ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Кучуков Виктор Андреевич

РАЗРАБОТКА МЕТОДОВ И ПРОГРАММНЫХ СРЕДСТВ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ОТКАЗОУСТОЙЧИВЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, РАБОТАЮЩИХ В МОДУЛЯРНОМ КОДЕ

Специальность 2.3.5 —
«Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»
(технические науки)

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
Доктор физико-математических наук, доцент
Бабенко Михаил Григорьевич

Ставрополь — 2024

Оглавление

Введение

Актуальность темы исследования. Бурное развитие мобильных вычислительных систем и облачных технологий, распределенных вычислений, их повсеместное внедрение, включая «умные» вещи, смартфоны, нательные датчики, приобретает глобальных характер, что наряду с безусловными преимуществами приводит к возникновению новых угроз безопасности и конфиденциальности критических данных.

Особую актуальность приобретают вычислительные системы, позволяющие автономно производить вычисления в удаленных местностях в условиях низкого качества связи, что характерно для систем контроля на железных дорогах и нефтегазопроводах. Автоматизированное решение задач для подобных объектов, например, обнаружение несанкционированного проникновения, поломки, утечки, должно производиться с достаточной скоростью и обладать требуемой отказоустойчивостью. Однако обеспечение высокопроизводительных и надежных хранения и обработки информации такими вычислительными системами затруднено в связи с ограниченностью их вычислительной мощности и автономностью. Поэтому важной задачей является поиск путей оптимизации архитектуры вычислительных систем по различным параметрам, таким как безопасность, скорость работы, отказоустойчивость и т.д. Главенствующая роль при решении обозначенных проблем отводится разработке вычислительных узлов распределенной среды обработки информации, для которых большое значение имеют сложность и надежность аппаратуры, время работы и потребляемая мощность, что во многом зависит от простоты программного обеспечения реализации вычислительных алгоритмов. Критическое значение при проектировании подобных вычислительных узлов распределенной среды приобретает выбор математического аппарата, вплоть до способа представления чисел и выполнения арифметических операций.

Трафик, генерируемый устройствами, растет, хранение и обработка информации с требуемыми для конкретных задач временем выполнения и отказоустойчивостью оказываются затруднены за счет того, что скорость и отказоустойчивость обработки информации существующими методами практически достигли предельных возможностей и требуют значительных схемных затрат. Поэтому необходима разработка методов и алгоритмов обработки с большей скоростью

и отказоустойчивостью вычислений, а также низкой аппаратурной избыточностью.

В качестве теоретической основы для проектирования вычислительных систем обработки данных, обладающих возможностью увеличения скорости вычислений за счет параллельного выполнения операций сложения и умножения, в цифровой обработке сигналов широко используют систему остаточных классов (СОК). При вычислениях в СОК используется не всё число, а информация об его остатках от деления на определенные модули, что позволяет проводить вычисления по независимым каналам с числами небольшой разрядности [55]. За счет этого достигается увеличение скорости вычислений [96]. При введении дополнительных модулей, т.е. избыточности данных, СОК приобретает корректирующие свойства и позволяет обнаруживать и исправлять ошибки, возникающие в процессе обработки информации.

Разработка специализированных вычислительных систем, работающих в СОК, началась в 50–60-е годы в СССР [107], сейчас же различное применение система остаточных классов нашла в разработках Cisco Technology (патент US 7027598), Toshiba, Samsung Electronics (патент US 7805478) и Google Inc. (патент US 8386802).

Эффективность использования СОК в цифровой обработке информации обусловлена выполнением большого количества операций сложения, вычитания и умножения. Однако в СОК существует ряд операций, называемых немодульными, таких как определение знака числа, сравнение чисел, деление, масштабирование, определение переполнения диапазона и др., реализация которых является вычислительно сложной. Для выполнения данных операций требуется знание позиции числа на числовой прямой. Применение существующих методов выполнения немодульных операций в СОК сопряжено с большими вычислительными затратами и не обеспечивает приемлемой скорости обработки. Поэтому тщательного исследования требует вопрос выбора оптимальных параметров немодульных операций и выработки общего метода, позволяющего увеличить скорость и отказоустойчивость выполнения немодульных операции для задач обработки информации.

Значительный научный вклад в рассматриваемую область внесли отечественные и зарубежные исследователи: И.Я. Акушский, Д.И. Юдицкий, В.М. Амербаев, А.И. Галушкин, Н.И. Червяков, А.А. Коляда, A. Omondi и другие.

Целью диссертационного исследования является повышение скорости работы отказоустойчивых вычислительных узлов обработки информации в распределенных средах за счет оптимизации вычислительно сложных процедур модулярного кода и разработки программных средств их проектирования.

Объектом диссертационного исследования является теория обеспечения надежности данных.

Предмет исследования — методы и алгоритмы распределенной обработки данных с использованием модулярной арифметики.

Научная задача диссертационной работы состоит в исследовании и разработке математических методов и алгоритмов выполнения немодульных операций на основе различных форм позиционной характеристики числа, способных повысить скорость и отказоустойчивость обработки информации вычислительными узлами распределенной среды, работающими в модулярном коде.

Для решения поставленной научной задачи была произведена её декомпозиция на следующие частные **задачи**:

1. Разработка и модификация методов и алгоритмов выполнения вычислительно сложных операций в СОК, таких как перевод из позиционной системы счисления (ПСС) в СОК и из СОК в ПСС, определение знака числа и сравнение чисел для выполнения обработки данных.
2. Модификация методов для повышения отказоустойчивости обработки данных вычислительными узлами распределенной среды, работающими в модулярном коде.
3. Разработка программного комплекса выполнения немодульных операций в модулярном коде для проектирования отказоустойчивых вычислительных узлов распределенной среды обработки данных.

Соответствие паспорту научной специальности. Область исследования соответствует паспорту специальности 2.3.5 – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей по следующим пунктам:

8. Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования.

9. Модели, методы, алгоритмы, облачные технологии и программная инфраструктура организации глобально распределенной обработки данных.

Научная новизна:

1. Модифицированы методы и алгоритмы перевода из позиционной системы счисления в СОК и из СОК в позиционную систему счисления, определения знака и сравнения чисел в СОК, отличающиеся от известных меньшей размерностью операндов и эффективной реализацией операций без необходимости нахождения остатка по большому модулю.
2. Модифицированы методы коррекции ошибок распределенной обработки и хранения информации в системе остаточных классов.
3. Разработан программный комплекс для выполнения немодульных операций вычислительными узлами распределенной среды, позволяющий повысить скорость и отказоустойчивость решения задач распределенной обработки данных.

Моделирование и вычислительный эксперимент проведены на ASIC в среде RTL и физического синтеза Cadence Genus Synthesis Solution с использованием библиотеки osu018_stdcells с использованием языков высокого уровня Java и Python для генерации модулей на языке Verilog и исследования свойств разработанных алгоритмов. В качестве критериев, позволяющих оценить эффективность разработанных методов и алгоритмов взяты время прохождения сигнала по схеме (пикосекунды, пс) и используемая площадь (квадратные микрометры, мкм^2).

Практическая значимость разработанных методов и алгоритмов заключается в возможности реализации на их основе программных средств реализации вычислительных узлов распределенной среды, характеризующихся высокой скоростью и отказоустойчивостью обработки данных, достигаемых за счет модификации алгоритмов модулярной арифметики. Полученные результаты могут быть использованы в специализированных высокопроизводительных цифровых системах обработки информации, таких как распределенные и облачные вычислительные системы, системы автоматизированного контроля, функционирующие в непозиционной системе счисления. Разработанные в рамках диссертационного исследования методы обнаружения и локализации ошибок распределенной обработки и хранения данных и методы перевода из системы остаточных классов в позиционную систему счисления внедрены в организации ООО «Инфоком-С» в системе интеллектуального реагирования на инциденты и события «Darvis».

Методология и методы исследования включают использование математического аппарата линейной алгебры, теории чисел, математического анализа, теории алгоритмов, теории надежности, численных методов, математического моделирования, теории вероятностей и математической статистики.

Основные положения, выносимые на защиту:

1. Метод перевода из СОК в позиционную систему счисления на основе модифицированной обобщенной позиционной системы счисления (ОПСС).
2. Приближенный метод определения знака и сравнения чисел в СОК на основе Китайской теоремы об остатках (КТО).
3. Алгоритм сравнения чисел и определения знака числа на основе функции ядра Акушского без критических ядер.
4. Метод обнаружения и локализации ошибок, основанный на использовании несбалансированной системы остаточных классов.

Достоверность полученных результатов обеспечивается строгостью проведения математических доказательств, при получении которых был использован научно-методический аппарат математического анализа, теории чисел и численных методов, и подтверждается проведенным сравнительным анализом разработанных методов и алгоритмов с известными ранее с точки зрения скорости обработки данных.

Личный вклад автора. Все изложенные в диссертационной работе результаты получены при непосредственном участии автора. Из результатов работ, выполненных коллективно, в диссертацию включены только полученные непосредственно автором. В работах [61; 103; 106; 117; 123] автором рассмотрены проблемы вычисления позиционных характеристик числа и предложена модификация метода вычисления позиционной характеристики на основе обобщенной позиционной системы счисления. В работах [68; 144] автором предложено уточнение разрядности коэффициентов приближенного метода на основе КТО для сравнения чисел. В работах [10; 102] автором предложен алгоритм для построения монотонной функции ядра Акушского без критических ядер для сравнения чисел. В работах [42; 43; 46; 108; 116; 120] автором рассмотрены особенности распределенного хранения и обработки данных с возможностью обнаружения и коррекции ошибок, предложен приближенный метод коррекции одиночной ошибки в СОК с одним надежным контрольным модулем. В работах [102; 114; 116–120] приведен разработанный автором комплекс вычислитель-

ных узлов распределенной среды для выполнения модифицированных методов и алгоритмов вычисления немодульных операций в СОК, полученный с использованием программного комплекса для выполнения немодульных операций в СОК, на который получены свидетельства о государственной регистрации программ для ЭВМ [124–137].

Апробация работы. Основные результаты диссертационного исследования докладывались на международных конференциях IEEE, среди которых «2016 IEEE CONFERENCE ON QUALITY MANAGEMENT, TRANSPORT AND INFORMATION SECURITY, INFORMATION TECHNOLOGIES (IT&MQ&IS-2016)» (г. Нальчик, Россия), «2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus-2017)» (г. Санкт-Петербург, Россия), «2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus-2018)» (г. Санкт-Петербург, Россия), «ОТКРЫТАЯ КОНФЕРЕНЦИЯ ИСП РАН ИМ. В.П. ИВАННИКОВА (ISPRAS 2019)» (г. Москва, Россия), «2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus-2020)» (г. Санкт-Петербург, Россия), «CPAMCS-2023: Current Problems in Applied Mathematics and Computer Science» (г. Ставрополь, Россия).

Публикации. Основные результаты по теме диссертационного исследования изложены в 31 публикации [1; 5; 7; 10; 19; 20; 23; 25; 34; 41–43; 46; 61; 68; 89; 98; 102; 103; 105; 106; 108; 112; 114; 116–120; 123; 144], 8 из которых изданы в журналах, рекомендованных ВАК [98; 103; 105; 106; 108; 112; 123; 144], 9 — в тезисах докладов конференции [1; 19; 20; 23; 25; 34; 41; 43; 89], 16 — в публикациях, входящих в международные базы цитирования Web of Science и Scopus [1; 5; 7; 10; 19; 20; 23; 25; 34; 41–43; 46; 61; 68; 89]. Получено 7 патентов на изобретения [102; 114; 116–120], из них один международный, 14 свидетельств о государственной регистрации программ для ЭВМ [124–137].

Внедрение. Результаты диссертационной работы были использованы при выполнении проектов: гранта РНФ № 19-71-10033 «Эффективная, безопасная и отказоустойчивая система распределенного хранения и обработки конфиденциальных данных с регулируемой избыточностью для проектирования мобильных облаков на маломощных вычислительных устройствах» (глава 2, параграф 2.1), гранта РФФИ № 20-37-70023 «Разработка методов и алгоритмов быстродействующего, отказоустойчивого математического сопроцессора для проектирования вычислительных систем с повышенным уровнем безопасности и низким энерго-

потреблением» (глава 2, параграфы 2.3, 2.4), гранта Министерства науки и высшего образования Российской Федерации «Фундаментальные алгоритмы, технологии глубокого обучения и безопасности для облачного хранения и обработки данных» № 075-15-2021-1010 (глава 2, параграф 2.5), стипендии Президента РФ № СП-2236.2018.5 «Разработка мобильного устройства кодирования и передачи видеопотока с низким энергопотреблением в условиях ограничения вычислительной мощности», стипендии Президента РФ № СП-3186.2022.5 «Исследование и разработка методов повышения быстродействия и отказоустойчивости математического сопроцессора для защищенного хранения и обработки цифровой информации на основе модулярной арифметики» (глава 2, параграф 2.2). Работа выполнена в Северо-Кавказском центре математических исследований в рамках соглашения № 075-02-2024-1451 с Министерством науки и высшего образования Российской Федерации.

Объем и структура работы. Диссертация состоит из введения, трех глав, заключения и пяти приложений.

В первой главе рассмотрены методы повышения отказоустойчивости вычислительных систем, в частности введение избыточности. При этом к арифметичным кодам, у которых информационная и контрольная части кода равноценны, относится система остаточных классов. Введение избыточных модулей в СОК позволяет не только обнаруживать, но и исправлять ошибки, однако в большинстве методы коррекции ошибок обладают высокой алгоритмической сложностью. Таким образом, сформулирована научная задача исследования: разработка математических методов и алгоритмов выполнения немодульных операций на основе различных форм позиционной характеристики чисел, способных повысить скорость и отказоустойчивость обработки информации вычислительным узлами распределенной среды, работающим в модулярном коде.

Во второй главе рассмотрены реализации немодульных операций в СОК. Рассмотрена проблема выбора модулей СОК для перевода чисел из позиционной системы счисления. Рассмотрены методы нахождения остатка от деления на модули специального вида 2^n, 2^n-1, 2^n+1. Предложена модификация на основе периода и полупериода числа, позволяющая найти наименьший неотрицательный вычет.

Исследованы методы перевода из СОК в позиционную систему счисления на основе Китайской теоремы об остатках, приближенного метода на основе КТО, обобщенной позиционной системы счисления, функции ядра и диагональ-

ной функции. Предложен новый метод на основе обобщенной позиционной системы счисления для перевода чисел в позиционную систему счисления и расширения оснований.

Рассмотрена проблема сравнения чисел в СОК и определения знака числа. Получена оценка необходимой точности при округлении констант строго возрастающей приближенной функции на основе КТО для сравнения чисел, предложен метод сравнения чисел, который позволяет уменьшить размер операндов. Также получена модификация функции ядра Акушского с заданными свойствами, разработаны алгоритмы выбора параметров для функции ядра без критических ядер, на основе которой построен алгоритм сравнения чисел и определения знака числа. Предложены методы сравнения чисел в СОК с чётным и нечётным динамическим диапазоном, которые позволяют получить искомый результат без операции деления на большой модуль.

Исследованы следующие методы модульного умножения: нейронная сеть конечного кольца, метод Карацубы-Офмана, метод Бута, алгоритмы Монтгомери. Также рассмотрена реализация умножения с накоплением (MAC) в системе остаточных классов.

Для коррекции одиночной ошибки рассмотрен метод проекций на основе Китайской теоремы об остатках с дробными значениями. Введен метод коррекции одиночной ошибки с одним избыточным модулем СОК, приведена приближенная реализация данного метода.

Таким образом, реализованы все операции, необходимые для построения отказоустойчивой вычислительной системы обработки данных, работающей в системе остаточных классов.

В третьей главе рассмотрена реализация разработанных во второй главе алгоритмов в виде архитектур вычислительных узлов распределенной среды для выполнения арифметических, в том числе немодульных, операций в СОК.

Рассмотрен алгоритм проектирования реализации отказоустойчивой вычислительной системы, работающей в модулярном коде. Построение такой системы включает выбор набора модулей СОК, выбор метода перевода из ПСС в СОК, методов выполнения сложения, вычитания и умножения в модулярном коде, выбор метода вычисления позиционной характеристики для задач сравнения чисел, определения знака числа, перевода из СОК в ПСС, обнаружения и локализации ошибок.

Для реализации алгоритма проектирования отказоустойчивой вычислительной системы на основе рассмотренных во второй главе методов и алгоритмов разработан комплекс программ для работы с модулярным кодом.

Рассмотрено моделирование перевода числа из позиционной системы счисления в систему остаточных классов. Классические методы на основе нейронной сети конечного кольца, периода и полупериода дают число в диапазоне удвоенного модуля. Предложенная модификация позволила получить искомые значения, при этом модифицированный метод на основе периода и полупериода числа позволили в среднем на 50% сократить необходимую площадь, на 23% — время вычислений, по сравнению с нейронной сетью конечного кольца.

Рассмотрено моделирование обратного преобразования из системы остаточных классов в позиционную систему счисления. В среднем предложенный подход на основе модифицированной обобщенной позиционной системе счисления на 23% быстрее и на 30% компактнее, чем метод КТО. При этом на диапазоне 32-48 бит предложенный подход быстрее на 46% и компактнее на 50% по сравнению с КТО. При сравнении с приближенным методом на основе КТО предложенный метод дает преимущество в среднем на 18% по времени и на 23% по площади только на диапазоне 32-48 бит. В среднем преимущество по сравнению с ОПСС по времени составило 15%, но метод на основе ОПСС занимает на 16% меньшую площадь. Таким образом, предложенный метод на основе модифицированной ОПСС позволяет повысить скорость вычислений и снизить площадь, являясь компромиссным решением между последовательной и компактной ОПСС, и параллельной, но громоздкой КТО.

Рассмотрены методы вычисления позиционной характеристики для сравнения чисел и определения знака числа в системе остаточных классов. Предложен ряд архитектур вычислительных узлов распределенной среды для сравнения и определения знака, на которые получены патенты на изобретения.

Предложенная архитектура вычислительного узла распределенной среды на основе функции ядра Акушского с заданными свойствами быстрее реализации на основе Китайской теоремы об остатках в среднем на 59%, на 15% быстрее приближенного метода на основе КТО, и на 22% быстрее ОПСС. При этом требуемая площадь на 61% меньше, чем у КТО, на 16% – чем у приближенного метода на основе КТО, но в среднем на 80% больше, чем у ОПСС.

Также уточнена точность приближенного метода на основе КТО для сравнения чисел. Архитектура вычислительного узла уточненного приближенного

метода на основе КТО в среднем на 58% быстрее КТО, на 10% быстрее известной оценки приближенной КТО, на 14% быстрее ОПСС. При этом уточненный приближенный метод на основе КТО требует в среднем на 59% меньшую площадь по сравнению с КТО, на 9% меньше известной оценки приближенного метода на основе КТО, но на 100% больше чем у ОПСС.

Таким образом, разработанные методы сравнения чисел и определения знака показали лучшее время работы, однако по используемой площади уступают обобщенной позиционной системе счисления.

Приведено моделирование умножения с накоплением для двоичной системы счисления и системы остаточных классов, в которых умножение реализовано стандартными методами Verilog. Реализация в СОК в среднем на 30% быстрее при приблизительно одинаковых занимаемых площадях.

Предложена архитектура вычислительного узла для обнаружения и коррекции ошибки модулярного кода, основанная на приближенном методе на основе Китайской теореме об остатках, на которую получен патент. Особенностью данного подхода является использование памяти для восстановления чисел по каждой проекции. Также рассмотрено моделирование алгоритма коррекции одиночной ошибки с одним избыточным модулем. Метод на основе Китайской теоремы об остатках, в среднем имеет на 52,15% меньшее время вычисления по сравнению с методом с одним избыточным основанием, но имеет на 140% большую площадь. Адаптация приближенного метода с одним избыточным основанием имеет в среднем на 0,43% большее время вычислений по сравнению с приближенным методом на основе КТО с двумя избыточными основаниями, но на 16,96% меньшую площадь.

Полный объём диссертации составляет 215 страниц, включая 30 рисунков и 33 таблицы. Список литературы содержит 145 наименований.

Глава 1. Исследование методов повышения отказоустойчивости вычислительных систем

1.1 Методы повышения отказоустойчивости вычислений

С момента появления первых вычислительных машин актуальной является проблема их надежной работы и гарантированной достоверности получаемых результатов. Усложнение вычислительных структур, условий их эксплуатации и решаемых задач требует разработки новых подходов к обеспечению надежности и достоверности вычислений. Работа в широком диапазоне температур, в условиях высокой солнечной и космической радиации увеличивает вероятность возникновения отказов в десятки и сотни раз по сравнению с работой в лабораторных условиях [101].

Во многих областях промышленности отказ функционирования вычислителя может привести к крупным экономическим последствиям, а в ряде случаев и к техногенным катастрофам. Так, отказ элементов при запуске ракеты Ariane 5 в 1996 году и ракеты Electron компании Rocket Lab привели к убыткам на сумму более 7 млн. долларов. Отказ вычислителей, управляющих производственным процессом в химической или ядерной промышленности может привести не только к огромных финансовым убыткам, но и к гибели людей.

Увеличение объема решаемых вычислительными системами задач вызывает рост их сложности, и следовательно, понижение надежности, что также приводит к учащению случаев искажения обрабатываемой информации.

Сложные структуры, каковыми являются подавляющее большинство современных вычислительных систем, содержат большое число элементов и функциональных связей между ними, дублирующих и вспомогательных устройств, и выход из строя некоторых элементов не обязательно приводит к полному отказу вычислительной системы, т.е. прекращению выполнения ею заданных функций, а лишь может в некоторой мере ухудшить качество функционирования. В связи с этим в теории надежности существует понятие функциональной надежности P_{Φ}, вероятности того, что данная вычислительная структура будет удовлетворительно выполнять свои функции в течение заданного времени. Данное понятие связано с понятием отказоустойчивости, под которой понимается свой-

ство образца в целом и (или) его функциональных систем и составных частей, характеризующее способность обеспечивать завершение цикла применения по назначению в ожидаемых условиях эксплуатации при возможных отказах и повреждениях без неприемлемого вреда лицам или имуществу, за исключением вреда, предусмотренного целевым назначением образца [145], т.е. возможность системы сохранять свою работоспособность после отказа одной или нескольких её составных частей.

При оценке функциональной надежности многопроцессорной или распределенной структуры обычно прежде всего выделяют основной комплекс вычислительных узлов, любая неисправность которых приводит к отказу всего вычислителя в целом. В мультипроцессорных вычислителях к таким устройствам относятся вычислительный блок, его запоминающие устройства и устройство управления.

Также большое внимание уделяется программному обеспечению вычислителей. Математическое (программное) обеспечение играет одну из основных ролей в процессе функционирования вычислительных систем и ошибки в них приводят к полному либо частичному выходу из строя вычислительных узлов. Поэтому обеспечение высокой надежности программного обеспечения является одной из важнейших задач теории надежности.

Увеличение уровня помех как внутреннего, обусловленного микроминиатюризацией современной базы вычислительных машин, так и внешнего (влияние электромагнитных излучений от промышленных объектов, всплески токов централизованного питания и др.) происхождений приводит к понижению готовности системы к её применению [121]. Коэффициент готовности $K_{\Г}$ определяется выражением

$$K_{\Г} = \frac{T_{\О}}{T_{\О} + T_{\В}}. \tag{1.1}$$

Так как среднее время наработки на отказ $T_{\О}$ содержится и в числителе и знаменателе выражения (1.1), а среднее время восстановления $T_{\В}$ — только в знаменателе, то для повышения коэффициента готовности более целесообразно сокращать время восстановления аппаратуры, чем увеличивать наработку на отказ. Поэтому актуальным является применение автоматического контроля с целью быстрого обнаружения сбоев. Организация непрерывного контроля может быть решена путем введения избыточности.

Все методы введения избыточности можно свести к двум обобщенным группам:

1. Структурное резервирование;
2. Информационное резервирование.

Наиболее распространенными методами первой группы являются дублирование и мажоритарный принцип, а к основным методам второй группы относятся помехоустойчивые коды, арифметические коды, контроль в системе остаточных классов.

Как известно, одним из необходимых условий успешного использования избыточности является согласование метода введения избыточности с наиболее вероятными ошибками защищаемого устройства [113]. Известно также, что применительно к каналам передачи данных вероятность однократных ошибок значительно превышает вероятность ошибок большей кратности (обобщенный биномиальный закон распределения ошибок), а в цифровых устройствах закон распределения ошибок приближается к равномерному [113; 138]. Более того, при обработке и хранении информации часты случаи перманентных «вспышек» ошибок на всей длине комбинации, что объясняется следующим:

1. Большинство операций в цифровых системах осуществляются параллельно, следовательно, воздействие помех на комбинацию носит одновременный характер.
2. Применение высокоскоростных интегральных микросхем (время переключения достигает 10^{-9} с) влечет возрастание скорости обработки и передачи информации в аппаратуре. В этом случае помеха заданной длительности «захватывает» большее число следующих подряд символов комбинации;
3. При выполнении арифметических операций из-за цепи переносов однократная ошибка также может привести к возникновению перманентной «вспышки» ошибок (эффект размножения ошибок).

Введение контроля в ЭВМ не только повышает её надежность, но и играет чрезвычайно важную роль при организации всех остальных методов повышения надежности. Действительно, если неизвестно, исправно ли работает узел, блок, устройство, то невозможно перейти ни к коррекции результатов, ни к включению резервного блока.

Резервирование является основным инструментом теории надежности. Понятие резервирования является достаточно общим и включает в себя различные

методы повышения отказоустойчивости путем введения структурной избыточности. Под резервированием понимается соединение однотипных элементов, узлов, блоков, его кратность равна числу элементов (узлов, блоков), включенных или подготовленных к включению последовательно или параллельно исходному элементу (узлу, блоку).

Резервирование можно выполнять на уровне отдельных элементов схемы или узлов и блоков. При этом, чем ниже уровень, на котором происходит резервирование, тем выше отказоустойчивость избыточной схемы [101]. Простое резервирование более сложных в функциональном отношении схем связано с трудностями принципиального и технического характера.

Недостатками методов резервирования являются резкое увеличение избыточного оборудования, массы, габаритов и стоимости оборудования; возрастание нагрузки на отдельные компоненты схемы, что приводит к снижению нагрузочной способности схемы, а следовательно, требуется введение дополнительных элементов обеспечения. Эти недостатки относятся ко всем методам резервирования, в частности, к мажоритарным, когда используется так называемый горячий резерв и вводится мажоритарный орган, который реализует функцию большинства, т.е. выдает сигнал логических единицы или нуля в зависимости от того, каких сигналов на его входе больше.

Существенное увеличение оборудования (его дополнительное введение) не решает никакой другой задачи, кроме повышения надежности.

Кроме того, остро стоит вопрос сложности кодирующих и декодирующих устройств, что выдвигает проблему «сторожа сторожей». С другой стороны, кодирующие и декодирующие устройства нарушают принцип однородности вычислительной машины и делают её менее технологичной с точки зрения современной микроэлектроники. Следовательно, важным условием при построении современных вычислительных систем на микроэлектронной базе с высокой степенью интеграции является обеспечение итерационности (однородности) её цифровых структур. Это выдвигает на первый план проблему обеспечения единого метода кодирования, т.е. введения избыточности в вычислительное устройство на этапе его проектирования [121].

Равновероятный характер ошибок, присущий аппаратуре вычислительного устройства, лишает смысла использование методов резервирования. Кроме того, резервирование приводит к ухудшению множества эксплуатационных параметров (высокая стоимость, энергопотребление, большой вес и габариты), а

необходимость работы контрольной аппаратуры одновременно с резервной снижает общую надежность системы.

С учетом особенностей возникновения ошибок в вычислительных устройствах можно сформулировать следующие требования к кодам.

1. Коды должны обнаруживать все однократные ошибки, которые наиболее распространены при передаче по каналам связи, а также при работе дискретной техники в случае малых уровней помех.
2. Обнаруживающая способность сплошных перманентных «вспышек» ошибок на всей длине кодовых комбинаций также должна равняться единице.
3. В остальном диапазоне ошибок желательно наличие равновероятной обнаруживавшей способности устройств контроля.

Таким образом, перспективным является использование избыточного кодирования, которое позволяет обнаруживать и исправлять ошибки без многократного увеличения оборудования, как в случае резервирования аппаратуры.

Выбор подходящего способа кодирования информации, обладающего необходимой корректирующей способностью, позволяет заметно снизить требования к надежности линий передач информации и используемому оборудованию, сделать их более простыми и дешевыми [96].

С целью обеспечения отказоустойчивости можно с помощью одного вычислительного узла дважды выполнить одни и те же вычисления, и если результаты этих вычислений совпадают, то принимается решение о том, что вычисления выполнены без ошибок. Если же результаты вычислений не совпадают, то это означает, что в процессе вычислений произошла ошибка. Вычисления проводятся еще раз, и обнаруженная ошибка устраняется. В этих случаях наличие избыточности позволяет провести обнаружение ошибок и повысить отказоустойчивость вычислительной системы. Однако проблема состоит не в том, чтобы просто повысить надежность за счет введения очень большой избыточности, а в том, как с помощью по возможности меньшей специальным образом вводимой избыточности достичь нужной степени надежности [140].

Основной задачей теории кодирования в настоящее время является повышение надежности систем связи и вычислительных систем с помощью целенаправленного эффективного введения избыточности в процессе представления и преобразования информации. Введение избыточности приводит к снижению количества сообщений, которые могут быть переданы или обработаны за опре-

деленный период времени, а кроме того, предполагает использование в системе дополнительных устройств для целенаправленного введения избыточности (кодеров), устройств для обнаружения и исправления возникающих ошибок (декодеров) и ряда других дополнительных устройств. Рассмотрим ряд методов помехоустойчивого кодирования.

Основная идея помехозащитного кодирования Рида-Соломона заключается в умножении информационного слова, представленного в виде полинома D, на неприводимый многочлен G, известный обеим сторонам, в результате которого получается кодовое слово $C = D \cdot G$, представленное в виде полинома. Декодирование осуществляется по алгоритму обратному приведенному выше: производится деление кодового слова C на полином G. Если в результате деления декодер получает не нулевой остаток, то он дает сигнал об произошедшей ошибке [104].

Если степень полинома G превосходит степень кодового слова на две степени, то может производится не только обнаружение, но и исправление одиночных ошибок. В случае если степень полинома G превосходит на k степень кодового слова, то в этом случае количество исправляемых ошибок равняется $t = \lfloor \frac{k}{2} \rfloor$.

Алгоритм разделения данных (Information Dispersal Algorithm, IDA) [70], первоначально разработанный М. Рабиным для телекоммуникационных систем, позволяет разделить данные таким образом, что в случае потери или недоступности части данных возможно восстановить исходные данные. Кроме того, согласно принципам алгоритмов IDA, для устранения одной ошибки, требуется увеличение объема данных в два раза, для исправления 8 ошибок увеличение объема в 4 раза. Как и RAID, IDA позволяет восстанавливать данные из подмножества исходных данных при некоторых накладных расходах на коды ошибок.

Коды с повторением относятся к наиболее исследованным и часто используемым в существующих и вновь разрабатываемых вычислительных устройствах, обеспечивая заданные требования по надежности обработки информации [121].

Пусть информацией, подлежащей передаче или обработке, является последовательность двоичных символов. Каждый символ a_i этой последовательности кодируется кодером с помощью последовательности из пяти импульсов 00000, если $a_i = 0$, и 11111, если $a_i = 1$. В декодере принимаемая последовательность импульсов разбивается на группы по пять импульсов, называемые блоками, в

соответствии с разбиением передаваемой последовательности. Если в принятом блоке содержится два и менее импульса 0, то принимается решение о том, что передавался символ $a_i = 1$. Если в принятом блоке содержится три и более импульсов 0, то считается, что $a_i = 0$. Очевидно, что переданный символ a_i будет декодирован верно тогда и только тогда, когда число ошибочно принятых импульсов в соответствующем блоке не больше двух. Следовательно, вероятность ошибочного декодирования любого символа a_i равна $\sum_{i=3}^{5} C_5^i p_0^{5-i} (1-p_0)^i$, где C_n^m — число сочетаний из n по m, а $1-p_0$ — вероятность того, что значения 0 и 1 могут быть приняты с ошибкой как 1 и 0 [140].

Этот метод передачи является простым, но требует пятикратного увеличения времени передачи. И вопрос о соотношении сложности и эффективности кодирующих и особенно декодирующих устройств кодов с повторением остается открытым.

Среди корректирующих кодов наибольшее распространение получили блочные двоичные коды, т.е. передача двоичного сообщения производится блоками, причем каждый блок содержит двоичных символов. Кодирование и декодирование каждого блока производится независимо от других блоков [99]. Коды стирания преобразуют сообщение из k символов в сообщение из n символов, $k < n$, что исходное сообщение может быть восстановлено по любым k' символам. Простейшим таким кодов является код проверки на четность. Некоторому кодовому слову $(x_1, x_2, \ldots, x_{n-1})$ ставится в соответствие символ

$$x_n = x_1 \oplus x_2 \oplus \ldots \oplus x_{n-1},$$

где $\oplus$ — сложение по модулю 2. Таким образом, $x_n = 0$, если сумма x_i четна и $x_n = 1$, если она нечетна. Если в процессе передачи или обработки последовательности $(x_1, x_2, \ldots, x_n)$, содержащей один избыточный двоичный символ, возникла одиночная ошибка в одном из двоичных символов, то число символов 1 в последовательности становится нечетным. Это позволяет обнаруживать одиночные ошибки. Данный метод, ввиду своей простоты, находит широкое применение, однако, недостатком описанных выше методов повышения надежности, как и многих других алгебраических кодов, является невозможность выполнения арифметических операций над избыточными значениями. К тому же, для большинства кодов обнаружения и исправления ошибок характерно наличие двух групп цифр — информационной и контрольной. В информационную входят цифры, составляющие числовое значение закодированной величины, а

в контрольную — цифры, дополнительно вводимые для целей обнаружения и коррекции возможных искажений, которые являются избыточными [96].

Поскольку по контрольным частям компонентов арифметической операции нет возможности составить контрольную часть результата — исключается возможность контроля правильности выполнения арифметических операций. Данная неарифметичность специальных позиционных кодов препятствует их применению в вычислительных машинах, поскольку для них особое значение приобретает контроль арифметических операций.

Применение методов специального кодирования для вычислительных средств связано с тем, что любая ЭВМ по сути представляет из себя систему передачи и обработки информации. Передача информации в вычислительных машинах осуществляется с большой скоростью и значительными объемами, при этом должна быть обеспечена достоверность арифметической и логической обработки информации. Методы повышения отказоустойчивости вычислений, выполняемых вычислительными машинами и другими устройствами, развивались независимо в двух направлениях.

Первое направление восходит к работам фон Неймана [110] и связано с построением надежных устройств из ненадежных элементов. Результаты развития этого направления были подытожены Виноградом и Коуэном [100]. Во всех работах, относящихся к этому направлению, повышение надежности достигается различными методами введения так называемой структурной избыточности.

Особенностью второго направления является то, что само вычислительное устройство не модифицируется, а данные, которые в него вводятся, кодируются так, чтобы на выходе можно было осуществить обнаружение и исправление ошибок, возникших в процессе вычислений. Такое кодирование эффективно не для всякого вычислительного устройства; однако оно с успехом применяется, например, в устройствах, которые оперируют с целыми числами. В данном случае речь идет об арифметических кодах, называемых также AN-кодами, начало исследованию которых было положено работами J.M. Diamond [21] и D.T. Brown [15].

Кодовыми словами AN-кода являются r-ичные представления чисел-сообщений, умноженных на некоторое фиксированное число (модуль) A, так что кодовое слово суммы и разности двух чисел всегда будет соответственно суммой и разностью кодовых слов, соответствующих исходным числам. Благодаря этому AN-коды могут обнаруживать и исправлять как обычные ошибки, возникающие

в каналах связи, так и ошибки, возникающие в вычислительных устройствах. Такие коды могут применяться в каналах передачи данных, соединяющих вычислительные машины.

AN-код представляет собой отображение целых чисел $0, 1, 2, \ldots, N_0$ в целые числа $0, A \cdot 1, A \cdot 2, \ldots, A \cdot N_0$, где A — некоторое фиксированное для каждого кода целое положительное число, называемое порождающим числом. Числа $0, A \cdot 1, A \cdot 2, \ldots, A \cdot N_0$ называют кодовыми числами, а их представления в системе счисления по основанию r-кодовыми словами, например, для $r = 2$ — двоичными.

Очень важным свойством AN-кодов является то, что сумма кодовых чисел также является кодовым числом, а именно для любых двух целых чисел N_1 и N_2 [140]:

$$A\,(N_1 + N_2) = AN_1 + AN_2.$$

Следовательно, если $N_1 + N_2 \leq N_0$, то число $A\,(N_1 + N_2) = AN_1 + AN_2$ также является одним из кодовых чисел. Поэтому, если закодированные числа сложить с помощью обычного сумматора, то полученная в результате сумма будет кодовым числом суммы исходных чисел. Благодаря этому AN-коды могут обнаруживать и исправлять ошибки, возникающие в сумматорах. Если r является делителем A, т. е. если r является делителем каждого кодового числа AN, то в представлении чисел по основанию r самый младший разряд всегда будет равен 0. Далее, если A и r не являются взаимно простыми, то в младших разрядах, начиная с некоторого, будут появляться только определенные символы, тогда как другие цифры никогда не появятся. Поскольку это нежелательно, будем предполагать, что A и r взаимно просты. В случае $r = 2$ это означает, что число A должно быть нечетно. AN-коды с $r = 2$ называются двоичными AN-кодами [140].

Пример 1. *Построим AN-код для* $A = 7$, $N = 3$ *и* $r = 2$, *т.е. двоичный 7N-код, значения которого приведены в таблице 1.*

Например, сумме $1 + 2 = 3$ *соответствуют следующие суммы кодовых чисел и кодовых слов этого кода:* $7 + 14 = 21$ *и* $00111 + 01110 = 10101$. *Если при переносе* 1 *из разряда* 2^3 *в разряд* 2^4 *происходит ошибка, в результате которой эта единица переноса превращается в* 0, *то результатом вычисления будет число* $(01101)_2 = (13)_{10}$. *Остаток от деления этого числа на* 7 *равен* 6,

Таблица 1 — Двоичный 7N-код

N	AN	Кодовые слова
0	0	00000
1	7	00111
2	14	01110
3	21	10101

а это означает, что полученная сумма не является кодовым числом 7N-кода. Следовательно, возникшая ошибка будет всегда обнаружена.

Так как максимальное кодовое число равно AN_0, то для представления чисел $0, A{\cdot}1, A{\cdot}2, \ldots, A{\cdot}N_0$ в двоичной системе счисления требуется n двоичных символов, где

$$n = \lfloor \log_2 AN_0 + 1 \rfloor . \tag{1.2}$$

Число n называют длиной двоичного AN-кода. Для представления исходных некодированных чисел $0, 1, \ldots, N_0$ в двоичной системе счисления достаточно иметь

$$k = \lfloor \log_2 N_0 + 1 \rfloor \tag{1.3}$$

двоичных знаков. Таким образом, разность

$$r = n - k \tag{1.4}$$

представляет собой число избыточных двоичных символов, необходимых для того, чтобы множество целых чисел $\{0, 1, 2, \ldots, N_0\}$ можно было представить с помощью AN-кода.

Из формул (1.2)–(1.4) получаем [140]:

$$\lfloor \log_2 A \rfloor \leq r \leq \lfloor \log_2 A \rfloor + 1.$$

Следовательно, величина $\log_2 A$ является мерой избыточности AN-кода. Избыточность 7N-кода равна $\log_2 7 \approx 2{,}8$.

В представленном выше примере для целых чисел от 0 до $2^2 - 1 = 3$ в двоичной системе счисления необходимо 2 разряда, а для их представления в виде кодовых слов 7N-кода требуется уже 5 двоичных символов. Следовательно, фактически необходимая избыточность равна трем двоичным символам.

Избыточность 3N-кода равна $\log_2 3 \approx 1{,}6$. Если этот код используется для представления десятичных цифр $0, 1, 2, \ldots, 9$, то избыточность этого кода будет равна 1, поскольку в данном случае длина кода равна 5, а минимальное число двоичных символов, необходимых для представления тех же цифр в двоичной системе счисления, равно 4. Таким образом, использование 3N-кода при построении двоично-десятичных систем позволяет обнаружить возникающие ошибки и повысить надежность вычислительной системы.

Также к числу арифметических кодов относятся коды в остаточных классах и линейные вычетные коды.

1.2 Применение системы остаточных классов для повышения отказоустойчивости вычислительных узлов распределенной среды

Тенденции развития современного информационного общества ведут к увеличению объемов хранимой и обрабатываемой информации. Одним из перспективных решений данной задачи является использование распределенной среды. Однако при хранении и обработки данных в распределенной среде остро стоит вопрос обеспечения целостности и конфиденциальности информации. Последние достижения в области обеспечения отказоустойчивости и защиты информации привели к построению гомоморфных кодов с использованием системы остаточных классов [29], которые позволяют надежно защищать и обрабатывать данные без их предварительной расшифровки.

При этом результаты исследований по повышению производительности вычислительных систем, методов организации эффективной системы обнаружения и исправления ошибок и построения высоконадежных вычислительных систем показывают, что в пределах позиционных систем счисления нельзя ожидать значительного улучшения текущих результатов без существенного увеличения рабочих частот элементов и усложнения аппаратурной части цифровых вычислительных структур [96].

Новые пути организации структуры и логики вычислительных систем включают в себя как усложнения структуры и логики для увеличения их эффективности, так и поиск новых систем счисления и новых методов организации совместной работы всех элементов устройства и системы в целом.

При разработке структуры вычислительных систем одним из основных вопросов является выбор целесообразного представления числовой информации, т.е. соответствующего кода. Системы счисления можно рассматривать как различные способы кодирования числовой информации.

Основными требованиями к любой вычислительной системе, предназначенной для решения реальных практических задач, являются [96]:

1. Возможность представления в данной системе любой величины из рассматриваемого, заранее назначенного диапазона;
2. Единственность представления — любая кодовая комбинация соответствует одному и только одному числу в заданном диапазоне;
3. Простота оперирования с числами в данной системе счисления.

Поиски новых путей повышения эффективности выполнения арифметических операций привели исследователей к заключению, что в рамках обычной позиционной системы значительного ускорения выполнения операций добиться почти невозможно. Те или иные отдельные приемы и усовершенствования алгоритмов выполнения операций, способствуя более рациональной организации работы арифметических устройств, оставляют производительность этих устройств в рамках одного и того же порядка. Выход за эти пределы требует привлечения новых идей, новой логики и новой арифметики.

Следует отметить, что позиционные системы счисления, в которых представляется и обрабатывается информация в современных вычислительных системах, обладают существенным недостатком — наличием межразрядных связей, которые накладывают свой отпечаток на способы реализации арифметических операций, усложняют аппаратуру и ограничивают быстродействие. Поэтому естественно изыскание возможностей построения такой арифметики, в которой бы межразрядные связи отсутствовали.

Подобная арифметика может быть построена на базе непозиционной системы счисления, в частности, системы остаточных классов. Первые ЭВМ на базе модулярной арифметики появились в 50-е годы XX века. Разрабатываемая в 60-е годы ЭВМ 5Э53 представляла собой 8-процессорный комплекс, состоящий их 4 модулярных и 4 двоичных процессоров [107]. Такое решение обусловлено тем, что многие арифметические операции могут быть эффективно выполнены в системе остаточных классов, но есть ряд операций, которые эффективнее выполнять в позиционной системе счисления. Такое разделение на современном этапе развития вычислительной техники может быть реализовано в виде специ-

ализированных интегральных схем (ASIC, application-specific integrated circuit), с помощью которых можно построить распределенную среду обработки данных, для чего требуется разработка новых методов и алгоритмов создания программ и программных систем распределенной обработки данных.

В системе остаточных классов числа представляются своими остатками от деления на выбранные основания, и все модульные операции могут выполняться параллельно над цифрами каждого разряда в отдельности.

Обозначим через $\alpha = |X|_p = X \bmod p$ наименьший неотрицательный остаток от деления X на p, который можно вычислить по формуле

$$\alpha = X \bmod p = X - \left\lfloor \frac{X}{p} \right\rfloor p.$$

Если задан ряд положительных целых чисел $p_1, p_2, \ldots, p_n$, называемых модулями или основаниями системы, то под системой остаточных классов понимается система, в которой целое неотрицательное число представляется в виде набора остатков по выбранным основаниям: $X = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, где $\alpha_i = X \bmod p_i$ для $i = 1, 2, \ldots, n$ [96]. Если основания удовлетворяют условию $p_1 < p_2 < \ldots < p_n$, то система называется упорядоченной.

Из теории чисел известно, что если модули p_i взаимно простые, то представление числа $X = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ является единственным. При этом X удовлетворяет условию $X < P$, где $P = p_1 p_2 \ldots p_n$ — динамический диапазон представления чисел.

Эффективность системы остаточных классов во многом зависит от выбора набора её модулей. Существует несколько подходов к решению данной задачи. В работе [59] рассмотрен набор модулей специального вида, $\{2^n - 1,\ 2^n,\ 2^n + 1\}$. Подход, рассмотренный в [63] и ориентированный на вычисления на GPU, заключается в использовании в качестве модулей всех простых чисел из некоторого диапазона.

Другой подход, предложенный в [139], основан на переходе от вычислений по специально отобранным основаниям $p_1, \ldots, p_n$ к модульным рекурсивным вычислениям по базисным основаниям. Так, для СОК с модулями $\{p_1, p_2, p_3, p_4, p_5\} = \{2, 3, 5, 29, 863\}$ осуществляется переход в систему базовых модулей $\{p_1, p_2\} = \{2, 3\}$ и число $A = 865 = (1, 1, 0, 24, 2)$ можно представить в виде $(1, 1, (0, 0), (0, 0, (0, 1)), (0, 2, (0, 2), (0, 2, (0, 2))))$. Преимуществом данного метода является выполнение вычислений с двубитными числами, однако операция перевода из СОК в позиционную систему счисления является вычисли-

тельно сложной, и применение данного метода затруднено для задач с большим количеством операций сравнения чисел в СОК.

Также при выборе динамического диапазона необходимо учитывать не только размер обрабатываемых чисел, но и возможного результата арифметических операций. Так, в [106] показано, что набора модулей $\{5, 7, 8\}$, покрывающего 8-битный диапазон представления пикселей изображения, не хватает в случае цифровой фильтрации, что может привести к переполнению диапазона и получению ошибочных значений.

Особенностью системы остаточных классов является возможность выполнения операций сложения, вычитания и умножения параллельно и независимо по каждому из модулей. Для чисел $A = (a_1, a_2, \ldots, a_n)$ и $B = (b_1, b_2, \ldots, b_n)$, представленных в СОК, справедливо

$$C = A * B = (a_1 * b_1, a_2 * b_2, \ldots, a_n * b_n), \text{ где } * = \{+, -, \times\}.$$

Добавим в систему остаточных классов с модулями $\{p_1, p_2, \ldots, p_n\}$ и динамическим диапазоном $P = \prod_{i=1}^{n} p_i$, который также часто называют рабочим, дополнительное основание $p_{n+1} > p_i$, $i = \overline{1,n}$, тогда полный диапазон системы будет $\overline{P} = P \cdot p_{n+1}$. В случае выполнения операций над числами, лежащими в диапазоне $[0,P)$, если результат операции меньше P, то он корректен, иначе некорректен. Таким образом, введение избыточного основания p_{n+1} позволяет обнаруживать ошибки вычислений. Введение ограничений на избыточное основание или введение дополнительных оснований позволяет не только обнаруживать, но и исправлять ошибки. При этом система остаточных классов является полностью арифметичным кодом, где контрольная и информационная части совершенно равноправны относительно любой операции.

Обнаружение и исправление ошибок в избыточной системе остаточных классов (ИСОК) находит широкое применение и использовано в отказоустойчивых приложениях, описанных в литературе [76; 82; 90; 94]. Например, ИСОК с шестью модулями применяется при построении гибридных хранилищ [31; 94]. По сравнению с кодами Рида-Соломона, этот код обеспечивает хранение большего объема данных при аналогичных возможностях коррекции ошибок [30;44]. Добавление двух избыточных модулей также, помимо исправления ошибки, позволяет обнаружить переполнение диапазона СОК.

Как правило, обнаружение и исправление ошибок в ИСОК выполняется в три последовательных этапа: проверка наличия ошибок, идентификация оши-

бочных цифр остатка и исправление ошибок. Алгоритмы, которые исправляют только одну ошибку разряда остатка, были предложены в [4; 26; 81; 92]. В случае использования двух избыточных модулей, декодирование величины и местоположения одиночной ошибки может было решено с использованием только одного синдрома и одной таблицы поиска [37; 81]. С другой стороны, обнаружение и исправление нескольких ошибок остаточных разрядов является более сложным и трудоёмким. Это в основном связано с огромным количеством комбинаций различных местоположений и величин остатков для поиска ошибок на втором шаге. Как правило, существующие алгоритмы обнаружения и исправления множественных ошибок используют три различных метода для определения местоположения ошибочных цифр остатка. Это проверка согласованности с использованием синдрома [40;91], восстановление чисел по Китайской теореме об остатках [28] и проекции модуля [27;40].

Для избыточной СОК используют обозначение (k, n), где k — количество рабочих оснований, n — общее количество оснований.

Рассмотрим в общем случае СОК с одним избыточным основанием p_{n+1}, для которой $p_i < p_{n+1}$, $i = \overline{1,n}$. Тогда любое искажение цифры по одному какому-либо разряду превращает это число в неправильное и позволяет тем самым обнаружить наличие ошибки [96]. При этом накладывают ограничения, что ошибка возникает только по информационным основаниям.

Одним из методов локализации ошибок модулярного кода является метод проекций. Проекцией A_i числа $A = (\alpha_1, \alpha_2, \ldots, \alpha_{n+1})$ по основанию p_i будет число, полученное вычеркиванием цифры α_i в представлении A.

Теорема 1.2.1. *Если в упорядоченной системе остаточных классов проекция A_i числа $A = (\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots, \alpha_n, \alpha_{n+1})$ по основанию p_i удовлетворяет условию*

$$A_i > \frac{\overline{P}}{p_{n+1}},$$

то цифра α_i правильная, если возможна лишь одиночная ошибка [96].

Введение только одного контрольного основания в общем случае не позволяет локализовать ошибку. Для коррекции ошибки можно использовать метод на основе Китайской теоремы об остатках и методе проекций. Запишем его в виде Алгоритма 1.1.

Рассмотрим пример некорректной локализации ошибки.

Алгоритм 1.1. Коррекция ошибок на основе метода проекций и КТО

Вход: $X' = (x'_1, x'_2, \ldots, x'_n, x'_{n+1})$

Выход: X

Данные в памяти: $\{p_1, p_2, \ldots, p_{n+1}\}$, $P = \prod_{i=1}^{n} p_i$, $\overline{P} = p_n \cdot P$

$w_i = \left|\overline{P}_i^{-1}\right|_{p_i} \cdot \overline{P}_i$, где $\overline{P}_i = \overline{P}/p_i$, для всех $i \in [1,n+1]$

$w_{i,j} = \left|\overline{P}_{i,j}^{-1}\right|_{p_j} \cdot \overline{P}_{i,j}$, где $\overline{P}_{i,j} = \overline{P}_i/p_j$, для всех $i \in [1,n+1]\, j \neq i$

$S = 0$
Цикл от $i = 1$ **до** $n+1$ **выполнять**
    $S = S + x'_i \cdot w_i$
Конец цикла
$S = S \bmod \overline{P}$
Если $S < P$ **тогда**
    $X = S$
    Возвратить X
иначе
    Цикл от $i = 1$ **до** $n+1$ **выполнять**
        $S_i = 0$
        Цикл от $j = 1$ **до** $n+1$ **выполнять**
            Если $i \neq j$ **тогда**
                $S_i = S_i + x'_i \cdot w_{i,j}$
            Конец условия
        Конец цикла
        $X = S_i \bmod \overline{P}_i$
        Если $X < P$ **тогда**
            Возвратить X
        Конец условия
    Конец цикла
Конец условия

Пример 2. *Пусть задана СОК* $\{3, 5, 7, 11, 13\}$ *с четырьмя информационными основаниями и одним контрольным. Тогда рабочим диапазоном будет* $P = 1155$*, полный диапазон* $\overline{P} = 15015$*.*

Возьмем число $X = 4 = (1, 4, 4, 4, 4)$ *и введем ошибку по второму основанию, получим* $X' = (1, 0, 4, 4, 4) = 6010$*. Поскольку* $X' = 6010 > P = 1155$ *можно сделать вывод, что* X' *содержит ошибку. Тогда проекции* $X'_1 = 1005 = (0, 4, 4, 4)$*,* $X'_2 = 4 = (1, 4, 4, 4)$*,* $X'_3 = 1720 = (1, 0, 4, 4)$*,* $X'_4 = 550 = (1, 0, 4, 4)$*,* $X'_5 = 235 = (1, 0, 4, 4)$*. Очевидно, что проекции* X'_1*,* X'_2*,* X'_4*,* X'_5 *удовлетворяют условию корректности, откуда следует что одно избыточное основание позволяет только обнаружить ошибку, но не локализовать её.*

Для обеспечения возможности коррекции ошибок введем два контрольных основания p_{n+1}, p_{n+2}.

Теорема 1.2.2. *Если в системе* $\{p_1, p_2, \ldots, p_n, p_{n+1}, p_{n+2}\}$ *с двумя контрольными основаниями задано неправильное число* $A' = (\alpha'_1, \alpha'_2, \ldots, \alpha'_i, \ldots, \alpha'_n, \alpha'_{n+1}, \alpha'_{n+2})$*, то необходимым и достаточным условием ошибочности цифры* α'_i *в* A' *является правильности его проекции* A'_i *по основанию* p_i *[96].*

Рассмотрим аналогичный пример с двумя контрольными основаниями.

Пример 3. *Возьмем СОК* $\{3, 5, 7, 11, 13, 17\}$*, рабочий диапазон у которой будет* $P = 1155$*, а полный диапазон* $\overline{P} = 255255$*. Возьмем число* $X = 4 = (1{,}4{,}4{,}4{,}4{,}4)$ *и введем ошибку по второму основанию, получим* $X' = (1{,}0{,}4{,}4{,}4{,}4) = 51055$*. Поскольку* $X' = 51055 > P = 1155$ *можно сделать вывод, что* X' *содержит ошибку. Тогда проекции* $X'_1 = 51055$*,* $X'_2 = 4$*,* $X'_3 = 14590$*,* $X'_4 = 4645$*,* $X'_5 = 11785$*,* $X'_6 = 6010$*. Очевидно, что только* $X'_2 = 4 < P$*, значит ошибка произошла по второму основанию и исходное число равно 4.*

Однако столь удобной в одном отношении системе остаточных классов присущ ряд недостатков в других отношениях: ограниченность действия этой системы полем целых неотрицательных чисел, трудность определения соотношений чисел по величине, определения выхода результата операции из диапазона и т.д. Так, для СОК $\{3, 5, 7\}$ расположение чисел на числовой прямой показано на рисунке 1.1. Так, по виду чисел в СОК $(1, 1, 1)$ и $(1, 0, 0)$ нельзя сразу оценить, какое из них больше, и необходимы специальные методы вычисления позиционной характеристики числа.

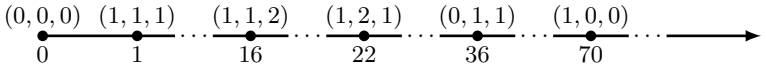

Рисунок 1.1 — Расположение модулярных чисел на числовой прямой

Недостатком исправления ошибки с использованием двух контрольных оснований с использованием Алгоритма 1.1 и Теоремы 1.2.2 является сложность восстановления чисел на основе КТО по каждой проекции.

Для того чтобы можно было строить распределенные вычислительные системы, работающие в системе остаточных классов, необходимо найти принципиальные пути преодоления этих трудностей и высокопроизводительные способы построения машинной арифметики.

1.3 Выводы по первой главе

Повышение сложности задач, решаемых системами обработки информации, в совокупности с ограниченностью вычислительных ресурсов устройств приводит исследователей к проблеме разработки высокоскоростных методов и методов, обладающих лучшей площадью устройств при едином времени обработки с учетом обеспечения требуемого уровня отказоустойчивости вычислений.

Повышение отказоустойчивости вычислительных систем сопряжено с ростом их сложности и стоимости. Одним из распространенных методов обеспечения отказоустойчивости является резервирование, однако, в этом случае значительно увеличивается аппаратная избыточность. Другим подходом является введение избыточности в процесс представления и преобразования информации. Большинство эффективных методов кодирования было разработано для обеспечения надежной передачи или хранения информации и исправления ошибок определенного вида.

Недостатком специальных позиционных кодов является их неарифметичность, связанная с неравноценностью информационной и контрольной частей кода. К кодам, позволяющим выполнять арифметические операции, относятся AN-коды, линейные вычетные коды и система остаточных классов.

Введение избыточных модулей в СОК позволяет не только обнаруживать, но и исправлять ошибки, однако в большинстве методы коррекции ошибок обладают высокой алгоритмической сложностью.

Высокие требования к производительности современных систем обработки данных приводят к необходимости разработки вычислительно эффективных, параллельных методов. Система остаточных классов обладает естественным параллелизмом, поэтому является наиболее подходящим для достижения поставленной цели инструментом. Возникает задача разработки математической модели системы обработки данных, использующей модулярную арифметику, а также разработки специализированных вычислительных узлов распределенной среды для выполнения немодульных операций в СОК.

В качестве критериев, позволяющих оценить эффективность разработанных методов и алгоритмов взяты время прохождения сигнала по схеме (пикосекунды, пс) и используемая площадь (квадратные микрометры, мкм^2).

Использование данных критериев позволяет сформулировать научную задачу исследования: исследование и разработка математических методов и алгоритмов выполнения немодульных операций на основе различных форм позиционной характеристики числа, способных повысить скорость и отказоустойчивость обработки информации вычислительными узлами распределенной среды, работающими в модулярном коде.

Далее в Главе 2 будет рассмотрена модификация алгоритма обнаружения и коррекции ошибок, для повышения производительности которого исследованы эффективные реализации как модульных операций, так и методы выполнения немодульных операций, таких как обратный перевод из системы остаточных классов в позиционную систему счисления, сравнение чисел, определение знака числа.

Глава 2. Повышение производительности математических методов и алгоритмов выполнения немодульных операций в СОК

2.1 Исследование методов перевода из позиционной системы счисления в СОК

Набор модулей системы остаточных классов существенно влияет на вычислительную сложность алгоритмов перевода чисел из позиционной системы счисления в СОК. Так как в сбалансированной СОК разрядность модулей удовлетворяет следующему условию: $b \geq \lceil \log_2 P/n \rceil$, где b — количество бит в модуле СОК, P — диапазон СОК, n — количество модулей СОК, то размер модуля зависит от диапазона СОК и от количества модулей. Учитывая, что вычислительная сложность перевода чисел из СОК в позиционную систему счисления с использованием Китайской теоремы об остатках, приближенного метода на основе КТО, зависит от количества модулей по линейному закону, то при выборе модулей следует учитывать следующие критерии:

1. Вычислительную сложность алгоритма перевода из ПСС в СОК.
2. Вычислительную сложность алгоритма перевода из СОК в ПСС.
3. Вычислительную сложность алгоритмов выполнения арифметических операций с числами, представленными в СОК.

Из всего вышесказанного следует, что по выбранным модулям должны иметься эффективные программные реализации основных модулярных операций (по площади, времени выполнения). С этой точки зрения можно выделить четыре класса модулей [59], алгоритмы нахождения остатков по которым будут рассмотрены ниже:

1. Чётные модули, которые являются степенями числа 2, т.е. $p = 2^n$, могут быть использованы только один раз, поскольку модули СОК должны быть взаимно простыми.
2. Числа Мерсенна вида $p = 2^n - 1$, которые среди нечётных модулей имеют самые эффективные реализации.
3. Целые числа вида $p = 2^n + 1$.
4. Числа общего вида.

Существует большое количество наборов модулей специального вида, для которых разработаны эффективные методы выполнения основных операций, например: $\{2^{n-1}-1, 2^n-1, 2^n\}$ [2], $\{2^n-1, 2^n, 2^n+1\}$ [17], $\{2^n-1, 2^k, 2^n+1\}$ [59], $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ [32]. Набор модулей $\{2^n-1, 2^k, 2^n+1\}$ позволяет за счет вариативности k гибко настроить систему под конкретную задачу.

