Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

ЛАВРИНЕНКО АНТОН ВИКТОРОВИЧ

РАЗРАБОТКА МЕТОДОВ МОДЕЛИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СТРУКТУР ОТКАЗОУСТОЙЧИВЫХ МОДУЛЯРНЫХ НЕЙРОКОМПЬЮТЕРОВ ДЛЯ ОБРАБОТКИ ДАННЫХ БОЛЬШОЙ РАЗМЕРНОСТИ

Специальность: 05.13.18 – Математическое моделирование, численные методы и комплексы программ

ДИССЕРТАЦИЯ

на соискание ученой степени кандидата технических наук

Научный руководитель:
доктор технических наук,
профессор Калмыков И.А.

Ставрополь – 2016

Оглавление

Введение

В условиях развития информационного общества появляются новые задачи, связанные с проблемой передачи и обработки многоразрядных чисел. Важные для теории и практики математические задачи, требующие таких вычислений и больших вычислительных ресурсов, лежат в области прикладной и вычислительной теории чисел. Большинство таких задач содержат целочисленные вычисления с числами, принимающими значения из больших или сверхбольших машинных диапазонов, причем результаты должны быть точными без округлений.

Особенностью традиционных вычислительных средств является наличие ограниченной разрядной сетки, которая приводит к вычислительной сложности при выполнении операций над числами большой размерности.

Вычисления с многоразрядными числами или вычисления с величинами, меняющимися в больших диапазонах, являются одной их областей, в которой СОК имеет преимущество перед позиционными системами счисления. В настоящее время проводятся исследования по обработке данных большой разрядности, при которых значения целочисленных переменных значительно, в 10^3–10^6 и более раз превышает динамический диапазон серийной вычислительной техники.

Модулярные вычисления играют важную роль в приложениях, где используются числа большой разрядности, например, в системах безопасности, цифровой подписи и других.

Так, для обеспечения высокой степени безопасности для защищенной информации в системах безопасности, числа имеют значения в диапазоне 2^{600} – 2^{700}. При модулярной обработке эти числа разделяются на малые форматы до единиц и десятков бит, что приводит к существенному преимуществу в скорости их компьютерной реализации [47-50].

Одним из перспективных направлений модулярной арифметики является разработка математических моделей для исследования методов и алгоритмов построения высокопроизводительных и надежных вычислительных средств.

Основным инструментом повышения показателей надежности является введение избыточности в систему.

Известно три основных метода повышения надежности вычислительных устройств: резервирование; применение сложных позиционных кодов и применение корректирующих кодов в модулярной арифметике. Применение резервирования приводит к большой избыточности, а применение позиционных кодов к отсутствию контроля выполнения арифметических операций. Коды модулярной арифметики полностью арифметичны, что позволяет их использовать для повышения надежности вычислений. Эта особенность модулярной арифметики широко применяется для решения проблемы повышения отказоустойчивости вычислительных структур и является мощным инструментом для автоматического обнаружения, локализации и коррекции ошибок.

Признавая важность исследований в рассматриваемой области, отметим, что научных работ посвященных сложным и многообразным проблемам теории и практики модулярной арифметики, реализуемой в нейросетевом базисе, явно недостаточно. Кроме того, недостаточно рассмотрены вопросы построения отказоустойчивых модулярных нейрокомпьютеров.

Значительный научный вклад в теорию и практику создания вычислительных нейросетевых структур на основе модулярной арифметики внесли отечественные и зарубежные исследователи: И.Я. Акушский, Д.И. Юдицкий, В.М. Амербаев, А.А. Коляда, А.И. Галушкин, И.Т. Пак, М.В. Синьков, В.А. Торгашев, И.А. Калмыков, О.А. Финько, Н.И. Червяков, Д. Свобода, N. Szabo, M. Valach, , Hiusat, A. Huang, B. Purhami, W. Ienkns, H. Krisha, A. Omondi, A. Premkumar, I. Ramires, A. Curcik, L.Yang, D. Zhang, P Steffan и другие.

Таким образом, как с теоретической, так и с практической точки зрения следует признать необходимость в исследовании вышеназванных проблем, носящих актуальный характер.

Существующие потребности обработки данных большой размерности обуславливают противоречие в практике, состоящее в том, что с одной стороны существует объективная необходимость в обработке данных большой размерности, с другой стороны существующие разрядные сетки современных ЭВМ не позволяют представить такие данные. Одним из путей преодоления вышеприведенного противоречия в практике является применение модулярной арифметики.

Алгоритмы выполнения модульных операций системы остаточных классов позволяют создавать устройства, обладающая высокой скоростью вычисления арифметических операций, но препятствием для широкого применения и внедрения устройств такого типа является отсутствие простых методов вычисления позиционных характеристик модулярного кода, являющиеся ядром немодульных логических операций, используемые при синтезе отказоустойчивых нейрокомпьютеров.

С целью удовлетворения требований, предъявленных к позиционной характеристике, в работе рассмотрена новая позиционная характеристика модулярного числа и ее применение для реализации немодульных операций в СОК.

Совместное применение системы остаточных классов и искусственных нейронных сетей позволяет создать новую отказоустойчивую модель в виде модулярных нейрокомпьютеров на базе FPGA, обладающую высокой скоростью вычислений.

Проблема обеспечения надежности модулярных нейрокомпьютерных систем является в настоящее время актуальной ввиду усложнения структуры самих нейрокомпьютеров и увеличения сложности выполняемых ими функций.

При разработке вычислительных средств на базе интеграции модулярной арифметики и искусственных нейронных сетей для вычислений в сверхбольших

диапазонах, необходимо преодолеть проблему вычисления трудных операций, которые требуют информацию о числовом значении модулярной величины, а не только значения остаточного представления. Для преодоления этой проблемы в работе используется подход, основанный на поиске простых алгоритмов вычисления немодульных процедур и чтобы они выполнялись модульно. Для реализации этого подхода предлагается позиционная характеристика модулярной величины или числовой величины в модулярном коде на основе КТО с дробными числами удовлетворяющей широкому спектру требований. От сложности вычисления позиционной характеристики зависит сложность выполнения немодульных операций в модулярной арифметике.

Таким образом, возникает объективное противоречие между необходимостью разработки высокопроизводительных отказоустойчивых нейрокомпьютеров и отсутствием единой методологии для решения задач синтеза отказоустойчивых вычислительных структур при ограниченном времени определения отказа или сбоя. Разрешение данного противоречия возможно при разработке методов и алгоритмов вычисления позиционных характеристик с малой вычислительной сложностью и синтеза на их основе устройств, реализующих быстрое вычисление немодульных (логических) операций модулярного кода.

За последние несколько лет исследователями были изучены вопросы определения и локализации ошибочных разрядов при вычислениях в модулярных вычислительных структурах. Однако, заметно отсутствие опубликованных ранее работ по проблеме быстрого исправления ошибок. Поскольку отдельная ошибка в любом элементе вычислительной структуры может привести к получению неправильного результата вычислений, полезно было бы включить способы быстрого определения, локализации и коррекции ошибки, которые позволяли бы получить правильный результат. Эта особенность очень важна в системе с высокоинтегрированными компонентами, такими, как программируемые логические интегральные схемы, которые широко используются для построения современных вычислительных систем. Архитектура модулярных

нейрокомпьютеров является отказоустойчивой за счет применения корректирующих свойств модулярной обработки данных.

Исходя из вышеизложенного, разработка методов и алгоритмов, используемых для синтеза отказоустойчивых модулярных нейрокомпьютеров, является актуальной научно – исследовательской задачей.

Целью диссертационного исследования является повышение отказоустойчивости модулярного нейрокомпьютера.

Объектом исследования выступают – модулярные нейрокомпьютеры.

Предметом исследования – являются математические методы и алгоритмы синтеза моделей отказоустойчивых модулярных нейрокомпьютеров для обработки данных большой разрядности.

Научная задача исследований состоит в разработке новых математических моделей функциональных устройств модулярных нейрокомпьютеров, численных методов вычисления немодульных операций на основе приближенных вычислений приближенной характеристики, а также комплексы программ, применение которых позволит повысить отказоустойчивость.

Для решения поставленной общей научной задачи была произведена ее декомпозиция на ряд частных задач:

1. Анализ математических моделей и алгоритмов создания многоразрядных отказоустойчивых нейрокомпьютеров на основе интеграции модулярной арифметики и искусственных нейронных сетей, отличающихся от традиционных подходов низкой вычислительной сложностью.

2. Развитие нового метода и алгоритмов приближенного вычисления позиционных характеристик для исследования математических моделей таких немодульных операций системы остаточных классов как: определение знака числа; сравнение модулярных чисел; деления и переполнения динамического диапазона; локализация ошибочного разряда и коррекция ошибок.

3. Разработка методов и алгоритмов итерационного модулярного деления многоразрядных чисел в формате СОК и на основе КТО с дробными числами.

4. Разработка численного метода аппроксимации и уточнения частного при итерационном делении модулярных чисел на основе КТО с дробными числами с целью проведения вычислительного эксперимента.

5. Разработка метода, алгоритма и модели многоразрядного отказоустойчивого модулярного нейрокомпьютера с автоматической коррекцией ошибок в динамике вычислительного процесса на основе расширения системы оснований СОК с использованием синдрома ошибок.

6. Разработка многофункциональной модели для контроля и диагностики вычислительных каналов модулярного нейрокомпьютера на основе адаптации дробной части позиционной характеристики к корректирующим свойствам кодов СОК.

7. Создание комплекса программ для экспериментального исследования на языке VHDL в среде ISE Design Suite 14.7 функциональных устройств модулярного нейрокомпьютера в базе программируемых логических интегральных схем.

Методы исследования базируются на использовании математического аппарата высшей алгебры, теории чисел, теории алгоритмов, численных методах, теории вероятности, теории надежности, теории искусственных нейронных сетей, нейроинформатике, математическом моделировании, системном анализе и теории приближенных вычислений.

Научная новизна результатов исследования:

1. Математические модели вычисления основных проблемных немодульных операций позволили устранить недостатки традиционного подхода и тем самым мотивировали исследования других альтернативных решений применения СОК в отказоустойчивых модулярных нейрокомпьютерах, отличающихся малой вычислительной сложностью.

2. Создание математических моделей и алгоритмов для вычисления знака и сравнения модулярных чисел на основе универсальной приближенной позиционной характеристики позволили разработать соответствующие

функциональные блоки, которые обладают высокой скоростью и малыми аппаратными затратами.

3. Разработан метод и аппаратная реализация модулярного деления чисел в формате СОК, когда делимое, делитель и промежуточные результаты представлены в СОК, чем исключаются промежуточные преобразования внутри алгоритма, присущее традиционным методам деления и использовать регулярную структуру современных СБИС.

4. Развитие нового метода и алгоритма итерационного деления модулярных чисел, основанного на использовании в каждой итерации только сравнения соседних промежуточных частных. Такой подход делает алгоритм более быстрым, точным и, следовательно, более подходящим для многоразрядного деления, что является усовершенствованием других публикаций и считать его лучшим алгоритмом деления на сегодняшний день.

5. Численный метод аппроксимации и уточнения частного при итерационном делении модулярных чисел для оптимального выбора наборов СОК.

6. Предложенные модель отказоустойчивого модулярного нейрокомпьютера с коррекцией одиночной ошибки на основе расширения системы оснований СОК и многофункциональная модель контроля вычислительных каналов нейрокомпьютера на основе приближенного метода позволили снизить вычислительную сложность по сравнению с традиционными в n раз, где n – число модулей СОК и обеспечивают контроль работы в динамике вычислительного процесса.

7. Разработан комплекс программ и математических методов моделирования для исследования основных блоков структурно-функциональной организации отказоустойчивых модулярных нейрокомпьютеров с целью оценки их характеристик и сравнения с характеристиками традиционных нейрокомпьютеров.

Достоверность результатов обеспечивается корректным и обоснованным применением методов математического моделирования и строгостью проводимых

математических доказательств. Справедливость выводов относительно эффективности предложенных моделей и методов подтверждена математическим моделированием в базе ПЛИС с использованием зарегистрированных в установленном порядке программных средств.

Практическая ценность результатов исследований. Разработанные программные продукты позволяют реализовать математические методы моделирования сложных функциональных устройств отказоустойчивых модулярных нейрокомпьютеров на базе нейронных сетей конечного кольца. Разработана структурно-функциональная организация и соответствующая математическая модель отказоустойчивого модулярного нейрокомпьютера, корректно выполняющая свои функции при отказе базисных элементов. Они доведены до решений в виде функциональных схем и подтверждены патентами РФ, позволяющие создавать перспективные специализированные отказоустойчивые нейросетевые системы обработки данных большой разрядности в реальном масштабе времени. Технические решения по разработке основных узлов вычислительной модулярной техники создают реальную основу для постановки ОКР.

На защиту выносятся следующие научные результаты:

1. Метод и математические модели приближенного вычисления позиционной характеристики модулярного кода на основе КТО с дробными числами и адаптация к корректирующим свойствам СОК с целью создания универсального алгоритма реализации немодульных операций, используемых при синтезе отказоустойчивых модулярных нейрокомпьютеров.

2. Математические модели и алгоритмы вычисления основных проблемных операций в СОК, используемые при разработке рациональных конфигураций нейросетевых функциональных устройств, учитывающие возможности программируемых логических интегральных схем типа FPGA.

3. Разработанные модель и алгоритм модулярного деления в формате СОК, являющийся основой аппаратной реализации, позволяющей расширить практическое применение СОК.

4. Развитие нового итерационного метода многоразрядного деления модулярных чисел, отличающегося от традиционных методов деления использованием в каждой итерации только операции сдвига и сложения (вычитания), чем и достигается высокое быстродействие операции деления.

5. Численный метод вычисления частного при модулярном делении числа на основе алгоритма приближенного вычисления позиционной характеристики модулярного кода, являющийся лучшим по вычислительной сложности.

6. Разработана модель отказоустойчивого модулярного нейрокомпьютера на основе искусственной нейронной сети и многофункциональная модель контроля и диагностики вычислительных потоков на базе приближенного метода вычисления позиционной характеристики, являющейся основной составляющей нейрокомпьютера. Предложенные модели расширят сферу применения СОК.

7. Комплекс программ моделирования основных блоков высокоскоростного отказоустойчивого модулярного нейрокомпьютера в базисе ПЛИС.

Практическое использование результатов работы. Результаты диссертации используются в учебном процессе СКФУ на кафедре высшей алгебры и геометрии в дисциплине «Теория чисел», «Защита информации в распределенных вычислительных сетях», «Интеллектуальные системы», «Компьютерные системы и сети ЭВМ», «Модулярные нейрокомпьютерные технологии», «Обработка информации в СОК», «Основы нечеткой логики и нейронных сетей», «Теория нейронных сетей». На кафедре ПМиММ в дисциплинах «Основы вычислительной техники», а также в опытно-конструкторских работах ООО «Центр информационных технологий».

Апробация работы. Основные результаты, положения и выводы диссертации были представлены на:

I-й международной конференции «Параллельная компьютерная алгебра и ее приложения в новых инфокоммуникационных системах» - 2014 (г.Ставрополь

– 2014г.). 3 международная конференция «Инфокоммуникационные технологии в науке, производстве и образовании». Инфоком (г.Ставрополь – 2008г.), 3 международной научно-практической конференции «Информационные системы, технологии и модели управления производством» - 2007 СтГАУ (г. Ставрополь – 2007) – АГРУС. Всероссийская конференция «Проблемы и перспективы развития инфокоммуникационных технологий в системах связи военного назначения» - 2008 г. VII Московском международном салоне инноваций и инвестиций «Инновационные технологии разработки нового класса модулярных нейрокомпьютеров цифровой обработки сигналов» Москва, ВВЦ, 2007 г. (серебряная медаль) XIII Московский международный салон изобретений и инновационных технологий «Архимед - 2010», «Модулярный нейрокомпьютер RISC структуры на базе ПЛИС серии Spartan – 3E» (золотая медаль). Россия, Москва, 2010 г.

Публикации по теме диссертации. Содержание диссертации опубликовано в 22 работах, среди которых имеются статьи в научных изданиях, входящих в перечень ВАК Минобрнауки и Scopus.

Личный вклад соискателя. Все выносимые на защиту результаты и выводы получены автором лично. Авторским вкладом является разработка моделей, методов и алгоритмов нейросетевой реализации. Разработка численных методов и моделей отказоустойчивых модулярных нейрокомпьютеров проведена автором, либо с его непосредственным участием. Разработан программный комплекс для моделирования модулярных нейросетевых структур.

Структура и объем диссертации. Работа состоит из введения, четырех глав, заключения, двух иллюстраций, трех приложений, списка сокращений и обозначений, а также списка литературы, содержащего 118 наименований. Основная часть работы содержит 179 страниц машинописного текста.

Краткое содержание работы.

Во введении обоснована актуальность темы диссертации, сформулированы цель и задачи работы, выбраны объект и предмет исследования, показана научная

новизна, практическая и теоретическая ценность полученных результатов, приведены основные положения, выносимые на защиту.

В первой главе представлен анализ методов и алгоритмов построения нейрокомпьютеров на основе системы остаточных классов с высоким уровнем отказоустойчивости.

Проблема высокой надежности не только передачи информации, но и ее обработки особенно актуальна в современных системах, работающих в реальном времени, где ошибки работы оборудования должны быть обнаружены и исправлены немедленно. В связи с этим наиболее перспективным путем решения рассматриваемой проблемы является придание вычислительным устройствам свойства устойчивости к отказам и сбоям в процессе функционирования [1, 2]. Принято считать вычислительную систему отказоустойчивой (fault-tolerant system), если при возникновении отказа она сохраняет свои функциональные возможности в полном (fail-save) или уменьшенном (fail-soft) объеме. При этом отказоустойчивость обеспечивается сочетанием избыточности системы и наличием механизма обнаружения ошибок, а также процедур для автоматического восстановления ее правильного функционирования [28]. Fail-save устойчивость к отказам характеризует способность вычислительной системы обеспечивать корректную работу, несмотря на возникновение сбоев, отказов, но с понижением качества, то есть находясь в состоянии постепенного снижения эффективности.

Разработка отказоустойчивых модулярных нейрокомпьютеров основана на методах проекций и расширении модулярного кода, которые связаны с вычислением значения числа согласно КТО или в ОПСС, что приводит к трудоемким вычислительным процедурам при обнаружении и локализации отказов или сбоев, которые основаны на использовании немодульных операций. К немодульным операциям относятся операции определения знаков чисел и переполнение динамического диапазона, сравнение, расширение, определение и локализация ошибочного модуля, деление и восстановление числа по остаткам.

В основе алгоритмов выполнения немодульных операций лежат методы вычисления позиционных характеристик ПХ, сложность которых непосредственно влияет на скорость выполнения немодульных операций в модулярной алгебре. Анализ известных ПХ показал, что общим недостатком для всех методов является их вычислительная сложность. В работе проведено развитие новой ПХ, на основе которой разработан универсальный алгоритм.

Применение данного алгоритма позволит эффективно решить проблему построения отказоустойчивых систем и служит мощным потенциалом для автоматического обнаружения и коррекции ошибок. В связи с этим, в данной главе проанализированы существующие принципы использования корректирующих свойств кодов СОК при обнаружении и коррекции ошибок на основе методов проекций и расширения.

Так как модулярная арифметика является параллельной структурой, желательно иметь и параллельную вычислительную базу. В качестве таковой в работе выбраны искусственные нейронные сети (ИНС). В диссертационной работе предложена интеграция СОК и ИНС при разработке базового элемента модулярных нейрокомпьютеров. В работе показана невозможность повысить отказоустойчивость модулярных нейрокомпьютеров традиционными методами без больших вычислительных затрат, поэтому предложен альтернативный вариант, позволяющий устранить необходимость большой вычислительной сложности.

Вторая глава посвящена разработке математического метода приближенного вычисления позиционной характеристики для исследования моделей вычисления немодульных процедур системы остаточных классов.

С целью повышения эффективности вычисления позиционной характеристики предлагается новый метод приближенного определения позиционной характеристики, который позволяет реализовать практически все немодульные процедуры модулярного кода.

Анализ немодульных операций показал, что их можно представить точно или приближенно, поэтому методы вычисления позиционных характеристик можно разделить на две группы:

- методы точного вычисления позиционных характеристик;
- методы приближенного вычисления позиционных характеристик.

Методы точного вычисления позиционных характеристик широко рассмотрены в литературе. В данной главе предлагается приближенный метод вычисления позиционной характеристики на основе КТО с дробными числами, которая позволяет существенно сократить аппаратные и временные затраты, обусловленные операциями, выполняемыми над позиционными кодами уменьшенной разрядности.

Разработанный и исследованный эффективный метод приближенного вычисления позиционной характеристики на основе использования относительных величин, представленных в виде периодических дробей обладает привлекательными свойствами: высокая скорость, малые аппаратурные затраты и универсальность. Основными немодульными операциями, которые допускают применение приближенного метода, являются: обнаружение и локализация ошибки числа, а также операции, в которых достаточно знать интервалы расположения числа (определение знака, сравнение чисел и переполнение динамического диапазона). Кроме того, приближенный метод успешно может быть использован в сочетании с точными методами для реализации некоторых других операций (округление, масштабирование и др.). Применение приближенного метода позволяет сократить временные и аппаратурные затраты на выполнение немодульных операций, что открывает хорошие перспективы для дальнейшего развития и применения теории модулярной арифметики на практике.

Проведенное математическое моделирование показало адекватность разработанной математической модели на основе данных вычислительного эксперимента.

Третья глава посвящена разработке методов и алгоритмов моделирования трудно выполнимой немодульной операции деления и масштабирования

модульных чисел. Так как известные методы модулярного деления сдерживают широкое применение СОК в различных приложениях, поэтому в данной главе проведены исследования по развитию эффективных методов деления, которые не привязаны к делителю, представляющего один или произведение модулей СОК.

В главе рассмотрены различные формы операций деления в СОК: деление с нулевым остатком, округление и масштабирование и основное деление. Разработана нейронная сеть для модулярного деления с нулевым остатком, которая используется для масштабирования модулярных чисел.

Большинство известных алгоритмов деления в СОК, основаны на использовании ОПСС, масштабировании, округлении, расширении и других операциях, которые являются медленными и требуют выполнения большого количества арифметических действий.

В данной главе разработан новый метод модулярного деления, который в каждой итерации содержит один сдвиг и одно вычитание. Это усовершенствование значительно уменьшает время каждой итерации алгоритма. Вычислительная сложность определяется сложением N-битного дробного числа, где N – количество разрядов дробного числа без учета операции сдвига.

В четвертой главе рассмотрены отказоустойчивые модели на основе методов проекций и расширения модулярных чисел.

Метод проекций позволяет установить в цифре по какому из модулей системы произошла ошибка, т.е. обеспечивает ее локализацию. Однако, как и сам алгоритм определения ошибочной цифры, так и последующая ее коррекция на основе метода проекций представляются малоэффективными и затратными с точки зрения практической реализации. А именно, вычисление каждой из проекций требует выполнения операции восстановления числа из модулярного представления в двоичное и последующих вычислений над числами большой разрядности, что вносит значительный вклад как в аппаратные затраты, так и в задержку работы устройства. Кроме того, такой метод подразумевает исправление лишь ошибочного остатка, а не всего числа, приводя к необходимости восстанавливать число с уже правильными значениями всех остатков. В связи с

этим возникает необходимость искать другие методы реализации механизма обнаружения и коррекции ошибок в СОК.

Одной из альтернатив методу проекций для обнаружения и коррекции ошибок в СОК является синдромное декодирование с вычислением так называемых синдромов ошибки по контрольным основаниям системы. В основе этого метода чаще всего лежит так называемая операция расширения системы оснований. Это немодульная операция, позволяющая по известным остаткам числа, соответствующим некоторым модулям СОК, определить значения остатков этого же числа для других оснований.

В главе предложена модель отказоустойчивого модулярного нейрокомпьютера с обнаружением, локализацией и коррекцией ошибок в СОК, состоящего из двух частей: модулярного нейрокомпьютера в традиционном исполнении и нейронной сети для обнаружения, локализации и коррекции ошибок в СОК, на основе синдрома ошибок.

Для достижения требуемого уровня отказоустойчивости нейрокомпьютера в общую схему введена нейронная сеть для обнаружения, локализации и коррекции ошибок, состоящая из преобразователей ПСС–СОК и вычислительных каналов по контрольным модулям p_{n+1} и p_{n+2}, блока расширения СОК, блока вычисления синдрома ошибок δ, блока памяти констант ошибок Δ_i, блока коррекции результата и блока, сигнализирующего состояние вычислительных каналов.

В заключении подведены итоги и обобщены результаты проведенных исследований.

В приложении приведен листинг программы на языке VHDL, результаты моделирования и тестирования модулярных нейросетевых узлов с использованием ресурсов ПЛИС FPGA.

Автор выражает искреннюю благодарность научному руководителю доктору технических наук профессору Калмыкову Игорю Анатольевичу, коллективу кафедры прикладной математики и математического моделирования, а также коллективу кафедры высшей алгебры и геометрии Северо-Кавказского

федерального университета за помощь, оказанную при написании диссертации, и критические замечания, высказанные при ее обсуждении.

1. Аналитический обзор моделей и методов создания вычислительных средств с высоким уровнем отказоустойчивости

1.1 Анализ моделей, методов и алгоритмов построения отказоустойчивых нейрокомпьютеров

Современный уровень развития вычислительных средств и широкое их внедрение во все области деятельности человека все более требует создания вычислительных средств с высоким уровнем отказоустойчивости. Высокий уровень отказоустойчивости необходим для всех видов вычислительных устройств, в том числе и для модулярных нейрокомпьютеров.

