумевается разрядность дробной части логарифмической величины. Результаты вычислительных экспериментов представлены в таблице 2.

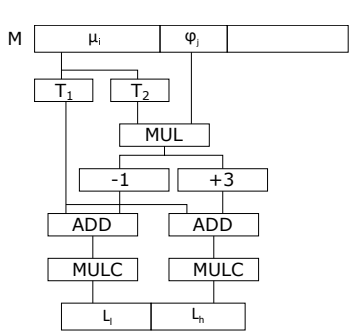
Таблица 2. Минимальные значения разрядностей чисел μ_i, μ_j для обеспечения требуемой точности представления r интервально-логарифмической величины

r, бит	Способ Intel	Разработанный способ	
	i_1, бит	i_2, бит	j_2, бит
16	8	5	4
17	8	5	4
18	9	5	5
19	9	5	5
20	10	6	5
21	11	6	6
22	13	7	7
23	15	8	8
24	17	9	9

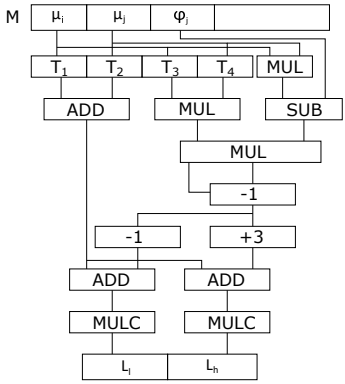
В таблице использованы следующие обозначения: r — точность представления дробной части ИЛХ, i_1 — длина старшей части мантиссы, используемой как индекс в модифицированном интервальном алгоритме Intel, i_2, j_2 — длина старших частей мантиссы, используемых как индексы в разработанном двухтабличном интервальном способе.

3.2.2. *Результаты моделирования*

Устройство преобразования вещественных чисел в интервально-логарифмический формат с использованием одноуровневых подстановочных таблиц (метод Intel) представлено на рисунке 5*а*, а двухуровневых — на рисунке 5*б*.



(*а*) с использованием одноуровневых подстановочных таблиц



(*б*) с использованием двухуровневых подстановочных таблиц

Рисунок 5. Схемы устройства преобразования вещественных чисел в интервально-логарифмический формат

В схемах используются следующие обозначения: M — регистр мантиссы вещественного числа, T — таблицы констант, MUL — умножители, «-1» — комбинационная схема, выполняющая декремент числа, «+3» — комбинационная схема, выполняющая увеличение числа на 3, ADD — двоичные сумматоры, MULC — умножители на константу. Комбинационные схемы коррекции чисел могут быть включены в структуры сумматоров, поэтому при дальнейшем рассмотрении не учитываются.

Результаты моделирования устройств преобразования вещественных чисел в интервально-логарифмический формат представлены в таблице 3 и на рисунке 6. В таблице использованы следующие обозначения: RAM1, RAM2 — общий объем подстановочных таблиц в алгоритмах Intel и разработанном соответственно.

Как видно из таблицы, разработанный способ позволяет сократить объем подстановочных таблиц в 5–16 раз по сравнению с аналогом, в то же время аппаратные затраты на комбинационные схемы возрастают почти в три раза. Также показано, что увеличение точности ведет к значительному повышению аппаратных затрат. Так, для увеличения точности с 16 до 21 бита требуется в 3,5 раза больший объем подстановочных таблиц и почти в 2 раза большее количество логических блоков.

Таблица 3. Результаты моделирования устройств преобразования в интервально-логарифмический формат

r, бит	Способ Intel		Разработанный способ	
	RAM1, бит	модулей ALM	RAM2, бит	модулей ALM
16	8192	123	1536	368
17	8704	133	1632	379
18	18432	143	2304	423
19	19456	158	2432	458
20	40960	207	3840	598
21	86016	242	5376	721

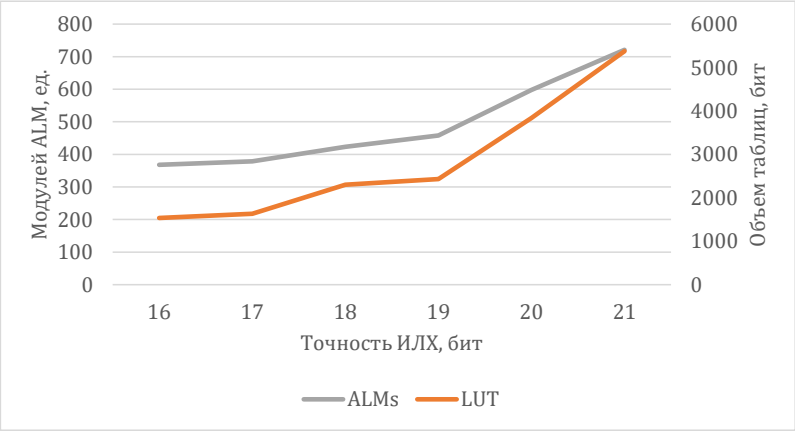


Рисунок 6. Зависимость аппартной сложности устройства преобразования в интервально-логарифмический формат от точности вычисления ИЛХ

3.3. Выполнение немодульных операций

К немодульным операциям СОК относятся: операции сравнения операндов, масштабирования данных, расширения базиса СОК. Для выполнения данных операций необходима информация об абсолютной величине модулярного числа.

3.3.1. *Расширение базиса*

Пусть $\langle X_{p_1}, X_{p_2}, \ldots, X_{p_n} \rangle$ — число, представленное в СОК с основаниями $p_1, p_2, \ldots, p_n$. Требуется найти представление этого же

числа $\langle X_{q_1}, X_{q_2}, \ldots, X_{q_m} \rangle$ в СОК с основаниями $q_1, q_2, \ldots, q_m$, взаимно простыми с $p_1, p_2, \ldots, p_n$, без восстановления числа в позиционную систему счисления. Существует несколько основных групп методов выполнения операции расширения базиса:

- методы, использующие системы счисления со смешанными основаниями [70, 77];
- методы, использующие Китайскую теорему об остатках (КТО) [78],
- приближенные методы [79, 80].

Точные методы расширения базиса были рассмотрены авторами в работе [81], а в работе [82] представлен точный метод расширения базиса СОК, поэтому далее рассматриваются только приближенные методы расширения базиса.

Согласно КТО, позиционное значение числа $X \in [0, P)$, представленного в СОК остатками $X_1, X_2, \ldots, X_n$ по основаниям $p_1, p_2, \ldots, p_n$, вычисляется по формуле

$$X = \left| \sum_{i=1}^{n} \left| X_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot P_i \right|_P = \sum_{i=1}^{n} \left| X_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot P_i - R \cdot P, \tag{9}$$

где $P_i = \frac{P}{p_i}$,$\left| P_i^{-1} \right|_{p_i}$ — мультипликативная инверсия P_i по модулю p_i, $i \in [1; n]$, n — количество модулей, $R \in [0, n-1]$ — позиционный индекс, $P = \prod_{i=1}^{n} p_i$.

Формально, коэффициент R может быть получен по формуле

$$R = \frac{1}{P} \sum_{i=1}^{n} \left| X_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot P_i - \frac{X}{P} = \left\lfloor \frac{1}{P} \sum_{i=1}^{n} \left| X_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot P_i \right\rfloor, \tag{10}$$

где $\lfloor\ \rfloor$ означает целую часть выражения.

Вычисление точного значения коэффициента R непосредственно по формуле (10) является трудоемкой задачей, поскольку требует вычислений с использованием чисел большой разрядности.

Обозначим

$$\tilde{X} = \frac{X + R \cdot P}{P} = \sum_{i=1}^{n} \left| X_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot \frac{1}{p_i}. \tag{11}$$

При этом целая часть величины $\tilde{X}$ является позиционным индексом R, а дробная – значением $\dfrac{X}{P}$.

Был предложен метод приближенного вычисления величины R с использованием вещественных интервалов с направленным округлением [**26**].

$$\tilde{X} \in \left[\sum_{i=1}^{n} \left| X_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot \frac{1}{p_i} \downarrow; \sum_{i=1}^{n} \left| X_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot \frac{1}{p_i} \uparrow \right], \tag{12}$$

где $\downarrow$, $\uparrow$ обозначают вычисления в формате с плавающей точкой с направленным округлением с недостатком (к минус бесконечности) и с избытком (к плюс бесконечности) соответственно. Недостатком является необходимость выполнения арифметических операций в формате с плавающей точкой с направленным округлением, включая операцию деления.

В данной работе предлагается заменить трудоемкие операции умножения и деления с направленным округлением в вещественном формате операциями с фиксированной точкой с первоначальной аппроксимацией величины $\dfrac{1}{p_i}$ к целочисленному интервалу $w_i \in [\underline{w_i}, \overline{w_i}]$, где $\underline{w_i}$,$\overline{w_i}$ – получены округлением вверх и вниз до s разрядов величины $\dfrac{2^{q+s-1}}{p_i}$, т.е. $\underline{w_i} = \left\lfloor \dfrac{2^{q+s-1}}{p_i} \right\rfloor$, $\overline{w_i} = \left\lceil \dfrac{2^{q+s-1}}{p_i} \right\rceil$, где модули СОК p_i представлены в виде q-разрядных двоичных чисел (рисунки 7,8).