Большинство модулей специального вида представлены тремя формами: $2^n-1, 2^n, 2^n+1$. Данный выбор обуславливается тем, что современные элементы и устройства вычислительной техники работают с двоичными данными. Очевидно, что использование модуля 2^n исключает использование других чётных модулей общего вида, поэтому будем рассматривать далее только нечётные модули общего вида. Наборы модулей с несколькими параметрами (k и n) за счет вариативности позволяют гибко настроить систему под конкретную задачу.

Рассмотрим подробнее операцию нахождения остатка от деления по этим модулям с точки зрения программной реализации.

Ниже представлен алгоритм выполнения операции $\bmod 2^n$ (Алгоритм 2.1). На вход подается число X размерности N бит, а на выход с помощью функции побитового логического умножения `AND` подаются младшие n бит входного числа.

Алгоритм 2.1. Вычисление остатка по модулю 2^n

Вход: X размерностью N бит

Выход: $X \bmod 2^n$ размерностью n бит

1: **Возвратить** (2^n-1) `AND` X

Программная реализация нахождения остатка по модулю 2^n не требует использования каких-либо логических элементов. Схема выполнения операции $\bmod 2^n$ представлена на рисунке 2.1.

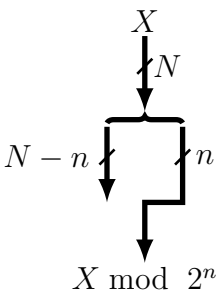

Рисунок 2.1 — Блок-схема операции $\bmod 2^n$

Таким образом, нахождение остатка по модулю 2^n является самым эффективным, но, как уже было сказано выше, может быть использовано всего один раз, поэтому возникает вопрос об оптимальном размере этого модуля. В виду простоты реализации нахождения остатка по модулю 2^n к данному модулю сводят некоторые методы обратного перевода из ПСС в СОК, рассмотренные в Параграфе 2.2.

Для вычисления остатков по модулям $2^n - 1$, $2^n + 1$ и производных от них разработаны специальные эффективные методы, например, основанные на использовании CSA (Carry-Save Adder) [65] или методы, использующие результаты вычисления остатка по одному модулю для вычисления остатков по другим [51]. Данные методы не так просты, как вычисления по модулю 2^n, однако, зачастую существенно эффективнее любых универсальных методов вычисления остатка от деления на произвольное положительное число. Проблема заключается в том, что количество модулей специального вида ограничено (обычно в наборе из 3–5), и либо приходится строить СОК с ограниченным числом вычислительных каналов, либо существенно жертвовать компактностью, используя модули различного порядка и получая заведомо разбалансированную СОК.

Нахождение остатков по нечётным модулям общего вида требует использования более сложных методов: нейронных сетей конечного кольца (НСКК), распределенной арифметики и т.д. Рассмотрим некоторые из них.

Распределенная арифметика. В статье [143] показан следующий метод вычисления остатка по модулю p. Пусть входное N-битное число X задано в двоичной системе счисления $X = \{x_{N-1} \dots x_1 x_0\}$. X можно представить в следующем виде:

$$X = \sum_{i=0}^{N-1} 2^i x_i, \tag{2.1}$$

где x_i — есть i-ый бит двоичного представления X. Тогда, используя свойства конечного кольца, можно получить, что

$$|X|_p \equiv \sum_{i=0}^{N-1} \left|2^i\right|_p x_i \bmod p. \tag{2.2}$$

Таким образом, отпадает необходимость умножения на числа больше модуля. Запишем реализацию формулы (2.2) в виде Алгоритма 2.2. На вход подается X размерности N бит, на выходе после вычисления суммы получаем результат $S \in [0, Np - N]$.

Алгоритм 2.2. Распределенная арифметика

Вход: $X = \{x_{N-1} \dots x_1 x_0\}$,

Выход: $S \in [0, Np - N]$

Данные в памяти: $\left|2^i\right|_p$

1: $S = 0$
2: **Цикл от** $i = 0$ **до** $N - 1$ **выполнять**
3: $\quad S = S + \left|2^i\right|_p \cdot x_i$
4: **Конец цикла**
5: **Возвратить** S

Использование распределенной арифметики для вычисления остатка от деления на модуль p позволяет найти число, сравнимое с модулем, но требует использования дополнительных методов для получения конечного результата. Для устранения данной проблемы распределенной арифметики в [93] предложено использование нейронной сети конечного кольца, состоящей из нескольких слоев. Рассмотрим алгоритм нахождения остатка от деления нейронной сетью конечного кольца (Алгоритм 2.3).

Алгоритм 2.3. Нейронная сеть конечного кольца

Вход: $X = \{x_{N-1} \dots x_1 x_0\}$

Выход: $x \in [0, 2p - 1]$

Данные в памяти: $\left|2^i\right|_p$

1: $x = X$
2: **Цикл от** $i = 0$ **до** $\lceil \log_2 N \rceil$ **выполнять**
3: $\quad S = 0$
4: $\quad$ **Цикл от** $j = 0$ **до** $\lfloor \log_2 x \rfloor + 1$ **выполнять**
5: $\quad\quad S = S + \left|2^j\right|_p \cdot x_j$
6: $\quad$ **Конец цикла**
7: $\quad x = S$
8: **Конец цикла**
9: **Возвратить** x

Рассмотрим на примере реализацию нейронной сети конечного кольца. Для наглядности возьмем нахождение остатка от деления 4-х битного числа на модуль 3.

Пример 4. *Найдем остаток от деления* 4*-х битного числа на модуль* 3*, т.е.* $X \bmod 3$*, где* $X = \{x_3x_2x_1x_0\}$.

Воспользуемся формулой (2.2)

$$|X|_3 = ||8|_3 \cdot x_3 + |4|_3 \cdot x_2 + |2|_3 \cdot x_1 + |1|_3 \cdot x_0|_3 = \\ = |2 \cdot x_3 + 1 \cdot x_2 + 2 \cdot x_1 + 1 \cdot x_0|_3 .$$

Если перевести коэффициенты перед x_i *в двоичный вид и применить к ним формулу* (2.1)*, можно свести нахождение остатка к использованию сумматоров и полусумматоров. Схема первого слоя нейронной сети показана на рисунке 2.2.*

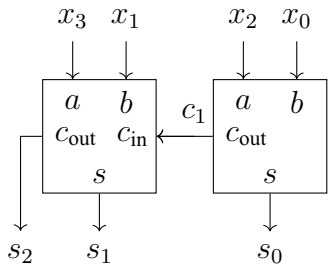

Рисунок 2.2 — Первый слой нейронной сети конечного кольца для $X \bmod 3$

Подавая на вход число $X = \{x_3x_2x_1x_0\} \in [0, 15]$*, на выходе получаем* $S = \{s_2s_1s_0\} \in [0, 6]$*. При этом входное и выходное значения сравнимы по данному модулю, например,* $X = 15_{10} = 1111_2 \equiv S = 6_{10} = 110_2 \bmod 3$*. Выход первого слоя не дает искомый остаток. Необходимо использовать дополнительные слои нейронной сети. Всего же таких слоев будет* $\lceil \log_2 N \rceil$*. Т.е. для данного примера их должно быть* 4*. Рассмотрим второй слой нейронной сети (рис. 2.3).*

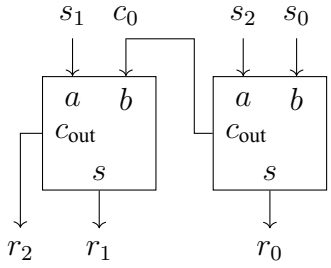

Рисунок 2.3 — Второй слой нейронной сети конечного кольца для $X \bmod 3$

Подавая $S = \{s_2s_1s_0\} \in [0, 6]$ *на вход схемы рисунка 2.3, получим* $R = \{r_2r_1r_0\} \in [0, 3]$*. Очевидно, что использование дополнительных слоев не*

приводит к желаемому результату. Подавая на вход число 3*, мы получим следующую последовательность*

$$3_{10} = 11_2 = 2^1 \cdot 1 + 2^0 \cdot 1 = |2^1|_3 \cdot 1 + |2^0|_3 \cdot 1 = 3.$$

Таким образом, данный метод дает результат в диапазоне $[0, 2p-1]$, что сопоставимо с рядом других методов, в частности с методом Баррета [54]. Для решения этой проблемы необходимо использовать дополнительные метод нахождения остатка.

Рассмотрим другой подход к нахождению остатка числа. В статье [65] описано свойство чисел, основанное на двух определениях.

Периодом $P(p)$ нечётного модуля p является минимальное расстояние между двумя последовательными степенями 2, для которых остаток по модулю p равен единице, т.е. $P(p) = \min\left\{k | k > 0 \text{ и } \left|2^k\right|_p = 1\right\}$.

Для заданного нечётного модуля p, если существует целое число k такое, что $\left|2^k\right|_p \equiv -1$, то *полупериодом* $HP(p)$ является минимальное расстояние между последовательными степенями числа 2, для которых остатки от деления на p равны 1 и $p-1$, т.е. $HP(p) = \min\left\{k | k > 0 \text{ и } \left|2^k\right|_p \equiv -1\right\}$.

Обратим внимание, что $|p-1|_p = -1$. Практическая значимость этих понятий заключается в следующих выражениях, где j – любое неотрицательное целое число:

$$\left|2^{jP(p)+i}\right|_p = \left|2^i\right|_p, \tag{2.3}$$

$$\left|2^{j \cdot HP(p)+i}\right|_p = (-1)^j \left|2^i\right|_p. \tag{2.4}$$

Используя формулу (2.2) и свойство $P(2^n - 1) = n$, можно разбить исходное число X на $r = \lceil N/n \rceil$ блоков B_i, начиная с младшего значащего бита, т.е. $X = \{B_{r-1} \dots B_1 B_0\}$, где $B_0 = \{x_{n-1} \dots x_1 x_0\}$, $B_1 = \{x_{2n-1} \dots x_{n+1} x_n\}$ и т.д. Если N не делится на n, то блок B_{r-1} может быть дополнен нулями [65]. Тогда из формулы (2.2) получим:

$$|X|_{2^n-1} = \left|\sum_{j=0}^{r-1} 2^{j \cdot n} \cdot B_j\right|_{2^n-1} = \left|\sum_{j=0}^{r-1} B_j\right|_{2^n-1}. \tag{2.5}$$

Сокращение разрядности операнд по модулю $2^n - 1$ с помощью периода числа можно описать Алгоритмом 2.4. На вход подается N-битное число X, разбитое на r блоков B_i по n бит каждый, а на выходе получаем число $S \in \left[0, \left\lceil \frac{N}{n} \right\rceil (2^n - 1)\right]$.

Алгоритм 2.4. Сокращение разрядности операнд по модулю $2^n - 1$ с использованием периода числа

Вход: $X = \left\{B_{\lceil N/n \rceil - 1} \dots B_1 B_0\right\}$
Выход: $S \in \left[0, \left\lceil \frac{N}{n} \right\rceil (2^n - 1)\right]$

1: $S = 0$
2: **Цикл от** $i = 0$ **до** $\lceil N/n \rceil - 1$ **выполнять**
3: $\quad S = S + B_i$
4: **Конец цикла**
5: **Возвратить** S

И опять в данном случае необходимо находить модуль суммы. Возможна рекурсивная версия данного алгоритма, в которой данный алгоритм повторяют до тех пор, пока S не станет меньше $2^{n+1} - 1$.

Для модуля $2^n + 1$ также можно воспользоваться формулой, аналогичной (2.5), однако при использовании периода размеры блоков будут равны $P\left(2^n + 1\right) = 2n$. Это позволит уменьшить размерность числа, но не позволит найти остаток без вычисления модуля суммы. Для модуля $2^n + 1$ возможно использование полупериода, т.е. используя формулу (2.4) в формуле (2.2), получим

$$|X|_{2^n+1} = \left| \sum_{j=0}^{r-1} 2^{j \cdot n} \cdot B_j \right|_{2^n+1} = \left| \sum_{j=0}^{r-1} (-1)^j B_j \right|_{2^n+1}.$$

Здесь также необходимо брать модуль суммы. При этом у каждого B_i необходим дополнительный бит для знака. Алгоритм 2.5 сокращения разрядности операнд по модулю $2^n + 1$ с помощью полупериода будет аналогичен Алгоритму 2.4.

Алгоритм 2.5. Сокращение разрядности операнд по модулю $2^n + 1$ с использованием полупериода числа

Вход: $X = \left\{B_{\lceil N/n \rceil - 1} \dots B_1 B_0\right\}$
Выход: $S \in \left[-\left\lceil \frac{N-n}{2n} \right\rceil (2^n - 1), \left\lceil \frac{N}{2n} \right\rceil (2^n - 1)\right]$

1: $S = 0$
2: **Цикл от** $i = 0$ **до** $\lceil N/n \rceil - 1$ **выполнять**
3: $\quad S = S + (-1)^i \cdot B_i$
4: **Конец цикла**
5: **Возвратить** S

Выход Алгоритма 2.5 может быть отрицательным, хотя и сравнимым с модулем $2^n + 1$, и для получения искомого остатка от деления необходимо производить сдвиг в положительную область, для чего к значению S прибавляют кратное число модулей, пока результат не станет положительным.

Данный метод применим и для нечётных модулей общего вида, однако размер периода в данном случае значительно больше, чем для модулей специального вида. Например, $P(13) = 12$, т.е. исходное число будет разбиваться на блоки по 12 бит, в то время как размер самого модуля 4 бита.

В статье [65] рассмотрено построение эффективных схем перевода в СОК одновременно по нескольким модулям, использующих общий ресурс. Суть метода заключается в том, что для модулей p_i находят периоды, а затем их наименьшее общее кратное, т.е. $P_{\text{НОК}} = \text{НОК}\{P(p_1), P(p_2), \ldots\}$. Затем входное число X размерностью N бит разбивается на блоки по $r = \lceil N/P_{\text{НОК}} \rceil$ бит и находится остаток по модулю $2^{P_{\text{НОК}}} - 1$. Это позволит получить число меньшей размерности, для которого легче найти остатки по модулям $p_1, p_2, \ldots, p_n$. Таким образом, нахождение остатка можно описать формулой

$$|X|_{p_i} = \left| |X|_{2^{P_{\text{НОК}}}-1} \right|_{p_i}.$$

Если модуль p_i имеет полупериод, то этот частный случай сводится к общему, т.е. к $P(p_i) = 2HP(p_i)$.

Отметим, что методу на основе периода и полупериода присущ тот же недостаток, что и нейронной сети конечного кольца, а именно, результат вычислений может не быть наименьшим неотрицательным вычетом и необходимы дополнительные вычисления.

В [67] рассмотрен следующий способ нахождения остатка от деления суммы чисел на модуль. Пусть даны числа a и b и модуль p, причем $0 \leq a, b < p$, а также $c = 2^n - p$. В данном случае n — размерность модуля. Тогда $(a + b) \bmod p$ можно найти выполнив следующие действия. Складываем числа, т.е. $s \leftarrow a + b$. Для проверки превышения модуля прибавим к результату суммы значение c, т.е. $t \leftarrow s + c$ и, если $t \geq 2^n$, то $s \leftarrow t \bmod 2^n$. Полученный результат s и будет остатком от деления (рис. 2.4).

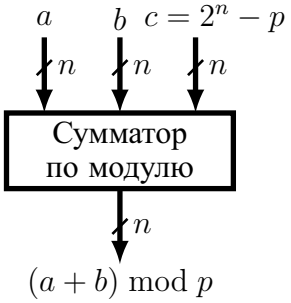

Рисунок 2.4 — Модулярный сумматор

2.1.1 Модифицированный метод нахождения остатка для модулей вида $2^n - 1$ и $2^n + 1$.

Введем метод нахождения остатка для модулей специального вида $2^n - 1$ и $2^n + 1$ на основе периода и полупериода числа и нахождения остатка от деления суммы чисел. Для этого применим метод нахождения остатка от деления на модуль p суммы $(a + b)$ к числу $X < 2p$, которое получается в результате рекурсивного применения Алгоритмов 2.4 и 2.5. Получим Алгоритм 2.6.

Данный алгоритм имеет эффективную аппаратную реализацию, т.к. нахождение остатка сводится к сложению и проверке старшего бита числа.

Алгоритм 2.6 можно обобщить для модулей вида $2^n + 1$, для которых константа $c = 2^n - 1$, на шаге 4 Алгоритма 2.6 необходимо выполнять $S = S + (-1)^i \cdot B_i$, учитывая возможность появления отрицательных чисел, на шаге 6 Алгоритма 2.6 проводится сдвиг в положительную область, а на шагах 9 и 10 Алгоритма 2.6 сравнение происходит с числом 2^{n+1}.

Для модулей общего вида данный алгоритм также применим, хотя и не так эффективен, как для случая $2^n - 1$.

Пример 5. *Рассмотрим в качестве примера Алгоритма 2.6 вычисление остатка от деления числа* $X = 155_{10} = 10011011_2$ *на модуль* $p = 2^3 - 1 = 7$.

Исходное число разбивается на блоки $B_2 = \{010\}$, $B_1 = \{011\}$, $B_0 = \{011\}$ *по 3 бита. В цикле (шаги 3–5 Алгоритма 2.6) выполняется сложение* B_i, *и на шаге 6 получаем* $X = 8_{10} = 1000_2$. *Поскольку* $X < 2^4$, *переходим к шагу 8, на котором получим* $t = 1000_2 + 1_2 = 1001_2$. *Для проверки условия* $t \geq 2^3$ *из шага 9 необходимо проверить старший бит* t *на равенство 1, и поскольку* $t \geq 2^3$ — *возвращаются 3 младших бита числа, а именно* $x = 001_2 = 1_{10}$.

Алгоритм 2.6. Модифицированный метод нахождения остатка по модулю $2^n - 1$ с использованием периода числа

Вход: $X = \{x_N \dots x_1 x_0\} = \{B_{\lceil N/n \rceil - 1} \dots B_1 B_0\}$, где $B_i = \{x_{(i+1)n-1} \dots x_{in}\}$, $i = \overline{0, \lceil N/n \rceil - 1}$

Выход: $x \in [0, p)$

Данные в памяти: $c = 1$ — константа для модулей вида $2^n - 1$

1: **До тех пор пока** $X \geq 2^{n+1}$ **выполнять**
2: $S = 0$
3: **Цикл от** $i = 0$ **до** $\left\lceil \frac{\log_2 X}{n} \right\rceil - 1$ **выполнять**
4: $S = S + B_i$
5: **Конец цикла**
6: $X = S$
7: **Конец цикла**
8: $t = X + c$
9: **Если** $t \geq 2^n$ **тогда**
10: **Возвратить** $x = t \bmod 2^n$
11: **иначе**
12: **Возвратить** $x = X$
13: **Конец условия**

Проверим корректность вычислений. Использование периода числа позволяет для числа $X = 155$ *получить сравнимое число* $X = 8$*, использование же суммы приводит к искомому результату* $x = 1$*. Недостаточность использования только периода числа обусловлена тем, что в случае получения на шаге 6* $X = \{B_0\} = 7_{10} = 111_2$*, т.е. модуля, дальнейшие вычисления невозможны, хотя и не получен искомый — 0, результат.*

Таким образом, разработан метод нахождения остатка по модулям вида $2^n - 1$, $2^n + 1$ с использованием периода и полупериода, позволяющий найти числа в диапазоне $[0, p)$.

Моделирование данного метода с использованием ASIC будет рассмотрено в Параграфе 3.2.

2.2 Исследование методов перевода чисел из СОК в позиционную системы счисления

Характеристики конкретной СОК полностью определяются её набором модулей. Применение модулей специального вида, а именно наборов $\{2^n-1, 2^n, 2^n+1\}$, $\{2^n-1, 2^n, 2^{n-1}-1\}$, показало свою эффективность для прямого преобразования [19] из ПСС в СОК. Рассмотрим процедуру обратного преобразования, т.е. перевода из СОК в ПСС. Существуют различные методы перевода, основанные на использовании просмотровых таблиц, на основе Китайской теоремы об остатках, обобщенной позиционной системы счисления, функции ядра и диагональной функции [47]. Рассмотрим эти методы подробнее.

Наиболее распространенным методом является метод на основе КТО. Если число X задано в виде остатков $(x_1, x_2, \ldots, x_n)$ от деления на модули $\{p_1, p_2, \ldots, p_n\}$, то на основе КТО [86] число X может быть получено из формулы

$$X = \left| \sum_{i=1}^{n} P_i \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_P, \tag{2.6}$$

где P — динамический диапазон, $P_i = P/p_i$, $\left|P_i^{-1}\right|_{p_i}$ — мультипликативная инверсия P_i по модулю p_i, а оператором $|X|_{p_i}$ обозначен остаток от деления X на p_i, т.е. $X \bmod p_i$. Запишем реализацию метода на основе КТО по формуле (2.6) в виде Алгоритма 2.7.

Пример 6. *Рассмотрим на примере процесс перевода числа из СОК в ПСС. Для простоты вычислений возьмем СОК* $\{3, 5, 7\}$ *и число* $X = (1, 2, 3) = 52$. *Вычислим параметры КТО:*

$$\begin{aligned} &P = p_1 \cdot p_2 \cdot p_3 = 105, \\ &P_1 = \frac{P}{p_1} = 35, \quad P_2 = \frac{P}{p_2} = 21, \quad P_3 = \frac{P}{p_3} = 15, \\ &\left|P_1^{-1}\right|_{p_1} = 2, \quad \left|P_2^{-1}\right|_{p_2} = 1, \quad \left|P_3^{-1}\right|_{p_3} = 1. \end{aligned}$$

Тогда по формуле (2.6) *получим*

$$X = |35 \cdot 1 \cdot 2 + 21 \cdot 2 \cdot 1 + 15 \cdot 3 \cdot 1|_{105} = 52.$$

Недостатком данного метода является необходимость нахождения остатка по большому модулю P, что является ресурсоемкой задачей.

Алгоритм 2.7. Перевод числа из СОК в ПСС на основе КТО

Вход: $(x_1, x_2, \ldots, x_n)$

Выход: X

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$

$P = \prod_{i=1}^{n} p_i$,

$P_i = P/p_i$, $i = 1, \ldots, n$,

$B_i = \left|P_i^{-1}\right|_{p_i} \cdot P_i$, $i = 1, \ldots, n$

1: $S = 0$

2: **Цикл от** $i = 1$ **до** n **выполнять**

3: $\quad S = S + B_i \cdot x_i$

4: **Конец цикла**

5: $X = S \bmod P$

6: **Возвратить** X

Еще одним способов повышения эффективности выполнения данной операции является использование приближенного метода на основе КТО. В статьях [78; 141] предложено дробное приближенное представление чисел на основе КТО. Разделим (2.6) на P и получим

$$\frac{X}{P} = \left| \sum_{i=1}^{n} x_i \cdot \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} \right|_1 = \left| \sum_{i=1}^{n} x_i \cdot k_i \right|_1, \tag{2.7}$$

где $k_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i}$ — константы выбранной системы, а операцией $|x|_1$ обозначено взятие дробной части числа x. При этом значение выражения (2.7) находится в интервале $[0, 1)$.

В данном случае операция нахождения остатка по большому модулю заменена на операцию взятия дробной части числа, что реализуется достаточно легко. Для получения точного значения нужно дробь $\frac{X}{P}$ умножить на P. Алгоритм 2.8 перевода числа из СОК в ПСС приближенным методом представлен ниже.

Рассмотрим аналогичный пример.

Пример 7. *Даны система остаточных классов* $\{3, 5, 7\}$ *и число* $X = (1, 2, 3) = 52$. *Найдем константы* k_i*:*

$$k_1 = \frac{2}{3}, \quad k_2 = \frac{1}{5}, \quad k_3 = \frac{1}{7}.$$

Алгоритм 2.8. Перевод числа из СОК в ПСС приближенным методом

Вход: $(x_1, x_2, \ldots, x_n)$

Выход: X

Данные в памяти:$\{p_1, p_2, \ldots, p_n\}$

$P = \prod_{i=1}^{n} p_i$

$P_i = P/p_i$, $i = 1, \ldots, n$

$k_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i}, i = 1, \ldots, n$

1: $S = 0$

2: **Цикл от** $i = 1$ **до** n **выполнять**

3: $\quad S = S + k_i \cdot x_i$

4: **Конец цикла**

5: $X = |S|_1 \cdot P$

6: **Возвратить** X

Тогда по формуле (2.7) *легко найти:*

$$\frac{X}{P} = \left|1 \cdot \frac{2}{3} + 2 \cdot \frac{1}{5} + 3 \cdot \frac{1}{7}\right|_1 = \frac{52}{105}, \quad X = \frac{52}{105} \cdot 105 = 52.$$

Вычислительная сложность Алгоритма 2.8 меньше, чем у Алгоритма 2.7 на основе КТО, однако дробные коэффициенты редко можно представить в виде конечной дроби, поэтому в случае аппаратной реализации возникает вопрос о точности округления.

Для выполнения приближенных вычислений дробные коэффициенты k_i умножаются на 2^N, где N — число двоичных знаков после запятой, обеспечивающее необходимый уровень точности вычислений, каждое полученное число округляется вверх до целого и после все вычисления производятся по модулю 2^N. Нахождение остатка по данному модулю аппаратно решается усечением, что является тривиальной задачей. Для вычислений можно воспользоваться оценкой, предложенной в статье [122]:

$$N = \left\lceil \log_2 \left(P \cdot \sum_{i=1}^{n} (p_i - 1) \right) \right\rceil.$$

Принципиально другим способом реализации обратного преобразования из СОК в ПСС является метод на основе обобщенной позиционной системы, который базируется на вычитании модулей и умножении на мультипликативную

инверсию модуля. В ОПСС переводимое число имеет следующий вид:

$$X = x_1 + d_1 p_1 + d_2 p_1 p_2 + d_3 p_1 p_2 p_3 + \ldots + d_{n-1} p_1 p_2 \ldots p_{n-1}, \qquad (2.8)$$

где $0 \leq d_i \leq (p_{i+1} - 1)$. Параметры d_i называются цифрами ОПСС. На каждом шаге алгоритма определяется только одна цифра ОПСС. После, полученные цифры ОПСС подставляются в выражение (2.8) для получения искомого числа.

Рассмотрим Алгоритм 2.9 нахождения числа с использованием ОПСС для трехмодульной СОК.

Алгоритм 2.9. Перевод числа из трехмодульной СОК в ПСС с использованием ОПСС

Вход: (x_1, x_2, x_3)

Выход: X

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$
$\left|p_3^{-1}\right|_{p_2}$, $\left|p_3^{-1}\right|_{p_1}$, $\left|p_2^{-1}\right|_{p_1}$

1: $d_0 = x_3$
2: $d_1 = \left(\left|x_2 - x_3\right|_{p_2} \cdot \left|p_3^{-1}\right|_{p_2}\right) \bmod p_2$
3: $y_1 = \left(\left|x_1 - x_3\right|_{p_1} \cdot \left|p_3^{-1}\right|_{p_1}\right) \bmod p_1$
4: $d_2 = \left(\left|y_1 - d_1\right|_{p_1} \cdot \left|p_2^{-1}\right|_{p_1}\right) \bmod p_1$
5: **Возвратить** $X = d_2 \cdot (p_2 p_3) + d_1 \cdot p_3 + d_0$

Данный алгоритм не требует выполнения операции нахождения остатка по модулю P, однако его нельзя выполнить параллельно. Обратим внимание, что для эффективного выполнения восстановления может быть выбран любой порядок модулей. Поясним данный метод на примере.

Пример 8. *Возьмем значения из предыдущего примера: СОК* $\{3, 5, 7\}$ *и число* $X = (1, 2, 3) = 52$. *Схема вычисления показана на рисунке 2.5. Зафиксируем значение* $x_3 = 3$. *Чтобы найти цифры ОПСС, вычислим*

$$d_1 = \left(\left|x_2 - x_3\right|_{p_2} \cdot \left|p_3^{-1}\right|_{p_2}\right) \bmod p_2 = \left(\left|2 - 3\right|_5 \cdot \left|7^{-1}\right|_5\right) \bmod 5 = (4 \cdot 3) \bmod 5 = 2,$$

$$y_1 = \left(\left|x_1 - x_3\right|_{p_1} \cdot \left|p_3^{-1}\right|_{p_1}\right) \bmod p_1 = \left(\left|1 - 3\right|_3 \cdot \left|7^{-1}\right|_3\right) \bmod 3 = (1 \cdot 1) \bmod 3 = 1,$$

$$d_2 = \left(\left|y_1 - d_1\right|_{p_1} \cdot \left|p_2^{-1}\right|_{p_1}\right) \bmod p_1 = \left(\left|1 - 2\right|_3 \cdot \left|5^{-1}\right|_3\right) \bmod 3 = (2 \cdot 2) \bmod 3 = 1.$$

Тогда по формуле (2.8) *результат будет равен* $X = d_2 \cdot (p_2 p_3) + d_1 \cdot p_3 + x_3 = 1 \cdot 35 + 2 \cdot 7 + 3 = 52$.

p_1	p_2	p_3
x_1	x_2	x_3
$y_1 = \left(\left\|x_1 - x_3\right\|_{p_1} \cdot \left\|p_3^{-1}\right\|_{p_1}\right) \bmod p_1$	$d_1 = \left(\left\|x_2 - x_3\right\|_{p_2} \cdot \left\|p_3^{-1}\right\|_{p_2}\right) \bmod p_2$	
$d_2 = \left(\left\|y_1 - d_1\right\|_{p_1} \cdot \left\|p_2^{-1}\right\|_{p_1}\right) \bmod p_1$		

Рисунок 2.5 — Перевод в ОПСС для трехмодульной СОК

В статье [33] была предложена реализация описанного метода с помощью ROM-памяти, где каждому остатку x_i соответствует цифра ОПСС, которые затем складываются по соответствующему модулю. Для предыдущего примера число $(1, 2, 3)$ разбивается на три: $(1, 0, 0)$, $(0, 2, 0)$, $(0, 0, 3)$, которым соответствуют вектора с цифрами ОПСС [2,0,0], [1,1,0], [1,1,3], которые необходимо сложить по соответствующим модулям $[(2 + 1 + 1) \bmod 3, (0 + 1 + 1) \bmod 5, (0 + 0 + 3) \bmod 7] = [1, 2, 3]$ и подставить в выражение $X = 35d_2 + 7d_1 + d_0 = 35 \cdot 1 + 7 \cdot 2 + 3 = 52$.

Интерес представляет применение ОПСС для специального набора модулей $\{2^n - 1, 2^n, 2^n + 1\}$, поскольку мультипликативные инверсии в данном случае достаточно просто реализуются аппаратно:

$$\left|(2^n + 1)^{-1}\right|_{2^n-1} = 2^{n-1}, \quad \left|(2^n + 1)^{-1}\right|_{2^n} = -1, \quad \left|(2^n)^{-1}\right|_{2^n-1} = 1,$$

т.е. умножение на данные значения выполняются сдвигом и инверсией.

Основным недостатком метода перевода числа из СОК в ПСС с помощью ОПСС является тот факт, что все вычисления выполняются последовательно, что замедляет выполнение преобразования при большом количестве модулей.

Во многих работах для перевода из СОК в ПСС применяется функция ядра [97]. В заданной СОК $\{p_1, p_2, \ldots, p_n\}$ выбирается константа $C(P)$, называемая ядром, которая может быть самым большим модулем СОК или произведением двух и более модулей.

Как и в методе на основе КТО, параметры СОК B_i определяются как $B_i = P_i \cdot \left|P_i^{-1}\right|_{p_i}$, также определим следующим образом веса w_i:

$$w_i = \left(\left|P_i^{-1}\right|_{p_i} \cdot C(P)\right) \bmod p_i.$$

При этом значения весов должны удовлетворять условию

$$C(P) = \sum_{i=1}^{n} P\frac{w_i}{p_i}, \tag{2.9}$$

откуда следует, что некоторые веса могут быть отрицательными. Для значений параметров СОК B_i, которые часто называют базисами СОК, также вычисляется функция ядра по формуле

$$C(B_i) = B_i \cdot \frac{C(P)}{P} - \frac{w_i}{p_i}, \tag{2.10}$$

при этом заметим, что данные параметры являются константами для выбранной системы остаточных классов.

Перевод в двоичную систему счисления может быть осуществлен путем определения функции ядра $C(X)$, которая для числа X задана формулой

$$C(X) = \left(\sum_{i=1}^{n} x_i \cdot C(B_i)\right) \bmod C(P) = \sum_{i=1}^{n} x_i \cdot C(B_i) - \alpha \cdot C(P), \tag{2.11}$$

где α — ранг функции. Тогда X может быть вычислено из выражения

$$X = \frac{P}{C(P)} \cdot \left(C(X) + \sum_{i=1}^{n} \frac{w_i}{p_i} x_i\right). \tag{2.12}$$

Чтобы избежать громоздких вычислений, а именно сложного деления, можно выбрать $C(P)$ равным степени 2.

Перевод из СОК в ПСС с использованием функции ядра может быть реализован Алгоритмом 2.10.

Алгоритм 2.10. Перевод числа из СОК в ПСС с использованием функции ядра

Вход: $(x_1, \ldots, x_n)$

Выход: X

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$,
P, $C(P)$, w_i, $C(B_i)$, $i = 1, \ldots, n$

1: $C(X) = (\sum_{i=1}^{n} x_i \cdot C(B_i)) \bmod C(P)$
2: $X = \frac{P}{C(P)} \cdot \left(C(X) + \sum_{i=1}^{n} \frac{w_i}{p_i} x_i\right)$
3: **Возвратить** X

Рассмотрим аналогичный предыдущим пример.

Пример 9. *В СОК* $\{3,5,7\}$ *возьмем число* $X = (1,2,3) = 52$. *Зафиксируем* $C(P) = 7$. *Из предыдущих примеров известны следующие значения:*

$$P = 105, P_1 = \frac{P}{p_1} = 35, P_2 = \frac{P}{p_2} = 21, P_3 = \frac{P}{p_3} = 15,$$
$$\left|P_1^{-1}\right|_{p_1} = 2, \left|P_2^{-1}\right|_{p_2} = 1, \left|P_3^{-1}\right|_{p_3} = 1.$$

Отсюда $B_1 = P_1 \cdot \left|P_1^{-1}\right|_{p_1} = 70$, $B_2 = 21$, $B_3 = 15$. *Вычислим значения весов:*

$$w_1 = (2 \cdot 7) \bmod 3 \equiv 2 \bmod 3 \equiv -1 \bmod 3,$$
$$w_2 = (1 \cdot 7) \bmod 5 \equiv 2 \bmod 5 \equiv -3 \bmod 5,$$
$$w_3 = (1 \cdot 7) \bmod 7 \equiv 0 \bmod 7.$$

Поскольку веса должны удовлетворять выражению (2.9), *то легко найти, что* $w_1 = 2$, $w_2 = -3$, $w_3 = 0$. *Тогда по формуле* (2.10) *получаем* $C(B_1) = 4$, $C(B_2) = 2$, $C(B_3) = 1$. *По формуле* (2.11) *находим* $C(X) = (1 \cdot 4 + 2 \cdot 2 + 1 \cdot 3) \bmod 7 = 4$.

По формуле (2.12) *находим*

$$X = \frac{105}{7} \cdot \left(4 + \frac{2 \cdot 1}{3} - \frac{3 \cdot 2}{5} + \frac{0 \cdot 3}{7}\right) = \frac{105}{7} \cdot \frac{4 \cdot 15 + 2 \cdot 5 - 6 \cdot 3}{15} = 52.$$

Несмотря на то, что результатом является целое число, программная реализация сложения дробей является нетривиальной задачей, которая может привести к накоплению ошибки. В данном случае в качестве решения можно взять числитель приведенной к общему знаменателю дроби в скобках, в случае выбора составного ядра $C(P)$ могут возникнуть трудности при сложении дробей с разными знаменателями. Также функция ядра может иметь критические ядра и результат вычисления будет отличаться на величину диапазона. Модификация функции ядра без критических ядер рассмотрена в Параграфе 2.3.

В литературе встречается еще один способ перевода чисел из СОК в ПСС и сравнения чисел, основанный на диагональной функции [22]. Для СОК $\{p_1, p_2, \ldots, p_n\}$ определяют параметр суммы коэффициентов (SQ) как $SQ = P_1 + P_2 + \ldots + P_n$, а также константы

$$k_i = \left|-p_i^{-1}\right|_{SQ}. \tag{2.13}$$

Диагональная функция для заданного числа $X = (x_1, x_2, \ldots, x_n)$ определяется как

$$D(X) = |x_1 k_1 + \ldots + x_n k_n|_{SQ}. \tag{2.14}$$

Обратим внимание, что диагональная функция $D(X)$ является монотонной и может быть использована для сравнения чисел [64]. Так, если $D(X) > D(Y)$, то $X > Y$, а вот равенство $D(X) = D(Y)$ не означает равенство чисел. В этом случае необходимо дополнительно сравнить какие-либо из соответствующих остатков, после чего сделать вывод о соотношении чисел. В статье [22] рассмотрена диагональная функция, не поддерживающая перевод из СОК в ПСС. Тем не менее, если формулу (2.6) умножить на $\frac{SQ}{P}$, получим масштабированное значение X:

$$\frac{X \cdot SQ}{P} = \left| \sum_{i=1}^{n} SQ \cdot \frac{x_i}{p_i} \cdot \left| P_i^{-1} \right|_{p_i} \right|_{SQ}. \tag{2.15}$$

Из определения k_i (2.13) можно вывести $\beta_i \cdot SQ - k_i \cdot p_i = 1$, откуда $\beta_i = \left| SQ^{-1} \right|_{p_i}$, которое эквивалентно $\beta_i = \left| P_i^{-1} \right|_{p_i}$. Таким образом,

$$k_i = \frac{SQ}{p_i} \cdot \left| P_i^{-1} \right|_{p_i} - \frac{1}{p_i},$$

откуда $\frac{SQ}{p_i} \cdot \left| P_i^{-1} \right|_{p_i} = k_i + \frac{1}{p_i}$. Тогда подставляя в (2.14) $k_i + \frac{1}{p_i}$ вместо k_i получим масштабированное значение $D'(X)$.

Таким образом, чтобы получить значение X подставим вычисленные значения в (2.15) и умножим на $\frac{P}{SQ}$.

$$X = \frac{P}{SQ} \cdot \left| \sum_{i=1}^{n} x_i \cdot \left(k_i + \frac{1}{p_i} \right) \right|_{SQ} = \frac{P \cdot D(X) + x_1 \cdot P_1 + \ldots + x_n \cdot P_n}{SQ}. \tag{2.16}$$

Получим Алгоритм 2.11 перевода числа из СОК в ПСС с использованием диагональной функции.

Алгоритм 2.11. Перевод числа из СОК в ПСС с использованием диагональной функции

Вход: $(x_1, \ldots, x_n)$

Выход: X

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$, $P_i, i = 1, \ldots, n$

$SQ = P_1 + \ldots + P_n$

$k_i = \left| -p_i^{-1} \right|_{SQ}$

1: $D(X) = |x_1 k_1 + \ldots + x_n k_n|_{SQ}$

2: $X = \frac{P \cdot D(X) + x_1 \cdot P_1 + \ldots + x_n \cdot P_n}{SQ}$

3: **Возвратить** X

Рассмотрим данный метод на примере.

Пример 10. *Аналогично даны СОК* $\{3,5,7\}$ *и число* $X = (1,2,3) = 52$. *Из предыдущих примеров известны* $P = 105$, $P_1 = 35$, $P_2 = 21$, $P_3 = 15$. *Тогда* $SQ = 71$ *и из* (2.13) $k_1 = 47$, $k_2 = 14$, $k_3 = 10$. *Найдем* $D(X) = |1 \cdot 47 + 2 \cdot 14 + 3 \cdot 10|_{71} = 34$. *Из* (2.16) *найдем искомое значение:*

$$X = \frac{105 \cdot 34 + 1 \cdot 35 + 2 \cdot 21 + 3 \cdot 15}{71} = 52.$$

Как видно из примера, диагональная функция является частным случаем функции ядра Акушского при значениях весов равных 1.

Рассмотрим еще один способ обратного преобразования на основе КТО, называемый Новая КТО (New CRT) [58], который описывается формулой

$$X = |x_1 + k_1 (x_2 - x_1) p_1 + k_2 (x_3 - x_2) p_1 p_2 + \ldots \\ \ldots + k_{n-1} (x_n - x_{n-1}) p_1 \ldots p_{n-1}|_P , \quad (2.17)$$

где

$$k_i = \left| \frac{1}{\prod_{j=1}^{i} p_j} \right|_{\prod_{j=i+1}^{n} p_j} , 1 \le i \le n-1. \quad (2.18)$$

Пример 11. *Для СОК* $\{7,8,9\}$ *коэффициенты* k_i *Новой КТО будут равны*

$$k_1 = \left| \frac{1}{p_1} \right|_{p_2 p_3} = 31, k_2 = \left| \frac{1}{p_1 p_2} \right|_{p_3} = 5.$$

Тогда по формуле (2.17) *для числа* $X = (3,2,1) = 10$ *получим*

$$X = |3 + 31 \cdot (2-3) \cdot 7 + 5 \cdot (1-2) \cdot 7 \cdot 8|_{504} = |-494|_{504} = 10.$$

В статье [88] предложены две модификации Новой КТО, которые обозначаются Новая КТО I (New CRT I) и Новая КТО II (New CRT II). Рассмотрим отдельно каждый из этих методов. В Новой КТО I константы k_i вычисляются так же по формуле (2.18), но далее они используются для нахождения коэффициентов a_i, $i = 0, \ldots, n-1$:

$$a_0 = |1 - k_1 p_1|_{p_1 p_2 \ldots p_{n-1} p_n} ,$$
$$a_1 = |k_1 - k_2 p_2|_{p_2 \ldots p_{n-1} p_n} ,$$
$$\ldots ,$$
$$a_{n-2} = |k_{n-2} - k_{n-1} p_{n-1}|_{p_{n-1} p_n} ,$$
$$a_{n-1} = |k_{n-1}|_{p_n} .$$

Далее для данных коэффициентов строится матрица характеристик

$$A = \begin{pmatrix} a_{0,0} & 0 & \dots & 0 & 0 \\ a_{0,1} & a_{1,1} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ a_{0,n-2} & a_{1,n-2} & \dots & a_{n-2,n-2} & 0 \\ a_{0,n-1} & a_{1,n-1} & \dots & a_{n-2,n-1} & a_{n-1,n-1} \end{pmatrix},$$

коэффициенты которой получаются из представлений a_i в обобщенной позиционной системе счисления, записанных по столбцам

$$\begin{aligned} a_0 &= a_{0,0} + a_{0,1}p_1 + \ldots + a_{0,n-1}p_1p_2\ldots p_{n-1}, \\ a_1 &= a_{1,1} + a_{1,2}p_2 + \ldots + a_{1,n-1}p_2\ldots p_{n-1}, \\ &\ldots, \\ a_{n-2} &= a_{n-2,n-2} + a_{n-2,n-1}p_{n-1}, \\ a_{n-1} &= a_{n-1,n-1}. \end{aligned}$$

Тогда из матрицы A и значения $X = (x_1, x_2, \ldots, x_n)$ может быть получен вектор $B = A \times X$, т.е.

$$B = \begin{pmatrix} B_0 \\ B_1 \\ \dots \\ B_{n-2} \\ B_{n-1} \end{pmatrix} = \begin{pmatrix} a_{0,0} & 0 & \dots & 0 & 0 \\ a_{0,1} & a_{1,1} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ a_{0,n-2} & a_{1,n-2} & \dots & a_{n-2,n-2} & 0 \\ a_{0,n-1} & a_{1,n-1} & \dots & a_{n-2,n-1} & a_{n-1,n-1} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_{n-1} \\ x_n \end{pmatrix}.$$

Тогда число X может быть восстановлено по формуле

$$X = |B_0 + B_1p_1 + B_2p_1p_2 + \ldots + B_{n-1}p_1p_2\ldots p_{n-1}|_{p_1p_2\ldots p_n}.$$

Рассмотрим аналогичный пример.

Пример 12. *Для СОК* $\{7, 8, 9\}$ *возьмем коэффициенты* k_i *из Примера 11 и вычислим коэффициенты* a_i*:*

$$\begin{aligned} a_0 &= |1 - 31 \cdot 7|_{504} = 288 = 1 + 1p_1 + 5p_1p_2, \\ a_1 &= |31 - 5 \cdot 8|_{72} = 63 = 7 + 7p_2, \\ a_2 &= |5|_9 = 5. \end{aligned}$$

Для числа $X = (3, 2, 1) = 10$ *получим вектор* B

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 7 & 0 \\ 5 & 7 & 5 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 17 \\ 34 \end{pmatrix},$$

из которого получим

$$X = |3 + 17 \cdot 7 + 34 \cdot 7 \cdot 8|_{504} = 10.$$

Для случая упорядоченной СОК с модулями, удовлетворяющими выражению $p_i > p_1 + \ldots_+ p_{i-1}$, или более строгому $p_i > 2p_{i-1}$, обратное преобразование может быть выполнено без взятия остатка по модулю P. В этом случае вычисляется функция $Y < 2P$

$$Y = B_0 + B_1 p_1 + B_2 p_1 p_2 + \ldots + |B_{n-1}|_{p_n} p_1 p_2 \ldots p_{n-1}$$

и искомое значение X равно

$$X = \begin{cases} Y, & \text{если } Y < P, \\ Y - P, & \text{если } Y \geq P. \end{cases} \tag{2.19}$$

Ограничение, накладываемое на модули, в этом случае приводит к несбалансированной СОК.

Пример 13. *Для 8-битного диапазона можно взять набор модулей* $\{3, 7, 13\}$*, размерность каждого модуля на 1 двоичный разряд больше. В этом случае*

$$k_1 = 61, k_2 = 5$$
$$a_0 = 91, a_1 = 26, a_2 = 5.$$

Тогда для числа $X = (1,3,10) = 10$ *получим*

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 5 & 0 \\ 4 & 3 & 5 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \\ 10 \end{pmatrix} = \begin{pmatrix} 1 \\ 17 \\ 63 \end{pmatrix}.$$

Искомое значение X *может быть получено из* (2.19) *и вектора* $B = \left(B_0, B_1, |B_2|_{p_3}\right)$*:* $Y = 1 + 17 \cdot 3 + 11 \cdot 3 \cdot 7 = 283$*, т.к.* $Y > P = 273$*, то* $X = Y - P = 10$*.*

Данный метод требует матричного умножения с числами малой разрядности, нахождения остатка по модулю СОК и умножения с накоплением для n слагаемых.

Другой метод, предложенный в статье [88], получил название Новая КТО II (New CRT II). Для двухмодульной СОК с модулями $\{p_1, p_2\}$ используется Алгоритм 2.12, обозначаемый $\texttt{findno}((x_1, x_2), \{p_1, p_2\}, X)$.

Алгоритм 2.12. Перевод числа из СОК в ПСС для двухмодульной СОК (алгоритм `findno`)

Вход: (x_1, x_2), $\{p_1, p_2\}$

Выход: $X < p_1 \cdot p_2$

Данные в памяти: $k_0 = \left|p_2^{-1}\right|_{p_1}$

1: **Возвратить** $X = x_2 + \left|k_0 (x_1 - x_2)\right|_{p_1} \cdot p_2$

Для упорядоченной СОК с модулями $p_1 < p_2 < \ldots < p_n$ используется рекурсивный Алгоритм 2.13 — $\texttt{translate}((x_1, x_2, \ldots, x_n), \{p_1, p_2, \ldots, p_n\}, X)$.

Алгоритм 2.13. Перевод числа из СОК в ПСС (алгоритм `translate`)

Вход: $(x_1, x_2, \ldots, x_n)$, $\{p_1, p_2, \ldots, p_n\}$

Выход: X

1: **Если** $n > 2$ **тогда**

2: $\quad \texttt{translate}\left(\left(x_1, \ldots, x_{\lfloor \frac{n}{2} \rfloor}\right), \left\{p_1, p_2, \ldots, p_{\lfloor \frac{n}{2} \rfloor}\right\}, N_1\right)$, $P_1 = \prod_{i=1}^{\lfloor \frac{n}{2} \rfloor} p_i$

3: $\quad \texttt{translate}\left(\left(x_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, x_n\right), \left\{p_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, p_n\right\}, N_2\right)$, $P_2 = \prod_{i=\lfloor \frac{n}{2} \rfloor + 1}^{n} p_i$

4: $\quad \texttt{findno}((N_1, N_2), \{P_1, P_2\}, X)$

5: **иначе если** $n = 2$

6: $\quad \texttt{findno}((x_1, x_2), \{p_1, p_2\}, X)$

7: **иначе если** $n = 1$

8: $\quad X = x_1 \bmod p_1$

9: **Конец условия**

Пример 14. *Рассмотрим пример применения Алгоритма 2.13 для СОК* $\{7, 8, 9\}$ *и числа* $X = (3, 2, 1) = 10$. *Так как* $n = 3$, *то в соответствии с шагом 1 алгоритма 2.13 получаем два новых вызова данного алгоритма* $\texttt{translate}((x_1), \{p_1\}, N_1)$ *и* $\texttt{translate}((x_2, x_3), \{p_2, p_3\}, N_2)$.

Вызов $\texttt{translate}((x_1), \{p_1\}, N_1)$ *приводит к шагу 7 Алгоритма 2.13, для которого получим* $N_1 = x_1 \bmod p_1$, $P_1 = p_1$, *т.е.* $N_1 = 3, P_1 = 7$. *Вызов*

`translate`$((x_2, x_3), \{p_2, p_3\}, N_2)$ *приводит к шагу 5 Алгоритма 2.13, для которого из Алгоритма 2.12* `findno` *получим* $N_2 = x_3 + \left|\left|p_3^{-1}\right|_{p_2} \cdot (x_2 - x_3)\right|_{p_2} \cdot p_3$, $P_2 = p_2 \cdot p_3$, *т.е.* $N_2 = 1 + |1 \cdot (2-1)|_8 \cdot 9 = 10, P_2 = 8 \cdot 9 = 72$.

Затем в соответствии с шагом 4 Алгоритма 2.13 происходит вызов `findno`$((N_1, N_2), \{P_1, P_2\}, X)$, *т.е. вычисление*

$$X = N_2 + \left|\left|P_2^{-1}\right|_{P_1} \cdot (N_1 - N_2)\right|_{P_1} \cdot P_2 = 10 + \left|\left|72^{-1}\right|_7 \cdot (3-10)\right|_7 \cdot 72 = 10.$$

Таким образом, метод Новая КТО II основан на рекурсивном использовании ОПСС с двумя модулями. В данном случае возникают сложности с вычислением мультипликативной инверсии на последних ступенях алгоритма, однако при программной реализации данные значения могут быть записаны в память.

В статье [60] рассмотрен набор нечётных модулей $\{p_1, p_2, \ldots, p_{n-1}\}$, диапазон которого $P_h = \prod_{i=1}^{n-1} p_i$, и из формулы (2.6) можно получить

$$X_h = \left|\sum_{i=1}^{n-1} P_i \left|\frac{x_i}{P_i}\right|_{p_i}\right|_{P_h}.$$

Добавление остатка x_n по модулю $p_n = 2^k$ приводит к выражению

$$X = \left|2^k \left|\frac{X_h}{2^k}\right|_{P_h} + P_h \left|\frac{x_n}{P_h}\right|_{2^k}\right|_{2^k P_h} = \left|2^k \sum_{i=1}^{n-1} P_i \left|\frac{x_i}{2^k P_i}\right|_{p_i} + P_h \left|\frac{x_n}{P_h}\right|_{2^k}\right|_{2^k P_h} \tag{2.20}$$

Для разбиения величины $x'_n = \left|\frac{x_n}{P_h}\right|_{2^k}$ на блоки x_q и x_p по $k-d$ и d бит соответственно, вычисляется значение $d = \lceil \log_2 (n-1) \rceil$, т.е. $x'_n = 2^{k-d} x_p + x_q$.

Далее вводятся переменные

$$x'_i = \left|\frac{x_i}{2^{k-d} P_i}\right|_{p_i}, x'_p = \left|x_p - \sum_{i=1}^{n-1} \frac{x'_i}{p_i}\right|_{2^d}$$

и $x''_n = 2^{k-d} x'_p + x_q = \left(x'_p || x_q\right)$, ($||$ — конкатенация двоичных значений).

Тогда формула (2.20) может быть переписана в виде

$$X = \left|2^{k-d} \sum_{i=1}^{n-1} P_i x'_i + P_h x''_n\right|_{2^k P_h}, \tag{2.21}$$

При этом, как и в (2.19), значение $X < 2 \cdot 2^k P_h$, т.е. не превосходит двойного диапазона и может быть вычислено без вычисления остатка по большому модулю.

Пример 15. *Рассмотрим аналогичный пример, с модулями* $\{7, 9, 8\}$*, тогда* $n = 3$*,* $k = \log_2 8 = 3$*,* $P_h = p_1 \cdot p_2 = 63$*,* $P_1 = \frac{P_h}{p_1} = p_2 = 9$*,* $P_2 = p_1 = 7$*,* $d = \lceil \log_2 (3-1) \rceil = 1$*.*

Для числа $X = (3{,}1{,}2) = 10$ *вычислим* x'_i *и* x_q*,* x_p*,* x'_p*,* x''_n*.*

$$x'_1 = \left| \frac{x_1}{2^{3-1} P_1} \right|_{p_1} = \left| 3 \cdot \left| \left(2^2 \cdot 9\right)^{-1} \right|_7 \right|_7 = 3, x'_2 = 1, x_p = 1, x_q = 2.$$

$$x'_p = \left| x_p - \sum_{i=1}^{n-1} \frac{x'_i}{p_i} \right|_{2^d} = 1, x''_n = 2^{k-d} x'_p + x_q = 6.$$

Тогда из (2.21) *получим* $X = |514|_{504} = 10$*.*

Данный метод позволяет перевести число из СОК в ПСС без вычисления остатка по большому модулю.

Для повышения производительности в СОК многие исследователи рассматривают модули специального вида $\{2^n - 1, 2^n, 2^n + 1\}$ [58], $\left\{2^{2n}, 2^{2n-1} - 1, 2^{2n} - 1, 2^{2n+1} - 1\right\}$ [49], $\left\{2^k - 3, 2^k - 2, 2^k - 1\right\}$ и $\left\{2^k + 1, 2^k + 2, 2^k + 3\right\}$ [62], $\left\{2^{2n-5} - 1, 2^{2n-3} - 1, 2^{2n-2} + 1, 2^{2n-1} - 1, 2^{2n-1} + 1, 2^{2n}, 2^{2n} + 1\right\}$ [45], $\left\{2^{2n}, 2^n + 1, 2^n - 1, 2^n + 3, 2^n - 3\right\}$ [50].

Разработано множество более эффективных методов обратного преобразования для наборов модулей специального вида, например, в статье [9] для числа $X = (x_1, x_2, x_3)$ в СОК $\{2^n - 1, 2^n, 2^n + 1\}$ используется формула $X = 2^n \cdot Y + x_2$, где

$$Y = \left(\left(2^n + 1\right) 2^{n-1} x_1 - 2^n x_2 + \left(2^n + 1\right) 2^{n-1} x_3 - x_3 \right) \bmod \left(2^{2n} - 1\right).$$

В данном случае нахождение остатка по большому модулю P заменяется на нахождение остатка по модулю специального вида $2^{2n} - 1$, для которого существуют эффективные реализации.

Наибольший интерес для методов обратного перевода представляет оптимизация СОК $\{2^n - 1, 2^n, 2^n + 1\}$, предложенная в статье [58]. Наибольшую эффективность имеют реализации на основе модуля $2^n - 1$.

Для модуля $2^n - 1$ выполняются следующие свойства:

1. Смена знака (аддитивная инверсия) достигается побитовым отрицанием, т.е. $|-z|_{2^n-1} = |\bar{z}|_{2^n-1} = |\bar{z}_{n-1} \dots \bar{z}_0|_{2^n-1}$, например, $|-5|_7 = |-101_2|_7 = \left|\overline{101}_2\right|_7 = |010_2|_7 = 2$.

2. Умножение на 2^d может быть выполнено циклическим сдвигом влево на d позиций, т.е. $\left|2^d z\right|_{2^n-1} = |z_{n-d-1}\dots z_0 z_{n-1}\dots z_{n-d}|_{2^n-1}$, например, $|2\cdot 5|_7 = |2\cdot 101_2|_7 = |011_2|_7 = 3$.
3. Умножение на мультипликативную инверсию $\left|2^{-d}\right|_{2^n-1}$ может быть выполнено циклическим сдвигом вправо на d позиций или влево на $n-d$ позиций, т.к. $\left|2^{-d}z\right|_{2^n-1} = \left|2^{n-d}z\right|_{2^n-1}$, т.е. $\left|2^{-d}z\right|_{2^n-1} = |z_{d-1}\dots z_0 z_{n-1}\dots z_d|_{2^n-1}$, например, $\left|2^{-1}\cdot 5\right|_7 = \left|2^{-1}\cdot 101_2\right|_7 = |110_2|_7 = 3$.
4. Для любого неотрицательного целого i: $\left|2^{i\cdot n}\right|_{2^n-1} = 1$, тогда остаток (sn)-битного числа по модулю 2^n-1 может быть представлен в виде

$$\left|\sum_{i=0}^{s-1} 2^{ni}\left(y_{n(i+1)-1}\dots y_{ni}\right)\right|_{2^n-1} = \left|\sum_{i=0}^{s-1}\left(y_{n(i+1)-1}\dots y_{ni}\right)\right|_{2^n-1}.$$

Статья [58] построена на идее, что выражение для обратного преобразователя на основе КТО (2.6) может быть представлено в форме, в которой можно разделить константу и переменные:

$$X = \left|\left(\sum_{i=1}^{n} b_i\cdot x_i\right) + C_k\right|_P.$$

Из набора модулей $\{p_1,p_2,p_3\} = \{2^n, 2^n-1, 2^n+1\}$ возьмем два модуля $\{p_2,p_3\} = \{2^n-1, 2^n+1\}$, для которых на основе КТО найдем число $X_{23} = (x_2, x_3)$:

$$\begin{aligned} X_{23} &= \left|p_3\cdot x_2\cdot\left|p_3^{-1}\right|_{p_2} + p_2\cdot x_3\cdot\left|p_2^{-1}\right|_{p_3}\right|_{p_2p_3} = \\ &= \left|(2^n+1)\cdot 2^{n-1}\cdot x_2 + (2^n-1)\cdot 2^{n-1}\cdot x_3\right|_{2^{2n}-1} = \\ &= \left|2^{n-1}\cdot\left((2^n+1)\cdot x_2 + (2^n-1)\,x_3\right)\right|_{2^{2n}-1}. \end{aligned}$$

На основе ОПСС для набора модулей $\{p_1, p_2p_3\}$ найдем число $X = (x_1, X_{23})$

$$X = x_1 + p_1\left|(X_{23}-x_1)\frac{1}{p_1}\right|_{p_2p_3} = x_1 + 2^n X_h, \tag{2.22}$$

где

$$\begin{aligned} X_h &= \left|(X_{23}-x_1)\,2^{-n}\right|_{2^{2n}-1} = \\ &= \left|\left(2^{n-1}\left((2^n+1)\,x_2 + (2^n-1)\,x_3\right) - x_1\right)2^{-n}\right|_{2^{2n}-1} = \\ &= \left|-x_1 2^{-n} + x_2\,(2^n+1)\,2^{-1} + x_3\,(2^n-1)\,2^{-1}\right|_{2^{2n}-1}. \end{aligned} \tag{2.23}$$

Оптимизация выражения (2.23) возможна за счет битовых операций. Из свойств 1 и 3 и свойства $|2^n|_{2^{2n}-1} = |2^{-n}|_{2^{2n}-1}$ первое слагаемое можно представить в виде

$$\begin{aligned}\left|-x_1 \cdot 2^{-n}\right|_{2^{2n}-1} &= \left|(\bar{x}_{1,n-1} \dots \bar{x}_{1,0}) \,||\, \underbrace{(1 \dots 1)}_{n}\right|_{2^{2n}-1} = \\ &= \left|(\bar{x}_{1,n-1} \dots \bar{x}_{1,0}) \,||\, \underbrace{(0 \dots 0)}_{n} + (2^n - 1)\right|_{2^{2n}-1} = \\ &= \left|2^{-1}\left((\bar{x}_{1,n-2} \dots \bar{x}_{1,0}) \,||\, \underbrace{(0 \dots 0)}_{n} \,||\bar{x}_{1,n-1}\right) + \left(2^{-n} - 1\right)\right|_{2^{2n}-1}.\end{aligned}$$

Второе слагаемое представляется в виде

$$\left|x_2 \left(2^n + 1\right) 2^{-1}\right|_{2^{2n}-1} = \left|2^{-1} \left(x_2 || x_2\right)\right|_{2^{2n}-1}.$$

Третье же слагаемое можно представить в виде

$$\begin{aligned}\left|x_3 \left(2^n - 1\right) 2^{-1}\right|_{2^{2n}-1} &= \left|2^{-1} \left(2^n x_3 - x_3\right)\right|_{2^{2n}-1} = \\ &= \left|2^{-1}\left(x_{3,n-1} \dots x_{3,0} \underbrace{0 \dots 0}_{n-1} x_{3,n} + \underbrace{1 \dots 1}_{n-1} \bar{x}_{3,n} \dots \bar{x}_{3,0}\right)\right|_{2^{2n}-1} = \\ &= \left|2^{-1}\left(x_{3,n-1} \dots x_{3,0} \underbrace{0 \dots 0}_{n-1} x_{3,n} + \underbrace{0 \dots 0}_{n-1} \bar{x}_{3,n} \dots \bar{x}_{3,0} + 2^{2n} - 2^{n+1}\right)\right|_{2^{2n}-1}.\end{aligned}$$

Поскольку x_3 является остатком от деления на модуль $2^n + 1$, то в случае $x_{3,n} = 1$ остальные биты числа равны 0. Таким образом можно получить следующие тождественные выражения

$$\left|x_3 \left(2^n - 1\right) 2^{-1}\right|_{2^{2n}-1} = \left|2^{-1} \left(\left(x_{3,n-1} \vee x_{3,n}\right) \dots \left(x_{3,0} \vee x_{3,n}\right) || \bar{x}_{3,n-1} \dots \bar{x}_{3,0} + \right.$$
$$\left. + 2^n + \left(1 - 2^{n+1}\right)\right)\right|_{2^{2n}-1}. \quad (2.24)$$

или

$$\left|x_3 \left(2^n - 1\right) 2^{-1}\right|_{2^{2n}-1} = \left|2^{-1} \left(x_{3,n-1} \dots x_{3,0} || \bar{x}_{3,n-1} \dots \bar{x}_{3,1} || \left(\overline{x_{3,0} \vee x_{3,n}}\right) + \right.\right.$$
$$\left.\left. + \underbrace{0 \dots 0}_{n-1} || \bar{x}_{3,n} || \underbrace{0 \dots 0}_{n-1} || x_{3,n} || 0 + \left(1 - 2^{n+1}\right)\right)\right|_{2^{2n}-1}. \quad (2.25)$$

В зависимости от выбранного представления третьего слагаемого (2.24) или (2.25), в статье [58] строится два варианта обратного преобразователя по формуле

$$X_h = |w_1 + w_2 + w_3 + C_k|_{2^{2n}-1}\,. \tag{2.26}$$

В первом случае на основе формулы (2.24) имеем

$$w_1 = \left(\bar{x}_{1,n-1}\ldots\bar{x}_{1,0}||\underbrace{(0\ldots0)}_{n}\right),$$
$$w_2 = (x_{2,0}||x_{2,n-1}\ldots x_{2,0}||x_{2,n-1}\ldots x_{2,1})\,,$$
$$w_3 = (\bar{x}_{3,0}||\,(x_{3,n-1}\vee x_{3,n})\ldots(x_{3,0}\vee x_{3,n})\,||\bar{x}_{3,n-1}\ldots\bar{x}_{3,1})\,,$$
$$C_k = \left|\left(2^{-n}-1\right)+2^{-1}\left(1-2^{n+1}\right)+2^{n-1}\right|_{2^{2n}-1} = 2^{2n-1}+2^{n-1}-1 =$$
$$= \left(1\underbrace{0\ldots0}_{n}\underbrace{1\ldots1}_{n-1}\right).$$

В статье [58] имеет место опечатка, где указано, что C_k имеет $n+2$ нуля. Однако, очевидно, что 2^{2n-1} содержит единицу и $2n-1$ ноль, а число $(2^{n-1}-1)$ содержит $n-1$ единицу. Тогда сложение чисел приведет к тому, что младшие $n-1$ бит будут равны единице, остальные $2n-1-(n-1)=n$ нулей останутся нулями. Например, для $n=2$, $2^3+2-1=9=1001_2$.