Одним из основных параметров при проектировании отказоустойчивого модулярного нейрокомпьютера является надежность его функционирования. Ведь, с одной стороны, постоянный рост требований к скоростным характеристикам вычислительных устройств приводит к необходимости организации параллельных вычислений, а с другой стороны, при этом увеличивается частота возникновения отказов, и возрастает время простоя процессоров, вызванное трудностью отыскания и устранения неисправности. Очевидно, что независимо от того, какие характеристики проявляет вычислительное устройство, единственная ошибка в любом из его блоков может отключить или повредить всю систему и в некоторых случаях привести к катастрофическим неисправностям [12,30,104].

Принято считать вычислительную систему отказоустойчивой (fault-tolerant system), если при возникновении отказа она сохраняет свои функциональные возможности в полном (fail-save) или уменьшенном (fail-soft) объеме. При этом отказоустойчивость обеспечивается сочетанием избыточности системы и наличием механизма обнаружения ошибок, а также процедур для автоматического восстановления ее правильного функционирования. Fail-save устойчивость к отказам характеризует способность вычислительной системы обеспечивать корректную работу, несмотря на возникновение отказа, но с

понижением качества, то есть находясь в состоянии постепенного снижения эффективности [31].

В модулярных нейрокомпьютерах отказоустойчивость обеспечивается корректирующими свойствами кодов СОК [104].

Избыточная модулярная арифметика или избыточная система остаточных классов (RRNS) обладает уникальными свойствами относительно обнаружения и коррекции ошибок [104].

RRNS привлекает многих исследователей в качестве основы для проектирования отказоустойчивых вычислительных структур, и интерес к ней в последнее десятилетие резко возрастает, что подтверждается большим числом публикаций.

В работах [2,32,70,104,110] проведен анализ свойств корректирующих кодов в СОК, основанный на оценке кодового расстояния Хэмминга. Причем в работах [104,110] используются методы коррекции, построенные на основе расширения системы рабочих оснований до системы оснований избыточной СОК. При этом анализ рассмотренных работ показывает, что все используемые в них методы связаны с вычислением значения числа согласно КТО или в ОПСС, что приводит к трудоемким вычислительным процедурам.

Полную отказоустойчивую систему, основанную на СОК, можно разделить, как показано на рисунке 1.1, на пять основных частей: входной блок, состоящий из прямого преобразователя, который разлагает взвешенные операции на арифметические остатки чисел по модулям $p_1, p_2,..., p_n, p_{n+1},..., p_{n+r}$, где $p_1, p_2,..., p_n$ - информационные модули, а $p_{n+1},..., p_{n+r}$ - контрольные, соответственно, $P_p = \prod_{i=1}^{n} p_i$ - рабочий диапазон, а $P_n = \prod_{i=1}^{n+2} p_i$ - полный диапазон, тогда некоторое число $X = (x_1, x_2,..., x_n, x_{n+1},..., x_{n+r})$, удовлетворяющее условию $x_i = X \bmod p_i$, где $i = 1,2,..., n+r$ будет представлено однозначно. Любую избыточную СОК можно характеризовать величиной $R = \frac{P_n}{P_p}$, называемой степенью избыточности представления чисел в этой системе. Именно наличие

определенной избыточности представления чисел в СОК позволяет обнаруживать и корректировать ошибки. Мультинейропроцессор, состоящий из вычислительных каналов по модулям p_i, выполняющий расчеты с остатками, выходной блок, состоящий из обратного преобразователя, который восстанавливает взвешенное число всех операций над вычетами и основных вычислительных процедур по обнаружению и коррекции отказоустойчивого нейрокомпьютера; блок контроля вычислительных каналов и блоки межмодулярных (немодульных) операций.

Реализация прямого преобразования и модульные операции над остатками глубоко исследованы в [104].

Разложение многоразрядных чисел на несколько вычетов по выбранным наборам модулей СОК позволяет уменьшить разрядность вычислительных каналов мультинейропроцессора и, таким образом, более экономно использовать аппаратные ресурсы, блоки 1, 2. Кроме того, ввиду отсутствия межразрядных переносов, арифметические вычисления над разными остатками выполняются параллельно, что значительно уменьшает время вычислений [2,89,91,38,145]. Таким образом, представление чисел в СОК обладает большим потенциалом для обработки многоразрядных чисел и широко применяется во многих областях, таких как шифрование данных, теоретико-числовых приложений обработки изображений, видеофильтрации и других.

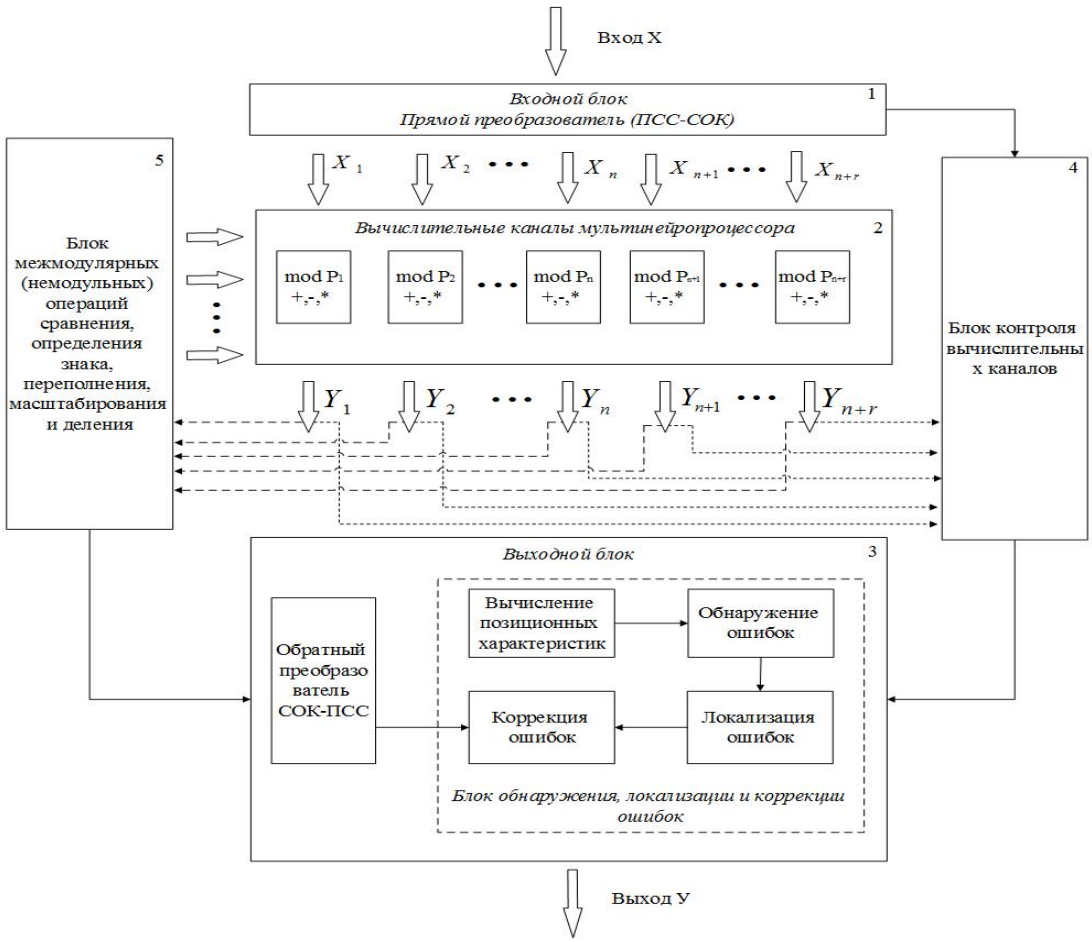


Рисунок 1.1 – Отказоустойчивая система, основанная на системе остаточных классов

Однако, производительность и вычислительная сложность отказоустойчивых систем в СОК во многом зависит от методов вычисления позиционных характеристик и немодульных операций на их основе блоки, 3, 4, 5. Тем временем, традиционные структуры обработки преобразователей, покрывающих множество различных немодульных операций, основанных на КТО и ОПСС, как правило, содержат много операторов, требующих длинных задержек и больших затрат ресурсов [96]. Это обстоятельство может привести к значительному увеличению вычислительной сложности, что безусловно является главным препятствием на пути к созданию высокопроизводительных отказоустойчивых систем. Поэтому возникает необходимость разработки высокоэффективных методов и алгоритмов вычисления основных немодульных операций (НО), позволяющих, с одной стороны, повысить быстродействие, а с другой снизить аппаратные ресурсы.

Однако, высокая эффективность вычислений НО, являющаяся ключевым компонентом СОК, все еще не достигнута из-за необходимости использования дорогостоящих и сложных операторов, которые требуют больших вычислительных ресурсов.

Метод, основанный на КТО, как правило, требует операций над большим модулем $P = p_1 \cdot p_2 \cdot ... \cdot p_n$, а, следовательно, неэффективен с точки зрения аппаратной реализации. Применение ОПСС требует только операций по модулям p_i или $\frac{P}{p_i}$, тем не менее, данный метод основан на последовательном алгоритме Гарнера, применение которого трудно достичь высокой производительности.

Далее в работе рассмотрим теоретическую основу реализации проблемных немодульных операций в СОК и построение на их основе функциональных блоков отказоустойчивых систем.

1.2 Анализ подходов использования корректирующих свойств кодов системы остаточных классов для придания вычислительным устройствам устойчивости к отказам и сбоям

Ранее были перечислены основные преимущества модулярного подхода, из которых можно заключить, что СОК обладает уникальными свойствами относительно обнаружения и коррекции ошибок. Исходя из этих свойств можно сделать вывод о том, что применение модулярной арифметики позволяет эффективно решить проблему построения отказоустойчивых систем и служит мощным инструментом для автоматического обнаружения и коррекции ошибок.

Первый шаг в процедуре обнаружения ошибки – это проверка, является ли полученный в результате вычислений вектор кодовым словом. Поскольку все члены, с которыми оперирует нейрокомпьютер, лежат в рабочем диапазоне $(0, P]$, где $P = \prod_{i=1}^{n} p_i$, то его называют легитимным диапазоном, а принадлежащие ему числа правильными. В то же время числа из диапазона $(0, P_r]$, где $P_r = \prod_{i=n+1}^{r} p_i$

являются неправильными, а сам избыточный диапазон называется не легитимным. Полный диапазон определяется как $\mathrm{P} = \prod_{i=1}^{n+r} p_i$ и состоит из двух частей – информационной $(1{:}n)$ и контрольной $(n+1{:}r)$:

$$X = (x_1,\ x_2,\ ...,\ x_{n+1} + ... + x_r). \qquad (1.1)$$

Обе части равнозначно участвуют во всех операциях внутри независимых вычислительных каналов, соответствующих каждому из модулей системы. Эта особенность кодов в модулярной арифметике используется для коррекции ошибок. Известно важное свойство СОК: если каждый контрольный модуль больше любого информационного, то минимальное расстояние кода на единицу больше числа контрольных модулей, т.е.

$$d_{\min} = r + 1 \qquad (1.2)$$

при условии упорядоченной СОК, т.е. $p_1 < p_2 < ... < p_n < p_{n+1} < ... < p_r$.

Если в результате какой-либо операции или при передаче числа оказалось, что получено число $X' > P$, то это говорит о том, что при проведении операции произошла ошибка. Таким образом, обнаружение одиночной ошибки основано на том факте, что любое искажение цифры по одному из оснований превращает все число в неправильное. Поэтому для обнаружения одиночной ошибки в числе X' его нужно сравнить с рабочим диапазоном P. При этом, если $X' \geq P$, значит, произошла ошибка по крайне мере в одном из каналов. Если $X' < P$, то либо ошибки нет, либо она носит более сложный характер.

Описанное выше свойство корректирующих кодов применяется при обнаружении, локализации и коррекции ошибок. Обнаружение ошибок может быть выполнено путем применения известных алгоритмов восстановления числа по его остаткам, например, метод ортогональных базисов на основе КТО или ОПСС (MRC) . Локализацию ошибочного разряда можно произвести на основе метода проекций [2]. Проекцией числа X по основанию p_i называется число X_i, полученное из X вычеркиванием соответствующего остатка x_i. Необходимо вычислить проекции числа X_i по каждому из модулей системы, и затем сравнить

их с величиной P . Если какая-либо из проекций X_i окажется меньше значения P, т.е. будет правильным числом, то ошибка произошла в соответствующем модуле. Таким образом, метод проекций позволяет установить в цифре по каждому из модулей системы, где произошла ошибка, т.е. обеспечивает ее локализацию.

Однако, как и алгоритмы определения ошибочной цифры, так и последующая ее коррекция на основе метода проекций представляется мало эффективным и затратным как с точки зрения аппаратных, так и временных характеристик. Одной из альтернатив методам определения ошибки и ее коррекции в СОК является синдромное дешифрование с вычислением так называемых синдромов ошибки по контрольным основаниям на основе метода расширения СОК.

Рассматриваемые ниже коды СОК могут быть использованы для корректирования ошибок, возникающих при передаче информации или выполнении арифметических операций.

Если в представление любого целого положительного числа X по модулям СОК $p_1, p_2, ..., p_n$ в виде набора остатков (вычетов) от деления $X = (x_1, x_2, ..., x_n)$ добавить разряд x_{n+1}, тогда получаем представление

$$X = (x_1, x_2, ..., x_{n+1}), \tag{1.3}$$

которое соответствует некоторому числу из диапазона $[0, R)$, где $R = P\, p_{n+1}$. При передаче информации или выполнении каких-либо операций будем пользоваться представлениями без добавленного разряда, которым соответствуют числа, заключенные в некоторую часть данного диапазона $[0, R)$; это дает возможность корректировать ошибки.

Если представление (1.3) удовлетворяет условию:

$$0 \le X < \frac{R}{p_{n+1}} = P_{n+1}, \tag{1.4}$$

то число X однозначно определяется n-разрядным представлением $(x_1, x_2, ..., x_{n+1})$. Поэтому разряд с основанием p_{n+1} можно считать избыточным. Оценим

корректирующие возможности кодов СОК с одним и двумя избыточными разрядами по основаниям p_{n+1} и p_{n+2}.

Введем понятие ошибки. Под одиночной ошибкой будем понимать искажение какой-либо одной цифры представления (1.3), причем искажение ограничивается лишь величиной основания. Под k- кратной ошибкой будем понимать искажение k цифр представления (1.3).

Пусть $X = (x_1, x_2, ..., x_n, x_{n+1})$ переданное сообщение. Для принятого сообщения введем обозначение $\overline{X} = (\overline{x_1}, \overline{x_2}, ..., \overline{x_n}, \overline{x_{n+1}})$. Сообщение назовем безошибочным, если $0 \le \overline{X} < P_{n+1}$. Если же $\overline{X} \ge P_{n+1}$, то соответствующее представление назовем ошибочным.

Если передано сообщение (1.3), то принятым может оказаться, вообще говоря, любое из R сообщений. При этом P_{n+1} из них будут приняты как безошибочные, и $R - P_{n+1}$ – как ошибочные, тогда число $R - P_{n+1}$ показывает суммарное количество обнаруживаемых возможных ошибок. Отношения $\frac{R - p_{n+1}}{R} = \frac{p_{n+1} - 1}{p_{n+1}}$ может служить мерой обнаруживающих способностей кода СОК. При упорядоченной СОК, когда основания расположены по возрастанию их величины, т.е. $p_1 < p_2 < ... < p_{n+1}$, тогда числа $P_1, P_2, ..., P_{n+1}$ будут удовлетворять неравенствам $P_1 > P_2 > ... > P_{n+1}$. В [2] показано, что наличие одного избыточного основания p_{n+1} является достаточным для обнаружения всех одиночных ошибок, т.е. если $\overline{\alpha_1} = \alpha_i$ при $i \ne k$ и $\overline{\alpha_1} \ne \alpha_i$ $(i = 1, 2, ..., n+1)$, то имеет место неравенство $\overline{X} \ge P_{n+1}$, которое обнаруживает ошибку. Необходимо отметить, что введение одного контрольного основания позволяет обнаруживать не только любую одиночную ошибку (в цифре по одному основанию) но и 95% двойных (в цифрах по двум основаниям) [2].

Процесс обнаружения ошибочности полученного представления может быть проведен в процессоре, работающем в СОК. Для этого достаточно перевести полученное представление в обобщенную позиционную систему счисления

(ОПСС) с теми же основаниями, что и в данной СОК. Если окажется, что старшая цифра ОПСС $x_{n+1}=0$, то представление безошибочное, если же $x_{n+1}\neq 0$, то полученное представление ошибочно. Если известен разряд СОК, в котором имеется ошибка, то ее легко исправить с помощью алгоритмов, изложенных в [70,71].

Таким образом, искажение цифры по одному какому-либо основанию переводит это число в неправильное и тем самым нелегитимный диапазон позволяет обнаруживать наличие ошибки. Введение только одного контрольного основания не позволяет в общем случае локализовать ошибочный разряд. Для корректной локализации ошибочного разряда и гарантированного исправления всех одиночных ошибок необходимо введение двух контрольных разрядов $p_{n+1},\ p_{n+2}$. Наличие любых избыточных разрядов является достаточным так же для обнаружения любой двоичной ошибки [2].

Пусть $X=(x_1,\ x_2,\ ...,\ x_n,\ x_{n+1},\ x_{n+2})$ – переданное сообщение или результат вычислений в СОК, а $\overline{X}=(\overline{x_1},\ \overline{x_2},\ ...,\ \overline{x_n},\ \overline{x_{n+1}},\ \overline{x_{n+2}})$ – принятое сообщение либо полученный результат вычислений в СОК, тогда одиночная ошибка может быть всегда исправлена, если числовые значения X передаваемых сообщений или точные результаты вычислений в СОК удовлетворяют неравенствам $0\le X<p_1,\ p_2,\ ...,\ p_n$, т.е. X входит в допустимую область. Рассмотрим всевозможные $(n+2)$ разрядные представления X_i, составленные из принятого сообщения вычеркиванием одной цифры (их будет $n+2$). Если в принятом сообщении нет ошибок, то для всех значений $i=1,\ 2,\ ...,\ n+2$ будет $\overline{X_i}=X_i=X<p_1,\ p_2,\ ...,\ p_n$. Если же в принятом сообщении содержится одиночная ошибка, то, за исключением одного, все $(n+1)$-разрядные представления $\overline{A_i}$ будут содержать одну ошибочную цифру, следовательно, для $(n+1)$-го значения i будет выполнено неравенство $\overline{X_i}>p_1,\ p_2,\ ...,\ p_n$, т.е. $\overline{X_i}$ входит в недопустимую область, и лишь для одного значения i, например, для $i=k$ будет $\overline{X_k}<p_1,\ p_2,\ ...,\ p_n$. Последнее соответствует случаю, когда

вычеркнута ошибочная цифра, но тогда, поскольку $X < p_1,\ p_2, ...,\ p_n$, получим $\overline{X_n} = X_n = X$.

Теперь уже можно найти истинное значение цифры x_k, для этого достаточной найти остаток от деления X_k на P_k. Далее вычисляем значения $(n+1)$ -разрядных представлений $\overline{X_n} = \overline{X} - r\,P_i$, где $\left[\frac{\overline{X}}{p_i}\right]$, т.е. для выполнения значений X достаточно из числа $\overline{X}$ вычитать число P_i до тех пор пока не получится число, лежащее в диапазоне $[0,\ P_i)$ [2].

При наличии двойной ошибки, по крайне мере, одно из $(n+1)$-разрядных представлений будет содержать одиночную ошибку и, следовательно, соответствующее ему числовое значение $\overline{X_i}$ выйдет из заданного диапазона $[0;\ p_1 \cdot p_2 \cdots p_n]$.

Таким образом, кодирование информации в СОК является устойчивым по отношению к возможным случайным искажениям и отказам, что позволяет при необходимости осуществлять коррекцию данных.

Описанное выше свойство корректирующих кодов применяется при обнаружении и коррекции ошибок на основе метода проекций, который позволяет установить в цифре по какому из модулей системы произошла ошибка, т.е. обеспечивает ее локализацию. Однако, как и сам алгоритм определения ошибочной цифры, так и последующая ее коррекция на основе метода проекций представляются малоэффективными и затратными с точки зрения практической реализации. А именно, вычисление каждой из проекций требует выполнения операции восстановления числа из модулярного представления в двоичное и последующих вычислений над числами большой разрядности, что вносит значительный вклад как в аппаратные затраты, так и в задержку работы устройства. Кроме того, такой метод подразумевает исправление лишь ошибочного остатка, а не всего числа, приводя к необходимости восстанавливать число с уже правильными значениями всех остатков. В связи с этим возникает

необходимость искать другие методы реализации механизма обнаружения и коррекции ошибок в СОК.

Одной из альтернатив методу проекций для обнаружения и коррекции ошибок в СОК является синдромное декодирование с вычислением так называемых синдромов ошибки по контрольным основаниям системы [104]. В основе этого метода чаще всего лежит так называемая операция расширения системы оснований. Это немодульная операция, позволяющая по известным остаткам числа, соответствующим некоторым модулям СОК, определить значения остатков этого же числа для других оснований. Обычно для реализации операции расширения системы оснований используют ОПСС, переход к которой носит итеративный характер и может привести к ухудшению быстродействия всего устройства.

Суть метода синдромного декодирования заключается в следующем. Пусть в результате вычислений в СОК получено некоторое число $X' = (x'_1, x'_2, ..., x'_n)$. Для определения правильности числа X' необходимо по известным остаткам $x'_1, x'_2, ..., x'_n$ определить значения его остатков по контрольным основаниям $x'_{n+1}, x'_{n+2}, ..., x'_{n+r}$. Затем следует сравнить значения $x'_{n+1}, x'_{n+2}, ..., x'_{n+r}$ с $x_{n+1}, x_{n+2}, ..., x_{n+r}$ – остатками по тем же модулям, но образованными уже в ходе вычислений над входными данными, аналогичных тем, в результате которых было получено само число X'. Сравнение остатков по контрольным основаниям можно осуществить их вычитанием:

$$\delta_{n+i} = \left| x'_{n+i} - x'_{n+i} \right|_{n+i}, \text{ где } i = 1, ..., r.$$

Числа δ_{n+i} называются синдромами ошибки, т.е. позволяют определить правильность результата вычислений в СОК.

Для иллюстрации данного механизма обнаружения и исправления ошибок с применением корректирующих кодов в модулярной арифметике, рассмотрим случай построения отказоустойчивой системы с исправлением одиночных ошибок. При этом код должен иметь минимальное расстояние $d_{\min} = 3$, которое достигается введением в СОК двух контрольных модулей p_{n+1} и p_{n+2}. Обычно

значения синдромов ошибки δ_{n+1} и δ_{n+2} используют как для обнаружения, так и для коррекции обнаруженных ошибок. Это становится возможным благодаря тому, что каждой паре значений $\{\delta_{n+1}, \delta_{n+2}\}$соответствует единственное значение вектора ошибки, а значит и корректирующего слова для ее исправления. Таким образом, коррекция ошибки при использовании данного метода осуществляется сложением числа, содержащего ошибку, и корректирующего слова, вычисленного по значениям синдромов δ_{n+1} и δ_{n+2}. Стоит также отметить, что при коррекции в данном случае складываются обычные двоичные числа, а поэтому нет необходимости в каких-либо дополнительных преобразованиях.

В связи с тем, что модулярная арифметика является параллельной структурой, желательно иметь и параллельную вычислительную базу. В качестве параллельной базы выбраны искусственные нейронные сети (ИНС).

Поэтому возникает необходимость разработки эффективных методов нейросетевой модулярной реализации основных вычислительных блоков отказоустойчивых систем, позволяющих избежать снижения быстродействия при их проектировании с одной стороны, а с другой - минимизировать аппаратную избыточность.

Актуальность данных исследований состоит в том, что реализация синдромного кодирования на основе совместного применения СОК и ИНС позволяет по новому взглянуть на существующие и новые принципы построения как отдельных отказоустойчивых вычислительных узлов, так и систем в целом, в том числе и систем повышенной надежности и производительности.

1.3 Интеграция системы статочных классов и нейронных сетей при разработке базового элемента модулярных нейрокомпьютеров

Нейрокомпьютеры и нейронные сети (НС) с самоорганизующейся архитектурой принципиально отличаются от ЭВМ с традиционной архитектурой массовым параллелизмом при обработке информации, высокой производительностью и быстрой самонастройкой на различные классы решаемых

задач. Одним из перспективных путей реализации НС и нейрокомпьютеров является использование в качестве элементной базы пороговых и полиномиальных преобразователей, а в качестве математического обеспечения – диофантовой арифметики и алгебры, теории чисел и рекурсивных функций, методов искусственного интеллекта и моделей параллельной обработки информации .

С появлением very large-scale integration (VLSI), Programmable logic device (PLD) стало возможным построение вычислительных устройств на одном vlsi ship, что открывает новые возможности применения вычислений кольца класса вычетов, являющееся своего рода конечным кольцом. Внутренняя реализация PLD включает в себя достаточно производительные сумматоры, умножители, LUT-up table, память и т.д., которые являются элементами искусственных нейронных сетей [77,78].

Новый вычислительный механизм нейроподобной сети для вычислительных систем конечного кольца, названный нейронной сетью конечного кольца (НСКК), может быть использован как базовый элемент, в котором суммируются произведения, определяя уровень функции активации нейрона.