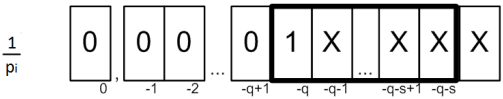


Рисунок 7. Представление величины $\dfrac{1}{p_i}$ в формате с фиксированной точкой

Обозначим

$$\begin{aligned} W_x &= \sum_{i=1}^{n} \left| X_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot w_i, \\ R &= \left\lfloor 2^{1-q-s} \cdot \sum_{i=1}^{n} \left| X_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot w_i \right\rfloor. \end{aligned} \tag{13}$$

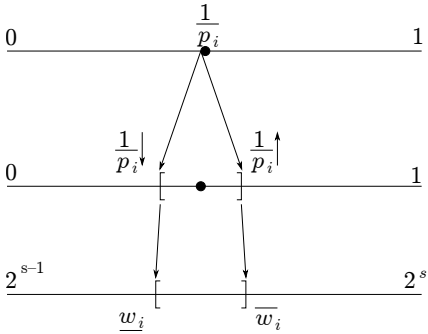


Рисунок 8. Представление величины $\frac{1}{p_i}$ в виде интервала заданной точности

Использование интервальной арифметики гарантирует, что результат вычисление значения W_X будет принадлежать интервалу

$$W_x \in \left[\sum_{i=1}^{n} \left| X_i \cdot \left|P_i^{-1}\right|_{p_i} \right|_{p_i} \cdot \underline{w_i}; \sum_{i=1}^{n} \left| X_i \cdot \left|P_i^{-1}\right|_{p_i} \right|_{p_i} \cdot \overline{w_i}; \right]. \tag{14}$$

Тогда R_X — старшие $\log_2 n$ разрядов числа W_X, X/P — младшие $q+s-1$ разрядов числа W_X. По комбинации значений $\underline{R_X}$, $\overline{R_X}$ можно судить о достоверности вычисления приближенной величины (13) (рис. 9):

- если $\underline{R_X} = \overline{R_X}$, то $R = \underline{R_X} = \overline{R_X}$;
- если $\overline{R_X} - \underline{R_X} = 1$, то необходима дополнительная информация об абсолютной величине числа, представленная интервально-логарифмической характеристикой. Если $X < (P-1)/2$, то $R = \overline{R_X}$, если $X \geq (P-1)/2$, то $R = \underline{R_X}$;
- если $\overline{R_X} - \underline{R_X} \geq 1$, то невозможно что-либо утверждать о значении коэффициента R. Такая ситуация может возникнуть, если точность s коэффициентов $\underline{w_i}$,$\overline{w_i}$ слишком мала.

Таким образом, вычисление с использованием целочисленных интервалов позволяет гарантированно оценить ситуации, когда значения коэффициента R вычислены верно, и когда требуются дополнительные вычисления.

Минимальное значение s, гарантирующее, что $\overline{R_X} - \underline{R_X} \leq 1$, равно $2 + \log_2 n$.

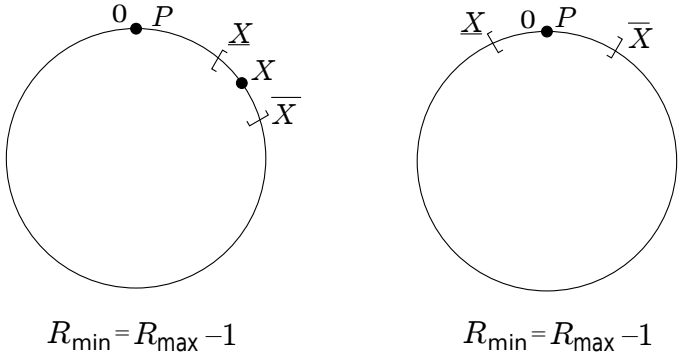


Рисунок 9. Иллюстрация комбинации значений комбинации значений R, позволяющей определить корректность вычисления относительной величины модулярного числа

3.3.2. *Масштабирование чисел, представленных в СОК*

Еще одной важной немодульной операцией является масштабирование, то есть процесс целочисленного деления на заранее определенный делитель (например, степень двойки или один из модулей или их произведение). Масштабирование тесно связано с процедурой расширения базиса и требуется для достаточно широкого круга задач, например, округление при представлении чисел с плавающей точкой в СОК.

Алгоритмы масштабирования представлены в большом числе работ, начиная от классических последовательных методов масштабирования чисел значением одного из оснований СОК (или их произведением) [**70**, **78**, **83**], методов масштабирования коэффициентами, взаимно простыми с основаниями СОК [**84**], алгоритмов масштабирования степенью двойки [**85**] до быстрых алгоритмов для специальных наборов модулей, например $\{2^n - 1, 2^n, 2^n + 1\}$ [**86**], $\{2^n - 1, 2^{n+p}, 2^n + 1\}$ [**87**]. Однако, на сегодняшний день нет эффективных методов масштабирования чисел в СОК, представленных большим числом произвольных модулей средней и высокой разрядности (16–32 бит каждый).

Пусть K — коэффициент масштабирования, Y — результат масштабирования числа X коэффициентом K, тогда результат масштабирования вычисляется по формуле:

$$X = \frac{X - |X|_K}{K}, \tag{15}$$

где $|X|_K$ — остаток от деления числа X по модулю K.

В зависимости от значений коэффициента масштабирования все методы можно разделить на три группы: масштабирование значением модулей СОК или их произведением [78, 83], масштабирование числом, взаимно простым с основаниями СОК [84] и масштабирование степенью двойки [85–87]. Большинство методов масштабирования основано на вычислении коэффициентов системы счисления со смешанными основаниями [77, 83, 87] или на использовании Китайской теоремы об остатках [78, 86].

В случае если коэффициент масштабирования K является взаимно простым с основаниями СОК, в том числе равен степени двойки или степени другого целого числа, то используется следующая общая схема масштабирования коэффициентом K [84, 86].

Шаг 1. Вычисление $|X|_K$, или так называемый этап расширения базиса — получение остатка x_K от деления числа, представленного в СОК остатками $x_1, x_2, \ldots, x_n$ по модулям $p_1, p_2, \ldots, p_n$, на число K. Расширение базиса является одной из основных немодульных операций в СОК.

Шаг 2. Непосредственно масштабирование по каждому модулю по формуле:

$$y_i = \left| |x_i - |X|_K|_{p_i} \cdot \left|K^{-1}\right|_{p_i} \right|_{p_i},$$

где $\left|K^{-1}\right|_{p_i}$ — мультипликативная инверсия по модулю p_i коэффициента K.

Поскольку второй шаг полностью модулярный, основная трудность заключается в нахождении остатка $|X|_K$, то есть в расширении базиса.

Если коэффициент R известен, то может быть вычислен остаток от деления числа по новому модулю с использованием Китайской теоремы об остатках:

$$|X|_K = \left| \left| \sum_{i=1}^{n} \left| X_i \cdot \left|P_i^{-1}\right|_{p_i} \right|_{p_i} \cdot |P_i|_K \right|_K - |R \cdot |P|_K|_K \right|_K. \tag{16}$$

Здесь значения $|P_i|_K$ и $|P|_K$ не зависят от значения конкретного числа в СОК и являются константами для конкретного набора модулей. Таким образом, операция нахождения остатка сводится к нескольким операциям по модулю K, исключается при этом операция нахождения остатка по модулю P.

Деление числа, представленного в СОК, на коэффициент 2^a (масштабирование степенью двойки) выполняется следующим образом. Поскольку степень a может быть произвольной, целесообразно для минимизации накладных расходов выполнять итерационное масштабирование коэффициентом, не превышающим 2^{q-1}.

Пусть $X = \langle x_1, x_2, \ldots, x_n \rangle$ — модулярное число. Для того чтобы найти результат деления модулярного числа $X = \langle x_1, x_2, \ldots, x_n \rangle$ на позиционное число 2^x, необходимо:

(1) Вычислить значение текущего коэффициента масштабирования α. Если $a > q-1$, то вычислить значение $j = \left\lfloor \dfrac{a}{q-1} \right\rfloor$, $\alpha = |a|_{q-1}$; если $a \leqslant q-1$, то $\alpha = a$.

(2) Вычислить значение минимального и максимального значений коэффициентов R_{min} и R_{max} для модулярного числа в соответствии с формулой (14).

(3) Если $R_{min} = R_{max}$, вычислить остаток от деления модулярного числа на $p_{n+1} = 2^{q-1}$:

$$x_{n+1} = |X|_{2^{q-1}} =$$

$$= \left| \left| \sum_{i=1}^{n} \left| \left| x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \cdot |P_i|_{2^{q-1}} \right|_{2^{q-1}} - |R \cdot |P|_{2^{q-1}}|_{2^{q-1}} \right|_{2^{q-1}},$$

где $P_i = \dfrac{P}{p_i}$, $\left| P_i^{-1} \right|_{p_i}$ — мультипликативная инверсия P_i по модулю p_i.

Если $R_{min} \neq R_{max}$, то вычислить остаток от деления модулярного числа на $p_{n+1} = 2^{q-1}$ точным методом расширения базиса, основанным на вычислении коэффициентов системы счисления со смешанными основаниями, описанным в [82] или по значению интервальной логарифмической характеристики. Если $X < (P-1)/2$, то $R = R_{max}$, если $X \geq (P-1)/2$, то $R = R_{min}$; Если $0 < \alpha \leqslant q-1$, то $x_{n+1} = |x_{n+1}|_{2^\alpha}$, если $\alpha = 0$, то уменьшить j на 1.