Во втором случае на основе формулы (2.25) имеем

$$w_1 = \left(\bar{x}_{1,n-1}\ldots\bar{x}_{1,0}||\bar{x}_{3,n}||\underbrace{(0\ldots0)}_{n-2}||x_{3,n}\right),$$
$$w_2 = (x_{2,0}||x_{2,n-1}\ldots x_{2,0}||x_{2,n-1}\ldots x_{2,1})\,,$$
$$w_3 = \left(\overline{(x_{3,0}\vee x_{3,n})}||x_{3,n-1}\ldots x_{3,0}||\bar{x}_{3,n-1}\ldots\bar{x}_{3,1}\right),$$
$$C_k = \left|\left(2^{-n}-1\right)+2^{-1}\left(1-2^{n+1}\right)\right|_{2^{2n}-1} = 2^{2n-1}-1 = \left(0\underbrace{1\ldots1}_{2n-1}\right).$$

Добавление C_k при вычислении X_h по формуле (2.26) равносильно добавлению 2^nC_k к X_{23}. Более того, последнее может быть также достигнуто путем одновременного добавления $c_2 = |2^nC_k|_{2^n-1} = |C_k|_{2^n-1}$ к x_2 по модулю 2^n-1 и $c_3 = |2^nC_k|_{2^n+1} = |-C_k|_{2^n+1}$ к x_3 по модулю 2^n+1 (при этом остаток x_1 по чётному модулю остается неизменным, так как $|2^nC_k|_{2^n}=0$).

В первом случае получим

$$c_2 = \left|2^n\left(2^{2n-1}+2^{n-1}-1\right)\right|_{2^n-1} = 0,$$
$$c_3 = \left|2^n\left(2^{2n-1}+2^{n-1}-1\right)\right|_{2^n+1} = 1.$$

Во втором случае

$$c_2 = \left|2^n\left(2^{2n-1}-1\right)\right|_{2^n-1} = 2^{n-1}-1 = 0\underbrace{1\ldots1}_{n-1},$$
$$c_3 = \left|2^n\left(2^{2n-1}-1\right)\right|_{2^n+1} = 2^{n-1}+1 = 01\underbrace{0\ldots0}_{n-2}1.$$

Тогда формула (2.26) сведется к

$$X_h = |w_1 + w_2 + w_3|_{2^{2n}-1}$$

с дальнейшим использованием в формуле (2.22).

Пример 16. *Рассмотрим аналогичный пример с СОК* $\{8,7,9\}$*,* $n=3$*, для числа* $X = 10 = (2,3,1)$.

Для первого варианта коэффициенты $c_2 = 0$ *и* $c_3 = 1$ *не зависят от значения* n*. Тогда смещенное значение* $X' = (2,3,|1+1|_9)$*. Отсюда*

$$x'_1 = 2 = 010_2, \quad w_1 = 101||000_2 = 40,$$
$$x'_2 = 3 = 011_2, \quad w_2 = 1||011||01_2 = 45,$$
$$x'_3 = 2 = 0010_2, \quad w_3 = 1||010||10_2 = 42.$$

Тогда $X_h = |40+45+42|_{63} = 1$*, и* $X = 2 + 2^3 \cdot 1 = 10$.

Для второго варианта коэффициенты равны $c_2 = 2^2 - 1 = 3$ *и* $c_3 = 2^3 + 1 = 5$*. Тогда смещенное значение* $X' = (2,|3+3|_7,|1+5|_9)$*. Отсюда*

$$x'_1 = 2 = 010_2, \quad w_1 = 101||1||0||0_2 = 44,$$
$$x'_2 = 6 = 110_2, \quad w_2 = 0||110||11_2 = 27,$$
$$x'_3 = 6 = 0110_2, \quad w_3 = 1||110||00_2 = 56.$$

Тогда $X_h = |44+27+56|_{63} = 1$*, и* $X = 2 + 2^3 \cdot 1 = 10$.

В первом варианте необходимо сложение только с c_3, однако при вычислении w_3 требуется n логических операторов OR, во втором варианте оператор OR один, однако сложение выполняется и по c_2 и по c_3.

В статье [72] для перевода из системы остаточных классов предложен метод опорных точек, основанный на использовании LUT-таблиц. Для заданного набора модулей $\{p_1, p_2, \ldots, p_n\}$ выбирается модуль опорной точки p_n. Для сокращения количества значений в памяти эффективнее выбирать наибольший модуль СОК. Тогда содержимое LUT будет состоять из $P_h = \prod_{i=1}^{n-1} p_i$ опорных точек, которые соответствуют числам $X = (x_1, x_2, \ldots, x_{n-1}, 0)$, кратным модулю p_n. Чтобы из произвольного числа $Y = (y_1, y_2, \ldots, y_n)$ получить число, нацело делящееся на p_n, необходимо из числа Y вычесть y_n, т.е. вычислить $x_i = |y_i - y_n|_{p_i}, i = 1, \ldots, n$.

Например, для СОК $\{7, 8, 9\}$ и исходного числа $Y = 10 = (3, 2, 1)$ опорной точкой будет значение $X\left(|3-1|_7, |2-1|_8, |1-1|_9\right) = (2, 1, 0)$.

Таким образом, по аналогии с Новой КТО II и обобщенной позиционной системой счисления для двух модулей, опорная точка будет иметь значение $X = \left|X_h \cdot \left|p_n^{-1}\right|_{P_h}\right|_{P_h} \cdot p_n$, где X_h — позиционное число, полученное из $(x_1, x_2, \ldots, x_{n-1})$ для набора модулей $\{p_1, p_2, \ldots, p_{n-1}\}$.

Пример 17. *Для числа* $Y = 228 = (4, 4, 3)$ *опорной точкой будет значение* $X\left(|4-3|_7, |4-3|_8, |3-3|_9\right) = (1, 1, 0)$. *Т.к.* $(1, 1)$ *для набора модулей* $\{7, 8\}$ *равно 1, а* $\left|9^{-1}\right|_{7\cdot 8} = 25$*, то опорная точка равна* $X = |1 \cdot 25|_{56} \cdot 9 = 225$*, что соответствует вычитанию из* $Y = 228$ *остатка* $y_n = 3$.

Можно заметить, что значение опорной точки зависит от числа $(x_1, x_2, \ldots, x_{n-1})$ с модулями $\{p_1, p_2, \ldots, p_{n-1}\}$ и диапазоном $P_h = \prod_{i=1}^{n-1} p_i$. Тогда вычисления можно заменить обращением к памяти, которое по значению $X_h = (x_1, x_2, \ldots, x_{n-1})$ будет выдавать значение $\left|X_h \cdot \left|p_n^{-1}\right|_{P_h}\right|_{P_h} \cdot p_n$. При этом обратный преобразователь на основе опорных точек может иметь иерархическую структуру, например, как показано на рисунке 2.6.

В данном случае на основе разностей остатков $|x_1 - x_3|_{p_1}$ и $|x_2 - x_3|_{p_2}$ по модулям p_1 и p_2 соответственно, используя LUT со значениями опорных точек, которые равны $\left|X_{12} \cdot \left|p_3^{-1}\right|_{p_1 p_2}\right|_{p_1 p_2} \cdot p_3$, где X_{12} — восстановленное значение по модулям p_1, p_2, и складывая значение опорной точки с остатком x_3 по модулю p_3 получим восстановленное число X_{123} по модулям p_1, p_2, p_3.

Далее аналогично находятся разности остатков по модулям p_4 и p_5 и восстановленного числа X_{123}, при этом поскольку значение X_{123} может значительно превосходить значения модулей p_4 и p_5, используются блоки прямого преобразования, которые могут быть реализованы как арифметически, так и с использова-

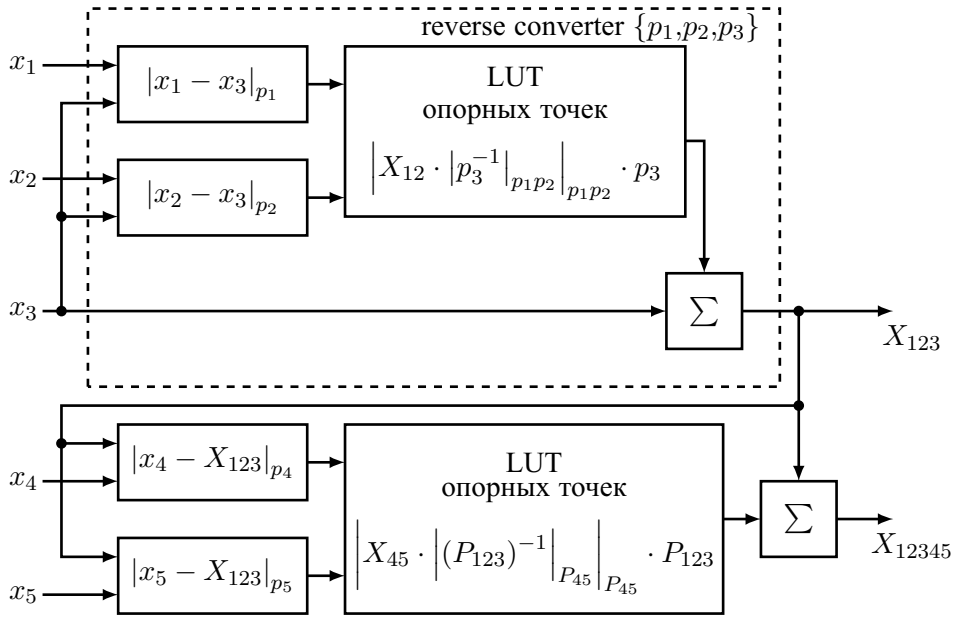

Рисунок 2.6 — Обратный преобразователь на основе опорных векторов

нием памяти. Аналогично из LUT опорных точек выбирается предвычисленное значение $\left|X_{45} \cdot \left|(P_{123})^{-1}\right|_{P_{45}}\right|_{P_{45}} \cdot P_{123}$ для $P_{123} = p_1 \cdot p_2 \cdot p_3$ и $P_{45} = p_4 \cdot p_5$, которое складывается с восстановленным ранее X_{123}.

Таким образом, варьируя количество входов LUT от 1 до $n-1$ и количество слоев обратного преобразователя, можно получить решение с лучшим временем или площадью [61].

Метод перевода чисел из системы остаточных классов в позиционную систему счисления на основе Китайской теоремы об остатках является самым распространенным в силу своей простоты и точности, однако существенным недостатком данного метода является необходимость нахождения остатка по большому модулю, равному динамическому диапазону. Обойти этот недостаток позволяет приближенный метод на основе КТО, но при недостаточной точности выполнения операций может произойти накопление погрешности округления, что приведет к ошибкам при программных вычислениях. Метод на основе обобщенной позиционной системы счисления выполняется последовательно и, хотя в нем отсутствует необходимость нахождения остатка по большому модулю, на каждом шаге происходит нахождение остатка по соответствующим модулям СОК. Функция ядра требует нахождения остатка по модулю, равному значению ядра. В случае выбора в качестве ядра степени двойки данная операция может быть выполнена максимально эффективно. Диагональная функция за счет двухступенчатого алгоритма требует больших вычислительных ресурсов и име-

ет большее время выполнения, в следствии чего мало пригодна для выполнения перевода из СОК в ПСС, при этом она находит широкое применение для сравнения чисел в СОК за счет своей монотонности.

2.2.1 Модифицированный метод обратного перевода и расширения оснований на основе ОПСС

В ряде случаев, например, для обнаружения переполнения диапазона или при обнаружении и исправлении ошибок необходимо добавление одного или нескольких дополнительных модулей. Пусть добавлен модуль p_{n+1}, тогда новый диапазон позволит отображать любое число $X < P \cdot p_{n+1}$.

Зададим систему остаточных классов с модулями $\{p_1, p_2, \ldots, p_n\}$, для которых вычисляются ортогональные базисы СОК $B_i = P_i \cdot \left|P_i^{-1}\right|_{p_i}$, $i = \overline{1,n}$. Вычисляются основания ОПСС $\hat{w}_j = \prod_{i=1}^{j-1} p_i$, $j = \overline{2,n}$, при этом $\hat{w}_1 = 1$. Ортогональные базисы СОК представляются в ОПСС как векторы значений $\hat{b}_{i,j}$, для которых $B_i = \sum_{j=1}^{n} \hat{b}_{i,j} \cdot \hat{w}_j$. При этом значения $\hat{b}_{i,j}$ образуют треугольную матрицу.

Пусть задано число $X = (x_1, x_2, \ldots, x_n)$, представленное в СОК. Переведем число X в позиционную систему счисления, а также найдем остаток от деления X по дополнительному модулю p_{n+1}, который будем обозначать $|X|_{p_{n+1}}$. Для этого $(x_1, x_2, \ldots, x_n)$ умножается на треугольную матрицу $\hat{b}_{i,j}$, т.е. находятся $U_i = \sum_{j=1}^{i} x_j \cdot \hat{b}_{j,i}$, $i = \overline{1,n}$. Затем вычисляются $\hat{x}_i = |\sigma_{i-1} + U_i|_{p_i}$ и $\sigma_i = \left\lfloor \frac{\sigma_{i-1}+U_i}{p_i} \right\rfloor$, при этом полагают, что $\sigma_0 = 0$. Можно заменить, что $\hat{x}_i$ и σ_i соотносятся как остаток от деления суммы $(\sigma_{i-1} + U_i)$ на модуль p_i и ранг суммы, т.е. значение, отражающее во сколько раз значение суммы превосходит модуль.

Затем для перевода числа из СОК в позиционную систему счисления находим значение выражения $X = \sum_{i=1}^{n} \hat{x}_i \cdot \hat{w}_i$, а для расширения оснований находим $|X|_{p_{n+1}} = |\sum_{i=1}^{n} \hat{x}_i \cdot \hat{w}_i|_{p_{n+1}}$.

На основе предложенного метода получим Алгоритм 2.14 перевода из СОК в позиционную систему счисления и расширения оснований [117].

Рассмотрим пример, иллюстрирующий работу Алгоритма 2.14.

Алгоритм 2.14. Перевод числа из СОК в ПСС и расширение оснований

Вход: $(x_1, \ldots, x_n)$

Выход: X, $|X|_{p_{n+1}}$

Данные в памяти:$\{p_1, p_2, \ldots, p_n\}$, p_{n+1}

$P_i = \frac{P}{p_i}$; $\left|P_i^{-1}\right|_{p_i}$; $B_i = P_i \cdot \left|P_i^{-1}\right|_{p_i}$

$\hat{w}_1 = 1$; $\hat{w}_j = \prod_{i=1}^{j-1} p_i$, $j = \overline{2,n}$

$B_i \xrightarrow{\text{ОПСС}} \left[\hat{b}_{i,j}\right]$, $i,j = \overline{1,n}$, $B_i = \sum_{j=1}^{n} \hat{b}_{i,j} \cdot \hat{w}_j$

1: $U_i = \sum_{j=1}^{i} x_j \cdot \hat{b}_{j,i}$, $i = \overline{1,n}$

2: $\sigma_0 = 0$, $\hat{x}_i = |\sigma_{i-1} + U_i|_{p_i}$, $\sigma_i = \left\lfloor \frac{\sigma_{i-1}+U_i}{p_i} \right\rfloor$

3: $X = \sum_{i=1}^{n} \hat{x}_i \cdot \hat{w}_i$

4: $|X|_{p_{n+1}} = \left|\sum_{i=1}^{n} \hat{x}_i \cdot \hat{w}_i\right|_{p_{n+1}}$

5: **Возвратить** X, $|X|_{p_{n+1}}$

Пример 18. *Пусть задана система остаточных классов с четырьмя основаниями* $p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11$. *Тогда* $P = \prod_{i=1}^{4} p_i = 1155$.

Вычислим $P_i = \frac{P}{p_i}$, $\left|P_i^{-1}\right|_{p_i}$ *и* $B_i = P_i \cdot \left|P_i^{-1}\right|_{p_i}$:

$$P_1 = 5 \cdot 7 \cdot 11 = 385, \quad \left|P_1^{-1}\right|_{p_1} = \left|385^{-1}\right|_3 = 1, \quad B_1 = 385,$$
$$P_2 = 3 \cdot 7 \cdot 11 = 231, \quad \left|P_2^{-1}\right|_{p_2} = \left|231^{-1}\right|_5 = 1, \quad B_2 = 231,$$
$$P_3 = 3 \cdot 5 \cdot 11 = 165, \quad \left|P_3^{-1}\right|_{p_3} = \left|165^{-1}\right|_7 = 2, \quad B_3 = 165 \cdot 2 = 330,$$
$$P_4 = 3 \cdot 5 \cdot 7 = 105, \quad \left|P_4^{-1}\right|_{p_4} = \left|105^{-1}\right|_{11} = 2, \quad B_4 = 105 \cdot 2 = 210.$$

Вычислим $\hat{w}_j = \prod_{i=1}^{j-1} p_i$:

$$\hat{w}_1 = 1, \hat{w}_2 = 3, \hat{w}_3 = 3 \cdot 5 = 15, \hat{w}_4 = 3 \cdot 5 \cdot 7 = 105.$$

Вычислим представление B_i *в ОПСС:*

$$B_1 \xrightarrow{\text{ОПСС}} \hat{B}_1 = \left[\hat{b}_{1,1},\hat{b}_{1,2},\hat{b}_{1,3},\hat{b}_{1,4}\right] = [1,3,4,3],\ 1 + 3 \cdot 3 + 4 \cdot 15 + 3 \cdot 105 = 385,$$
$$B_2 \xrightarrow{\text{ОПСС}} \hat{B}_2 = \left[\hat{b}_{2,1},\hat{b}_{2,2},\hat{b}_{2,3},\hat{b}_{2,4}\right] = [0,2,1,2],\ 0 + 2 \cdot 3 + 1 \cdot 15 + 2 \cdot 105 = 231,$$
$$B_3 \xrightarrow{\text{ОПСС}} \hat{B}_3 = \left[\hat{b}_{3,1},\hat{b}_{3,2},\hat{b}_{3,3},\hat{b}_{3,4}\right] = [0,0,1,3],\ 0 + 0 \cdot 3 + 1 \cdot 15 + 3 \cdot 105 = 330,$$
$$B_4 \xrightarrow{\text{ОПСС}} \hat{B}_4 = \left[\hat{b}_{4,1},\hat{b}_{4,2},\hat{b}_{4,3},\hat{b}_{4,4}\right] = [0,0,0,2],\ 0 + 0 \cdot 3 + 0 \cdot 15 + 2 \cdot 105 = 210.$$

Пусть задано число $X \xrightarrow{\text{СОК}} (1,2,3,4)$, *тогда вычислим* $U_i = \sum_{j=1}^{i} x_j \cdot \hat{b}_{j,i}$:

$$U_1 = x_1 \cdot \hat{b}_{1,1} = 1 \cdot 1 = 1,$$
$$U_2 = x_1 \cdot \hat{b}_{1,2} + x_2 \cdot \hat{b}_{2,2} = 1 \cdot 3 + 2 \cdot 2 = 7,$$
$$U_3 = x_1 \cdot \hat{b}_{1,3} + x_2 \cdot \hat{b}_{2,3} + x_3 \cdot \hat{b}_{3,3} = 1 \cdot 4 + 2 \cdot 1 + 3 \cdot 1 = 9,$$
$$U_4 = x_1 \cdot \hat{b}_{1,4} + x_2 \cdot \hat{b}_{2,4} + x_3 \cdot \hat{b}_{3,4} + x_4 \cdot \hat{b}_{4,4} = 1 \cdot 3 + 2 \cdot 2 + 3 \cdot 3 + 4 \cdot 2 = 24.$$

Вычислим $\hat{x}_i = |\sigma_{i-1} + U_i|_{p_i}$ *и* $\sigma_i = \left\lfloor \frac{\sigma_{i-1}+U_i}{p_i} \right\rfloor$:

$$\hat{x}_1 = |U_1|_{p_1} = |1|_3 = 1, \qquad \sigma_1 = \left\lfloor \frac{U_1}{p_1} \right\rfloor = \left\lfloor \frac{1}{3} \right\rfloor = 0,$$
$$\hat{x}_2 = |\sigma_1 + U_2|_{p_2} = |0 + 7|_5 = 2, \qquad \sigma_2 = \left\lfloor \frac{\sigma_1 + U_2}{p_2} \right\rfloor = \left\lfloor \frac{7}{5} \right\rfloor = 1,$$
$$\hat{x}_3 = |\sigma_2 + U_3|_{p_3} = |1 + 9|_7 = 3, \qquad \sigma_3 = \left\lfloor \frac{\sigma_2 + U_3}{p_3} \right\rfloor = \left\lfloor \frac{1+9}{7} \right\rfloor = 1,$$
$$\hat{x}_4 = |\sigma_3 + U_4|_{p_4} = |1 + 24|_1 \, 1 = 3.$$

Отсюда получим

$$X = \sum_{i=1}^{n} \hat{x}_i \cdot \hat{w}_i = 1 \cdot 1 + 2 \cdot 3 + 3 \cdot 15 + 3 \cdot 105 = 367.$$

Проверим полученное значение, $|367|_3 = 1$, $|367|_5 = 2$, $|367|_7 = 3$ *и* $|367|_{11} = 4$.

Пусть необходимо расширить основания на $p_5 = 13$. *Тогда*

$$|X|_{p_{n+1}} = \left| \sum_{i=1}^{n} \hat{x}_i \cdot \hat{w}_i \right|_{p_{n+1}} = |1 \cdot 1 + 2 \cdot 3 + 3 \cdot 15 + 3 \cdot 105|_{13} = |367|_{13} = 3.$$

Предложенный метод (Алгоритм 2.14) наравне с приближенным методом на основе КТО показал свою эффективность за счет отсутствия ресурсоемких операций нахождения остатка от деления на большой модуль, что делает их наиболее применимыми для задач цифровой обработки сигналов.

2.3 Исследование методов определения знака и сравнения чисел в системе остаточных классов

При использовании непозиционных систем счисления, таких как система остаточных классов, выполнение высокопроизводительных вычислений возможно за счет отсутствия переносов между разрядами, однако во многих прикладных задачах возникает необходимость сравнения чисел и определения знака числа, которые в СОК относятся к вычислительно сложным операциям. Данная операция является базовой при реализации большого числа алгоритмов защиты информации [6; 73; 79], цифровой обработки сигналов [18; 98], систем беспроводной связи [69], облачных вычислений [24; 83–85] и т.д.

В силу непозиционной природы СОК немодульные операции, такие как сравнение чисел, определение знака числа и переполнения динамического диапазона, относятся к вычислительно сложным операциям. В позиционной системе счисления существуют простые алгоритмы сравнения чисел, которые сводятся к их поразрядному сравнению. В СОК простых алгоритмов сравнения чисел нет [80].

Реализация алгоритма сравнения чисел в СОК состоит из двух этапов. Первый этап — вычисление позиционной характеристики (ПХ) модулярных чисел $X = (x_1, \ldots, x_n)$ и $Y = (y_1, \ldots, y_n)$. Второй этап — сравнение позиционных характеристик ПХ(X) и ПХ(Y) модулярных чисел в позиционной системе счисления. На рисунке 2.7 представлена общая схема сравнения чисел $X = (x_1, \ldots, x_n)$ и $Y = (y_1, \ldots, y_n)$ в СОК.

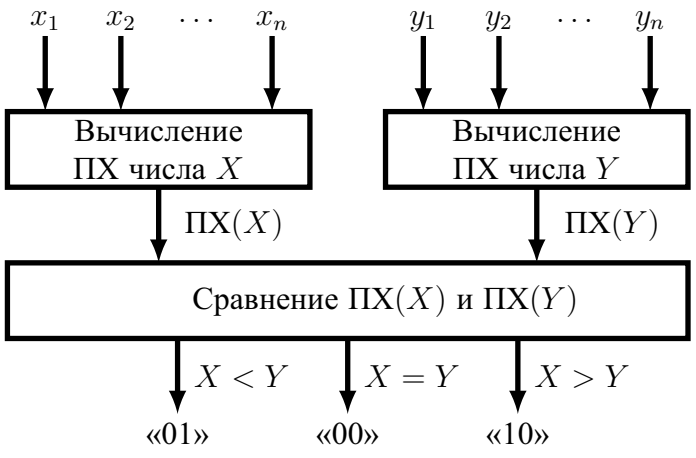

Рисунок 2.7 — Общая схема сравнения чисел в СОК

В качестве ПХ модулярного числа может выступать его представление в ПСС. Для перевода числа из СОК в ПСС можно использовать один из алгоритмов, рассмотренных в Параграфе 2.2, например, основанный на Китайской теореме об остатках, обобщенной позиционной системе счисления [12], рекурсивном алгоритме сдваивания чисел (nCRT) [88], и их модификаций.

Большая вычислительная сложность алгоритмов перевода числа из СОК в ПСС сподвигла исследователей на поиск их аппроксимаций. С целью уменьшения вычислительной сложности операции сравнения чисел в СОК исследователи предложили в качестве ПХ модулярного числа использовать следующие функции: диагональная функция [22], функция ядра [16], фактор-функция [74], монотонная функция Pirlo [66] и др. Предлагаемые алгоритмы вычисления ПХ позволяют уменьшить вычислительную сложность за счет уменьшения размерности операндов при выполнении операции деления с остатком.

Показано [75], что оптимальным в случае перевода чисел в позиционную систему счисления является подход, основанный на приближенном методе на основе Китайской теоремы об остатках, так как он позволяет заменить операцию деления с остатком на операцию взятия старших бит числа. Далее будет представлена оптимизация приближенного метода для сравнения чисел в СОК, заключающаяся в уменьшении количества операций деления с остатком и уточнении разрядности операндов, требуемой для корректной работы алгоритма.

Некоторые приложения в СОК требуют использования отрицательных чисел. В системе остаточных классов с модулями $\{p_1, p_2, \ldots, p_n\}$ и динамическим диапазоном $P = \prod_{i=1}^{n} p_i$, диапазон может быть каким-либо образом поделен на поддиапазоны представления отрицательных и неотрицательных чисел. Например, рассмотрим число X, удовлетворяющее следующим соотношениям:

$$-\frac{P-1}{2} \leq X \leq \frac{P-1}{2}, \text{ если } P \text{ нечётное}, \tag{2.27}$$

$$-\frac{P}{2} \leq X \leq \frac{P}{2} - 1, \text{ если } P \text{ чётное}. \tag{2.28}$$

Тогда, согласно [55], если $X = (x_1, x_2, \ldots, x_n)$, то отрицательное число $-X = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$, где $\overline{x_i}$ является дополнением x_i до модуля p_i. Например, для СОК $\{3, 5, 7\}$ и числа $X = 17 = (2, 2, 3)$ получим $-X = (3-2, 5-2, 7-3) = (1, 3, 4)$. Для перехода от восстановленного числа к отрицательной форме необходимо отнять значение динамического диапазона,

т.е. $(1,3,4) = 88 \equiv 88 - 105 = -17$. Тогда для СОК $\{3,5,7\}$ числа будут распределены следующим образом: $\underrightarrow{0,1,\ldots,52,}\ \underrightarrow{-52,-51,\ldots,-1}$.

Ниже будут рассмотрены различные методы определения знака и сравнения чисел, описанные в литературе. Отметим, что в большинстве методов задача сравнения решается через вычитание сравниваемых чисел и определение знака результата, поэтому изучать данные задачи надо комплексно.

Одним из подходов к определению знака является перевод числа из СОК в ПСС и сравнение с серединой диапазона. Так, согласно [55] для перевода числа из СОК в ПСС используется стандартное восстановление с помощью КТО, которое можно описать формулой

$$X = \left| \sum_{i=1}^{n} P_i \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_P, \tag{2.29}$$

где $P_i = \frac{P}{p_i}$, а $\left| P_i^{-1} \right|_{p_i}$ — мультипликативная инверсия P_i. Рассмотрим пример перевода числа из СОК в ПСС по формуле (2.29) и сравнения чисел.

Пример 19. *Пусть дана СОК* $\{3,5,7\}$ *и числа* $X = (2,2,3)$, $Y = (1,3,4)$, *которые равны соответственно* 17 *и* 88. *Динамический диапазон данной системы остаточных классов равен* $P = 3 \cdot 5 \cdot 7 = 105$. *Вычислим* P_i:

$$P_1 = \frac{3 \cdot 5 \cdot 7}{3} = 35, P_2 = \frac{3 \cdot 5 \cdot 7}{5} = 21, P_3 = \frac{3 \cdot 5 \cdot 7}{7} = 15.$$

Чтобы вычислить мультипликативную инверсию P_i, *нужно найти такое* x, *которое удовлетворяет сравнению* $x \cdot P_i \equiv 1 \bmod p_i$. *Таким образом,* $\left| P_1^{-1} \right|_3 = 2$, $\left| P_2^{-1} \right|_5 = 1$, $\left| P_3^{-1} \right|_7 = 1$. *Для проверки* $\left| P_1^{-1} \right|_3 = 2$, *подставим* 2 *в выражение* $2 \cdot 35 \equiv 1 \bmod 3$. *Сравнение выполняется, т.е. мультипликативные инверсии найдены правильно. Таким образом, все необходимые для вычисления* (2.29) *данные получены, найдем значение первого числа*

$$X = |35 \cdot 2 \cdot 2 + 21 \cdot 2 \cdot 1 + 15 \cdot 3 \cdot 1|_{105} = |227|_{105} = 17$$

и второго числа

$$Y = |35 \cdot 1 \cdot 2 + 21 \cdot 3 \cdot 1 + 15 \cdot 4 \cdot 1|_{105} = |193|_{105} = 88.$$

Поскольку $17 < 88$, *то* $X < Y$.

Таким образом, получим Алгоритм 2.15, основанный на формуле (2.29).

Алгоритм 2.15. Сравнение чисел в СОК с использованием КТО

Вход: $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$

Выход: $X > Y$ — «10», $X < Y$ — «01», $X = Y$ — «00»

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$,
$P = \prod_{i=1}^{n} p_i$, $P_i = \frac{P}{p_i}$, $w_i = \left|P_i^{-1}\right|_{p_i}$ для $i = \overline{1,n}$

1: $S_X = 0, S_Y = 0$
2: **Цикл** $i = 1$ до n **выполнять**
3: $\quad S_X = S_X + w_i \cdot P_i \cdot x_i$
4: $\quad S_Y = S_Y + w_i \cdot P_i \cdot y_i$
5: **Конец цикла**
6: $S_X = S_X \bmod P$
7: $S_Y = S_Y \bmod P$
8: **Если** $S_X > S_Y$ **тогда**
9: $\quad$ **Возвратить** «10»
10: **иначе если** $S_X < S_Y$
11: $\quad$ **Возвратить** «01»
12: **иначе**
13: $\quad$ **Возвратить** «00»
14: **Конец условия**

Учитывая вычислительную сложность Алгоритма 2.15 получения остатка от деления на большое число P, с целью повышения производительности исследователями был предложен альтернативный подход, основанный на обобщенной позиционной системе счисления.

ОПСС за счет своих свойств позволяет сравнивать два числа без прямого восстановления самого числа. Число в ОПСС задается кортежем $[a_1, a_2, \ldots, a_n]$, а основаниями системы являются p_1, p_1p_2, $p_1p_2p_3$, $\ldots$, $p_1p_2 \ldots p_{n-1}$, где $p_1, p_2, \ldots, p_n$ — модули СОК. Связь между двоичной системой счисления и ОПСС определяется следующей формулой

$$X = a_1 + a_2 \cdot p_1 + a_3 \cdot p_1 \cdot p_2 + \ldots + a_n \cdot p_1 \cdot p_2 \cdot \ldots \cdot p_{n-1}.$$

Так как ОПСС является позиционной системой счисления, то сравнение чисел равносильно сравнению двух кортежей $[a_1, a_2, \ldots, a_n]$ и $[b_1, b_2, \ldots, b_n]$. Для перевода числа $X = (x_1, x_2, \ldots, x_n)$ из СОК в ОПСС $X = [a_1, a_2, \ldots, a_n]$ исполь-

зуется следующий подход

$$
\begin{aligned}
&a_1 = x_1,\\
&a_2 = \left|(x_2 - a_1)\cdot p_1^{-1}\right|_{p_2},\\
&a_3 = \left|(x_3 - a_1 - a_2\cdot p_1)\cdot p_1^{-1}\cdot p_2^{-1}\right|_{p_3},\\
&\ldots\\
&a_n = \left|(x_n - a_1 - a_2\cdot p_1 - \ldots - a_{n-1}\cdot p_1\cdot p_2\ldots p_{n-2})\cdot p_1^{-1}\cdot\ldots\cdot p_{n-1}^{-1}\right|_{p_n}.
\end{aligned}
$$

Эффективная реализация сравнения чисел с использованием ОПСС из работы [35] представлена Алгоритмом 2.16.

Таким образом, использование ОПСС позволяет уйти от вычисления остатка от деления на большое число P, но приводит к использованию большего количества модульных операций по модулям СОК.

Для исключения операции деления с остатком на большое простое число в КТО в статье [86] предложен приближенный метод, основанный на отображении, переводящем $[0,P)$ в $[0,2)$. Для этого перепишем (2.29) в виде

$$X = \sum_{i=1}^{n} P_i\cdot x_i\cdot\left|P_i^{-1}\right|_{p_i} - P\cdot r_X, \tag{2.30}$$

где r_X — некоторое неотрицательное целое число, называемое рангом числа X. Разделив (2.30) на $\frac{P}{2}$, получим

$$X_s = \left(\frac{2}{P}\right)\cdot X = \sum_{i=1}^{n}\frac{2}{p_i}\cdot x_i\cdot\left|P_i^{-1}\right|_{p_i} - 2r_X. \tag{2.31}$$

Таким образом, из (2.31), X_s может быть вычислен как сумма дробных чисел с отбрасыванием кратной 2 целой части результата. Это может быть получено довольно тривиально, поскольку вычисления выполняются в двоичном виде. В этом случае старший бит числа по формуле (2.31) будет определять знак числа. Проиллюстрируем это на примере.

Пример 20. *Пусть в СОК* $\{3,5,7\}$ *дано число* $X = (2,2,3) = 17$*, тогда по формуле* (2.31) *получим*

$$X_s = \left|\frac{2}{3}\cdot 2\cdot 2 + \frac{2}{5}\cdot 2\cdot 1 + \frac{2}{7}\cdot 3\cdot 1\right|_2 = \left|4\frac{34}{105}\right|_2 = \frac{34}{105}.$$

Значит, число положительное.

Алгоритм 2.16. Сравнение чисел в СОК с использованием ОПСС

Вход: $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$

Выход: $X > Y$ — «10», $X < Y$ — «01», $X = Y$ — «00»

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$,
$P_i = \prod_{j=1}^{i} p_j, i = \overline{1, n-2}$

1: $a_1 = x_1$, $b_1 = y_1$
2: $t_X = |x_2 - a_1|_{p_2}$, $t_Y = |y_2 - b_1|_{p_2}$
3: $a_2 = \left|\frac{t_X}{P_1}\right|_{p_2}$, $b_2 = \left|\frac{t_Y}{P_1}\right|_{p_2}$
4: $RX = x_1$, $RY = y_1$
5: **Цикл** $i = 3$ до n **выполнять**
6: $\quad RX = \left|a_{i-1}\,|P_{i-2}|_{p_i} + RX\right|_{p_i}$
7: $\quad RY = \left|b_{i-1}\,|P_{i-2}|_{p_i} + RY\right|_{p_i}$
8: $\quad t_X = |x_i - RX|_{p_i}$
9: $\quad t_Y = |y_i - RY|_{p_i}$
10: $\quad a_i = \left|\frac{t_X}{P_{i-1}}\right|_{p_i}, b_i = \left|\frac{t_Y}{P_{i-1}}\right|_{p_i}$
11: **Конец цикла**
12: **Цикл** $i = n$ до 1 **выполнять**
13: $\quad$ **Если** $a_i > b_i$ **тогда**
14: $\quad\quad$ **Возвратить** «10»
15: $\quad$ **иначе если** $a_i < b_i$
16: $\quad\quad$ **Возвратить** «01»
17: $\quad$ **Конец условия**
18: **Конец цикла**
19: **Возвратить** «00»

Заметим, что данный метод работает, однако слагаемые в данном выражении редко могут быть представлены в виде конечной дроби. И для представления в виде десятичной (двоичной) дроби каждое слагаемое должно быть определенным образом округлено. Если для каждого слагаемого суммы в формуле (2.31) выделить $t + 1$ бит, 1 на целую часть и t на дробную и усекать оставшиеся биты, то ошибка в каждом слагаемом будет удовлетворять неравенству $0 \leq e_i < 2^{-t}$. И поскольку таких слагаемых n, то максимальная ошибка при усечении (2.31) будет $e < n2^{-t}$.

Числа X распределены равномерно на интервале $[0, P)$, поэтому расстояние между двумя соседними числами X_s равно $2/P$. Кроме того, интервал между наибольшим положительным числом и 1 равен $2/P$ для чётного P и $1/P$ для нечётного P. Докажем это. При нечётном P наибольшее положительное число равно $X = \frac{P-1}{2}$, а при дробном представлении $X_s = \frac{2}{P} \cdot \frac{P-1}{2} = \frac{P-1}{P}$. Тогда расстояние между $1 = \frac{P}{P}$ и $X_s = \frac{P-1}{P}$ равно

$$\frac{P}{P} - \frac{P-1}{P} = \frac{1}{P}.$$

Для чётного P проверка проводится аналогично.

Таким образом, для того, чтобы усеченное значение X_s соотносилось с точным значением X_s, ошибка должна удовлетворять следующим соотношениям:

$$n \cdot 2^{-t} \leq \frac{2}{P} \text{ для чётного } P,$$
$$n \cdot 2^{-t} \leq \frac{1}{P} \text{ для нечётного } P,$$

или, если выразить t

$$t \geq \lceil \log_2 P \cdot n \rceil - 1 \text{ для чётного } P, \tag{2.32}$$
$$t \geq \lceil \log_2 P \cdot n \rceil \text{ для нечётного } P. \tag{2.33}$$

Хотя дробное представление и требует примерно на $2\lceil \log_2 n \rceil$ бит больше, простота и скорость выполнения операций компенсируют эту избыточность. Данный способ вычисления X_s может быть относительно просто реализован с помощью табличных методов, на вход которых подается остаток $|X|_{p_i}$, а на выход поступает усеченное значение, далее усеченные значения складываются по модулю 2, что легко реализуется аппаратно.

Рассмотрим численный пример.

Пример 21. *Пусть в СОК* $\{3, 5, 7\}$ *заданы два числа* $1 = (1, 1, 1)$ *и* $104 = (2, 4, 6)$. *Для данной системы* $t \geq \lceil \log_2 105 \cdot 3 \rceil = 9$. *Рассмотрим первое число по слагаемым.*

$$\left|\frac{2}{3} \cdot 1 \cdot 2\right|_2 = \left|\frac{4}{3}\right|_2 = 1.010101010101010\ldots \approx 1.010101010,$$
$$\left|\frac{2}{5} \cdot 1 \cdot 1\right|_2 = \left|\frac{2}{5}\right|_2 = 0.011001100110011\ldots \approx 0.011001100,$$
$$\left|\frac{2}{7} \cdot 1 \cdot 1\right|_2 = \left|\frac{2}{7}\right|_2 = 0.010010010010010\ldots \approx 0.010010010.$$

Просуммируем по модулю 2 вычисленные слагаемые и получим 0.000001000.

Рассмотрим второе число.

$$\left|\frac{2}{3}\cdot 2\cdot 2\right|_2 = \left|\frac{8}{3}\right|_2 = \left|\frac{2}{3}\right|_2 = 0.101010101010101\ldots \approx 0.101010101,$$
$$\left|\frac{2}{5}\cdot 1\cdot 4\right|_2 = \left|\frac{8}{5}\right|_2 = 1.100110011001100\ldots \approx 1.100110011,$$
$$\left|\frac{2}{7}\cdot 1\cdot 6\right|_2 = \left|\frac{12}{7}\right|_2 = 1.101101101101101\ldots \approx 1.101101101.$$

Просуммируем по модулю 2 вычисленные слагаемые и получим 1.111110101. *Сравнив полученные значения, можно сделать вывод, что* $(1,1,1) < (2,4,6)$.

Данный метод эффективнее, чем восстановление числа с помощью стандартной КТО, однако возникает вопрос о достаточности или избыточности оценок точности полученных в формулах (2.32)–(2.33).

Стоит заметить, что рассмотренный подход к вычислению позиционной характеристики может быть реализован в целых числах посредством следующей формулы

$$V(X) = \left|\sum_{i=1}^{n}\left\lceil\frac{2}{p_i}\left|\left|P_i^{-1}\right|_{p_i}\cdot x_i\right|_{p_i}\right\rceil_{2^{-N}}\right|_2,$$

где $\lceil x\rceil_{2^{-N}} = \left\lceil 2^N x\right\rceil / 2^N$, N — размерность операнд, необходимая для однозначного получения позиционной характеристики.

Таким образом, на основе приближенного метода из работы [86] может быть получен Алгоритм 2.17 сравнения чисел в СОК.

С целью уменьшения количества операций в приближенном методе в [75] предложено использовать следующую формулу:

$$F(X) = \left|\sum_{i=1}^{n} W_i x_i\right|_1,$$

где $W_i = \left\lceil\frac{2^N\left|P_i^{-1}\right|_{p_i}}{p_i}\right\rceil / 2^N$, $|x|_1$ — дробная часть числа x, $N = \lceil\log_2(P\rho)\rceil$ и $\rho = -n + \sum_{i=1}^{n} p_i$.

Преимущество Алгоритма 2.18 относительно Алгоритма 2.17 состоит в том, что он не требует дополнительных операции округления вверх, однако при этом увеличились размеры операндов.

Алгоритм 2.17. Сравнение чисел в СОК на основе приближенного метода [86]

Вход: $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$

Выход: $X > Y$ — «10», $X < Y$ — «01», $X = Y$ — «00»

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$,
$w_i = \left|P_i^{-1}\right|_{p_i}$ для всех $i = \overline{1,n}$, $N = \lceil \log_2 (nP) \rceil$

1: $S_X = 0$, $S_Y = 0$
2: **Цикл** $i = 1$ до n **выполнять**
3: $\quad S_X = S_X + \left\lceil 2^{N+1} \cdot w_i \cdot x_i / p_i \right\rceil / 2^N$
4: $\quad S_Y = S_Y + \left\lceil 2^{N+1} \cdot w_i \cdot y_i / p_i \right\rceil / 2^N$
5: **Конец цикла**
6: $S_X = S_X \bmod 2$
7: $S_Y = S_Y \bmod 2$
8: **Если** $S_X > S_Y$ **тогда**
9: $\quad$ **Возвратить** «10»
10: **иначе если** $S_X < S_Y$
11: $\quad$ **Возвратить** «01»
12: **иначе**
13: $\quad$ **Возвратить** «00»
14: **Конец условия**

С целью уменьшения вычислительной сложности алгоритма сравнения чисел в статье [22] предложено использовать монотонную диагональную функцию.

Отличным от вышеизложенных методов сравнения чисел является метод на основе специальной диагональной функции, которая определяется как сумма коэффициентов P_i и называется метод суммы коэффициентов (Sum of Quotients Technique, SQT). Описание SQT можно найти, например, в [22]. Диагональная функция является монотонно возрастающей функцией, и на её основе возможно сравнение чисел.

Диагональная функция имеет вид

$$D(X) = \left\lfloor \frac{X}{p_1} \right\rfloor + \left\lfloor \frac{X}{p_2} \right\rfloor + \ldots + \left\lfloor \frac{X}{p_n} \right\rfloor. \tag{2.34}$$

Однако формула (2.34) является малопригодной на практике. В связи с этим была предложена аппроксимация диагональной функции [22]

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

Алгоритм 2.18. Сравнение чисел в СОК на основе приближенного метода [75]

Вход: $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$

Выход: $X > Y$ — «10», $X < Y$ — «01», $X = Y$ — «00»

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$

$W_i = \left\lceil \frac{2^N \left|P_i^{-1}\right|_{p_i}}{p_i} \right\rceil / 2^N$ для всех $i = \overline{1,n}$, $N = \lceil \log_2(\rho P) \rceil$

1: $S_X = 0$, $S_Y = 0$
2: **Цикл** $i = 1$ до n **выполнять**
3: $\quad S_X = (S_X + W_i \cdot x_i) \bmod 1$
4: $\quad S_Y = (S_Y + W_i \cdot y_i) \bmod 1$
5: **Конец цикла**
6: **Если** $S_X > S_Y$ **тогда**
7: $\quad$ **Возвратить** «10»
8: **иначе если** $S_X < S_Y$
9: $\quad$ **Возвратить** «01»
10: **иначе**
11: $\quad$ **Возвратить** «00»
12: **Конец условия**

где $k_i^* = \left|-p_i^{-1}\right|_{SQ}$, где $i = \overline{1,n}$, $SQ = P_1 + P_2 + \ldots + P_n$.

Так как диагональная функция (2.34) является монотонно возрастающей, то она может быть использована для сравнения чисел, т.е. если $D(X) < D(Y)$, то $X < Y$. Однако, возможны случаи, когда $D(X) = D(Y)$, при этом $X < Y$, когда $x_i < y_i$, $i = \overline{1,n}$.

Рассмотрим пример сравнения чисел.

Пример 22. *Возьмем для примера использовавшиеся ранее числа* $X = (2,2,3) = 17$ *и* $Y = (1,3,4) = 88$ *в СОК* $\{3,5,7\}$. *Для начала вычислим значения коэффициентов*

$$SQ = 35 + 21 + 15 = 71,$$
$$k_1^* = \left|-p_1^{-1}\right|_{71} = \left|-3^{-1}\right|_{71} = 47,$$
$$k_2^* = \left|-p_2^{-1}\right|_{71} = \left|-5^{-1}\right|_{71} = 14,$$
$$k_3^* = \left|-p_3^{-1}\right|_{71} = \left|-7^{-1}\right|_{71} = 10.$$

Найдем значения диагональной функции:

$$D(X) = |2 \cdot 47 + 2 \cdot 14 + 3 \cdot 10|_{71} = 10,$$
$$D(Y) = |1 \cdot 47 + 3 \cdot 14 + 4 \cdot 10|_{71} = 58.$$

Т.к. $D(X) < D(Y)$*, то* $X < Y$.

Реализация сравнения чисел в СОК на основе диагональной функции представлена Алгоритмом 2.19.

Алгоритм 2.19. Сравнение чисел в СОК на основе диагональной функции

Вход: $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$

Выход: $X > Y$ — «10», $X < Y$ — «01», $X = Y$ — «00»

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$

$SQ = \sum_{i=1}^{n} P_i$, $k_i^* = \left|-p_i^{-1}\right|_{SQ}$ для всех $i = \overline{1, n}$

$S_X = 0$, $S_Y = 0$
Цикл $i = 1$ до n **выполнять**
    $S_X = S_X + k_i^* \cdot x_i$
    $S_Y = S_Y + k_i^* \cdot y_i$
Конец цикла
$S_X = S_X \bmod SQ$
$S_Y = S_Y \bmod SQ$
Если $S_X > S_Y$ **тогда**
    Возвратить «10»
иначе если $S_X < S_Y$
    Возвратить «01»
иначе
    Если $x_1 > y_1$ **тогда**
        Возвратить «10»
    иначе если $x_1 < y_1$
        Возвратить «01»
    иначе
        Возвратить «00»
    Конец условия
Конец условия

Обобщив результат, полученный в [22], исследовательская группа Pirlo в работе [66] предложила использовать минимальную функцию ядра Акушского

без критических ядер. Данный подход является аналогичным методу диагональной функции. Функция Pirlo имеет следующий вид

$$\text{Pir}(X) = \left\lfloor \frac{X}{p_n} \right\rfloor. \tag{2.35}$$

Формула (2.35) является мало пригодной на практике, в связи чем была предложена аппроксимация функции Pirlo:

$$\text{Pir}(X) = \left| \sum_{i=1}^{n} k_i^{**} \cdot x_i \right|_{P_n},$$

где $k_i^{**} = \left\lfloor \frac{\left|P_i^{-1}\right|_{p_i} P_i}{p_n} \right\rfloor$.

Так как функция Pirlo (2.35) является монотонно возрастающей, то она может быть использована для сравнения чисел, т.е. если $\text{Pir}(X) < \text{Pir}(Y)$, то $X < Y$. Однако возможны случаи, когда $\text{Pir}(X) = \text{Pir}(Y)$, и в этом случае $X < Y$, когда $x_n < y_n$.

Рассмотрим пример сравнения чисел.

Пример 23. *Возьмем ранее использовавшиеся числа* $X = (2, 2, 3) = 17$ *и* $Y = (1, 3, 4) = 88$ *в СОК* $\{3, 5, 7\}$. *Для начала вычислим значения коэффициентов*

$$\begin{aligned}
&P_3 = 15,\\
&k_1^{**} = \left\lfloor \frac{\left|P_1^{-1}\right|_{p_1} P_1}{p_3} \right\rfloor = \left\lfloor \frac{2 \cdot 35}{7} \right\rfloor = 10,\\
&k_2^{**} = \left\lfloor \frac{\left|P_2^{-1}\right|_{p_2} P_2}{p_3} \right\rfloor = \left\lfloor \frac{1 \cdot 21}{7} \right\rfloor = 3,\\
&k_3^{**} = \left\lfloor \frac{\left|P_3^{-1}\right|_{p_3} P_3}{p_3} \right\rfloor = \left\lfloor \frac{1 \cdot 15}{7} \right\rfloor = 2.
\end{aligned}$$

Найдем значения функции Pirlo

$$\begin{aligned}
&\text{Pir}(X) = |2 \cdot 10 + \ 2 \cdot 3 + 3 \cdot 2|_{15} = 2,\\
&\text{Pir}(Y) = |1 \cdot 10 + \ 3 \cdot 3 + 4 \cdot 2|_{15} = 12.
\end{aligned}$$

и поскольку $\text{Pir}(X) < \text{Pir}(Y)$, *то* $X < Y$.

Как показано в работе [48], метод сравнения на основе функции Pirlo проигрывает методу на основе Китайской теореме об остатках, так как требует дополнительных сравнений чисел.

С целью оптимизации алгоритма сравнения чисел иногда целесообразно использовать на втором этапе вместо алгоритма сравнения алгоритм определения знака числа.

В СОК с отрицательными числами для определения знака числа необходимо сравнить это число с серединой диапазона. Следует также обратить внимание, что в данном случае отрицательные числа представлены положительными, и для сравнения чисел сначала нужно определить их знак.

Существует ряд подходов к определению знака чисел в СОК: восстановление числа с помощью КТО, использование ОПСС и другие. Проблемой КТО является необходимость нахождения остатка по большому модулю P, что является довольно трудоемкой задачей, и последующего сравнения с константой. Введем функцию знака $S(X)$ для системы с чётным динамическим диапазоном P согласно выражению (2.28)

$$S(X) = \begin{cases} 0, \text{если } 0 \leq X < \frac{P}{2}, \\ 1, \text{если } \frac{P}{2} \leq X < P. \end{cases} \tag{2.36}$$

Использование приближенного метода (2.31) позволяет упростить формулу (2.36)

$$S(X) = \begin{cases} 0, \text{если } 0 \leq X_s < 1, \\ 1, \text{если } 1 \leq X_s < 2. \end{cases} .$$

Для СОК с нечётным диапазоном граница отрицательных чисел определяется в соответствии с выражением (2.27), т.е. число $0 \leq X < \frac{P+1}{2}$ — положительное, иначе — отрицательное.

Стоит отметить, что для корректного сравнения на основе алгоритма определения знака числа требуется удвоение диапазонов СОК, что ведет к дополнительным вычислительным нагрузкам при обработке данных.

2.3.1 Построение монотонной функции ядра для сравнения чисел в СОК

Введем модификацию функции ядра И.Я. Акушского с заданными свойствами [102]. Пусть задана функция ядра

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

Числа w_i, называемые весами, в данной формуле могут быть в определенном смысле произвольными. Именно они определяют каждую конкретную функцию ядра и могут меняться в зависимости от задачи. Базовым свойством функции ядра является то, что ее максимальный диапазон может меняться и может быть значительно меньше числа P в зависимости от выбора весов. Например, в качестве $C(P)$ можно использовать некоторое произвольное значение C_P, обладающее необходимыми для решения конкретной задачи свойствами. Значения функции ядра $C(X)$, заданной весами $w_1, w_2, \ldots, w_n$, при условии $0 \leq\ C(X) < C_P$, $X \in [0,P)$, можно вычислить с использованием формулы

$$C(X) = \left| \sum_{i=1}^{n} c_i x_i \right|_{C_P},$$

где $c_i = C(B_i)$. Однако в общем случае функция ядра не обладает монотонностью, необходимой для сравнения чисел.

Для построения функции ядра с модулем специального вида $C_P = R(N)$ и неотрицательными коэффициентами введем Алгоритм 2.20, результатом работы которого будут коэффициенты $w_1, w_2, \ldots, w_n$ функции ядра.

Выполнение условий

$$C(p_k) = \sum_{i=1}^{k} w_i \cdot \left\lfloor \frac{p_k}{p_i} \right\rfloor \geq 0 \text{ и } \sum_{i=1}^{n} \left(\left\lfloor \frac{p_k}{p_i} \right\rfloor + 1 \right) \cdot w_i - w_k > 0,$$

для всех $k = 1, 2, \ldots, n$ означает соответственно отсутствие критических ядер снизу и сверху. В данном случае взяты неотрицательные коэффициенты $w_1, w_2, \cdots, w_n$ и $w_k \neq 0$ — первый не равный нулю коэффициент.

Тогда функция ядра с заданными свойствами будет иметь вид

$$C(X) = \left| \sum_{i=1}^{n} C(B_i) \cdot x_i \right|_{C_P}, \tag{2.37}$$

Алгоритм 2.20. Подбор параметров функции ядра специального вида для заданного набора модулей

Вход: $\{p_1, p_2, \ldots, p_n\}$

Выход: $w_1, w_2, \ldots, w_n$

1: $N = \lceil \log_2 P_n \rceil$
2: $w_i^* = \left| R(N) \cdot P_i^{-1} \right|_{p_i}$, для $i = \overline{1,n}$, $C_P^* = P \cdot \sum_{i=1}^{n} \frac{w_i^*}{p_i}$, где $R(N) = 2^N$
3: $Q = \frac{R(N) - C_P^*}{P}$
4: **Если** $Q < 0$ **тогда**
5: $N = N + 1$ и перейти к строке 2
6: **иначе**
7: перейти к строке 9
8: **Конец условия**
9: Подобрать q_i так, чтобы $Q = q_1 + q_2 + \cdots + q_n$
10: $w_i = q_i \cdot p_i + w_i^*$ для $i = \overline{1,n}$.
11: **Если** $C(p_k) = \sum_{i=1}^{k} w_i \left\lfloor \frac{p_k}{p_i} \right\rfloor \geq 0$ `AND` $\sum_{i=1}^{n} \left(\left\lfloor \frac{p_k}{p_i} \right\rfloor + 1 \right) \cdot w_i - w_k > 0$ для всех $k = \overline{1,n}$ **тогда**
12: **Возвратить** $w_1, w_2, \ldots, w_n$
13: **иначе**
14: $N = N + 1$ и перейти к строке 2
15: **Конец условия**

где $C(B_i) = \frac{B_i \cdot C_P}{P} - \frac{w_i}{p_i}$ и $B_i = P_i \cdot \left| P_i^{-1} \right|_P$. Использование $C_P = 2^N$ позволит эффективно выполнять операции деления и нахождения остатка от деления.

Ранг числа $X = (x_1, x_2, \ldots, x_n)$ вычисляется по следующей формуле

$$r_X = \left\lfloor \frac{\sum_{i=1}^{n} x_i \cdot C(B_i)}{C_P} \right\rfloor. \tag{2.38}$$

Для сравнения чисел воспользуемся следующим Алгоритмом 2.21.

Для определения знака числа, необходимо построить такую функцию ядра, чтобы $C(X) \leq C(K)$ для неотрицательных чисел и $C(X) > C(K)$ для отрицательных, где $K = \frac{P}{2} - 1$ если P — чётное; $K = \frac{(P-1)}{2}$ если P — нечётное. K является серединой диапазона СОК. Т.е. для определения знака достаточно воспользоваться Алгоритмом 2.21 для X и K.

Рассмотрим пример.