Известно [78,80] несколько хорошо разработанных методов реализации арифметики конечного кольца: высокопроизводительный цифровой нейрон в СОК, основанный на ПЗУ [83]; модифицированные двоичные арифметические элементы [78]; параллельные и конвейерные структуры одноразрядного уровня и с применением распределенной арифметики [47,93].

В качестве базового элемента используется нейронная сеть конечного кольца (НСКК) [78].

Структура подсети показана на рисунке 1.2, где синаптические веса для z_i равны $\omega_i = \left|2^i\right|_p, \quad i = 0, 1, ..., \ n-1.$

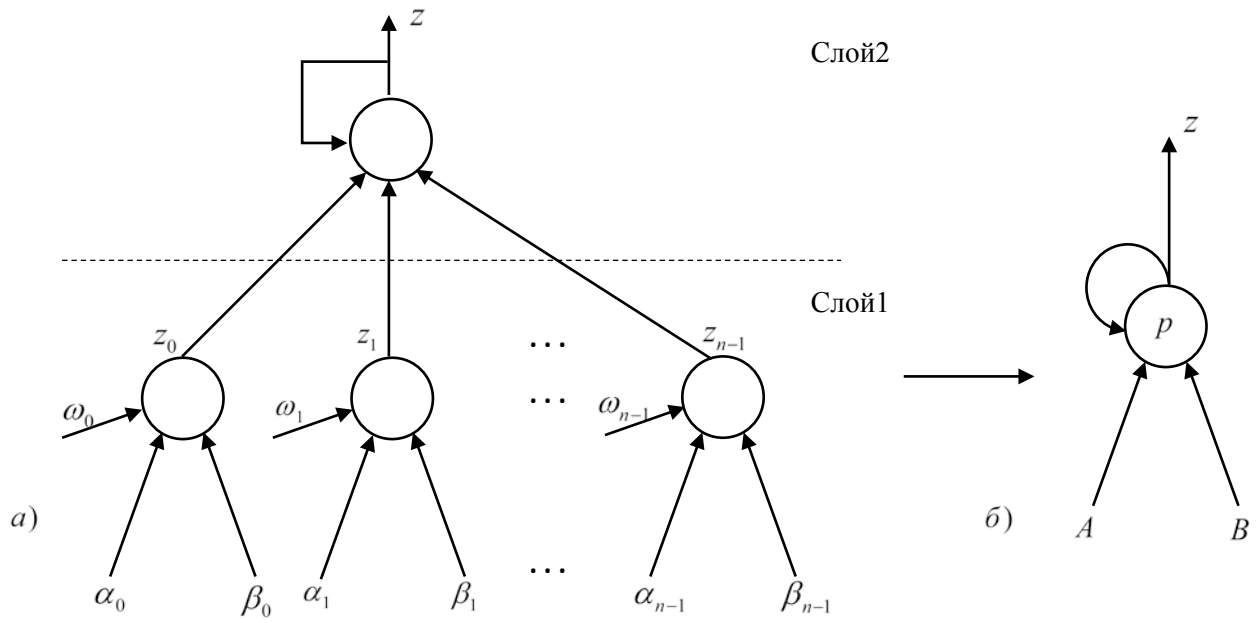


Рисунок. 1.2 - Структура подсети (а) и ее символическое отображение (б).

Из рисунка видно, что основными элементами НСКК являются сумматоры с умножителями, которые легко реализуются на FPGA.

Надежность нейронных сетей повышается за счет отказоустойчивости нейронов. Избыточность основана на однородности, как структуры, так и операций нейронов в нейронных сетях. НСКК обучена в процессе проектирования и ее синаптические веса вычислены и не изменяются.

Хранимые множественные наборы весов в нейроне и вычисляя заново выходы этого нейрона в резервных нейронах, можно обнаружить дефекты в нейроне и исправить ошибки.

Вследствие известных причин возникает принципиальная возможность использования преимуществ модулярной арифметики и ИНС при разработке различных функциональных устройств информационных и телекоммуникационных систем, включая проблему построения отказоустойчивых модулярных нейрокомпьютеров.

1.4 Постановка задачи исследования

Результаты теоретических и практических исследований отечественных и зарубежных ученых показали, что одним из перспективных направлений среди

современных подходов к созданию высокопроизводительных отказоустойчивых вычислительных средств для обработки данных большой размерности является использование системы остаточных классов и искусственных нейронных сетей. Состояние свойств модулярной арифметики и искусственных нейронных сетей позволяет проектировать модулярные нейрокомпьютеры принципиально нового класса, которые по сравнению с традиционными обеспечивают высокую надежность их функционирования.

Проблема высокой отказоустойчивости особенно актуальна в современных системах, реализованных на СБИС ПЛИС, сложность которых многократно возрастает, а вместе с ней возрастает и вероятность возникновения отказов в процессе эксплуатации устройств, что нежелательно для систем, у которых ошибки корректируются в реальном масштабе времени. Таким образом, высокая отказоустойчивость в этом случае должна добиться не столько совершенствованием элементной базы средств обработки данных, сколько за счет применения таких способов ее кодирования, которые были бы устойчивы по отношению к возможным случайным сбоям и отказам и позволяли бы при необходимости осуществить коррекцию данных.

В связи с этим наиболее перспективным путем решения рассматриваемой проблемы является придание модулярным нейрокомпьютерам устойчивости к отказам и сбоям в процессе функционирования за счет введения избыточности, которая приведет к росту разрядности модулей и, соответственно, вычислительных каналов. Это обстоятельство может привести к значительному ухудшению быстродействия всего нейрокомпьютера. Поэтому возникает необходимость разработки эффективных методов и алгоритмов аппаратной реализации основных узлов отказоустойчивых систем, позволяющих с одной стороны повысить устойчивость вычислительных средств к сбоям и отказам элементов системы, а с другой сводящих к снижению вычислительной сложности коррекции данных.

В соответствии с темой и целью диссертационной работы, а также основываясь на результатах исследований, проведенных в первой главе, сформулируем научную задачу проведения исследований.

Цель диссертационного исследования - повышение отказоустойчивости модулярного нейрокомпьютера.

Объект исследования – модулярные нейрокомпьютеры.

Предмет исследования – математические модели, методы и алгоритмы синтеза моделей отказоустойчивых модулярных нейрокомпьютеров для обработки данных большой разрядности.

Научная задача - исследований состоит в разработке новых математических моделей функциональных устройств модулярных нейрокомпьютеров, численных методов вычисления немодульных операций на основе приближенных вычислений приближенной характеристики, а также комплексы программ, применение которых позволит повысить отказоустойчивость.

Для решения поставленной общей научной задачи была произведена ее декомпозиция на ряд частных задач:

1. Анализ математических моделей и алгоритмов создания многоразрядных отказоустойчивых нейрокомпьютеров на основе интеграции модулярной арифметики и искусственных нейронных сетей, отличающихся от традиционных подходов низкой вычислительной сложностью.

2. Развитие нового метода и алгоритмов приближенного вычисления позиционных характеристик для исследования математических моделей таких немодульных операций системы остаточных классов как: определение знака числа; сравнение модулярных чисел; деления и переполнения динамического диапазона; локализация ошибочного разряда и коррекция ошибок.

3. Разработка методов и алгоритмов итерационного модулярного деления многоразрядных чисел в формате СОК и на основе КТО с дробными числами.

4. Разработка численного метода аппроксимации и уточнения частного при итерационном делении модулярных чисел на основе КТО с дробными числами с целью проведения вычислительного эксперимента.

5. Разработка метода, алгоритма и модели многоразрядного отказоустойчивого модулярного нейрокомпьютера с автоматической коррекцией ошибок в динамике вычислительного процесса на основе расширения системы оснований СОК с использованием синдрома ошибок.

6. Разработка многофункциональной модели для контроля и диагностики вычислительных каналов модулярного нейрокомпьютера на основе адаптации дробной части позиционной характеристики к корректирующим свойствам кодов СОК.

7. Разработка комплекса программ для экспериментального исследования на языке VHDL в среде ISE Design Suite 14.7 функциональных устройств модулярного нейрокомпьютера в базе программируемых логических интегральных схем.

1.5 Выводы по первой главе

1. Результаты исследований, проведенных в первой главе показали, что разработка перспективных средств обработки данных большой размерности связаны с применением модулярной арифметики, позволяющей разбивать исходные данные на малые форматы которые обрабатываются параллельно, и в качестве параллельной вычислительной базы использовать искусственные нейронные сети.

2. Учитывая свойства модулярной арифметики, широко развиваемой и применяемой в настоящее время, связанные с коррекцией ошибок и ее совместное применение с отказоустойчивыми моделями нейронной обработки позволяет решить проблему высокой надежности за счет придания модулярным нейрокомпьютером свойства устойчивости к отказам и сбоям в процессе их функционирования.

3. Анализ вычислений в пределах СОК показал, что используемые операции входят в две категории, которые можно описать различными путями. К первой категории относятся арифметические операции, составляющие группу легко выполнимыми, точными и обратимыми операциями, в то время как в другой группе операции являются, относительно сложными, не точными и не обратимыми. Вторая категория включает основное деление, масштабирование, определение переполнения и знака, сравнение чисел, коррекция ошибок и другие. Следовательно, для решения проблемы надежности вычислительных средств, необходимо исследовать виды моделей модулярных операций, которые будут использованы для построения отказоустойчивых структур.

4. Анализ подходов проектирования надежных модулярных нейрокомпьютеров показал невозможность обеспечения высокой их устойчивости без снижения производительности на основе существующих методов повышения отказоустойчивости, основанных на традиционных подходах, с одной стороны, и совместного использование системы остаточных классов, обладающей высоким потенциалом высокоскоростной и надежной обработки информации и нейронных сетей, с другой стороны, определяющим тему, цель и научную задачу данной диссертационной работы.

2. Разработка модели и метода вычисления проблемных операций на основе КТО с дробными числами

2.1 Разработка модели и метода вычисления немодульных операций на основе новой позиционной характеристики

Рассмотрим метод вычисления основных проблемных операций на основе межмодульных преобразований.

Структура метода основана на относительных значениях модулярного числа

$$\frac{X}{P} = \left| \sum_{i=1}^{n} \frac{\left| P_i^{-1} \right|_{p_i}}{p_i} x_i . \right|_1 \tag{2.1}$$

где X – исследуемое число, p_i - модули СОК, а $\left| P_i^{-1} \right|_{p_i}$ обозначает мультипликативную инверсию числа P_i^{-1} по модулю p_i,

$$P_i = \frac{P}{p_i} = p_1 p_2 \ldots p_{i-1} p_{i+1} \ldots p_n , \tag{2.2}$$

Введем обозначение

$$k_i = \frac{\left| P_i^{-1} \right|_{p_i}}{p_i} . \tag{2.3}$$

Тогда выражение (2.1) может быть переписано в компактной форме

$$\frac{X}{P} = \left| \sum_{i=1}^{n} k_i x_i . \right|_1 . \tag{2.4}$$

Очевидно, что величины $\frac{X}{P}, k_1, k_2, ..., k_n$ представляют собой некоторые дроби (бесконечные периодические), принадлежащие интервалу [0,1). Кроме того, величины k_i зависят только от модулей СОК, то есть являются константами СОК. Модель (2.4) представляет собой новый метод КТО с дробными числами, основанный на вычислении относительных значений модулярных чисел к полному диапазону СОК.

В таблице 2.1 приведены функциональные и адаптированные модели вычисления проблемных операций.

Таблица 2.1 Адаптация функциональных моделей к типу немодульных операций универсального алгоритма на основе КТО с дробными числами

№ п/п	Функциональная модель	Адаптированная модель
а	Определение знак числа в случае, если $p_1 = 2$	
	$Sign\left\|\frac{X}{P}\right\|_1 = \begin{cases} "+",\ если\ \left\|\frac{X}{P}\right\|_1 < \frac{1}{2}, \\ "-",\ если\ \left\|\frac{X}{P}\right\|_1 \geq \frac{1}{2} \end{cases}$	$\left[0, \frac{P}{2}\right)$ – положительное, $\left[\frac{P}{2}, P\right)$ – отрицательное, $\frac{P}{2}$ – принимается в качестве нуля, $\left\|\frac{X}{P}\right\|_1 < \frac{1}{2}$ – положительное, $\left\|\frac{X}{P}\right\|_1 \geq \frac{1}{2}$ – отрицательное.
б	Сравнение модулярных чисел X и Y	
	$\left\|\frac{X}{P} - \frac{Y}{P}\right\|_1 = \begin{cases} 0,\ если\ X = Y, \\ > 0,\ если\ X > Y, \\ < 0,\ если\ X < Y \end{cases}$	1. Определить знаки X и Y. 2. Если X и Y без знаков, то положительный знак разности относительных величин означает большее число. 3. Если X и Y имеют одинаковые знаки, то проверяем $\left\|\frac{X}{P} - \frac{Y}{P}\right\|_1$. 4. Если X и Y имеют разные знаки, то $\left\|\frac{X}{P} - \frac{Y}{P}\right\|_1 < 0{,}5$ при $X < Y$ и $0{,}5 \leq \left\|\frac{X}{P} - \frac{Y}{P}\right\|_1 < 1$ при $X \geq Y$.
в	Обнаружение ошибки и переполнения динамического диапазона	
	Утверждение $\left\|\frac{\overline{X}}{P_{изб.}}\right\|_1 < \left\|\frac{M}{P_{изб.}}\right\|_1$ тогда ошибки нет, где $\overline{X}$ – результат выполнения операции; $P_{изб.} = p_{n+1}\ p_{n+2}\ P$ –	$\overline{X} = X_{ОПСС} = x_0 + x_1 p_1 + x_2 p_1 p_2 + ... + x_n p_1 p_2$, если $x_n = 1$, тогда $\overline{X} > P = p_1 p_2 \cdots p_n$ – переполнение динамического диапазона

Продолжение таблицы 2.1

	избыточный диапазон при 2-х избыточных модулях p_{n+1} и p_{n+2}; $M = P = \prod_{i=1}^{n} p_i$ – рабочий диапазон	*Доказательство:* $\frac{\overline{X}}{P\,p_{n+1}p_{n+2}} < \frac{P}{P\,p_{n+1}p_{n+2}} = \frac{1}{p_{n+1}p_{n+2}}$, тогда $\overline{X}\,p_{n+1}p_{n+2} < P p_{n+1}p_{n+2}$ или $\overline{X} < p_1 p_2 \cdots p_n$, т.е. число $\overline{X}$ попало в допустимую область и поэтому является корректным. Если $\overline{X} > p_1 p_2 \cdots p_n$ – т.е. число $\overline{X}$ попало в недопустимую область, потому является ошибочным.
г	Локализация ошибочного разряда	
	Утверждение $\left\|\frac{\overline{X_i}}{P_i}\right\|_1 < \left\|\frac{M_i}{P_i}\right\|_1$ то в разряде i нет ошибки. $\left\|\frac{\overline{X_i}}{P_i}\right\|_1 \geq \left\|\frac{M_i}{P_i}\right\|_1$, то в разряде i есть ошибка, где: $\overline{X_i} = (x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n, x_{n+1}, x_{n+2}$ – проекция искажённого числа $\overline{X}$; $M_i = (m_1, m_2, ..., m_{i-1}, m_{i+1}, ..., m_n, m_{n+1}, m$ – проекция рабочего диапазона	Доказательство аналогично утверждению в. Если $\overline{X} = X_i = X < p_1 p_2 \cdots p_n$ – проекция входит в допустимую область, поэтому в i разряде ошибки нет. Если $\overline{X} > p_1 p_2 \cdots p_n$, то проекция входит в недопустимую область и в i разряде есть ошибка
д	Восстановление позиционного числа X по остаткам	
	$X \cong P\left\|k_i\, x_i\right\|_1$	$X = \tilde{X}P$, где $\frac{X}{P} = \left\|\sum_{i=1}^{n} k_i x_i\right\|$. Тогда $\tilde{X} = \left[\frac{X}{P}\right]_{2^{-N}} P$ – после несложных преобразований получим $N > \log_2 P \sum_{i=1}^{n}(p_i - 1)$ – условие точного восстановления позиционного числа по его остаткам

Таким образом, метод вычисления проблемных операций основан на использовании результатов межмодульных преобразований при котором численные значения модулярных чисел и их знак заключены в интервале [0,1). Предложенный метод и реализованный на его основе алгоритм можно считать универсальным, позволяющим вычислить все проблемные операции в СОК.

В процессе межмодульных преобразований используются бесконечные дроби, которые необходимо округлять. Однако погрешности округления, возникающие в ходе вычислений, могут приводить к некорректному выполнению проблемных операций. Поэтому естественным образом возникает вопрос: с какой степенью точности нужно производить вычисления, чтобы позиционное представление числа P определялось верно?

Обозначим через $F\left(\frac{X}{P}\right)$ бесконечную дробь в двоичной системе счисления, равную $\frac{X}{P}$. Конечную дробь, полученную из $F\left(\frac{X}{P}\right)$ путем отбрасывания всех бит, начиная с $(N-1)_{\text{-го}}$ бита после запятой, будем обозначать $\left[F\left(\frac{X}{P}\right)\right]_{2^{-N}}$. Например, если $F\left(\frac{X}{P}\right)=0{,}010101010...$, то $\left[F\left(\frac{X}{P}\right)\right]_{2^{-5}}=0{,}01010$. Очевидно, что выполняется соотношение

$$\left[F\left(\frac{X}{P}\right)\right]_{2^{-N}} < F\left(\frac{X}{P}\right) < \left[F\left(\frac{X}{P}\right)\right]_{2^{-N}} + 2^{-N},$$

описывающее положение точного значения $F\left(\frac{X}{P}\right)$ относительно округленной величины $\left[F\left(\frac{X}{P}\right)\right]_{2^{-N}}$. Оценим погрешность для конкретного вычисления немодульных процедур СОК.

Так как для вычислений по формуле (2.4) используются округленные константы $\left[F(k_i)\right]_{2^{-N}}$. Тогда на вопрос о корректном применении таких округленных величин для вычисления немодульных процедур докажем следующую теорему.

Теорема 2.1. Наименьшее значение N, при котором восстановление позиционной величины числа X по формуле

$$\frac{X}{P} = \left| \sum_{i=1}^{n} x_i \left[F(k_i) \right]_{2^{-N}} \right|_1, \tag{2.5}$$

где x_i остатки числа X по модулям p_i, $i = 1, 2, ..., n$, будет корректным, равно

$$N = \lceil \log_2 (P\,\mu) \rceil, \tag{2.6}$$

где $\mu = -n + \sum_{i=1}^{n} p_i$.

Доказательство. Так как

$$\left[F(k_i) \right]_{2^{-N}} < F(k_i) < \left[F(k_i) \right]_{2^{-N}} + 2^N,$$

для всех $i = 1, 2, ..., n$, то

$$\sum_{i=1}^{n} \left[F(k_i) \right]_{2^{-N}} x_i \le \sum_{i=1}^{n} F(k_i) x_i \le \sum_{i=1}^{n} \left[F(k_i) \right]_{2^{-N}} x_i + 2^N \sum_{i=1}^{n} \alpha_i . \tag{2.7}$$

Так как величина $\left| \sum_{i=1}^{n} k_i x_i \right|_1$ соответствует точному местоположению числа X на числовой оси, то для однозначного (точного) определения величины числа X необходимо таким образом подобрать параметр N, чтобы в произвольный интервал $\left[\sum_{i=1}^{n} \left[F(k_i) \right]_{2^{-N}} x_i ,\ \sum_{i=1}^{n} \left[F(k_i) \right]_{2^{-N}} x_i + 2^N \sum_{i=1}^{n} x_i \right]$ попадало лишь одно возможное значение из диапазона СОК. Это требование равносильно условию

$$2^N \sum_{i=1}^{n} x_i < \frac{1}{P}. \tag{2.8}$$

Если обозначить $\mu = -n + \sum_{i=1}^{n} p_i$, то наименьшее значение N, при котором возможно точное восстановление позиционной формы числа с использование формулы (2.4) определяется выражением $N = \lceil \log_2 (P\,\mu) \rceil$. Теорема доказана.

Далее в работе при выполнении немодульных операций будем использовать позиционную характеристику в виде округленно величины $\pi\left(\frac{X}{P}\right) = f\left[F\left(\frac{X}{P}\right) \right]_{2^{-N}}$.

2.2 Реализация метода на основе КТО с дробными числами для определения знака модулярного числа

Синтез выполняется на основе математической модели (2.1) Известно [2], что для определения знака числа используются номера интервалов, в которых расположено число, что позволяет получить оценку исследуемого числа по его величине с точностью до величины интервала. Числовой диапазон P может быть разбит на p_i интервалов величиной

$$\left[j\frac{P}{p_i}, (j+1)\frac{P}{p_i} \right], \ j = 1, 2, ..., p_i. \tag{2.9}$$

В качестве второго машинного нуля выбирается точка числового диапазона $\frac{p_{n+1}}{2}\frac{P}{p_n}$. Числа, расположенные в поддиапазонах $\left[0, \frac{p_{n+1}}{2}\frac{P}{2}\right)$ и $\left[\frac{p+1}{2}\frac{P}{p_n}, P\right)$ считаются числами разных знаков.

Если дано представление $(\alpha_1, \alpha_2, ..., \alpha_n)$, то для того чтобы установить знак числа, которое оно представляет, достаточно решить задачу о принадлежности этого числа к определенному интервалу. В случае если $p_i = 2$ достаточно решить задачу о принадлежности этого числа к первой $\left[0, \frac{P}{2}\right)$ или второй $\left[\frac{P}{2}, P\right)$ половине диапазона [0,P). Эта задача решается сравнением данного представления с представлением $\frac{P}{2}$, при условии, что $p_1 = 2$. Для определения знака числа в СОК по его дробной величине $F\left(\frac{X}{P}\right)$ необходимо выполнить проверку следующих соотношений. Если $0 < F\left(\frac{X}{P}\right) < \frac{1}{2}$, то число X положительное. Если $\frac{1}{2} \le F\left(\frac{X}{P}\right) < 1$, то число X отрицательное. Все известные методы реализуют данный алгоритм на основе использования абсолютных величин, здесь же мы предлагаем использовать относительные величины, что

существенно упрощает преобразование, сохраняя при этом основные функциональные возможности.

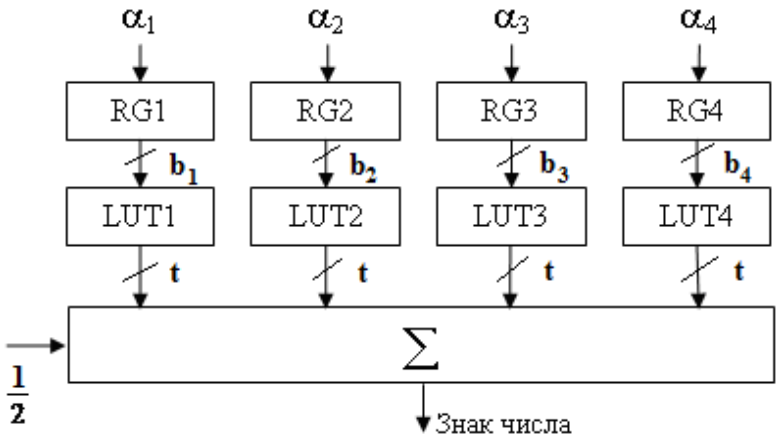


Рисунок 2.1 - Схема определения знака числа

На рисунке 2.1 приведена схема для определения знака модулярного числа, представленного по модулям $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$. Схема содержит входные регистры RG_i, $\forall i = [1...4]$ для временного хранения остатков чисел по соответствующим модулям, сравнительно небольшие просмотровые таблицы $\mathrm{LUT_i}$, $\forall i = [1...4]$ для хранения произведений $\left|\frac{P_i^{-1}}{p_i}\right|_{p_i} \cdot \alpha_i$ и параллельный сумматор. LUT-таблицы введены для ускорения операции умножения.

Процесс определения интервала сводится к выявлению принадлежности данного числа к одной из двух половин диапазона [0,*P*), к первой $\left[0, \frac{P}{2}\right)$ или второй $\left[\frac{P}{2}, P\right)$, где $\frac{P}{2}$ принимается в качестве нуля. Эта задача решается сравнением относительной величины $\frac{X}{P}$ с относительной величиной $\frac{K}{P} = \frac{P}{2P} = \frac{1}{2}$. Исходное число положительное, если $\frac{X}{P} < \frac{1}{2}$, и отрицательное, если $\frac{X}{P} \geq \frac{1}{2}$.

Схема работает следующим образом.

Код числа X, для которого необходимо определить интервал, что равносильно определению знака числа, поступает на входные регистры RG_i в

двоичном коде (каждый разряд СОК кодируется двоичным кодом с шириной шины $\lceil \log_2 p_i \rceil$ бит). Сигналы с выходов регистров поступают на входы просмотровых LUT-таблиц. В просмотровых таблицах хранятся произведения констант k_i и остатков α_i, то есть $\frac{\left|P_i^{-1}\right|_{p_i}}{p_i}\alpha_i$, представленные в естественной форме двоичной дроби в дополнительном коде с размером шины $\left\lceil \log_2(P(\sum_{i=1}^{n} p_i - n)) \right\rceil$ бит, где, p_i *i=1,2…n* - модули СОК, $P = p_1 p_2 ... p_n$ - диапазон СОК. В случае примера 2.1 формат двоичной дроби равен $\left\lceil \log_2 210(\sum_{i=1}^{4} p_i - 4) \right\rceil = 12$. Размер шины с таким форматом обеспечивает гарантированное и корректное определение знака и сравнение чисел в СОК.