(4) Вычислить значение $X^* = X - |X|_{2^\alpha}$, для чего вычесть из каждого значения знакопозиций модулярного числа X значение остатка от деления X на 2^α:

$$X^* = X - |X|_{2^\alpha} = \left\langle |x_1 - x_{n+1}|_{p_1}, |x_2 - x_{n+1}|_{p_2}, ..., |x_n - x_{n+1}|_{p_n} \right\rangle.$$

(5) Вычислить значение $X^{**} = \dfrac{X^*}{2^\alpha}$, для чего умножить каждое значение знакопозиций модулярного числа X на мультипликативные инверсии 2^α по соответствующим модулям.

$$X^{**} = \frac{X^*}{2^\alpha} =$$

$$= \left\langle \left| |x_1 - x_{n+1}|_{p_1} \cdot \left| \frac{1}{2^\alpha} \right|_{p_1} \right|_{p_1}, ..., \left| |x_n - x_{n+1}|_{p_n} \cdot \left| \frac{1}{2^\alpha} \right|_{p_n} \right|_{p_n} \right\rangle,$$

где $\left| \dfrac{1}{2^\alpha} \right|_{p_i}$ — мультипликативная инверсия числа 2^α по модулю p_i — результат сравнения $\left| \dfrac{1}{2^\alpha} \right|_{p_i} \cdot |2^\alpha|_{p_i} \equiv 1 \mod p_i$.

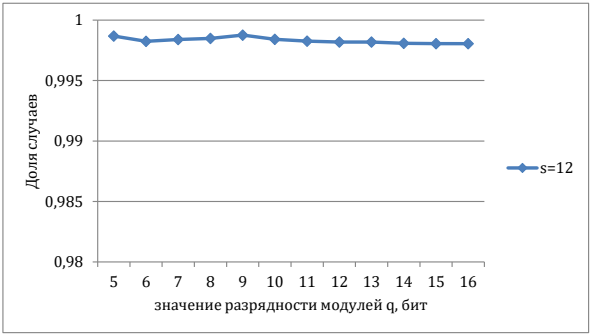
(6) Шаги 2–5 выполнить j раз для $\alpha = q - 1$.

(7) Если на последнем шаге $x_{n+1} \geqslant 2^{\alpha-1}$, то прибавить к X величину $1 = \{1_{p_1}, 1_{p_2}, ..., 1_{p_n}\}$.

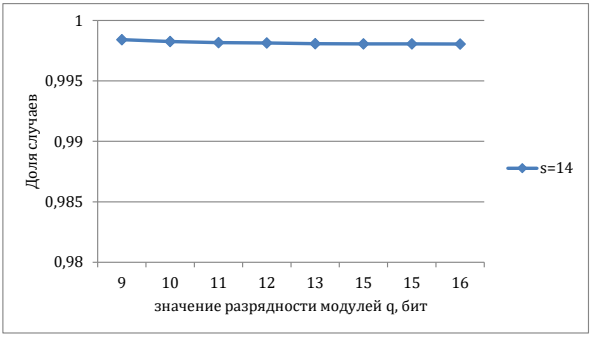
Преимущества данного алгоритма заключаются в том, что в отличие от приближенных методов не требует хранения матриц поправочных коэффициентов, не требует реализации устройств с направленным округлением. Все операции целочисленные. В памяти необходимо хранить только значения мультипликативных инверсий по каждой степени двойки и таблицу коэффициентов w_i. Большинство операций могут быть выполнены параллельно.

3.3.3. *Результаты вычислительных экспериментов*

В качестве исходных данных были рассмотрены несколько вариантов значений разрядности модулей q (от 5 до 16) и количества модулей n (от 8 до 32). Весь диапазон представления чисел $[0, P-1]$ разбивался на $M = 10^{12}$ отрезков, из каждого отрезка выбирался первый элемент, таким образом, формировалась представительная выборка генеральной совокупности. Для каждого из чисел выборки были вычислены модулярное представление, а также интервальное значение величины R. Отдельно вычислялись количество случаев, для которых значения верхней и нижней оценок коэффициентов R были равны (количество попаданий), и для которых значения этих коэффициентов были не равны (количество промахов). На рисунках 10*а*,10*б* представлена зависимость вероятности случаев $R_{min} = R_{max}$.



(*а*) для 8 модулей



(*б*) для 32 модулей

Рисунок 10. Зависимость доли случаев $R_{min} = R_{max}$ от разрядности модулей при фиксированной разрядности s коэффициентов w_i

Для определения среднего количества итераций масштабирования была разработана программная модель модулярно-позиционных интервально-логарифмических вычислений. Из полного диапазона представления модулярных мантисс равномерно выбирали два числа и производили операцию умножения, при этом фиксировали количество итераций масштабирования.

Результаты вычислительного эксперимента представлены в таблицах 4, 5, где n — количество модулей, q — разрядность модулей.

Вычислительные эксперименты показывают, что при выполнении операций масштабирования при умножении произвольных чисел,

Таблица 4. Среднее число итераций масштабирования

n	$q = 8$	q=16
4	1,987718	1,997955
8	3,993018	3,991795
16	7,987538	7,990615
32	15,98973	15,99718
64	31,99013	31,99423

Таблица 5. Количество случаев, для которых было выполнено j итераций масштабирования для $n = 32$

Количество итераций j	Количество случаев	
	q=8	q=16
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	3	0
13	593	0
14	151992	2
15	38909912	152585
16	9960937500	9999847413

среднее количество итераций масштабирования близко к значению $n/2$, что должно быть учтено при оценке времени выполнения операции умножения.

3.3.4. *Результаты моделирования*

Результаты синтеза схем устройств вычисления интервальных целочисленных характеристик на языке описания аппаратуры Verilog в САПР Intel Quartus Prime 17.1 Lite Edition для ПЛИС Altera Cyclone V представлены в таблице 6 и на рисунке 11.

Таблица 6. Результат моделирования устройства, вычисляющего целочисленную интервальную характеристику

q, bits	Схема для R_{min}		Схема для R_{max}		p(R)
	ALMs	Время, нс	ALMs	Время, нс	
9	128	12,816	180	13,004	98,44
10	126	13,946	321	14,045	99,22
11	312	14,344	317	14,351	99,61
12	360	14,276	334	14,673	99,81
13	363	14,723	387	13,310	99,90
14	391	14,280	408	14,962	99,95
15	427	14,450	420	15,100	99,98

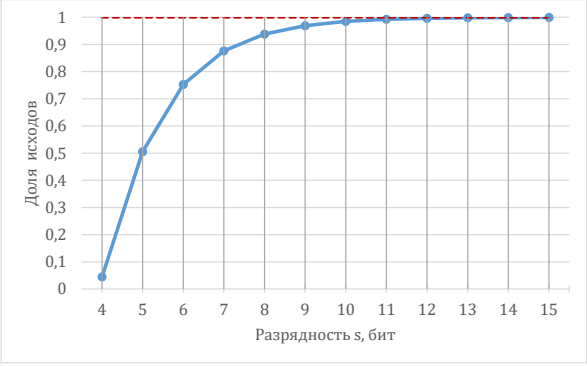


Рисунок 11. Зависимость доли случаев $R_{min} = R_{max}$ от разрядности s коэффициентов w_i

3.4. Апробация разработанного способа представления длинных чисел на примере операции умножения

В рамках данной работы в качестве примера рассмотрена операция умножения двух чисел, представленных в гибридной модулярно-позиционной интервально-логарифмической форме представления.

Разработанный способ умножения [88] сравнили с методами, предложенными в [30] (обозначен «Lei at al., 2013», выполняется за $n^2/4 + 2n + 8$ тактов) и в [31] (обозначен «Ishii, 2017», выполняется за $n^2 + n + 7$ тактов). Результаты сравнения времени выполнения различных методов умножения длинных чисел представлены на рисунке 12.

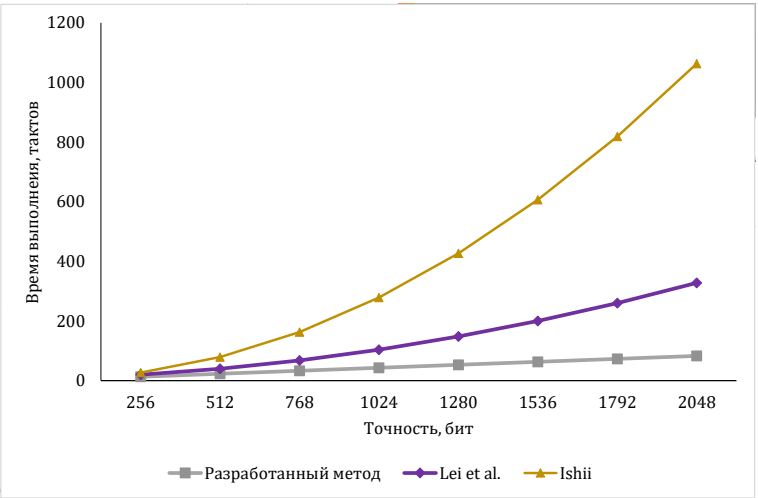


Рисунок 12. Зависимость времени выполнения умножения от разрядности операндов

Результаты синтеза схем устройства умножения на языке описания аппаратуры Verilog в САПР Intel Quartus Prime 17.1 Lite Edition для ПЛИС Altera Cyclone V представлены в таблице 7. Использовались два режима моделирования: с использованием только логических модулей (в таблице обозначены как ALM) и комбинированный с использованием блоков цифровой обработки сигналов (DSP) для построения модулярных каналов.

Таблица 7. Результаты моделирования устройства умножения с масштабированием

n	Точность, бит	Только ALM		ALM и DSP		
		ALMs	Время, нс	DSPs	ALMs	Время, нс
4	64	199	12,472	10	52	15,588
8	128	511	12,231	24	116	27,027
12	192	782	16,008	39	171	26,021
16	256	989	16,872	54	208	26,178

Для сравнения, для размещения параллельного 64-разрядного умножителя требуется задействовать 1886 логических модулей, при этом время задержки на нем составит 13,722 нс, для 128-разрядного умножителя необходимо соответственно 7810 логических модулей с временем выполнения 20,524 нс. Таким образом, разработанное устройство требует на порядок меньше ресурсов ПЛИС при сопоставимой скорости работы.