Алгоритм 2.21. Сравнение чисел, представленных в СОК, с использованием функции ядра с неотрицательными коэффициентами

Вход: $X = (x_1, x_2, \ldots, x_n)$, $Y = (y_1, y_2, \ldots, y_n)$

Выход: $X > Y$ — «10», $X < Y$ — «01», $X = Y$ — «00»

Данные в памяти:$\{p_1, p_2, \ldots, p_n\}$, $\{w_1, w_2, \ldots, w_n\}$

$C(B_i) = \frac{B_i \cdot C_P}{P} - \frac{w_i}{p_i}$, $B_i = P_i \cdot \left|P_i^{-1}\right|_P$

1: $C(X) = |\sum_{i=1}^{n} C(B_i) \cdot x_i|_{C_P}$, $C(Y) = |\sum_{i=1}^{n} C(B_i) \cdot y_i|_{C_P}$,

2: **Если** $C(X) < C(Y)$ **тогда**

3: **Возвратить** «01»

4: **иначе если** $C(X) > C(Y)$

5: **Возвратить** «10»

6: **иначе**

7: **Если** $x_k < y_k$ **тогда**

8: **Возвратить** «01»

9: **иначе если** $x_k > y_k$

10: **Возвратить** «10»

11: **иначе**

12: **Возвратить** «00»

13: **Конец условия**

14: **Конец условия**

Пример 24. *Возьмем в качестве СОК* $\{11, 13, 17, 19\}$. *Тогда* $P = 46189$, $P_1 = 4199$, $P_2 = 3553$, $P_3 = 2717$, $P_4 = 2431$, $\left|P_1^{-1}\right|_{11} = 7$, $\left|P_2^{-1}\right|_{13} = 10$, $\left|P_3^{-1}\right|_{17} = 11$, $\left|P_4^{-1}\right|_{19} = 18$, $B_1 = 29393$, $B_2 = 35530$, $B_3 = 29887$, $B_4 = 43758$. *Воспользовавшись Алгоритмом 2.20, получим* $N = 17$, $w_1 = 16$, $w_2 = 8$, $w_3 = 5$, $w_4 = 9$.

Тогда вспомогательные значения $C(B_i)$ *будут равны* $C(B_1) = 83408$, $C(B_2) = 100824$, $C(B_3) = 84811$, $C(B_4) = 124173$. *И функция ядра примет вид*

$$C(X) = |83408 \cdot x_1 + 100824 \cdot x_2 + 84811 \cdot x_3 + 124173 \cdot x_4|_{2^{17}} .$$

Серединой диапазона СОК будет $K = 23094$, *для которого* $C(K) = 65517$.

Определим знаки и сравним два числа, представленные в СОК: $A = (8, 10, 4, 7)$ *и* $B = (7, 7, 14, 17)$.

По формуле (2.37) *найдем значения функции ядра:* $C(A) = 375$, $C(B) = 399$. *Из алгоритма 2.21, т.к.* $C(A) < C(B) < C(K)$, *то оба чис-*

ла положительные и $A < B$. *Поскольку* $A = 140$, $B = 150$, *то вычисления корректны.*

Реализуемость и новизна Алгоритмов 2.20 и 2.21 подтверждается евразийским патентом на изобретение [102].

2.3.2 Модификация метода на основе КТО для сравнения чисел

Произведем модификацию метода сравнения чисел в СОК, для чего в качестве позиционной характеристики рассмотрим следующую функцию [144]:

$$f(X) = \left| \sum_{i=1}^{n} \overline{k_i}\, x_i \right|_{2^N}, \tag{2.39}$$

где $\overline{k_i} = \left\lfloor \frac{2^N \left| P_i^{-1} \right|_{p_i}}{p_i} \right\rfloor$.

Рассмотрим случай сравнение чисел в СОК с нечётным диапазоном.

Лемма 2.3.1. *Если* $N = \lceil \log_2 (-n + n \cdot P) \rceil$, *то справедливо следующее равенство*

$$\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i}\, x_i}{2^N} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} k_i x_i}{P} \right\rfloor, \tag{2.40}$$

где $\overline{k_i} = \left\lfloor \frac{2^N \left| P_i^{-1} \right|_{p_i}}{p_i} \right\rfloor$ *и* $k_i = \left| P_i^{-1} \right|_{p_i} P_i$.

Доказательство. Так как k_i и $\overline{k_i}$ связаны равенством $\overline{k_i} = \frac{2^N k_i}{P} - \frac{\left|2^N k_i\right|_P}{P}$, то выражение $\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} \cdot x_i}{2^N} \right\rfloor$ примет вид

$$\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} \cdot x_i}{2^N} \right\rfloor = \left\lfloor \frac{1}{P} \sum_{i=1}^{n} k_i x_i - \frac{1}{P \cdot 2^N} \sum_{i=1}^{n} \left|2^N k_i\right|_P \cdot x_i \right\rfloor. \tag{2.41}$$

Подставим $\frac{1}{P} \sum_{i=1}^{n} k_i x_i = \left\lfloor \frac{1}{P} \sum_{i=1}^{n} k_i x_i \right\rfloor + \frac{X}{P}$ в (2.41), получим

$$\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} x_i}{2^N} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} k_i x_i}{P} \right\rfloor + \left\lfloor \frac{X}{P} - \frac{1}{P \cdot\ 2^N} \cdot \sum_{i=1}^{n} \left|2^N k_i\right|_P x_i \right\rfloor. \tag{2.42}$$

Из (2.42) следует, что условие Леммы 2.3.1 эквивалентно следующему неравенству

$$0 \leq \frac{X}{P} - \frac{1}{P \cdot\ 2^N} \cdot \sum_{i=1}^{n} \left|2^N k_i\right|_P x_i < 1. \tag{2.43}$$

Согласно Китайской теореме об остатках, X удовлетворяет условию $0 \leq X < P$, следовательно, $0 \leq \frac{X}{P} < 1$. Принимая во внимание, что $\frac{1}{P\cdot\ 2^N} \cdot \sum_{i=1}^{n} \left|2^N k_i\right|_P x_i \geq 0$, мы получаем, что правая часть двойного неравенства (2.43) верна для всех N.

Рассмотрим левую часть двойного неравенства (2.43). При $X = 0$ она выполняется для любого N. Пусть X удовлетворяет неравенству $1 \leq X < P$, тогда левую часть неравенства (2.43) можно представить в следующем виде

$$2^N \geq \frac{1}{X} \sum_{i=1}^{n} \left|2^N k_i\right|_P x_i. \tag{2.44}$$

Так как $\left|2^N k_i\right|_P \leq P - 1$, то $\sum_{i=1}^{n} \left|2^N k_i\right|_P x_i \leq (P-1) \sum_{i=1}^{n} x_i$, следовательно, для всех $1 \leq X < P$ справедливо следующее неравенство:

$$\frac{1}{X} \sum_{i=1}^{n} \left|2^N k_i\right|_P x_i \leq n \cdot (P-1). \tag{2.45}$$

Из (2.44) и (2.45) следует, что если $N = \lceil \log_2 (-n + n \cdot P) \rceil$, то левая часть неравенства (2.43) выполняется, следовательно, равенство (2.40) выполняется. Лемма доказана. □

Теорема 2.3.2. *Если $N = \lceil \log_2 (-n + n \cdot P) \rceil$, то функция $f(X)$ — строго возрастающая.*

Доказательство. Для того, чтобы $f(X)$ являлась строго возрастающей функцией, необходимо и достаточно, чтобы для всех целых чисел $1 \leq X \leq P - 1$ выполнялось следующее условие

$$f(X) - f(X-1) > 0.$$

Так как $|X|_{2^N} = X - \left\lfloor \frac{X}{2^N} \right\rfloor \cdot 2^N$, то функцию $f(X)$ можно представить в следующем виде

$$f(X) = \sum_{i=1}^{n} \overline{k_i} x_i - \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} x_i}{2^N} \right\rfloor \cdot 2^N$$

Принимая во внимание, что

$$\sum_{i=1}^{n} \overline{k_i} \ \left(x_i - |x_i - 1|_{p_i} \right) = \sum_{i=1}^{n} \overline{k_i} - \sum_{x_i=0} \bar{k}_i p_i,$$

получим

$$f(X) - f(X-1) = \\ = \sum_{i=1}^{n} \overline{k_i} - \sum_{x_i=0} \overline{k_i} p_i - \left(\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} x_i}{2^N} \right\rfloor - \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} \, |x_i - 1|_{p_i}}{2^N} \right\rfloor \right) \cdot 2^N \quad (2.46)$$

Согласно Леммы 2.3.1

$$\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} x_i}{2^N} \right\rfloor - \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} \, |x_i - 1|_{p_i}}{2^N} \right\rfloor = \\ = \left\lfloor \frac{\sum_{i=1}^{n} k_i x_i}{P} \right\rfloor - \left\lfloor \frac{\sum_{i=1}^{n} k_i \, |x_i - 1|_{p_i}}{P} \right\rfloor. \quad (2.47)$$

С учетом теоремы из работы [6] и Леммы 2.3.1, формула (2.47) примет вид

$$\left\lfloor \frac{\sum_{i=1}^{n} k_i x_i}{P} \right\rfloor - \left\lfloor \frac{\sum_{i=1}^{n} k_i \, |x_i - 1|_{p_i}}{P} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} k_i}{P} \right\rfloor - \sum_{x_i=0} \left| P_i^{-1} \right|_{p_i} = \\ = \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i}}{2^N} \right\rfloor - \sum_{x_i=0} \left| P_i^{-1} \right|_{p_i}. \quad (2.48)$$

Подставив (2.48) в (2.46), получим

$$f(X) - f(X-1) = \sum_{i=1}^{n} \overline{k_i} - \sum_{x_i=0} \overline{k_i} p_i - \left(\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i}}{2^N} \right\rfloor - \sum_{x_i=0} \left| P_i^{-1} \right|_{p_i} \right) \cdot 2^N. \quad (2.49)$$

Так как $\sum_{i=1}^{n} \overline{k_i} - \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i}}{2^N} \right\rfloor \cdot 2^N = \left| \sum_{i=1}^{n} \overline{k_i} \right|_{2^N}$ и $\left| P_i^{-1} \right|_{p_i} \cdot 2^N - \bar{k}_i p_i = \frac{\left| 2^N k_i \right|_P}{P_i}$ то для всех $i = \overline{1, n}$ формула (2.49) примет вид

$$f(X) - f(X-1) = \left| \sum_{i=1}^{n} \overline{k_i} \right|_{2^N} + \sum_{x_i=0} \frac{\left| 2^N k_i \right|_P}{P_i}. \quad (2.50)$$

Так как $\left| \sum_{i=1}^{n} \overline{k_i} \right|_{2^N} > 0$, то из (2.50) следует, что $f(X) - f(X-1) > 0$, и, следовательно, функция $f(X)$ строго возрастает. Теорема доказана. $\square$

Алгоритм 2.22. Сравнения чисел в СОК с нечётным диапазоном на базе функции $f(X)$

Вход: $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$

Выход: $X > Y$ — «10», $X < Y$ — «01», $X = Y$ — «00»

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$

$\overline{k_i} = \left\lfloor 2^N \cdot \left| P_i^{-1} \right|_{p_i} / p_i \right\rfloor$ для всех $i = \overline{1,n}$, где $N = \lceil \log_2 (-n + nP) \rceil$

1: $f_X = 0$, $f_Y = 0$
2: **Цикл** $i = 1$ до n **выполнять**
3: $\quad f_X = \left| f_X + \overline{k_i} \cdot x_i \right|_{2^N}$
4: $\quad f_Y = \left| f_Y + \overline{k_i} \cdot y_i \right|_{2^N}$
5: **Конец цикла**
6: **Если** $f_X > f_Y$ **тогда**
7: $\quad$ **Возвратить** «10»
8: **иначе если** $f_X < f_Y$
9: $\quad$ **Возвратить** «01»
10: **иначе**
11: $\quad$ **Возвратить** «00»
12: **Конец условия**

Из Теоремы 2.3.2 следует, что введенная функция является строго монотонной, следовательно, ее можно использовать в качестве позиционной характеристики для сравнения чисел в СОК. На базе доказанной Теоремы 2.3.2 был разработан Алгоритм 2.22 сравнения чисел в СОК.

Предложенный метод позволяет уменьшить вычислительную сложность алгоритма сравнения чисел в СОК. Эффективная реализация операции $|x \cdot y|_{2^N}$ позволяет уменьшить логическую схему по сравнению с классическим умножением двух чисел $x \cdot y$.

Теперь рассмотрим случай сравнения чисел в СОК, когда один из модулей равен степени двойки.

Так как модули СОК являются взаимно простыми числами, возможен только один чётный модуль. Значит, без потери общности будем считать, что n-ый модуль имеет следующий вид $p_n = 2^t$. Так как $p_n = 2^t$, то используя свойство СОК, числа X, Y могут быть представлены в следующем виде

$$X = A \cdot 2^t + x_n, Y = B \cdot 2^t + y_n. \tag{2.51}$$

Для сравнения чисел, заданных выражением (2.51), используем Алгоритм 2.23.

Алгоритм 2.23. Сравнение чисел X и Y, представленных в виде $X = A \cdot p_n + x_n$, $Y = B \cdot p_n + y_n$

Вход: $X = (A, x_n)$ и $Y = (B, y_n)$

Выход: $X > Y$ — «10», $X < Y$ — «01», $X = Y$ — «00»

1: **Если** $A > B$ **тогда**
2: **Возвратить** «10»
3: **иначе если** $A < B$
4: **Возвратить** «01»
5: **иначе**
6: **Если** $x_n > y_n$ **тогда**
7: **Возвратить** «10»
8: **иначе если** $x_n < y_n$
9: **Возвратить** «01»
10: **иначе**
11: **Возвратить** «00»
12: **Конец условия**
13: **Конец условия**

Так как n-ый модуль СОК чётный, следовательно, модули $p_1, p_2, \ldots, p_{n-1}$ являются нечётными числами, тогда P_n — нечётное число. Коэффициенты A и B удовлетворяют неравенствам: $0 \leq A < P_n$ и $0 \leq B < P_n$. Вычислив значения A и B по модулям $p_1, p_2, \ldots, p_{n-1}$, мы можем сравнить их, используя Алгоритм 2.22. Таким образом, получим Алгоритм 2.24 для сравнения чисел X и Y в СОК.

Алгоритм 2.24. Сравнение чисел X и Y в СОК с чётным диапазоном

Вход: $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$

Выход: $X > Y$ — «10», $X < Y$ —«01», $X = Y$ — «00»

Данные в памяти: $\{p_1, p_2, \ldots, p_{n-1}, p_n\}$

$I_i = \left|\frac{1}{p_n}\right|_{p_i}$ для всех $i = \overline{1,n-1}$

$\overline{k_i} = \left\lfloor 2^N \cdot |1/P_i^*|_{p_i} / p_i \right\rfloor$ для всех $i = \overline{1,n-1}$

$N = \lceil \log_2 (-n + nP_n) \rceil$

$P_i^* = \frac{P_n}{p_i}$ для всех $i = \overline{1,n-1}$

Цикл $i = 1$ до $n-1$ **выполнять**

    $a_i = |x_i - x_n|_{p_i}$

    $b_i = |y_i - y_n|_{p_i}$

Конец цикла

Цикл $i = 1$ до $n-1$ **выполнять**

    $a_i = |a_i \cdot I_i|_{p_i}$

    $b_i = |b_i \cdot I_i|_{p_i}$

Конец цикла

$S_A = 0$, $S_B = 0$

Цикл $i = 1$ до $n-1$ **выполнять**

    $S_A = \left|S_A + \bar{k_i} \cdot a_i\right|_{2^N}$

    $S_B = \left|S_B + \bar{k_i} \cdot b_i\right|_{2^N}$

Конец цикла

Если $S_A > S_B$ **тогда**

    Возвратить «10»

иначе если $S_A < S_B$

    Возвратить «01»

иначе если $x_n > y_n$

    Возвратить «10»

иначе если $x_n < y_n$

    Возвратить «01»

иначе

    Возвратить «00»

Конец условия

2.3.3 Разработка метода определения знака в СОК с чётными модулями

Рассмотрим разработанный метод определения знака числа [118]. Возьмем систему остаточных классов с модулями $p_1 < p_2 < \ldots < p_{n-1} < p_n = 2^m$. Знак в системе остаточных классов чаще всего вводится разбиением диапазона на две части, тогда посредством динамического диапазона P в СОК можно представить числа $-\frac{P}{2} \leq X \leq \frac{P}{2} - 1$, если P — чётное. Функция определения знака числа, представленного в СОК, определяется выражением (2.36). Таким образом $S(X)$ можно представить в виде:

$$S(X) = \left\lfloor \frac{2X}{P} \right\rfloor. \tag{2.52}$$

Используя свойство $\left\lfloor \left\lfloor \frac{X}{a} \right\rfloor / b \right\rfloor = \left\lfloor \frac{X}{a \cdot b} \right\rfloor$ и формулу (2.52) определение знака можно свести к двухэтапному алгоритму: первый этап — деление на $P_n = \frac{P}{p_n} \in N$, второй этап — деление на $\frac{p_n}{2} \in N$. Формально этот алгоритм математически определяется следующей формулой

$$S(X) = \left\lfloor \frac{2X}{P} \right\rfloor = \left\lfloor \left\lfloor \frac{X}{P_n} \right\rfloor \cdot \frac{2}{p_n} \right\rfloor.$$

Запишем процесс определения знака в виде Алгоритма 2.25 [118].

Алгоритм 2.25. Определение знака числа в СОК с чётным диапазоном

Вход: $X = (x_1, x_2, \ldots, x_n)$

Выход: $X > 0$ — «0», $X < 0$ — «1»

Данные в памяти: $p_1 < p_2 < \ldots < p_{n-1} < p_n = 2^m$

$w_{i,j} = \left| p_i^{-1} \right|_{p_j}$

1: **Цикл** $i = 1$ до $n - 1$ **выполнять**

2: **Цикл** $j = i + 1$ до n **выполнять**

3: $x_j = \left| (x_j - x_i) \cdot w_{i,j} \right|_{p_j}$

4: **Конец цикла**

5: **Конец цикла**

6: **Возвратить** $\left\lfloor \frac{2x_n}{p_n} \right\rfloor$

На первом этапе вычисляется $\left\lfloor \frac{X}{P_n} \right\rfloor$ с помощью $n - 1$ деления на модули СОК $p_1, p_2, \ldots, p_{n-1}$ соответственно. На втором этапе вычисляется $S(X) = \left\lfloor \left\lfloor \frac{X}{P_n} \right\rfloor \cdot \frac{2}{p_n} \right\rfloor$.

Рассмотрим пример на границе положительных и отрицательных чисел.

Пример 25. *Пусть задана система остаточных классов с модулями* $p_1 = 17, p_2 = 19, p_3 = 23, p_4 = 32$*, для которой веса* $w_{i,j}$ *равны*

$$w_{1,2} = \left|p_1^{-1}\right|_{p_2} = \left|\frac{1}{17}\right|_{19} = 9, \quad w_{1,3} = \left|p_1^{-1}\right|_{p_3} = \left|\frac{1}{17}\right|_{23} = 19,$$
$$w_{1,4} = \left|p_1^{-1}\right|_{p_4} = \left|\frac{1}{17}\right|_{32} = 17, \quad w_{2,3} = \left|p_2^{-1}\right|_{p_3} = \left|\frac{1}{19}\right|_{23} = 17,$$
$$w_{2,4} = \left|p_2^{-1}\right|_{p_4} = \left|\frac{1}{19}\right|_{32} = 27, \quad w_{3,4} = \left|p_3^{-1}\right|_{p_4} = \left|\frac{1}{23}\right|_{32} = 7.$$

Определим знак числа $X = 118863 = (16, 18, 22, 15)$*. В соответствии с шагом 3 Алгоритма 2.25 получим*

$$x_2 = |(x_2 - x_1) \cdot w_{1,2}|_{p_2} = |(18 - 16) \cdot 9|_{19} = 18,$$
$$x_3 = |(x_3 - x_1) \cdot w_{1,3}|_{p_3} = |(22 - 16) \cdot 19|_{23} = 22,$$
$$x_4 = |(x_4 - x_1) \cdot w_{1,4}|_{p_4} = |(15 - 16) \cdot 17|_{32} = 15,$$
$$x_3 = |(x_3 - x_2) \cdot w_{2,3}|_{p_3} = |(22 - 18) \cdot 17|_{23} = 22,$$
$$x_4 = |(x_4 - x_2) \cdot w_{2,4}|_{p_4} = |(15 - 18) \cdot 27|_{32} = 15,$$
$$x_4 = |(x_4 - x_3) \cdot w_{3,4}|_{p_4} = |(15 - 22) \cdot 7|_{32} = 15.$$

Тогда результатом работы Алгоритма 2.25 будет $\left\lfloor \frac{2x_n}{p_n} \right\rfloor = \left\lfloor \frac{2 \cdot 15}{32} \right\rfloor = 0$*, т.е. число* $X = 118863 = (16, 18, 22, 15)$ *— положительное.*

Определим знак числа $Y = 118864 = (0, 0, 0, 16)$*. Аналогично*

$$y_2 = |(y_2 - y_1) \cdot w_{1,2}|_{p_2} = |(0 - 0) \cdot 9|_{19} = 0,$$
$$y_3 = |(y_3 - y_1) \cdot w_{1,3}|_{p_3} = |(0 - 0) \cdot 19|_{23} = 0,$$
$$y_4 = |(y_4 - y_1) \cdot w_{1,4}|_{p_4} = |(16 - 0) \cdot 17|_{32} = 16,$$
$$y_3 = |(y_3 - y_2) \cdot w_{2,3}|_{p_3} = |(0 - 0) \cdot 17|_{23} = 0,$$
$$y_4 = |(y_4 - y_2) \cdot w_{2,4}|_{p_4} = |(16 - 0) \cdot 27|_{32} = 16,$$
$$y_4 = |(y_4 - y_3) \cdot w_{3,4}|_{p_4} = |(16 - 0) \cdot 7|_{32} = 16.$$

Тогда результатом работы Алгоритма 2.25 будет $\left\lfloor \frac{2y_n}{p_n} \right\rfloor = \left\lfloor \frac{2 \cdot 16}{32} \right\rfloor = 1$*, т.е. число* $Y = 118864 = (0, 0, 0, 16)$ *— отрицательное.*

Поскольку все вычисления в Алгоритме 2.25 выполняются над целочисленными значениями малой размерности, увеличивается скорость вычислений

и отсутствуют ошибки округления, которые могут возникать в приближенном методе (формула (2.31)) [68].

Рассмотрим сравнение чисел на основе определения знака с использованием Алгоритма 2.25. Для сравнения двух чисел X и Y можно в случае, если $X \neq Y$, определить знаки чисел X и Y, если числа разного знака, то можно сделать однозначный вывод, какое число больше. Если же числа одного знака, то необходимо определить знак разности $X-Y$. Если знак $X-Y$ — отрицательный, то $X < Y$, если положительный, то $X > Y$. Необходимость определения знака каждого числа связана с возможностью переполнения динамического диапазона при выполнении арифметических операций.

2.3.4 Разработка метода определения знака в СОК с нечётными модулями

Рассмотрим метод определения знака для чисел с нечётными модулями СОК $\{p_1, p_2, \ldots, p_n\}$ [114]. Чтобы определить знак, рассмотрим две функции, $g_1(X) = \left\lfloor \frac{X}{P_n} \right\rfloor$ и $g_2(X) = \left\lfloor \frac{A}{P_n} \right\rfloor$, где $A = \left| X + \frac{P_n - 1}{2} \right|_P$, тогда функцию знака числа $S(X)$ можно получить из выражения

$$S(X) = \begin{cases} 1, \text{если } \left(g_1(X) > \frac{p_n-1}{2}\right) \text{OR } \left(\left(g_1(X) = \frac{p_n-1}{2}\right) \text{AND } \left(g_2(X) = \frac{p_n+1}{2}\right)\right), \\ 0, \text{иначе.} \end{cases}$$

Рассмотрим пример.

Пример 26. *Возьмем систему остаточных классов с модулями* $p_1 = 17$, $p_2 = 19$, $p_3 = 23$, $p_4 = 31$. *В данной системе динамический диапазон* $P = \prod_{i=1}^{4} p_i = 230299$, *и число* X *будет неотрицательным, если* $0 \leq X \leq \frac{P-1}{2}$, *т.е.* $0 \leq X \leq 115149$, *а отрицательным для* $\frac{P+1}{2} \leq X < P$, *т.е.* $1115150 \leq X < 230299$.

Рассмотрим значения на границе перехода от неотрицательных к отрицательным числам.

$$P_4 = \frac{P}{p_4} = 7429.$$

Для числа $X = 115149$, $A = \left| X + \frac{P_4 - 1}{2} \right|_P = 118863$, *тогда*

$$g_1(X) = \left\lfloor \frac{X}{P_4} \right\rfloor = \left\lfloor \frac{115149}{7429} \right\rfloor = 15, \quad g_2(X) = \left\lfloor \frac{A}{P_4} \right\rfloor = \left\lfloor \frac{118863}{7429} \right\rfloor = 15.$$

Поскольку $g_1(115149) = 15 = \frac{p_4-1}{2}$, *а* $g_2(115149) = 15 \neq \frac{p_4+1}{2}$, *то* $S(X) = 0$, *т.е. число положительное.*

Для числа $Y = 115150$, $A = \left|Y + \frac{P_4-1}{2}\right|_P = 118864$, *тогда*

$$g_1(Y) = \left\lfloor \frac{Y}{P_4} \right\rfloor = \left\lfloor \frac{115150}{7429} \right\rfloor = 15, \quad g_2(Y) = \left\lfloor \frac{A}{P_4} \right\rfloor = \left\lfloor \frac{118864}{7429} \right\rfloor = 16.$$

Поскольку $g_1(115150) = 15 = \frac{p_4-1}{2}$, *а* $g_2(115150) = 16 = \frac{p_4+1}{2}$, *то* $S(Y) = 1$, *т.е. число отрицательное.*

Запишем алгоритм определения $S(X)$ — знака числа X, представленного в СОК с нечётным динамическим диапазоном. Алгоритм 2.26 основан на последовательном приближении значения $\left\lfloor \frac{X}{P_n} \right\rfloor$ и использует свойство $\left\lfloor \left\lfloor \frac{X}{a} \right\rfloor / b \right\rfloor = \left\lfloor \frac{X}{a \cdot b} \right\rfloor$.

Алгоритм 2.26. Определение знака числа в СОК с нечётным диапазоном

Вход: $X = (x_1, x_2, \ldots, x_n)$

Выход: $X > 0$ — «0», $X < 0$ — «1»

Данные в памяти: $\{p_1, p_2, \ldots, p_n\}$

$w_{i,j} = \left|p_i^{-1}\right|_{p_j}$

$P_n = \frac{\prod_{i=1}^{n} p_i}{p_n}$

$\Delta = \frac{P_n-1}{2} = (\Delta_1, \Delta_2, \ldots, \Delta_n) = \left(\left|\frac{P_n-1}{2}\right|_{p_1}, \left|\frac{P_n-1}{2}\right|_{p_2}, \ldots, \left|\frac{P_n-1}{2}\right|_{p_n}\right)$

1: $A = \left|X + \frac{P_n-1}{2}\right|_P = (a_1, a_2, \ldots, a_n) =$
$= \left(|x_1 + \Delta_1|_{p_1}, |x_2 + \Delta_2|_{p_2}, \ldots, |x_n + \Delta_n|_{p_n}\right)$

2: **Цикл** $i = 1$ до $n-1$ **выполнять**

3: **Цикл** $j = i+1$ до n **выполнять**

4: $x_j = |(x_j - x_i) \cdot w_{i,j}|_{p_j}$

5: $a_j = |(a_j - a_i) \cdot w_{i,j}|_{p_j}$

6: **Конец цикла**

7: **Конец цикла**

8: **Если** $\left(x_n > \frac{p_n-1}{2}\right)$ `OR` $\left(\left(x_n = \frac{p_n-1}{2}\right)$ `AND` $\left(a_n = \frac{p_n+1}{2}\right)\right)$ **тогда**

9: **Возвратить** «1»

10: **иначе**

11: **Возвратить** «0»

12: **Конец условия**

Рассмотрим пример, иллюстрирующий работу Алгоритма 2.26.

Пример 27. *Для СОК с нечётными модулями* $p_1 = 17$, $p_2 = 19$, $p_3 = 23$, $p_4 = 31$, *диапазон СОК* $P = 230299$ *и* $\frac{P_n-1}{2} = 3714 = (8, 9, 11, 25)$. *Веса* $w_{i,j}$ *равны*

$$w_{1,2} = \left|p_1^{-1}\right|_{p_2} = \left|\frac{1}{17}\right|_{19} = 9, \quad w_{1,3} = \left|p_1^{-1}\right|_{p_3} = \left|\frac{1}{17}\right|_{23} = 19,$$
$$w_{1,4} = \left|p_1^{-1}\right|_{p_4} = \left|\frac{1}{17}\right|_{31} = 11, \quad w_{2,3} = \left|p_2^{-1}\right|_{p_3} = \left|\frac{1}{19}\right|_{23} = 17,$$
$$w_{2,4} = \left|p_2^{-1}\right|_{p_4} = \left|\frac{1}{19}\right|_{31} = 18, \quad w_{3,4} = \left|p_3^{-1}\right|_{p_4} = \left|\frac{1}{23}\right|_{31} = 27.$$

Для определения знака числа $X = 115149 = (8, 9, 11, 15)$, *найдем вспомогательное значение* $A = \left|X + \frac{P_n-1}{2}\right|_P$

$$A = \left(\left|8+8\right|_{17}, \left|9+9\right|_{19}, \left|11+11\right|_{23}, \left|15+25\right|_{31}\right) = (16, 18, 22, 9).$$

Для удобства запишем вычисления в виде Таблицы 2, где $x^{(i)}$ *означает значение* x, *вычисленное на* i *вычислительной ступени,* $i = \overline{1, n-1}$.

Таблица 2 — Вычислительные ступени для числа $X = 115149 = (8, 9, 11, 15)$

Вычислительная ступень	Операция	p	17	19	23	31
		X	8	9	11	15
		A	16	18	22	9
1	$X^{(1)} = \left\lfloor \frac{X}{p_1} \right\rfloor$	$-x_1$	0	1	3	7
		$\times w_{1,j}$	-	9	11	15
	$A^{(1)} = \left\lfloor \frac{A}{p_1} \right\rfloor$	$-a_1$	0	2	6	24
		$\times w_{1,j}$	-	18	22	16
2	$X^{(2)} = \left\lfloor \frac{X^{(1)}}{p_2} \right\rfloor$	$-x_2^{(1)}$	-	0	2	6
		$\times w_{2,j}$	-	-	11	15
	$A^{(2)} = \left\lfloor \frac{A^{(1)}}{p_2} \right\rfloor$	$-a_2^{(1)}$	-	0	4	29
		$\times w_{2,j}$	-	-	22	26
3	$X^{(3)} = \left\lfloor \frac{X^{(2)}}{p_3} \right\rfloor$	$-x_3^{(2)}$	-	-	0	4
		$\times w_{3,j}$	-	-	-	**15**
	$A^{(3)} = \left\lfloor \frac{A^{(2)}}{p_3} \right\rfloor$	$-a_3^{(2)}$	-	-	0	4
		$\times w_{3,j}$	-	-	-	**15**

После трех вычислительных ступеней сравним значения $x_n = 15$ *и* $a_n = 15$ *с* $\frac{p_4-1}{2} = 15$ *и* $\frac{p_4+1}{2} = 16$. *Поскольку* $x_n = \frac{p_4-1}{2}$, *а* $a_n \neq \frac{p_4+1}{2}$, *то число положительное.*

Для определения знака числа $X = 115150 = (9, 10, 12, 16)$, *найдем вспомогательное значение* $A = \left|X + \frac{P_n-1}{2}\right|_P$

$$A = \left(|9+8|_{17}, |10+9|_{19}, |12+11|_{23}, |16+25|_{31}\right) = (0, 0, 0, 10).$$

Аналогично, для удобства запишем вычисления в виде Таблицы 3, где $x^{(i)}$ *означает значение* x, *вычисленное на* i *вычислительной ступени,* $i = \overline{1, n-1}$.

Таблица 3 — Вычислительные ступени для числа $X = 115150 = (9, 10, 12, 16)$

Вычислительная ступень	Операция	p	17	19	23	31
		X	9	10	12	16
		A	0	0	0	10
1	$X^{(1)} = \left\lfloor \frac{X}{p_1} \right\rfloor$	$-x_1$	0	1	3	7
		$\times w_{1,j}$	-	9	11	15
	$A^{(1)} = \left\lfloor \frac{A}{p_1} \right\rfloor$	$-a_1$	0	0	0	10
		$\times w_{1,j}$	-	0	0	17
2	$X^{(2)} = \left\lfloor \frac{X^{(1)}}{p_2} \right\rfloor$	$-x_2^{(1)}$	-	0	2	6
		$\times w_{2,j}$	-	-	11	15
	$A^{(2)} = \left\lfloor \frac{A^{(1)}}{p_2} \right\rfloor$	$-a_2^{(1)}$	-	0	0	17
		$\times w_{2,j}$	-	-	0	27
3	$X^{(3)} = \left\lfloor \frac{X^{(2)}}{p_3} \right\rfloor$	$-x_3^{(2)}$	-	-	0	4
		$\times w_{3,j}$	-	-	-	**15**
	$A^{(3)} = \left\lfloor \frac{A^{(2)}}{p_3} \right\rfloor$	$-a_3^{(2)}$	-	-	0	27
		$\times w_{3,j}$	-	-	-	**16**

После трех вычислительных ступеней сравним значения $x_n = 15$ *и* $a_n = 16$ *с* $\frac{p_4-1}{2} = 15$ *и* $\frac{p_4+1}{2} = 16$. *Поскольку* $x_n = \frac{p_4-1}{2}$, *а* $a_n = \frac{p_4+1}{2}$, *то число отрицательное.*

Данный метод имеет точную реализацию, без накопления ошибок, как в приближенном методе, однако вычисление двух функций приводит к увеличению используемых ресурсов.

Моделирование данных методов с использованием ASIC будет рассмотрено в Параграфе 3.5.

2.4 Разработка метода модульного умножения в СОК

Эффективность и применимость систем цифровой обработки сигналов во многом зависит от скорости выполнения арифметических операций и надежности вычислений, в частности от модульного умножения и умножения с накоплением. Повышению эффективности модульного умножения посвящено большое количество работ. Рассмотрим наиболее известные методы модульного умножения.

Алгоритмы модульного умножения $A \cdot B \bmod p$ можно разбить на два класса:

1. Нахождение остатка от деления на фиксированный модуль после вычисления произведения.
2. Нахождение остатка в процессе умножения (чередование операций).

Наибольшее распространение получили следующие методы: нейронная сеть конечного кольца, методы умножения Карацубы-Офмана [38] и Бута [13], метод Барретта [11] для редукции, а также алгоритмы Монтгомери [52] и метод Брикелла [14; 87] для чередования умножения и нахождения остатка.

Методы последовательного умножения и нахождения остатка состоят в том, чтобы сначала вычислить произведение, а затем сократить его по отношению к данному модулю. Использование этих методов, как правило, предпочтительнее, поскольку существуют готовые очень быстрые алгоритмы умножения. Рассмотрим некоторые из них: нейронную сеть конечного кольца, метод Карацубы-Офмана и метод Бута.

Нейронная сеть конечного кольца представляет собой высокопараллельную структуру с топологией направленного графа [109; 142]. При этом нейронами являются арифметические элементы, имеющие характеристики оператора по модулю. Арифметика конечного кольца часто применяется как альтернатива традиционной двоичной арифметики. Среди областей применения модулярной арифметики можно выделить цифровую обработку сигналов и криптографию, где одной из основных операций является операция модулярного умножения [111].

В СОК мы имеем дело с конечными вычислительными структурами (кольца или поля), которые используются для реализации арифметических кольцевых

операций. Основной при этом является операция вычисления остатка целочисленного деления по модулю.

Умножение двух n-битных чисел дает $2n$-битное число. Таким образом, перед нами стоит задача нахождения остатка $2n$-битного числа по n-битному модулю p.

Структура нейронной сети конечного кольца зависит от размера исходного числа и величины модуля и во время работы не изменяется. Нейронная сеть состоит из входного слоя, скрытых слоев, и выходного слоя. Веса связей определяются константами $c_i \equiv 2^i \bmod p$, где $i = 1, 2, \ldots, 2n$. Количество констант определяется разрядностью произведения. Скрытые слои рекурсивно объединяются, организуя логарифмическое суммирование. Выходной слой определяет наименьший неотрицательный остаток $A \cdot B \equiv R \bmod p$. Пусть число записано в двоичной системе счисления

$$A \cdot B = a_{2n}2^{2n} + a_{2n-1}2^{2n-1} + \ldots + a_1 2^1 + a_0 2^0, \quad a_i = \{0,1\}.$$

Подставляя сюда значения c_i и учитывая свойства сравнений, получим

$$A \cdot B = a_{2n}c_{2n} + a_{2n-1}c_{2n-1} + \ldots + a_1 c_1 + a_0 c_0 \equiv R_1 \bmod p. \tag{2.53}$$

Разрядность числа R_1 меньше, чем у произведения. Далее необходимо сравнить полученное число с модулем. Если $R_1 < p$, то R_1 является искомым остатком, иначе необходимо повторно провести аналогичную операцию над числом R_1. Для реализации данного метода необходимы коэффициенты c_i, которые являются константами для данного модуля и могут быть вычислены заранее. Реализация данного метода отражена в Алгоритме 2.3.

Рассмотрим пример.

Пример 28. *Пусть в результате умножения двух двоичных четырехразрядных чисел* $A = 15_{10} = 1111_2$ *и* $B = 9_{10} = 1001_2$ *получено число* $A \cdot B = 135_{10} = 10000111_2$. *Найдем остаток от деления по модулю* $p = 13_{10} = 1101_2$. *Коэффициенты* c_i *по модулю* 13 *равны* $c_0 \equiv |1|_{13}$, $c_1 \equiv |2|_{13}$, $c_2 \equiv |4|_{13}$, $c_3 \equiv |8|_{13}$, $c_4 \equiv |16|_{13} \equiv |3|_{13}$, $c_5 \equiv |32|_{13} \equiv |6|_{13}$, $c_6 \equiv |64|_{13} \equiv |12|_{13}$, $c_7 \equiv |128|_{13} \equiv |11|_{13}$.

Выражение (2.53) *для двоичного представления числа означает сумму только тех коэффициентов* c_i, *для которых* $a_i = 1$. *Таким образом, остаток от деления может быть получен только с использованием сложения.*

Найдем значение R_1 с учетом коэффициентов c_i

$$R_1 = 1 \cdot 1 + 1 \cdot 2 + 1 \cdot 4 + 0 \cdot 8 + 0 \cdot 3 + 0 \cdot 6 + 0 \cdot 12 + 1 \cdot 11 = 18.$$

Поскольку $R_1 > 13$, то повторим действия для $R_1 = 18_{10} = 10010_2$.

$$R_2 = 0 \cdot 1 + 1 \cdot 2 + 0 \cdot 4 + 0 \cdot 8 + 1 \cdot 3 = 5.$$

Таким образом, результат модулярного умножения $15 \cdot 9 \equiv 135 \bmod 13 = 5$.

Алгоритм Карацубы-Офмана считается одним из самых быстрых способов умножения длинных целых чисел. Было показано, что обобщения этого алгоритма даже быстрее, чем метод быстрого преобразования Фурье (БПФ) Шёнхаге-Штрассена [77; 95].

Пусть A и B будут двоичным представлением двух длинных целых чисел: $A = \sum_{i=0}^{k-1} a_i \cdot 2^i$ и $B = \sum_{i=0}^{k-1} b_i \cdot 2^i$. Вычислим произведение $A \cdot B$. Операнды A и B можно разложить на части одинакового размера A_H и A_L, B_H и B_L соответственно, которые представляют старшие и младшие n бит чисел A и B. Пусть $k = 2n$, если k нечётное, числа могут быть дополнены слева нулем.

$$A = 2^n \left(\sum_{i=0}^{n-1} a_{i+n} 2^i \right) + \sum_{i=0}^{n-1} a_i 2^i = A_H 2^n + A_L,$$
$$B = 2^n \left(\sum_{i=0}^{n-1} b_{i+n} 2^i \right) + \sum_{i=0}^{n-1} b_i 2^i = B_H 2^n + B_L.$$

Таким образом, произведение $M = A \cdot B$ может быть вычислено как

$$M = (A_H 2^n + A_L)(B_H 2^n + B_L) = 2^{2n}(A_H B_H) + 2^n (A_H B_L + A_L B_H) + A_L B_L.$$

Приведенная выше формула для вычисления произведения M реализуется посредством четырех n-битных умножений.

Вычисление M можно ускорить [54], заметив следующее

$$A_H B_L + A_L B_H = (A_H + A_L)(B_H + B_L) - A_H B_H - A_L B_L.$$

Алгоритм Карацубы-Офмана основан на приведенном выше наблюдении, и поэтому $2n$-битное умножение может быть сокращено до трех n-битных умножений, а именно $A_H B_H$, $A_L B_L$ и $(A_H + A_L)(B_H + B_L)$. Метод умножения Карацубы-Офмана можно выразить Алгоритмом 2.27, где функция $\texttt{Size}(X)$ возвращает число бит в X, функция $\texttt{High}(X)$ возвращает старшую половину X,

функция $\texttt{Low}(X)$ возвращает младшую половину X, $\texttt{LeftShift}(X, n)$ возвращает $X2^n$ и $\texttt{OneBitMultiplication}(X,Y)$ возвращает XY, где X и Y однобитные операнды. Если $\texttt{Size}(X)$ нечётный, то X дополняют нулем слева перед извлечением старшей и младшей половины $\texttt{High}(X)$ и $\texttt{Low}(X)$ соответственно.

Алгоритм 2.27. Рекурсивное умножение Карацубы-Офмана

Вход: A, B

Выход: $M = A \cdot B$

1: **Функция** $\texttt{KaratsubaOfman}(A, B)$
2: **Если** $\texttt{Size}(A) = 1$ **тогда**
3: $M = \texttt{OneBitMultiplier}(A, B)$
4: **иначе**
5: $M_1 = \texttt{KaratsubaOfman}(\texttt{High}(A), \texttt{High}(B))$;
6: $M_2 = \texttt{KaratsubaOfman}(\texttt{Low}(A), \texttt{Low}(B))$;
7: $M_3 = \texttt{KaratsubaOfman}(\texttt{High}(A) + \texttt{Low}(A), \texttt{High}(B) + \texttt{Low}(B))$;
8: $M = \texttt{LeftShift}(M_1, \texttt{Size}(A)) + \texttt{LeftShift}(M_3 - M_1 - M_2, \texttt{Size}(A)/2) + M_2$
9: **Конец условия**
10: **Конец функции**
11: **Возвратить** M

Рассмотрим улучшение Алгоритма 2.27. Наибольшую сложность имеет вычисление третьего произведения M_3, операнды которого содержат $\texttt{Size}(A)/2 + 1$ бит. Обозначим эти операнды через Z и U и дополним их слева $\texttt{Size}(A)/2 - 1$ нулями. Теперь Z и U имеют размер $\texttt{Size}(A)$ бит. Теперь можно записать произведение M_3 следующим образом

$$M_3 = ZU = (Z_H 2^n + Z_L)(U_H 2^n + U_L) = \\ = 2^{2n}(Z_H U_H) + 2^n(Z_H U_L + Z_L U_H) + Z_L U_L,$$

при этом $\texttt{Size}(A) = 2n$, Z_H и U_H — старшие части Z и U соответственно и Z_L и U_L — младшие части Z и U соответственно. Обратим внимание, что Z_H и U_H могут быть равны 0 или 1.

В зависимости от значений Z_H и U_H, M_3 может быть получено с использованием одного из вариантов Таблицы 4.

Таблица 4 — Значения третьего рекурсивного вызова M_3

Z_H	U_H	M_3
0	0	$Z_L U_L$
0	1	$2^n Z_L + Z_L U_L$
1	0	$2^n U_L + Z_L U_L$
1	1	$2^{2n} + 2^n(U_L + Z_L) + Z_L U_L$

Как видно из Таблицы 4, вычисление третьего произведения требует одного умножения размера n бит и некоторых дополнительных операций сложения, сдвига и мультиплексирования.

Пример 29. *Рассмотрим аналогичный пример умножения с операндами* $A = 15_{10} = 1111_2$ *и* $B = 9_{10} = 1001_2$. *Тогда* $A_H = 11_2 = 3_{10}$, $A_L = 11_2 = 3_{10}$, $B_H = 10_2 = 2_{10}$, $B_L = 01_2 = 1_{10}$. *Для получения результата необходимо параллельно вычислить три произведения*

$$A_H \cdot B_H = 3 \cdot 2 = 6, \qquad A_L \cdot B_L = 3 \cdot 1 = 3,$$
$$(A_H + A_L)(B_H + B_L) = (3 + 3)(2 + 1) = 18.$$

Тогда произведение будет равно

$$A \cdot B = 2^4 \cdot A_H B_H + 2^2 \left((A_H + A_L)(B_H + B_L) - A_H B_H - A_L B_L\right) + A_L B_L =$$
$$= 16 \cdot 6 + 4 \cdot (18 - 6 - 3) + 3 = 96 + 36 + 3 = 135.$$

Таким образом, произведение сводится к умножениям с числами меньшей разрядности, которые можно выполнить параллельно, суммированию и сдвигу влево.

Другим алгоритмом вычисления произведения является алгоритм Бута, который основан на вычислении частичных произведений. Пусть A и B — множимое и множитель соответственно, а n и m — их соответствующие размеры. Представим A и B следующим образом

$$A = \sum_{i=0}^{n-1} a_i \times 2^i \text{ и } B = \sum_{i=0}^{m-1} b_i \times 2^i \implies A \times B = \sum_{i=0}^{n-1} a_i \times B \times 2^i.$$

Метод Бута [13] генерирует n частичных произведений $a_i \times B$, $0 \leq i < n$, сдвигает их влево и складывает с предыдущими. Количество сгенерированных

частичных произведений ограничено сверху размером операнда A. Процесс вычисления произведения методом Бута описан Алгоритмом 2.28.

Алгоритм 2.28. Умножение Бута

Вход: $A = \sum_{i=0}^{n-1} a_i \times 2^i, B = \sum_{i=0}^{m-1} b_i \times 2^i$

Выход: $M = A \times B$

1: $M = 0$

2: **Цикл от** $i = 0$ **до** $n - 1$ **выполнять**

3: $\quad M := M + 2^i \cdot a_i \cdot B$

4: **Конец цикла**

5: **Возвратить** M

Пример 30. *Рассмотрим пример на предыдущих значениях* $A = 15_{10} = 1111_2$ *и* $B = 9_{10} = 1001_2$.

$$M = 1 \cdot 9 \cdot 2^0 + 1 \cdot 9 \cdot 2^1 + 1 \cdot 9 \cdot 2^2 + 1 \cdot 9 \cdot 2^3 = 9 + 18 + 36 + 72 = 135.$$

Таким образом, можно эффективно найти произведение двух чисел. Следующим этапом модулярного умножения будет вычисление остатка от целочисленного деления. Его можно обозначить следующим образом

$$X \bmod p = X - \left\lfloor \frac{X}{p} \right\rfloor \times p.$$

Однако эта операция является ресурсоемкой и требует значительной оптимизации.

Самый простой алгоритм последовательного деления (Алгоритм 2.29) последовательно сдвигает и вычитает модуль, пока не будет найден остаток, неотрицательный и меньший, чем модуль. Обратим внимание, что после вычитания может быть получен отрицательный остаток. В этом случае последний неотрицательный остаток должен быть восстановлен и возвращен в качестве результата.

Поскольку на вход Алгоритма 2.29 поступает результат умножения двух n-битных чисел, то размерность X равна $2n$. Алгоритм 2.29 неэффективен, поскольку может потребовать 2^{n-1} вычитаний, $2^{n-1} + 2$ сравнений и дополнительного сложения.

Улучшение данного алгоритма возможно за счет вычитания числа, кратного модулю p в двоичной системе счисления. Число, кратное p в двоичной

Алгоритм 2.29. Нахождение остатка от деления

Вход: X, p

Выход: $X \bmod p$

1: $R = X$
2: **До тех пор пока** $R \geq 0$ **выполнять**
3: $\quad R = R - p$
4: **Конец цикла**
5: **Если** $R \neq 0$ **тогда**
6: $\quad R = R + p$
7: **Конец условия**
8: **Возвратить** R

системе счисления, получается сдвигом влево. Необходимо вычесть из X произведение p на 2 в максимальной степени. Получим Алгоритм 2.30, который требует n вычитаний, n сравнений и $2n$ сдвигов.

Другим подходом к модулярному умножению является нахождение остатка в процессе умножения.

Один из подходов к повышению эффективности выполнения модулярного умножения предложен Монтгомери в статье [52]. Он заключается в замене вычислений по большому модулю p на операции по модулю R, который выбирается таким образом, чтобы операции деления и нахождения остатка были эффективными. Часто R берется в виде степени числа 2. Также должен выполняться ряд условий: $R > p$, R и p взаимно простые, а также вводятся R^{-1} и p', удовлетворяющие выражениям $0 < R^{-1} < p$, $0 < p' < R$ и $RR^{-1} - pp' = 1$. Последнее выражение является диафантовым уравнением и может быть решено, например, методом Евклида. Тогда нахождение произведения методом Монтгомери можно описать Алгоритмом 2.31.

Использование данного алгоритма в вычислительных устройствах позволяет уменьшить аппаратную сложность по сравнению со стандартным модулярным умножением. Однако недостатком этого подхода является то, что результатом алгоритма $t =$`MontMult(`$A \cdot B$`)` является не точный результат $A \cdot B \bmod p$, а масштабированный. Для перевода значения t в немасштабированный вид остатка по модулю p, необходимо вычислить $t \cdot R \bmod p$ или, что эквивалентно, применить алгоритм `MontMult(`$(t \bmod p)\left(R^2 \bmod p\right)$`)`. Однако представление Монтгомери является полноценным представлением чисел, допускающим выполне-

Алгоритм 2.30. Нахождение остатка от деления

Вход: X, p

Выход: $X \bmod p$

1: $R_0 = X$
2: $n = 1$
3: $N = \texttt{LeftShift}(p, n)$
4: **До тех пор пока** $N < R_0$ **выполнять**
5: $\quad n = n + 1$
6: $\quad N = \texttt{LeftShift}(p, n)$
7: **Конец цикла**
8: $n = n - 1$
9: $N = \texttt{LeftShift}(p, n)$
10: **Цикл от** $i = 1$ **до** n **выполнять**
11: $\quad R_i = R_{i-1} - N$
12: $\quad$ **Если** $R_i < 0$ **тогда**
13: $\quad\quad R_i = R_{i-1}$
14: $\quad$ **Конец условия**
15: $\quad N = \texttt{RightShift}(N, 1)$
16: **Конец цикла**
17: **Возвратить** R_n

Алгоритм 2.31. Произведение Монтгомери

Вход: A, B, R, p'

Выход: $\texttt{MontMult}(A \cdot B)$

1: $m := (A \cdot B \bmod R)\, p' \bmod R$
2: $t := (A \cdot B + m \cdot p)/R$
3: **Если** $t \geq p$ **тогда**
4: $\quad$ **Возвратить** $t - p$
5: **иначе**
6: $\quad$ **Возвратить** t
7: **Конец условия**

ние операций над масштабированными значениями, поэтому перевод к немасштабированному виду необходим лишь в конце вычислений.

Многие современные высокоскоростные двоичные процессоры используют специализированные инструкции, такие как умножение с накоплением. Кроме того, существуют специальные реализации функций умножения с накоплением для двоичных компьютеров, например, «объединенные» блоки умножения с накоплением (multiply–accumulate, MAC). Причина в том, что многие компьютерные вычисления требуют умножения двух операндов и добавления третьего операнда к результату умножения. Операция

$$P = C + A \times B \tag{2.54}$$

является основной для задач цифровой фильтрации [57;71], таких как очистка от шума, эквализация, задач, возникающих при реализации нейронных сетей [36] и др.

В статье [3] рассмотрена реализация усеченного MAC (truncated multiply-accumulate, TMAC) для конвейерного выполнения умножения с накоплением в рамках цифровой фильтрации. Для получения итогового результата, согласно формуле (2.54), нет необходимости выполнять полное умножение $A \times B$. Вместо этого достаточно использовать генератор k частичных произведений, где $k = \lfloor \log_2 B \rfloor + 1$ бит $-$ размерность коэффициента B, и дерево сумматоров с сохранением переноса (carry-save adder, CSA) [56] без использования на каждом шаге конвейера конечного сложения сумматором Когге-Стоуна (Kogge–Stone adder, KSA) [39].

В статье [36] был предложен блок MAC, который поддерживает три режима умножения: (1) четыре умножения 4-бита $\times$ 16-бит, (2) два умножения 8- бит $\times$ 16-бит и (3) одно умножение 16-бит $\times$ 16-бит. Несмотря на то, что данный умножитель с накоплением полностью использует свои внутренние ресурсы для операций 4-бита $\times$ 16-бит, 8-бит $\times$ 16-бит и 16-бит $\times$ 16-бит, выполнение умножения 16-бит $\times$ n-бит, n = 8, 7, ... является неэффективным, так как оно не полностью использует внутреннюю логику, соответствующую старшим разрядам точности или старшим битам операнда с фиксированной точностью. Более того, в структуре данного умножителя с накоплением происходит существенная трата вычислительных ресурсов даже при незначительной разнице между размерами входных операндов, например, при умножении 9 бит $\times$ 7 бит.

Реализация MAC доступна, в частности, в рамках среды разработки Vivado Design Suite фирмы Xilinx и поддерживает умножение беззнаковых чисел размерностью от 1 до 52 бит и знаковых от 2 до 53 бит, а также сложение или вычитание беззнаковых чисел размерностью от 1 до 105 бит и от 2 до 106 бит для чисел со знаком. При этом размерность выхода может быть задана вручную и быть меньше необходимой [53]. В общем случае же размерность результата умножения равна сумме размерностей операндов, а размерность результата сложения равна максимальной размерности операнда, увеличенной на один.

Повышение эффективности умножения с накоплением возможно за счет использования непозиционных систем счисления, в которых нет необходимости учитывать межразрядные переносы. При выборе системы счисления можно непосредственно снизить количество операций, длину операндов, количество и/или длину глобальных соединений, что может привести к снижению площади, задержки и рассеивания мощности [59]. Реализация умножения с накоплением в СОК рассмотрена в Параграфе 3.5.

2.4.1 Метод нахождения остатка при модулярном умножении

Введем метод, позволяющий эффективно производить модулярное умножение. Данный метод относится к двухэтапным и сводится к вычислению произведения $X = A \times B$ методом Карацубы-Офмана или Бута и нахождению остатка от полученного числа по модулю p, т.е. $X \bmod p$.

Эффективная реализация нахождения остатка от деления $X = (x_1, \ldots, x_n)$ по модулю простого числа p в СОК может быть получена из Китайской теоремы об остатках, выраженной формулой (2.7), и подхода, описанного в статье [141] с помощью следующего математического аппарата:

$$\frac{X}{P} = \left| \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i \right|_1 = \left| \sum_{i=1}^{n} k_i x_i \right|_1 = p \left| \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p \cdot p_i} x_i \right|_1 = \\ = \left| p \sum_{i=1}^{n} \overline{k_i} x_i \right|_1 = p \sum_{i=1}^{n} \overline{k_i} x_i - \left\lfloor \sum_{i=1}^{n} k_i x_i \right\rfloor, \quad (2.55)$$

где $k_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i}$, $\overline{k_i} = \frac{\left|P_i^{-1}\right|_{p_i}}{p \cdot p_i}$ — константы выбранной системы.

Из формулы (2.55) следует, что X можно представить в виде

$$X = \left(p \sum_{i=1}^{n} \overline{k_i} x_i - \left\lfloor \sum_{i=1}^{n} k_i x_i \right\rfloor \right) \cdot P. \tag{2.56}$$

Используя формулу (2.56) вычислим значение X/p

$$\frac{X}{p} = \left(\sum_{i=1}^{n} \overline{k_i} x_i - \frac{1}{p} \left\lfloor \sum_{i=1}^{n} k_i x_i \right\rfloor \right) \cdot P.$$

Отсюда, значение $X \bmod p$ можно найти по формуле

$$X \bmod p = X - \left\lfloor \frac{X}{p} \right\rfloor \cdot p = X - \left\lfloor \left(\sum_{i=1}^{n} \overline{k_i} x_i - \frac{1}{p} \left\lfloor \sum_{i=1}^{n} k_i x_i \right\rfloor \right) \cdot P \right\rfloor \cdot p. \tag{2.57}$$

Для эффективной работы модулярной арифметики необходима её целочисленная реализация, при этом каждая вещественная константа умножается на 2^N, где N — число двоичных знаков после запятой, обеспечивающее необходимую точность вычислений. После этого каждое полученное значение округляется вверх до следующего целого числа и дальнейшие вычисления производятся в кольце классов вычетов по модулю 2^N.

Используя оценку из статьи [8], получим

$$N = \left\lceil \log_2 \left(P \cdot \left(\sum_{i=1}^{n} (p_i - 1) \right) \right) \right\rceil.$$

Тогда формула (2.57) примет следующий вид

$$X \bmod p = X - K \cdot p, K = \left\lfloor \left(\sum_{i=1}^{n} \tilde{k}_i x_i - \mu \left\lfloor \frac{\sum_{i=1}^{n} \hat{k}_i x_i}{2^N} \right\rfloor \right) \frac{P}{2^N} \right\rfloor, \tag{2.58}$$

где $\tilde{k}_i = \left\lfloor 2^N \frac{\left|P_i^{-1}\right|_{p_i}}{p_i \cdot p} \right\rfloor$, $\hat{k}_i = \left\lceil 2^N \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} \right\rceil$, $\mu = \left\lfloor \frac{2^N}{p} \right\rfloor$, $N = \lceil \log_2 (P \cdot (\sum_{i=1}^{n} (p_i - 1))) \rceil$.

Данный метод может быть реализован Алгоритмом 2.32.

Результат работы Алгоритма 2.32 — $T = X \bmod p$ удовлетворяет неравенству $0 \leq T < 2p$, что аналогично алгоритму Монтгомери, который будет рассмотрен ранее. Покажем работу данного метода на примере.

Пример 31. *Возьмем систему остаточных классов* $\{p_1, p_2, p_3, p_4\} = \{2, 3, 5, 7\}$ *и два множителя* $A = 4 = (0, 1, 4, 4)$ *и* $B = 6 = (0, 0, 1, 6)$. *Вычислим произведение* $A \cdot B$ *по модулю* $p = 13$.

Алгоритм 2.32. Метод нахождения остатка

Вход: X, p

Выход: $T = X \bmod p$

1: $S = 0$, $\bar{S} = 0$
2: **Цикл от** $i = 1$ **до** n **выполнять**
3: $\quad k_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i}$,
4: $\quad \bar{k}_i = \frac{k_i}{p}$
5: $\quad S = S + k_i x_i$
6: $\quad \bar{S} = \bar{S} + \overline{k_i} x_i$
7: **Конец цикла**
8: $S = \bar{S} - \frac{\lfloor S \rfloor}{p}$
9: $S = \lfloor S \cdot P \rceil \cdot p$
10: $T = X - S$
11: **Возвратить** T

Для сокращения записи воспользуемся формулой (2.57)*. Величина динамического диапазона выбранной СОК* $P = \prod_{i=1}^{4} p_i = 210$*. Тогда*

$$P_1 = \frac{P}{p_1} = 105, \quad \left|P_1^{-1}\right|_{p_1} = 1, \quad k_1 = \frac{\left|P_1^{-1}\right|_{p_1}}{p_1} = \frac{1}{2}, \quad \overline{k_1} = \frac{k_1}{p} = \frac{1}{26},$$
$$P_2 = \frac{P}{p_2} = 70, \quad \left|P_2^{-1}\right|_{p_2} = 1, \quad k_2 = \frac{\left|P_2^{-1}\right|_{p_2}}{p_2} = \frac{1}{3}, \quad \overline{k_2} = \frac{k_2}{p} = \frac{1}{39},$$
$$P_3 = \frac{P}{p_3} = 42, \quad \left|P_3^{-1}\right|_{p_3} = 3, \quad k_3 = \frac{\left|P_3^{-1}\right|_{p_3}}{p_3} = \frac{3}{5}, \quad \overline{k_3} = \frac{k_3}{p} = \frac{3}{65},$$
$$P_4 = \frac{P}{p_4} = 30, \quad \left|P_4^{-1}\right|_{p_4} = 4, \quad k_4 = \frac{\left|P_4^{-1}\right|_{p_4}}{p_4} = \frac{4}{7}, \quad \overline{k_4} = \frac{k_4}{p} = \frac{4}{91}.$$

Так как $A \cdot B = 24 = (0, 0, 4, 3)$*. Тогда*

$$A \cdot B \bmod p = 24 - \left\lfloor \left(\left(0 \cdot \frac{1}{26} + 0 \cdot \frac{1}{39} + 4 \cdot \frac{3}{65} + 3 \cdot \frac{4}{91} \right) - \right.$$
$$\left. - \left\lfloor 0 \cdot \frac{1}{2} + 0 \cdot \frac{1}{3} + 4 \cdot \frac{3}{5} + 3 \cdot \frac{4}{7} \right\rfloor \cdot \frac{1}{13} \right) \cdot 210 \Bigg\rfloor \cdot 13 =$$
$$= 24 - \left\lfloor \left(\frac{144}{455} - \left[\frac{144}{35} \right] \cdot \frac{1}{13} \right) \cdot 210 \right\rfloor \cdot 13 = 24 - \left\lfloor \left(\frac{144}{455} - \frac{4}{13} \right) \cdot 210 \right\rfloor \cdot 13 =$$
$$= 24 - \left\lfloor \frac{24}{13} \right\rfloor \cdot 13 = 24 - 1 \cdot 13 = 11.$$

Поскольку $4 \cdot 6 \bmod 13 = 11$*, метод работает корректно.*

Реализация данного метода возможна с использованием нейронной сети конечного кольца [123]. Преимущество данного метода перед аналогами заключается в отсутствии операций сравнения, деления и нахождения остатка, что позволяет получить выигрыш в производительности.