Выходные сигналы просмотровых таблиц в дополнительном двоичном коде поступают на вход сумматора, в котором уже записана константа *0,5* во время начальной установки. (Дополнительный код используется для того, чтобы операцию вычитания заменить операцией сложения). Знак результата сложения определяет интервал: первый или второй, что соответственно определяет знак числа.

Пример 2.1. Определить знак числа $X = 200 = (0,2,0,4)$ при основаниях СОК $p_1 = 2,\ p_2 = 3,\ p_3 = 5,\ p_4 = 7$ и *P=210,* тогда $k_1 = 0,5, k_2 \approx 0,3333, k_3 = 0,6,$ $k_4 \approx 0,5714$.

Отсюда $\mu = -4 + 2 + 3 + 5 + 7 = 13;\ N = \lceil \log_2(210 \cdot 13) \rceil;\ k_1 = \frac{1}{2}; k_2 = \frac{1}{3}; k_3 = \frac{3}{5}$ и $k_4 = \frac{4}{7}.$

Дроби, соответствующие константам $k_i,\ i = 1,...,4,$ будут иметь вид

$[F(k_1)]_{2^{-12}} = 0,1000000000\ \ 00, [F(k_2)]_{2^{-12}} = 0,0101010101\ \ 01,$

$[F(k_3)]_{2^{-12}} = 0,1001100110\ \ 01,\ [F(k_4)]_{2^{-12}} = 0,1000100010\ \ 001000.$

Тогда

$$\left[F\left(\frac{X}{P}\right)\right]_{2^{-12}} = 0{,}100000000000 + 10 \cdot 0{,}010101010101 + 0 \cdot 0{,}100110011001$$
$$+ 100 \cdot 0{,}100100100100 = 0{,}111100111010.$$

Так как $0{,}1 < 0{,}111100111010 < 1$, то число $X = 200 = (0,2,0,4)$ отрицательное.

Определение знака осуществляется за $O(n)$ суммирований, где n - число оснований (модулей) СОК. Для уменьшения временной сложности до величины $O(\log n)$, суммирование можно реализовать по принципу дерева (рекурсивного сдваивания). Для сравнения, процедура определения знака с использованием алгоритма Танаки имеет временную сложность $O(n^2)$. Таким образом, предложенная схема определения знака числа улучшает аппаратную и временную сложность по сравнению с известной реализацией на базе ОПСС.

2.3 Аппаратная реализация метода и алгоритма сравнения модулярных чисел

Рассмотрим аппаратную реализацию метода сравнения чисел в СОК, основанную на алгоритме межразрядных преобразований.

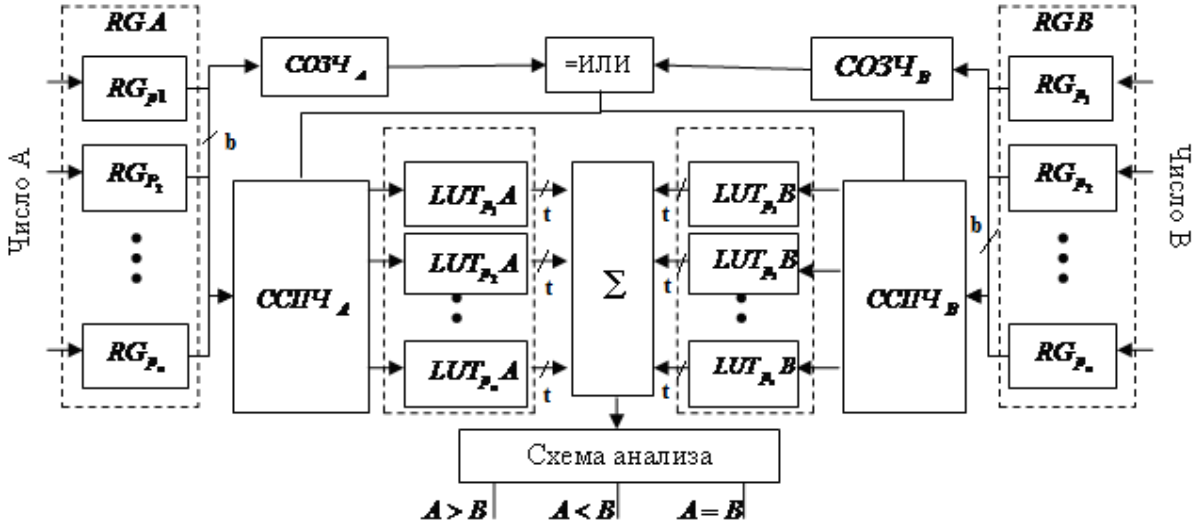


Рисунок 2.2 - Схема сравнения модулярных чисел

Схема модулярного сравнения чисел (рисунок 2.2) содержит входные регистры *RGA* и *RGB* для хранения сравниваемых чисел *A* и *B*, схемы определения знаков чисел *A* и *B* (*СОЗЧ*$_A$ и *СОЗЧ*$_B$), логического элемента «исключающее или», схемы сдвига полярности *ССПЧ*$_A$ и *ССПЧ*$_B$, просмотровые таблицы (память) $LUT_{p_i}A$ и $LUT_{p_i}B$, i=[1…n], сумматор для сложения выходных значений *LUT*-таблиц и схемы сравнения знака разности для формирования сигналов $X=Y$, $X>Y$ и $X<Y$.

На входные регистры *RGA* и *RGB* поступают исходные числа, представленные в СОК по модулям $p_1, p_2, ... p_n$ с размером шины b. С выходов регистров сигналы поступают на входы схем определения знаков числа, *СОЗЧ*$_A$ и *СОЗЧ*$_B$. Выходные сигналы схем определения знаков чисел X и Y поступают на вход элемента «исключающее или». При разных знаках чисел X и Y выходной сигнал элемента «исключающее или», которому приписано значение c_i, поступает на входы схем сдвига полярности *ССПЧ*$_A$ и *ССПЧ*$_B$. Выходные сигналы *ССПЧ*$_A$ и *ССПЧ*$_B$ являются адресными входами просмотровых таблиц $LUT_{p_i}A$ и $LUT_{p_i}B$. Элементы памяти *LUT* хранят константы $k_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} a_i \cdot$, $k_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} \beta_i \cdot$, где $a_i \equiv A \bmod p_i$, $\beta_i \equiv B \bmod p_i$, $\forall i \in [1,4]$. Сигналы с выхода просмотровых таблиц $LUT_{p_i}A$ и $LUT_{p_i}B$ поступают на вход сумматора, где производится взвешенное суммирование $\frac{A}{P}$ и $\frac{B}{P}$ с формированием знака полученной разности чисел X и Y, который анализируется в схеме анализа результата сравнения ($X=Y$, $X<Y$ и $X>Y$).

2.4 Экспериментальное исследование разработанных моделей с использованием математического моделирования в среде пакета WebPack ISE

При моделировании схемы сравнения в качестве алгоритмов восстановления взвешенного значения чисел использовались: КТОк – Китайская

Теорема об Остатках с использованием ортогональных базисов, ОПСС и КТОд - Китайская Теорема об Остатках с дробными величинами.

В качестве платформы для моделирования была выбрана плата Xilinx Kintex 7 XC7K70T (таблица 2.2) на базе ISE Design Suite 4.7 WebPack. Основными критериями оценки алгоритмов стали занимая алгоритмом площадь устройства (Device Utilization) и итоговая максимальная задержка алгоритма. Площадь будем оценивать, основываясь на количестве использованных блоков *Slice*, которые являются базовой вычислительной единицей устройства для Xilinx серии 7. Кроме того был рассмотрены варианты реализации алгоритмов с учетом и без учета использования блоков *DSP*48*E*1, предназначенных для оптимизации алгоритмов специальной структуры.

Таблица 2.2 Комплектация платы Kintex 7 XC7K70T.

Slice Logic Items	**Available**
Number of Slice Registers	82000
Number of Slice LUTs	41000
Number of Slices	10250
Number of IOBs	300
Number of DSP48E1s	240

Результаты моделирования описанных алгоритмов приведены в таблицах 2.3 и 2.4 и на рисунках 2.3 и 2.4. Сравнение известных классических методов восстановления, основанных на КТОк и ОПСС, позволяет сказать, что меньшую площадь требует алгоритм ОПСС, а меньшую задержку – КТОк. Метод перевода с использованием ОПСС требует большого количества малоразрядных операций. Длинный путь сигнала от входов к выходу, связанный с большим количеством операций, делает данный метод самым медленным. При сравнении предлагаемого метода КТОд с известными, основанными на КТОк и ОПСС, видно, что как наименьшую площадь платы, так и наименьшую задержку требует метод КТОд, основанный на использовании дробных величин.

Таблица 2.3 Ресурсы платы FPGA, требуемы для реализации алгоритмов при фиксированном модулей $n = 4$.

Набор модулей	**19, 23, 29, 31**		**1009, 1013, 1019, 1021**		**32713, 32717, 32719, 32749**		**1048549, 1048559, 1048571, 1048573**	
Разрядность диапазона	19		40		60		80	
Разрядность модулей	5		10		15		20	
	Slices	*DSP48E1*	*Slices*	*DSP48E1*	*Slices*	*DSP48E1*	*Slices*	*DSP48E1*
КТОд	13	7	77	29	118	42	236	69
КТОк	92	4	352	16	706	20	1133	32
ОПСС	54	9	224	13	576	17	1108	24

Таблица 2.4 Ресурсы платы FPGA, требуемы для реализации выбранных алгоритмов при различном количестве 6-битных модулей.

Набор модулей	**47, 53, 59, 61**		**41, 43, 47, 53, 59, 61**		**31, 37, 41, 43, 47, 53, 59, 61**	
Разрядность диапазона	24		34		45	
Количество модулей	4		6		8	
	Slices	*DSP48E1*	*Slices*	*DSP48E1*	*Slices*	*DSP48E1*
КТОд	17	10	46	30	129	43
КТОк	117	4	213	12	285	32
ОПСС	75	11	164	16	191	45

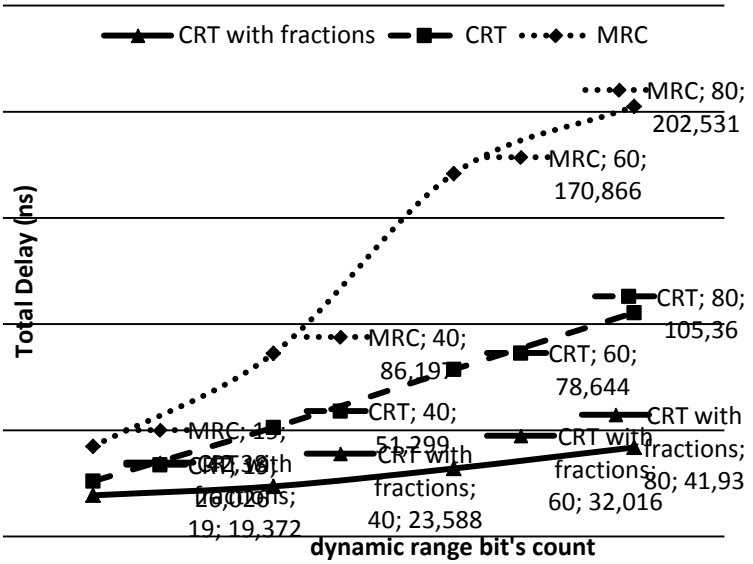


Рисунок 2.3 - Зависимость максимальной задержки алгоритмов от разрядности модулей при количестве модулей равном 4 с использованием блоков DSP48E1.

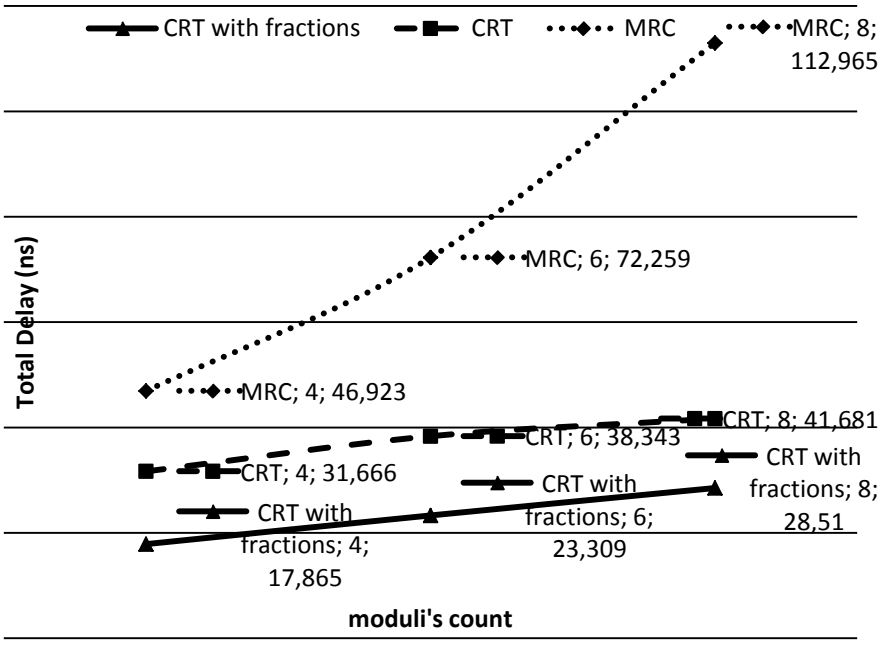


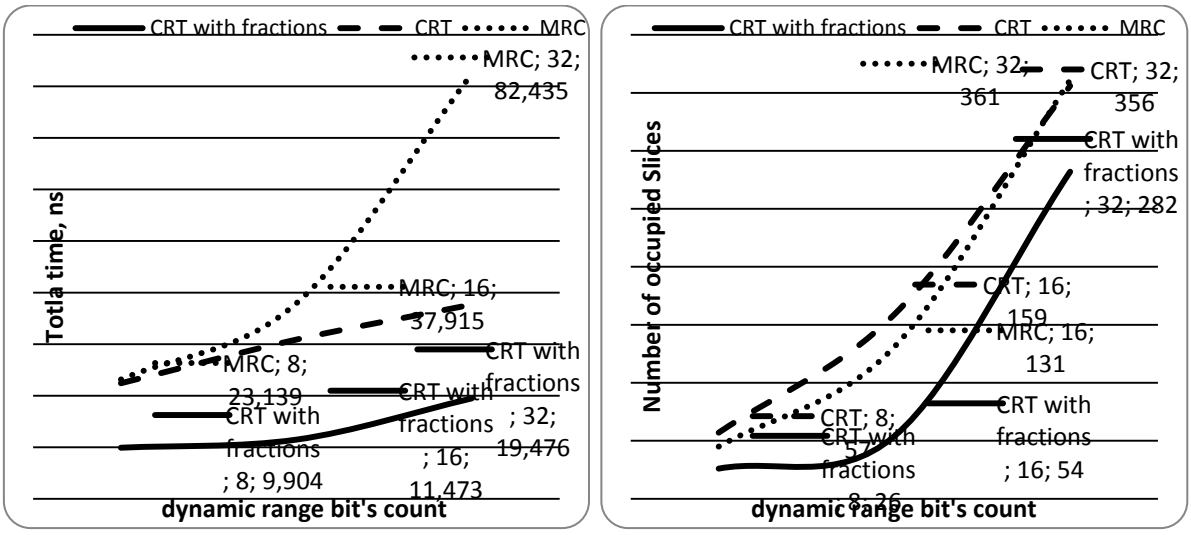
Рисунок 2.4 - Зависимость максимальной задержки алгоритмов от количества модулей при разрядности модулей равном 6 бит с использованием блоков DSP48E1.

Рассмотрим далее задачу минимально избыточного покрытия диапазонов, равных стандартным диапазонам арифметических устройств, с использованием максимум 8-битных чисел. В таблице 2.5 приведены различные варианты наборов модулей СОК, позволяющие перекрыть диапазоны 8, 16 и 32 бита.

Таблица 2.5 Наборы модулей СОК, минимально перекрывающие машинные диапазоны.

Диапазон	*Набор модулей*	*Диапазон СОК*	*Разрядность дробных величин*
$2^8 = 256$	7, 37	259	14
$2^{16} = 65536$	11, 59, 101	65549	24
$2^{32} = 4294967296$	19, 67, 89, 167, 227	4294975973	42

На рисунке 2.5 отражены максимальная задержка и занимаемая площадь всех рассмотренных алгоритмов при использовании наборов модулей из таблицы 2.5. Отметим, что приближенный метод показывает наилучшие результаты, несмотря на то, что приводит к вычислениям, превосходящим разрядность диапазона. Например, для диапазона 32 бита точное вычисление позиционного числа потребует расчетов с числами до 42 бит.

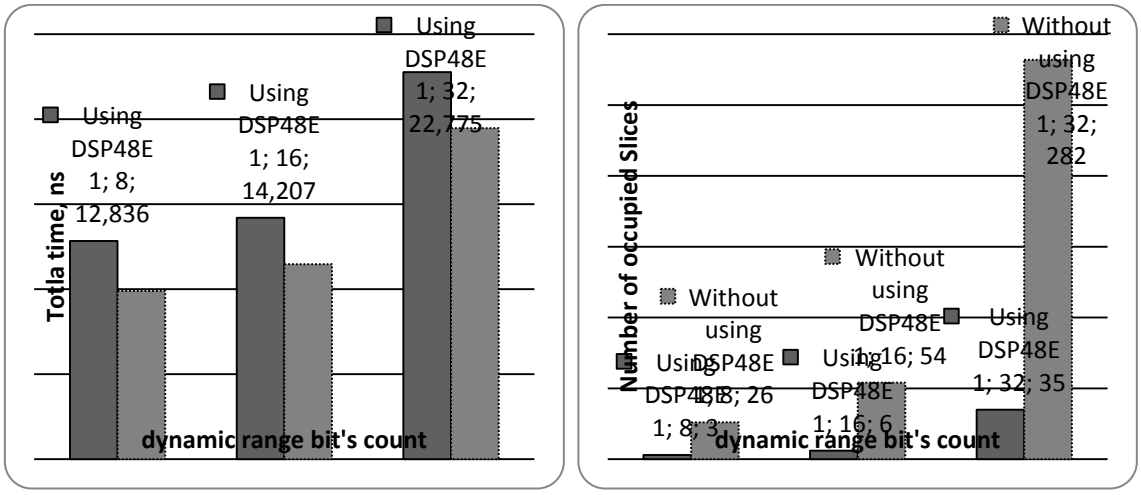


a b

Рисунок 2.5 - Максимальная задержка (a) и использование ресурсов платы (b) исследуемых алгоритмов для наборов модулей из таблицы 2.5 без использования блоков DSP48E1.

Использование блоков цифровой обработки сигналов DSP48E1, включенных в использованную плату, позволяет существенно снизить занимаемую алгоритмами площадь схемы (рисунок 2.6), однако при этом

снижается скорость работы алгоритма (рисунок 2.6 а). В различных практических ситуациях можно использовать данный факт для улучшения требуемой характеристики.



a b

Рисунок 2.6 - Максимальная задержка (a) и использование ресурсов платы (b) приближенного метода для наборов модулей из таблицы 2.4 с и без использования блоков DSP48E1.

Полученные результаты подтверждают теоретические выводы и доказывают высокую эффективность приближенного метода с точки зрения скорости работы алгоритма, что достигается благодаря отсутствию вычислений остатков от деления и малому числу операций. Кроме того, на всех рассмотренных наборах модулей данный метод требует наименьшее количество ресурсов в сравнении с двумя другими.

2.5 Выводы по второй главе

1. Исследования КТО с дробными числами позволили разработать эффективный метод вычисления позиционной характеристики, имеющий линейную алгоритмическую сложность, на базе которой предложены новые алгоритмы выполнения немодульных операций.

2. Проведенный в работе анализ влияния позиционных характеристик на вычислительную сложность определения немодульных операций показал, что точные методы вычисления позиционных характеристик (метод ортогональных

базисов, метод интервальных оценок, метод с использованием коэффициентов ОПСС и др.) отличаются высокой временной или аппаратной сложностью их реализации.

Альтернативой точных методов является метод КТО с дробными числами, вычисляющий приближенную позиционную характеристику модулярных чисел с меньшими временными и алгоритмическими затратами, который обеспечивает простоту определения немодульных операций. Показана его универсальность и низкая вычислительная сложность.

3. Разработаны модели сравнения и определения знака модулярных чисел, основанные на использовании приближенной позиционной характеристики и приведены доказательства корректности ее применения при вычислении немодульных операций.

4. Сравнительная оценка применения приближенного метода с точным методом на основе ОПСС, который считается наиболее быстродействующим, показал высокую эффективность приближенного метода используемого для сравнения модулярных чисел. Так, для приведенного в работе примера при десятичной арифметике разрядность чисел, используемых при вычислении ПУ увеличилась в 2 раза, скорость вычисления увеличилась в 1,5 раза, а при двоичных вычислениях разрядность уменьшилась в 1,8 раза, а скорость увеличилась в 2 раза.

Применение метода приближенного вычисления позволяет заменить операторы вычисления коэффициентов ОПСС недорогими и простыми разрядными операциями, что приводит к значительному снижению ресурсов, а также перспективному повышению производительности вычисления немодульных процедур.

5. Впервые экспериментально исследованы предложенные аппаратные реализации схем определения знака и сравнения модулярных чисел с использованием математического моделирования в среде пакета Web Pack ISE на базе FPGA. Результаты моделирования показали эффективность приближенного метода по сравнению с известными, основанными на классической КТО и ОПСС

по использованию наименьшей площади платы и наименьшей задержке сигнала. Так при разрядности 80 бит выигрыш в Slices и задержке сигнала более чем в 4 раза.

3. Разработка методов математического моделирования исследования модулярного деления чисел в базисе искусственных нейронных сетей

3.1 Разработка нейронной сети для модулярного деления с нулевым остатком

Во многих алгоритмах для которых СОК особенно хороша, самой распространенной операцией является вычисление промежуточных результатов, состоящей из последовательностей операций умножения и сложения. Это неизбежно может привести к увеличению промежуточных результатов и если не принять надлежащие меры может наступить переполнение. Во избежание переполнения надо промасштабировать (уменьшить) значения операндов. Промасштабированные величины затем используются в следующих итерациях. Это означает, что операция масштабирования должна применяться к данным с использованием заранее заданной константы, которая округляется до ближайшего целого. Все эти операции связаны с операцией деления.

Деление в модулярной арифметике относится к немодульным операциям и является одной из важнейших операций в модулярной компьютерной арифметике, так как лежит в основе многих других операций и входит в состав операций вычислительных алгоритмов.

Операцию деления в СОК можно отнести к одной из трех различных форм [110]:

1. Деление с нулевым остатком.
2. Округление и масштабирование.
3. Основное деление.

На рисунке 3.1 представлена схема нейронной сети для деления чисел, представленных в СОК.

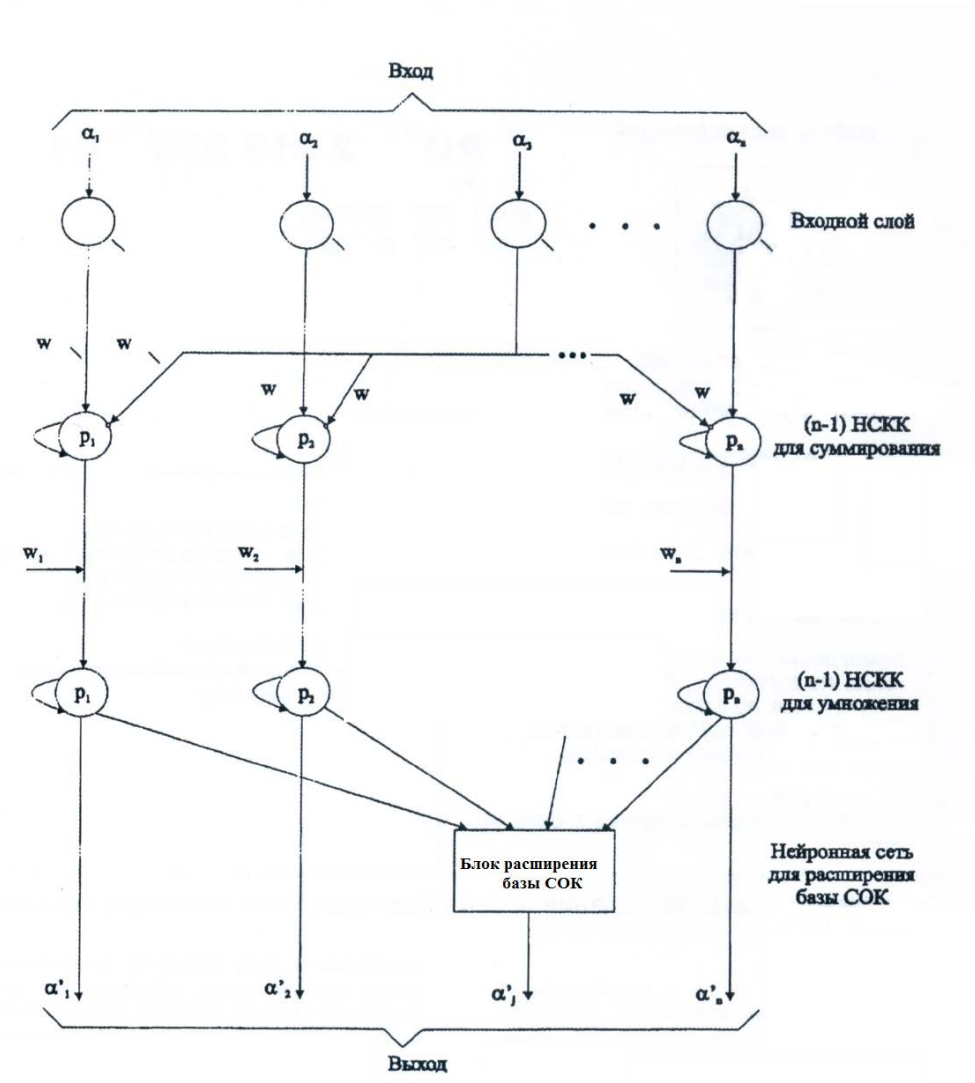


Рисунок 3.1 - Нейронная сеть для модулярного деления с нулевым остатком

Нейронная сеть для деления чисел, представленных в системе остаточных классов, позволяет выполнить операцию деления исходного числа $A=(\alpha_1,\alpha_2,...,\alpha_n)$ на делитель, равный одному из модулей СОК $p_j=5$.