4. Заключение

Для практической реализации операций обработки данных в нестандартных системах счисления эффективным с точки зрения скорости их выполнения является применение специализированных процессоров в составе универсальных вычислительных узлов, в том числе, и в вычислительных платформах и системах с гибридной и реконфигурируемой архитектурой. В последнем случае процессоры-ускорители могут быть использованы, а лучше всего архитектурно адаптированы, в аппаратном пространстве вычислительной платформы или системы по решения конкретной прикладной задачи. В качестве платформы для реализации спецпроцессоров-ускорителей сегодня популярно применение разнообразных ARM-систем.

Использование модулярно-позиционных и модулярно-логарифмических систем счисления с уникальными форматами представления и обработки информации в различных модификациях обеспечивает следующие основные преимущества при решении пользовательских задач.

- Возможность распараллеливания вычислений до уровня параллельной обработки цифр чисел (модулярные системы счисления

методологически являются параллельными), что позволяет применять для обработки целых и вещественных чисел мало разрядные вычислительные ядра.

- На аппаратном уровне — возможность динамического изменения диапазонов представления целых и вещественных чисел за счет объединением мало разрядных вычислительных ядер в секции для параллельной обработки цифр чисел большей разрядности; использование малоразрядных ядер позволяет обеспечить простоту их технической реализации и соответственно экономное использование ресурсов ПЛИС. Кроме этого экономная реализация вычислительного ядра позволяет реализовать процессоры с большим количеством ядер и соответственно большим уровнем пространственно-параллельной обработки потоков данных, что обеспечивает повышение скорости обработки не за счет повышения тактовой частоты работы процессора, а за счет совмещения во времени параллельной обработки большого числа потоков мало разрядных чисел.
- Возможность реализации вычислений произвольной, в том числе и высокой, точности — двойной, тройной, n-ой точности в зависимости от потребностей пользователя и/или в зависимости от ситуаций, возникающих в процессе вычислений.
- Возможность решения фундаментальной проблемы — проблемы переполнения разрядной сетки, что обеспечивает повышение точности и соответственно достоверности вычислений.

Разработан новый способ представления числовой информации модулярно-позиционная интервально-логарифмическая форма, а также показаны его преимущества по отношению к длинным позиционным формам представления на примере операции умножения.

Результаты исследований базовых операций модулярно-позиционной интервально-логарифмической арифметики высокой точности позволяют рассматривать ее как более эффективную по сложности выполнения немодульных операций сравнения и определения выхода за границы интервала по сравнению с модулярной арифметикой, а также лучшую по точности по сравнению с модулярно-позиционным форматом.

В дальнейшем планируется расширить набор операций, включая такие сложные операции, как деление, вычисление элементарных функций, что позволит организовать полноценные вычисления в новой модулярно-позиционной интервально-логарифмической арифметике.

Список литературы

[1] T. Kawahira. “The Riemann hypothesis and holomorphic index in complex dynamics”, *Experimental Mathematics*, **27**:1 (2018), pp. 37–46. ↑81,84

[2] W. Worden. “Experimental statistics of veering triangulations”, *Experimental Mathematics*, 2018, 22 pp. ↑81,83,84

[3] A. Ash, L. Beltis, R. Gross, W. Sinnott. “Frequencies of successive pairs of prime residues”, *Experimental Mathematics*, **20**:4 (2011), pp. 400–411. ↑81

[4] A. Voros. “Discretized Keiper/Li approach to the Riemann hypothesis”, *Experimental Mathematics*, 2018, 18 pp. ↑81,83,84

[5] N. K. Johnson-McDaniel, A. G. Shah, B. F. Whiting. “Experimental mathematics meets gravitational self-force”, *Physical Review D*, **92**:4 (2015), 044007. ↑81,83,84

[6] D. H. Bailey, J. M. Borwein. “High-precision numerical integration: Progress and challenges”, *Journal of Symbolic Computation*, **46**:7 (2011), pp. 741–754. ↑81,83,84

[7] E. Panzer. “Algorithms for the symbolic integration of hyperlogarithms with applications to Feynman integrals”, *Computer Physics Communications*, **188** (2015), pp. 148–166. ↑81,83,84

[8] D. H. Bailey, J. M. Borwein, J. S. Kimberley, W. Ladd. “Computer discovery and analysis of large Poisson polynomials”, *Experimental Mathematics*, **26**:3 (2017), pp. 349–363. ↑81,82,83,84

[9] K. K. H. Cheung, A. Gleixner, D. E. Steffy. “Verifying integer programming results”, IPCO 2017: Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science, vol. **10328**, Springer, 2017, pp. 148–160. ↑81,84

[10] A. V. Panyukov, V. A. Golodov. «Parallel algorithms of integer arithmetic in radix notations for heterogeneous computation systems with massive parallelism», *Вестн. ЮУрГУ. Сер. Матем. моделирование и программирование*, **8**:2 (2015), с. 117–126 (in English). ↑81,84

[11] M. Miltenberger, T. Ralphs, D. E. Steffy. “Exploring the numerics of branch-and-cut for mixed integer linear optimization”, Operations Research Proceedings 2017, Operations Research Proceedings, Springer, 2018, pp. 151–157. ↑81,84

[12] W. Cook, Th. Koch, D. E. Steffy, K. Wolter. “An exact rational mixed-integer programming solver”, IPCO 2011: Integer Programming and Combinatoral Optimization, Lecture Notes in Computer Science, vol. **6655**, Springer, Berlin–Heidelberg, 2011, pp. 104–116. ↑81,84

[13] A. M. Gleixner, D. E. Steffy, K. Wolter. “Iterative refinement for linear programming”, *INFORMS Journal on Computing*, **28**:3 (2016), pp. 449–464. ↑81,84

[14] A. F. Cheviakov, J. Heß. “A symbolic computation framework for constitutive modelling based on entropy principles”, *Applied Mathematics and Computation*, **324** (2018), pp. 105–118. ↑81,84

[15] M. Wei, J. Cai. “The exact rational solutions to a shallow water wave-like equation by generalized bilinear method”, *Journal of Applied Mathematics and Physics*, **5**:03 (2017), pp. 715–721. ↑81,84

[16] Z. Cao, X. Hou. “A symbolic computation approach to parameterizing controller for polynomial Hamiltonian systems”, *Mathematical Problems in Engineering*, **2014** (2014), 806428, 8 pp. ↑81,84

[17] Z. Krougly, M. Davison, S. Aiyar. “The role of high precision arithmetic in calculating numerical Laplace and inverse Laplace transforms”, *Applied Mathematics*, **8**:04 (2017), pp. 562. ↑82,83,84

[18] L. N. Gergidis, D. Kourounis, S. Mavratzas, A. Charalambopoulos. “Numerical investigation of the acoustic scattering problem from penetrable prolate spheroidal structures using the Vekua transformation and arbitrary precision arithmetic”, *Mathematical Methods in the Applied Sciences*, **41**:13. ↑82,83,84

[19] R. Barrio, A. Dena, W. Tucker. “A database of rigorous and high-precision periodic orbits of the Lorenz model”, *Computer Physics Communications*, **194** (2015), pp. 76–83. ↑82,83,84

[20] G. Khanna. “High-precision numerical simulations on a CUDA GPU: Kerr black hole tails”, *Journal of Scientific Computing*, **56**:2 (2013), pp. 366–380. ↑82,83,84

[21] L. Yang, D. Ma, A. Ebrahim, C. J. Lloyd, M. A. Saunders, B. O. Palsson. “solveME: fast and reliable solution of nonlinear ME models”, *BMC bioinformatics*, **17**:1 (2016), 391. ↑82,83,84

[22] M. Fasi, N. J. Higham. “Multiprecision algorithms for computing the matrix logarithm”, *SIAM Journal on Matrix Analysis and Applications*, **39**:1 (2018), pp. 472–491. ↑82,83,84

[23] R. Iakymchuk, D. Defour, S. Collange, S. Graillat. “Reproducible and accurate matrix multiplication”, SCAN 2015: Scientific Computing, Computer Arithmetic, and Validated Numerics, Lecture Notes in Computer Science, vol. **9553**, Springer, 2015, pp. 126–137. ↑82

[24] M. Cornea. “Precision, accuracy, rounding, and error propagation in exascale computing”, 2013 IEEE 21st Symposium on Computer Arithmetic (7–10 April 2013, Austin, TX, USA), pp. 231–234. ↑82

[25] К. С. Исупов, В. С. Князьков. «Арифметика многократной точности на основе систем остаточных классов», *Программные системы: теория и приложения*, **7**:1 (2016), с. 61–97. ↑82

[26] K. Isupov, V. Knyazkov. “A modular-positional computation technique for multiple-precision floating-point arithmetic”, PaCT 2015: Parallel Computing

Technologies, Lecture Notes in Computer Science, vol. **9251**, Springer, 2015, pp. 47–61. ↑82,92,101

[27] N. Nakasato, H. Daisaka, T. Fukushige, A. Kawai, J. Makino, T. Ishikawa, F. Yuasa. "GRAPE-MPs: Implementation of an SIMD for quadruple/hexuple/octuple-precision arithmetic operation on a structured ASIC and an FPGA", 2012 IEEE 6th International Symposium on Embedded Multicore SoCs (20–22 Sept. 2012, Aizu-Wakamatsu, Japan), 2012, pp. 75–83. ↑83

[28] H. Daisaka, N. Nakasato, T. Ishikawa, F. Yuasa. "Application of GRAPE9-MPX for high precision calculation in particle physics and performance results", *Procedia Computer Science*, **51** (2015), pp. 1323–1332. ↑83

[29] E. El-Araby, I. Gonzalez, T. A El-Ghazawi. "Bringing high-performance reconfigurable computing to exact computations", 2007 International Conference on Field Programmable Logic and Applications (27–29 Aug. 2007, Amsterdam, Netherlands), 2007, pp. 79–85. ↑83