В данном параграфе были рассмотрены различные методы модульного умножения $A \times B \bmod p$, которые можно разделить на две категории, согласно подходу к реализации: получить произведение, а затем найти остаток от деления; или находить остаток в процессе вычисления произведения. Преимущество методов первой категории заключается в том, что можно использовать любые существующие методы для умножения и нахождения остатка. Отметим, что все эффективные методы в той или иной степени основаны на методах умножения Карацубы-Офмана и Бута и методе нахождения остатка Барретта. Другим недостатком использования методов первой категории является то, что произведение, как правило, имеет большой размер и, следовательно, требует много места для его хранения перед нахождением остатка. Также возможны случаи выхода результата за диапазон СОК. Напротив, методы, которые чередуют этапы умножения и нахождения остатка для получения модульного произведения, не должны хранить произведение. Однако, большинство из них основаны на методе Монтгомери, который дает масштабированное модульное произведение и требует дополнительных вычислительных ресурсов.

Для повышения производительности модулярного умножения введен метод умножения с нахождением остатка результата произведения, который за счет отсутствия операций сравнения, деления и нахождения остатка, позволяет получить выигрыш в производительности.

2.5 Модификация метода обнаружения, локализации и исправления ошибок в СОК

Одним из подходов к повышению отказоустойчивости вычислений является использование избыточной системы остаточных классов. Выполнение операций в ИСОК происходит параллельно без межразрядных переносов, что позволяет эффективно реализовать сложение, вычитание и умножение. Независимость выполнения действий над каждым модулем обеспечивает внутренние коррек-

тирующие способности модулярного кода, поскольку каждый остаток содержит информацию о всём числе.

Рассмотрим модификацию метода проекций с использованием приближенного метода на основе КТО. Для обнаружения и исправления одиночной ошибки в СОК рассмотрим добавление двух избыточных модулей p_{n+1} и p_{n+2}, тогда диапазон избыточной системы остаточных классов составит $\overline{P} = \prod_{i=1}^{n+2} p_i = P \cdot p_{n+1} \cdot p_{n+2}$, где $P = \prod_{i=1}^{n} p_i$ — рабочий диапазон. Разрешенным считается число $A = (\alpha_1, \alpha_2, \ldots, \alpha_n, \alpha_{n+1}, \alpha_{n+2})$, если $A \in [0, P)$, в случае же $A \in \left[P, \overline{P}\right)$ можно сказать, что число содержит ошибку. При этом P представляется в избыточной СОК и очевидно, что $P = (0, \ldots, 0, \pi_{n+1}, \pi_{n+2})$, где $\pi_{n+1} = P \bmod p_{n+1}$, $\pi_{n+2} = P \bmod p_{n+2}$.

Таким образом, для определения и исправления ошибок нужно найти позиционную характеристику числа, примеры которых рассмотрены в параграфе 2.3.

Для эффективной аппаратной реализации берут относительное приближенное значение

$$\frac{A}{\overline{P}} = \left| \sum_{i=1}^{n+2} \frac{\left|\overline{P}_i^{-1}\right|}{p_i} \alpha_i \right|_1 = \left| \sum_{i=1}^{n+2} k_i \alpha_i \right|_1,$$

где $k_i = \frac{\left|\overline{P}_i^{-1}\right|}{p_i}$ — константа выбранной СОК, $\overline{P}_i = \frac{\overline{P}}{p_i}$, $\left|\overline{P}_i^{-1}\right|$ — мультипликативная инверсия. В данном случае для определения ошибки используется сравнение относительной величины $\frac{A}{\overline{P}}$ с константой $\frac{P}{\overline{P}}$.

Для обнаружения и исправления ошибок применим метод проекций, который заключается в следующем. Предполагают, что ошибка допущена по первому модулю p_1 и повторяют вычисления без участия этого модуля, т.е. берут в качестве избыточного диапазона $\overline{P}_1 = \frac{\overline{P}}{p_1}$, для нового диапазона вычисляют новые константы k_i, и в конечном счете сравнивают $\frac{A}{\overline{P}_1}$ с константой $\frac{P}{\overline{P}_1}$. В случае, если $\frac{A}{\overline{P}_1} < \frac{P}{\overline{P}_1}$, то считают, что ошибка произошла по модулю p_1 и для нахождения точного значения остатка A по модулю p_1 умножают $\frac{A}{\overline{P}_1}$ на $\overline{P}_1$ и находят остаток по модулю p_1. В случае $\frac{A}{\overline{P}_1} > \frac{P}{\overline{P}_1}$ можно сказать, что ошибка еще присутствует и для её обнаружения и исправления вместо p_1 исключают p_2 и проводят аналогичную проверку. Такие проекции строятся по всем модулям $p_1, \ldots, p_{n+2}$.

Значения констант k_i берут с точностью N, необходимой для корректных вычислений [116].

Использование приближенного метода позволяет заменить вычислительно сложную операцию нахождения остатка от деления на диапазон системы взятием наименьших значащих бит числа, что дает возможность снизить вычислительную сложность с квадратичной до линейно-логарифмической сложности.

Рассмотрим модификацию Алгоритма 1.1 обнаружения и локализации ошибки в системе остаточных классов, рассмотренного в Параграфе 1.2.

Введем метод обнаружения и локализации ошибок, основанный на использовании несбалансированной системы остаточных классов, в котором нет необходимости вычисления позиционной характеристики для каждой из проекций. Пусть задана система остаточных классов с одним контрольным основанием $p_1 < p_2 < p_3 < \ldots < p_n < p_{n+1}$, при этом считается, что контрольное основание надежное и не может содержать ошибки. Тогда введем Алгоритм 2.33.

Алгоритм 2.33. Коррекция ошибки на основе метода проекций и КТО

Вход: $X' = (x'_1, x'_2, \ldots, x'_n, x'_{n+1})$

Выход: X

Данные в памяти: $\{p_1, p_2, \ldots, p_{n+1}\}$, $P = \prod_{i=1}^{n} p_i$, $\overline{P} = p_n \cdot P$
$w_i = \left|\overline{P}_i^{-1}\right|_{p_i}$, $\overline{P}_i = \overline{P}/p_i$, $i \in [1,n+1]$

1: $S = 0$
2: **Цикл от** $i = 1$ **до** $n+1$ **выполнять**
3: $\quad S = S + x'_i \cdot w_i \cdot \overline{P}_i$
4: **Конец цикла**
5: $X = S \bmod \overline{P}$
6: **Если** $X < P$ **тогда**
7: $\quad$ **Возвратить** X
8: **иначе**
9: $\quad k = 1$
10: $\quad X = S \bmod \overline{P}_1$
11: $\quad$ **До тех пор пока** $X > P$ `AND` $k \leq n$ **выполнять**
12: $\quad\quad k = k + 1$
13: $\quad\quad X = S \bmod \overline{P}_k$
14: $\quad$ **Конец цикла**
15: **Конец условия**

Теорема 2.5.1. *Если $p_{n+1} > p_n \cdot p_{n-1}$, то Алгоритм 2.33 корректен.*

Доказательство. Пусть ошибка $E = (0, \ldots, e_k, \ldots, 0)$ произошла по k-му основанию, где $k \in [1,n]$. Для доказательства теоремы достаточно показать, что для всех $X \in [0,P)$, $j \in [1,n]$ и $j \neq k$, выполняется следующее неравенство:

$$P \leq \left| X + e_k \cdot w_k \cdot \overline{P}_k \right|_{\overline{P}_j}. \tag{2.59}$$

Вычислим $\left| X + e_k \cdot w_k \cdot \overline{P}_k \right|_{\overline{P}_j}$ используя формулу $|a|_m = a - \left\lfloor \frac{a}{m} \right\rfloor \cdot m$, получим:

$$\left| X + e_k \cdot w_k \cdot \overline{P}_k \right|_{\overline{P}_j} = X + e_k \cdot w_k \cdot \overline{P}_k - \left\lfloor \frac{X + e_k \cdot w_k \cdot \overline{P}_k}{\overline{P}_j} \right\rfloor \cdot \overline{P}_j. \tag{2.60}$$

Так как $\frac{\overline{P}_k}{\overline{P}_j} = \frac{p_j}{p_k}$, то представим выражение $\left\lfloor \frac{X + e_k \cdot w_k \cdot \overline{P}_k}{\overline{P}_j} \right\rfloor$ в следующем виде:

$$\left\lfloor \frac{X + e_k \cdot w_k \cdot \overline{P}_k}{\overline{P}_j} \right\rfloor = \left\lfloor \frac{X}{\overline{P}_j} + \frac{e_k \cdot w_k \cdot \overline{P}_k}{\overline{P}_j} \right\rfloor = \left\lfloor \frac{X}{\overline{P}_j} + \frac{e_k \cdot w_k \cdot p_j}{p_k} \right\rfloor. \tag{2.61}$$

Оценим значение величины $\frac{X}{\overline{P}_j}$ для всех $j \in [1,n]$ и $j \neq k$, получим

$$\frac{X}{\overline{P}_j} \leq \frac{P-1}{\overline{P}_j} = \frac{P}{\overline{P}_j} - \frac{1}{\overline{P}_j}. \tag{2.62}$$

Так как $\frac{P}{\overline{P}_j} = \frac{p_j}{p_{n+1}}$ и по условию теоремы $p_{n+1} > p_n \cdot p_{n-1}$, то $\frac{p_j}{p_{n+1}} < \frac{1}{p_k}$ для всех $j \in [1,n]$ и $j \neq k$, следовательно, уравнение (2.62) примет вид:

$$\frac{X}{\overline{P}_j} < \frac{1}{p_k}. \tag{2.63}$$

Учитывая неравенство (2.63), выражение (2.61) примет вид:

$$\left\lfloor \frac{X + e_k \cdot w_k \cdot \overline{P}_k}{\overline{P}_j} \right\rfloor = \left\lfloor \frac{e_k \cdot w_k \cdot p_j}{p_k} \right\rfloor. \tag{2.64}$$

Подставляя выражения (2.60) и (2.64) в неравенство (2.59), получим:

$$P \leq X + e_k \cdot w_k \cdot \overline{P}_k - \left\lfloor \frac{e_k \cdot w_k \cdot p_j}{p_k} \right\rfloor \cdot \overline{P}_j. \tag{2.65}$$

Разделим левую часть неравенства (2.65) на положительное число P, получим:

$$1 \leq X + e_k \cdot w_k \cdot \frac{\overline{P}_k}{P} - \left\lfloor \frac{e_k \cdot w_k \cdot p_j}{p_k} \right\rfloor \cdot \frac{\overline{P}_j}{P}. \tag{2.66}$$

Так как $\frac{\overline{P}_k}{P} = \frac{p_{n+1}}{p_k}$ и $\frac{\overline{P}_j}{P} = \frac{p_{n+1}}{p_j}$, то выражение (2.66) примет вид:

$$1 \leq X + e_k \cdot w_k \cdot \frac{p_{n+1}}{p_k} - \left\lfloor \frac{e_k \cdot w_k \cdot p_j}{p_k} \right\rfloor \cdot \frac{p_{n+1}}{p_j} \tag{2.67}$$

Умножим правую и левую часть неравенства (2.65) на положительное число $\frac{p_k \cdot p_j}{p_{n+1}}$, получим:

$$\frac{p_k \cdot p_j}{p_{n+1}} \leq X \cdot \frac{p_k \cdot p_j}{p_{n+1}} + e_k \cdot w_k \cdot p_j - \left\lfloor \frac{e_k \cdot w_k \cdot p_j}{p_k} \right\rfloor \cdot p_k. \tag{2.68}$$

Так как

$$e_k \cdot w_k \cdot p_j - \left\lfloor \frac{e_k \cdot w_k \cdot p_j}{p_k} \right\rfloor \cdot p_k = \left| e_k \cdot w_k \cdot p_j \right|_{p_k},$$

то формула (2.68) примет вид:

$$\frac{p_k \cdot p_j}{p_{n+1}} \leq X \cdot \frac{p_k \cdot p_j}{p_{n+1}} + \left| e_k \cdot w_k \cdot p_j \right|_{p_k}. \tag{2.69}$$

Учитывая, что $X \in [0,P-1)$ и $1 \leq \left| e_k \cdot w_k \cdot p_j \right|_{p_k} \leq p_k - 1$, то неравенство (2.69) выполняется, если выполняется для всех $j \in [1,n]$ и $j \neq k$, следующее, условие:

$$\frac{p_k \cdot p_j}{p_{n+1}} \leq 1. \tag{2.70}$$

Так как модули СОК удовлетворяют условию $p_1 < p_2 < p_3 < \ldots < p_n$, то из неравенства (2.70), следует, что необходимым и достаточным условием выполнения неравенство (2.59), является: $p_{n+1} \geq p_n p_{n-1}$. Теорема доказана. □

Рассмотрим пример обнаружения и исправления ошибки в СОК с одним контрольным основанием.

Пример 32. *Выберем систему оснований СОК, удовлетворяющую теореме 2.5.1:* $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, $n = 4$. *Контрольное основание* $p_5 = 37 > 7 \cdot 5$. *Искомое число* $X = 53 = (1, 2, 3, 4, 16)$.

Параметры СОК:

$$P = 2 \cdot 3 \cdot 5 \cdot 7 = 210 - \textit{рабочий диапазон};$$
$$\overline{P} = p_5 \cdot P = 35 \cdot 210 = 7770 - \textit{полный диапазон СОК};$$
$$\overline{P}_1 = \frac{\overline{P}}{p_1} = 3885, \overline{P}_2 = \frac{\overline{P}}{p_2} = 2590, \overline{P}_3 = \frac{\overline{P}}{p_3} = 1554,$$
$$\overline{P}_4 = \frac{\overline{P}}{p_4} = 1110, \overline{P}_5 = \frac{\overline{P}}{p_5} = 210.$$
$$w_1 = \left|\overline{P}_1^{-1}\right|_{p_1} = 1, w_2 = \left|\overline{P}_2^{-1}\right|_{p_2} = 1, w_3 = \left|\overline{P}_3^{-1}\right|_{p_3} = 4,$$
$$w_4 = \left|\overline{P}_4^{-1}\right|_{p_4} = 2, w_5 = \left|\overline{P}_5^{-1}\right|_{p_5} = 3.$$

Введем вектор ошибки $E = (0, 0, 1, 0, 0)$, $X' = E + X = (1, 2, 4, 4, 16)$.

Вычислим

$$S = \left|\sum_{i=1}^{n+1} w_i \overline{P}_i x_i'\right|_{\overline{P}} =$$
$$= |1 \cdot 3885 \cdot 1 + 1 \cdot \ 2590 \cdot 2 + 4 \cdot 1554 \cdot 4 + 2 \cdot 1110 \cdot 4 + 3 \cdot \ 210 \cdot 16|_{7770} = 6269.$$

Так как S=6269>210, следовательно, есть ошибка, вычислим ее: $|S|_{\overline{P}_1} = 2384 > P$, $|S|_{\overline{P}_2} = 1089 > P$, $|S|_{\overline{P}_3} = 53 < P$, *следовательно* $X = 53$.

Таким образом, введение единственного избыточного основания позволяет обнаружить и исправить одиночную ошибку.

Рассмотрим применение приближенного метода к Алгоритму 2.33. Недостатком вычислений по данному алгоритму является необходимость нахождения остатка по большому модулю. Введем Алгоритм 2.34, позволяющий исправить ошибку по рабочему модулю в СОК с одним избыточным основанием $p_{n+1} > p_n \cdot p_{n-1}$ [42].

Алгоритм 2.34. Коррекция ошибки на основе метода проекций и приближенной КТО

Вход: $X' = (x'_1, x'_2, \ldots, x'_n, x'_{n+1})$

Выход: X

Данные в памяти: $\{p_1, p_2, \ldots, p_{n+1}\}$, $P = \prod_{i=1}^{n} p_i$,
$\overline{P} = p_n \cdot P$, $\pi = P \bmod p_{n+1}$
$\overline{P}_i = \overline{P}/p_i$, $w_i = \frac{\left|\overline{P}_i^{-1}\right|_{p_i}}{p_i}$, $i \in [1,n+1]$
$\overline{P}_{i,j} = \overline{P}_i/p_j$, $w_{i,j} = \frac{\left|\overline{P}_{i,j}^{-1}\right|_{p_j}}{p_j}$, $i \in [1,n]$, $j \in [1,n+1]$, $i \neq j$

1: $X = \left|\sum_{i=1}^{n+1} x'_i \cdot w_i\right|_1$
2: $W = |\pi \cdot w_{n+1}|_1$
3: **Если** $X < W$ **тогда**
4: **Возвратить** $X \cdot \overline{P}$
5: **иначе**
6: **Цикл от** $i = 1$ **до** n **выполнять**
7: $X_i = \left|\sum_{j=1, j\neq i}^{n+1} x'_j \cdot w_{i,j}\right|_1$
8: $W_i = |\pi \cdot w_{i,n+1}|_1$
9: **Если** $X_i < W_i$ **тогда**
10: **Возвратить** $X_i \cdot \overline{P}_i$
11: **Конец условия**
12: **Конец цикла**
13: **Конец условия**

Рассмотрим пример работы Алгоритма 2.34 для СОК $\{3, 5, 7, 37\}$ и числа $X' = (1, 2, 3, 17) = 1312$. Данные, которые хранятся в памяти:

$$P = 105, \overline{P} = 3885, \pi = 31,$$
$$P_1 = 1295, P_2 = 777, P_3 = 555, P_4 = 105,$$
$$P_{1,2} = P_{2,1} = 259, P_{1,3} = P_{3,1} = 185, P_{1,4} = P_{4,1} = 35, P_{2,3} = P_{3,2} = 111,$$
$$P_{2,4} = P_{4,2} = 21, P_{3,4} = P_{4,3} = 15,$$
$$w_1 = \frac{2}{3}, w_2 = \frac{3}{5}, w_3 = \frac{4}{7}, w_4 = \frac{6}{37},$$
$$w_{1,2} = \frac{4}{5}, w_{1,3} = \frac{5}{7}, w_{1,4} = \frac{18}{37},$$
$$w_{2,1} = \frac{1}{3}, w_{2,3} = \frac{6}{7}, w_{2,4} = \frac{30}{37},$$
$$w_{3,1} = \frac{2}{3}, w_{3,2} = \frac{1}{5}, w_{3,4} = \frac{5}{37}.$$

Тогда

$$X = \left|1 \cdot \frac{2}{3} + 3 \cdot \frac{3}{5} + 3 \cdot \frac{4}{7} + 15 \cdot \frac{6}{37}\right|_1 = \frac{2383}{3885}, W = \left|31 \cdot \frac{6}{37}\right|_1 = \frac{1}{37}.$$

Поскольку $2383/3885 > 1/37$, то число содержит ошибку. Построим проекции.

Тогда первая проекция

$$X_1 = \left|3 \cdot \frac{4}{5} + 3 \cdot \frac{5}{7} + 15 \cdot \frac{18}{37}\right|_1 = \frac{1088}{1295}, W_1 = \left|31 \cdot \frac{18}{37}\right|_1 = \frac{3}{37}.$$

Поскольку $1088/3885 > 3/37$, то число содержит ошибку. Вторая проекция

$$X_2 = \left|1 \cdot \frac{1}{3} + 3 \cdot \frac{6}{7} + 15 \cdot \frac{30}{37}\right|_1 = \frac{52}{777}, W_2 = \left|31 \cdot \frac{30}{37}\right|_1 = \frac{5}{37}.$$

Поскольку $52/777 < 5/37$, то число не содержит ошибку.

Третья проекция

$$X_3 = \left|1 \cdot \frac{2}{3} + 3 \cdot \frac{1}{5} + 15 \cdot \frac{5}{37}\right|_1 = \frac{373}{555}, W_3 = \left|31 \cdot \frac{5}{37}\right|_1 = \frac{7}{37}.$$

Поскольку $373/555 > 7/37$, то число содержит ошибку.

Таким образом ошибка была по второму модулю. Для восстановления числа необходимо $52/777$ умножить на $P_2 = 777$, таким образом, корректное число равно 52.

Данный подход также может содержать ошибку только по рабочим модулям.

2.6 Выводы по второй главе

Система остаточных классов, как одна из непозиционных систем счисления, позволяет эффективно реализовывать высокопроизводительные вычисления за счет отсутствия переносов между разрядами. Однако ряд операций в СОК, называемых немодульными, является ресурсоемким и требует тщательной оптимизации.

Рассмотрена проблема выбора модулей СОК для перевода чисел из позиционной системы счисления, рассмотрены следующие алгоритмы нахождения

остатка при делении на модули специального вида 2^n, $2^n - 1$, $2^n + 1$: нейронная сеть конечного кольца, алгоритмы на основе периода и полупериода числа. Проблемой большинства этих алгоритмов является получение не искомого результата, а сравнимого с ним. Для решения этой проблемы предложен алгоритм на основе периода и полупериода числа, позволяющий при переводе из ПСС в СОК найти наименьший неотрицательный вычет.

Исследованы методы перевода из СОК в позиционную систему счисления на основе Китайской теоремы об остатках, приближенного метода на основе КТО, обобщенной позиционной системы счисления, функции ядра и диагональной функции. Рассмотрены особенности этих методов, установлены их достоинства и недостатки. Предложен новый алгоритм на основе ОПСС для перевода чисел в позиционную систему счисления и расширения оснований, преимуществами которого являются простота и универсальность.

Рассмотрена проблема сравнения чисел в СОК и определения знака числа. Реализация алгоритма сравнения чисел в СОК состоит из двух этапов. Первый этап — вычисление позиционной характеристики модулярных чисел $X = (x_1, \ldots, x_n)$ и $Y = (y_1, \ldots, y_n)$. Второй этап — сравнение позиционных характеристик ПХ(X) и ПХ(Y) модулярных чисел в позиционной системе счисления. Получена оценка необходимой точности при округлении констант строго возрастающей приближенной функции на основе КТО для сравнения чисел, предложен метод сравнения чисел, который позволяет уменьшить размер операндов по сравнению с алгоритмами из работ [75; 86]. Также получена модификация функции ядра Акушского с заданными свойствами, разработаны алгоритмы выбора параметров для функции ядра без критических ядер, на основе которой построен алгоритм сравнения чисел и определения знака числа. Предложены методы сравнения чисел в СОК с чётным и нечётным динамическим диапазоном, которые позволяют получить искомый результат без операции деления на большой модуль.

Исследованы следующие методы модульного умножения: нейронная сеть конечного кольца, метод Карацубы-Офмана, метод Бута, алгоритмы Монтгомери. Для повышения производительности модулярного умножения введен метод нахождения остатка результата произведения на произвольный простой модуль, который за счет отсутствия операций сравнения, деления и нахождения остатка, позволяет получить выигрыш в производительности. Также рассмотрена реализация умножения с накоплением (MAC) в системе остаточных классов.

Рассмотрен метод проекций на основе Китайской теоремы об остатках с дробными значениями. Введен метод коррекции одиночной ошибки с одним избыточным модулем СОК.

Таким образом, реализованы все операции, необходимые для построения отказоустойчивой системы обработки данных, работающей в системе остаточных классов.

Глава 3. Разработка программного комплекса выполнения немодульных операций в модулярном коде для проектирования отказоустойчивых вычислительных узлов распределенной среды

3.1 Программный комплекс для проектирования отказоустойчивой вычислительной системы, работающей в модулярном коде

При разработке вычислительных узлов распределенной среды для современных информационных вычислительных систем особое внимание уделяется техническим характеристикам: скорости работы, объему оборудования и т.д.

Использование непозиционных систем счисления, таких как система остаточных классов, позволяет выполнять сложение и умножение чисел по параллельным вычислительным каналам без переноса разрядов между каналами, что позволяет повысить скорость выполнения арифметических операций. При этом размеры модулей СОК могут не совпадать с разрядностью современных ЭВМ и для достижения максимальной производительности необходима разработка специализированных вычислительных узлов распределенной среды на базе FPGA и ASIC, которые позволят производить вычисления над числами заданной разрядности.

Работа вычислительной системы, работающей в модулярном коде, в общем виде показана на рисунке 3.1.

Данная вычислительная система содержит блоки перевода из позиционной системы счисления в систему остаточных классов, блоки выполнения модульных операций сложения, умножения и вычитания по модулям СОК, блок вычисления позиционной характеристики для выполнения сравнения чисел, определения знака, деления, а также для перевода из системы остаточных классов в позиционную систему счисления.

Для реализации алгоритма проектирования отказоустойчивой вычислительной системы на основе рассмотренных во второй главе методов и алгоритмов разработан комплекс программ для работы с модулярным кодом, структура которого показана на рисунке 3.2. В открытом доступе есть библиотека vscripts.ru, в которой представлены генераторы Verilog-модулей для реализации базовых операций в системе остаточных классов, однако блоков реализации

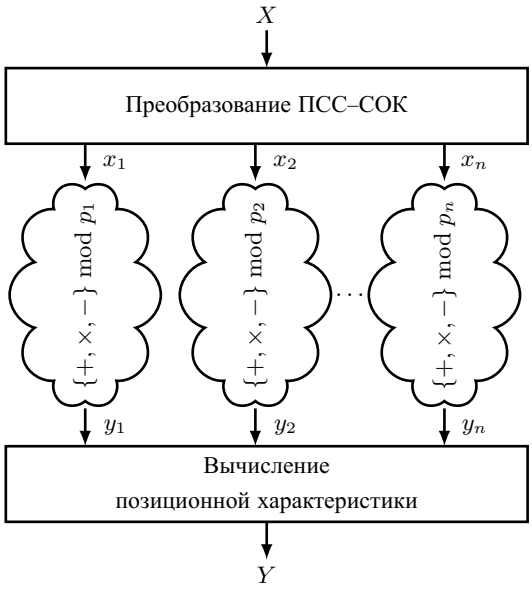

Рисунок 3.1 — Структурная схема отказоустойчивой вычислительной системы, работающей в модулярном коде

сравнения чисел и обнаружения и коррекции ошибок там нет. Доступные коды для прямого и обратного преобразования были использованы в качестве аналогов для сравнения, отсутствующие методы были реализованы самостоятельно.

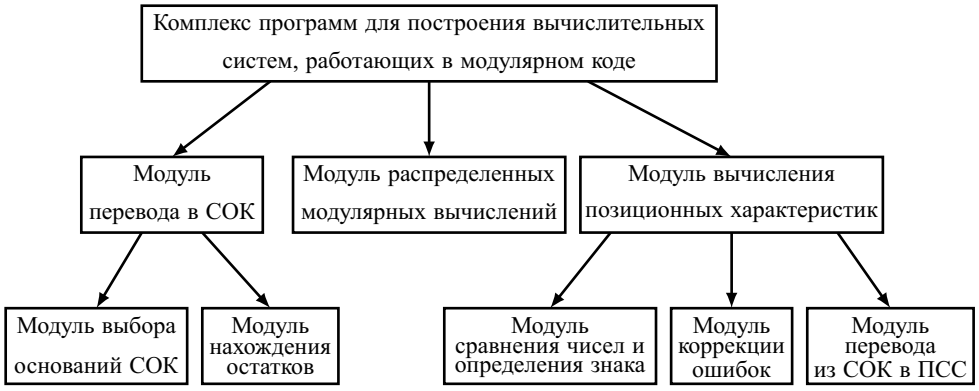

Рисунок 3.2 — Структура программного комплекса для построения вычислительных систем, работающих в модулярном коде

Проектирование архитектуры отказоустойчивой вычислительной системы, работающей в модулярном коде, начинается с выбора набора модулей СОК. Как показано в главе 2, выбор модулей влияет как на вычислительную сложность операций перевода из ПСС в СОК, из СОК в ПСС, так и на корректирующие свойства отказоустойчивого кода. Добавление двух избыточных модулей, или одного специально выбранного, для коррекции одиночных ошибок, должно быть

выбрано заранее. Также необходимо учитывать рабочий диапазон СОК, в котором будут входить результаты вычислений.

Таким образом, первым шагом алгоритма проектирования отказоустойчивой вычислительной системы, работающей в модулярном коде, является выбор набора модулей СОК, который в соответствии с рисунком 3.2 представлен модулем выбора оснований СОК. Для целей подбора модулей была разработана программа для ЭВМ [131]. Программа предназначена для анализа структуры динамического диапазона и оптимального выбора оснований системы остаточных классов. Результат работы программы для выбранных оснований СОК позволяет оценить такую важную характеристику модулярного кода, как избыточность, и построить наиболее оптимизированную систему. Программа может быть использована при построении систем хранения и обработки данных, позволяя оптимизировать модель обмена сообщениями за счет снижения нагрузки на сеть и повышения уровня отказоустойчивости работы системы.

Следующим шагом алгоритма проектирования отказоустойчивой вычислительной системы является выбор метода перевода чисел из позиционной системы счисления в систему остаточных классов. Основные методы перевода рассмотрены в Параграфе 2.1. Для проверки данных методов создан модуль нахождения остатков (рис. 3.2), представленный комплексом программ [124–128; 134], которые позволяют наглядно продемонстрировать производительность каждого алгоритма и выявить наиболее эффективные для реализации задач цифровой обработки сигналов. Также данные программы использовались для оценки отказоустойчивости и вероятности бесперебойной работы облачных провайдеров в зависимости от кодов обнаружения, локализации и исправления ошибок при моделировании распределенных систем хранения данных в облаках. При этом основные преимущества системы остаточных классов раскрываются при реализации алгоритмов в виде специализированных интегральных схем ASIC, описание которых осуществляется на языке описания аппаратуры Verilog. Для генерации Verilog-модулей реализации нахождения остатка на наборы модулей системы остаточных классов с использованием методов периода и полупериода, нейронной сети конечного кольца на языке высокого уровня Python была написана программа для ЭВМ [134]. Входом программы является набор модулей СОК, выходом – сгенерированный Verilog-файл. Данная программа позволила подготовить массив данных для анализа, отражающий зависимость результатов от выбранного метода для наборов с различным количеством модулей в диапазоне от

8 до 64 бит. Результаты моделирования сгенерированных модулей рассмотрены в Параграфе 3.2.

Следующим шагом алгоритма проектирования отказоустойчивой вычислительной системы, работающей в модулярном коде, является выполнение модульных операций, которые выполняются независимо по каждому из оснований. Программная реализация данных операций реализована модулем распределенных модулярных вычислений (рис. 3.2). В качестве примера такой операции может быть выбрано широко используемое в цифровой обработке сигналов и криптографии модулярное умножение, рассмотренное в Параграфе 2.4. Другой важной операцией, широко используемой в нейронных сетях, является умножение с накоплением, для реализации которой разработан генератор Verilog-модулей умножителя с накоплением (MAC) [132]. Моделирование полученных модулей рассмотрено в Параграфе 3.5.

Однако для ряда операций, например сравнения чисел и определения знака числа, необходимо знать позиционную характеристику числа, для чего в программном комплексе реализован модуль вычисления позиционных характеристик (рис. 3.2). Как показано в Параграфе 2.3, методы сравнения чисел и определения знака могут быть получены как в результате обратного перевода из СОК в ПСС, рассмотренного в Параграфе 2.2, так и методов, пригодных только для сравнения чисел и определения знака. Для исследования методов перевода из СОК в ПСС введен модуль перевода из СОК в ПСС (рис. 3.2), реализованный программами для ЭВМ [124–127; 129], при этом для моделирования на ASIC разработаны генераторы Verilog-модулей [133; 135]. Результаты моделирования перевода из СОК в ПСС показаны в Параграфе 3.3, а определения знака и сравнения чисел рассмотрены в Параграфе 3.4.

Для случая построения отказоустойчивых вычислительных систем необходимо обеспечение коррекции ошибок в СОК, которое рассмотрено в Параграфе 2.5, что соответствует модулю коррекции ошибки (рис. 3.2). Для исследования корректирующих свойств модулярного кода разработаны программы для ЭВМ [126; 127; 129; 130]. Моделирование исследуемых методов коррекции ошибок на ASIC, рассмотренное в Параграфе 3.6, построено с использование генераторов Verilog-модулей [136; 137]. Таким образом, заключительным шагом алгоритма проектирования отказоустойчивой вычислительной системы является выбор метода вычисления позиционной характеристики числа для целей срав-

нения чисел, определения знака числа, перевода в позиционную систему счисления, коррекции ошибок вычислений.

Рассмотрим разработанные архитектуры вычислительных узлов распределенной среды, реализующих методы и алгоритмы, описанные в главе 2. Моделирование вычислительных узлов было произведено на ASIC в среде RTL и физического синтеза Cadence Genus Synthesis Solution с использованием библиотеки osu018_stdcells.

3.2 Архитектура вычислительного узла для нахождения остатков по модулям СОК

В параграфе 2.1 рассмотрены методы вычисления остатка на модули специального вида $2^n - 1$, 2^n, $2^n + 1$.

В основе нейронной сети конечного кольца лежит формула (2.2)

$$|X|_p = \left| \sum_{i=0}^{N-1} \left|2^i\right|_p \{x_i\} \right|_p ,$$

где коэффициенты $\left|2^i\right|_p$ образуют кольцо вычетов по данному модулю. Для моделирования были выбраны входные числа размерностью 2^n бит, $n = 3, 4, \ldots, 8$, для которых взяты модули вида $2^n - 1$ от 3 до $2^{128} - 1$. Результаты моделирования представлены в приложении А в таблицах 13 (площадь) и 16 (время).

Можно заметить, что при приближении размера модуля к размеру входного числа площадь и время увеличиваются.

Использование полупериода по формуле (2.4)

$$\left|2^{j \cdot HP(p)+i}\right|_p = (-1)^j \left|2^i\right|_p$$

позволят для модулей вида $2^n + 1$ свести нахождение остатка к сложению и вычитанию n-битных чисел. Для моделирования были выбраны входные числа размерностью 2^n бит, $n = 3, 4, \ldots, 8$, для которых взяты модули вида $2^n + 1$ от 5 до $2^{128} + 1$. Результаты моделирования представлены в приложении А в таблицах 14 (площадь) и 17 (время).

Использование периода по формуле (2.3)

$$\left|2^{jP(p)+i}\right|_p = \left|2^i\right|_p$$

позволят для модулей вида $2^n - 1$ свести нахождение остатка к сложению n-битных чисел. Для моделирования были выбраны входные числа размерностью 2^n бит, $n = 3, 4, \ldots, 8$, для которых взяты модули вида $2^n - 1$ от 3 до $2^{128} - 1$. Результаты моделирования представлены в приложении А в таблицах 15 (площадь) и 18 (время).

При использовании периода и полупериода чисел приближение размера модуля к размеру входного числа площадь уменьшается, а время увеличивается.

На рисунках 3.3–3.4 показаны соответственно используемая площадь и время вычисления для нейронной сети конечного кольца для модулей вида $2^{n-1} - 1$, для метода полупериода числа для модулей вида $2^{n-1} + 1$, для метода периода числа для модулей вида $2^{n-1} - 1$, при входных числах размерностью 2^n бит.

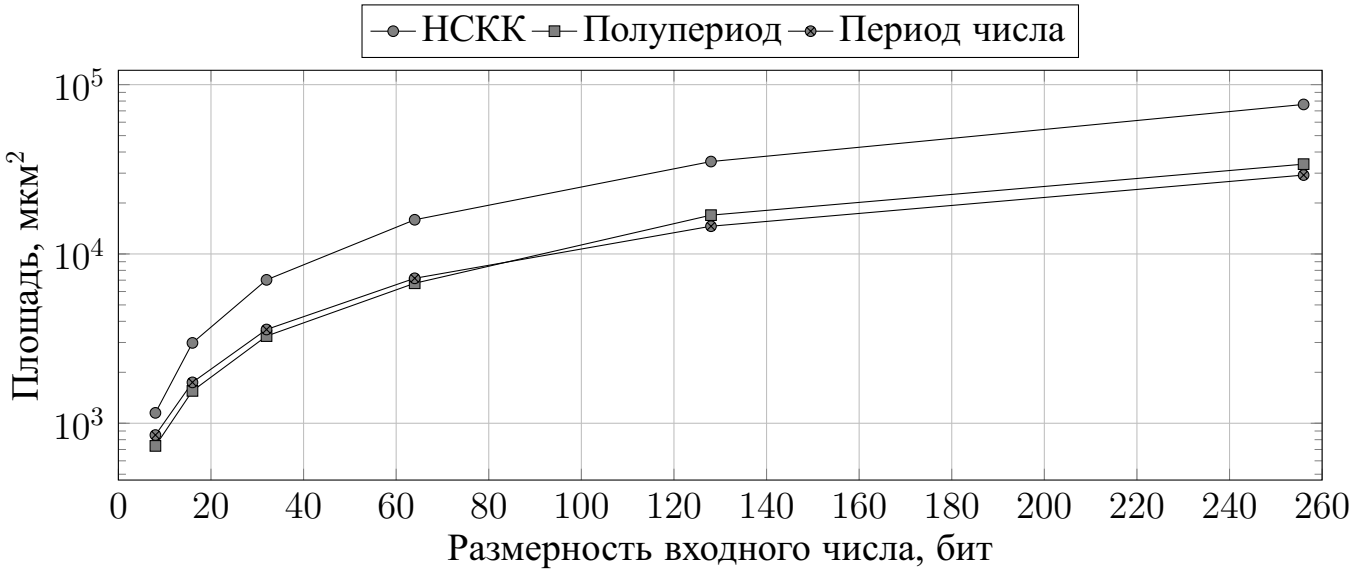

Рисунок 3.3 — Сравнение используемой площади для методов нахождения остатка от деления по модулю $2^n \pm 1$.

Использование периода и полупериода за счет более эффективных логических элементов позволяет достичь лучших результатов по всем рассматриваемым параметрам по отношению к нейронной сети конечного кольца. Таким образом далее для перевода из позиционной системы счисления в систему остаточных классов будут использованы методы на основе периода и полупериода.

В параграфе 2.1 также рассмотрено построение эффективных схем перевода в СОК одновременно по нескольким модулям, обладающих общими ресурсами. Суть метода заключается в том, что для модулей p_i находят периоды, а затем их наименьшее общее кратное, т.е. $P_{\text{НОК}} = \text{НОК}\{P(p_1), P(p_2), \ldots\}$. Затем

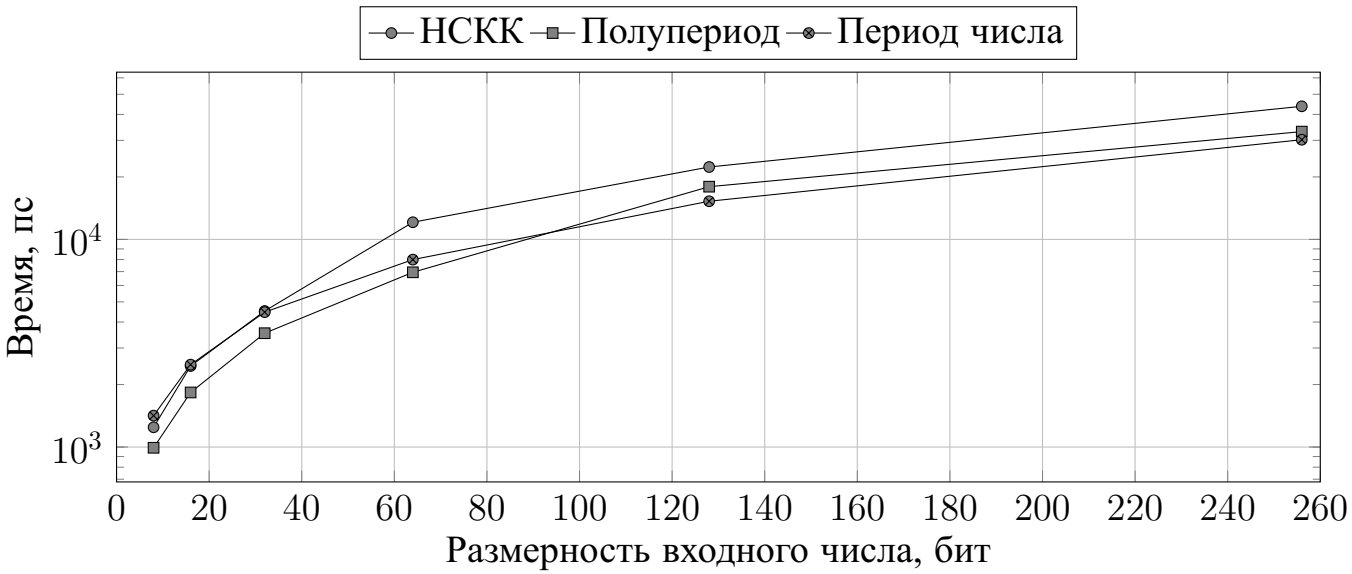

Рисунок 3.4 — Сравнение времени для методов нахождения остатка от деления по модулю $2^n \pm 1$.

входное число X размерностью N бит разбивается на блоки по $r = \lceil N/P_{\text{НОК}} \rceil$ бит и находится остаток по модулю $2^{P_{\text{НОК}}} - 1$.

Результаты моделирования применения общих ресурсов для наборов СОК, покрывающих диапазон 16 бит, показаны в таблице 5.

Таблица 5 — Результаты моделирования перевода в СОК для схем с общими ресурсами

Набор модулей СОК	Время, пс	Площадь, мкм^2
{7, 9, 17, 64}	2503	7274
{7, 9, 17, 64} с общими ресурсами	3016	6582
{7, 15, 17, 64}	2468	6854
{7, 15, 17, 64} с общими ресурсами	2582	5986
{31, 33, 128}	2761	4933
{31, 33, 128} с общими ресурсами	2852	4158

Из таблицы 5 видно, что использование общих ресурсов позволяет сократить используемую площадь, однако время вычислений в этом случае возрастает.

Таким образом для построения высокопроизводительных вычислительных узлов распределенной среды будем рассматривать независимое нахождения остатков с использованием периода и полупериода на основе предложенного Алгоритма 2.6, рассмотренного в Параграфе 2.1.1. Для моделирования выбран

диапазон от 8 до 64 бит, в котором наборы СОК имеют разное количество модулей. Результаты моделирования представлены в таблице 6.

Таблица 6 — Результаты моделирования перевода в СОК с использование периода и полупериода

Набор модулей СОК	Время, пс	Площадь, мкм^2
8 бит		
{3, 5, 32}	1672	2102
{5, 7, 8}	1712	2202
16 бит		
{17, 31, 128}	2744	4190
{31, 63, 64}	2689	4578
{7, 9, 17, 64}	2627	6206
24 бита		
{127, 255, 1024}	3688	6398
{31, 63, 65, 256}	3265	9352
{5, 7, 9, 17, 31, 128}	3164	17037
32 бита		
{1023, 2047, 4096}	5064	8926
{127, 255, 511, 512}	4657	12845
{31, 63, 65, 127, 512}	3896	17937
40 бит		
{8191, 16383, 16384}	6056	11355
{511, 1023, 2047, 2048}	5335	16202
{127, 255, 257, 511, 512}	4655	21618
{17, 31, 65, 129, 511, 512}	4788	27016
48 бит		
{65535, 65537, 131072}	7673	11940
{2047, 4095, 8191, 8192}	6208	19564
{257, 511, 1023, 1025, 2048}	5475	25741
{65, 127, 129, 257, 511, 2048}	5367	31255
{17, 31, 65, 127, 129, 511, 1024}	5276	38217
56 бит		

{262143, 524287, 1048576}	8262	15690
{8191, 16383, 32767, 32768}	6740	23199
{127, 257, 511, 513, 2047, 8192}	6019	37886
{17, 31, 65, 127, 129, 257, 511, 1024}	5530	52271
64 бита		
{2097151, 4194303, 4194304}	9411	17992
{32767, 65535, 131071, 131072}	7924	25584
{257, 511, 1025, 2049, 8191, 8192}	7056	41720
{65, 127, 257, 511, 1023, 2047, 8192}	6128	51223

Из таблицы 6 видно, что увеличение количества модулей СОК позволяет снизить время выполнения алгоритма, однако используемая площадь в этом случае значительно возрастает. При этом выбор конкретного набора СОК с оптимальным соотношением времени и используемой площади должен проходить с учетом не только перевода в СОК, а также обратного преобразования и выполнения необходимых арифметических операций.

3.3 Архитектура вычислительного узла для перевода чисел из системы остаточных классов и расширения оснований

Исследование методов перевода из системы остаточных классов рассмотрено в параграфе 2.2. Наибольшее распространение получил метод на основе Китайской теоремы об остатках по формуле (2.6)

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

при этом возможна вариация данной формулы, в которой остаток от деления берется для каждого произведения, что можно выразить формулой

$$X = \left| \sum_{i=1}^{n} \left| P_i \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_P \right|_P . \tag{3.1}$$

Результаты моделирования метода перевода из СОК на основе КТО представлены в таблице 19 приложения Б. При этом вычисления по формуле (3.1)

дают преимущества на диапазонах 8-16 бит, с увеличением же диапазонов и площадь и время показывают лучшие результаты для формулы (2.6). В этом случае вычисление остатка от динамического диапазона становится ресурсозатратнее вычислений с числами большой разрядности. При этом ограничением данной реализации является использование констант, размер которых сопоставим с размером динамического диапазона.

Использование приближенного метода по формуле (2.7)

$$\frac{X}{P} = \left| \sum_{i=1}^{n} x_i \cdot \frac{\left| P_i^{-1} \right|_{p_i}}{p_i} \right|_1 = \left| \sum_{i=1}^{n} x_i \cdot k_i \right|_1,$$

с масштабированием на 2^N, где $N = \lceil \log_2 (P \cdot \sum_{i=1}^{n} (p_i - 1)) \rceil$ увеличивает размер констант k_i, однако операция нахождения остатка от деления в этом случае сводится к взятию младших бит числа. Результаты моделирования приближенного метода перевода из СОК на основе КТО представлены в таблице 20 приложения Б.

Обобщенная позиционная система счисления (формула (2.8))

$$X = x_1 + d_1 p_1 + d_2 p_1 p_2 + d_3 p_1 p_2 p_3 + \ldots + d_{n-1} p_1 p_2 \ldots p_{n-1},$$

имеет линейную архитектуру, в которой цифры ОПСС d_i вычисляются последовательно. При этом особенностью ОПСС является возможность использовать любой порядок модулей. В упорядоченной СОК с четным последним модулем для применения ОПСС возьмем обратный порядок модулей. Тогда значения разностей, показанных в алгоритме 2.9 после прибавления модуля будут в удвоенном диапазоне, например, для $\left| x_3 - x_2 \right|_{p_3}$ можно свести к положительному значению в диапазоне $[p_3 - p_2 - 1 : 2p_3 - 1]$. Умножение на мультипликативную инверсию приводит к числам, превышающим двухкратный размер модуля и для нахождения остатка используются рассмотренные в Параграфе 2.1.1 методы на основе периода и полупериода числа. Результаты моделирования метода перевода из СОК на основе ОПСС представлены в таблице 21 приложения Б. Можно сделать вывод, что значения площади и времени прямо пропорциональны количеству модулей и наибольшую эффективность показывают СОК с тремя модулями.

В Параграфе 2.2 для методов обратного перевода рассмотрена оптимизация СОК $\{2^n - 1, 2^n, 2^n + 1\}$, предложенная в статье [58]. Восстановление числа происходит за счет битовых операций и нахождения остатка от деления на мо-

дуль специального вида $2^{2n} - 1$. Результаты моделирования метода перевода из СОК с использованием спецмодулей представлены в таблице 22 приложения Б.

Одним из предложенных методов является метод перевода чисел из системы остаточных классов и расширения оснований, выраженный Алгоритмом 2.14 из Параграфа 2.2.1. Реализация данного алгоритма возможна с помощью вычислительного узла, на который получен патент на изобретение [117]. Рассмотрим архитектуру вычислительного узла для перевода чисел из СОК и расширения оснований, которая представлена на рисунке 3.5.

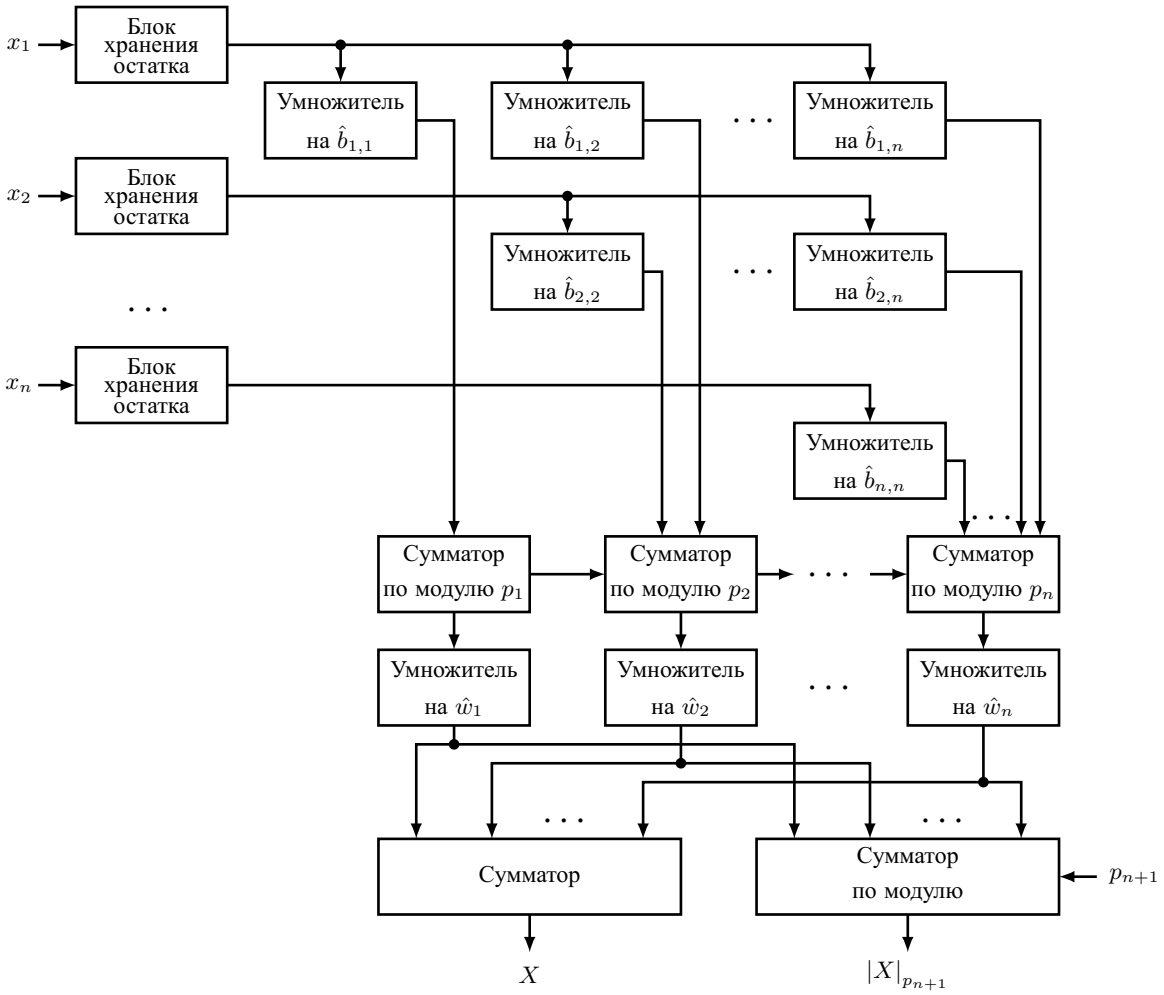

Рисунок 3.5 — Архитектура вычислительного узла перевода чисел из СОК и расширения оснований

На n входов остатков, где n – количество оснований системы остаточных классов, подается число $(x_1, x_2, \ldots, x_n)$, представленное в СОК. Остатки с соответствующих входов остатков записываются в n блоков хранения остатков. С выходов блоков хранения остатков остатки по модулям $\{p_1, p_2, \ldots, p_n\}$ подаются на умножители, которые образуют треугольную матрицу и вычисля-

ют значения произведений остатка на коэффициенты базиса в ОПСС $x_j \cdot \hat{b}_{j,i}$, где $B_i = P_i \cdot \left|P_i^{-1}\right|_{p_i} = \sum_{j=1}^{n} \hat{b}_{i,j} \cdot \hat{w}_j$, B_i — ортогональные базисы СОК, $\hat{w}_j = \prod_{i=1}^{j-1} p_i, j = \overline{1,n}$ — основания ОПСС, при этом $\hat{w}_0 = 1$.

Выход первого блока хранения остатка по модулю p_1 соединен со входами n умножителей на $\hat{b}_{1,1}$, …, $\hat{b}_{1,n}$, которые вычисляют значения $x_1 \cdot \hat{b}_{1,i}$, $i = \overline{1,n}$.

Выход второго блока хранения остатка по модулю p_2 соединен со входами $(n-1)$-го умножителя на $\hat{b}_{2,2}$, …, $\hat{b}_{2,n}$, которые вычисляют значения $x_2 \cdot \hat{b}_{2,i}$, $i = \overline{2,n}$, и так далее, выход n-го блока хранения остатка по модулю p_n соединен со входом умножителя на $\hat{b}_{n,n}$, который вычисляет значение $x_n \cdot \hat{b}_{n,n}$.

Выходы умножителей соединены со входами n модулярных сумматоров по модулям p_i, $i = \overline{1,n}$, при этом выход умножителя на $\hat{b}_{1,1}$ соединен со входом первого модулярного сумматора по модулю p_1 и выполняет вычисления $\hat{x}_1 = |U_1|_{p_1} = \left|x_1 \cdot \hat{b}_{1,1}\right|_{p_1}$, $\sigma_1 = \left\lfloor \frac{x_1 \cdot \hat{b}_{1,1}}{p_1} \right\rfloor$, значение $\hat{x}_1$ поступает на выход первого модулярного сумматора по модулю p_1, значение σ_1 поступает на выход переноса первого модулярного сумматора по модулю p_1.

Выходы умножителей на $\hat{b}_{1,2}$ и на $\hat{b}_{2,2}$ соединены со входами второго модулярного сумматора по модулю p_2, вход переноса которого соединен со выходом переноса первого модулярного сумматора по модулю p_1, и выполняет вычисления $\hat{x}_2 = |\sigma_1 + U_2|_{p_2} = \left|\sigma_1 + x_1 \cdot \hat{b}_{1,2} + x_2 \cdot \hat{b}_{2,2}\right|_{p_2}$, $\sigma_2 = \left\lfloor \frac{\sigma_1 + x_1 \cdot \hat{b}_{1,2} + x_2 \cdot \hat{b}_{2,2}}{p_2} \right\rfloor$, значение $\hat{x}_2$ поступает на выход второго модулярного сумматора по модулю p_2, значение σ_2 поступает на выход переноса второго модулярного сумматора по модулю p_2.

И так далее, выходы умножителей на $\hat{b}_{i,n}$, $i = \overline{1,n}$, соединены со входами n-го модулярного сумматора по модулю p_n, вход переноса которого соединен с выходом переноса $(n-1)$-го модулярного сумматора по модулю p_{n-1}, и выполняет вычисления $\hat{x}_n = |\sigma_{n-1} + U_n|_{p_n} = \left|\sigma_{n-1} + \sum_{i=1}^{n} x_i \hat{b}_{i,n}\right|_{p_n}$, $\sigma_n = \left\lfloor \frac{\sigma_{n-1} + \sum_{i=1}^{n} x_i \hat{b}_{i,n}}{p_n} \right\rfloor$, значение $\hat{x}_n$ поступает на выход n-го модулярного сумматора по модулю p_n, значение σ_n поступает на выход переноса n-го модулярного сумматора по модулю p_n, который не подключен к другим элементам.

Значения $\hat{x}_i$ с выходов i-х модулярных сумматоров по модулю p_i подаются на входы i-х умножителей на основания ОПСС $\hat{w}_i$, $i = \overline{1,n}$, в которых происходит умножение $\hat{x}_i$ на $\hat{w}_i = \prod_{j=1}^{i-1} p_j$.

Выход i-го умножителя на основания ОПСС $\hat{w}_i$ подключен к i-м входам сумматора и модулярного сумматора, $i = \overline{1,n}$, на вход расширенного основания модулярного сумматора подается значение модуля p_{n+1}, на выход восстановлен-

ного числа сумматора подается восстановленное число X в позиционной системе счисления, а на выход остатка по расширенному основанию модулярного сумматора значение числа по расширенному основанию, т.е. $|X|_{p_{n+1}}$.

Рассмотрим работу вычислительного узла для $n = 4$, $p_1 = 3$, $p_2 = 5$, $p_3 = 7$, $p_4 = 11$. На входы остатков подаются значения $X = (1, 2, 3, 4) = 367$, которые затем записываются в блоков хранения остатков по соответствующим модулям.

Значение 1 с выхода блока хранения остатка по модулю p_1 одновременно поступает на входы умножителя на $\hat{b}_{1,1}$, где происходит умножение остатка $x_1 = 1$, на первый коэффициент базиса B_1 в ОПСС $\hat{b}_{1,1} = 1$, результат умножения $x_1 \cdot \hat{b}_{1,1} = 1$, умножителя на $\hat{b}_{1,2}$, где происходит умножение на $\hat{b}_{1,2} = 3$, результат умножения $x_1 \cdot \hat{b}_{1,2} = 3$, умножителя на $\hat{b}_{1,3}$, где происходит умножение на $\hat{b}_{1,3} = 4$, результат умножения $x_1 \cdot \hat{b}_{1,3} = 4$, умножителя на $\hat{b}_{1,4}$, где происходит умножение на $\hat{b}_{1,4} = 3$, результат умножения $x_1 \cdot \hat{b}_{1,4} = 3$.

Значение 2 с выхода блока хранения остатка по модулю p_2 одновременно поступает на входы умножителя на $\hat{b}_{2,2}$, где происходит умножение остатка $x_2 = 2$, на второй коэффициент базиса B_2 в ОПСС $\hat{b}_{2,2} = 2$, результат умножения $x_2 \cdot \hat{b}_{2,2} = 4$, умножителя на $\hat{b}_{2,3}$, где происходит умножение на $\hat{b}_{2,3} = 1$, результат умножения $x_2 \cdot \hat{b}_{2,3} = 2$, умножителя на $\hat{b}_{2,4}$, где происходит умножение на $\hat{b}_{2,4} = 2$, результат умножения $x_2 \cdot \hat{b}_{2,4} = 4$.

Значение 3 с выхода блока хранения остатка по модулю p_3 одновременно поступает на входы умножителя на $\hat{b}_{3,3}$, где происходит умножение остатка $x_3 = 3$, на третий коэффициент базиса B_3 в ОПСС $\hat{b}_{3,3} = 1$, результат умножения $x_3 \cdot \hat{b}_{3,3} = 2$, умножителя на $\hat{b}_{3,4}$, где происходит умножение на $\hat{b}_{3,4} = 3$, результат умножения $x_3 \cdot \hat{b}_{3,4} = 9$.

Значение 4 с выхода блока хранения остатка по модулю p_4 поступает на вход умножителя на $\hat{b}_{4,4}$, где происходит умножение остатка $x_4 = 4$, на четвертый коэффициент базиса B_4 в ОПСС $\hat{b}_{4,4} = 2$, результат умножения $x_4 \cdot \hat{b}_{4,4} = 8$. Значения коэффициентов ОПСС $b_{i,j}$ зависят только от заданной СОК и могут быть записаны в память.