Остатки делимого числа $A=(\alpha_1,\alpha_2,...,\alpha_n)$ по системе модулей $p_1, p_2,..., p_n$ поступают на вход нейронов входного слоя. С выходов нейронов входного слоя значения остатков по модулям $p_1, p_2, p_{j-1}p_{j+1},..., p_n$ поступают на входы (n-1) НСКК для суммирования с весовыми коэффициентами w=1 для реализации вычислительной модели $\left|\alpha_i+\left|p_i-\alpha_j\right|_{p_i}\right|_{p_i}$, где:

α_i - остатки числа A в СОК по p_i модулям; α_j - остатки делителя по модулю p_j для i=1, 2,..., n, при этом i не включает j.

Выходные значения (n-1) НСКК суммирования поступают на входы (n-1) НСКК умножения, у которой весовые коэффициенты w_i равны $\left|\frac{1}{p_j}\right|_{p_i}$. На выходах НСКК, кроме модуля делителя p_j.формируется результат частного без разряда делителя в виде $[\alpha_1^{'},\alpha_2^{'},...,\alpha_{j-1}^{'}\alpha_{j+1}^{'},...,\alpha_n^{'})$, которые являются выходами нейронной сети.

Неизвестная цифра $\alpha_j^{'}$ по модулю делителя определяется путем расширения базы СОК по известным остаткам $\alpha_i^{'}$, которые поступают на вход нейронной сети для расширения кортежа числовой системы вычетов. Вычисленный остаток $\alpha_j^{'}$ является разрядом частного по модулю p_j и поступает на выход сети, восстанавливая кортеж частного $[\alpha_1,\alpha_2^{'},...,\alpha_j^{'},...,\alpha_n^{'})$.

Время деления числа определяется тремя циклами синхронизации.

Аналогичным образом реализуется нейронная сеть для деления положительных чисел несколькими модулями. К недостаткам данного метода можно отнести привязанность делителя к основанию СОК или их произведению.

При масштабировании на произведение модулей процесс превращается в итерационный, что снижает скорость выполнения операции масштабирования. Для исключения итераций в данной работе в случае применения в качестве масштабного коэффициента подмножества

$$\prod_{j=1}^{n} p_0 = P_j \approx \sqrt{P}$$

предлагается метод масштабирования чисел СОК на основе функции ядра [91, 130], аппаратная реализация в каждом вычислительном канале определяется логической глубиной в одну НСКК. Время масштабирования модулярного числа на произведение модулей в предложенной схеме определяется временем масштабирования на один модуль.

3.2 Разработка параллельного алгоритма деления модулярных чисел в формате СОК

3.2.1 Алгоритм модулярного деления чисел в формате СОК

Проблема деления в общем виде в СОК привлекает внимание многих исследователей для разработки высокопроизводительных мультинейропроцессоров. Большинство известных алгоритмов деления в СОК, основаны на использовании ОПСС, масштабировании, округлении, расширении и других операциях, которые являются медленными и требуют выполнения большого количества арифметических действий. В связи с этим возникает необходимость упростить структуру вычислений при делении. Одно из направлений упрощения структуры устройства деления состоит в том, что делимое, делитель и остаток представлены только в формате СОК.

Рассмотрен параллельный алгоритм деления в СОК, преимущество которого состоит в том, что делимое, делитель и все промежуточные вычисления выполняются в формате СОК [116]. Возможность такой реализации основана на введении вспомогательного набора модулей СОК. Сложность операции деления определяется на преобразованиях между основной и вспомогательной СОК, которые выполняются в формате СОК. Последовательность действий алгоритма изложена ниже.

1. Пусть $p_1, p_2, ..., p_m$ есть набор модулей, $(a_1, a_2, ..., a_m)$ - делимое и $(b_1, b_2, ..., b_m)$ - делитель.

2. Определим вспомогательную систему оснований $(p_1', p_2', ..., p_m')$, где $НОД(p_i, p_j') = 1$ для $i, j = 1, 2, ..., m$.

3. Вычислим $(q_1, q_2, ..., q_m)$, где $q_i = p_1' \cdot p_2' \cdot ... \cdot p_m' \bmod p_i$.

4. Переведем $(a_1, a_2, ..., a_m)$ по основаниям $p_1, p_2, ..., p_m$ в $(a_1', a_2', ..., a_m')$ по основаниям $p_1', p_2', ..., p_m'$ методом расширения базы СОК.

5. Переведем $(b_1,b_2,...,b_m)$ по основаниям $p_1,p_2,...,p_m$ в $(b_1',b_2',...,b_m')$ по основаниям $p_1',p_2',...,p_m'$ методом расширения базы СОК.

6. Вычисляем $\left(b_1^{-1},b_2^{-1},...,b_m^{-1}\right)$ по основаниям $p_1,p_2,...,p_m$, $\left(b_1'^{-1},b_2'^{-1},...,b_m'^{-1}\right)$ по основаниям $p_1',p_2',...,p_m'$ и $\left(q_1^{-1},q_2^{-1},...,q_m^{-1}\right)$ по основаниям $p_1,p_2,...,p_m$.

7. Вычисляем $(c_1',c_2',...,c_m')=(a_1',a_2',...,a_m')\cdot\left(b_1'^{-1},b_2'^{-1},...,b_m'^{-1}\right)$.

8. Переведем $\left(b_1'^{-1},b_2'^{-1},...,b_m'^{-1}\right)$ по основаниям $p_1',p_2',...,p_m'$ в $\left(b_1''^{-1},b_2''^{-1},...,b_m''^{-1}\right)$ по основаниям $p_1,p_2,...,p_m$ и $(c_1',c_2',...,c_m')$ по основаниям $p_1',p_2',...,p_m'$ в $(c_1,c_2,...,c_m)$ по основаниям $p_1,p_2,...,p_m$ методом расширения базы СОК.

9. Вычисляем

$$(a_1,a_2,...,a_m)=\left((a_1,a_2,...,a_m)\cdot\left(b_1''^{-1},b_2''^{-1},...,b_m''^{-1}\right)-(c_1,c_2,...,c_m)\right)\cdot\left(q_1^{-1},q_2^{-1},...,q_m^{-1}\right).$$

10. Вычисляем

$$(b_1,b_2,...,b_m)=\left((b_1,b_2,...,b_m)\cdot\left(b_1''^{-1},b_2''^{-1},...,b_m''^{-1}\right)-(c_1,c_2,...,c_m)\right)\cdot\left(q_1^{-1},q_2^{-1},...,q_m^{-1}\right).$$

11. Если $(b_1,b_2,...,b_m)=(1,1,...,1)$, то $(a_1,a_2,...,a_m)$ есть частное; иначе - перейти к шагу 4.

3.2.2 Численный метод модулярного деления с использованием вспомогательных модулей СОК.

Возьмем исходную систему оснований $(p_1,p_2,p_3)=(7,11,13)$ и вспомогательную систему оснований $(p_1',p_2',p_3')=(5,17,19)$. Для делимого $a=212$, равного в СОК $(2,3,4)_{(7,11,13)}$ и делителя $b=16$, равного в СОК $(2,5,3)_{(7,11,13)}$ частное $z=\left\lfloor\frac{212}{16}\right\rfloor=13$, а в СОК равно $(6,2,0)_{(7,11,13)}$. Операция деления выполняется следующим образом.

1. Вычисляем $(q_1,q_2,q_3)_{(p_1,p_2,p_3)}=(5,9,3)_{(7,11,13)}$ и

$\left(q_1^{-1},q_2^{-1},q_3^{-1}\right)_{(p_1,p_2,p_3)}=(3,5,9)_{(7,11,13)}$.

2. Переводим $(a_1, a_2, a_3)_{(p_1, p_2, p_3)} = (2,3,4)_{(7,11,13)}$ в $(a_1', a_2', a_3')_{(p_1', p_2', p_3')} = (2,8,3)_{(5,17,19)}$.

3. Переводим $(b_1, b_2, b_3)_{(p_1, p_2, p_3)} = (2,5,3)_{(7,11,13)}$ в

$(b_1', b_2', b_3')_{(p_1', p_2', p_3')} = (1,16,16)_{(5,17,19)}$.

4. Переводим $(b_1', b_2', b_3')_{(p_1', p_2', p_3')} = (1,16,16)_{(5,17,19)}$ в

$(b_1'^{-1}, b_2'^{-1}, b_3'^{-1})_{(p_1', p_2', p_3')} = (1,16,6)_{(5,17,19)}$.

5. Вычисляем

$$(c_1', c_2', c_3')_{(p_1', p_2', p_3')} = (a_1', a_2', a_3')_{(p_1', p_2', p_3')} \cdot (b_1'^{-1}, b_2'^{-1}, b_3'^{-1})_{(p_1', p_2', p_3')} =$$

$$= (2,8,3) \cdot (1,16,6) = (2,9,18)_{(5,17,19)}.$$

6. Переводим $(b_1'^{-1}, b_2'^{-1}, b_3'^{-1})_{(p_1', p_2', p_3')} = (1,16,6)_{(5,17,19)}$ в

$(b_1''^{-1}, b_2''^{-1}, b_3''^{-1})_{(p_1, p_2, p_3)} = (3,2,10)_{(7,11,13)}$.

7. Переводим $(c_1', c_2', c_3')_{(p_1', p_2', p_3')} = (2,9,18)_{(5,17,19)}$ в

$(c_1, c_2, c_3)_{(p_1, p_2, p_3)} = (4,10,1)_{(7,11,13)}$.

8. Вычисляем $(a_1, a_2, a_3) = ((a_1, a_2, a_3) \cdot (b_1''^{-1}, b_2''^{-1}, b_3''^{-1}) - (c_1, c_2, c_3)) \cdot (q_1^{-1}, q_2^{-1}, q_3^{-1}) =$

$= (|(2 \cdot 3 - 4) \cdot 3|_7, |(3 \cdot 2 - 10) \cdot 5|_{11}, |(4 \cdot 10 - 1) \cdot 9|_{13}) = (6,2,0)$.

9. Вычисляем

$(b_1, b_2, b_3) = ((b_1, b_2, b_3) \cdot (b_1''^{-1}, b_2''^{-1}, b_3''^{-1}) - (1,1,1)) \cdot (q_1^{-1}, q_2^{-1}, q_3^{-1}) = (1,1,1)$.

Так как $(b_1, b_2, b_3) = (1,1,1)$, то частное вычисляется как $(6,2,0)$.

3.2.3 Аппаратная реализации алгоритма деления с двумя наборами СОК

На рисунке 3.2 представлена схема устройства для основного деления модулярных чисел в формате системы остаточных классов. Схема устройства пунктирной линией разделена на левую и правую части. В левой части вычисления ведутся в основной системе СОК, а в правой части во вспомогательной СОК. Принцип работы данного устройства излагается ниже.

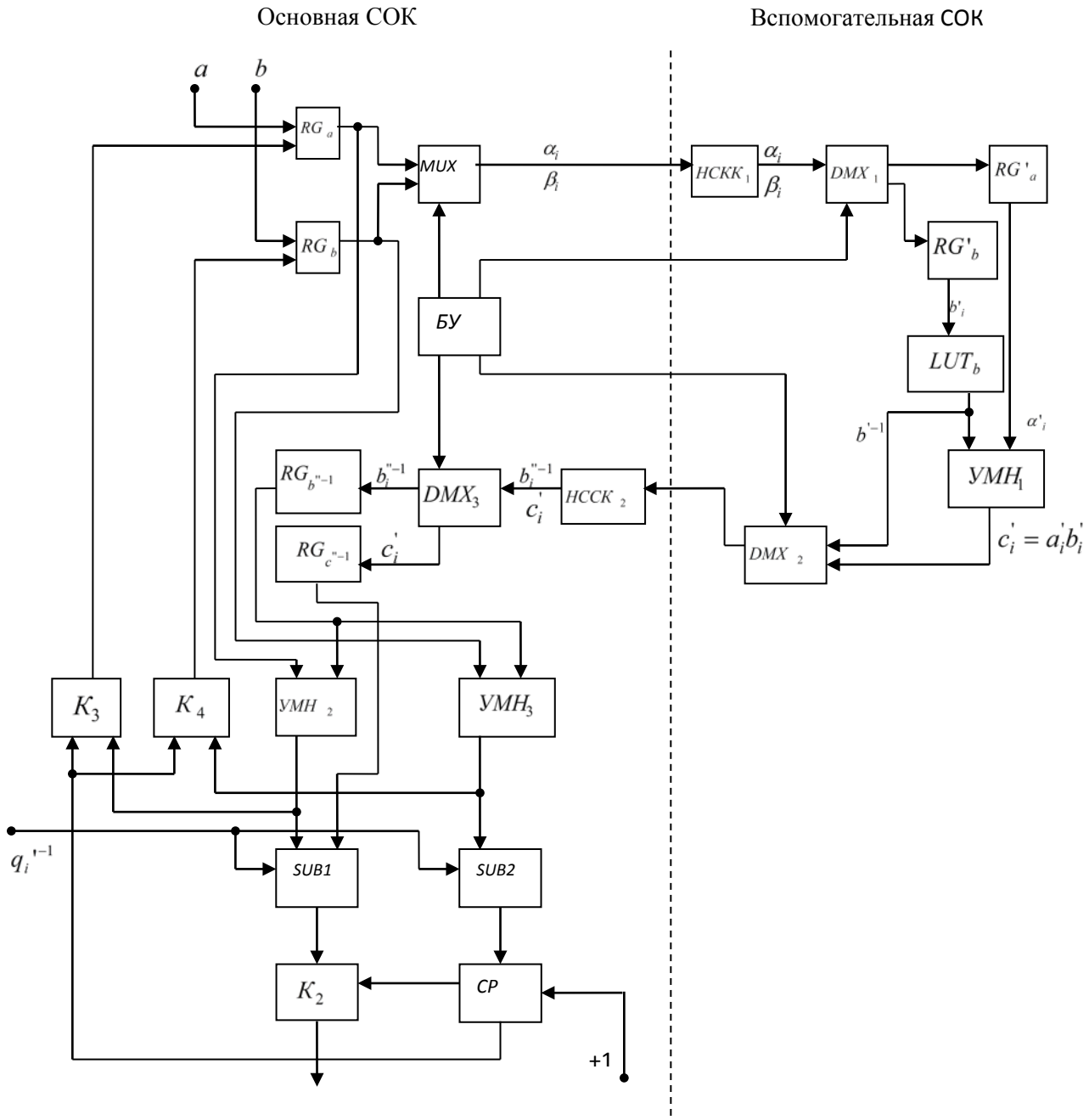


Рисунок 3.2 – Функциональная схема основного деления модулярных чисел в формате системы остаточных классов

Делимое a и делитель b, представленные в основной системе остаточных классов поступают на входные регистры RG_a и RG_b и далее на информационные входы мультиплексора MUX_1 2:1. Под действием кода на адресном входе формируемого блоком управления БУ, выходные сигналы MUX поочередно поступают на вход нейронной сети для расширения HC_p кортежа числовой системы вычетов (патент RU 2256226, Бюл. № 19 от 10.07.2005). Преобразователь

преобразует делимое a_i, представленное в основной СОК в делимое a_i', представленное во вспомогательной СОК, где $i = 1, 2, \ldots, m$ - модули СОК, которое через мультиплексор DMX_1 1:2 поступает на регистр RG'_a. Адресные входы демультиплексора DMX_1 подключены к блоку управления БУ.

После преобразования делимого аналогичным образом осуществляется преобразование делителя b, представленного в основной СОК в делитель b', представленный во вспомогательной СОК. При этом демультиплексор DMX_1 помещает данные делителя в регистр RG'_b. С выхода регистра RG'_b делитель b_i', представленный во вспомогательной СОК поступает на вход LUT_b-таблицы, где происходит преобразование делителя в мультипликативную инверсию $b_i'^{-1}$ и далее поступает на информационный вход демультиплексора DMX_2 1:2 и первый вход умножителя $УМН_1$, на второй вход поступают данные делимого a_i', представленные во вспомогательной СОК. Результат умножения $c_i' = a_i' \cdot b_i'$ поступает на информационный вход демультиплексора DMX_2 1:2. Под действием адресного входа, поступающего с блока управления, демультиплексор DMX_2 поочередно подает на вход преобразователя $НСКК_1$ $b_i'^{-1}$ и c_i'. Преобразование $НСКК_2$, аналогичен преобразователю $НСКК_1$ и осуществляет расширение $b_i'^{-1}$ и c_i', представленных во вспомогательной СОК в значения $b_i''^{-1}$ и c_i, представленные в основной СОК. С выхода $НСКК_2$ данные коммутируются DMX_3 1:2 на регистры $RG_{b''}^{''-1}$ и $RG_{c_i}^{''-1}$. Данные $b_i''^{-1}$ с выхода регистра $RG_i^{''-1}$ поступают на первые входы умножителей $УМН_2$ и $УМН_3$, а на вторые входы которых поступают, соответственно, делимое и делитель, представленные в основной СОК. Результаты умножения $ab_i''^{-1}$ и $bb_i''^{-1}$ поступают на первые входы схем вычитания SUB_1 и SUB_2 с умножением результатов вычитания на константу $q_i^{-1} = p_1' \cdot p_2' \cdot \ldots \cdot p_m' \bmod p_i$, где $i = 1, 2, \ldots, m$, на вторые входы которых поступают

c_i^{n-1} и 1. Результаты вычитания $\left(ab_i''^{-1}-c\right)\cdot q_i'$ и $\left(bb_i''^{-1}-1\right)\cdot q_i'$, соответственно схемы SUB_1 и SUB_2 с учетом умножения на константу q_i^{-1}, поступают на вход ключей K_2 и схемы сравнения с единицей CP. Если результат схемы сравнения равен 1, то на выходе ключа K_2 формируется частное, если результат не равен 1, то выходной CP открывает ключи K_3 и K_4, которые подключают данные $ab_i''^{-1}$ и $bb_i''^{-1}$ для следующей итерации.

Итак, основное деление модулярных чисел осуществляется только с помощью модульных схем, что и обеспечивает регулярную структуру устройства.

Анализ сложности устройства базируется на числе модулярных вычислений. Вычислительная сложность определяется вычислениями только по одному остатку, так как применяется параллельный алгоритм обработки. Модульное умножение выполняется по одному разу, а модульное сложение $m+1$ раз в преобразованиях a, b к a', b' и $m+1$, раз при преобразовании c', b'^{-1} к c, b''^{-1}. Вычисление b'^{-1} составляет $O(d)$, где d - количество цифр в модуле. Сложность устройства представлена в таблице 3.1.

Таблица 3.1 Сложность устройства деления модулярных чисел.

Операции	Модульное умножение	Модульное сложение
$a \to a'$, $b \to b'$	1	$\lfloor \log n \rfloor + 1$
$b' \to b'^{-1}$	d	-
$c' \to c$, $b'^{-1} \to b''^{-1}$	1	$\lfloor \log n \rfloor + 1$
$\left(ab''^{-1}-c\right)\cdot q_i^{-1}$, $\left(bb''^{-1}-1\right)\cdot q_i^{-1}$	2	-

В устройстве все числа имеют остаточное представление, поэтому не возникает необходимость преобразования из остаточного представления в бинарное. Частное после вычисления также представляется в остатках. Таким образом, устройство обладает большой эффективностью при выполнении операции деления модулярных чисел. К недостаткам алгоритма и устройства можно отнести необходимость использования двух наборов СОК, который

нивелируется применением базовых регулярных структур цифровой схемотехники, которые легко реализуются на FPGA.

3.3 Разработка нового высокоскоростного метода общего деления многоразрядных модулярных чисел с использованием КТО с дробными числами

3.3.1 Основные предпосылки и подходы к разработке алгоритма деления на основе метода приближенного вычисления позиционных характеристик

СОК дает преимущества быстрого сложения, вычитания и умножения, что обуславливает большой интерес к СОК в тех областях, где требуются большие объемы вычислений. Однако, некоторые операции, такие как сравнение и деление чисел весьма сложны в СОК. Нахождение более эффективных алгоритмов сравнения, определения знака и деления позволит найти новые перспективные области применения СОК.

Уже известные алгоритмы деления в СОК могут быть разделены на две класса: на основе сравнения чисел и на основе вычитания.

Алгоритмы, основанные на сравнении чисел, определяют частное при помощи итерации

$$a' = a - 2^i q_i b ,$$

где a' и a, соответственно, текущее и следующее делимое; b – делитель; q_i – разряд частного. Для определения q_i необходимо делимое a сравнить с величиной $2^i b$.

Второй класс алгоритмов определяет частное на основе итераций $a' = a - Q_i b$. Частное Q_i генерируется на каждой итерации из полного диапазона СОК, а не выбирается из небольшого множества.

Известен алгоритм целочисленного деления, который работает аналогично обычной операции двоичного деления [110]. Применение этого алгоритма и его

модификаций имеет основной недостаток, что каждая итерация требует сравнения величин чисел.

Алгоритм не имеющий таких недостатков, предложенный в [91] основан на замене реального делителя на приближенный, который может быть одним или произведением нескольких модулей СОК. Алгоритм обеспечивает корректность при условии $b \le \bar{b} < 2b$, где b – реальный делитель, и $\bar{b}$ – приближенный делитель. Легко видеть, что это условие не может быть удовлетворено для любого набора модулей (например: $p_1 = 9, p_2 = 11, b = 4$).

Основными недостатками данного алгоритма являются его потребность использования ОПСС, операции масштабирования и специальной логики и таблиц для определения полученного приблизительного делителя. Было предложено несколько алгоритмов решения проблемы деления на основе методов сравнения чисел и определения знака числа. Все предложенные алгоритмы, тем не менее, имеют недостаток длительного времени выполнения и больших аппаратурных затрат, поскольку используют ОПСС, КТО и другие затратные операции.

В работах [113, 115] представлен высокоскоростной алгоритм деления, в котором вместо использования ОПСС и КТО при делении модулярных чисел использовались сравнения высших степеней делимого и делителя. Время и аппаратные затраты в этих алгоритмах меньше, чем в других алгоритмах. Хотя этот алгоритм содержит избыточные этапы. Для ускорения вычисления текущих частных Y.H. Yang предложил алгоритм деления на основе проверки четности, в котором вычисление частного происходит в два раза быстрее. Однако вычисления высших степеней двойки отнимает много времени в СОК, которые выполняются в каждом цикле.

Большинство известных алгоритмов используют трудно выполнимые операции, такие как: КТО, преобразование в обобщенную позиционную систему счисления (ОПСС), операции масштабирования, расширения, определение знака числа и сравнения чисел, которые снижают скорость выполнения и требуют

больших аппаратных затрат при выполнении операции деления модулярных чисел.

Рассмотренный алгоритм деления в формате СОК кроме основного набора модулей СОК использует еще и замещающую систему модулей, которая является вспомогательной для сохранения остатков делимого и делителя. Представленные в СОК делимое и делитель преобразуются в различные СОК представления в различных системах модулей. Использование двух наборов СОК ведет к большой избыточности и необходимости прямого и обратного перехода от основного набора модулей к вспомогательному и обратно при выполнении операции деления, что резко снижает скорость выполнения операции деления. В работе [111] автором предложен быстрый алгоритм деления на основе использования индексного преобразования над полем Галуа $GF(p)$, который просто реализуется с помощью табличного поиска. Однако, этот алгоритм эффективен при обработке данных не более 6–10 бит и когда модуль представляет собой простое число. Для больших диапазоном этот алгоритм не эффективен, так как генератор простого числа P должен быть очень большим, что бы целые числа были отображены над полем Галуа.

Большинство из предложенных итерационных алгоритмов содержат большое количество операций в каждой итерации. Алгоритм на основе КТО с дробями является лучшим, у которого временная сложность доходит до $O(nb)$, где n – количество модулей СОК, b – количество битовых разрядов в каждом модуле при предположении, что величина модулей более-менее одинакова. Недостатком этого алгоритма является множество операций, выполняемых в каждой итерации: операции сложения, умножения, сравнения и проверки четности. Кроме того, в конце выполнения алгоритма требуется преобразование частного из системы $\{-1,0,1\}$ в систему $\{0,1\}$ дает дополнительную нагрузку на время выполнения процедуры деления модулярных чисел.

В данном параграфе предлагается разработать новый алгоритм модулярного деления, который в каждой итерации содержит один сдвиг и одно

вычитание. Это усовершенствование значительно уменьшает время каждой итерации алгоритма. Вычислительная сложность определяется сложением N-битного дробного числа, где N – количество разрядов дробного числа без учета операции сдвига.