[30] Y. Lei, Y. Dou, J. Zhou. "FPGA-specific custom VLIW architecture for arbitrary precision floating-point arithmetic", *IEICE Transactions on Information and Systems*, **E94.D**:11 (2011), pp. 2173–2183. ↑83,110

[31] M. Ishii, J. Detrey, P. Gaudry, A. Inomata, K. Fujikawa. "Fast modular arithmetic on the Kalray MPPA-256 processor for an energy-efficient implementation of ECM", *IEEE Transactions on Computers*, **66**:12 (2017), pp. 2019–2030. ↑83,84,110

[32] M. J. Schulte, E. E. Swartzlander. "A family of variable-precision interval arithmetic processors", *IEEE Transactions on Computers*, **49**:5 (2000), pp. 387–397. ↑83

[33] B. Pan, Y. Wang, S. Tian. "A high-precision single shooting method for solving hypersensitive optimal control problems", *Mathematical Problems in Engineering*, **2018** (2018), 7908378, 11 pp. ↑83,84

[34] I. V. Grossu, C. Besliu, D. Felea, A. Jipa. "High precision framework for chaos many-body engine", *Computer Physics Communications*, **185**:4 (2014), pp. 1339–1342. ↑83,84

[35] V. Nehra, R. Sehgal. "Symbolic computation of mathematical transforms and its application: A MATLAB computational project-based approach", *IUP Journal of Electrical and Electronics Engineering*, **8**:1 (2015), pp. 53–76. ↑84

[36] M. A. Agwa, A. P. Da Costa. "Using symbolic computation in the characterization of frictional instabilities involving orthotropic materials", *International Journal of Applied Mathematics and Computer Science*, **25**:2 (2015), pp. 259–267. ↑84

[37] E. Dovlo, N. Baddour. "Building a symbolic computer algebra toolbox to compute 2D Fourier transforms in polar coordinates", *MethodsX*, **2** (2015), pp. 192–197. ↑84

[38] J. G. Liu, Z. F. Zeng. "Extended generalized hyperbolic-function method and new exact solutions of the generalized hamiltonian and NNV equations

by the symbolic computation", *Fundamenta Informaticae*, **132**:4 (2014), pp. 501–517. ↑84

[39] S. Asif, Y. Kong. "Highly parallel modular multiplier for elliptic curve cryptography in residue number system", *Circuits, Systems, and Signal Processing*, **36**:3 (2017), pp. 1027–1051. ↑84

[40] Y. Li, J. Wang, X. Zeng, X. Ye. "Fast Montgomery modular multiplication and squaring on embedded processors", *IEICE Transactions on Communications*, **100**:5 (2017), pp. 680–690. ↑84

[41] J.-C. Bajard, L. Imbert. "A full RNS implementation of RSA", *IEEE Transactions on Computers*, **53**:6 (2004), pp. 769–774. ↑85,89

[42] S. Antao, J. C. Bajard, L. Sousa. "RNS-based elliptic curve point multiplication for massive parallel architectures", *The Computer Journal*, **55**:5 (2011), pp. 629–647. ↑85,89

[43] O. Harrison, J. Waldron. "Efficient acceleration of asymmetric cryptography on graphics hardware", AFRICACRYPT 2009: Progress in Cryptology – AFRICACRYPT 2009, Lecture Notes in Computer Science, vol. **5580**, Springer, 2009, pp. 350–367. ↑85,89

[44] K. Bigou, A. Tisserand. "Single base modular multiplication for efficient hardware RNS implementations of ECC", CHES 2015: Cryptographic Hardware and Embedded Systems – CHES 2015, Lecture Notes in Computer Science, vol. **9293**, Springer, 2015, pp. 123–140. ↑85,89

[45] S. Asif, M. S. Hossain, Y. Kong, W. Abdul. "A fully RNS based ECC processor", *Integration*, **61** (2018), pp. 138–149. ↑85

[46] Н. Н. Непейвода. «Использование локализации и переполнения для управления параллельными и распределёнными вычислениями», *Программные системы: теория и приложения*, **8**:3 (2017), с. 87–107. ↑85

[47] N. I. Chervyakov, P. A. Lyakhov, M. G. Babenko, I. N. Lavrinenko, A. V. Lavrinenko, A. S. Nazarov. "The architecture of a fault-tolerant modular neurocomputer based on modular number projections", *Neurocomputing*, **272** (2018), pp. 96–107. ↑85

[48] М. Г. Бабенко, А. Н. Черных, Н. И. Червяков, В. А. Кучуков, В. Миранда-Лопес, Р. Ривера-Родригес, Чж. Ду. «Эффективное сравнение чисел в системе остаточных классов на основе позиционной характеристики», *Труды ИСП РАН*, **31**:2 (2019), с. 187–202. ↑85

[49] D. V. Telpukhov, R. A. Solovyev, V.M. Amerbaev, E.S. Balaka. «Hardware implementation of FIR filter based on number-theoretic fast Fourier transform in residue number system», *Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС)*, 2015, №4, с. 42–42 (in English). ↑85

[50] N. Revol. "Introduction to the IEEE 1788-2015 standard for interval arithmetic", NSV 2017: Numerical Software Verification, Lecture Notes in Computer Science, vol. **10381**, Springer, 2017, pp. 14–21. ↑86,91

[51] F. Johansson. “Arb: efficient arbitrary-precision midpoint-radius interval arithmetic”, *IEEE Transactions on Computers*, **66**:8 (2017), pp. 1281–1292. ↑86,91

[52] *1788-2015 IEEE Standard for Interval Arithmetic*, 2015. ↑86

[53] N. Revol, Ph. Theveny. “Numerical reproducibility and parallel computations: Issues for interval algorithms”, *IEEE Transactions on Computers*, **63**:8 (2014), pp. 1915–1924. ↑86

[54] M. G. Arnold, J. Garcia, M. J. Schulte. “The interval logarithmic number system”, *Proceedings 2003 16th IEEE Symposium on Computer Arithmetic* (15–18 June 2003, Santiago de Compostela, Spain), pp. 253–261. ↑86

[55] U. Lotrič, P. Bulić. “Logarithmic arithmetic for low-power adaptive control systems”, *Circuits, Systems, and Signal Processing*, **36**:9 (2017), pp. 3564–3584. ↑87

[56] U. Lotrič, P. Bulić. “Applicability of approximate multipliers in hardware neural networks”, *Neurocomputing*, **96** (2012), pp. 57–65. ↑87

[57] M. S. Kim, A. A Del Barrio, R. Hermida, N. Bagherzadeh. “Low-power implementation of Mitchell’s approximate logarithmic multiplication for convolutional neural networks”, 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC) (22–25 Jan. 2018, Jeju, South Korea), pp. 617–622. ↑87

[58] H. Kim, B.-G. Nam, J.-H. Sohn, J.-H. Woo, H.-J. Yoo. “A 231-MHz, 2.18-mW 32-bit logarithmic arithmetic unit for fixed-point 3-D graphics system”, *IEEE Journal of Solid-State Circuits*, **41**:11 (2006), pp. 2373–2381. ↑87

[59] A. Avramovć, Z. Babić, D. Raič, D. Strle, P. Bulić. “An approximate logarithmic squaring circuit with error compensation for DSP applications”, *Microelectronics Journal*, **45**:3 (2014), pp. 263–271. ↑87

[60] M. Gautschi, M. Schaffner, F. K. Gürkaynak, L. Benini. “An extended shared logarithmic unit for nonlinear function kernel acceleration in a 65-nm CMOS multicore cluster”, *IEEE Journal of Solid-State Circuits*, **52**:1 (2017), pp. 98–112. ↑87

[61] D. Nandan, J. Kanungo, A. Mahajan. “An error-efficient gaussian filter for image processing by using the expanded operand decomposition logarithm multiplication”, *Journal of Ambient Intelligence and Humanized Computing*, 2018, pp. 1–8. ↑87

[62] J. Coleman, R C. Ismail. “LNS with co-transformation competes with floating-point”, *IEEE Transactions on Computers*, **65**:1 (2016), pp. 136–146. ↑87

[63] J. Le Maire, N. Brunie, F. De Dinechin, J. M. Muller. “Computing floating-point logarithms with fixed-point operations”, 2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH) (10–13 July 2016, Santa Clara, CA, USA), pp. 156–163. ↑87

[64] H. Fu, O. Mencer, W. Luk. “Comparing floating-point and logarithmic number representations for reconfigurable acceleration”, 2006 IEEE International Conference on Field Programmable Technology (13–15 Dec. 2006, Bangkok, Thailand), pp. 337–340. ↑87

[65] M. Chugh, B. Parhami. “Logarithmic arithmetic as an alternative to floating-point: A review”, 2013 Asilomar Conference on Signals, Systems and Computers (3–6 Nov. 2013, Pacific Grove, CA, USA), 2013, pp. 1139–1143. ↑87

[66] M. Haselman, M. Beauchamp, A. Wood, S. Hauck, K. Underwood, K. S. Hemmert. “A comparison of floating point and logarithmic number systems for FPGAs”, 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM’05) (18–20 April 2005, Napa, CA, USA, USA), pp. 181–190. ↑87

[67] R. C. Ismail, J. N. Coleman, N. Norzahiyah, Z. Sauli. “A comparative analysis between logarithmic number system and floating-point ALU”, *Advances in Environmental Biology*, **7**:12 (2013), pp. 3601–3606. ↑87

[68] M. G. Arnold. “Iterative methods for logarithmic subtraction”, *Proceedings IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, ASAP 2003 (24–26 June 2003, The Hague, Netherlands), pp. 315–325. ↑88