Далее значение 1 с умножителя на $\hat{b}_{1,1}$ поступает на первый модулярный сумматор по модулю $p_1 = 3$. Поскольку $\hat{x}_1 = |1|_3 = 1$ и $\sigma_1 = \left\lfloor \frac{1}{3} \right\rfloor = 0$, на выход модулярного сумматора по модулю p_1 подается 1, а на выход переноса 0.

Значение 3 с умножителя на $\hat{b}_{1,2}$, значение 4 с умножителя на $\hat{b}_{2,2}$ и значение 0 с выхода переноса модулярного сумматора по модулю p_1 поступают на второй модулярный сумматор по модулю $p_2 = 5$. Поскольку $\hat{x}_2 = |3 + 4 + 0|_5 = 2$

и $\sigma_2 = \left\lfloor \frac{7}{5} \right\rfloor = 1$, на выход модулярного сумматора по модулю p_2 подается 2, а на выход переноса 1.

Значение 4 с умножителя на $\hat{b}_{1,3}$, значение 2 с умножителя на $\hat{b}_{2,3}$, значение 3 с умножителя на $\hat{b}_{3,3}$ и значение 1 с выхода переноса модулярного сумматора по модулю p_2 поступают на третий модулярный сумматор по модулю $p_3 = 7$. Поскольку $\hat{x}_3 = |4 + 2 + 3 + 1|_7 = 3$ и $\sigma_3 = \left\lfloor \frac{10}{7} \right\rfloor = 1$, на выход модулярного сумматора по модулю p_3 подается 3, а на выход переноса 1.

Значение 3 с умножителя на $\hat{b}_{1,4}$, значение 4 с умножителя на $\hat{b}_{2,4}$, значение 9 с умножителя на $\hat{b}_{3,4}$, значение 8 с умножителя на $\hat{b}_{4,4}$ и значение 1 с выхода переноса модулярного сумматора по модулю p_3 поступают на четвертый модулярный сумматор по модулю $p_4 = 11$. Поскольку $\hat{x}_4 = |3 + 4 + 9 + 8|_{11} = 3$, на выход модулярного сумматора по модулю p_4 подается 3.

Значение 1 с выхода модулярного сумматора по модулю p_1 поступает на вход первого умножителя на основания ОПСС, где умножается на $\hat{w}_1 = 1$, на выход первого умножителя подается 1. Значение 2 с выхода модулярного сумматора по модулю p_2 поступает на вход второго умножителя на основания ОПСС, где умножается на $\hat{w}_2 = 3$, на выход второго умножителя подается 6. Значение 3 с выхода модулярного сумматора по модулю p_3 поступает на вход третьего умножителя на основания ОПСС, где умножается на $\hat{w}_3 = 15$, на выход третьего умножителя подается 45. Значение 3 с выхода модулярного сумматора по модулю p_4 поступает на вход четвертого умножителя на основания ОПСС, где умножается на $\hat{w}_4 = 105$, на выход четвертого умножителя подается 315. При этом основания ОПСС $\hat{w}_i$ зависят только от модулей СОК и могут быть записаны в память.

Значения с выходов умножителей на основания ОПСС поступают на входы сумматора, где происходит сложение $1+6+45+315 = 367$, результат которого подается на выход восстановленного числа. Одновременно значения с выходов умножителей на основания ОПСС поступают на входы модулярного сумматора, на вход расширенного модуля которого поступает значение $p_{n+1} = 13$. В модулярном сумматоре вычисляется сложение по модулю $|1 + 6 + 45 + 315|_{13} = 3$, результат которого подается на выход остатка по расширенному основанию.

Можно заметить, что при реализации вычислительного узла отсутствует ресурсоемкая операция нахождения остатка по большому модулю $P = 1155$. Также сокращено количество умножителей и упрощены реализуемые ими функции. За счет введения входа расширенного основания вычислительный узел поз-

воляет вычислить остаток по произвольному расширенному основанию, а введение сумматора позволяет получить позиционное представление числа. Реализация вычислительного узла возможна как в виде модуля в составе программной вычислительной системы, реализованной на ЭВМ, так и с использованием интегральных схем, таких как FPGA и ASIC.

При этом деление в Verilog может быть реализовано с использованием последовательного вычитания масштабированного делителя по формуле (3.2)

$$R = A - B \cdot q_n \cdot 2^n - B \cdot q_{n-1} \cdot 2^{n-1} - \ldots - B \cdot q_1 \cdot 2^1 - B \cdot q_0 \cdot 2^0. \qquad (3.2)$$

при котором одновременно получают неполное частное и остаток от деления.

Таким образом, проведено моделирование методов перевода из СОК в ПСС на основе Китайской теоремы об остатках, её приближенной реализации, обобщенной позиционной системы счисления и предложенной модификации на основе обобщенной позиционной системы счисления, а также модулей специального вида. Для сравнения данных методов из таблиц 19–23 выберем для каждой размерности лучшие значения по времени и площади в таблицы 24–25. На основе полученных данных построим графики сравнения рассмотренных методов.

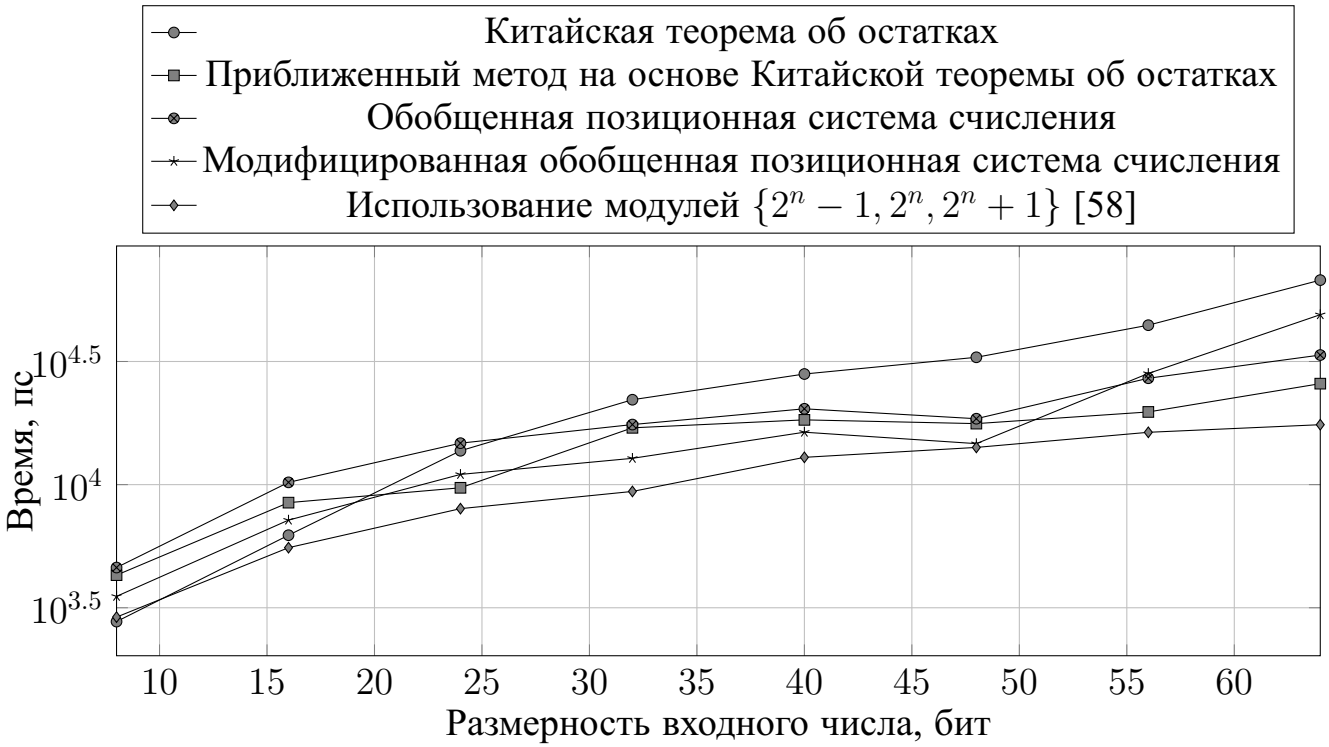

Рисунок 3.6 — Сравнение времени для методов перевода из СОК в ПСС.

Из графиков 3.6–3.7 видно, что использование модулей специального вида $2^n - 1, 2^n, 2^n + 1$ [58] занимают значительно меньшую площадь, по сравнению с

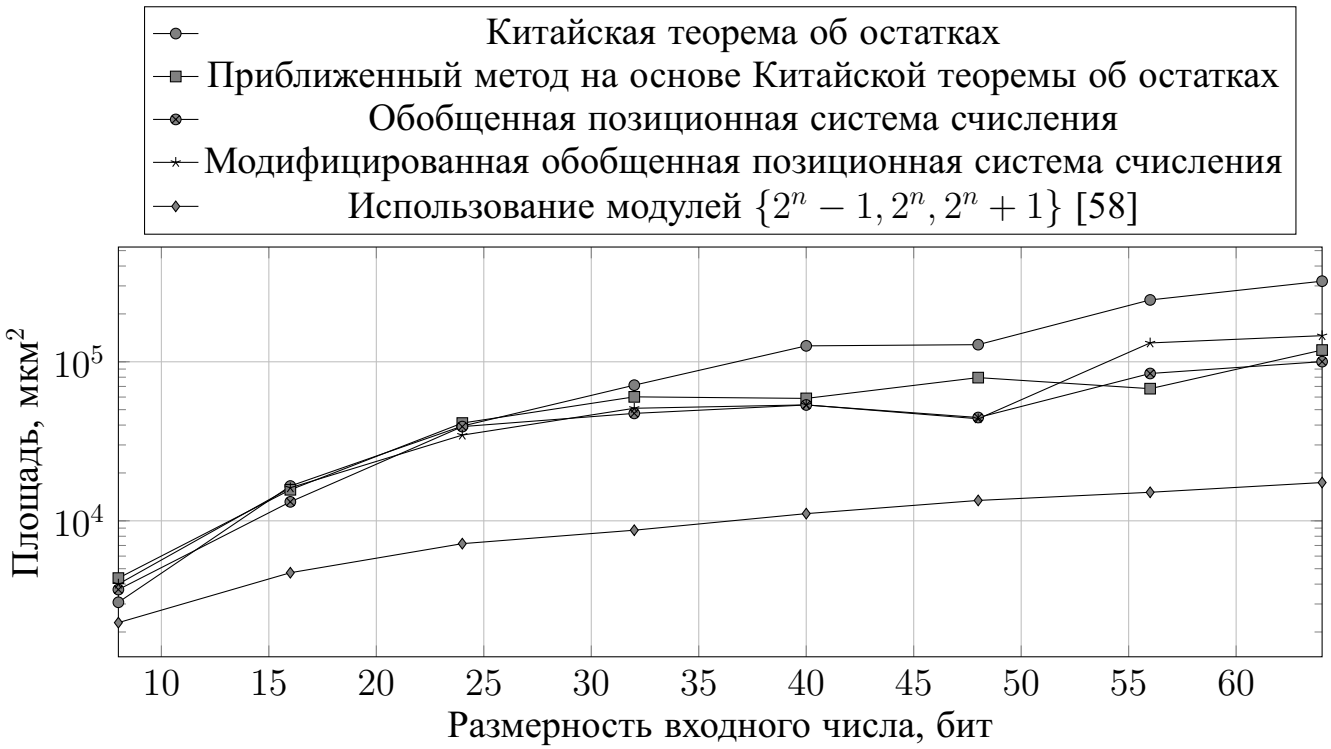

Рисунок 3.7 — Сравнение используемой площади для методов перевода из СОК в ПСС.

другими методами. Также можно сделать вывод, что приближенные методы на основе КТО имеют лучшие характеристики по времени работы по сравнению с ОПСС, однако проигрывают ей по используемой площади. Предложенный метод на основе модифицированной ОПСС является компромиссным решением, имея сопоставимое с приближенным методом на основе КТО время вычислений и сопоставимую с ОПСС используемую площадь. При этом для некоторых размерностей данный метод является лучшим как по площади, так и по времени.

3.4 Архитектура вычислительного узла сравнения и определения знака чисел, представленных в системе остаточных классов

Сравнение чисел является одной из основных операций в цифровой обработке сигналов, в детектировании движения, цифровой фильтрации и определении переполнения диапазона.

Сравнение чисел на основе Китайской теоремы об остатках показано в Алгоритме 2.15. Результаты моделирования методов сравнения чисел в СОК на основе КТО представлены в таблице 26 приложения В.

В Параграфе 2.3 рассмотрены две оценки точности для реализации приближенного метода на основе КТО. В статье [86] дана следующая оценка необходимой точности $N = \lceil \log_2 (nP) \rceil$, где n — количество модулей, а P — динамический диапазон СОК. В статье [75] используется оценка $N = \lceil \log_2 (P\rho) \rceil$ и $\rho = -n + \sum_{i=1}^{n} p_i$. Результаты моделирования методов сравнения чисел в СОК приближенным методом на основе КТО с данными оценками представлены в таблице 27 приложения В.

Сравнение на основе обобщенной позиционной системы счисления представлено Алгоритмом 2.16. Результаты моделирования методов сравнения в СОК на основе ОПСС представлены в таблице 28 приложения В.

В Параграфе 2.3.1 предложен Алгоритм 2.20 построения функции ядра Акушского с заданными свойствами и Алгоритм 2.21 для сравнения чисел на основе построенной функции. Рассмотрим реализацию данного алгоритма [102], которая поясняется архитектурой вычислительного узла сравнения и определения знака чисел, представленных в системе остаточных классов, изображенной на рисунке 3.8.

Вычислительный узел сравнения и определения знака чисел содержит входы первого и второго чисел, при этом количество входов для каждого числа равно количеству модулей СОК $p_1, p_2, \ldots, p_n$. Вход знака указывает на режим работы вычислительного узла (использование только положительных чисел, или положительных и отрицательных). Числа с входов первого и второго чисел поступают соответственно в регистры хранения первого и второго чисел, где сохраняются остатки первого и второго чисел по каждому модулю. Остатки по первому модулю p_1 с первых входов первого и второго чисел подключены дополнительно к пятому и шестому входу решающего блока.

Далее хранимые остатки из регистров хранения первого и второго чисел поступают соответственно в блоки умножения первого и второго чисел, где происходит умножение остатка на значение функции ядра от базисов B_i СОК, т.е. $C(B_i)$, где функция ядра $C(X)$ подбирается из условия монотонности и отсутствия критических ядер (по Алгоритму 2.20). При этом блоки умножения первого и второго чисел могут быть реализованы как с помощью умножителей, напрямую перемножая числа, так и табличным методом, когда в зависимости

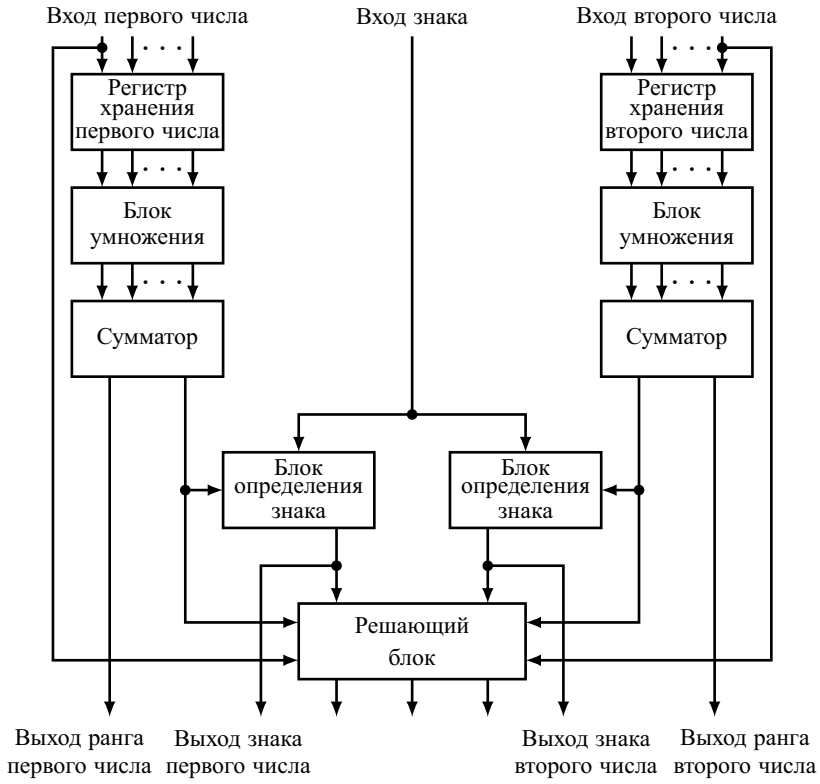

Рисунок 3.8 — Архитектура вычислительного узла сравнения и определения знака чисел

от значения входа на выход подается предвычисленное значение произведения. Далее полученные произведения складываются в соответствующих сумматорах первого и второго чисел. На первые выходы сумматоров первого и второго чисел подаются младшие N бит суммы, где N соответствует максимальному диапазону функции ядра $C_P = 2^N$, таким образом реализуется нахождение остатка по модулю C_P по формуле (2.37). На вторые выходы сумматоров первого и второго чисел подаются оставшиеся биты суммы, что соответствует формуле (2.38), полученные значения поступают на выходы ранга первого и второго чисел.

На первые входы блоков определения знака первого и второго чисел с входа знака поступает сигнал наличия отрицательных чисел. Если в работе вычислительного узла используются отрицательные числа, то в блоке определения знака первого числа значение функции ядра первого числа, поступающее на второй вход блока определения знака первого числа с первого выхода сумматора первого числа сравнивается с $C(K)$ — функцией ядра от середины диапазона СОК K, где $K = \frac{P}{2} - 1$ если $P = \prod_{i=1}^{n} p_i$ — четное и $K = \frac{(P-1)}{2}$ если P — нечетное; в блоке определения знака второго числа значение функции ядра второго числа, поступающее на второй вход блока определения знака второго числа с

первого выхода сумматора второго числа сравнивается с функцией ядра от середины диапазона СОК K и если число меньше, чем $C(K)$, то сравниваемое значение положительное, иначе отрицательное, и на выход соответствующего блока определения знака подается знак числа.

Если в работе вычислительного узла не используются отрицательные числа, то на выход соответствующего блока определения знака под действием сигнала с входа знака подается, что числа положительные. Выход блока определения знака первого числа является выходом знака первого числа и дополнительно подключен ко второму входу решающего блока, первый вход которого подключен к первому выходу сумматора первого числа.

Выход блока определения знака второго числа является выходом знака второго числа и дополнительно подключен к третьему входу решающего блока, четвертый вход которого подключен к первому выходу сумматора второго числа. Решающий блок на основе знаков первого и второго числа, поступающих на второй и третий входы, принимает решение сразу, если числа имеют разные знаки, в случае одинаковых знаков происходит сравнение по Алгоритму 2.21, т.е. сначала сравниваются значения функции ядра первого и второго числа, поступающие с первого и четвертого входов, а в случае их равенства, остатки по первому модулю, подаваемые на пятый и шестой входы с входов первого и второго чисел. Результат сравнения подается на один из выходов решающего блока: на выход «первое число меньше второго», на выход «первое и второе числа равны», на выход «первое число больше второго».

Рассмотрим работу вычислительного узла на примере СОК $\{11, 13, 17, 19\}$. На вход знака поступает сигнал, что числа могут быть отрицательные. На вход первого числа поступают числа $(8, 10, 4, 7) = 140$, которые сохраняются в регистр хранения первого числа. Далее остатки $(8, 10, 4, 7)$ с выходов регистра хранения первого числа поступают на соответствующие первый-четвертый входы блока умножения первого числа, где происходит их умножение на значения функции ядра от базиса $C(B_1) = 83408$, $C(B_2) = 100824$, $C(B_3) = 84811$, $C(B_4) = 124173$. Далее значения произведений $8 \cdot 83408$, $10 \cdot 100824$, $4 \cdot 84811$, $7 \cdot 124173$ поступают на вход сумматора первого числа, где получают число 2883959, что соответствует двоичному числу 10110000000000101110111. Младшие $N = 17$ бит, что равно 375, поступают на второй вход блока определения знака первого числа и на первый вход решающего блока. Оставшиеся биты суммы 10110, что соответствует числу 22, поступают на выход ранга первого числа.

Одновременно с этим на входы второго числа поступают числа $(7, 7, 14, 17) = 150$, которые сохраняются в регистр хранения второго числа. Далее остатки $(7, 7, 14, 17)$ с выходов регистра хранения второго числа поступают на соответствующие первый-четвертый входы блока умножения второго числа, где происходит их умножение на значения функции ядра от базиса $C(B_1) = 83408$, $C(B_2) = 100824$, $C(B_3) = 84811$, $C(B_4) = 124173$. Далее значения произведений $7 \cdot 83408$, $7 \cdot 100824$, $14 \cdot 84811$, $17 \cdot 124173$ поступают на вход сумматора второго числа, где получают число 4587919, что соответствует двоичному числу 10001100000000110001111. Младшие $N = 17$ бит, что равно 399, поступают на второй вход блока определения знака второго числа и на четвертый вход решающего блока. Оставшиеся биты суммы 100011, что соответствует числу 35, поступают на выход ранга второго числа.

Поскольку с входа знака поступает сигнал, что СОК содержит отрицательные числа, то необходимо сравнить значения функции ядра чисел со значением функции ядра от середины диапазона $C(K) = 65517$. В блоке определения знака первого числа происходит сравнение 375 и 65517 и поскольку $375 < 65517$, то на выход блока определения знака первого числа подается сигнал, что первое число положительное. В блоке определения знака второго числа происходит сравнение 399 и 65517 и поскольку $399 < 65517$, то на выход блока определения знака подается сигнал, что второе число положительное. Если бы число было больше $C(K)$, то оно было бы отрицательным. В случае, если со входа знака поступил бы сигнал отсутствия отрицательных чисел, то на выход блока определения знака независимо от чисел поступает, что число положительное.

С выхода блока определения знака первого числа на выход знака первого числа и на второй вход решающего блока поступает сигнал, что первое число положительное. С выхода блока определения знака второго числа на выход знака второго числа и на третий вход решающего блока поступает сигнал, что второе число положительное.

Поскольку оба числа положительные в решающем блоке происходит сравнение значений функции ядра от первого и второго числа, поступающих на первый и четвертый входы, и так как $375 < 399$, то первое число меньше второго и сигнал поступает на выход «первое число меньше второго». В случае, когда значения функции ядра равны, происходит сравнение остатков по первому модулю, поступающих с входа первого числа и входа второго числа соответственно на пятый и шестой входы решающего блока и если первый остаток первого чис-

ла больше первого остатка второго числа, то на выходе «первое число больше второго» формируется сигнал. Равенство остатков по первому модулю говорит о равенстве чисел и сигнал поступает на выход «первое и второе число равны».

Результаты моделирования сравнения в СОК на основе функции ядра по Алгоритму 2.21 представлены в таблице 29 приложения В.

Также в Параграфе 2.3 введен Алгоритм 2.25 определения знака числа, представленного в системе остаточных классов с четным диапазоном. Рассмотрим архитектуру реализации данного алгоритма в виде вычислительного узла [118].

Архитектура вычислительного узла поясняется рисунком 3.9, который содержит n входов остатка, которые соединены с n регистрами для хранения разрядов исходного числа. Вычислительный узел содержит $n-1$ вычислительных ступеней, при этом i-я вычислительная ступень, где $i = 1, \ldots, n-1$, содержит $n-i$ сумматоров по модулю p_j и $n-i$ блоков умножения на веса $w_{i,j}$ по модулю p_j, где $j = i+1, \ldots, n$ и $w_{i,j}$ — мультипликативная инверсия модуля p_i по модулю p_j.

В первой вычислительной ступени первые информационные входы i-х сумматоров по модулю p_{i+1} через инверторы подключены к выходу первого регистра для хранения разрядов исходного числа, вторые информационные входы i-х сумматоров по модулю p_{i+1} подключены к выходам соответствующих $(i+1)$-х регистров для хранения разрядов исходного числа, на входы переносов i-х сумматоров по модулю p_{i+1} подается сигнал логической единицы, выходы i-х сумматоров по модулю p_{i+1} подключены ко входам соответствующих блоков умножения на веса $w_{1,i+1}$ по модулю p_{i+1}, $i = 1, \ldots, n-1$. Во второй вычислительной ступени первые информационные входы i-х сумматоров по модулю p_{i+2} через инверторы подключены к выходу первого блока умножения на веса $w_{1,2}$ по модулю p_2 первой ступени, вторые информационные входы i-х сумматоров по модулю p_{i+2} подключены к выходам соответствующих $(i+1)$-ых блоков умножения на веса $w_{1,i+2}$ по модулю p_{i+2} первой ступени, на входы переносов i-х сумматоров по модулю p_{i+2} второй ступени подается сигнал логической единицы, выходы i-х сумматоров по модулю p_{i+2} второй ступени подключены ко входам соответствующих i-ых блоков умножения на веса $w_{2,i+2}$ по модулю p_{i+2}, $i = 1, \ldots, n-2$. И так далее, на $n-1$ вычислительной ступени первый информационный вход сумматора по модулю p_n через инвертор подключен к выходу первого блока умножения на веса $w_{n-2,n-1}$ по модулю p_{n-1} $(n-2)$-й ступени,

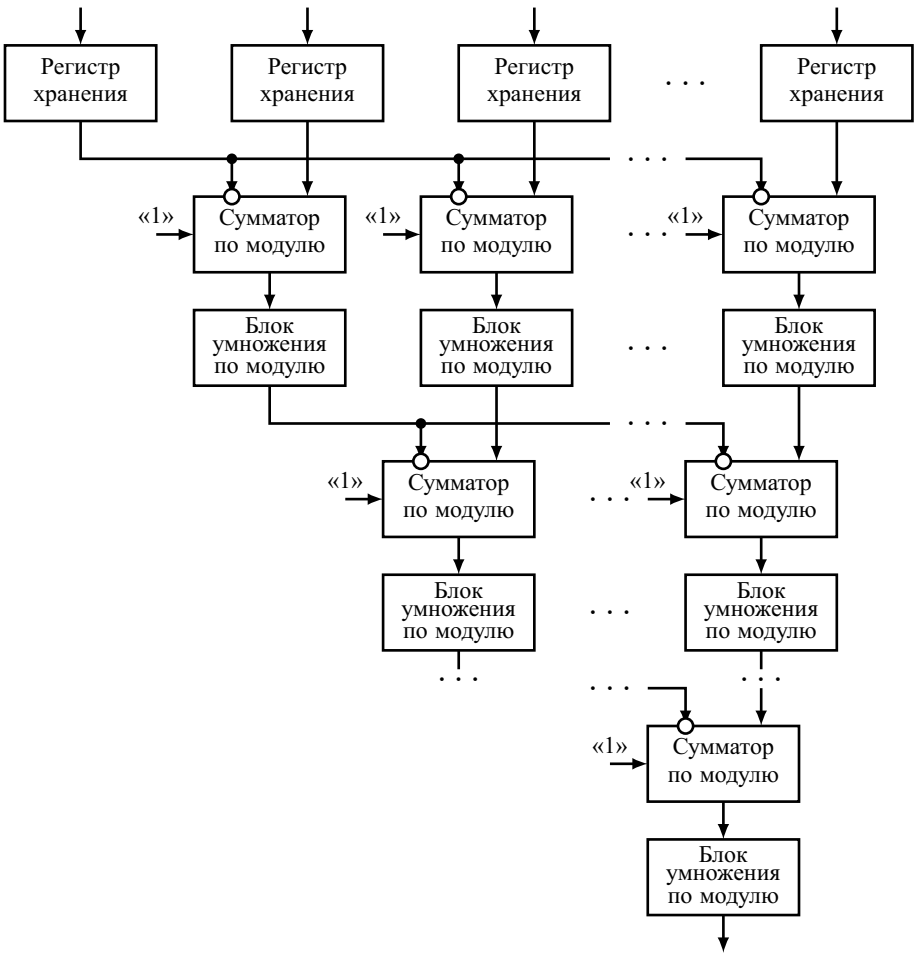

Рисунок 3.9 — Архитектура вычислительного узла определения знака числа

второй информационный вход сумматора по модулю p_n подключен к выходу второго блока умножения на веса $w_{n-2,n}$ по модулю p_n $(n-2)$-й ступени, на вход переноса поступает сигнал логической единицы, а выход соединен со входом первого блока умножения на веса $w_{n-1,n}$ по модулю p_n $(n-1)$-й ступени, старший бит выхода которого является выходом знака.

При этом блоки умножения на веса $w_{i,j}$ по модулю p_j могут быть выполнены как в виде памяти, так и в виде вычислительных узлов распределенной среды. Сумматоры по модулю p_i за счет инвертирования сигнала с одного из входов и сигнала логической единицы на входе переноса фактически выполняют операцию вычитания.

На основе примера рассмотрим работу вычислительного узла. Пусть задана система остаточных классов с модулями $p_1 = 17$, $p_2 = 19$, $p_3 = 23$, $p_4 = 32$. Тогда количество входов остатка и регистров для хранения разрядов исходного числа равно 4, а количество вычислительных ступеней равно 3. Для них веса, на которые происходит умножение в блоках умножения на веса $w_{i,j}$ по модулю

p_j, равны

$$w_{1,2} = \left|p_1^{-1}\right|_{p_2} = \left|\frac{1}{17}\right|_{19} = 9, \quad w_{1,3} = \left|p_1^{-1}\right|_{p_3} = \left|\frac{1}{17}\right|_{23} = 19,$$
$$w_{1,4} = \left|p_1^{-1}\right|_{p_4} = \left|\frac{1}{17}\right|_{32} = 17, \quad w_{2,3} = \left|p_2^{-1}\right|_{p_3} = \left|\frac{1}{19}\right|_{23} = 17,$$
$$w_{2,4} = \left|p_2^{-1}\right|_{p_4} = \left|\frac{1}{19}\right|_{32} = 27, \quad w_{3,4} = \left|p_3^{-1}\right|_{p_4} = \left|\frac{1}{23}\right|_{32} = 7.$$

Пусть на входы остатков поступает число $X = 118863 = (16, 18, 22, 15)$, тогда в регистрах для хранения разрядов исходного числа будут храниться соответственно числа 16, 18, 22, 15.

В первом сумматоре по модулю p_2 первой вычислительной ступени выполняется операция вычитания из значения 18 второго регистра для хранения разрядов исходного числа значения 16 первого регистра для хранения разрядов исходного числа, на выход сумматора по модулю p_2 поступает значение 2. Во втором сумматоре по модулю p_3 первой вычислительной ступени выполняется операция вычитания из значения 22 третьего регистра для хранения разрядов исходного числа значения 16 первого регистра для хранения разрядов исходного числа, на выход сумматора по модулю p_3 поступает значение 6. В третьем сумматоре по модулю p_4 первой вычислительной ступени выполняется операция вычитания из значения 15 четвертого регистра для хранения разрядов исходного числа значения 16 первого регистра для хранения разрядов исходного числа, на выход сумматора по модулю p_4 поступает значение 31.

В первом блоке умножения на веса $w_{1,2}$ по модулю p_2 первой ступени происходит умножение по модулю значения 2 с выхода сумматора по модулю p_2 на сохраненное значение веса $w_{1,2} = 9$, т.е. $|2 \cdot 9|_{19} = 18$. Во втором блоке умножения на веса $w_{1,3}$ по модулю p_3 первой ступени происходит умножение по модулю значения 6 с выхода сумматора по модулю p_3 на сохраненное значение веса $w_{1,3} = 19$, т.е. $|6 \cdot 19|_{23} = 22$. В третьем блоке умножения на веса $w_{1,4}$ по модулю p_4 первой ступени происходит умножение по модулю значения 31 с выхода сумматора по модулю p_4 на сохраненное значение веса $w_{1,4} = 17$, т.е. $|31 \cdot 17|_{32} = 15$.

В первом сумматоре по модулю p_3 второй вычислительной ступени выполняется операция вычитания из значения 22 второго блока умножения на веса $w_{1,3}$ по модулю p_3 первой ступени значения 18 первого блока умножения на веса $w_{1,2}$ по модулю p_2 первой ступени, на выход первого сумматора по модулю p_3

поступает значение 4. Во втором сумматоре по модулю p_4 второй вычислительной ступени выполняется операция вычитания из значения 15 третьего блока умножения на веса $w_{1,4}$ по модулю p_4 первой ступени значения 18 первого блока умножения на веса $w_{1,2}$ по модулю p_2 первой ступени, на выход второго сумматора по модулю p_4 поступает значение 29.

В первом блоке умножения на веса $w_{2,3}$ по модулю p_3 второй ступени происходит умножение по модулю значения 4 с выхода первого сумматора по модулю p_3 на сохраненное значение веса $w_{2,3} = 17$, т.е. $|4 \cdot 17|_{23} = 22$. Во втором блоке умножения на веса $w_{2,4}$ по модулю p_4 второй ступени происходит умножение по модулю значения 29 с выхода второго сумматора по модулю p_4 второй ступени на сохраненное значение веса $w_{2,4} = 27$, т.е. $|29 \cdot 27|_{32} = 15$.

В первом сумматоре по модулю p_4 третьей вычислительной ступени выполняется операция вычитания из значения 15 второго блока умножения на веса $w_{2,4}$ по модулю p_4 первой ступени значения 22 первого блока умножения на веса $w_{2,3}$ по модулю p_3 второй ступени, на выход первого сумматора по модулю p_4 поступает значение 25.

В первом блоке умножения на веса $w_{3,4}$ по модулю p_4 третьей ступени происходит умножение по модулю значения 25 с выхода первого сумматора по модулю p_4 на сохраненное значение веса $w_{3,4} = 7$, т.е. $|25 \cdot 7|_{23} = 15$. В двоичной системе счисления данное число равно 01111 и на выход знака подается старший бит 0.

Таким образом, в СОК с модулями $p_1 = 17$, $p_2 = 19$, $p_3 = 23$, $p_4 = 32$ число $X = 118863 = (16, 18, 22, 15)$ положительное.

Рассмотрим другой пример: $Y = 118864 = (0, 0, 0, 16)$, тогда в регистрах для хранения разрядов исходного числа будут храниться соответственно числа 0, 0, 0, 16.

В первом сумматоре по модулю p_2 первой вычислительной ступени выполняется операция вычитания из значения 0 второго регистра для хранения разрядов исходного числа значения 0 первого регистра для хранения разрядов исходного числа, на выход сумматора по модулю p_2 поступает значение 0. Во втором сумматоре по модулю p_3 первой вычислительной ступени выполняется операция вычитания из значения 0 третьего регистра для хранения разрядов исходного числа значения 0 первого регистра для хранения разрядов исходного числа, на выход сумматора по модулю p_3 поступает значение 0. В третьем сумматоре по модулю p_4 первой вычислительной ступени выполняется операция

вычитания из значения 16 четвертого регистра для хранения разрядов исходного числа значения 0 первого регистра для хранения разрядов исходного числа, на выход сумматора по модулю p_4 поступает значение 16.

В первом блоке умножения на веса $w_{1,2}$ по модулю p_2 первой ступени происходит умножение по модулю значения 0 с выхода сумматора по модулю p_2 на сохраненное значение веса $w_{1,2} = 9$, т.е. $|0 \cdot 9|_{19} = 0$. Во втором блоке умножения на веса $w_{1,3}$ по модулю p_3 первой ступени происходит умножение по модулю значения 0 с выхода сумматора по модулю p_3 на сохраненное значение веса $w_{1,3} = 19$, т.е. $|0 \cdot 19|_{23} = 0$. В третьем блоке умножения на веса $w_{1,4}$ по модулю p_4 первой ступени происходит умножение по модулю значения 16 с выхода сумматора по модулю p_4 на сохраненное значение веса $w_{1,4} = 17$, т.е. $|16 \cdot 17|_{32} = 16$.

В первом сумматоре по модулю p_3 второй вычислительной ступени выполняется операция вычитания из значения 0 второго блока умножения на веса $w_{1,3}$ по модулю p_3 первой ступени значения 0 первого блока умножения на веса $w_{1,2}$ по модулю p_2 первой ступени, на выход первого сумматора по модулю p_3 поступает значение 0. Во втором сумматоре по модулю p_4 второй вычислительной ступени выполняется операция вычитания из значения 16 третьего блока умножения на веса $w_{1,4}$ по модулю p_4 первой ступени значения 0 первого блока умножения на веса $w_{1,2}$ по модулю p_2 первой ступени, на выход второго сумматора по модулю p_4 поступает значение 16.

В первом блоке умножения на веса $w_{2,3}$ по модулю p_3 второй ступени происходит умножение по модулю значения 0 с выхода первого сумматора по модулю p_3 на сохраненное значение веса $w_{2,3} = 17$, т.е. $|0 \cdot 17|_{23} = 0$. Во втором блоке умножения на веса $w_{2,4}$ по модулю p_4 второй ступени происходит умножение по модулю значения 16 с выхода второго сумматора по модулю p_4 второй ступени на сохраненное значение веса $w_{2,4} = 27$, т.е. $|16 \cdot 27|_{32} = 16$.

В первом сумматоре по модулю p_4 третьей вычислительной ступени выполняется операция вычитания из значения 16 второго блока умножения на веса $w_{2,4}$ по модулю p_4 первой ступени значения 0 первого блока умножения на веса $w_{2,3}$ по модулю p_3 второй ступени, на выход первого сумматора по модулю p_4 поступает значение 16.

В первом блоке умножения на веса $w_{3,4}$ по модулю p_4 третьей ступени происходит умножение по модулю значения 16 с выхода первого сумматора по модулю p_4 на сохраненное значение веса $w_{3,4} = 7$, т.е. $|16 \cdot 7|_{23} = 16$. В двоичной

системе счисления данное число равно 10000 и на выход знака подается старший бит 1.

Таким образом, в СОК с модулями $p_1 = 17$, $p_2 = 19$, $p_3 = 23$, $p_4 = 32$ число $Y = 118864 = (0, 0, 0, 16)$ отрицательное.

Поскольку все вычисления выполняются над целочисленными значениями малой размерности, увеличивается скорость вычисления и отсутствуют ошибки округления.

На основе рассмотренного вычислительного узла определения знака числа предложен вычислительный узел сравнения чисел X и Y, представленных в СОК [119].

Вычислительный узел для сравнения чисел X и Y поясняется рисунками 3.10–3.12. На рисунке 3.10 показана архитектура вычислительного узла сравнения чисел, на рисунке 3.11 — архитектура вычислительного блока проверки равенства, на рисунке 3.12 — архитектура вычислительного блока анализа знаков чисел, схема определения знака соответствует архитектуре вычислительного узла определения знака числа (рис. 3.9), представленного в системе остаточных классов с четным диапазоном [118].

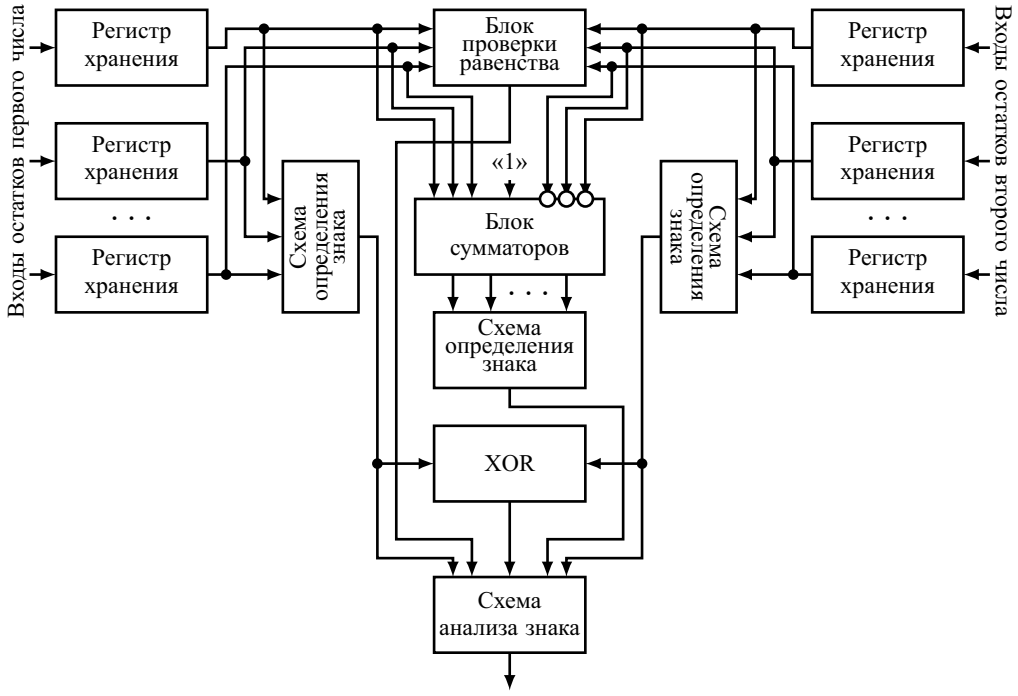

Рисунок 3.10 — Архитектура вычислительного узла сравнения чисел, представленных в системе остаточных классов

На входы остатков первого числа подаются остатки от деления числа X на модули p_1, p_2, …, p_n, которые затем поступают для хранения в регистры

хранения первого числа. На входы остатков второго числа подаются остатки от деления числа Y на модули p_1, p_2, …, p_n, которые затем поступают для хранения в регистры хранения второго числа. Далее данные с выходов регистров хранения первого числа поступают на соответствующие входы вычислительного узла определения знака первого числа, которая поясняется рисунком 3.9, данные с выходов регистров хранения второго числа поступают на соответствующие входы вычислительного узла определения знака второго числа, знаки первого и второго числа с выходов вычислительных узлов определения знака первого и второго чисел поступают на входы логического элемента XOR и первый и пятый входы вычислительного блока анализа знака. Выход логического элемента XOR поступает на третий вход схемы анализа знака.

Также данные с выходов регистров хранения первого числа поступают на первые информационные входы блока проверки равенства, на вторые информационные входы которого поступают данные с выходов регистров хранения второго числа.

Также данные с выходов регистров хранения первого числа поступают на первые информационные входы блока сумматоров, на вторые информационные входы которого через инверторы поступают данные с выходов регистров хранения второго числа, на вход переноса поступает сигнал логической единицы. Это эквивалентно вычитанию из числа X, представленного в СОК, числа Y. Результаты вычитания по каждому модулю поступают на входы вычислительного узла определения знака разности, которая поясняется рисунком 3.9. Выход вычислительного узла определения знака разности подключен к четвертому входу вычислительного блока анализа знака, которая поясняется рисунком 3.12.

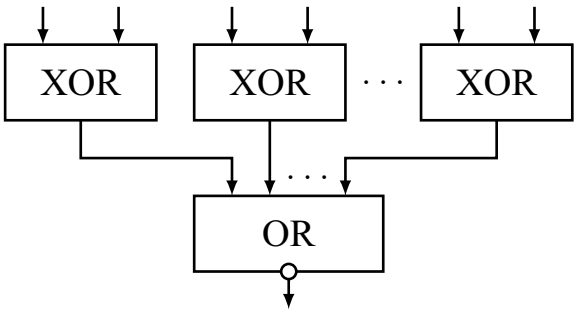

Рисунок 3.11 — Архитектура вычислительного блока проверки равенства

Архитектура вычислительного блока проверки равенства поясняется рисунком 3.11. Она содержит n логических элементов XOR и логический элемент OR. В логических элементах XOR происходит сравнение поступающих остатков

чисел X и Y по модулям p_i, поступающих на первые и вторые информационные входы блока проверки равенства. На выходы данных логических элементов XOR подается сигнал логической единицы, если на входы подаются отличные значения, и сигнал логического нуля, если на входах одинаковые значения. Логический элемент OR через инвертор выдает сигнал логической единицы в случае, если числа равны и логического нуля, если хотя бы один из остатков у чисел X и Y не совпадает. Выход блока проверки равенства подключен ко второму входу схемы анализа знака.

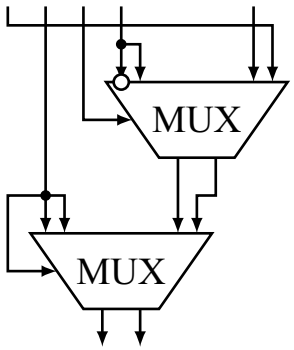

Рисунок 3.12 — Архитектура вычислительного блока анализа знаков чисел

Вычислительный блок анализа знака содержит первый и второй мультиплексоры. При этом первый вход вычислительного блока анализа знаков подключен к младшему биту второго информационного входа первого мультиплексора. Второй вход вычислительного блока анализа знаков подключен к управляющему входу и старшему и младшему битам первого информационного входа второго мультиплексора. Третий вход вычислительного блока анализа знаков подключен к управляющему входу первого мультиплексора. Четвертый вход вычислительного блока анализа знаков подключен к младшему биту и через инвертор к старшему биту первого информационного входа первого мультиплексора. Пятый вход вычислительного блока анализа знаков подключен к старшему биту второго информационного входа первого мультиплексора, выходы которого подключены ко второму информационному входу второго мультиплексора, выходы которого являются выходом вычислительного блока анализа знаков. Если на третий вход вычислительного блока анализа знака с выхода логического элемента XOR на управляющий вход первого мультиплексора поступает сигнал логической единицы, это означает, что знаки первого и второго числа, полученные в вычислительных узлах определения знака первого и второго чисел отличаются. Если знак первого числа, поступающий на первый вход вычислительного бло-

ка анализа знака равен 0 (положительное), а знак второго числа, поступающий на пятый вход вычислительного блока анализа знака равен 1 (отрицательное), то на выход первого мультиплексора подается сигнал 10 (первое число больше второго), если первое равно 1, а второе 0, то на выход первого мультиплексора подается 01 (второе число больше первого). Если значение с выхода логического элемента XOR равно 0, то на основе знака разности, поступающего с вычислительного узла определения знака разности на четвертый вход вычислительного блока анализа знака, на выход первого мультиплексора подается 01, если знак разности равен 1 (разность первого и второго числа отрицательная), и 10, если знак разности равен 0 (разность первого и второго числа положительны). Если на управляющий вход второго мультиплексора с выхода блока проверки равенства поступает 1, то на выход подается значение 11 (числа равны), иначе подается значение, полученное от первого мультиплексора.

На основе примера рассмотрим работу вычислительного узла для сравнения чисел. Пусть задана система остаточных классов с модулями $p_1 = 17$, $p_2 = 19$, $p_3 = 23$, $p_4 = 32$. Тогда количество входов остатка первого числа и второго числа, регистров хранения первого числа и второго числа равно 4, а количество вычислительных ступеней в вычислительных узлах определения знака первого числа, второго числа и разности равно 3.

Если на входы остатков первого и второго чисел поступают числа $X = 118863 = (16, 18, 22, 15)$ и $Y = 118863 = (16, 18, 22, 15)$, то через регистры хранения первого и второго числа значения поступают на блок проверки равенства, и поскольку числа равны, то на второй вход вычислительного блока анализа знака поступает логическая единица, которая поступает на управляющий и первый информационный входы второго мультиплексора и на выход сравнения поступает сигнал 11, который означает, что «первое число равно второму».

Если на входы остатков первого и второго чисел поступают числа $X = 118863 = (16, 18, 22, 15)$ и $Y = 118864 = (0, 0, 0, 16)$, то через регистры хранения первого и второго числа значения поступают в вычислительные узлы определения знака первого и второго чисел. В вычислительном узле определения знака первого числа после вычислений на выход подается 0, таким образом, в СОК с модулями $p_1 = 17$, $p_2 = 19$, $p_3 = 23$, $p_4 = 32$ число $X = 118863 = (16, 18, 22, 15)$ положительное. Одновременно с этим происходит определение знака $Y = 118864 = (0, 0, 0, 16)$ в вычислительном

узле определения знака второго числа и поскольку выход блока равен 1, то $Y = 118864 = (0, 0, 0, 16)$ отрицательное.

Значения 0 и 1 с выходов вычислительных узлов определения знака первого и второго чисел соответственно поступают на первый и пятый входы вычислительного блока анализа знака и входы логического элемента XOR и поскольку они не равны, на выходе логического элемента XOR будет сигнал логической единицы, который поступает на третий вход вычислительного блока анализа знака, где он подается на управляющий вход первого мультиплексора, на выход которого со второго информационного входа поступают значения знаков второго и первого чисел, т.е. 10. Поскольку значение равенства с выхода блока проверки равенства равно 0, то на выход второго мультиплексора, который является выходом сравнения вычислительного узла будет подан сигнал со второго информационного входа, т.е. 10, что означает «первое число больше второго».

В случае, когда первое и второе число имеют один знак, который поступает на входы логического элемента XOR, то выход сравнения зависит от знака, поступающего из вычислительного блока определения знака разности.

Поскольку все вычисления выполняются над целочисленными значениями малой размерности, увеличивается скорость вычисления и отсутствуют ошибки округления. Реализация вычислительного узла возможна с использованием программируемых логических интегральных схем или в составе программных вычислительных систем.

Также в параграфе 2.3 приведена функция (2.39):

$$f(X) = \left| \sum_{i=1}^{n} \overline{k_i} x_i \right|_{2^N},$$

где $\overline{k_i} = \left\lfloor \frac{2^N \left| P^{-1} \right|_{p_i}}{p_i} \right\rfloor$.

Для данной функции доказана теорема 2.3.2, в которой сказано, что при $N = \lceil \log_2 (-n + n \cdot P) \rceil$ функция $f(X)$ является строго возрастающей.

Таким образом, получена уточненная оценка точности для приближенного метода на основе КТО. Результаты моделирования сравнения в СОК на основе уточненной оценки точности приближенного метода на основе КТО представлены в таблице 30 приложения В.

Таким образом, проведено моделирование методов сравнения чисел в СОК на основе Китайской теоремы об остатках, её приближенной реализации, обобщенной позиционной системы счисления, а также предложенной уточненной

оценки приближенного метода на основе Китайской теоремы об остатках и монотонной функции ядра без критических ядер. Для сравнения данных методов из таблиц 26–30 выберем для каждой размерности лучшие значения по времени и площади в таблицы 31–32. На основе полученных данных построим графики сравнения рассмотренных методов.

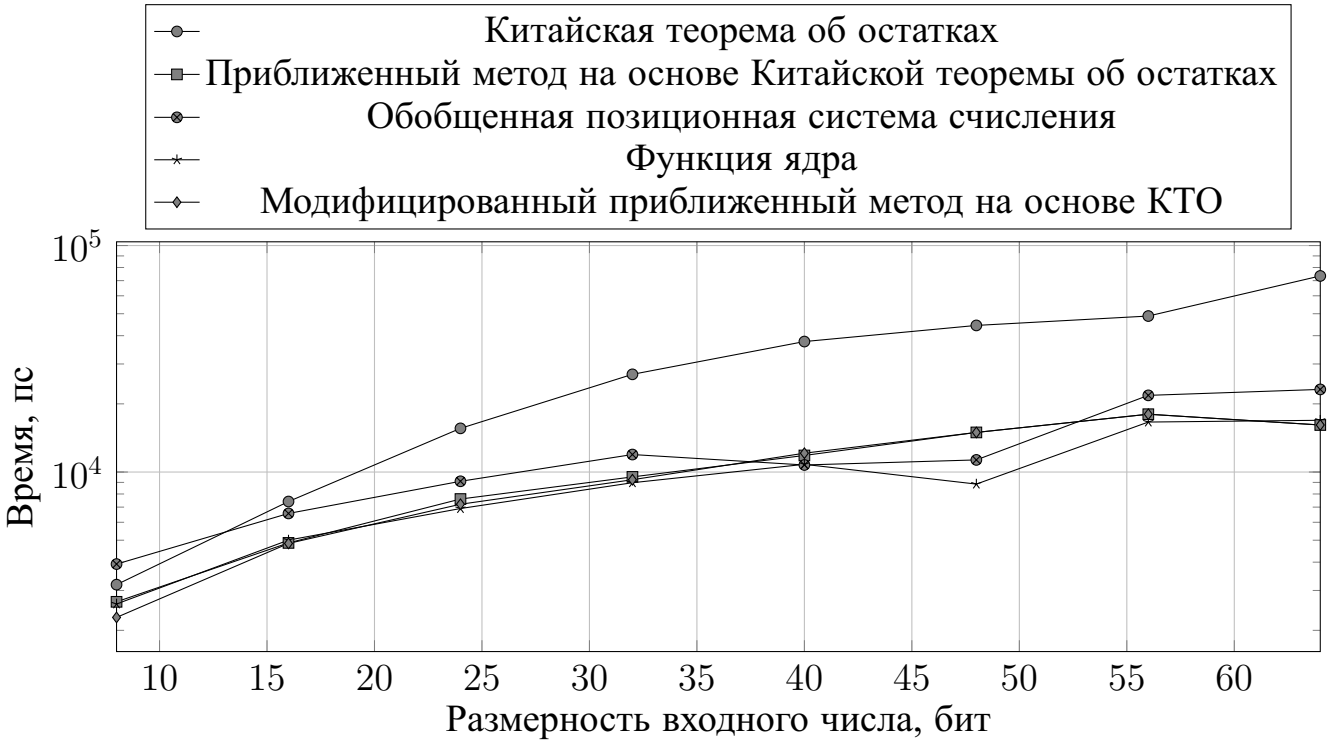

Рисунок 3.13 — Сравнение времени для методов сравнения чисел в СОК.

Из графиков 3.13–3.14 можно сделать вывод что приближенный метод на основе КТО и предложенная модификация, а также предложенная функция ядра имеют лучшие характеристики по времени работы, по сравнению с КТО и ОПСС, однако используемая площадь у ОПСС значительно ниже.

Предложенная оценка точности приближенного метода в большинстве случаем показывает лучшие результаты по сравнению с приближенным методом из статей [75] и [86].

Функция ядра показывает сопоставимые с модифицированным приближенным методом на основе КТО результаты, и в ряде случаев имеет меньшее время вычислений. При этом в функции ядра нет округлений, она является точным методом.

Таким образом, разработанные методы имеют лучшие результаты по времени вычислений по сравнению с известными.

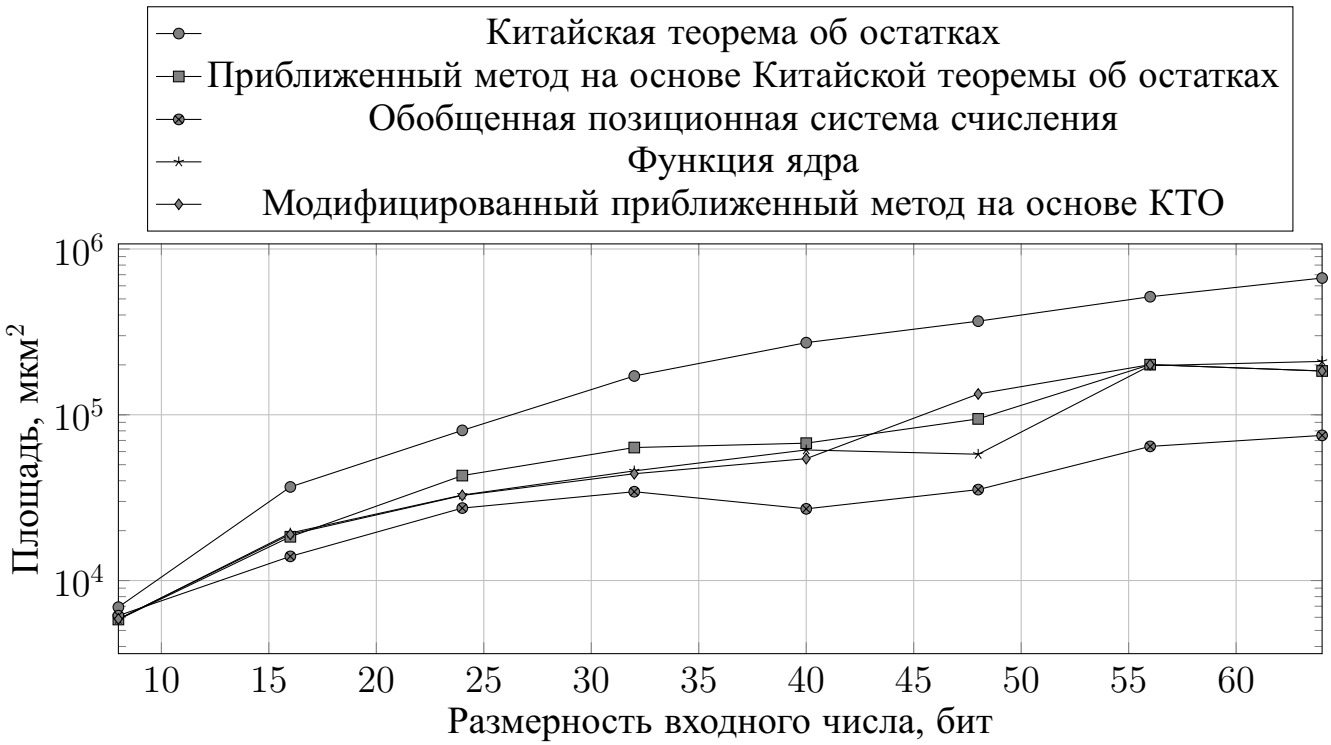

Рисунок 3.14 — Сравнение используемой площади для методов сравнения чисел в СОК.

3.5 Архитектура вычислительного узла для умножения в СОК

В Параграфе 2.4 рассмотрена реализация умножения с накоплением. Выполнение операции умножения с накоплением (MAC) $P = C + A \times B$ является основной операцией для многих задач цифровой обработки сигналов. Повышение эффективности вычисления умножения с накоплением возможно за счет использования непозиционных систем счисления, в которых нет необходимости учитывать межразрядные переносы. При выборе системы счисления можно непосредственно снизить количество операций, длину операндов, количество и/или длину глобальных соединений, что может привести к снижению площади, задержки и рассеивания мощности [59].

Для моделирования и сравнения был взят диапазон от 8 до 64 бит с шагом 8 бит.

Для позиционной системы счисления взяты множители размером $n/2$ бит, слагаемое размером n бит, результат n бит для возможности конвейерной обработки с учетом, что старший $n + 1$-й бит суммы отбрасывается (т.е. подразумевается, что результат сложения не выходит за диапазон n бит).

В системе остаточных классов наборы модулей выбирались так, чтобы для динамического диапазона P выполнялось условие $\lfloor \log_2 P \rfloor = n$ бит.

В качестве метода нахождения остатка от деления использован период числа, рассмотренный в Параграфе 2.1, ввиду простоты его реализации и масштабируемости. Реализация МАС в СОК использует вычисление умножения с последующим нахождением остатка при вычислении суммы. Псевдокод данной реализации представлен Алгоритмом 3.1, где $A\,[a:b]$ означает выборку бит с a до b сигнал A.

Алгоритм 3.1. Умножение с накоплением по модулю $p = 2^n - 1$

Вход: $A, B, C \in [0, p)$

Выход: $P = (A \cdot B + C) \mod p$

1: $Mult = A \cdot B$
2: $Sum_1 = Mult\,[n-1:0] + Mult\,[2n-1:n] + C$
3: $Sum_2 = Sum_1\,[n-1:0] + Sum_1\,[n+1:n]$
4: $Sum_{out} = Sum_2 + 1$
5: **Если** $Sum_{out}[n] = 1$ **тогда**
6: **Возвратить** $P = Sum_{out}\,[n-1:0]$
7: **иначе**
8: **Возвратить** $P = Sum_2\,[n-1:0]$
9: **Конец условия**

Преимущества системы остаточных классов связаны с независимостью вычислений по каждому модулю, таким образом, взяты трех-, четырех-, пяти- и т.д. модульные наборы, чтобы максимально раскрыть потенциал параллелизма вычислений. Промоделированные наборы модулей представлены в таблице 7.

Из таблицы 33 приложения Г видно, что выигрыш во времени и площади растет с увеличением количества модулей и как следствие с уменьшением их размера.

Сравнение двоичной реализации МАС с реализациями в СОК представлено на рисунках 3.15-3.16, откуда видно, что начиная с 24 бит реализация с использованием системы остаточных классов выигрывает по всем параметрам.

Применение системы остаточных классов со специальными модулями вида $2^n - 1$, 2^n, $2^n + 1$ с использованием периода числа позволяет на диапазонах от 16 бит сократить время выполнения, на диапазонах от 24 бит сократить требуемую площадь.