На основе КТО с дробями мы предлагаем новый алгоритм деления модулярных чисел. Частное $\left\lfloor \frac{a}{b} \right\rfloor$ определяется на основе итераций $\Delta_i = \Delta_{i-1} - 2^{j-i} q_{j-i} F(b)$, где j – наивысшая степень частного; q_j – наивысший разряд частного; i – номер итерации, $\Delta_1 = F(a) - F(b) q_j 2^j \approx [F(a)]_{2^{-t}} - [F(b)]_{2^{-t}} q_j 2^j$, $1 \le i, j < \left\lfloor \log_2 \frac{P}{2} \right\rfloor$, дробные величины $F(a) = \left| \frac{a}{P} \right|_1$, $F(b) = \left| \frac{b}{P} \right|_1$, $P = \prod_{i=1}^{n} p_i$ – полный диапазон, p_i – модули СОК; $[F(a,b)]_{2^{-t}} < F(a,b) < [F(a,b)]_{2^{-t}} + 2^{-t}$, округление величин $F(a)$ и $F(b)$ до 2^{-t} бита, при котором возникающая погрешность на оказывает влияние на точность вычислений, обозначается как $[F(a)]_{2^{-t}}$ и $[F(b)]_{2^{-t}}$, Δ_i и Δ_{i-1}, соответственно, текущее и следующее значение $F(a)$, определяемое сдвигом вправо на один разряд делителя и вычитания его из делимого. Цифры двоичного представления частного формируются последовательно на основе знака результата вычитания. Предложенный алгоритм отличается от известных простой реализацией, так как временная сложность i итераций определяется временем выполнения одной операцией сдвига и одной операцией вычитания.

Достоинством предложенного алгоритма является то, что он не использует в процессе деления в качестве промежуточных данных числа, представленные в ОПСС, а также такие трудно выполнимые операции в СОК, как сравнение, масштабирование, расширение базы и определение знака, что позволило повысить вычислительную эффективность модулярного деления.

3.3.2 Новый алгоритм деления в СОК на основе КТО с дробными числами

В данном параграфе представлен новый алгоритм деления, подробное описание которого будет представлено в следующем параграфе.

Алгоритм работает следующим образом. Получив на вход алгоритма a и b, вычисляются частное Q и остаток R такие, что $a = Qb + R$ и $0 \le R < b$. Результат Q определен до ближайшего целого числа, так что $Q = \left[\frac{a}{b}\right] + R$. Делимое a и делитель b должны быть в интервале $-\left\lfloor \frac{P}{2} \right\rfloor \le a, b \le \left\lfloor \frac{P-1}{2} \right\rfloor$.

Алгоритм деления целых чисел $\frac{a}{b}$ можно описать итеративной схемой, которая выполняется в два этапа. На первом этапе осуществляется поиск старшей степени 2^i при аппроксимации частного двоичным рядом. На втором этапе осуществляется уточнение аппроксимирующего ряда. Основной проблемой первого этапа является поиск наивысшей степени 2, содержащейся в некотором числе, эффективное решение которой будет показано в следующем разделе. При выполнении итераций может возникнуть ситуация, когда промежуточные данные превысят динамический диапазон P. В этом случае потребуется операция расширения диапазона до величины $P' = P \cdot p_{n+1}$, то есть потребуется расширить базу СОК, добавив дополнительный модуль. Чтобы избежать этого расширения базы, которое является вычислительно сложной операцией, необходимо сравнивать не делимое с промежуточными делителями, а текущие результаты итераций *(i)* с предыдущими значениями итераций (*i*-1*)*. Процесс удваивания делителя повторяем до тех пор, пока значение промежуточного делителя при *i*-итерации будет меньше, чем при *i*-1 итерации.

Структура предлагаемого алгоритма основана на использовании относительных значений делимого и делителя. Деление выполняется как последовательность вычитаний делителя сначала из делимого, а затем из образующихся результатов итераций в процессе выполнения операций деления.

При итерационном процессе деления в позиционной системе счисления, для поиска старшей степени ряда аппроксимации частного и для уточнения аппроксимирующего ряда сравниваются делимое с удвоенными делителями или с суммой членов ряда. Применение этого принципа для СОК может привести к

некорректной работе алгоритма, так как при переполнении динамического диапазона промежуточного делителя, восстановленное число может выйти за пределы рабочего диапазона, вызванное цикличностью СОК, значение которого будет меньше делимого, что не соответствует действительности, так как на самом деле числа будут превышать диапазон P и алгоритм перейдет в режим "зацикливание". Например, если модули СОК равны $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, тогда диапазон $P = 2 \cdot 3 \cdot 5 \cdot 7 = 210$. Допустим при восстановлении получили число $X = 220$. В СОК $X = 220 = (0,1,0,3)$, то есть $X = 220$ и $X' = 10$ имеют одинаковые представления в СОК. Приведенная неоднозначность может привести к нарушению работы алгоритма. Для преодоления этой трудности необходимо в СОК сравнивать результаты текущих значений итераций с предыдущими, что позволяет правильно определить большее или меньшее число. Итак, факт переполнения динамического диапазона в СОК можно использовать для принятия решения «больше - меньше». На первой итерации происходит сравнение делимого с делителем, а на остальных итерациях происходит сравнение частных остатков $\Delta_i < \Delta_{i-1}$. На каждой новой итерации происходит сравнение текущего значения с предыдущим. Последовательное применение этих итераций приводит к формированию цепочки неравенств $\Delta_1 > \Delta_2 > ... > \Delta_m < \Delta_{m+1} < 0$, определяющих необходимое количество требуемых итераций зависящих от величин делимого и делителя. Таким образом, алгоритм реализуется за конечное число итераций. Пусть на $m+1$ итерации зафиксирован случай окончания возрастающей последовательности $\Delta_m < \Delta_{m+1}$, что соответствует переполнению диапазона СОК, то есть $\Delta_{m+1} > P$ и $a < \Delta_{m+1}$. На этом процесс формирования интерполяции частного двоичным рядом или набором констант в СОК завершается. Итак, процесс аппроксимации частного может осуществляться путем сравнения соседних Δ_i и Δ_{i-1}.

На втором этапе отыскиваются степени 2, представленные в СОК которые входят в аппроксимационный ряд частного.

3.3.3 Детальное описание нового алгоритма деления

Приведем детальное описание предложенного алгоритма деления модулярных чисел в СОК на основе приближенного метода определения позиционной характеристики.

Определение знака частного

Шаг 1. Вычисляем приближенные значения делимого $[F(a)]_{2^{-N}}$ и делителя $[F(b)]_{2^{-N}}$ в соответствии с моделью (2.4) и определяем их знаки.

Шаг 2. Если числа a и b имеют разные знаки, то частное будет отрицательным, если одинаковые знаки – то положительным. В дальнейших вычислениях используем относительные величины делимого a и делителя b.

Аппроксимация частного.

Шаг 3. Сравниваем путем вычитания приближенные значения делимого $[F(a)]_{2^{-N}}$ и делителя $[F(b)]_{2^{-N}}$. Если $[F(a)]_{2^{-N}} \le [F(b)]_{2^{-N}}$, то процесс деления заканчивается и частное $\left\lfloor \frac{a}{b} \right\rfloor$, соответственно, равно 0 или 1. Если $F(a) > F(b)$, то осуществляется поиск старшей степени 2^{-j} при аппроксимации частного a двоичной дробью, где $-j$ - первый значащий разряд двоичной дроби.

Проведем поиск старшей степени двоичной дроби.

Шаг 4. Сдвигаем функцию $[F(b)]_{2^{-N}}$ влево до изменения первого двоичного разряда после запятой (аналогично нормализации). Количество сдвигов определяет старшую степень j, которая регистрируется счетчиком импульсов, соединенных с памятью V, где хранятся остаточные представления степеней 2, представленные в СОК и входящие в диапазон делимого.

Необходимо отметить, что при аппаратной реализации операции сдвига можно реализовать за один такт на специальной разработанной комбинационной схеме, состоящей из схем двух входовых элементов «И», один из входов, который подключен к разрядам регистра хранения функций $[F(b)]_{2^{-N}}$, а второй вход – к сигналу синхронизации. Выходные сигналы элементов «И» отождествлены с

соответствующими степенями 2. И там, где на выходе определенного элемента «И» появится первый сигнал «1» после впереди стоящих нулей и будет соответствовать старшей степени аппроксимационного ряда.

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

Уточнение аппроксимационного ряда частного.

Шаг 5. В процессе аппроксимации частного счетчик установлен в состояние, соответствующее старшей степени j. Выходы счетчика являются адресными входами памяти V. Из памяти выбираем константу $2^j \bmod p_i$ (старшая степень ряда), где $i = 1, 2, ..., n$ и умножаем ее на делитель. Величину $F_1(b) = 2^j F(b)$ сравниваем с делимым $F(a)$ путем вычитания. Константы $2^j \bmod p_i$, $1 \le j \le \left[\log_2 \frac{P}{2}\right]$ предварительно записаны в память V, Q частное.

Шаг 6. Вычисляем частный остаток $\Delta_1 = F(a) - F_1(b)2^j$. Если в знаковом разряде Δ_1 значение "1", то соответствующая степень ряда отбрасывается, если значение "0", то в сумматор частного добавляем значение члена ряда с этой степенью, то есть $2^j \bmod p_i$, $1 \le i \le n$, $0 \le j \le N$.

Шаг 7. Для проверки члена ряда со степенью 2^{j-1} делим $\Delta_1 / 2$ путем сдвига вправо и вычитания из Δ_1 – $2^{j-1}b$ и анализируем знак результата. Если $\Delta_1 < 2^{j-1}b$, то знак отрицателен и соответствующая степень ряда отбрасывается, если $\Delta_1 > 2^{j-1}b$, то знак положительный и в сумматор частного добавляем значение члена ряда с этой степенью, то есть $2^{j-1} \bmod p_i$ и $\Delta_2 = \Delta_1 - 2^{j-1}b$.

Шаг 8. Аналогично проверяем все оставшиеся члены ряда до нулевой степени. Последнее $\Delta_i = R = \Delta_{i-1} - F_{i-1}$, то есть $0 \le R < b$ будет остатком от деления a на b. Частным Q будет сумма всех $2^j \bmod p_i$, необходимых для образования частного, которые были накоплены в сумматоре со знаком, определенным на втором шаге. Работа алгоритма завершается.

Итак, в предложенном алгоритме в каждой итерацией используются только две операции: операция сдвига и операция вычитания. В известных алгоритмах [90-92] в каждой итерации выполняются операции умножения, сложения, вычитания и определения четности. Поэтому предложенный алгоритм проще и выполняется быстрее. По сравнению с другими алгоритмами предложенный алгоритм в каждой итерации уменьшает вычислительную сложность более чем в два раза.

3.3.4 Численный метод определения частного при модулярном делении

Пример работы нового алгоритма деления.

Найти частное $Q=\frac{a}{b}$ от деления числа a = 97 на число b = -8в СОК с основаниями $p_1=2,\ p_2=3,\ p_3=5,\ p_4=7$.

Представим в СОК числа a и b по основаниям 2,3,5,7, тогда

$a_{10}=97\rightarrow(1,1,2,6)_{СОК}$,

$b_{10}=-8\rightarrow(0,1,2,6)_{СОК}$.

Вычислим приближенные значения $F(a)$ и $F(b)$ по формуле (2.1) и определим знаки чисел a и b по формуле (2.4).

$$[F(a)]_{2^{-12}}=|1\cdot 0{,}100000000000+1\cdot 0{,}010101010101+$$

$$+10\cdot 0{,}100110011001+110\cdot 0{,}100100100100|_1=0{,}011101011111.$$

Так как $0<[F(a)]_{2^{-12}}<0{,}1$, то число a положительное.

$$[F(b)]_{2^{-12}}=|0\cdot 0{,}100000000000+1\cdot 0{,}010101010101+$$

$$+10\cdot 0{,}100110011001+110\cdot 0{,}100100100100|_1=0{,}111101011111.$$

Так как $0{,}1<[F(b)]_{2^{-12}}<1$, то число b отрицательное.

Числа a и b имеют разные знаки, поэтому знак частного будет отрицательный. Для нахождения абсолютной величины частного разделим a на $-b=(0,0,0,0)-(0,1,2,6)=(0,2,3,1)$ по алгоритму, изложенному выше.

Относительные величины делимого a и делителя $-b$ равны:

$$[F(a)]_{2^{-12}} = 0{,}011101011111; \qquad [F(-b)]_{2^{-12}} = 0{,}000010100001.$$

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

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

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

$$\Delta_1 = [F(a)]_{2^{-12}} - 2^3 \cdot [F(-b)]_{2^{-12}} = 0{,}011101011111 - 0{,}010100001 = 0{,}001001010111$$

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

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

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

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

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

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

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

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

$$\Delta_4 = \Delta_2 - 2^{00} \cdot [F(-b)]_{2^{-12}} = 0{,}000000110011 - 0{,}000010011001 = 1{,}111110011010.$$

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

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

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

С учетом знака, окончательно получим, что

$$\left\lfloor \frac{\mathrm{a}}{\mathrm{b}} \right\rfloor = -12 \overset{COK}{\rightarrow} (0,0,3,2).$$

Для оценки эффективности предложенного метода проанализируем сложность выполнения операций на каждом из этапов работы алгоритма.

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

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

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

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

Таким образом, для выполнения алгоритма требуется $N\lceil \log_2 n \rceil + (N+2)\lceil \log_2 P \rceil$ времени при предположении, что сложение/умножение происходит за единицу времени и $N(p_1 + p_2 + ... + p_n)$ бит памяти.

Известный алгоритм [115], требует выполнения в худшем случае $2\lceil \log_2 P \rceil$ операций сравнения в СОК на основе ОПСС. Для каждой операции преобразования необходимо выполнить одно табличное умножение и

суммирование n чисел ширины $\max\{\log_2 p_i\}$, $1 \le i \le n$, которое в силу особенностей метода ОПСС не может быть организованно параллельно. Таким образом, вычислительная сложность известного алгоритма составляет $2n\lceil \log_2 P \rceil \cdot \max\{\log_2 p_i\}$.

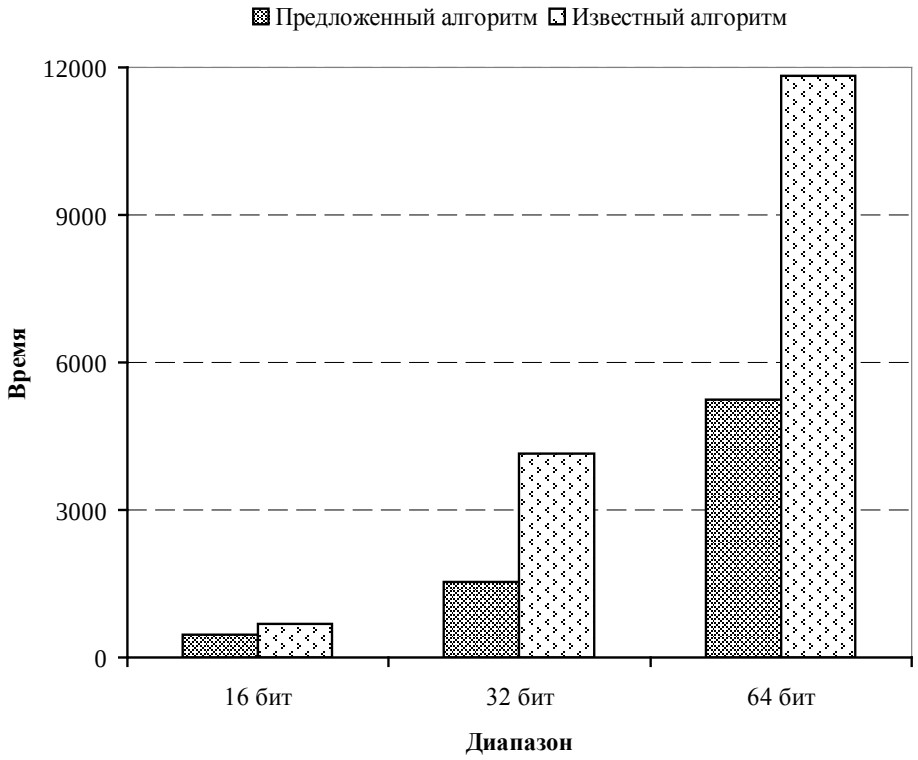


Рисунок 3.3 – Сравнение работы алгоритмов деления в СОК.

Сравнение алгоритмов деления в СОК приведено на рисунке 3.3. Для сравнения была смоделирована работа предложенного нового алгоритма деления и алгоритма из [92] в СОК с диапазонами 16, 32 и 64 бит. В качестве критерия сравнения было использовано количество тактов вычислительной системы, необходимых для выполнения алгоритмов. Результаты моделирования показывают, что предложенный алгоритм работает быстрее. Для СОК с диапазоном 16 бит время работы сокращается примерно в 1,4 раз; для СОК с диапазоном 32 бит – в 2,6 раз; для СОК с диапазоном 64 бит – в 2,3 раз. Данный результат свидетельствует о том, что разработанный алгоритм превосходит известные аналоги по скорости и является лучшим на сегодняшний день.

3.3.5 Аппаратная реализация нового алгоритма деления

Алгоритм деления модулярных чисел на основе КТО с дробными числами является лучшим алгоритмом на сегодняшний день. В данном пункте рассмотрим его аппаратную реализацию.

Предположим, что СОК содержит *n* модулей, *n* модулярных процессоров. Пусть *m* означает количество бит, требуемых для представления каждого остатка. Для удобства анализа сложности аппаратной реализации алгоритма деления модулярных чисел будем предполагать, что величины модулей более менее одинаковы. Данное предположение приводит к тому, что общая ширина шины модулярных процессоров будет содержать $M = n \cdot m$ бит.

Рассмотрим аппаратную реализацию алгоритма деления модулярных чисел, когда и делимое и делитель представляют собой произвольные целые числа и делитель не приводится к случаю попарно простого с модулями СОК.

Аппаратная реализация (рисунок 3.4) содержит шины *a* и *b* для подачи, соответственно делимого и делителя, шину частного *Q*. Ширина шин *a, b* и *Q* определяется *M* битами для каждой шины. Процесс деления определяется подачей однобитного сигнала по шине «Деление». Получив на выходы *a* и *b*, вычисляется *Q* такое, что $a = Q \cdot b + R$, где $0 \le R < b$.

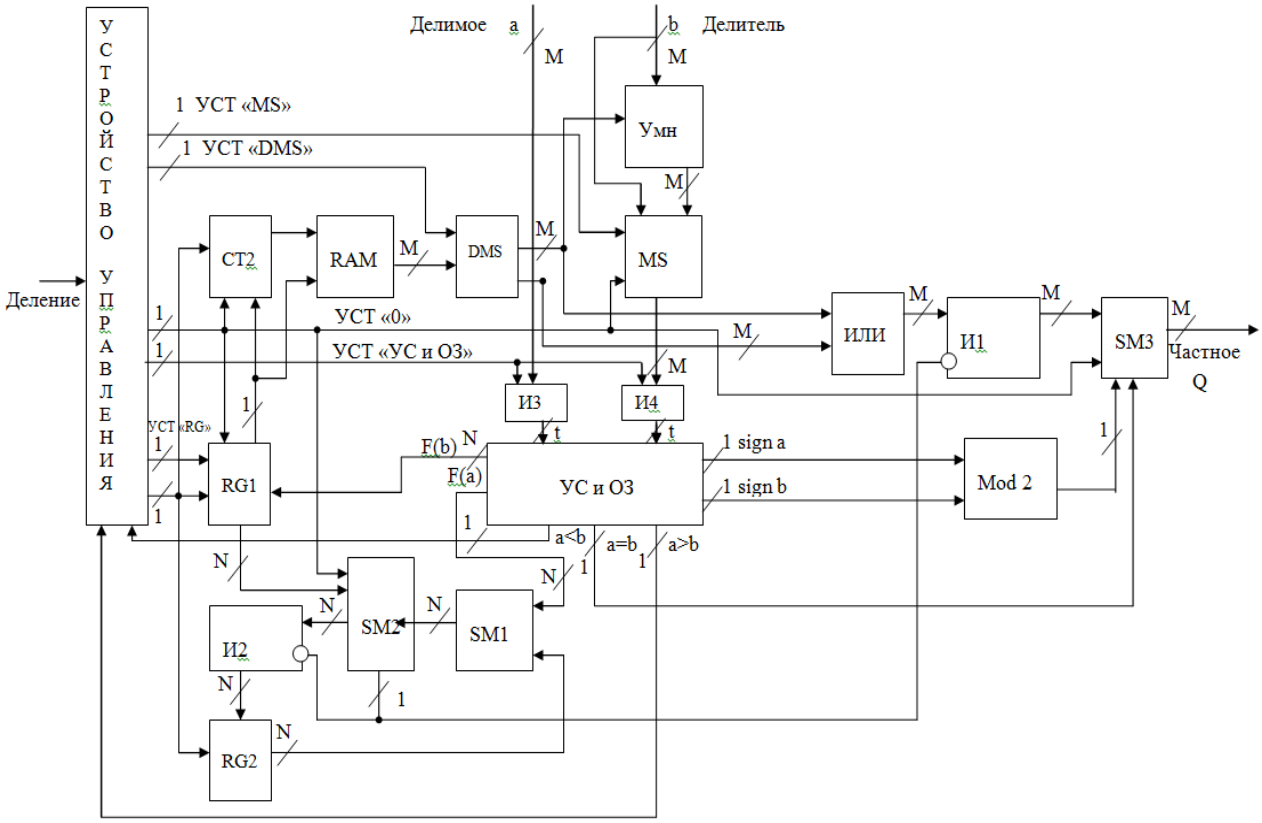


Рисунок 3.4 – Аппаратная реализация нового алгоритма деления

Режим аппроксимации частного двоичным рядом.

В исходном состоянии после поступления сигнала «Деление» устройство управления (УУ) формирует по однобитных шинам следующие сигналы (рисунок 3.4):

– «Уст. 0» для установки в нулевое состояние функциональных устройств;

– тактовые ТИ для управления работой регистров и счетчиков;

– установка адресного кода 2:1 мультиплексора «Уст. MS»;

– установка адресного кода 1:2 демультиплексора «Уст. DMS». Демультиплексор DMS в зависимости от адресного кода коммутирует наивысшую степень 2 либо одновременно на умножитель «УМН» и элемент ИЛИ, либо только на элемент ИЛИ, выход которого соединен со входом элемента «Запрет» $И_1$ Мультиплексор MS в зависимости от адресного кода коммутирует на выход устройства сравнения и определения знака (УСиОЗ) либо непосредственно делитель b, либо делитель, умноженный на наивысшую степень $b\ \ 2^k$;

– установка регистра для сдвига вправо «Уст. RG_1»;

– установка блокировки УСиОЗ «Уст. УСиОЗ».

Делимое a и делитель b, представленные в системе остаточных классов по модулям СОК по М-битным шинам поступают на вход УСиОЗ, в котором происходит вычисление относительных значений делимого $[F(a)]_{2^{-N}}$ и делителя $[F(b)]_{2^{-N}}$, определение их знаков и сравнение. Делимое через элемент $И_3$ непосредственно поступает на вход УСиОЗ, а делитель через мультиплексор MS, на адресном входе которого установлен соответствующий адрес. УСиОЗ. Если $[F(a)]_{2^{-N}} = \left|\frac{a}{P}\right|_1 < [F(b)]_{2^{-N}} = \left|\frac{b}{P}\right|_1$, то УСиОЗ формирует сигнал $b>a$ (делитель больше делимого), который поступает на вход схемы управления и устройство деления устанавливается в начальное состояние, при этом частное Q=0. Если $[F(a)]_{2^{-N}} = \left|\frac{a}{P}\right|_1 = [F(b)]_{2^{-N}} = \left|\frac{b}{P}\right|_1$, то УСиОЗ выдает сигнал равенства делимого и делителя $a=b$, который поступает на вход сумматора SM_3, где записывается константа $1 = (1,1,1,1)_{СОК}$. Если $[F(a)]_{2^{-N}} = \left|\frac{a}{P}\right|_1 > [F(b)]_{2^{-n}} = \left|\frac{b}{P}\right|_1$, то УСиОЗ формирует сигнал $a>b$, под действием которого схема управления переводит устройство деления в режим аппроксимации ряда частного.

Далее анализируются знаки делимого a и делителя b. Знак частного Q определяется выражением $sign Q = sign a \oplus sign b = sign a \overline{sign b} + \overline{sign a}\, sign b$. Сигналы знака делимого $sign\ a$ и знака делителя $sign\ b$ с выхода УСиОЗ по однобитным шинам поступают на вход схемы «Исключающее ИЛИ» (mod2), которая в соответствии с таблицей истинности выдает сигнал «$a=0$» или «$a=1$», который подается на вход сумматора частного SM_3. Если выходной сигнал элемента «Исключающее ИЛИ» равен 0, то частному присваивается положительный знак, т.е. равен 0; если равен 1, то частному присваивается отрицательный знак, т.е. знак частного равен 1. Кроме того, значения $[F(a)]_{2^{-N}}$ и $[F(b)]_{2^{-N}}$ по N-битным шинам, соответственно, поступают на вход сумматора SM_1 и регистр сдвига RG_1.