[69] A. R. Omondi, B. Premkumar, *Residue Number Systems: Theory and Implementation*, Advances in Computer Science and Engineering Texts, World Scientific, 2007, ISBN 978-1860948664, 296 pp. ↑88

[70] N. S. Szabo, R. I. Tanaka, *Residue Arithmetic and its Applications to Computer Technology*, McGraw-Hill Series in Information Processing and Computers, McGraw-Hill, 1967, 236 pp. ↑88,100,103

[71] И. Я. Акушский, Д. И. Юдицкий. *Машинная арифметика в остаточных классах*, Сов. радио, М., 1968, 440 с. ↑88

[72] K. Bigou, A. Tisserand. “RNS modular multiplication through reduced base extensions”, 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors (18–20 June 2014, Zurich, Switzerland), pp. 57–62. ↑89

[73] J. C. Bajard, J. Eynard, N. Merkiche. “Montgomery reduction within the context of residue number system arithmetic”, *Journal of Cryptographic Engineering*, **8**:3 (2018), pp. 189–200. ↑89

[74] M. Langhammer, B. Pasca. “Single precision natural logarithm architecture for hard floating-point and DSP-enabled FPGAs”, 2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH) (10–13 July 2016, Santa Clara, CA, USA), pp. 164–171. ↑91

[75] *754-2008 – IEEE Standard for Floating-Point Arithmetic*, IEEE, 2008. ↑92

[76] M. Cornea, J. Harrison, P. T. P. Tang. *Scientific Computing on Itanium-Based Systems*, Intel Press, Hillsboro, 2002, ISBN 978-0971288775, 280 pp. ↑94,97

[77] M. Czyzak, R. Smyk, Z. Ulman. "Pipelined scaling of signed residue numbers with the mixed-radix conversion in the programmable gate array", *Poznan University of Technology Academic Journals. Electrical Engineering*, 2013, no.76, pp. 89–99. ↑100,104

[78] A. P. Shenoy, R. Kumaresan. "Fast base extension using a redundant modulus in RNS", *IEEE Transactions on Computers*, **38**:2 (1989), pp. 292–297. doi ↑100,103,104

[79] S. Kawamura, M. Koike, F. Sano, A. Shimbo. "Cox-Rower architecture for fast parallel Montgomery multiplication", EUROCRYPT 2000: Advances in Cryptology – EUROCRYPT 2000, Lecture Notes in Computer Science, vol. **1807**, Springer, 2000, pp. 523–538. doi ↑100

[80] K. C. Posch, R. Posch. "Base extension using a convolution sum in residue number systems", *Computing*, **50**:2 (1993), pp. 93–104. doi ↑100

[81] А. С. Коржавина, В. С. Князьков. «Методы расширения базиса в системе остаточных классов: обзор и анализ вычислительной сложности», *Современные наукоемкие технологии*, 2017, №12, с. 37–42. ✻ ↑100

[82] А. С. Коржавина, В. С. Князьков. «Метод расширения базиса систем остаточных классов с применением систем счисления со смешанными основаниями», *Научно-технический вестник Поволжья*, 2017, №6, с. 204–207. URL ↑100,105

[83] G. A. Jullien. "Residue number scaling and other operations using ROM arrays", *IEEE Transactions on Computers*, **C-27**:4 (1978), pp. 325–336. doi ↑103,104

[84] Y. Kong, B. Phillips. "Fast scaling in the residue number system", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **17**:3 (2009), pp. 443–447. doi ↑103,104

[85] S. Ma, J. Hu, Y. Ye, L. Zhang, X. Ling. "A 2^n scaling scheme for signed RNS integers and its VLSI implementation", *Science in China Series F: Information Sciences*, **53**:1 (2010), pp. 203–212. doi ↑103,104

[86] C. H. Chang, J. Y. S. Low. "Simple, fast, and exact RNS scaler for the three-moduli set three-moduli set $\{2^n-1, 2^n, 2^n+1\}$", *IEEE Transactions on Circuits and Systems I: Regular Papers*, **58**:11 (2011), pp. 2686–2697. doi ↑103,104

[87] A. Hiasat. "Efficient RNS scalers for the extended three-moduli set $\{2^n-1, 2^{n+p}, 2^n+1\}$", *IEEE Transactions on Computers*, **66**:7 (2017), pp. 1253–1260. doi ↑103,104

[88] А. С. Коржавина, В. С. Князьков. *Способ организации выполнения операции умножения двух чисел в модулярно-логарифмическом формате*

представления с плавающей точкой на гибридных многоядерных процессорах, Патент № 2666285, 2018. ↑110

Поступила в редакцию 21.02.2019
Переработана 18.09.2019
Опубликована 30.09.2019

Рекомендовал к публикации *д.ф.-м.н. Н. Н. Непейвода*

Пример ссылки на эту публикацию:

А. С. Коржавина, В. С. Князьков. «Реализация высокоточных вычислений в базисе модулярно-интервальной арифметики». *Программные системы: теория и приложения*, 2019, **10**:3(42), с. 81–127.
10.25209/2079-3316-2019-10-3-81-127
http://psta.psiras.ru/read/psta2019_3_81-127.pdf

Об авторах:

Анастасия Сергеевна Коржавина

Старший преподаватель кафедры электронных вычислительных машин Вятского государственного университета. Область научных интересов: высокоточные вычисления, компьютерная арифметика, реконфигурируемые вычислительные системы.

0000-0001-8270-2097
e-mail: as_korzhavina@vyatsu.ru

Владимир Сергеевич Князьков

Главный научный сотрудник НИИ прикладных и фундаментальных исследований Пензенского государственного университета, доктор технических наук. Область научных интересов: параллельные вычислительные архитектуры, высокопроизводительные вычисления, реконфигурируемые вычислительные системы.

0000-0003-3820-6541
e-mail: kniazkov@list.ru

CSCSTI 50.07.05, 50.09.31
UDC 004.222.3:681.5.07+004.421.4

Anastasia S. Korzhavina, Vladimir S. Knyazkov. *High-precision computations using residue-interval arithmetic on FPGAs.*

Abstract. The problem of round-off errors arises in a large number of issues in various fields of knowledge, including computational mathematics, mathematical physics, biochemistry, quantum mechanics, mathematical programming. Today, experts place particular emphasis on accuracy, fault tolerance, stability, and reproducibility of computation results of numerical models when solving a wide range of industrial and scientific problems, such as: mathematical modeling and structural designs of aircrafts, cars, ships; process modeling and computations for solving large-scale problems in the field of nuclear physics, aerodynamics, gas, and hydrodynamics; problems on reliable predictive modeling of climatic processes and forecasting of global changes in the atmosphere and water environments; faithful modeling of chemical processes and synthesis of pharmaceuticals, etc.

Floating-point arithmetic is the dominant choice for most scientific applications. However, there are a lot of unsolvable with double-precision arithmetic problems. A vast number of floating-point arithmetic operations would be required to solve such problems. Each operation carries round-off errors leading to uncontrolled round-off errors and, consequently, incorrect results. Many modeling and simulation problems need to increase the accuracy of number representation to 100-1000 decimal digits or more to obtain reliable results. In this regard, arbitrary-precision arithmetic is becoming ever important. With this arithmetic, one can use numbers, whose arbitrary precision is many times greater than the word length of the conventional system.

The paper proposes a new way of representing integers and floats for computations in super-large ranges – hybrid residue-positional interval logarithmic number representation for performing high-precision and reliable calculations in super-large numerical ranges.

Key words and phrases: residue arithmetic, hybrid number systems, the interval logarithmic number evaluation, high-precision computations, long numbers.

2010 *Mathematics Subject Classification*: 68M07; 65G30, 65D99

10.25209/2079-3316-2019-10-3-81-127

References

[1] T. Kawahira. "The Riemann hypothesis and holomorphic index in complex dynamics", *Experimental Mathematics*, **27**:1 (2018), pp. 37–46. ↑81,84

[2] W. Worden. "Experimental statistics of veering triangulations", *Experimental Mathematics*, 2018, 22 pp. ↑81,83,84

[3] A. Ash, L. Beltis, R. Gross, W. Sinnott. "Frequencies of successive pairs of prime residues", *Experimental Mathematics*, **20**:4 (2011), pp. 400–411. ↑81

[4] A. Voros. "Discretized Keiper/Li approach to the Riemann hypothesis", *Experimental Mathematics*, 2018, 18 pp. ↑81,83,84

[5] N. K. Johnson-McDaniel, A. G. Shah, B. F. Whiting. "Experimental mathematics meets gravitational self-force", *Physical Review D*, **92**:4 (2015), 044007. ↑81,83,84

[6] D. H. Bailey, J. M. Borwein. "High-precision numerical integration: Progress and challenges", *Journal of Symbolic Computation*, **46**:7 (2011), pp. 741–754. ↑81,83,84

[7] E. Panzer. "Algorithms for the symbolic integration of hyperlogarithms with applications to Feynman integrals", *Computer Physics Communications*, **188** (2015), pp. 148–166. ↑81,83,84

[8] D. H. Bailey, J. M. Borwein, J. S. Kimberley, W. Ladd. "Computer discovery and analysis of large Poisson polynomials", *Experimental Mathematics*, **26**:3 (2017), pp. 349–363. ↑81,82,83,84

[9] K. K. H. Cheung, A. Gleixner, D. E. Steffy. "Verifying integer programming results", IPCO 2017: Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science, vol. **10328**, Springer, 2017, pp. 148–160. ↑81,84

[10] A. V. Panyukov, V. A. Golodov. "Parallel algorithms of integer arithmetic in radix notations for heterogeneous computation systems with massive parallelism", *Bulletin of the South Ural State University, Series: Mathematical Modelling, Programming and Computer Software*, **8**:2 (2015), pp. 117–126. ↑81,84