Таблица 7 — Наборы СОК, выбранные для моделирования

Размерность динамического диапазона, бит	Наборы модулей СОК
8	$\{3, 5, 32\}$, $\{5, 7, 8\}$
16	$\{31, 63, 64\}$, $\{7, 9, 17, 64\}$
24	$\{255, 257, 512\}$, $\{31, 63, 65, 256\}$, $\{5, 7, 9, 17, 31, 128\}$
32	$\{1023, 2047, 4096\}$, $\{127, 255, 511, 512\}$, $\{31, 63, 65, 127, 51\}$
40	$\{8191, 16383, 16384\}$, $\{511, 1023, 2047, 2048\}$, $\{127, 255, 257, 511, 512\}$, $\{17, 31, 65, 129, 511, 512\}$
48	$\{65535, 65537, 131072\}$, $\{2047, 4095, 8191, 8192\}$, $\{257, 511, 1023, 1025, 2048\}$, $\{65, 127, 129, 257, 511, 2048\}$, $\{17, 31, 65, 127, 129, 511, 1024\}$
56	$\{262143, 524287, 1048576\}$, $\{8191, 16383, 32767, 32768\}$, $\{511, 1025, 2049, 8191, 16384\}$, $\{127, 257, 511, 513, 2047, 8192\}$, $\{31, 65, 127, 257, 513, 2047, 2048\}$, $\{17, 31, 65, 127, 129, 257, 511, 1024\}$
64	$\{2097151, 4194303, 4194304\}$, $\{32767, 65535, 131071, 131072\}$, $\{2047, 4097, 8191, 16383, 32768\}$, $\{257, 511, 1025, 2049, 8191, 8192\}$, $\{65, 127, 257, 511, 1023, 2047, 8192\}$

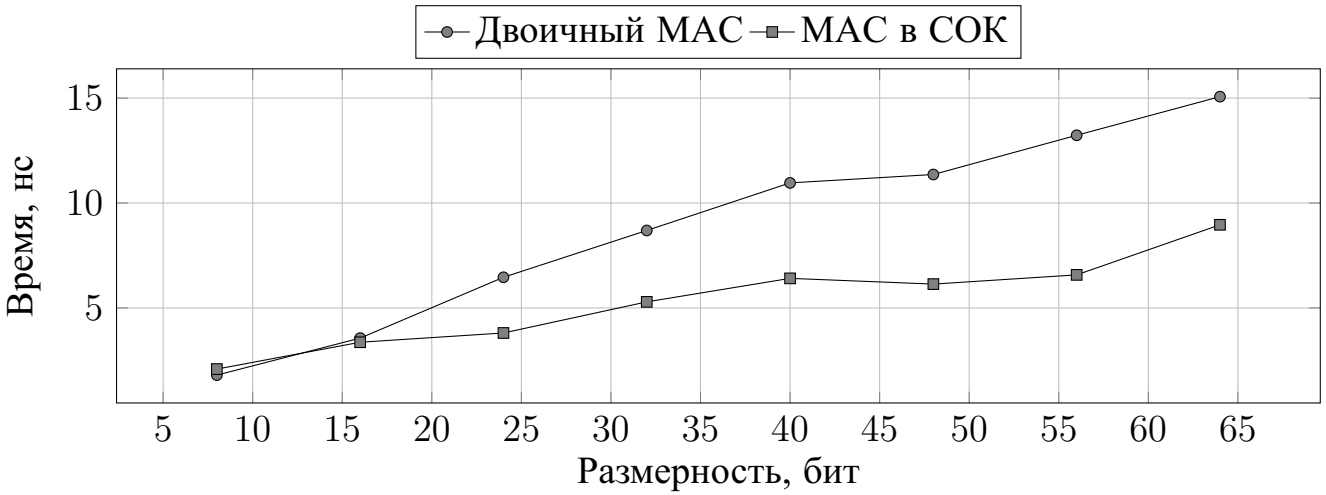

Рисунок 3.15 — График времени выполения MAC

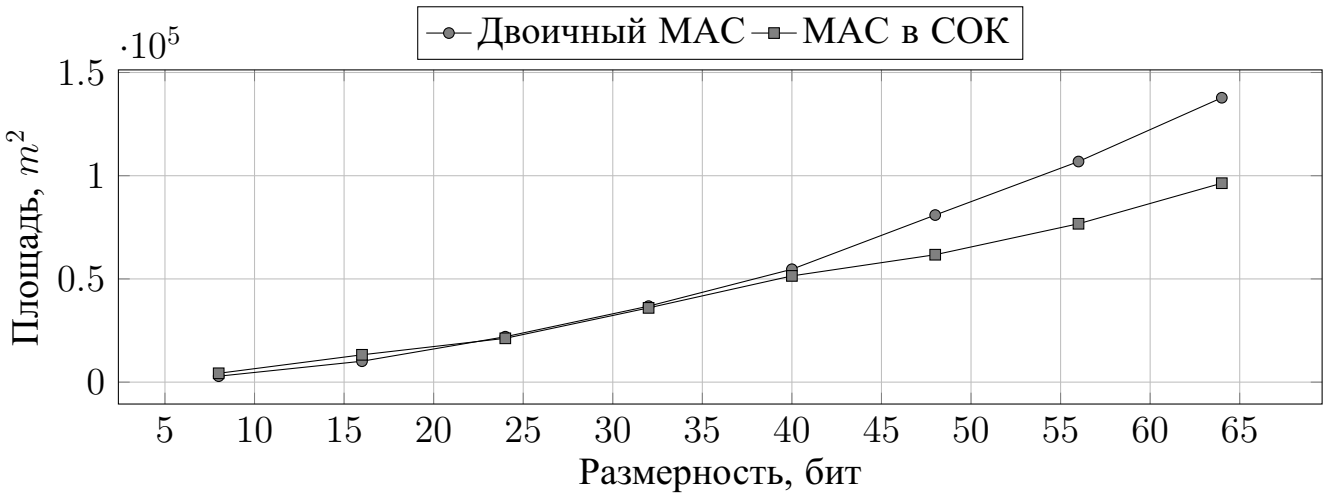

Рисунок 3.16 — График площади MAC

Анализ таблицы 33 показывает, что лучшие результаты достигаются СОК с большим количеством модулей, при этом стоит проблема их несбалансированности по размеру. При этом четырехмодульные наборы со сбалансированными модулями также дают преимущество по сравнению с двоичной реализацией, хотя и не такое явное. При этом не рассмотрен вопрос специализированных умножителей с предсказанием переноса и их модификации.

3.6 Архитектура вычислительного узла распределенной среды для обнаружения, локализации и исправления ошибок в СОК

Для анализа и моделирования предложенного Алгоритма 2.33 с одним избыточным модулем были выбраны наборы модулей СОК с 4, 5, 6 рабочими основаниями, рабочий диапазон которых покрывает диапазон в 8, 16, 24 и 32 бита. Избыточность данных является важным вопросом. В избыточной (k,n) СОК, используя любые k остатков из n, можно восстановить данные. Тогда избыточность можно выразить выражением

$$\frac{\sum_{i=1}^{n} d_i}{\sum_{i=1}^{k} d_i} - 1,$$

где d_i размерность остатков по основанию p_i.

Для моделирования работы Алгоритма 1.1 были выбраны наборы с двумя контрольными основаниями и вычислена их избыточность. Результаты представлены в Таблице 8.

Таблица 8 — Избыточность ИСОК с двумя контрольными основаниями

Набор модулей СОК	Размерность, бит	Избыточность
{3, 5, 7, 11, 13, 17}	10	0.75
{13, 17, 19, 23, 29, 31}	16	0.53
{59, 61, 67, 71, 73, 79}	24	0.54
{251, 257, 263, 269, 271, 277}	32	0.51
{2, 3, 5, 7, 11, 13, 17}	11	0.69
{5, 7, 11, 13, 17, 19, 23}	16	0.53
{23, 29, 31, 37, 41, 43, 47}	24	0.44
{79, 83, 89, 97, 101, 103, 107}	32	0.40
{2, 3, 5, 7, 11, 13, 17, 19}	14	0.59
{3, 5, 7, 11, 13, 17, 19, 23}	17	0.48
{11, 13, 17, 19, 23, 29, 31, 37}	24	0.39
{31, 37, 41, 43, 47, 53, 59, 61}	32	0.34

Аналогично для моделирования работы Алгоритма 2.33 были выбраны наборы с одним контрольным основанием, удовлетворяющие условиям теоремы 2.5.1 и вычислена их избыточность. Результаты представлены в таблице 9.

В среднем избыточность предложенного Алгоритма 2.33 на 14% меньше, чем у СОК с двумя контрольными основаниями.

Метод проекций, представленный Алгоритмом 1.1 требует большого количества восстановлений чисел с использованием КТО для каждой проекции. На рис. 3.17 показано уменьшение количества комбинаций предлагаемого метода по сравнению с методом проекций.

Одним из возможных применений Алгоритма 2.33 может быть система распределенного хранения данных [120], в которой выполняется подготовка исходных файлов для надежного распределенного хранения, посредством перевода в систему остаточных классов, удовлетворяющую теореме 2.5.1 и для восстановления полученных файлов, принятых из распределенной среды в случае ошибки или неполучения одной из частей файла по Алгоритму 2.33.

Существует большое количество систем хранения данных, описанных в патентах США, России, Китая [заявка WO2019199288, опубл. 17.10.2019, заявка

Таблица 9 — Избыточность ИСОК с одним контрольным основанием

Набор модулей СОК	Размерность, бит	Избыточность
{3, 5, 7, 11, 79}	10	0.58
{13, 17, 19, 23, 439}	16	0.47
{59, 61, 67, 71, 4759}	24	0.50
{251, 257, 263, 269, 70753}	32	0.49
{2, 3, 5, 7, 11, 79}	11	0.54
{5, 7, 11, 13, 17, 223}	16	0.42
{23, 29, 31, 37, 41, 1523}	24	0.41
{79, 83, 89, 97, 101, 9803}	32	0.40
{2, 3, 5, 7, 11, 13, 149}	14	0.47
{3, 5, 7, 11, 13, 17, 223}	17	0.38
{11, 13, 17, 19, 23, 29, 673}	24	0.36
{31, 37, 41, 43, 47, 53, 2503}	32	0.34

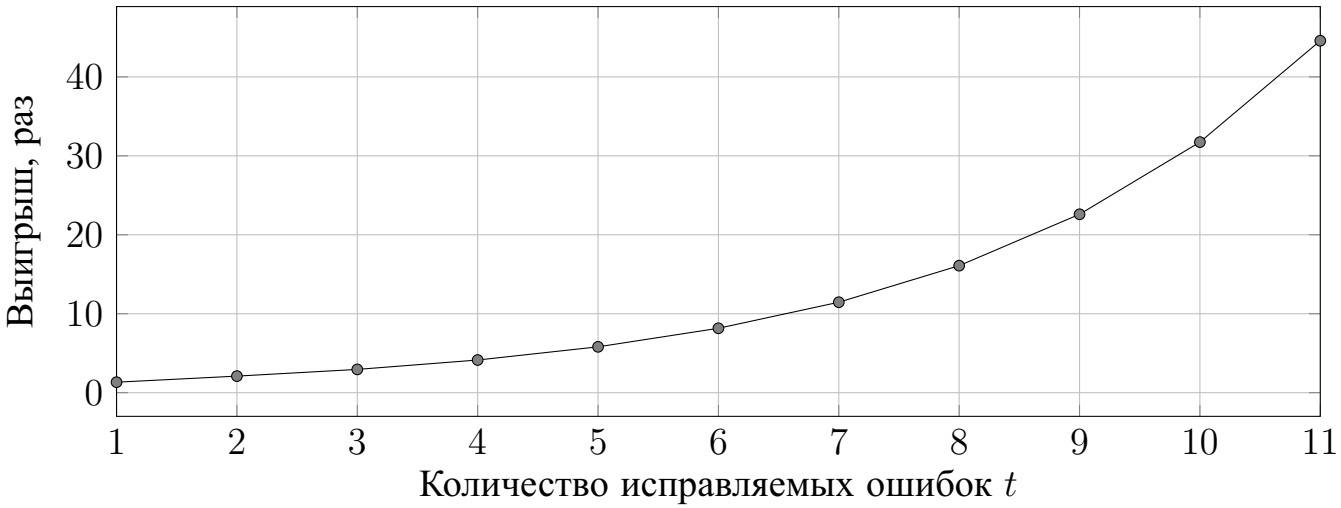

Рисунок 3.17 — Соотношение количества комбинаций метода проекций $(2t-1, 2t)$ к количеству комбинаций предлагаемого метода $(2t-1,t)$.

US2013173916, опубл. 04.07.2013, патент RU2656836, опубл. 06.06.2018, патент CN103957264, опубл. 30.07.2014].

Предложенная система может быть реализована архитектурой, представленной на рисунке 3.18. Исходное значение X поступает на входы блоков $\mathrm{mod}\ p_i$ нахождения остатков по модулю p_i, $i \in [1,n+1]$, которые могут быть выполнены как с использованием вычислительных устройств, например, интегральных схем или FPGA, так и в виде памяти, которая в ответ на значение исходного числа X подает на выход блока $\mathrm{mod}\ p_i$ остаток x_i от деления на модуль p_i. При этом модули удовлетворяют условиям $p_1 < p_2 < p_3 < \ldots < p_n < p_{n+1}$ и $p_{n+1} > p_n \cdot p_{n-1}$.

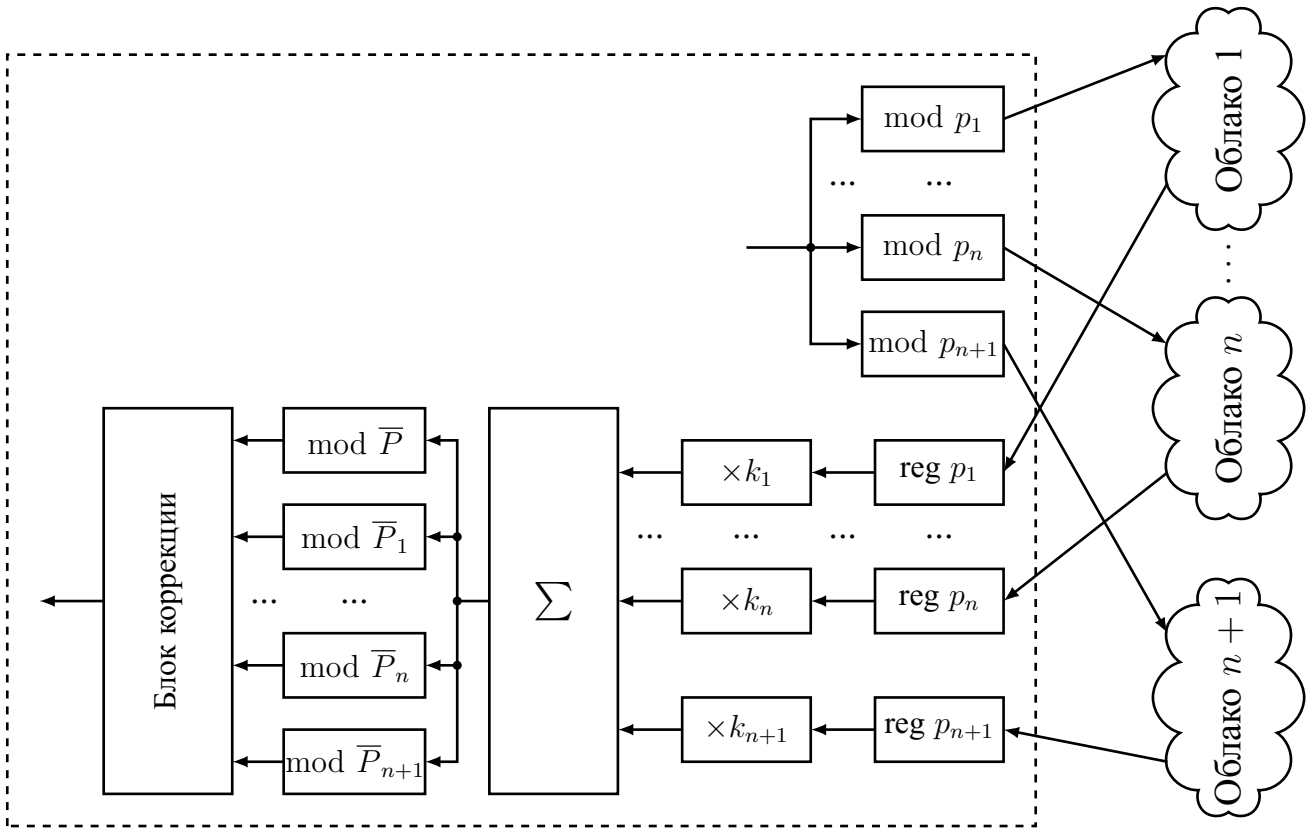

Рисунок 3.18 — Архитектура системы распределенного хранения данных

Данные с выходов блоков нахождения остатков по модулю p_i передаются в распределенное хранилище, которое может быть использовано как для хранения, так и обработки данных, поскольку особенностью системы остаточных классов является возможность выполнения арифметических операций сложения и умножения независимо по каждому модулю. При этом распределенное хранилище может быть представлено одним или несколькими облачными провайдерами, или внутренними частными облаками/хранилищами, что позволит распределить

данные при хранении, тем самым повысить надежность восстановления данных в случае выхода из строя одного или нескольких хранилищ за счет избыточной структуры системы остаточных классов.

После хранения данные поступают из распределенного хранилища на входы регистров хранения остатков по модулю p_i, выходы которых соединены со входами соответствующих блоков умножения на $k_i = w_i\overline{P}_i$, выходы которых подключены к входами сумматора произведений, значение суммы поступает на входы блока нахождения остатков по модулю $\overline{P}$ и блоков нахождения остатков по модулю $\overline{P}_i$, результаты с которых поступают на вход блока коррекции ошибки, выход которого является выходом системы.

На рисунке 3.19 показана архитектура вычислительного блока коррекции ошибки. Значение $|S|_{\overline{P}}$ поступает на блок сравнения $|S|_{\overline{P}}$ с рабочим диапазоном, в котором проверяется логическое выражение $|S|_{\overline{P}} < P$, если оно истинно, то сигнал с выхода блок сравнения $|S|_{\overline{P}}$ с рабочим диапазоном поступает на управляющий вход мультиплексора, пропуская корректное значение $|S|_{\overline{P}}$ через первый информационный вход на выход блока коррекции ошибки. Второй информационный вход мультиплексора $|S|_{\overline{P}}$ подключен к выходу мультиплексора $|S|_{\overline{P}_1}$.

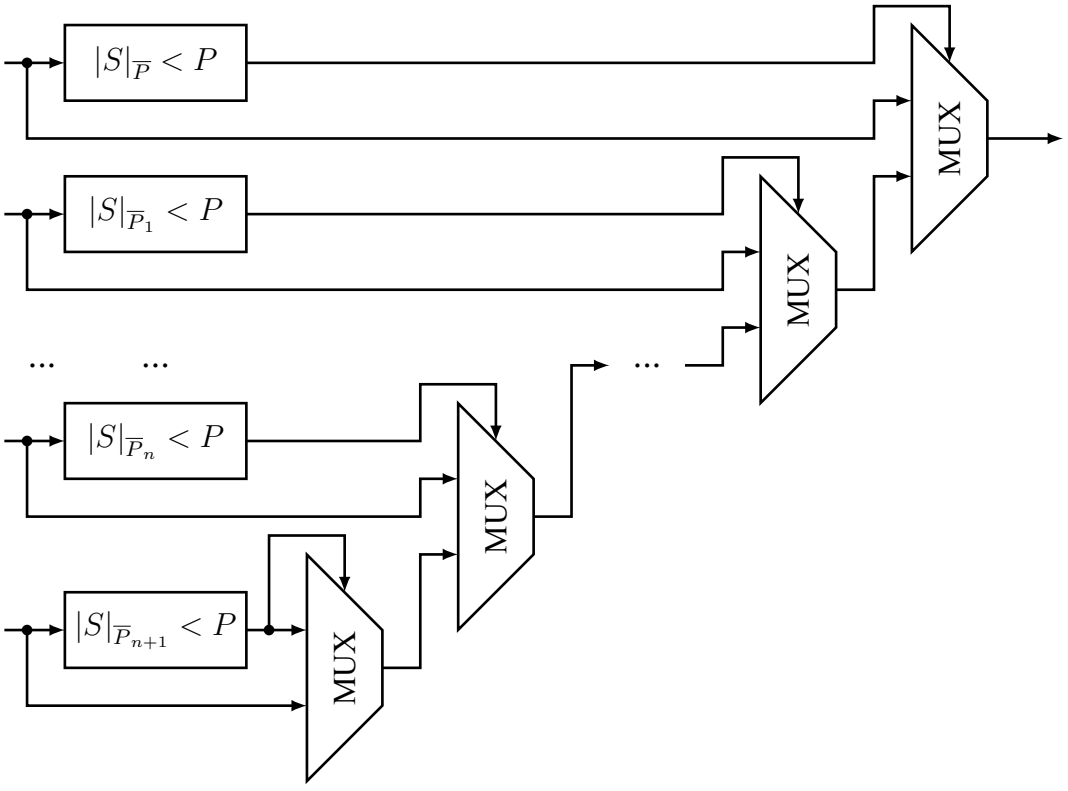

Рисунок 3.19 — Архитектура вычислительного блока коррекции

Значения $|S|_{\overline{P}_i}$ с выходов блоков нахождения остатков по модулю $\overline{P}_i$ поступают на входы соответствующих блоков сравнения $|S|_{\overline{P}_i}$ с рабочим диапазоном, в которых проверяются логические выражения $|S|_{\overline{P}_i} < P$, если оно не выполняется, т.е. равно 0, то сигнал 0 с выхода блок сравнения $|S|_{\overline{P}_i}$ с рабочим диапазоном поступает на управляющий вход мультиплексора $|S|_{\overline{P}_i}$, на первый информационный вход которого подается значения $|S|_{\overline{P}_i}$ с выхода блока нахождения остатков по модулю $\overline{P}_i$. Выход мультиплексора $|S|_{\overline{P}_i}$ подключен ко второму информационному входу мультиплексора $|S|_{\overline{P}_{i-1}}$. Второй информационный вход мультиплексора $|S|_{\overline{P}_{n+1}}$ подключен к выходу блок сравнения $|S|_{\overline{P}_{n+1}}$.

Блок коррекции ошибки осуществляет вывод первого значения $|S|_{\overline{P}}$ или $|S|_{\overline{P}_i}$, которое меньше рабочего диапазона.

Преимуществом Алгоритма 2.33 является снижение аппаратных издержек коррекции ошибок модулярных чисел, полученных из систем распределенного хранения данных, что связано с отсутствием необходимости вычисления $S = \sum_{i=1}^{n+1} k_i\, x_i'$ для каждой проекции $\overline{P}_i$, S вычисляется один раз, и в дальнейшем необходимо вычисления только остатка от деления. Реализация всей системы возможна с использованием программируемых логических интегральных схем, специализированных интегральных схем, а также в виде программы для ЭВМ для выполнения подготовки файлов к надежному распределенному хранению данных.

Однако недостатком данного подхода является использование несбалансированной СОК и необходимость обеспечения надежности вычислений по контрольному основанию.

Для обеспечения точной и надежной передачи, обработки информации в вычислительных системах и расширения функциональных возможностей аналогов, в частности [115], введем вычислительный узел обнаружения и коррекции ошибок модулярного кода [116], который позволяет не только обнаруживать, но и исправлять ошибки модулярного кода. Поясним вычислительный узел рисунками 3.20–3.22.

На рисунке 3.20 представлена архитектура вычислительного узла обнаружения и коррекции ошибок, которая содержит входы остатка α_i, где $i = 1, \ldots, n+2$, регистры хранения остатка α_i, регистры хранения остатков π_{n+1} и π_{n+2}, блок формирования проекций, блоки LUT α_i хранения произведений $k_i\alpha_i$, блоки LUT π_{n+i} хранения произведений $k_{n+1}\pi_{n+1}$ и $k_{n+2}\pi_{n+2}$, сумматор произведений $k_i\alpha_i$, сумматор произведений $k_{n+1}\pi_{n+1}$ и $k_{n+2}\pi_{n+2}$, блок управле-

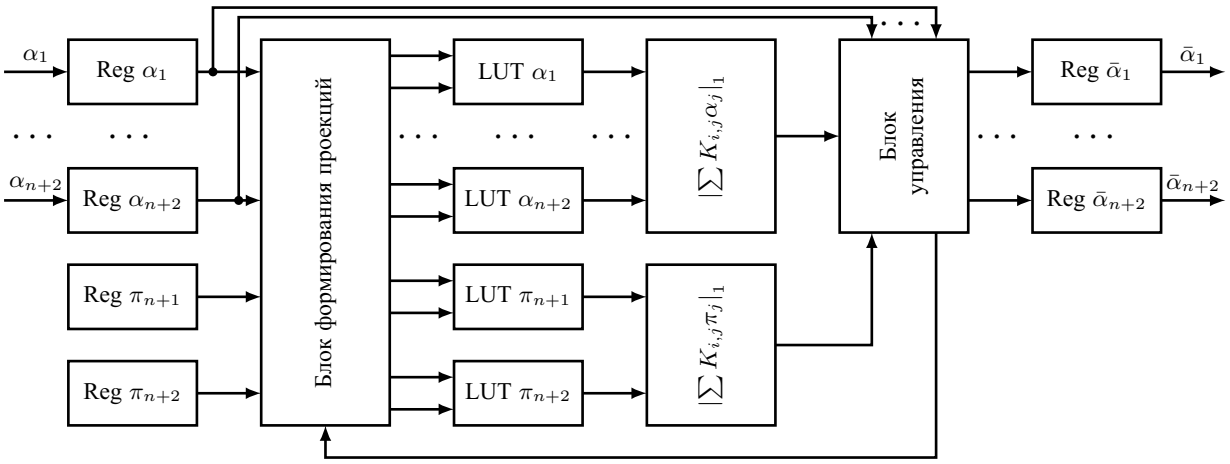

Рисунок 3.20 — Архитектура вычислительного узла обнаружения и коррекции ошибок

ния, регистры хранения скорректированного остатка α_i, выходы скорректированного остатка α_i.

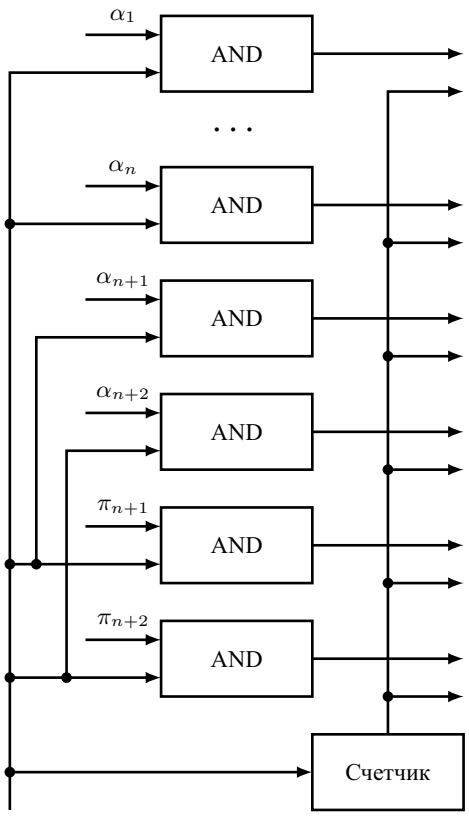

Рисунок 3.21 — Архитектура вычислительного блока формирования проекций

На рисунке 3.21 показана архитектура вычислительного блока формирования проекций, состоящего из элементов AND формирования проекций α_i, элементов AND формирования проекций π_{n+1} и π_{n+2} и счетчика.

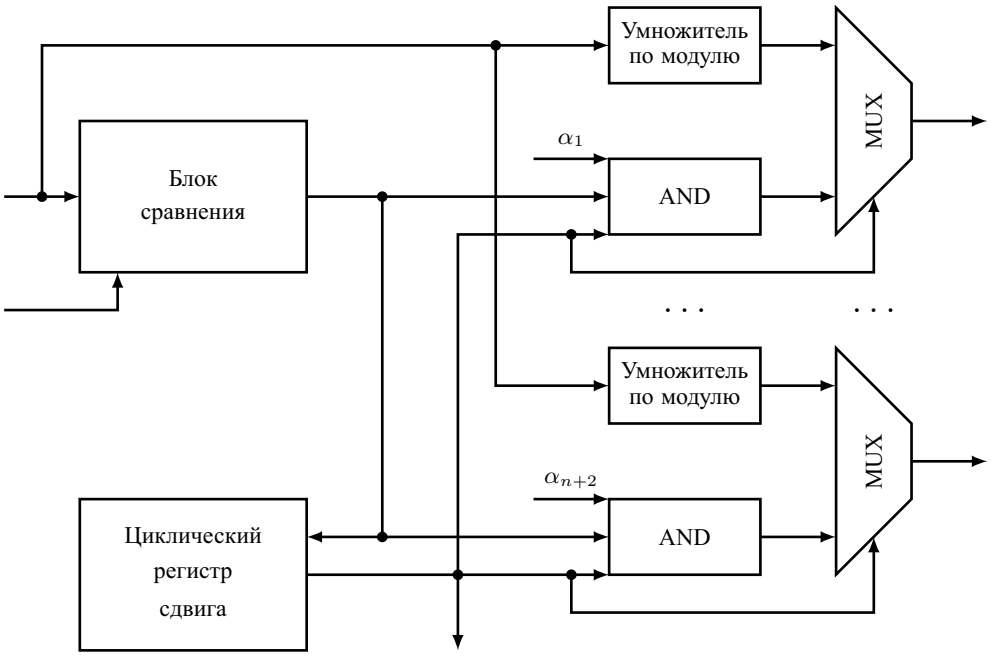

Рисунок 3.22 — Архитектура блока управления

На рисунке 3.22 представлена архитектура блока управления, который состоит из блока сравнения, $n+3$-битного циклического регистра сдвига, модулярных умножителей на $\overline{P}_i$ по модулю p_i, элементов AND локализации ошибки по модулю p_i, мультиплексоров выбора корректного остатка α_i.

Поясним работу вычислительного узла обнаружения и коррекции ошибок примерами. Пусть задана СОК с модулями $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, $p_5 = 11$, $p_6 = 13$, $p_7 = 17$, следовательно $n = 5$. Значение рабочего диапазона $P = 2310$. В качестве сомножителей в модулярных умножителях на $\overline{P}_i$ по модулю p_i блока управления используются следующие значения: $\overline{P}_1 = 255255$, $\overline{P}_2 = 170170$, $\overline{P}_3 = 102102$, $\overline{P}_4 = 72930$, $\overline{P}_5 = 46410$, $\overline{P}_6 = 39270$, $\overline{P}_7 = 30030$.

В блок хранения произведения $k_1\alpha_1$ записывают все произведения $k_1 \cdot \alpha_1$ при $\alpha_1 = [0, p_1) = [0, 1]$ и k_1, выбираемом из таблицы 10 в зависимости от адреса со счетчика, при этом для удобства значения представлены в виде обыкновенных дробей. Так, например, на вход блока хранения произведения $k_1\alpha_1$ подается значение $\alpha_1 = 1$ и адрес 0, тогда $k_1 = \frac{1}{2}$ и на выход блока хранения произведения $k_1\alpha_1$ будет подано значение 0.5.

Аналогично в блоке хранения произведения $k_2\alpha_2$ записаны все произведения $k_2 \cdot \alpha_2$ при $\alpha_2 = [0, p_2) = [0, 2]$ и k_2, выбираемом из таблицы 10 в зависимости от адреса со счетчика.

В блоке хранения произведения $k_3\alpha_3$ записаны все произведения $k_3 \cdot \alpha_3$ при $\alpha_3 = [0, p_3) = [0, 4]$ и k_3, выбираемом из таблицы 10 в зависимости от адреса со счетчика.

В блоке хранения произведения $k_4\alpha_4$ записаны все произведения $k_4 \cdot \alpha_4$ при $\alpha_4 = [0, p_4) = [0, 6]$ и k_4, выбираемом из таблицы 10 в зависимости от адреса со счетчика.

В блоке хранения произведения $k_5\alpha_5$ записаны все произведения $k_5 \cdot \alpha_5$ при $\alpha_5 = [0, p_5) = [0, 10]$ и k_5, выбираемом из таблицы 10 в зависимости от адреса со счетчика.

В блоках хранения произведения $k_6\alpha_6$ и хранения произведения $k_6\pi_6$ записаны все произведения $k_6 \cdot \alpha_6$ и $k_6 \cdot \pi_6$ при $\alpha_6 = [0, p_6) = [0{,}12]$, $\pi_6 = \{0, P \bmod p_6\} = \{0, 9\}$ и k_6, выбираемом из таблицы 10 в зависимости от адреса со счетчика.

В блоках хранения произведения $k_7\alpha_7$ и хранения произведения $k_7\pi_7$ записаны все произведения $k_7 \cdot \alpha_7$ и $k_7 \cdot \pi_7$ при $\alpha_7 = [0, p_7) = [0{,}16]$, $\pi_7 = \{0, P \bmod p_7\} = \{0, 15\}$ и k_7, выбираемом из таблицы 10 в зависимости от адреса со счетчика.

Адрес	0	1	2	3	4	5	6	7
k_1	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
k_2	$\frac{1}{3}$	$\frac{2}{3}$	0	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{2}{3}$
k_3	$\frac{3}{5}$	$\frac{1}{5}$	$\frac{4}{5}$	0	$\frac{1}{5}$	$\frac{3}{5}$	$\frac{4}{5}$	$\frac{1}{5}$
k_4	$\frac{2}{7}$	$\frac{4}{7}$	$\frac{6}{7}$	$\frac{3}{7}$	0	$\frac{1}{7}$	$\frac{5}{7}$	$\frac{6}{7}$
k_5	$\frac{1}{11}$	$\frac{2}{11}$	$\frac{3}{11}$	$\frac{5}{11}$	$\frac{7}{11}$	0	$\frac{2}{11}$	$\frac{6}{11}$
k_6	$\frac{4}{13}$	$\frac{8}{13}$	$\frac{12}{13}$	$\frac{7}{13}$	$\frac{2}{13}$	$\frac{5}{13}$	0	$\frac{3}{13}$
k_7	$\frac{15}{17}$	$\frac{13}{17}$	$\frac{11}{17}$	$\frac{7}{17}$	$\frac{3}{17}$	$\frac{12}{17}$	$\frac{8}{17}$	0

Таблица 10 — Значения k_i блоков хранения произведений

Рассмотрим конструкцию и принцип работы $n + 3$-битного циклического регистра сдвига. В начале работы он содержит значение “0111...11”. Старший бит, который на первом шаге равен 0 подается на счетчик, который подает адрес 0 на блоки хранения произведений $k_i\alpha_i$ и $k_6\pi_6$ и $k_7\pi_7$. Остальные биты данного числа подаются на элементы AND формирования проекций α_i, элементы AND формирования проекций π_6 и π_7, элементы AND локализации ошибки по модулю p_i. В случае возникновения ошибки с блока сравнения в $n+3$-битный цикли-

ческий регистр сдвига поступает сигнал и его значение меняется на “1011...11”. Тогда на счетчик подается старший бит, равный 1 и адрес 1 подается на блоки хранения произведений $k_i\alpha_i$ и $k_6\pi_6$ и $k_7\pi_7$. Остальные биты данного числа подаются на элементы AND формирования проекций α_i, элементы AND формирования проекций π_6 и π_7, элементы AND локализации ошибки по модулю p_i. При этом $n+2$-й бит со значением 0 подается на элемент AND формирования проекций α_1 и элемент AND локализации ошибки по модулю p_1, что соответствует вычеркиванию первого основания в методе проекций. Аналогично нулевой бит на каждом шаге сдвигается вправо до достижения значения “1111...10”, при этом нулевой бит во всех случаях в результате логического умножения подает на выход соответствующих элементов нулевое значение, что означает выбор проекции.

Рассмотрим вычисления по схеме (рисунок 3.20) на примере значения $A = (1, 0, 1, 6, 10, 12, 12) = 2001$. На входы остатков $\alpha_1, \ldots, \alpha_7$ подаются соответственно значения 1, 0, 1, 6, 10, 12, 12, которые затем записываются в регистры хранения остатков $\alpha_1, \ldots, \alpha_7$.

Значения поступают в блок формирования проекций, где проходя через элементы AND формирования проекций $\alpha_1, \ldots, \alpha_7$ поступают на соответствующие первые входы блоков хранения произведений $k_i\alpha_i$, на второй вход которых подается адрес 0. С выходов блоков хранения произведений $k_i\alpha_i$ значения соответствующих произведений поступают на сумматор произведений $k_i\alpha_i$, где происходит суммирование по модулю 1 и на выходе сформируется значение, эквивалентное 0.0039196. В это время аналогичные процессы проходят для коэффициентов π_6 и π_7 и на выходе сумматора произведений $k_6\pi_6$ и $k_7\pi_7$ получится значение 0.0045249.

В блоке сравнения данные значения сравниваются и поскольку $0.0039196 < 0.0045249$, то на выходе блока сравнения будет значение 1, которое поступает на входы элементов AND локализации ошибки по модулю p_i, на вход которых также поступают значения $1, 0, 1, 6, 10, 12, 12$ с выходов регистров хранения остатков $\alpha_1, \ldots, \alpha_7$. С выхода $n+3$-битного циклического регистра сдвига на входы элементов AND локализации ошибки по модулю p_i и на управляющие входы мультиплексоров выбора корректного остатка α_i поступают значения 1. Таким образом, мультиплексоры выбора корректного остатка α_i подают значения $1, 0, 1, 6, 10, 12, 12$ с выходов элементов AND локализации ошибки по модулю p_i на соответствующие регистры хранения скорректиро-

ванного остатка α_i, откуда значения поступают на выходы скорректированного остатка $\alpha_1, \ldots, \alpha_7$. Таким образом на выход вычислительного блока подается корректное значение.

Рассмотрим случай, когда по третьему основанию возникла ошибка и на входы остатков $\alpha_1, \ldots, \alpha_7$ поступило значение $(1, 0, 4, 6, 10, 12, 12) = 410409$. Работа вычислительного узла происходит аналогично и на выходе сумматора произведений $k_i\alpha_i$ формируется сигнал, эквивалентный 0.8039196. Поскольку $0.8039196 > 0.0045249$, то на выходе блока сравнения будет значение 0, и в результате на выходах скорректированного остатка $\alpha_1, \ldots, \alpha_7$ будут значения 0, при этом сигнал 0 с выхода блока сравнения поступает на вход $n + 3$-битного циклического регистра сдвига, где происходит сдвиг значения и формирования первой проекции в результате которой выходы элемента AND формирования проекций α_1 и элемента AND локализации ошибки по модулю p_1 будут нулевыми, а значение счетчика изменится на 1, что соответствует адресу 1 таблицы 10.

Аналогично происходят вычисления сумм и на выходе сумматора произведений $k_i\alpha_i$ формируется сигнал, эквивалентный 0.6078392, на выходе сумматора произведений $k_6\pi_6$ и $k_7\pi_7$ формируется сигнал, эквивалентный 0.0090497 и поскольку $0.6078392 > 0.0090497$, то на выходе блока сравнения будет значение 0, что соответствует наличию ошибки и в $n + 3$-битном циклическом регистре сдвига происходит сдвиг и формирование второй проекции.

В результате вычислений по второй проекции на выходе сумматора произведений $k_i\alpha_i$ формируется сигнал, эквивалентный 0.4117588, на выходе сумматора произведений $k_6\pi_6$ и $k_7\pi_7$ формируется сигнал, эквивалентный 0.0135747 и поскольку $0.4117588 > 0.0135747$, то на выходе блока сравнения будет значение 0, что соответствует наличию ошибки и в $n + 3$-битном циклическом регистре сдвига происходит сдвиг и формирование третьей проекции.

В результате вычислений по третьей проекции на выходе сумматора произведений $k_i\alpha_i$ формируется сигнал, эквивалентный 0.0195980, на выходе сумматора произведений $k_6\pi_6$ и $k_7\pi_7$ формируется сигнал, эквивалентный 0.0226244 и поскольку $0.0195980 < 0.0226244$, то на выходе блока сравнения будет значение 1, что соответствует отсутствию ошибки и следующий сдвиг в $n+3$-битном циклическом регистре сдвига не происходит. Следовательно, ошибка произошла по третьему основанию. Значение 0.0195980 с выхода сумматора произведений $k_i\alpha_i$ поступает на модулярный умножитель на $\overline{P}_3$ по модулю p_3, на выходе кото-

рого формируется значение 1. Поскольку значение, подаваемое на управляющий вход мультиплексора выбора корректного остатка α_3, равно нулю, то на выход подается значение 1 с выхода модулярного умножителя на $\overline{P}_3$ по модулю p_3. На остальные управляющие входы мультиплексоров выбора корректного остатка α_i подаются значения 1 и на выход подаются значения с соответствующих элементов AND локализации ошибки по модулю p_i, которые равны полученным α_i, которые хранятся в регистрах хранения остатков α_i. Таким образом на выходы скорректированного остатка $\alpha_1, \ldots, \alpha_7$ поступают значения $1, 0, 1, 6, 10, 12, 12$, что соответствует исправленному значению.

Данный вычислительный узел позволяет исправить ошибки модулярного кода, возникшие как в случае ошибок в вычислительных каналах, так и при передаче данных по каналам связи.

Моделирование предложенных алгоритмов было реализовано на языке Verilog на ASIC в среде RTL и физического синтеза Cadence Genus Synthesis Solution с использованием библиотеки osu018_stdcells.

Все значения, которые могли быть предвычислены, записаны в память. Для моделирования выбраны наборы СОК с одним и двумя избыточными модулями с 4-6 рабочими модулями, покрывающие диапазоны 8, 16, 24 и 32 бита, представленные в таблицах 8–9.

В таблицах 11–12 представлены результаты моделирования времени прохождения сигнала по схеме (пикосекунды, пс) и используемой площади (квадратные микрометры, мкм^2) соответственно. Для моделирования выбраны Алгоритм 1.1 с двумя избыточными основаниями, обозначенный как `I`, Алгоритм 2.33 с одним избыточным основанием, обозначенный в таблицах как `II`, приближенный метод на основе КТО с двумя избыточными основаниями, описанный в патенте [116], обозначенный как `III`, приближенный метод на основе КТО с одним избыточным основанием, описанный Алгоритмом 2.34 [42], обозначенный как `IV`.

Метод на основе Китайской теоремы об остатках, представленный Алгоритмом 1.1 в среднем имеет на 52,15% меньшее время вычисления по сравнению с Алгоритмом 2.33 с одним избыточным основанием, но имеет на 140% большую площадь.

Метод на основе Китайской теоремы об остатках, представленный Алгоритмом 1.1, в среднем имеет на 27,55% большее время вычислений, по сравне-

Таблица 11 — Результаты моделирования времени прохождения сигнала по схеме, пс

Кол-во раб. модулей	Алго-ритм	Покрываемый рабочий диапазон, бит			
		8	16	24	32
4	I	15447	22338	33628	43106
	II	25097	39429	84673	119623
	III	11369	15876	21509	31825
	IV	11476	15707	24436	29211
5	I	13701	18392	29237	38483
	II	22903	36698	76512	111360
	III	12424	15646	23134	27172
	IV	12374	16354	22028	27055
6	I	16409	20503	26721	34146
	II	29077	41193	66569	66711
	III	15867	17340	22252	27402
	IV	15542	18717	21620	26845

Таблица 12 — Результаты моделирования используемой площади, мкм^2

Кол-во раб. модулей	Алго-ритм	Покрываемый рабочий диапазон, бит			
		8	16	24	32
4	I	156978	320430	672019	1093897
	II	60759	132332	319916	538922
	III	186419	393166	816822	1470043
	IV	148011	314708	684626	1206436
5	I	164846	306875	646014	1091506
	II	62136	131767	287343	493900
	III	258575	422888	877680	1448269
	IV	200183	347296	715083	1329741
6	I	280610	356641	657436	1080177
	II	94223	155115	282315	399128
	III	444720	566191	916218	1393492
	IV	343388	473257	783562	1281555

нию с приближенным методом на основе КТО с двумя избыточными основаниями, описанным в патенте [116], но имеет на 26,41% меньшую площадь.

Адаптация приближенного метода с одним избыточным основанием, описанная Алгоритмом 2.34 имеет в среднем на 0,43% большее время вычислений по сравнению с приближенным методом на основе КТО с двумя избыточными основаниями, описанным в патенте [116], но на 16,96% меньшую площадь.

Таким образом, применение приближенного метода с одним избыточным основанием позволяет при одинаковом времени вычислений получить снижение требуемой площади.

3.7 Выводы по третьей главе

Система остаточных классов, как непозиционная система счисления, позволяет получить преимущество за счет независимости выполнения арифметических действий по модулям, при этом высокую эффективность показывают модули специального вида $\{2^n - 1, 2^n, 2^n + 1\}$ рассмотренные в параграфе 2.1. Преимущества параллельности и малого размера модулей СОК слабо раскрываются при реализации алгоритмов средствами стандартных ЭВМ, вычисления в которых привязаны к разрядности процессора. В связи с этим возникает потребность в разработке специализированных вычислительных узлов распределенной среды для выполнения арифметических, в том числе немодульных, операций в СОК.

В данной главе рассмотрено 6 архитектур вычислительных узлов для работы с системой остаточных классов, на которые получены патенты [102;116–120].

В параграфе 3.1 рассмотрен алгоритм проектирования отказоустойчивой вычислительной системы, работающей в модулярном коде. Построение такой системы включает выбор набора модулей СОК, выбор метода перевода из ПСС в СОК, методов выполнения сложения, вычитания и умножения в модулярном коде, выбор метода вычисления позиционной характеристики для задач сравнения чисел, определения знака числа, перевода из СОК в ПСС. Для исследования разработанных методов и алгоритмов и генерации Verilog-модулей разработан программный комплекс для построения вычислительных систем, рабо-

тающих в системе остаточных классов, на который получено 14 программ для ЭВМ [124–137].

В параграфе 3.2 рассмотрено моделирование перевода числа из позиционной системы счисления в систему остаточных классов. Классические методы на основе нейронной сети конечного кольца [93], периода и полупериода [65] дают число в диапазоне удвоенного модуля. Предложенная модификация позволила получить искомые значения, при этом модифицированный метод на основе периода и полупериода числа позволили в среднем на 50% сократить необходимую площадь, на 23% – время вычислений, по сравнению с нейронной сетью конечного кольца.

В параграфе 3.3 рассмотрено обратное преобразование из системы остаточных классов в позиционную систему счисления. Предложена архитектура, на которую получен патент [117], основанная на модифицированной обобщенной позиционной системе счисления. В среднем предложенный подход на 23% быстрее и на 30% компактнее, чем метод КТО. При этом на диапазоне 32-48 бит предложенный подход быстрее на 46% и компактнее на 50% по сравнению с КТО. При сравнении с приближенным методом на основе КТО предложенный метод дает преимущество в среднем на 18% по времени и на 23% по площади только на диапазоне 32-48 бит. В среднем преимущество по сравнению с ОПСС по времени составило 15%, но метод на основе ОПСС занимает на 16% меньшую площадь.

Таким образом, предложенный метод на основе модифицированной ОПСС позволяет повысить скорость вычислений и снизить площадь, являясь компромиссным решением между последовательной и компактной ОПСС, и параллельной, но громоздкой КТО.

В параграфе 3.4 рассмотрены методы вычисления позиционной характеристики для сравнения чисел и определения знака числа в системе остаточных классов. Предложен ряд архитектур вычислительных узлов сравнения и определения знака, на которые получены патенты на изобретения [102; 118; 119]. Патент [102] основан на алгоритме 2.20 построения функции ядра Акушского с заданными свойствами, сравнение чисел в которой происходит по алгоритму 2.21. Архитектура вычислительного узла на основе функции ядра быстрее реализации на основе Китайской теоремы об остатках в среднем на 59%, на 15% быстрее приближенного метода на основе КТО, и на 22% быстрее ОПСС. При

этом требуемая площадь на 61% меньше, чем у КТО, на 16% – чем у приближенного метода на основе КТО, но в среднем на 80% больше, чем у ОПСС.

Также уточнена точность приближенного метода на основе КТО для сравнения чисел. Архитектура вычислительного узла, использующего уточненный приближенный метод на основе КТО в среднем на 58 % быстрее КТО, на 10% быстрее оценки приближенного метода на основе КТО из [75], на 14% быстрее ОПСС. При этом уточненный приближенный метод на основе КТО требует в среднем на 59% меньшую площадь по сравнению с КТО, на 9% меньше оценки приближенного метода на основе КТО из [75], но на 100% больше чем у ОПСС.

Таким образом, разработанные методы показали лучшее время работы, однако по используемой площади уступают обобщенной позиционной системе счисления.

В параграфе 3.5 приведено моделирование умножения с накоплением для двоичной системы счисления и системы остаточных классов, в которых умножение реализовано стандартными методами Verilog. Реализация в СОК в среднем на 30% быстрее при приблизительно одинаковых занимаемых площадях.

В параграфе 3.6 предложена архитектура вычислительного узла для обнаружения и коррекции ошибки модулярного кода, основанная на приближенном методе на основе Китайской теореме об остатках, на которую получен патент [116]. Особенностью данного подхода является использование памяти для восстановления чисел по каждой проекции. Также рассмотрено моделирование алгоритма коррекции одиночной ошибки с одним избыточным модулем. Метод на основе Китайской теоремы об остатках, представленный Алгоритмом 1.1 в среднем имеет на 52,15% меньшее время вычисления по сравнению с Алгоритмом 2.33 с одним избыточным основанием, но имеет на 140% большую площадь. Адаптация приближенного метода с одним избыточным основанием, описанная Алгоритмом 2.34 имеет в среднем на 0,43% большее время вычислений по сравнению с приближенным методом на основе КТО с двумя избыточными основаниями, описанным в патенте [116], но на 16,96% меньшую площадь.

Таким образом, были разработаны архитектуры вычислительных узлов выполнения немодульных операций, таких как перевод в позиционную систему счисления и расширение оснований, сравнение чисел, определение знака числа, модулярное умножение по произвольному модулю, исправление ошибок модулярного кода. Данные вычислительные узлы могут быть использованы при

реализации модулей вычислительных систем обработки данных, работающих в системе остаточных классов.

Заключение

Проведенное исследование показывает, что алгоритмы, используемые в современных системах обработки информации, обладают большой вычислительной сложность, что создает ряд проблем при их реализации вычислительными системами. Для уменьшения вычислительной сложности алгоритмов обработки информации предложено использовать систему остаточных классов, которая позволяет выполнять параллельно алгоритмы вычисления арифметических операций сложения и умножения чисел, но при этом возникает задача уменьшения вычислительной сложности выполнения следующих операций: перевода из ПСС в СОК, перевода из СОК в ПСС, определения знака числа и сравнения чисел, обнаружения и исправления ошибок и др.

Рассмотрена проблема подбора модулей СОК для перевода чисел из позиционной системы счисления, рассмотрены алгоритмы нахождения остатка при делении на модули специального вида 2^n, 2^n-1, 2^n+1: нейронная сеть конечного кольца, алгоритмы на основе вычисления периода и полупериода числа. Проблемой большинства этих алгоритмов является получение не искомого результата, а сравнимого с ним. Предложена их модификация, позволяющая найти искомый остаток для чисел, не превосходящих двухкратного размера модуля, что соответствует выходу методов нейронной сети конечного кольца и периода/полупериода. Моделирование перевода числа из позиционной системы счисления в систему остаточных классов проведено на ASIC в среде RTL и физического синтеза Cadence Genus Synthesis Solution с использованием библиотеки osu018_stdcells. Предложенная модификация позволила получить искомые значения, при этом модифицированный метод на основе периода и полупериода числа позволили в среднем на 50% сократить необходимую площадь, на 23% – время вычислений, по сравнению с нейронной сетью конечного кольца.

Исследованы методы перевода из СОК в позиционную систему счисления на основе Китайской теоремы об остатках, приближенного метода на основе КТО, обобщенной позиционной системе счисления и их модификации, на основе функции ядра и диагональной функции, методе опорных точек и с использований модулей специального вида. Рассмотрены особенности этих методов, найдены их недостатки. Предложена архитектура вычислительного вычислительного, на которую получен патент [117], основанная на модифицированной обобщен-

ной позиционной системе счисления. В среднем предложенный подход на 23% быстрее и на 30% компактнее, чем метод КТО. При этом на диапазоне 32-48 бит предложенный подход быстрее на 46% и компактнее на 50% по сравнению с КТО. При сравнении с приближенным методом на основе КТО предложенный метод дает преимущество в среднем на 18% по времени и на 23% по площади только на диапазоне 32-48 бит. В среднем преимущество по сравнению с ОПСС по времени составило 15%, но метод на основе ОПСС занимает на 16% меньшую площадь.

Таким образом, предложенный метод на основе модифицированной ОПСС позволяет повысить скорость вычислений и снизить площадь, являясь компромиссным решением между последовательной и компактной ОПСС, и параллельной, но громоздкой КТО.

Рассмотрена проблема сравнения чисел в СОК и определения знака числа. Реализация алгоритма сравнения чисел в СОК состоит из двух этапов. Первый этап — вычисление позиционной характеристики (ПХ) модулярных чисел $X = (x_1, \ldots, x_n)$ и $Y = (y_1, \ldots, y_n)$. Второй этап — сравнение позиционных характеристик ПХ(X) и ПХ(Y) модулярных чисел в позиционной системе счисления. Уточнена точность строго возрастающей приближенной функции на основе КТО для сравнения чисел, предложен метод для сравнения чисел, который позволяет уменьшить вычислительную сложность алгоритма сравнения чисел в СОК. Предложенный метод позволяет уменьшить размер операндов по сравнению с алгоритмами из работ [75; 86]. Также введена модификация функции ядра Акушского с заданными свойствами, разработаны алгоритмы подбора параметров для функции ядра без критических ядер, на основе которой построен алгоритм сравнения чисел и определения знака числа. Предложены методы сравнения чисел для СОК как с четным динамическим диапазоном, так и с нечетным. Предложенные методы позволяют получить искомый результат без операции деления на большой модуль, а также являются точными, в отличие от приближенного метода на основе КТО, в котором возможно накопление ошибок за счет погрешности округления.

Предложен ряд архитектур вычислительных узлов сравнения и определения знака, на которые получены патенты на изобретения [102; 118; 119]. Патент [102] основан на алгоритме 2.20 построения функции ядра Акушского с заданными свойствами, сравнение чисел в которой происходит по алгоритму 2.21. Архитектура вычислительного узла на основе функции ядра быстрее реа-

лизации на основе Китайской теоремы об остатках в среднем на 59%, на 15% быстрее приближенного метода на основе КТО, и на 22% быстрее ОПСС. При этом требуемая площадь на 61% меньше, чем у КТО, на 16% – чем у приближенного метода на основе КТО, но в среднем на 80% больше, чем у ОПСС.

Также уточнена точность приближенного метода на основе КТО. Уточненный приближенный метод на основе КТО в среднем на 58 % быстрее КТО, на 10% быстрее оценки приближенного метода на основе КТО из [75], на 14% быстрее ОПСС. При этом уточненный приближенный метод на основе КТО требует в среднем на 59% меньшую площадь по сравнению с КТО, на 9% меньше оценки приближенного метода на основе КТО из [75], но на 100% больше чем у ОПСС.

Таким образом, разработанные методы показали лучшее время работы, однако по используемой площади уступают обобщенной позиционной системе счисления.

Исследованы методы модульного умножения: нейронная сеть конечного кольца, метод Карацубы-Офмана, метод Бута, алгоритмы Монтгомери. Рассмотрена реализация умножения с накоплением (MAC) в системе остаточных классов. Приведено моделирование умножения с накоплением для двоичной системы счисления и системы остаточных классов, в которых умножение реализовано стандартными методами Verilog. Реализация в СОК в среднем на 30% быстрее при приблизительно одинаковых занимаемых площадях.

Предложена архитектура вычислительного узла обнаружения, локализации и исправления ошибок в СОК на базе приближенного метода, на которую получен патент [116], которая позволяет на основе метода проекций исправлять ошибки передачи данных в беспроводных каналах связи. Особенностью данного подхода является использование памяти для восстановления чисел по каждой проекции. Введен метод коррекции одиночной ошибки с одним избыточным модулем СОК. Адаптация приближенного метода с одним избыточным основанием имеет в среднем на 0,43% большее время вычислений по сравнению с приближенным методом на основе КТО с двумя избыточными основаниями, описанным в патенте [116], но на 16,96% меньшую площадь.

Таким образом, были разработаны архитектуры вычислительных узлов выполнения немодульных операций, таких как перевод в позиционную систему счисления и расширение оснований, сравнение чисел, определение знака числа, исправление ошибок модулярного кода. Данные вычислительные узлы могут

быть использованы при реализации вычислительных систем обработки информации, работающих в системе остаточных классов.

Полученные в диссертационном исследовании результаты позволяют сделать следующие выводы:

1. Предложенный метод для перевода из СОК в ПСС на основе модифицированной обобщенной позиционной системы счисления за счет сокращения количества операций позволяет сократить время вычислений по сравнению с обобщенной позиционной системой счисления и за счет уменьшения размеров операндов сократить используемую площадь по сравнению с приближенным методом на основе Китайской теоремой об остатках.
2. Модифицированный приближенный метод определения знака и сравнения чисел в СОК на основе КТО позволяет повысить скорость вычислений по сравнению с приближенным методом на основе Китайской теоремы об остатках за счет уточнения необходимой точности вычислений коэффициентов приближенного метода и замены операции нахождения остатка от деления взятием младших бит числа.
3. Алгоритм сравнения чисел и определения знака числа на основе функции ядра Акушского без критических ядер позволяет за счет модуля специального вида повысить эффективность и точность целочисленных вычислений.
4. Использование надежного специального избыточного модуля в избыточной СОК с одним контрольным модулем позволяет не только обнаружить ошибку по рабочему модулю, но и локализовать её, при этом снижая требуемую площадь при сохранении скорости вычислений.
5. Комплекс программ и архитектур вычислительных узлов распределенной среды, реализующих немодульные операции в СОК, используемых при решении задач обработки информации на основе системы остаточных классов, позволяет повысить скорость и надежность вычислений.

Список литературы

1. **A high-speed residue-to-binary converter based on approximate Chinese Remainder Theorem** / N.N. Kucherov, V.A. Kuchukov, N.N. Kuchukova, A.E. Shangina //2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – IEEE, 2018. – С. 325-328.

2. **A high-speed residue-to-binary converter for three-moduli** $(2^k, 2^k - 1, 2^{k-1} - 1)$ **RNS and a scheme for its VLSI implementation** / W. Wang, M.N.S. Swamy, M.O. Ahmad, Y. Wang //IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. – 2000. – Т. 47, №. 12. – С. 1576-1581.

3. **A method of increasing digital filter performance based on truncated multiply-accumulate units** / P. Lyakhov, M. Valueva, G. Valuev, N. Nagornov //Applied Sciences. — 2020. — Vol. 10. — №. 24. — p. 9052.

4. **A new single-error correction scheme based on self-diagnosis residue number arithmetic** / Y. Tang, E. Boutillon, C. Jego, M. Jezequel //2010 Conference on Design and Architectures for Signal and Image Processing (DASIP). – IEEE, 2010. – С. 27-33.

5. **AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage** / A. Tchernykh, M. Babenko, N. Chervyakov [et al.] //International Journal of Approximate Reasoning. – 2018. – Т. 102. – С. 60-73.

6. **AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security** / N. Chervyakov, M. Babenko, A. Tchernykh [et al.]//Future Generation Computer Systems. – 2019. – Т. 92. – С. 1080-1092.

7. **An Efficient Method for Comparing Numbers and Determining the Sign of a Number in RNS for Even Ranges** /A. Tchernykh, M. Babenko, E. Shiriaev [et al.] // Computation. – 2022. – Т. 10. – №. 2. – С. 17.

8. **An approximate method for comparing modular numbers and its application to the division of numbers in residue number systems** /N.I. Chervyakov, M.G. Babenko, P.A. Lyakhov, I.N. Lavrinenko //Cybernetics and Systems Analysis. – 2014. – Т. 50, №. 6. – С. 977-984.

9. **Andraos, S.** A new efficient memoryless residue to binary converter / S. Andraos, H. Ahmad //IEEE Transactions on circuits and systems. — 1988. — Т. 35, №. 11. — С. 1441–1444.

10. **Babenko, M.** Improved modular division implementation with the Akushsky core function / M. Babenko, A. Tchernykh, V. Kuchukov // Computation. — 2022. – Т. 10. – № 1. – С. 9.

11. **Barrett, P.** Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor / P. Barrett // Conference on the Theory and Application of Cryptographic Techniques. — Springer, Berlin, Heidelberg, 1986. — С. 311–323.

12. **Bi, S.** The mixed-radix Chinese remainder theorem and its applications to residue comparison /S. Bi, W.J. Gross //IEEE Transactions on Computers. – 2008. – Т. 57, №. 12. – С. 1624-1632.