В SM_1 значение $F(a)$, преобразуется в дополнительный код и по N-битной шине поступает на вход схемы вычитателя SM_2. Значение $[F(b)]_{2^{-N}}$ по N-битной шине поступает на вход регистра сдвига RG_1. Под действием тактовых импульсов устройства управления содержимое регистра RG_1 сдвигается влево на число нулевых разрядов, стоящих перед первым значащим разрядом, которое регистрируется счетчиком $CT2$. Полученное число сдвигов соответствует наивысшей степени 2, содержащейся в делителе. Использование относительных величин позволило без вычисления итераций определить наивысшую степень двойки, входящей в аппроксимационный ряд частного, регистрируемого счетчиком $CT2$. Как только старший значащий разряд делителя $[F(b)]_{2^{-N}}$ в процессе сдвига становится единичным (аналогично нормализации чисел) RG_1 по одноразрядной шине фиксирует состояние $CT2$ и активирует сигнал «разрешение считывания» информации из памяти *RAM*. (аналогично указателю стека). Счетчик $CT2$ активирует адрес *RAM*, определяющий наивысшую степень 2^j ряда частного, которая должна содержаться в частном $Q = \left[\frac{a}{b}\right]$. В буферном регистре RG_2 в исходном состоянии записано нулевое значение. На этом режим аппроксимации частного заканчивается и на выходе *RAM* активируется ячейка памяти, где помещена наивысшая степень 2^j, входящая в аппроксимационный ряд частного. Итак, при аппроксимации частного b УСиОЗ выполняет одну операцию сравнения, одну операцию $[\log n]$ суммирования N-битных чисел, а RG_1 выполняет одну операцию сдвига на j разрядов.

Допустим $[F(b)]_{2^{-T}} = 0{,}000010100001$, тогда после сдвига содержимого регистра RG_1 состояние счетчика $CT2$ будет равно 0011, который активирует ячейку памяти, в которой заранее записана степень $2^3 \xrightarrow{COK} \left(\left| 2^3 \right|_{p_1}, \left| 2^3 \right|_{p_2}, ..., \left| 2^3 \right|_{p_n} \right)$.

Режим уточнения аппроксимирующего ряда частного от деления a на b.

Определенная в режиме аппроксимации наивысшая степень 2^j хранимая в памяти *RAM* в СОК по шине памяти шириной М-бит под действием однобитного

сигнала «Уст. DMS», выход которого коммутируется на схему умножения «УМН» и через элементы ИЛИ и «Запрет» $И_1$ на входы сумматора SM_3. Адресный код мультиплексора MS по однобитной шине «Уст. MS» устанавливается в следующее состояние и теперь на выход мультиплексора MS коммутируется значение $b2^j$, которое по шине шириной M бит поступает на вход УСиОЗ в котором вычисляется значение $\left[F\left(b\,2^{j}\right)\right]_{2^{-N}}$ и подается по шине $F(b)$ шириной N бит на вход регистра сдвига RG_1. Старая информация RG_1 стирается. Под действие сигнала «Уст. УСиОЗ», поступающего на входы $И_3$ и $И_4$ шины делимого a и делителя b от схемы УСиОЗ отключаются. Под действием управляющих сигналов устройства управления содержимое RG_1 по шине шириной N бит поступает на один из входов SM_2, на второй вход которого с выхода сумматора SM_1 поступило значение $F(a)$ по шине шириной N бит. Далее сигнал «Уст. RG_1» переводит регистр RG_1 в режим сдвига вправо.

Из содержимого сумматора SM_2 вычитается делитель умноженный на высшую степень 2 и анализируется знак результата вычитания, т.е. вычисляется значение $[F(a)]_{2^{-N}} - \left[F\left(b\,2^{k}\right)\right]_{2^{-N}}$. Если в знаковом разряде результата вычитания в сумматоре SM_2 стоит «ноль», то есть делимое $[F(a)]_{2^{-N}} > \left[F\left(b\,2^{k}\right)\right]_{2^{-N}}$, тогда на запрещающие входы схем «запрета» «$И_1$» и «$И_2$» поступают «нули» и схемы «Запрет» $И_1$ и $И_2$ пропускают высшую степень частного со второго выхода DMS через элементы ИЛИ и $И_1$ на вход сумматора частного SM_3, а результат вычитания сумматора SM_2, то есть остаток делимого, приходящий на остальные степени 2 по шине N бит через элемент «запрета» $И_2$ подается на вход буферного регистра RG_2, и затем по шине шириной N бит поступает на вход сумматора SM_1, в котором стирается старое содержимое и записывается новое значение. Далее происходит сдвиг «вправо» содержимого регистра RG_1 и процесс происходит аналогично вышеизложенному. При этом, если в знаковом разряде результата суммирования в сумматоре SM_2 будет стоять «1», то есть относительное значение делителя больше делимого, то появившаяся единица, поступает на запрещающие входы схемы «запрета» «$И_1$» и «$И_2$», которые запрещают прохождение соответствующей

степени на сумматор частного SM_3 и результата суммирования сумматора SM_2 на вход буфера RG_2, то есть регистр сохраняет предыдущее значение результата вычисления. На вход сумматора частного SM_3 поступают только те уточненные степени 2, которые являются членами ряда частного. Процесс преобразования заканчивается после анализа степени 2^0. Таким образом, при уточнении итеративно удаляются лишние члены аппроксимационного ряда частного путем несложных преобразований, состоящих из операций сдвига и вычитания.

Уточнение аппроксимации частного выражается как последовательность вычитания делителя, сначала умноженного на наивысшую степень двойки из делимого, а затем сдвига и вычитания из образующихся частных остатков в процессе вычисления частного. Отличительной особенностью при уточнении аппроксимационного рада частного нового алгоритма от известных является то, что только в первой итерации используется делимое a и произведение делителя на наивысшую степень $b2^j$. Остальные итерации базируются на принципе, предложенном в работе, а именно сдвиг вправо $b2^j$ на один разряд в каждой итерации, что равносильно умножению на 2 и вычитания результатов соседних итераций, т.е. $\Delta_1 - \Delta_{i-1}$. Данный принцип уникален тем, что каждая итерация содержит две операции: операцию сдвига на один разряд вправо и операцию вычитания с анализом знака вычитания, что значительно увеличивает скорость операции деления. В известных алгоритмах в каждой итерации выполняются операции умножения, сложения, сравнения и проверки на четность. Если предположить, что каждая операция выполняется за единицу времени t, то общее время выполнения итерации равно $4t$. В предложенном алгоритме каждая итерация выполняется за время выполнения одного вычитания и одного сдвига, т.е. $2t$. Выигрыш примерно равен 2. Анализ работы предложенного алгоритма, состоящего соответственно, из аппроксимации и уточнения частного показал существенное преимущество его с известными по времени деления модулярных чисел.

Это лучший на сегодняшний день алгоритм основного деления модулярных чисел. Сокращение аппаратных и временных затрат предложенного алгоритма по сравнению с известными достигается тесной связью архитектурных вычислений с аппаратной реализацией, что позволило значительно сократить вычислительную сложность деления модулярных чисел. Предложенный алгоритм отличается от известных простотой его реализации, который требует меньшего объема вычислений по сравнению с существующими алгоритмами.

3.3.6 Экспериментальные исследования математического метода моделирования нового алгоритма деления в среде ISE Design Suite 14.7 в базисе программируемых логических интегральных схем

Анализ схемы деления модулярных чисел показал, что основным блоком, определяющим вычислительную сложность, является устройство сравнения и определения знака числа, реализация которого может быть выполнена на базе КТО, ОПСС или модификации КТО с использованием дробей. В работе исследованы модели всех трех методов.

Моделирование проводилось с использованием средств среды ISE Design Suite 14.7. В качестве цели компилирования использована плата Kintex-7 KC705 XC7K70T-2FBG676 без блоков DSP48E1. Данная плата содержит 10250 блоков Slice и 300 блоков ввода/вывода. Моделирование проводилось при фиксированном количестве оснований с изменением разрядности модулей. В каждом случае подбирались простые основания системы заданной разрядности, общие для каждого из методов. Было использовано 4 основания с количеством бит модулей 5, 9, 13, 17, 21 и 25. Динамический диапазон системы приближенно равен произведению количества оснований на разрядность каждого из них. На рисунке 3.5 представлен график, отражающий использование ресурсов платы при использовании модулей различной разрядности для каждого из анализируемых методов.

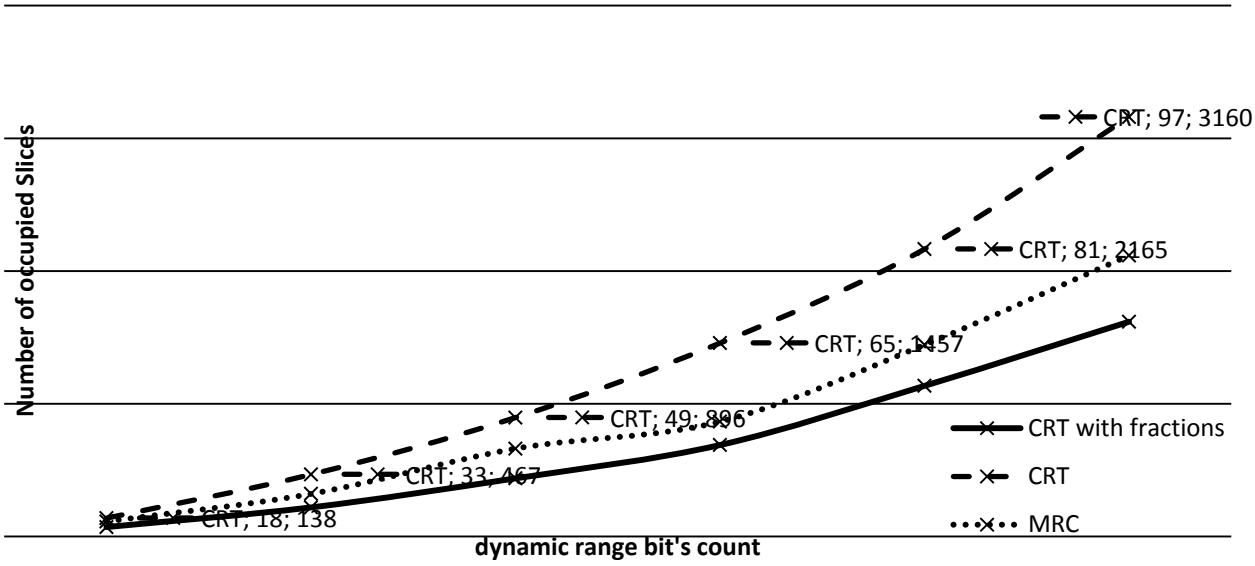


Рисунок 3.5 – Зависимость количества использованных блоков Slice платы FPGA Kintex-7 KC705 от разрядности используемых модулей системы.

Под задержкой схемы будем понимать максимальное время, затрачиваемое произвольным сигналом на преодоление всей схемы от некоторого входа до некоторого выхода. Оценка задержки позволяет получить представление о быстродействии реализованного алгоритма, в том числе оценить частоту работы схемы. На рисунке 3.6 представлен график зависимости частот схем каждого из алгоритмов для систем оснований с модулями различной разрядности.

В качестве примера рассмотрим разрядность диапазона 64 как одну из наиболее распространенных в компьютерных системах. Для того, чтобы можно было представлять числа в данной системе достаточно представить каждый из четырех модулей 16-ти или 17-ти битным числом. Пусть в данном случае система модулей имеет вид {65537, 65539, 65543, 65551}, диапазон данной системы представляет собой 65-битное число, что покрывает диапазон 64 бита. Для данной разрядности приближенный метод требует всего 689 блоков Slice, в то время как метод ортогональных базисов и улучшенная схема ОПСС 1457 и 865 блоков соответственно. С другой стороны, частота приближенного метода составляет 62,5 МГц, что в 7,6 раза больше, чем требует восстановление по КТО, и в 10,1 раз больше, чем улучшенный ОПСС.

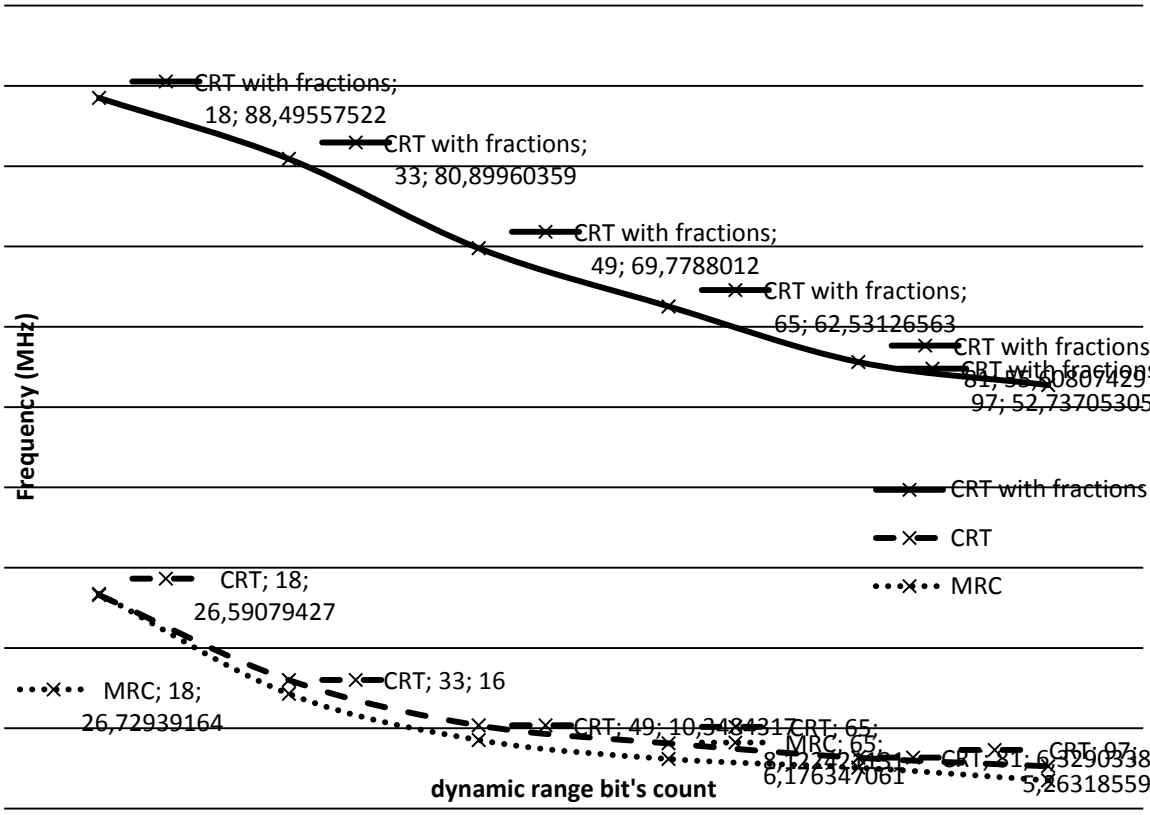


Рисунок 3.6 – Зависимость частоты от разрядности используемых модулей системы.

Описанный в работе новый алгоритм позволяет ускорить выполнение процедуры деления в СОК по сравнению с известными аналогами. Данное обстоятельство объясняется тем, что структура алгоритма содержит весьма простые операции: при аппроксимации частного используются операции сложения и сдвига, а при уточнении частного – операции сдвига и вычитания. Использование дробных чисел КТО позволило избежать необходимости использования в алгоритме таких операций как нахождение остатка числа по модулю и перевод числа в ОПСС. Итак, все промежуточные вычисления внутри алгоритма, являются короткоразрядными простыми операциями, что приводит к значительному ускорению работы алгоритма. Предложенные новые технологии делают алгоритм более быстрым, точными, и, следовательно более подходящим для многоразрядного модулярного деления, что является усовершенствованием предыдущих публикаций. Моделирование алгоритма на FPGA Kintex-7 показало снижение аппаратурных затрат и существенное увеличение скорости обработки по сравнению с алгоритмами, основанными на КТО и ОПСС.

Перспективным направлением дальнейших исследований является поиск быстрых и эффективных алгоритмов выполнения таких проблемных операций в СОК как обратный перевод из СОК в позиционную систему счисления, оптимальный выбор набора модулей СОК при различных диапазонах для конкретных приложений, каждый из которых может дать существенный импульс развитию данной области компьютерной математики за счет появления новых направлений практического использования СОК.

3.4 Выводы по третьей главе

1. Предложена нейронная сеть нейронная сеть для выполнения операции деления на делитель, равный одному из модулей СОК. Нейронная сеть для деления состоит из совокупности нейронных сетей конечного кольца и может быть использована в качестве базового элемента при делении чисел на делитель, представляющий собой произведение модулей.

2. Разработана функциональная схема деления в формате СОК. Особенностью операции деления является то, что делимое, делитель и промежуточные результаты имеют остаточное представление и не требуется преобразование из остаточного представления в бинарное. Частное после вычисления также представлено в остатках, поэтому вычисления могут быть эффективными при дальнейшем использовании частного. К недостаткам данной схемы можно отнести наличие вспомогательной СОК, которая увеличивает время операции деления.

3. Впервые разработан новый метод и алгоритм деления модулярных чисел с использованием приближенного вычисления позиционной характеристики для аппроксимации и уточнения частного. Все промежуточные преобразования внутри алгоритма являются простыми операциями типа сдвига и сложения. Весь алгоритм имеет значительное ускорение за счет представления промежуточных делимого и делителя в качестве степени двойки. Все эти новые технологии делают алгоритм более быстрым, точным и, следовательно, более подходящим для многоразрядного модулярного деления, что является

усовершенствованием предыдущих публикаций и считать его лучшим алгоритмом деления на сегодняшний день.

4. Предложенный новый и упрощенный алгоритм деления модулярных чисел и аппаратная его реализация на основе метода приближенного вычисления позиционной характеристики значительно снижает сложность аппаратных средств и увеличивает скорость деления. Экспериментальное исследование математического метода моделирования алгоритма в среде ISE Design Suite 14.7 при использовании четырех модулей $p_1 = 65537,\ p_2 = 65539,\ p_3 = 65543,\ p_4 = 65551$, покрывающие диапазон $P = 64$ бита показала следующие результаты: предложенный метод требует всего 689 блоков Slice, в то время как метод деления на основе ортогональных базисов и улучшенный метод ОПСС 1417 и 865 блоков, соответственно; частота приближенного метода составляет 65,5 МГц, что в 7,6 раза больше, чем требует деление на основе КТО, и в 10,1 раз больше, чем улучшенный метод на основе ОПСС. Предложенный метод может быть использован в тех приложениях, где кроме коротких операций сложения и умножения часто используется операция деления, что позволяет расширить область применения СОК.

5. Разработан численный метод определения частного при модулярном делении, который способен оптимизировать схему деления на базе платформы FPGA, перестраиваясь под различные совокупности наборов модулей СОК.

4. Разработка моделей отказоустойчивых нейрокомпьютеров на основе системы остаточных классов

4.1 Разработка модели отказоустойчивого модулярного нейрокомпьютера на основе проекций модулярного числа

Обобщенная вычислительная модель отказоустойчивого нейрокомпьютера на основе проекций модулярного числа представлена на рисунке 4.1.

Предложенная модель нейрокомпьютера в СОК содержит:

– мультинейропроцессор, содержащий n нейропроцессоров по модулям p_i $(i = 1, 2, ..., n)$ реализующих вычислительные модели модулярных итераций [10];

– преобразователь ПСС–СОК;

– преобразователь СОК–ПСС;

– блок обнаружения, локализации и коррекции одиночной ошибки на основе проекций модулярного числа, базируется на обратном преобразовании из СОК в ПСС.

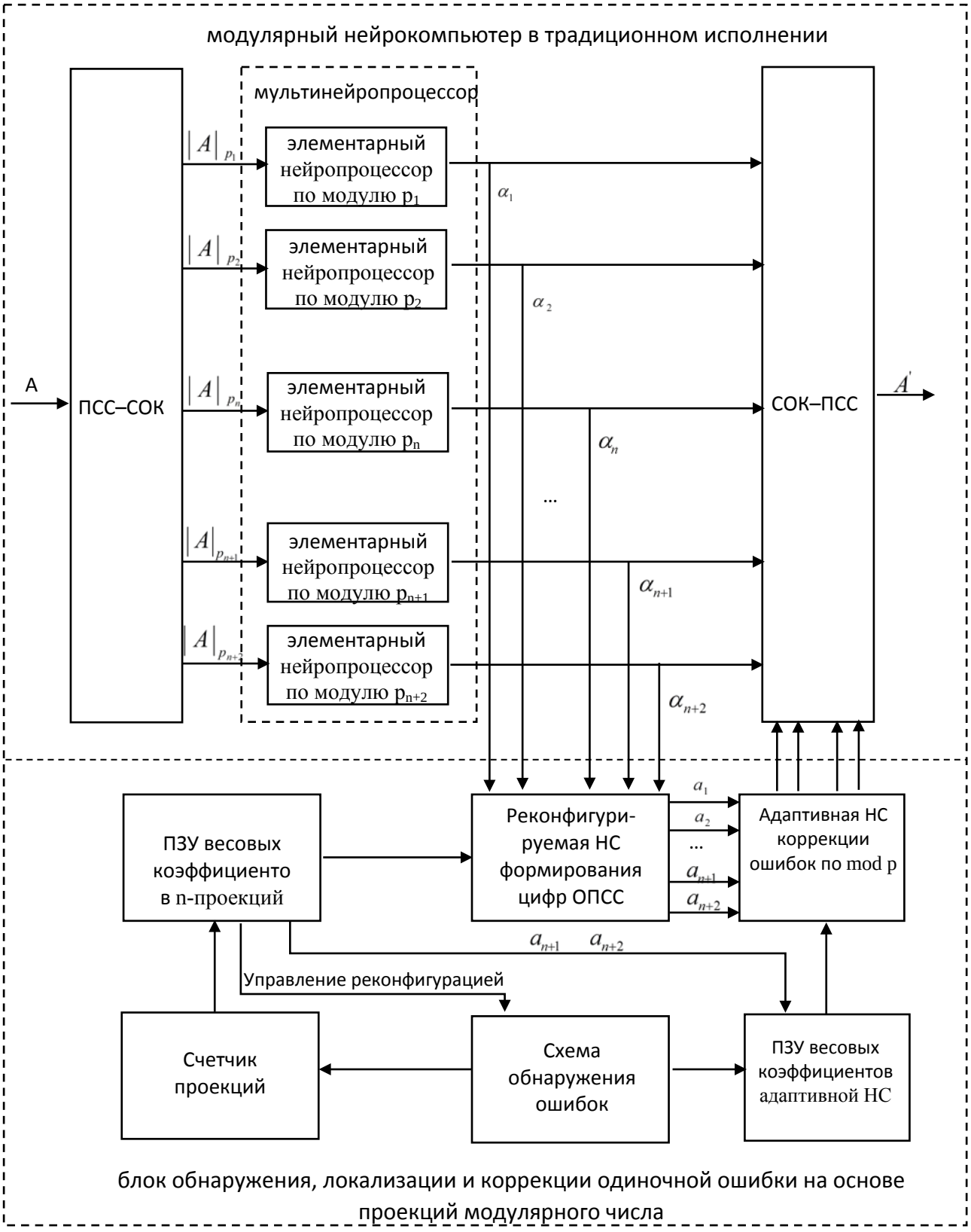


Рисунок 4.1 – Обобщенная модель отказоустойчивого модулярного нейрокомпьютера на основе проекций модулярного числа.

Как перспективный метод коррекции ошибок, обеспечивающий лучшие эксплуатационные характеристики, СОК широко применяется во многих ключевых приложениях для обработки данных. Однако, высокая эффективность обратного перевода в позиционную систему счисления все еще не достигнута из – за необходимости использования дорогостоящих и сложных операторов, которые требуют большого количества вычислительных ресурсов и много времени на выполнение. В случае использования нейрокомпьютера для решения задач безопасности предлагается весьма эффективный обратный преобразователь на основе ИНС, который способен поддерживать пороговые схемы при специальном наборе модулей СОК.

Особенностью предлагаемой нейронной сети является возможность достоверного преобразования остаточного кода в двоичный позиционный код при искаженных или зашумленных остаточных разрядах. Допустимое количество ошибочных разрядов определяется конкретной пороговой структурой сети.

Преимуществом предлагаемой нейронной сети с пороговой структурой перед другими известным сетями является возможность правильно преобразовывать остаточный код в двоичный при наличии ошибочных разрядов, т.е. без их исправления, что ведет к увеличению скорости преобразования, так как из цикла обнаружения ошибок, их локализации, исправлении и преобразования из системы остаточных классов в двоичный код исключается шаг исправления ошибки.

В пороговых (k, t) – схемах множество S делится на m - частей посредством таких алгебраических преобразований, что только по $n \le m$ истинным частям можно восстановить S, где в качестве S используется обычный кортеж числа, представленный в системе остаточных классов. Данное свойство схемы разделения кортежа числа дополняется следующим требованиям: если предъявлены $k + j = t$ долей кортежа, из которых $e < j$ могут отсутствовать или быть ложными (искаженными), то пороговая схема обеспечивает восстановление кортежа по подлинным $k + j – e$ частям, что дает возможность контроля над предоставлением полученного результата сторонам с ложными (искаженными) частями кортежа.

Принцип разделения кортежа основан на применении Китайской теоремы об

остатках (КТО).

Рассмотрим принцип восстановления кортежа в пороговых системах .

Пусть К – кортеж, который имеет ошибочные разряды, состоит из n компонентов. При этом требуется, чтобы имея n ($n > L$), которые являются правильными остатками, можно было восстановить кортеж.