[11] M. Miltenberger, T. Ralphs, D. E. Steffy. "Exploring the numerics of branch-and-cut for mixed integer linear optimization", Operations Research Proceedings 2017, Operations Research Proceedings, Springer, 2018, pp. 151–157. ↑81,84

[12] W. Cook, Th. Koch, D. E. Steffy, K. Wolter. "An exact rational mixed-integer programming solver", IPCO 2011: Integer Programming and Combinatoral Optimization, Lecture Notes in Computer Science, vol. **6655**, Springer, Berlin–Heidelberg, 2011, pp. 104–116. ↑81,84

[13] A. M. Gleixner, D. E. Steffy, K. Wolter. "Iterative refinement for linear programming", *INFORMS Journal on Computing*, **28**:3 (2016), pp. 449–464. ↑81,84

[14] A. F. Cheviakov, J. Heß. "A symbolic computation framework for constitutive modelling based on entropy principles", *Applied Mathematics and Computation*, **324** (2018), pp. 105–118. ↑81,84

[15] M. Wei, J. Cai. "The exact rational solutions to a shallow water wave-like equation by generalized bilinear method", *Journal of Applied Mathematics and Physics*, **5**:03 (2017), pp. 715–721. ↑81,84

[16] Z. Cao, X. Hou. "A symbolic computation approach to parameterizing controller for polynomial Hamiltonian systems", *Mathematical Problems in Engineering*, **2014** (2014), 806428, 8 pp. ↑81,84

[17] Z. Krougly, M. Davison, S. Aiyar. "The role of high precision arithmetic in calculating numerical Laplace and inverse Laplace transforms", *Applied Mathematics*, **8**:04 (2017), pp. 562. ↑82,83,84

[18] L. N. Gergidis, D. Kourounis, S. Mavratzas, A. Charalambopoulos. "Numerical investigation of the acoustic scattering problem from penetrable prolate spheroidal structures using the Vekua transformation and arbitrary precision arithmetic", *Mathematical Methods in the Applied Sciences*, **41**:13. ↑82,83,84

[19] R. Barrio, A. Dena, W. Tucker. "A database of rigorous and high-precision periodic orbits of the Lorenz model", *Computer Physics Communications*, **194** (2015), pp. 76–83. ↑82,83,84

[20] G. Khanna. "High-precision numerical simulations on a CUDA GPU: Kerr black hole tails", *Journal of Scientific Computing*, **56**:2 (2013), pp. 366–380. ↑82,83,84

[21] L. Yang, D. Ma, A. Ebrahim, C. J. Lloyd, M. A. Saunders, B. O. Palsson. "solveME: fast and reliable solution of nonlinear ME models", *BMC bioinformatics*, **17**:1 (2016), 391. ↑82,83,84

[22] M. Fasi, N. J. Higham. "Multiprecision algorithms for computing the matrix logarithm", *SIAM Journal on Matrix Analysis and Applications*, **39**:1 (2018), pp. 472–491. ↑82,83,84

[23] R. Iakymchuk, D. Defour, S. Collange, S. Graillat. "Reproducible and accurate matrix multiplication", SCAN 2015: Scientific Computing, Computer Arithmetic, and Validated Numerics, Lecture Notes in Computer Science, vol. **9553**, Springer, 2015, pp. 126–137. ↑82

[24] M. Cornea. "Precision, accuracy, rounding, and error propagation in exascale computing", 2013 IEEE 21st Symposium on Computer Arithmetic (7–10 April 2013, Austin, TX, USA), pp. 231–234. ↑82

[25] K. S. Isupov, V. S. Knyaz'kov. "Parallel multiple-precision arithmetic based on residue number system", *Program Systems: Theory and Applications*, **7**:1 (2016), pp. 61–97 (in Russian). ↑82

[26] K. Isupov, V. Knyazkov. "A modular-positional computation technique for multiple-precision floating-point arithmetic", PaCT 2015: Parallel Computing Technologies, Lecture Notes in Computer Science, vol. **9251**, Springer, 2015, pp. 47–61. ↑82,92,101

[27] N. Nakasato, H. Daisaka, T. Fukushige, A. Kawai, J. Makino, T. Ishikawa, F. Yuasa. "GRAPE-MPs: Implementation of an SIMD for quadruple/hexuple/octuple-precision arithmetic operation on a structured ASIC and an FPGA", 2012 IEEE 6th International Symposium on Embedded Multicore SoCs (20–22 Sept. 2012, Aizu-Wakamatsu, Japan), 2012, pp. 75–83. ↑83

[28] H. Daisaka, N. Nakasato, T. Ishikawa, F. Yuasa. "Application of GRAPE9-MPX for high precision calculation in particle physics and performance results", *Procedia Computer Science*, **51** (2015), pp. 1323–1332. ↑83

[29] E. El-Araby, I. Gonzalez, T. A El-Ghazawi. "Bringing high-performance reconfigurable computing to exact computations", 2007 International Conference on Field Programmable Logic and Applications (27–29 Aug. 2007, Amsterdam, Netherlands), 2007, pp. 79–85. ↑83

[30] Y. Lei, Y. Dou, J. Zhou. "FPGA-specific custom VLIW architecture for arbitrary precision floating-point arithmetic", *IEICE Transactions on Information and Systems*, **E94.D**:11 (2011), pp. 2173–2183. ↑83,110

[31] M. Ishii, J. Detrey, P. Gaudry, A. Inomata, K. Fujikawa. "Fast modular arithmetic on the Kalray MPPA-256 processor for an energy-efficient implementation of ECM", *IEEE Transactions on Computers*, **66**:12 (2017), pp. 2019–2030. ↑83,84,110

[32] M. J. Schulte, E. E. Swartzlander. "A family of variable-precision interval arithmetic processors", *IEEE Transactions on Computers*, **49**:5 (2000), pp. 387–397. ↑83

[33] B. Pan, Y. Wang, S. Tian. "A high-precision single shooting method for solving hypersensitive optimal control problems", *Mathematical Problems in Engineering*, **2018** (2018), 7908378, 11 pp. ↑83,84

[34] I. V. Grossu, C. Besliu, D. Felea, A. Jipa. "High precision framework for chaos many-body engine", *Computer Physics Communications*, **185**:4 (2014), pp. 1339–1342. ↑83,84

[35] V. Nehra, R. Sehgal. "Symbolic computation of mathematical transforms and its application: A MATLAB computational project-based approach", *IUP Journal of Electrical and Electronics Engineering*, **8**:1 (2015), pp. 53–76. ↑84

[36] M. A. Agwa, A. P. Da Costa. "Using symbolic computation in the characterization of frictional instabilities involving orthotropic materials", *International Journal of Applied Mathematics and Computer Science*, **25**:2 (2015), pp. 259–267. ↑84

[37] E. Dovlo, N. Baddour. "Building a symbolic computer algebra toolbox to compute 2D Fourier transforms in polar coordinates", *MethodsX*, **2** (2015), pp. 192–197. ↑84

[38] J. G. Liu, Z. F. Zeng. "Extended generalized hyperbolic-function method and new exact solutions of the generalized hamiltonian and NNV equations by the symbolic computation", *Fundamenta Informaticae*, **132**:4 (2014), pp. 501–517. ↑84

[39] S. Asif, Y. Kong. "Highly parallel modular multiplier for elliptic curve cryptography in residue number system", *Circuits, Systems, and Signal Processing*, **36**:3 (2017), pp. 1027–1051. ↑84

[40] Y. Li, J. Wang, X. Zeng, X. Ye. "Fast Montgomery modular multiplication and squaring on embedded processors", *IEICE Transactions on Communications*, **100**:5 (2017), pp. 680–690. ↑84

[41] J.-C. Bajard, L. Imbert. "A full RNS implementation of RSA", *IEEE Transactions on Computers*, **53**:6 (2004), pp. 769–774. ↑85,89

[42] S. Antao, J. C. Bajard, L. Sousa. "RNS-based elliptic curve point multiplication for massive parallel architectures", *The Computer Journal*, **55**:5 (2011), pp. 629–647. ↑85,89

[43] O. Harrison, J. Waldron. "Efficient acceleration of asymmetric cryptography on graphics hardware", AFRICACRYPT 2009: Progress in Cryptology – AFRICACRYPT 2009, Lecture Notes in Computer Science, vol. **5580**, Springer, 2009, pp. 350–367. ↑85,89

[44] K. Bigou, A. Tisserand. "Single base modular multiplication for efficient hardware RNS implementations of ECC", CHES 2015: Cryptographic Hardware and Embedded Systems – CHES 2015, Lecture Notes in Computer Science, vol. **9293**, Springer, 2015, pp. 123–140. ↑85,89

[45] S. Asif, M. S. Hossain, Y. Kong, W. Abdul. "A fully RNS based ECC processor", *Integration*, **61** (2018), pp. 138–149. ↑85

[46] N. N. Nepeyvoda. "Using overflows to control parallel and distributed computations", *Program Systems: Theory and Applications*, **8**:3 (2017), pp. 87–107 (in Russian). ↑85

[47] N. I. Chervyakov, P. A. Lyakhov, M. G. Babenko, I. N. Lavrinenko, A. V. Lavrinenko, A. S. Nazarov. "The architecture of a fault-tolerant modular neurocomputer based on modular number projections", *Neurocomputing*, **272** (2018), pp. 96–107. ↑85

[48] M. G. Babenko, A. N. Chernykh, N. I. Chervyakov, V. A. Kuchukov, V. Miranda-Lopes, R. Rivera-Rodriges, Chzh. Du. "Efficient number comparison in the residue number system based on positional characteristics", *Proceedings of ISP RAS*, **31**:2 (2019), pp. 187–202 (in Russian). ↑85