13. **Booth, A.D.** A signed binary multiplication technique / A.D. Booth // The Quarterly Journal of Mechanics and Applied Mathematics. — 1951. — Т. 4, вып. 2. — с. 236–240.

14. **Brickell, E.F.** A survey of hardware implementations of RSA / E.F. Brickell //Conference on the Theory and Application of Cryptology. – Springer, New York, 1989. — С. 368-370.

15. **Brown, D.T.** Error detecting and correcting binary codes for arithmetic operations / D.T. Brown //IRE Transactions on Electronic Computers. – 1960. – № 3. – С. 333-337.

16. **Burgess, N.** Scaling an RNS number using the core function / N. Burgess //Proceedings 2003 16th IEEE Symposium on Computer Arithmetic. – IEEE, 2003. – С. 262-269.

17. **Chaves, R.** $\{2^n+1, 2^{n+k}, 2^n-1\}$: a new RNS moduli set extension / R. Chaves, L. Sousa // Digital System Design, 2004. DSD 2004. Euromicro Symposium on. — IEEE, 2004. — С. 210-217.

18. **Chervyakov, N.I.** Digital filtering of images in a residue number system using finite-field wavelets / N.I. Chervyakov, P.A. Lyakhov, M.G. Babenko //Automatic Control and Computer Sciences. – 2014. – Т. 48, №. 3. – С. 180-189.

19. **Chervyakov, N.I.** Research of effective methods of conversion from positional notation to RNS on FPGA / N.I. Chervyakov, M.G. Babenko, V.A. Kuchukov //2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – IEEE, 2017. – С. 277-281.

20. **Data Reliability and Redundancy Optimization of a Secure Multi-Cloud Storage Under Uncertainty of Errors and Falsifications** / A. Tchernykh, M. Babenko, V. Kuchukov [et al.] //2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). – IEEE, 2019. – С. 565-572.

21. **Diamond, J. M.** Checking codes for digital computers / J.M. Diamond //PROCEEDINGS OF THE INSTITUTE OF RADIO ENGINEERS. – 1955. – Т. 43. – №. 4. – С. 487-488.

22. **Dimauro, G.** A new technique for fast number comparison in the residue number system / G. Dimauro, S. Impedovo, G. Pirlo //IEEE transactions on computers. — 1993. — Т. 42, №. 5. — С. 608–612.

23. **Efficient implementation of error correction codes in modular code** / N.N. Kucherov, V.A. Kuchukov, E. Golimblevskaia [et al.] //ICCS-DE. – 2021. – pp. 107-118.

24. **Experimental Analysis of Secret Sharing Schemes for Cloud Storage Based on RNS** /V. Miranda-López, A. Tchernykh, J.M. Cortés-Mendoza [et al.] // Latin American High Performance Computing Conference. — 2017. — С. 370-383.

25. **Fast modular multiplication execution in residue number system** /N.I. Chervyakov, M.G. Babenko, V.A. Kuchukov [et al.] //2016 IEEE Conference on Quality Management, Transport and Information Security, Information Technologies (IT&MQ&IS). – IEEE, 2016. – С. 30-32.

26. **Goh, V. T.** A novel error correction scheme based on the Chinese remainder theorem /V.T. Goh, M. Tinauli, M.U. Siddiqi //The Ninth International Conference onCommunications Systems, 2004. ICCS 2004. – IEEE, 2004. – С. 461-465.

27. **Goh, V. T.** Multiple error detection and correction based on redundant residue number systems /V.T. Goh, M.U. Siddiqi //IEEE Transactions on Communications. – 2008. – Т. 56. – №. 3. – С. 325-330.

28. **Goldreich, O.** Chinese remaindering with errors /O. Goldreich, D. Ron, M. Sudan //Proceedings of the thirty-first annual ACM symposium on Theory of computing. – 1999. – С. 225-234.

29. **Gomathisankaran, M.** HORNS: A homomorphic encryption scheme for Cloud Computing using Residue Number System / M. Gomathisankaran, A. Tyagi, K. Namuduri //2011 45th Annual Conference on Information Sciences and Systems. – IEEE, 2011. – С. 1-5.

30. **Hanzo, L.** Turbo coding, turbo equalisation and space-time coding / L. Hanzo, T.H. Liew, B.L. Yeap. – John Wiley & Sons, 2002.

31. **Haron, N. Z.** Redundant residue number system code for fault-tolerant hybrid memories / N. Z. Haron, S. Hamdioui //ACM journal on emerging technologies in computing systems (JETC). – 2011. – Т. 7. – №. 1. – С. 1-19.

32. **Hosseinzadeh, M.** An improved reverse converter for the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ / M. Hosseinzadeh, A.S. Molahosseini, K. Navi //IEICE Electronics Express. — 2008. — Т. 5, №. 17. — С. 672-677.

33. **Huang, C.H.** A fully parallel mixed-radix conversion algorithm for residue number applications / C.H. Huang // IEEE Transactions on computers. — 1983. — №. 4. — С. 398–402.

34. **Increasing reliability and fault tolerance of a secure distributed cloud storage** / N.N. Kucherov, M.G. Babenko, A. Tchernykh [et al.] //ICCS-DE. – 2020. – С. 166-180.

35. **Isupov, K.** An Algorithm for Magnitude Comparison in RNS based on Mixed-Radix Conversion II/ K. Isupov //International Journal of Computer Applications. — 2016. — Т. 975. — С. 8887.

36. **Kang, J**. PV-MAC: Multiply-and-accumulate unit structure exploiting precision variability in on-device convolutional neural networks/ J. Kang, T. Kim // Integration. — 2020. — Vol. 71. – pp. 76-85.

37. **Keller, T.** Adaptive redundant residue number system coded multicarrier modulation / T. Keller, T. H. Liew, L. Hanzo //IEEE Journal on Selected Areas in Communications. – 2000. – Т. 18. – №. 11. – С. 2292-2301.

38. **Knuth, D.E.** The art of computer programming: seminumerical algorithms, vol 2, 2nd Edition / D.E. Knuth // Addison-Wesley, Reading, Mass. — 1981. — 782 c.

39. **Kogge, P. M.** A parallel algorithm for the efficient solution of a general class of recurrence equations / P.M. Kogge, H.S. Stone //IEEE transactions on computers. – 1973. – Т. 100. – №. 8. – С. 786-793.

40. **Krishna, H.** A coding theory approach to error control in redundant residue number systems. I. Theory and single error correction /H. Krishna, K.Y. Lin, J.D. Sun //IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. – 1992. – Т. 39. – №. 1. – С. 8-17.

41. **Kuchukov, V.** Cloud-fog-edge Computing Model for Video Surveillance Based on Modular Arithmetic / V. Kuchukov, A. Nazarov, I. Vashchenko //2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – IEEE, 2020. – С. 374-376.

42. **Kuchukov, V.** Study of a Redundant Residue Number System for Single Error Correction / V. Kuchukov, M. Babenko, S. Al-Galda //Advances in Systems Science and Applications. – 2023. – Т. 23. – №. 04. – С. 31-39.

43. **Kuchukov, V.** The application of modular arithmetic for matrix calculations/ V. Kuchukov, M. Babenko //2019 Ivannikov Ispras Open Conference (ISPRAS). – IEEE, 2019. – С. 49-54.

44. **Liew, T. H.** Systematic redundant residue number system codes: Analytical upper bound and iterative decoding performance over AWGN and Rayleigh channels /T.H. Liew, L.L. Yang, L. Hanzo //IEEE transactions on communications. – 2006. – Т. 54. – №. 6. – С. 1006-1016.

45. **Madhavi Latha, M. V. N.** An improved RNS-to-binary converter for 7-modulus set $\{2^{n-5}-1, 2^{n-3}-1, 2^{n-2}+1, 2^{n-1}-1, 2^{n-1}+1, 2^n, 2^n+1\}$ for n even /M.V.N. Madhavi Latha, R.R. Rachh, P.V. Ananda Mohan //Sadhana. – 2020. – Т. 45. – №. 1. – С. 1-4.

46. **Modified Error Detection and Localization in the Residue Number System** / A. Gladkov, V. Kuchukov, M. Babenko [et al.] // Programming and Computer Software. — 2022. — Т. 48. — №. 8. — pp. 598–605.

47. **Mohan, P.V.A.** RNS to Binary Conversion / P.V.A. Mohan // Residue Number Systems. — Springer, 2016. – С. 81-132.

48. **Mohan, P.V.A.** RNS to binary conversion using diagonal function and Pirlo and impedovo monotonic function/ P.V.A. Mohan //Circuits, Systems, and Signal Processing. — 2016. — Т. 35. — №. 3. — С. 1063–1076.

49. **Mohan, P.V.A.** Reverse Converters for the Moduli Set $\{2^n, 2^{n-1}-1, 2^n-1, 2^{n+1}-1\}(n$ Even)/ P.V. Ananda Mohan //Circuits, Systems, and Signal Processing. – 2018. – Т. 37. – №. 8. – С. 3605-3634.

50. **Mojahed, M.** Multifunctional unit for reverse conversion and sign detection based on five-moduli set $\{2^{2n}, 2^n+1, 2^n-1, 2^n+3, 2^n-3\}$ /M. Mojahed, A.S. Molahosseini, A.A.E. Zarandi //Computer Science. – 2021. – Т. 22. – С. 101-121.

51. **Molahosseini, A.S.** Efficient Reverse Converter Designs for the New 4-Moduli Sets $\left\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\right\}$ and $\left\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\right\}$ Based on New CRTs / A.S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, S. Timarchi //IEEE Transactions on Circuits and Systems I: Regular Papers. — 2009. — Т. 57. — №. 4. — С. 823–835.

52. **Montgomery, P.L.** Modular multiplication without trial division / P.L. Montgomery // Mathematics of computation. — 1985. — Т. 44, №. 170. — с. 519–521.

53. **Multiply Adder v3.0. LogiCORE IP Product Guide. Vivado Design Suite. Xilinx**: [сайт]. — 2021. — URL: https://www.xilinx.com/content/dam/xilinx/support/documents/ip_documentation/xbip_multadd/v3_0/pg192-multadd.pdf (дата обращения: 03.03.2023).

54. **Nedjah, N.** A review of modular multiplication methods ands respective hardware implementation / N. Nedjah, L. de Macedo Mourelle //Informatica. — 2006. — Т. 30, №. 1. — с. 111-129.

55. **Omondi, A.R.** Residue number systems: theory and implementation / A.R. Omondi, B. Premkumar. – World Scientific, 2007. – Т. 2. — 312 с.

56. **Parhami, B.** Computer arithmetic/ B. Parhami. – New York, NY : Oxford university press, 2010. – Т. 20. — ISBN 978-0-195-32848-6.

57. **Patil, P. A.** Multiply accumulate unit using radix-4 booth encoding /P.A. Patil, C. Kulkarni //2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS). – IEEE, 2018. – С. 1076-1080.

58. **Patronik, P.** Design of RNS reverse converters with constant shifting to residue datapath channels / P. Patronik, S.J. Piestrak //Journal of Signal Processing Systems. – 2018. – Т. 90. – №. 3. – С. 323-339.

59. **Patronik, P.** Design of Reverse Converters for General RNS Moduli Sets $\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$ and $\{2^k, 2^n-1, 2^n+1, 2^{n-1}-1\}(n$ even)/ P. Patronik, S.J. Piestrak //IEEE Transactions on Circuits and Systems I: Regular Papers. — 2014. — Т. 61, №. 6. — С. 1687-1700.

60. **Patronik, P.** On reverse converters for arbitrary multi-moduli RNS / P. Patronik //Integration. – 2020. – Т. 75. – С. 158-167.

61. **Performance Analysis of Hardware Implementations of Reverse Conversion from the Residue Number System** / V. Kuchukov, D. Telpukhov, M. Babenko [et al.] //Applied Sciences. – 2022. – Т. 12. – №. 23. – С. 12355.

62. **Phalguna, P. S.** RNS-to-Binary Converters for New Three-Moduli Sets $\{2^k - 3, 2^k - 2, 2^k - 1\}$ and $\{2^k + 1, 2^k + 2, 2^k + 3\}$ /P.S. Phalguna, D.V. Kamat, P.V. Ananda Mohan //Journal of Circuits, Systems and Computers. – 2018. – Т. 27. – №. 14. – С. 1850224.

63. **Phatak, D.S.** New distributed algorithms for fast sign detection in residue number systems (RNS)/ D.S. Phatak, S.D. Houston //Journal of Parallel and Distributed Computing. — 2016. — Т. 97. — С. 78-95.

64. **Piestrak, S.J.** A note on RNS architectures for the implementation of the diagonal function / S.J. Piestrak // Information Processing Letters. — 2015. — Т. 115, №. 4. — С. 453–457.

65. **Piestrak, S.J.** Design of multi-residue generators using shared logic / S.J. Piestrak // Circuits and Systems (ISCAS), 2011 IEEE International Symposium on. — IEEE, 2011. — С. 1435-1438.

66. **Pirlo, G.** A new class of monotone functions of the residue number system /G. Pirlo, D. Impedovo // International Journal of Mathematical Models and Methods in Applied Sciences. – 2013. – Т. 7, №. 9. – С. 803-809.

67. **Plantard, T.** Arithmétique modulaire pour la cryptographie / T.Plantard // : дис. – Université Montpellier II-Sciences et Techniques du Languedoc, 2005. — 133 с.

68. **Positional Characteristics for Efficient Number Comparison over the Homomorphic Encryption** / M. Babenko, A. Tchernykh, N. Chervyakov [et al.] //Programming and Computer Software. – 2019. – Т. 45, №. 8. – С. 532-543.

69. **RESIDENT: a reliable residue number system-based data transmission mechanism for wireless sensor networks** /R. Ye, A. Boukerche, H. Wang [et al.] // Wireless Networks. – 2018. – Т. 24, №. 2. – С. 597-610.

70. **Rabin, M. O.** Efficient dispersal of information for security, load balancing, and fault tolerance / M.O. Rabin //Journal of the ACM (JACM). – 1989. – Т. 36. – №. 2. – С. 335-348.

71. **Rakesh, H. M.** Design and implementation of Novel 32-bit MAC unit for DSP applications /H.M. Rakesh, G.S. Sunitha //2020 International Conference for Emerging Technology (INCET). – IEEE, 2020. – С. 1-6.

72. **Reference Points Based RNS Reverse Conversion for General Moduli Sets** / A. Stempkovsky, D. Telpukhov, I. Mkrtchan, A. Zhigulin //International Conference on Mathematics and its Applications in new Computer Systems. – Springer, Cham, 2022. – С. 253-262.

73. **Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications** /C.H. Chang , A.S. Molahosseini, A.A.E. Zarandi, T.F. Tay //IEEE circuits and systems magazine. – 2015. – Т. 15, №. 4. – С. 26-44.

74. **Residue-to-binary conversion by the «quotient function»** / G. Dimauro, S. Impedovo, R. Modugno [et al.] //IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. – 2003. – Т. 50, №. 8. – С. 488-493.

75. **Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem** / N. I. Chervyakov, A.S. Molahosseini, P.A. Lyakhov [et al.] // International journal of computer mathematics. – 2017. – Т. 94, №. 9. – С. 1833-1849.

76. **Sengupta, A.** Performance of systematic RRNS based space-time block codes with probability-aware adaptive demapping / A. Sengupta, B. Natarajan //IEEE transactions on wireless communications. – 2013. – Т. 12. – №. 5. – С. 2458-2469.

77. **Shindler, V.** High-speed RSA hardware based on low-power piplined logic : Ph. D. Thesis / V. Shindler ; Institut für Angewandte Informations-verarbeitung und Kommunikationstechnologie, Technishe Universität Graz. — 1997. — 102 с.

78. **Soderstrand, M.** An improved residue number system digital-to-analog converter / M. Soderstrand, C. Vernia, J.H. Chang // IEEE transactions on circuits and systems. — 1983. — Т. 30, №. 12. — С. 903–907.

79. **Sousa, L.** Combining residue arithmetic to design efficient cryptographic circuits and systems /L. Sousa, S. Antao, P. Martins //IEEE Circuits and Systems Magazine. – 2016. – Т. 16, №. 4. – С. 6-32.

80. **Szabo, N.S.** Residue arithmetic and its applications to computer technology / N.S. Szabo, R.I. Tanaka. — McGraw-Hill, 1967. — 236 с.

81. **Tay, T. F.** A new algorithm for single residue digit error correction in Redundant Residue Number System / T.F. Tay, C.H. Chang //2014 IEEE International Symposium on Circuits and Systems (ISCAS). – IEEE, 2014. – С. 1748-1751.

82. **The use of modified correction code based on residue number system in WSN** / V. Yatskiv, N. Yatskiv, S. Jun [et al.] //2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS). – IEEE, 2013. – Т. 1. – С. 513-516.

83. **Towards mitigating uncertainty of data security breaches and collusion in cloud computing** / A. Tchernykh, M. Babenko, N. Chervyakov [et al.] // 2017 28th International Workshop on Database and Expert Systems Applications (DEXA). — 2017. — С. 137-141.

84. **Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability** /A. Tchernykh, U. Schwiegelsohn, E.G. Talbi, M. Babenko //Journal of Computational Science. – 2019. – Т. 36. – С. 100581.

85. **Unfairness correction in P2P grids based on residue number system of a special form** /M. Babenko, N. Chervyakov, A. Tchernykh [et al.] //2017 28th International Workshop on Database and Expert Systems Applications (DEXA). — 2017. — c. 147-151.

86. **Van Vu, T.** Efficient implementations of the Chinese remainder theorem for sign detection and residue decoding / T. Van Vu //IEEE Transactions on Computers. — 1985. — T. 100, №. 7. — c. 646-651.

87. **Walter, C. D.** A verification of Brickell's fast modular multiplication algorithm / C.D. Walter, S. E. Eldridge //International Journal of Computer Mathematics. — 1990. — T. 33, №. 3-4. — C. 153-169.

88. **Wang, Y.** Residue-to-binary converters based on new Chinese remainder theorems/Y. Wang //IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. – 2000. – T. 47, №. 3. – C. 197-205.

89. **Weighted two-levels secret sharing scheme for multi-clouds data storage with increased reliability** / V. Miranda-Lopez, A. Tchernykh, M. Babenko [et al.] //2019 International Conference on High Performance Computing and Simulation (HPCS). – IEEE, 2019. – C. 915-922.

90. **Yang, L. L.** A residue number system based parallel communication scheme using orthogonal signaling. I. System outline / L.L. Yang, L. Hanzo //IEEE transactions on vehicular technology. – 2002. – T. 51. – №. 6. – C. 1534-1546.

91. **Yau, S. S. S.** Error correction in redundant residue number systems / S.S.S. Yau, Y.C. Liu //IEEE Transactions on Computers. – 1973. – T. 100. – №. 1. – C. 5-11.

92. **Yin, P.** A new algorithm for single error correction in RRNS / P. Yin, L .Li //2013 International Conference on Communications, Circuits and Systems (ICCCAS). – IEEE, 2013. – T. 2. – C. 178-181.

93. **Zhang, D.** A neural-like network approach to finite ring computations / D. Zhang, G.A. Jullien, W.C. Miller //IEEE transactions on circuits and systems. — 1990. — T. 37, №. 8. — C. 1048–1052.

94. **Zhang, S.** Redundant residue number system assisted multicarrier direct-sequence code-division dynamic multiple access for cognitive radios / S. Zhang,

L.L. Yang, Y. Zhang //IEEE transactions on vehicular technology. – 2012. – Т. 61. – №. 3. – С. 1234-1250.

95. **Zuras, D.** On squaring and multiplying large integers / D. Zuras //Proceedings of IEEE 11th Symposium on Computer Arithmetic. – IEEE, 1993. – С. 260-271.

96. **Акушский, И.Я.** Машинная арифметика в остаточных классах / И.Я. Акушский, Д.И. Юдицкий. — Москва: Советское радио, 1968. — 440 с.

97. **Акушский, И.Я.** О новой позиционной характеристике непозиционного кода и ее применении / И.Я. Акушский, В.М. Бурцев, И.Т. Пак // Теория кодирования и оптимизации сложных систем. — Алма-Ата: Наука. — 1977. — с. 8–16.

98. **Анализ методов обнаружения движения в цифровых системах видеонаблюдения** / В.А. Кучуков, М.Г. Бабенко, Е.А. Кучукова, Н.Г. Гудиева // Современная наука и инновации. — 2018. — №3. — с. 8-14.

99. **Вернер, М.** Основы кодирования./ М. Вернер. — М.: Техносфера, 2004. — 288 с.

100. **Виноград, С.** Надежные вычисления при наличии шумов / С. Виноград, Д. Д. Коуэн.–М.: «Наука». — 1968.

101. **Гуляев, В.А.** Организация живучих вычислительных структур / В.А. Гуляев, А.Г. Додонов, С.П. Пелехов //Киев: Наук. думка. – 1982.

102. **Евразийский патент на изобретение № 038389**, Устройство сравнения и определения знака чисел, представленных в системе остаточных классов / Дерябин М.А., Бабенко М.Г., Кучуков В.А., Назаров А.С., Кучеров Н.Н.; заявитель и патентообладатель ФГАОУ ВО "Северо-Кавказский федеральный университет". — № 202090736; заявл. 14.04.2020; опубл. 20.08.2021.

103. **Исследование эффективных методов перевода чисел из системы остаточных классов в позиционную систему счисления на FPGA** / Н.И. Червяков, В.А. Кучуков, Н.Н. Кучеров, Н.Н. Кучукова //Современная наука и инновации. – 2017. – №. 3. – С. 46-52.

104. **Касперски, К.** Могущество кодов Рида-Соломона, или Информация, воскресшая из пепла / К. Касперски //Системный администратор. – 2003. – №. 8. – С. 88-94.

105. **Кучуков, В.А.** Применение системы остаточных классов для повышения эффективности операции умножения с накоплением / В.А. Кучуков, Н.Н. Кучеров // Вестник современных цифровых технологий. — 2022. — № 12. — С. 38-45.

106. **Кучуков, В.А.** РЕАЛИЗАЦИЯ ФИЛЬТРА ПОВЫШЕНИЯ РЕЗКОСТИ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ НА FPGA / В.А. Кучуков, Н.И. Червяков // Инфокоммуникационные технологии. — 2015. – Т. 13. — №. 4. – С. 361-365.

107. **Малашевич, Б.М.** Краткие основы и история создания отечественных модулярных ЭВМ. Истоки модулярной арифметики / Б.М. Малашевич // Сборник трудов SoRuCom-2017. — 2017. — с.193–207.

108. **Модификация алгоритма обнаружения и локализации ошибки в системе остаточных классов** / А. Гладков, В. Кучуков, М. Бабенко [и др.] // Труды Института системного программирования РАН. – 2022. – Т. 34. – № 3. – С. 75-88.

109. **Модулярные параллельные вычислительные структуры нейропроцессорных систем** / Н.И. Червяков, П.А. Сахнюк, А.В. Шапошников, С.А. Ряднов. - Москва : Физматлит, 2003. - 287 с.

110. **Нейман, Д.** Вероятностная логика и синтез надежных организмов из ненадежных компонент /Д. Нейман //В сб. ст. под ред. К. Шеннона и Дж. Маккарти. Автоматы. Пер. с англ. Под ред. Ляпунова ААМ: ИЛ. — 1956.

111. **Нейрокомпьютеры в остаточных классах**; Кн. 11: Учеб. пособие для вузов/ Н. И. Червяков, П. А. Сахнюк, А. В. Шапошников, А. Н. Макоха. – Москва: Радиотехника, 2003. – 272 с.

112. **Новая схема хранения информации в облачной среде на основе системы остаточных классов и схем разделения секрета** / Н.И. Червяков, М.Г. Бабенко, Н.Н. Кучеров [и др.] // Современная наука и инновации. — 2017. — № 4 (20). — С. 21–25.

113. **Обнаружение и исправление ошибок в дискретных устройствах** / Под ред. В.С. Толстякова. М.: Сов. радио, 1972. — 288 с.

114. **Пат. 2767450 Российская Федерация**, Способ определения знака числа в системе остаточных классов / Бабенко М.Г., Кучуков В.А., Черных А.Н., Кучеров Н.Н.; заявитель и патентообладатель ФГАОУ ВО "Северо-Кавказский федеральный университет". — № 2021108953; заявл. 01.04.2021; опубл. 17.03.2022 Бюл. № 8.

115. **Патент № 2483346 Российская Федерация, МПК G06F 11/08, G06F 7/72.** УСТРОЙСТВО ДЛЯ ОБНАРУЖЕНИЯ ПЕРЕПОЛНЕНИЯ ДИНАМИЧЕСКОГО ДИАПАЗОНА, ОПРЕДЕЛЕНИЯ ОШИБКИ И ЛОКАЛИЗАЦИИ НЕИСПРАВНОСТИ ВЫЧИСЛИТЕЛЬНОГО КАНАЛА В ЭВМ, ФУНКЦИОНИРУЮЩИХ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ : № 2011145755/08; заявл. 10.11.2011; опубл. 27.05.2013 /Червяков Н.И., Бабенко М.Г., Ляхов П.А., Лавриненко И.Н., Лавриненко А.В.; заявитель и патентообладатель Федеральное государственное автономное образовательное учреждение высшего профессионального образования "Северо-Кавказский федеральный университет". — 10 с.

116. **Патент № 2653257 Российская Федерация, МПК G06F 11/08, G06F 7/72.** Устройство обнаружения и коррекции ошибки модулярного кода : № 2017126350; заявл. 21.07.2017; опубл. 07.05.2018 / Червяков Н.И., Кучуков В.А., Бабенко М.Г., Кучукова Н.Н.; заявитель и патентообладатель ФГАОУ ВО «Северо-Кавказский федеральный университет». — 3 с.

117. **Патент № 2744815 Российская Федерация, МПК G06F 7/72.** Устройство для перевода чисел из системы остаточных классов и расширения оснований : № 2020120649; заявл. 22.06.2020; опубл. 16.03.2021 / Бабенко М.Г., Кучуков В.А., Черных А.Н., Кучеров Н.Н.; заявитель и патентообладатель Федеральное государственное автономное образовательное учреждение высшего образования "Северо-Кавказский федеральный университет". — 13 с.

118. **Патент № 2747371 Российская Федерация, МПК G06F 7/38, G06F 7/72.** Устройство определения знака числа, представленного в системе остаточных классов : № 2020134778; заявл. 22.10.2020; опубл.

04.05.2021 / Бабенко М.Г., Кучуков В.А.; заявитель и патентообладатель Федеральное государственное автономное образовательное учреждение высшего образования "Северо-Кавказский федеральный университет". — 15 с.

119. **Патент № 2751992 Российская Федерация**, Устройство сравнения чисел, представленных в системе остаточных классов / Бабенко М.Г., Кучуков В.А.; заявитель и патентообладатель ФГАОУ ВО "Северо-Кавказский федеральный университет". – № 2020134772; заявл. 22.10.2020; опубл. 21.07.2021.

120. **Патент № 2780148 Российская Федерация**, Система распределенного хранения данных / Бабенко М.Г., Кучуков В.А., Кучеров Н.Н., Гладков А.В.; заявитель и патентообладатель ФГАОУ ВО "Северо-Кавказский федеральный университет". – № 2021138986; заявл. 27.12.2021; опубл. 19.09.2022, Бюл. № 26.

121. **Повышение эффективности обработки информации в АСУ**. / Под ред. В.И. Ключко. МО СССР, 1985. — 328 с.

122. **Приближенный метод определения знака числа в системе остаточных классов и его техническая реализация** /Н.И. Червяков, М.Г. Бабенко, П.А. Ляхов, И.Н. Лавриненко //Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика. Телекоммуникации. Управление. — 2013. — №. 4 (176). — С. 131-141.

123. **Разработка нового нейросетевого метода вычисления модульного умножения в системе остаточных классов** / Н.И. Червяков, М.Г. Бабенко, А.Н. Черных [и др.] //Нейрокомпьютеры: разработка, применение. – 2016. – №. 10. – С. 41-48.

124. **Свидетельство о государственной регистрации программы для ЭВМ 2017660854 Российская Федерация**. Программа моделирования алгоритмов кодирования и декодирования данных для цифровой обработки сигналов на базе системы остаточных классов / Червяков Н.И., Бабенко М.Г., Дерябин М.А., Кучеров Н.Н., Кучуков В.А., Кучукова Н.Н.; заявитель и правообладатель Федеральное государственное автономное

образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2017617581; заявл. 31.07.2017; опубл 28.09.2017. – 1 с.

125. **Свидетельство о государственной регистрации программы для ЭВМ 2019610808 Российская Федерация**. Модуль кодирования и декодирования данных в системе остаточных классов / Бабенко М.Г., Червяков Н.И., Черных А.Н., Кучуков В.А., Кучеров Н.Н., Кучукова Е.А., Аль-Гальда С. Ч.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2018665334; заявл. 28.12.2018; опубл 18.01.2019. – 1 с.

126. **Свидетельство о государственной регистрации программы для ЭВМ 2019611375 Российская Федерация**. Распределенная система надежного хранения и обработки данных в мультиоблачной среде / Бабенко М.Г., Червяков Н.И., Черных А.Н., Кучеров Н.Н., Кучуков В.А., Кучукова Е.А., Аль-Гальда С. Ч.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2018665335; заявл. 28.12.2018; опубл 18.01.2019. – 1 с.

127. **Свидетельство о государственной регистрации программы для ЭВМ 2020610041 Российская Федерация**. Программа избыточного кодирования и декодирования модулярного кода / Кучуков В.А., Бабенко М.Г.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2019666586; заявл. 17.12.2019; опубл 09.01.2020. – 1 с.

128. **Свидетельство о государственной регистрации программы для ЭВМ 2020618967 Российская Федерация**. Программа подготовки файлов для распределенного хранения данных в облаках / Кучеров Н.Н., Бабенко М.Г., Кучуков В.А., Ващенко И.С.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2020618381; заявл. 03.08.2020; опубл 10.08.2020. – 1 с.

129. **Свидетельство о государственной регистрации программы для ЭВМ 2020619140 Российская Федерация**. Программа восстановления полученных данных при распределенном хранении данных в облаках / Кучеров Н.Н., Бабенко М.Г., Кучуков В.А., Ващенко И.С.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2020618376; заявл. 03.08.2020; опубл 12.08.2020. – 1 с.

130. **Свидетельство о государственной регистрации программы для ЭВМ 2020660257 Российская Федерация**. Система моделирования исправления ошибок в модулярном коде / Бабенко М.Г., Кучуков В.А., Ващенко И.С.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2020619583; заявл. 28.08.2020; опубл 01.09.2020. – 1 с.

131. **Свидетельство о государственной регистрации программы для ЭВМ 2020660531 Российская Федерация**. Модуль выбора оснований системы остаточных классов для оптимизации минимально избыточного кода / Бабенко М.Г., Кучуков В.А., Ващенко И.С.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2020619552; заявл. 28.08.2020; опубл 04.09.2020. – 1 с.

132. **Свидетельство о государственной регистрации программы для ЭВМ 2022667010 Российская Федерация**. Генератор Verilog-модулей умножителя с накоплением (MAC)/ Кучуков В.А.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2022665935; заявл. 31.08.2022; опубл 13.09.2022. – 1 с.

133. **Свидетельство о государственной регистрации программы для ЭВМ 2022667011 Российская Федерация**. Генератор Verilog-модулей перевода из системы остаточных классов в позиционную систему счисления/ Кучуков В.А., Кучеров Н.Н., Вершков Н.А., Безуглова Е.С.;

заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2022665961; заявл. 31.08.2022; опубл 13.09.2022. – 1 с.

134. **Свидетельство о государственной регистрации программы для ЭВМ 2022667102 Российская Федерация**. Генератор Verilog-модулей перевода из позиционной системы счисления в систему остаточных классов / Кучуков В.А., Кучеров Н.Н., Вершков Н.А., Безуглова Е.С.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2022665941; заявл. 31.08.2022; опубл 14.09.2022. – 1 с.

135. **Свидетельство о государственной регистрации программы для ЭВМ 2023668328 Российская Федерация**. Генератор Verilog-модулей восстановления числа в СОК с модулями специального вида/ Кучуков В.А., Баршацкая Е.А.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2023667440; заявл. 22.08.2023; опубл 25.08.2023. – 1 с.

136. **Свидетельство о государственной регистрации программы для ЭВМ 2023668861 Российская Федерация**. Генератор Verilog-модулей исправления ошибки в избыточной системе остаточных классов / Кучуков В.А., Баршацкая Е.А.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2023667428; заявл. 22.08.2023; опубл 05.09.2023. – 1 с.

137. **Свидетельство о государственной регистрации программы для ЭВМ 2023669462 Российская Федерация**. Генератор Verilog-модулей исправления ошибки приближенным методом / Кучуков В.А., Баршацкая Е.А.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – № 2023667563; заявл. 22.08.2023; опубл 14.09.2023. – 1 с.

138. **Стахов, А.П.** Введение в алгоритмическую теорию измерения / А.П. Стахов. — Москва: Сов. радио, 1977. — 288 с.

139. **Стемпковский, А.Л.** Принципы рекурсивных модулярных вычислений / А.Л. Стемпковский, В.М. Амербаев, Р.А. Соловьев // Информационные технологии. — 2013. — № 2. — С. 22–27.

140. **Теория кодирования** / Т. Касами, Н. Токура, Ё. Ивадари, Я. Инагакию — Москва: Изд-во Мир. – 1978.

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

142. **Червяков, Н.И.** Оптимизация структуры нейронных сетей конечного кольца / Н.И. Червяков, А.В. Шапошников, П.А. Сахнюк //Нейрокомпьютеры: разработка, применение. — 2001. — №. 10. — С. 13-18.

143. **Эффективная реализация операции вычисления остатка от деления многоразрядных чисел на FPGA** / Н.И. Червяков, А.С. Назаров, Ю.В. Черногорова, Н.Н. Кучеров // Современная наука и инновации. – 2018. — №1(21). — С. 15-21.

144. **Эффективное сравнение чисел в системе остаточных классов на основе позиционной характеристики** / М.Г. Бабенко, А.Н. Черных, Н.И. Червяков [и др.] //Труды Института системного программирования РАН. – 2019. – Т. 31. — № 2. — с.187-202.

145. ГОСТ Р 56111–2014. Интегрированная логистическая поддержка экспортируемой продукции военного назначения. НОМЕНКЛАТУРА ПОКАЗАТЕЛЕЙ ЭКСПЛУАТАЦИОННО-ТЕХНИЧЕСКИХ ХАРАКТЕРИСТИК. М., 2015. 46 с. (Система стандартов по информ., библ. и изд. делу).

Список рисунков

Список таблиц

Приложение А

Результаты моделирования перевода чисел из позиционной системы счисления в систему остаточных классов

Таблица 13 — Результаты моделирования площади, мкм^2, для нейронной сети конечного кольца для чисел $2^n - 1$

Размерность входа, бит	n для $2^n - 1$						
	2	4	8	16	32	64	128
8	1019	1152					
16	2148	2424	2983				
32	4810	4997	5527	7031			
64	9709	9990	10567	12134	15919		
128	19580	19773	20518	22143	26103	35138	
256	39091	39488	40046	42053	46071	55499	76399

Таблица 14 — Результаты моделирования площади, мкм^2, для чисел $2^n + 1$ с использованием полупериода

Размерность входа, бит	n для $2^n + 1$						
	2	4	8	16	32	64	128
8	1271	735					
16	2900	1985	1553				
32	5740	5258	4129	3273			
64	11327	10048	8079	8423	6713		
128	22842	19784	15763	16472	16999	16949	
256	45700	39385	37041	32409	33394	36999	33916

Таблица 15 — Результаты моделирования площади, мкм^2, для чисел $2^n - 1$ с использованием периода

Размерность входа, бит	n для $2^n - 1$						
	2	4	8	16	32	64	128
8	1092	852					
16	2212	2022	1744				
32	4477	4227	4026	3580			
64	10236	8772	8186	7997	7182		
128	20545	18037	16446	15962	16010	14580	
256	41075	36027	33034	31961	31718	31807	29236

Таблица 16 — Результаты моделирования времени, пс, для нейронной сети конечного кольца для чисел $2^n - 1$

Размерность входа, бит	n для $2^n - 1$						
	2	4	8	16	32	64	128
8	1258	1245					
16	1938	1893	2456				
32	2517	2260	2985	4520			
64	3073	3505	4047	6695	12098		
128	3898	4220	4985	7335	13050	22323	
256	4594	5148	5738	8229	13502	23162	43758

Таблица 17 — Результаты моделирования времени, пс, для чисел $2^n + 1$ с использованием полупериода

Размерность входа, бит	n для $2^n + 1$						
	2	4	8	16	32	64	128
8	1378	992					
16	2677	1719	1832				
32	3140	3123	2500	3534			
64	3491	3889	2673	4244	6945		
128	4956	4405	3258	4357	7619	17946	
256	5762	5196	5733	4960	7990	18395	33055

Таблица 18 — Результаты моделирования времени, пс, для чисел $2^n - 1$ с использованием периода

Размерность входа, бит	n для $2^n - 1$						
	2	4	8	16	32	64	128
8	1586	1417					
16	2069	2407	2494				
32	4307	2958	3904	4464			
64	3326	3495	4741	7698	7995		
128	4529	4235	4635	8038	14633	15270	
256	4944	4950	5327	8555	14292	25321	30204

Приложение Б

Результаты моделирования перевода чисел из системы остаточных классов в позиционную систему счисления

Таблица 19 — Результаты моделирования перевода из СОК с использованием КТО

Набор модулей СОК	Время, пс		Площадь, мкм^2	
	КТО по формуле (3.1)	КТО по формуле (2.6)	КТО по формуле (3.1)	КТО по формуле (2.6)
8 бит				
{3, 5, 32}	3179	3778	3907	4387
{5, 7, 8}	2779	3959	3075	4913
16 бит				
{17, 31, 128}	11341	10315	24121	20733
{31, 63, 64}	9274	10902	19667	24606
{7, 9, 17, 64}	6232	9050	16551	20343
24 бита				
{127,255, 1024}	15308	16908	40473	39430
{31, 63, 65, 256}	13725	14786	54494	49298
{5, 7, 9, 17, 31, 128}	18710	18076	62996	50355
32 бита				
{1023, 2047, 4096}	23243	25670	86156	71282
{127, 255, 511, 512}	22117	26285	133813	96027
{31, 63, 65, 127, 512}	22648	24817	128017	97130
40 бит				
{8191, 16383, 16384}	36248	37082	180695	125940
{511, 1023, 2047, 2048}	34897	34517	221970	144970
{127, 255, 257, 511, 512}	28128	33364	187680	154589
{17, 31, 65, 129, 511, 512}	45871	39973	244390	146080
48 бит				
{65535, 65537, 131072}	32890	43082	128123	164738

{2047, 4095, 8191, 8192}	57580	46710	323001	186941
{257, 511, 1023, 1025, 2048}	47964	42236	304181	178997
{65, 127, 129, 257, 511, 2048}	44267	40408	311685	179716
{17, 31, 65, 127, 129, 511, 1024}	54379	46049	350783	202306
56 бит				
{8191, 16383, 32767, 32768}	63351	59613	410304	253370
{127, 257, 511, 513, 2047, 8192}	70819	55633	488340	264243
{17, 31, 65, 127, 129, 257, 511, 1024}	54756	44401	477065	244818
64 бита				
{32767, 65535, 131071, 131072}	71915	67670	510127	321253
{257, 511, 1025, 2049, 8191, 8192}	87625	69056	666328	348775
{65, 127, 257, 511, 1023, 2047, 8192}	90562	70824	697428	344874

Таблица 20 — Результаты моделирования перевода из СОК с использованием приближенного метода на основе КТО

Набор модулей СОК	Время, пс	Площадь, мкм^2
8 бит		
{3, 5, 32}	4305	5938
{5, 7, 8}	4291	4364
16 бит		
{17, 31, 128}	8450	19480
{31, 63, 64}	10122	15677
{7, 9, 17, 64}	9133	19409
24 бита		
{127, 255, 1024}	9708	41187
{31, 63, 65, 256}	11430	47429
{5, 7, 9, 17, 31, 128}	14269	59562
32 бита		
{1023, 2047, 4096}	17009	60189
{127, 255, 511, 512}	18292	83002
{31, 63, 65, 127, 512}	18538	83338

40 бит		
{8191, 16383, 16384}	18324	58853
{511, 1023, 2047, 2048}	21462	112655
{127, 255, 257, 511, 512}	21502	113206
{17, 31, 65, 129, 511, 512}	23474	166070
48 бит		
{65535, 65537, 131072}	17690	79530
{2047, 4095, 8191, 8192}	26012	134988
{257, 511, 1023, 1025, 2048}	25612	157479
{65, 127, 129, 257, 511, 2048}	25986	186805
{17, 31, 65, 127, 129, 511, 1024}	25869	192802
56 бит		
{262143, 524287, 1048576}	19733	67753
{8191, 16383, 32767, 32768}	25859	159475
{127, 257, 511, 513, 2047, 8192}	27881	212852
{17, 31, 65, 127, 129, 257, 511, 1024}	29990	258356
64 бита		
{2097151, 4194303, 4194304}	25703	118507
{32767, 65535, 131071, 131072}	26257	148716
{257, 511, 1025, 2049, 8191, 8192}	34038	289442
{65, 127, 257, 511, 1023, 2047, 8192}	31842	288270

Таблица 21 — Результаты моделирования перевода из СОК с использованием ОПСС

Набор модулей СОК	Время, пс	Площадь, мкм^2
8 бит		
{3, 5, 32}	4606	5362
{5, 7, 8}	4959	3697
16 бит		
{17, 31, 128}	10228	13162
{31, 63, 64}	10731	18486
{7, 9, 17, 64}	10214	17601
24 бита		

{127, 255, 1024}	14735	39128
{31, 63, 65, 256}	16093	41614
{5, 7, 9, 17, 31, 128}	23258	54688
32 бита		
{1023, 2047, 4096}	17537	47372
{127, 255, 511, 512}	25181	79377
{31, 63, 65, 127, 512}	27825	82852
40 бит		
{8191, 16383, 16384}	20319	53511
{511, 1023, 2047, 2048}	30332	93958
{127, 255, 257, 511, 512}	33560	118166
{17, 31, 65, 129, 511, 512}	36468	127930
48 бит		
{65535, 65537, 131072}	18521	44630
{2047, 4095, 8191, 8192}	33406	106273
{257, 511, 1023, 1025, 2048}	35415	134867
{65, 127, 129, 257, 511, 2048}	37324	150295
{17, 31, 65, 127, 129, 511, 1024}	46403	190258
56 бит		
{262143, 524287, 1048576}	27038	84413
{8191, 16383, 32767, 32768}	33748	135546
{127, 257, 511, 513, 2047, 8192}	45441	216881
{17, 31, 65, 127, 129, 257, 511, 1024}	57559	272161
64 бита		
{2097151, 4194303, 4194304}	33588	100191
{32767, 65535, 131071, 131072}	43339	160552
{257, 511, 1025, 2049, 8191, 8192}	50734	266294
{65, 127, 257, 511, 1023, 2047, 8192}	55446	295339

Таблица 22 — Результаты моделирования перевода из СОК с использованием модулей специального вида $2^n - 1, 2^n, 2^n + 1$

Размерность	Набор модулей СОК	Время, пс	Площадь, мкм^2
8 бит	{7, 8, 9}	2898	2290

16 бит	{63, 64, 65}	5546	4717
24 бита	{511, 512, 513}	7990	7186
32 бита	{2047, 2048, 2049}	9383	8731
40 бит	{16383, 16384, 16385}	12907	11097
48 бит	{131071, 131072, 131073}	14162	13436
56 бит	{524287, 524288, 524289}	16311	15118
64 бита	{4194303, 4194304, 4194305}	17514	17409

Таблица 23 — Результаты моделирования перевода из СОК с использованием модифицированной ОПСС

Набор модулей СОК	Время, пс	Площадь, мкм^2
8 бит		
{3, 5, 32}	3596	5677
{5, 7, 8}	4572	4001
16 бит		
{17, 31, 128}	9897	16990
{31, 63, 64}	11682	17766
{7, 9, 17, 64}	12972	19970
24 бита		
{127,255, 1024}	18788	38082
{31, 63, 65, 256}	22753	49281
{5, 7, 9, 17, 31, 128}	22700	49107
32 бита		
{1023, 2047, 4096}	26528	58980
{127, 255, 511, 512}	18881	85976
{31, 63, 65, 127, 512}	36575	90601
40 бит		
{8191, 16383, 16384}	38077	68848
{511, 1023, 2047, 2048}	47533	131283
{127, 255, 257, 511, 512}	54753	133459
{17, 31, 65, 129, 511, 512}	51500	136026
48 бит		
{65535, 65537, 131072}	42286	61424

{2047, 4095, 8191, 8192}	60749	157200
{257, 511, 1023, 1025, 2048}	63138	167454
{65, 127, 129, 257, 511, 2048}	64953	174963
{17, 31, 65, 127, 129, 511, 1024}	66226	190144
56 бит		
{262143, 524287, 1048576}	60389	139424
{8191, 16383, 32767, 32768}	71026	205156
{127, 257, 511, 513, 2047, 8192}	78990	268058
{17, 31, 65, 127, 129, 257, 511, 1024}	79344	258256
64 бита		
{2097151, 4194303, 4194304}	71272	157398
{32767, 65535, 131071, 131072}	90076	253470
{257, 511, 1025, 2049, 8191, 8192}	97191	317046
{65, 127, 257, 511, 1023, 2047, 8192}	94274	318773

Таблица 24 — Время выполнения перевода из СОК в ПСС, пс

Размерность, бит	8	16	24	32	40	48	56	64
Китайская теорема об остатках	2779	6232	13725	22117	28128	32890	44401	67670
Приближенная Китайская теорема об остатках	4291	8450	9708	17009	18324	17690	19733	25703
Обобщенная позиционная система счисления	4606	10214	14735	17537	20319	18521	27038	33588
Модифицированная обобщенная позиционная система счисления	3519	7178	11002	12788	16326	14672	28275	48984
Использование модулей $\{2^n - 1, 2^n, 2^n + 1\}$ [58]	2898	5546	7990	9383	12907	14162	16311	17514

Таблица 25 — Необходимая площадь для выполнения перевода из СОК в ПСС, мкм^2

Размерность, бит	8	16	24	32	40	48	56	64
Китайская теорема об остатках	3075	16551	39430	71282	125940	128123	244818	321253
Приближенная Китайская теорема об остатках	4364	15677	41187	60189	58853	79530	67753	118507
Обобщенная позиционная система счисления	3697	13162	39128	47372	53511	44630	84413	100191
Модифицированная обобщенная позиционная система счисления	4001	16143	34591	51015	53584	43822	131380	145944
Использование модулей $\{2^n - 1, 2^n, 2^n + 1\}$ [58]	2290	4717	7186	8731	11097	13436	15118	17409

Приложение В

Результаты моделирования сравнения чисел в системе остаточных классов

Таблица 26 — Результаты моделирования сравнения чисел в СОК с использованием КТО

Набор модулей СОК	Время, пс	Площадь, мкм^2
8 бит		
{3, 5, 32}	4502	9493
{5, 7, 8}	4861	10316
16 бит		
{17, 31, 128}	11807	43249
{31, 63, 64}	11939	51248
{7, 9, 17, 64}	10090	41390
24 бита		
{127,255, 1024}	17819	80510
{31, 63, 65, 256}	17033	103928
{5, 7, 9, 17, 31, 128}	19935	104343
32 бита		
{1023, 2047, 4096}	30411	171373
{127, 255, 511, 512}	27000	182927
{31, 63, 65, 127, 512}	29231	195520
40 бит		
{8191, 16383, 16384}	43218	272342
{511, 1023, 2047, 2048}	38621	291939
{127, 255, 257, 511, 512}	37974	301287
{17, 31, 65, 129, 511, 512}	43453	302396
48 бит		
{65535, 65537, 131072}	48279	375097
{2047, 4095, 8191, 8192}	52115	410671
{257, 511, 1023, 1025, 2048}	45699	396071
{65, 127, 129, 257, 511, 2048}	44394	367009

{17, 31, 65, 127, 129, 511, 1024}	49592	418652
56 бит		
{8191, 16383, 32767, 32768}	63430	532299
{127, 257, 511, 513, 2047, 8192}	63488	555283
{17, 31, 65, 127, 129, 257, 511, 1024}	48837	515335
64 бита		
{32767, 65535, 131071, 131072}	74673	668241
{257, 511, 1025, 2049, 8191, 8192}	73415	725673
{65, 127, 257, 511, 1023, 2047, 8192}	74934	717064

Таблица 27 — Результаты моделирования сравнения чисел в СОК с использованием приближенного метода на основе КТО

Набор модулей СОК	Время, пс		Площадь, мкм^2	
	КТО из [75]	КТО из [86]	КТО из [75]	КТО из [86]
8 бит				
{3, 5, 32}	2760	2673	7523	7719
{5, 7, 8}	4090	3018	7852	5821
16 бит				
{17, 31, 128}	6037	4865	29382	27920
{31, 63, 64}	6617	5229	20471	18370
{7, 9, 17, 64}	5847	5102	26984	24877
24 бита				
{127,255, 1024}	8878	7600	49854	42976
{31, 63, 65, 256}	7869	7866	61299	53905
{5, 7, 9, 17, 31, 128}	8974	8218	58490	51637
32 бита				
{1023, 2047, 4096}	12647	10883	67128	63441
{127, 255, 511, 512}	11864	9516	88572	83200
{31, 63, 65, 127, 512}	11465	9333	97176	75929
40 бит				
{8191, 16383, 16384}	13652	13000	67403	91384
{511, 1023, 2047, 2048}	14150	11835	122991	109628

{127, 255, 257, 511, 512}	15225	13398	138348	133039
{17, 31, 65, 129, 511, 512}	15507	13285	134004	136291
48 бит				
{65535, 65537, 131072}	15726	14958	94430	133483
{2047, 4095, 8191, 8192}	17158	16846	168299	151515
{257, 511, 1023, 1025, 2048}	18574	15640	180966	172364
{65, 127, 129, 257, 511, 2048}	18394	17184	176308	172598
{17, 31, 65, 127, 129, 511, 1024}	16496	15829	180479	164904
56 бит				
{8191, 16383, 32767, 32768}	19579	18000	207403	200591
{127, 257, 511, 513, 2047, 8192}	19885	18148	230073	218963
{17, 31, 65, 127, 129, 257, 511, 1024}	19230	20086	228507	219172
64 бита				
{32767, 65535, 131071, 131072}	16972	16135	202357	183722
{257, 511, 1025, 2049, 8191, 8192}	18814	19926	287363	304776
{65, 127, 257, 511, 1023, 2047, 8192}	19879	18877	241079	250255

Таблица 28 — Результаты моделирования сравнения чисел в СОК с использованием ОПСС

Набор модулей СОК	Время, пс	Площадь, мкм^2
8 бит		
{3,5,32}	3924	6349
{5,7,8}	4137	6142
16 бит		
{17,31,128}	7300	16561
{31,63,64}	6564	13988
{7,9,17,64}	8581	15869
24 бита		
{127,255,1024}	9102	27359
{31,63,65,256}	11175	27413
{5,7,9,17,31,128}	14329	31054

32 бита		
{1023,2047,4096}	11947	34346
{127,255,511,512}	15005	45259
{31, 63, 65, 127, 512}	17649	43878
40 бит		
{8191, 16383, 16384}	10743	27074
{511,1023,2047, 2048}	18372	65223
{127,255,257,511, 512}	18704	49650
{17,31,65,129,511,512}	24812	63647
48 бит		
{65535,65537,131072}	11311	35366
{2047,4095,8191,8192}	19092	54998
{257,511,1023,1025,2048}	23021	77816
{65,127,129,257,511,2048}	23018	71005
{17,31,65,127,129,511, 1024}	29150	89818
56 бит		
{8191,16383,32767,32768}	21804	64455
{127,257,511,513,2047, 8192}	32326	117203
{17,31,65,127,129,257,511, 1024}	37258	117266
64 бита		
{32767,65535,131071, 131072}	23163	75101
{257,511,1025,2049,8191, 8192}	36865	130936
{65,127,257,511,1023, 2047, 8192}	39124	134526

Таблица 29 — Результаты моделирования сравнения чисел в СОК с использованием функции ядра

Набор модулей СОК	Время, пс	Площадь, мкм^2
8 бит		
{3,5,32}	2615	6791
{5,7,8}	2938	5834
16 бит		
{17,31,128}	5406	25781
{31,63,64}	5276	19360

{7,9,17,64}	5005	22465
24 бита		
{127,255,1024}	6901	32794
{31,63,65,256}	7322	53824
{5,7,9,17,31,128}	7625	55213
32 бита		
{1023,2047,4096}	8983	45870
{127,255,511,512}	8971	85485
{31, 63, 65, 127, 512}	10466	92992
40 бит		
{8191, 16383, 16384}	10814	61357
{511,1023,2047, 2048}	12774	111684
{127,255,257,511, 512}	11432	129387
{17,31,65,129,511,512}	13273	129856
48 бит		
{65535,65537,131072}	8847	57804
{2047,4095,8191,8192}	14824	147287
{257,511,1023,1025,2048}	14884	175945
{65,127,129,257,511,2048}	16776	167217
{17,31,65,127,129,511, 1024}	15099	160452
56 бит		
{8191,16383,32767,32768}	16625	197892
{127,257,511,513,2047, 8192}	18103	219940
{17,31,65,127,129,257,511, 1024}	16924	212332
64 бита		
{32767,65535,131071, 131072}	16925	209885
{257,511,1025,2049,8191, 8192}	20086	294789
{65,127,257,511,1023, 2047, 8192}	17995	247680

Таблица 30 — Результаты моделирования сравнения чисел в СОК с использованием модифицированного приближенного метода на основе КТО

Набор модулей СОК	Время, пс	Площадь, мкм^2
8 бит		

{3,5,32}	2396	7451
{5,7,8}	2281	5863
16 бит		
{17,31,128}	4839	25124
{31,63,64}	5512	18980
{7,9,17,64}	6215	23447
24 бита		
{127,255,1024}	7207	32612
{31,63,65,256}	8273	53476
{5,7,9,17,31,128}	7662	51566
32 бита		
{1023,2047,4096}	9952	44088
{127,255,511,512}	9794	84556
{31, 63, 65, 127, 512}	9248	80171
40 бит		
{8191, 16383, 16384}	12111	54354
{511,1023,2047, 2048}	12686	115094
{127,255,257,511, 512}	12236	132533
{17,31,65,129,511,512}	12398	132910
48 бит		
{65535,65537,131072}	14958	133483
{2047,4095,8191,8192}	17553	144851
{257,511,1023,1025,2048}	15661	172113
{65,127,129,257,511,2048}	17286	163297
{17,31,65,127,129,511, 1024}	16423	165445
56 бит		
{8191,16383,32767,32768}	18000	200591
{127,257,511,513,2047, 8192}	18148	218963
{17,31,65,127,129,257,511, 1024}	20086	219172
64 бита		
{32767,65535,131071, 131072}	16135	183722
{257,511,1025,2049,8191, 8192}	19926	304776
{65,127,257,511,1023, 2047, 8192}	18877	250255

Таблица 31 — Время выполнения сравнения чисел в СОК, пс

Размерность, бит	8	16	24	32	40	48	56	64
Китайская теорема об остатках	3183	7410	15590	27000	37682	44394	48837	73415
Приближенная Китайская теорема об остатках	2673	4865	7600	9516	11835	14958	18000	16135
Обобщенная позиционная система счисления	3924	6564	9102	11947	10743	11311	21804	23163
Функция ядра	2615	5005	6901	8971	10814	8847	16625	16925
Модифицированный приближенный метод на основе КТО	2281	4839	7207	9248	12111	14958	18000	16135

Таблица 32 — Необходимая площадь для выполнения сравнения чисел в СОК, мкм^2

Размерность, бит	8	16	24	32	40	48	56	64
Китайская теорема об остатках	6910	36705	80510	171373	272342	367009	515335	668241
Приближенная Китайская теорема об остатках	5821	18370	42976	63441	67403	94430	200591	183722
Обобщенная позиционная система счисления	6142	13988	27359	34346	27074	35366	64455	75101
Функция ядра	5834	19360	32794	45870	61357	57804	197892	209885
Модифицированный приближенный метод на основе КТО	5863	18980	32612	44088	54354	133483	200591	183722

Приложение Г

Результаты моделирования умножения с накоплением в систему остаточных классов

Таблица 33 — Результаты моделирования умножения с накоплением

Модель	Время, пс	Площадь, m^2
8 бит		
двоичное представление	1799	2881
{3, 5, 32}	2087	4282
{5, 7, 8}	2168	4423
16 бит		
двоичное представление	3563	10132
{31, 63, 64}	4623	14351
{7, 9, 17, 64}	3366	13241
24 бита		
двоичное представление	6461	22000
{255, 257, 512}	6576	32284
{31, 63, 65, 256}	5071	25825
{5, 7, 9, 17, 31, 128}	3807	21229
32 бита		
двоичное представление	8692	36806
{1023, 2047, 4096}	9029	47713
{127, 255, 511, 512}	6154	40409
{31, 63, 65, 127, 512}	5290	35975
40 бит		
двоичное представление	10959	54686
{8191, 16383, 16384}	11518	71198
{511, 1023, 2047, 2048}	9052	60357
{127, 255, 257, 511, 512}	6576	55403
{17, 31, 65, 129, 511, 512}	6410	51456

48 бит		
двоичное представление	11359	80942
{65535, 65537, 131072}	14767	103205
{2047, 4095, 8191, 8192}	10290	82370
{257, 511, 1023, 1025, 2048]}	9233	75995
{65, 127, 129, 257, 511, 2048}	6576	68117
{17, 31, 65, 127, 129, 511, 1024}	6137	61729
56 бит		
двоичное представление	13228	106871
{262143, 524287, 1048576}	14403	126156
{8191, 16383, 32767, 32768}	12362	106798
{511, 1025, 2049, 8191, 16384}	10339	99219
{127, 257, 511, 513, 2047, 8192}	8915	88215
{31, 65, 127, 257, 513, 2047, 2048}	8972	84918
{17, 31, 65, 127, 129, 257, 511, 1024}	6576	76724
64 бита		
двоичное представление	15068	137773
{2097151, 4194303, 4194304}	16771	162240
{32767, 65535, 131071, 131072}	13287	135746
{2047, 4097, 8191, 16383, 32768}	11539	120158
{257, 511, 1025, 2049, 8191, 8192}	10386	112625
{65, 127, 257, 511, 1023, 2047, 8192}	8962	96373

Приложение Д

Акт о внедрении результатов диссертационного исследования

Infocom

Общество с ограниченной ответственностью
«Инфоком-С»

355035, г. Ставрополь, ул. Суворова, 7
info@infocom-s.ru www.infocom-s.ru
+7 (8652) 20-58-20

ИНН : 2635811319 ОКПО : 38852042
КПП: 263501001 ОГРН: 1122651011559
+7 (495) 700-00-65

04.04.24 № 71

На №_____от_____

Утверждаю
Генеральный директор
ООО «Инфоком-С»
_____ В.В. Копытов

Акт

о внедрении результатов диссертационного исследования Кучукова В.А. на тему «Разработка методов и программных средств повышения производительности отказоустойчивых вычислительных систем, работающих в модулярном коде».

Комиссия в составе

председатель – генеральный директор, доктор технических наук, профессор Копытов В.В.
Члены комиссии – директор по проектам, кандидат технических наук, доцент Демурчев Н.Г.
руководитель департамента комплексной безопасности Касимов Р.И.

Составили настоящий акт о том, что результаты диссертационного исследования Кучукова В.А. на тему «Разработка методов и программных средств повышения производительности отказоустойчивых вычислительных систем, работающих в модулярном коде», включающей такие результаты как:

1. Метод обнаружения и локализации ошибок распределенной обработки и хранения информации, основанный на использовании несбалансированной системы остаточных классов.

2. Метод перевода из СОК в позиционную систему счисления на основе модифицированной обобщенной позиционной системы счисления.

Указанные результаты в работе представленной на соискание ученой степени кандидата технических наук использованы при выполнении договора на выполнение прикладных научных исследований и экспериментальных разработок по теме «Доработка кроссплатформенной PSIM-системы Darvis для обеспечения безопасности промышленных предприятий» (в соответствии с соглашением о предоставлении гранта № 2022-550-28 от 26.12.2022 (Шифр 08.08.6)) с Российским фондом развития интернет технологий.

Использование указанных результатов позволяет организовать защищенный обмен данными с информационными сенсорами с высокой надежностью и низкой избыточностью при построении систем промышленной безопасности на крупных промышленных объектах.

Председатель комиссии: _____ Копытов В.В.

Члены комиссии: _____ Демурчев Н.Г.

_____ Касимов Р.И.