[49] D. V. Telpukhov, R. A. Solovyev, V.M. Amerbaev, E.S. Balaka. "Hardware implementation of FIR filter based on number-theoretic fast Fourier transform in residue number system", *Problemy razrabotki perspektivnykh mikro-i nanoelektronnykh sistem (M'YeS)*, 2015, no.4, pp. 42–42 (in English). ↑85

[50] N. Revol. "Introduction to the IEEE 1788-2015 standard for interval arithmetic", NSV 2017: Numerical Software Verification, Lecture Notes in Computer Science, vol. **10381**, Springer, 2017, pp. 14–21. ↑86,91

[51] F. Johansson. "Arb: efficient arbitrary-precision midpoint-radius interval arithmetic", *IEEE Transactions on Computers*, **66**:8 (2017), pp. 1281–1292. ↑86,91

[52] *1788-2015 IEEE Standard for Interval Arithmetic*, 2015. ↑86

[53] N. Revol, Ph. Theveny. "Numerical reproducibility and parallel computations: Issues for interval algorithms", *IEEE Transactions on Computers*, **63**:8 (2014), pp. 1915–1924. ↑86

[54] M. G. Arnold, J. Garcia, M. J. Schulte. "The interval logarithmic number system", *Proceedings 2003 16th IEEE Symposium on Computer Arithmetic* (15–18 June 2003, Santiago de Compostela, Spain), pp. 253–261. ↑86

[55] U. Lotrič, P. Bulić. "Logarithmic arithmetic for low-power adaptive control systems", *Circuits, Systems, and Signal Processing*, **36**:9 (2017), pp. 3564–3584. ↑87

[56] U. Lotrič, P. Bulić. "Applicability of approximate multipliers in hardware neural networks", *Neurocomputing*, **96** (2012), pp. 57–65. ↑87

[57] M. S. Kim, A. A Del Barrio, R. Hermida, N. Bagherzadeh. "Low-power implementation of Mitchell's approximate logarithmic multiplication for convolutional neural networks", 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC) (22–25 Jan. 2018, Jeju, South Korea), pp. 617–622. ↑87

[58] H. Kim, B.-G. Nam, J.-H. Sohn, J.-H. Woo, H.-J. Yoo. "A 231-MHz, 2.18-mW 32-bit logarithmic arithmetic unit for fixed-point 3-D graphics system", *IEEE Journal of Solid-State Circuits*, **41**:11 (2006), pp. 2373–2381. ↑87

[59] A. Avramović, Z. Babić, D. Raič, D. Strle, P. Bulić. "An approximate logarithmic squaring circuit with error compensation for DSP applications", *Microelectronics Journal*, **45**:3 (2014), pp. 263–271. ↑87

[60] M. Gautschi, M. Schaffner, F. K. Gürkaynak, L. Benini. "An extended shared logarithmic unit for nonlinear function kernel acceleration in a 65-nm CMOS multicore cluster", *IEEE Journal of Solid-State Circuits*, **52**:1 (2017), pp. 98–112. ↑87

[61] D. Nandan, J. Kanungo, A. Mahajan. "An error-efficient gaussian filter for image processing by using the expanded operand decomposition logarithm multiplication", *Journal of Ambient Intelligence and Humanized Computing*, 2018, pp. 1–8. ↑87

[62] J. Coleman, R C. Ismail. "LNS with co-transformation competes with floating-point", *IEEE Transactions on Computers*, **65**:1 (2016), pp. 136–146. ↑87

[63] J. Le Maire, N. Brunie, F. De Dinechin, J. M. Muller. “Computing floating-point logarithms with fixed-point operations”, 2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH) (10–13 July 2016, Santa Clara, CA, USA), pp. 156–163. ↑87

[64] H. Fu, O. Mencer, W. Luk. “Comparing floating-point and logarithmic number representations for reconfigurable acceleration”, 2006 IEEE International Conference on Field Programmable Technology (13–15 Dec. 2006, Bangkok, Thailand), pp. 337–340. ↑87

[65] M. Chugh, B. Parhami. “Logarithmic arithmetic as an alternative to floating-point: A review”, 2013 Asilomar Conference on Signals, Systems and Computers (3–6 Nov. 2013, Pacific Grove, CA, USA), 2013, pp. 1139–1143. ↑87

[66] M. Haselman, M. Beauchamp, A. Wood, S. Hauck, K. Underwood, K. S. Hemmert. “A comparison of floating point and logarithmic number systems for FPGAs”, 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM’05) (18–20 April 2005, Napa, CA, USA, USA), pp. 181–190. ↑87

[67] R. C. Ismail, J. N. Coleman, N. Norzahiyah, Z. Sauli. “A comparative analysis between logarithmic number system and floating-point ALU”, *Advances in Environmental Biology*, **7**:12 (2013), pp. 3601–3606. ↑87

[68] M. G. Arnold. “Iterative methods for logarithmic subtraction”, *Proceedings IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, ASAP 2003 (24–26 June 2003, The Hague, Netherlands), pp. 315–325. ↑88

[69] A. R. Omondi, B. Premkumar, *Residue Number Systems: Theory and Implementation*, Advances in Computer Science and Engineering Texts, World Scientific, 2007, ISBN 978-1860948664, 296 pp. ↑88

[70] N. S. Szabo, R. I. Tanaka, *Residue Arithmetic and its Applications to Computer Technology*, McGraw-Hill Series in Information Processing and Computers, McGraw-Hill, 1967, 236 pp. ↑88,100,103

[71] I. Ya. Akushskiy, D. I. Yuditskiy. *Machine Arithmetic in Residual Classes*, Sov. radio, M., 1968 (in Russian), 440 pp. ↑88

[72] K. Bigou, A. Tisserand. “RNS modular multiplication through reduced base extensions”, 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors (18–20 June 2014, Zurich, Switzerland), pp. 57–62. ↑89

[73] J. C. Bajard, J. Eynard, N. Merkiche. “Montgomery reduction within the context of residue number system arithmetic”, *Journal of Cryptographic Engineering*, **8**:3 (2018), pp. 189–200. ↑89

[74] M. Langhammer, B. Pasca. “Single precision natural logarithm architecture for hard floating-point and DSP-enabled FPGAs”, 2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH) (10–13 July 2016, Santa Clara, CA, USA), pp. 164–171. ↑91

[75] *754-2008 – IEEE Standard for Floating-Point Arithmetic*, IEEE, 2008. ↑92

[76] M. Cornea, J. Harrison, P. T. P. Tang. *Scientific Computing on Itanium-Based Systems*, Intel Press, Hillsboro, 2002, ISBN 978-0971288775, 280 pp. ↑94,97

[77] M. Czyzak, R. Smyk, Z. Ulman. “Pipelined scaling of signed residue numbers with the mixed-radix conversion in the programmable gate array”, *Poznan University of Technology Academic Journals. Electrical Engineering*, 2013, no.76, pp. 89–99. ↑100,104

[78] A. P. Shenoy, R. Kumaresan. "Fast base extension using a redundant modulus in RNS", *IEEE Transactions on Computers*, **38**:2 (1989), pp. 292–297. ↑100,103,104

[79] S. Kawamura, M. Koike, F. Sano, A. Shimbo. "Cox-Rower architecture for fast parallel Montgomery multiplication", EUROCRYPT 2000: Advances in Cryptology – EUROCRYPT 2000, Lecture Notes in Computer Science, vol. **1807**, Springer, 2000, pp. 523–538. ↑100

[80] K. C. Posch, R. Posch. "Base extension using a convolution sum in residue number systems", *Computing*, **50**:2 (1993), pp. 93–104. ↑100

[81] A. S. Korzhavina, V. S. Knyaz'kov. "Base extension in residue number systems: a review and cost analysis", *Modern high technologies*, 2017, no.12, pp. 37–42 (in Russian). ↑100

[82] A. S. Korzhavina, V. S. Knyaz'kov. "Base extension in residue number systems using mixed-radix systems", *Scientific and Technical Volga region Bulletin*, 2017, no.6, pp. 204–207 (in Russian). ↑100,105

[83] G. A. Jullien. "Residue number scaling and other operations using ROM arrays", *IEEE Transactions on Computers*, **C-27**:4 (1978), pp. 325–336. ↑103,104

[84] Y. Kong, B. Phillips. "Fast scaling in the residue number system", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **17**:3 (2009), pp. 443–447. ↑103,104

[85] S. Ma, J. Hu, Y. Ye, L. Zhang, X. Ling. "A 2^n scaling scheme for signed RNS integers and its VLSI implementation", *Science in China Series F: Information Sciences*, **53**:1 (2010), pp. 203–212. ↑103,104

[86] C. H. Chang, J. Y. S. Low. "Simple, fast, and exact RNS scaler for the three-moduli set three-moduli set $\{2^n-1, 2^n, 2^n+1\}$", *IEEE Transactions on Circuits and Systems I: Regular Papers*, **58**:11 (2011), pp. 2686–2697. ↑103,104

[87] A. Hiasat. "Efficient RNS scalers for the extended three-moduli set $\{2^n-1, 2^{n+p}, 2^n+1\}$", *IEEE Transactions on Computers*, **66**:7 (2017), pp. 1253–1260. ↑103,104

[88] A. S. Korzhavina, V. S. Knyaz'kov. *Method of organizing implementation of multiplication of two numbers in modular logarithmic format of representation with floating point on hybrid multi-nuclear processors*, Patent No 2666285, 2018 (in Russian). ↑110

Sample citation of this publication:

Anastasia S. Korzhavina, Vladimir S. Knyazkov. "High-precision computations using residue-interval arithmetic on FPGAs". *Program Systems: Theory and Applications*, 2019, **10**:3(42), pp. 81–127. (*In Russian*).

10.25209/2079-3316-2019-10-3-81-127

http://psta.psiras.ru/read/psta2019_3_81-127.pdf