Для решения этой задачи сформулируем условия выбора оснований СОК. Выберем упорядоченный вектор оснований СОК $q < p_1 < p_2 < ... < p_n$. Числа p_i для $i = 1,2,..,n$ взаимно простые, т.е. $(p_i, p_j) = 1$ для $i \neq j$. Каждый остаток кортежа представляется как большое число $p_i (i = 1,2,...n)$. Выберем $q < K$, такое, что $(q, p_j) = 1$ для всех $i = 1,2,...n$. Произведение L наименьших чисел p_i, т.е. $p_1 p_2 ... p_L > q p_{n-L+2} p_{n-L+2} ... p_n$. Если $R = p_1 p_2 ... p_L$, тогда $\frac{R}{q}$ должно быть больше, чем произведение любых $L-1$ чисел p_i.

Выберем случайное число r в интервале $\left[0, \left(\frac{R}{q} - 1\right)\right]$ и вычислим $K' = K + rq$, чтобы K' попало в интервал $[0, R-1]$. Для восстановления кортежа K нейронной сети с пороговой структурой необходима информация в виде r и $K_i \equiv K' \bmod p_i$, $i = 1,2,...n$.

Пример 4.1. Для простоты вычислений будем использовать небольшие числа, хотя на практике применяются очень большие числа. Пусть кортеж в позиционной системе $K = 9$, число правильных остатков $L = 2$, число всех остатков $n = 3$. Выберем взаимно простые числа $q = 13$, $p_1 = 17$, $p_2 = 19$, $p_3 = 23$. Проверим условие $R = p_1 p_2 > q p_3 = 17 \cdot 19 > 13 \cdot 23 = 323 > 299$, условие выполняется. Выберем случайное число r в интервале $\left[0, \left(\frac{p_1 p_2}{q} - 1\right)\right] = [0,23]$, например $r = 2$. Найдем $K' = K + qr = 9 + 2 \cdot 13 = 35$. Распределяемые числа r и K_i' определяются как $K_i \equiv K_i' \bmod p_i, i = 1,2,...,n$. Итак, $K_1 \equiv 35 \bmod 17 = 1$, $K_2 \equiv 35 \bmod 19 = 16$, $K_3 \equiv 35 \bmod 23 = 12$. По любым двум их этих чисел можно восстановить кортеж K, а

в общем случае количество выборок C_n^L определяется сочетанием из n элементов по L - подмножество мощности L исходного конечного множества мощности n. Число сочетаний из n элементов по L обозначается C_n^L и равно $C_n^L = \frac{n!}{L!(n-L)!}$. Например, для K_1 и K_3, имеем $K_1^{'} \equiv 1(\mathrm{mod}17)$, $K_3^{'} \equiv 12(\mathrm{mod}17)$. На основании Китайской теоремы об остатках восстановим $K^{'}$ и определим кортеж числа в позиционной системе счисления

$$K = K^{'} - qr \tag{4.1}$$

Используя метод ортогональных базисов [2], восстановим кортеж K. Вначале определим $K^{'} \equiv (K_1 B_1 + K_1 B_3) \mathrm{mod}(p_1 p_3)$, где B_1 и B_3 - ортогональные базисы. По определению ортогональные базисы могут быть представлены в виде: $\mathrm{B}_1 = m_1 p_3, B_3 = m_3 p_1$, где m_1 и m_3 - веса базисов, соответственно B_1 и B_3. Причем m_1 и m_3 выбираются из сравнений $m_1 p_3 \equiv 1 \mathrm{mod}\, p_1, m_3 p_1 \equiv 1 \mathrm{mod}\, p_3$. Ввиду малости величин оснований решим сравнения путем подбора m_1 и m_3, $m_1 = 3$ и $m_3 = 19$. Тогда, $\mathrm{B}_1 = 3 \cdot 23 = 69$, $\mathrm{B}_3 = 19 \cdot 17 = 323$, $K^{'} = 1 \cdot 16 + 12 \cdot 323 \equiv 394 \,\mathrm{mod}\, 391 = 35$. Отсюда $K = 35 - 2 \cdot 13 = 9$. Таким образом, восстановленный кортеж $K = 9$.

Недостаток рассмотренного метода заключается в том, что приходится иметь дело с большими числами $P_{C_n^L} = p_1 p_2 ... p_L$, где $i = [1, L]$ и, кроме того, действия сложения и умножения надо выполнять в позиционной системе счисления, а полученный результат необходимо вводить в диапазон вычитаемой величины, кратной $P_{C_n^L}$.

На рисунке 4.2 представлена нейронная сеть с пороговой (*k, t*) структурой для преобразования остаточного кода в двоичный позиционный код.

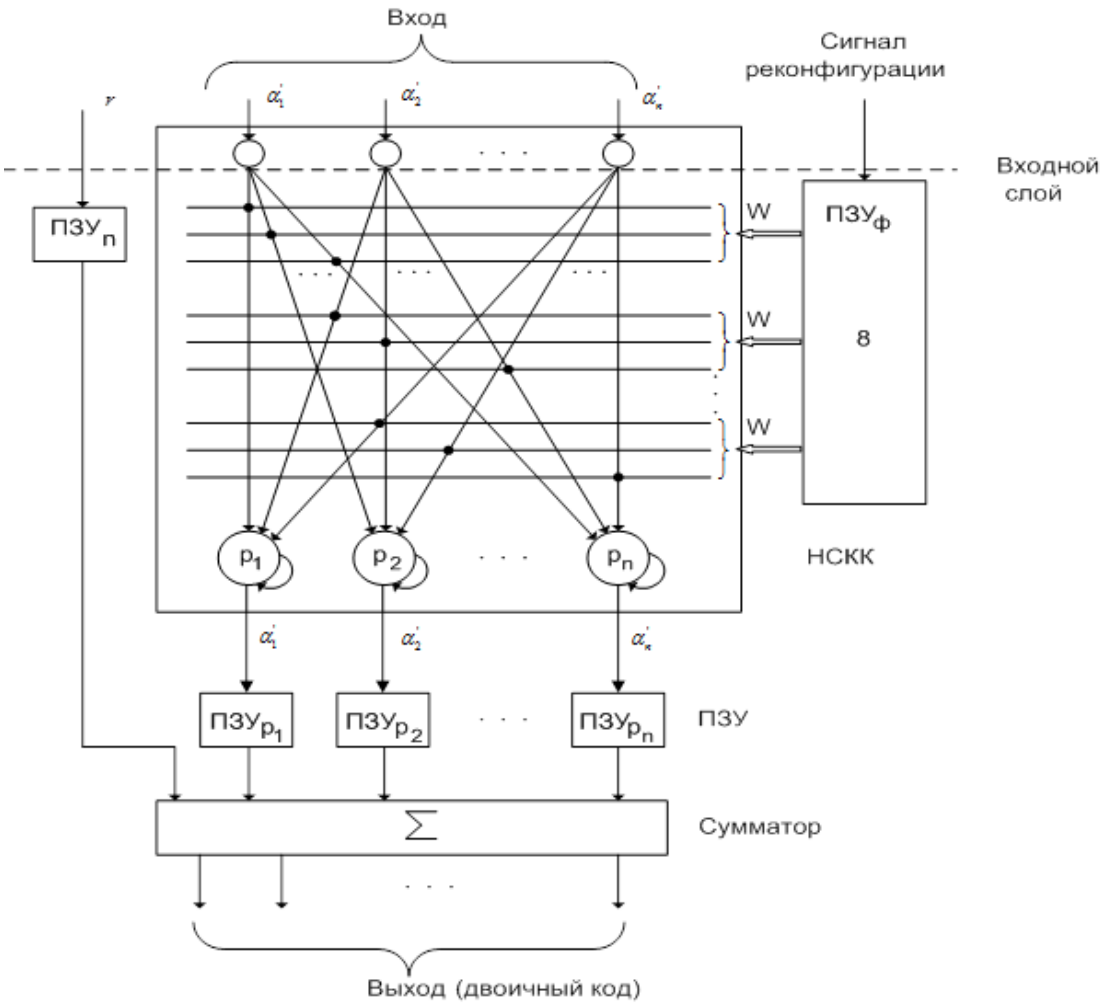


Рисунок 4.2 – Нейронная сеть с пороговой *(k, t)* структурой для преобразования остаточного кода в двоичный позиционный код

Нейронная сеть с пороговой структурой (*k, t*) для преобразования остаточного кода в двоичный позиционный код содержит входной слой нейронов, *n* – нейронных сетей конечного кольца (НСКК), n – постоянных запоминающих устройств (ПЗУ), сумматор, выход нейронной сети P_i, весовые коэффициенты *w*, ПЗУф весовых коэффициентов при реконфигурации нейронных сетей конечного кольца, ПЗУп поправки выходного двоичного разряда, сигнала реконфигурации и сигнала поправки r.

Функционирование нейронной сети зависит от весовых коэффициентов между слоями, которые определяются заранее и хранятся в $ПЗУ_ф$ формирования весовых коэффициентов.

Процесс вычисления коэффициентов ОПСС осуществляется активацией определенных входов НСКК, которые выбраны с помощью весовых коэффициентов. Таким образом, реконфигурация сети осуществляется весовыми коэффициентами нейронных связей на активированных входах нейронной сети.

Весовые коэффициенты определяются заранее и помещаются в $ПЗУ_ф$ формирования весовых коэффициентов при реконфигурации, и в зависимости от поступающих сигналов выдаются необходимые весовые коэффициенты для конкретных вариантов, соответственно, определяя структуру нейронной сети, в которой при определенных внешних параметрах достигается надежное и быстрое вычисление двоичного кода кортежа.

НСКК p_i преобразуют числа $K_i^{'}$, представленные в остаточном коде в код ОПСС. Весовой коэффициент, связанный с каждой цифрой ОПСС $a_i^{'}$ равен $p_1 p_2 ... p_{i-1}$. Такая система имеет тот же диапазон представления, что и СОК $P = p_1 p_2 ... p_n$. Коэффициенты кода ОПСС являются адресами памяти, где хранятся значения произведений $a_i^{'} p_1 p_2 ... p_{i-1}$. Двоичные эквиваленты значений $a_i^{'} p_1 p_2 ... p_{i-1}$, хранящиеся в памяти, поступают на вход сумматора, где по остаткам восстанавливается двоичное число. Для определения кода кортежа на вход сумматора поступает поправка, представляющая произведение чисел - rq, которая формируется $ПЗУ_n$, а адресом является число r.

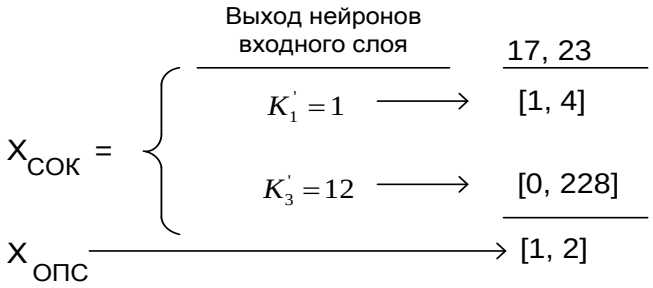
Распределенные числа $\alpha_1^{'}, \alpha_2^{'}, ..., \alpha_L^{'}$, представленные в СОК своими остатками $\alpha_i^{'}$ (вход 1) по модулям $p_i (i = 1,2,..., L)$ поступают на входной слой нейронов , а число r поступает на адресные входы $ПЗУ_n$. Выбранный вариант реконфигурации C_n^L определяется сигналом реконфигурации, подаваемый на вход $ПЗУ_ф$. Входной слой связан с L НСКК.

Кроме того, на вход сумматора подается число - rq в дополнительном коде, выбранном по адресу r из $ПЗУ_n$. Дополнительный код использован для замены операции вычитания операцией сложения. Рассмотрим числовую интерпретацию, реализованную моделью, приведенной на рисунке 4.2. Число всех остатков n = 3,

а число правильных остатков $L = 2$. При этом, распределяемые числа $K_i^{'}$ поступают на вход нейронов входного слоя, а r – на вход ПЗУ$_{\text{n}}$ в качестве адресных входов. На вход ПЗУ$_{\text{ф}}$ поступает сигнал реконфигурации. В соответствии с $K_1^{'} = 1$, $K_3^{'} = 12$ ПЗУ$_{\text{ф}}$ осуществляет конфигурацию нейронной сети путем подачи соответствующих весовых коэффициентов, которые определяются на основе значений $P_{C_n^j} = B_{C_n^L}$, где $L = 2,3,...,n-1$; $j \in \{2,3,...,L\}$; $L \subset n$; C_n^L - число сочетаний из n по L. На основе выражения ОПСС $B_{C_n^L} = b_{i1} + b_{i2} p_1$ определим b_{ij}, тогда $b_{11} = 1$, $b_{13} = 4$ и $b_{21} = 0$, $b_{23} = 19$. Представим b_{ij} в виде таблицы, тогда

$$\begin{vmatrix} b_{11} & b_{13} \\ b_{21} & b_{12} \end{vmatrix} = \begin{vmatrix} 1 & 4 \\ 0 & 19 \end{vmatrix}.$$

Выражение для данной конфигурации системы $K^{'} = b_1 + b_2 17$, тогда получим значения для коэффициентов ОПСС



Итак, $K^{'} = 1 + 2 \cdot 17 = 35$.

Значения $X_{опс}$ = [1, 2] являются адресными входами ПЗУ$_1$ и ПЗУ$_3$. На выходе ПЗУ$_1$ формируется число, равное 1, а на выходе ПЗУ$_3$ – число, равное $2 \cdot 17 = 34$. В сумматоре эти значения суммируются 1 + 34 = 35 из которого вычисляется значение $rq = 2 \cdot 13 = 26$. Все операции в сумматоре выполняются в дополнительном коде. Так как используется положительные целые числа, то дополнительный код числа, равен самому числу. Итак, код кортежа равен K = $35 - 26 = 9$.

Нейронная сеть с пороговой (k, t) структурой для преобразования остаточного кода в двоичный позиционный код работает следующим образом.

Входное число $A=(\alpha_1',\alpha_2',...,\alpha_n')$, представленное в СОК своими остатками α_i' по модулям $p_i(i=1,2,...,n)$ поступает на входной слой. Входной слой связан с n НСКК в соответствии с заданной реконфигурацией, которая зависит от одного внешнего параметра и адаптируется к нему посредством загрузки весовых коэффициентов *w*, формируемых ПЗУ$_ф$ по сигналу реконфигурации. НСКК p_i реализует вычислительную модель, на выходе которых формируются коэффициенты ОПСС $a_i'(i=1,n)$ числа *x*. Коэффициенты a_i' являются адресными входами ПЗУр$_i$, где хранятся двоичные эквиваленты $a_i' p_1 p_2 ... p_{i-1}$. Считанные из ПЗУр$_i$ двоичные коды, соответствующие произведениям $a_i' p_1 p_2 ... p_{i-1}$ суммируются взвешенным сумматором с учетом поправки - rq, формируемой ПЗУ$_n$ по сигналу поправки. На выходе формируется двоичный код кортежа числа. Время преобразования определяется *n* + 1 тактами синхронизации.

4.2 Разработка обобщенной модели отказоустойчивого модулярного нейрокомпьютера на основе расширения системы остаточных классов

4.2.1 Разработка метода и алгоритма параллельного расширения остатков по вновь введенным модулям СОК

Метод проекций позволяет установить в цифре по какому из модулей системы произошла ошибка, т.е. обеспечивает ее локализацию. Однако, как и сам алгоритм определения ошибочной цифры, так и последующая ее коррекция на основе метода проекций представляются малоэффективными и затратными с точки зрения практической реализации. А именно, вычисление каждой из проекций требует выполнения операции восстановления числа из модулярного представления в двоичное и последующих вычислений над числами большой разрядности, что вносит значительный вклад как в аппаратные затраты, так и в задержку работы устройства. Кроме того, такой метод подразумевает исправление лишь ошибочного остатка, а не всего числа, приводя к необходимости восстанавливать число с уже правильными значениями всех остатков. В связи с

этим возникает необходимость искать другие методы реализации механизма обнаружения и коррекции ошибок в СОК.

Одной из альтернатив методу проекций для обнаружения и коррекции ошибок в СОК является синдромное декодирование с вычислением так называемых синдромов ошибки по контрольным основаниям системы. В основе этого метода чаще всего лежит так называемая операция расширения системы оснований. Это немодульная операция, позволяющая по известным остаткам числа, соответствующим некоторым модулям СОК, определить значения остатков этого же числа для других оснований. Обычно для реализации операции расширения системы оснований используют ОПСС, переход к которой носит итеративный характер и может привести к ухудшению быстродействия всего устройства. Рассмотрим метод ускоренного параллельного расширения оснований СОК.

Пример 4.2. Перевести число $A=(1,2,1,4)$, представленное в СОК, в число $A=[a_1,a_2,a_3,a_4]$ в ОПСС.

Решение. Найдем связанные цифры в ОПСС и СОК.

$$A=a_1+a_2(2)+a_3(2\cdot 3)+a_4(2\cdot 3\cdot 5)$$

Используя (4.1) представим B_i в ОПСС и вычислим b_{ij}. Так как СОК не расширена, то $B_i=b_{i1}+b_{i2}(2)+b_{i3}(2\cdot 3)+b_{i4}(2\cdot 3\cdot 5)$, тогда

$$\begin{aligned}
&105=b_{11}+b_{12}(2)+b_{13}(2\cdot 3)+b_{14}(2\cdot 3\cdot 5)\rightarrow b_{11}=1,\ b_{12}=1,\ b_{13}=2,\ b_{14}=3,\\
&70=b_{21}+b_{22}(2)+b_{23}(2\cdot 3)+b_{24}(2\cdot 3\cdot 5)\rightarrow b_{21}=0,\ b_{22}=2,\ b_{23}=1,\ b_{24}=2\\
&126=b_{31}+b_{32}(2)+b_{33}(2\cdot 3)+b_{34}(2\cdot 3\cdot 5)\rightarrow b_{31}=0,\ b_{32}=0,\ b_{33}=1,\ b_{34}=4\\
&120=b_{41}+b_{42}(2)+b_{43}(2\cdot 3)+b_{44}(2\cdot 3\cdot 5)\rightarrow b_{41}=0,\ b_{42}=0,\ b_{43}=0,\ b_{44}=4.
\end{aligned} \qquad (4.2)$$

Для вычисления цифр ОПСС используем выражение (4.2). Процесс преобразования числа из СОК в ОПСС приведен в таблице 4.1.

Таблица 4.1 Перевод чисел из СОК в ОПСС

Вычеты числа A по модулю p_i	Модули			
	$p_1 = 2$	$p_2 = 3$	$p_3 = 5$	$p_4 = 7$
$\alpha_1 = 1$	$\alpha_1 b_{11} = 1 \cdot 1 = 1$	$\alpha_1 b_{12} = 1 \cdot 1 = 1$	$\alpha_1 b_{13} = 1 \cdot 2 = 2$	$\alpha_1 b_{14} = 1 \cdot 3 = 3$
$\alpha_2 = 2$	$\alpha_2 b_{21} = 2 \cdot 1 = 0$	$\alpha_2 b_{22} = 2 \cdot 2 = 4$	$\alpha_2 b_{23} = 2 \cdot 1 = 2$	$\alpha_2 b_{24} = 2 \cdot 2 = 4$
$\alpha_3 = 1$	$\alpha_3 b_{31} = 1 \cdot 2 = 0$	$\alpha_3 b_{32} = 1 \cdot 0 = 0$	$\alpha_3 b_{33} = 1 \cdot 1 = 1$	$\alpha_3 b_{34} = 1 \cdot 4 = 4$
$\alpha_4 = 4$	$\alpha_4 b_{41} = 4 \cdot 0 = 0$	$\alpha_3 b_{42} = 4 \cdot 0 = 0$	$\alpha_4 b_{43} = 4 \cdot 0 = 0$	$\alpha_3 b_{44} = 4 \cdot 4 = 16$
Коэффициенты ОПСС числа A	$a_1 = \left\lvert \sum_{\substack{i=1 \\ j=1}}^{4} \alpha_i b_{ij} \right\rvert_2 = 1$	$a_2 = \left\lvert \sum_{\substack{i=1 \\ j=1}}^{4} \alpha_i b_{ij} \right\rvert_3 = 2$	$a_3 = \left\lvert \sum_{\substack{i=1 \\ j=1}}^{4} \alpha_i b_{ij} \right\rvert_5 = 1$	$a_4 = \left\lvert \sum_{\substack{i=1 \\ j=1}}^{4} \alpha_i b_{ij} \right\rvert_7 = 0$

Стрелками показаны межразрядные переносы. Перенос старшего разряда игнорируется.

Действительно, $A = (\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (|11|_2, |11|_3, |11|_5, |11|_7) = (1,2,1,4)_{СОК} = [1,2,1,0]_{ОПСС}$.

Пример 4.3. Пусть задана система с тремя модулями $p_1 = 2,\ \ p_2 = 3,\ \ p_3 = 5$, тогда $P = 2 \cdot 3 \cdot 5 = 30$.Возьмем $A = 11 = (1,\ 2,\ 1)$. Добавим $p_4 = 7$. Найдем $|A|_7$.

Решение.

Вначале определим цифры ОПСС аналогично тому, как сделано в примере (4.2), но в процесс преобразования включим значение $|A|_7$.

Набор констант b_{ij} приведен в (4.2) и задается матрицей $\begin{pmatrix} 1 & 1 & 2 & 3 \\ 0 & 2 & 1 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 4 \end{pmatrix}$.

Процесс вычисления коэффициентов ОПСС приведен в таблице 4.2.

Таблица 4.2 Расширение на одно основание

Вычеты числа A по модулю p_i	Модули			
	$p_1 = 2$	$p_2 = 3$	$p_3 = 5$	$p_4 = 7$
$\alpha_1 = 1$	1	1	2	3
$\alpha_2 = 2$	0	4	2	4
$\alpha_3 = 1$	0	0	1	4
$\alpha_4 = \lvert A \rvert_7$	0	0	0	$4\lvert A \rvert_7$
Коэффициенты ОПСС числа $A = [a_1, a_2, a_3, a_4]$	1	2 →	1 →	$5 + 4\lvert A \rvert_7$

При расширении базы множитель $|A|_7$ не зависит от остаточного представления $A = (1,2,1)$, следовательно, множитель, используемый на последнем этапе преобразования, определяется мультипликативной инверсией модуля, тогда $\alpha_4 = |A|_7 = 2 \cdot \left|\frac{1}{4}\right|_7 = 2 \cdot 2 = 4$,так как $a_4 = \left|4|A|_7 + 5\right|_7 = 0$ и $\left|4|A|_7\right|_7 = p_4 - 5 = 2$.

Пример 4.4. Пусть задана система оснований (модулей) СОК $p_1 = 2,\ \ p_2 = 3,\ \ p_3 = 5$. Расширим систему оснований, добавляя $p_4 = 7,\ \ p_5 = 11$.

В примере 4.3 показано расширение на модуль $p_4 = 7$. Аналогично, как в примере 2, проведем вычисления при расширении на $p_5 = 11$, тогда $P = 2 \cdot 3 \cdot 5 \cdot 11 = 330$, а базисы, соответственно: $B_1 = 165,\ B_2 = 220,\ B_3 = 66,\ B_4 = 30$.

Используя ОПСС определим набор констант b_{ij} с учетом расширяющего модуля $p_5 = 11$, тогда:

$$165 = \bar{b}_{11} + \bar{b}_{12}(2) + \bar{b}_{13}(2\cdot 3) + \bar{b}_{15}(2\cdot 3\cdot 5) \to \bar{b}_{11} = 1, \bar{b}_{12} = 1, \bar{b}_{13} = 2, \bar{b}_{15} = 5$$
$$220 = \bar{b}_{21} + \bar{b}_{22}(2) + \bar{b}_{23}(2\cdot 3) + \bar{b}_{25}(2\cdot 3\cdot 5) \to \bar{b}_{21} = 0, \bar{b}_{22} = 2, \bar{b}_{23} = 1, \bar{b}_{25} = 7$$
$$66 = \bar{b}_{31} + \bar{b}_{32}(2) + \bar{b}_{33}(2\cdot 3) + \bar{b}_{35}(2\cdot 3\cdot 5) \to \bar{b}_{31} = 0, \bar{b}_{32} = 0, \bar{b}_{33} = 1, \bar{b}_{35} = 2$$
$$30 = \bar{b}_{51} + \bar{b}_{52}(2) + \bar{b}_{53}(2\cdot 3) + \bar{b}_{55}(2\cdot 3\cdot 5) \to \bar{b}_{51} = 0, \bar{b}_{52} = 0, \bar{b}_{53} = 0, \ \bar{b}_{55} = 7$$

Представим выражение (4.2) с учетом констант по модулю $p_5 = 11$, тогда получим выражение (4.3).

$$\begin{aligned}
&\bar{b}_{11} = 1, \bar{b}_{12} = 1, \bar{b}_{13} = 2, \bar{b}_{14} = 3, \bar{b}_{15} = 5\\
&\bar{b}_{21} = 0, \bar{b}_{22} = 2, \bar{b}_{23} = 1, \bar{b}_{24} = 2, \bar{b}_{25} = 7\\
&\bar{b}_{31} = 0, \bar{b}_{32} = 0, \bar{b}_{33} = 1, \bar{b}_{34} = 4, \bar{b}_{35} = 2\\
&\bar{b}_{41} = 0, \bar{b}_{42} = 0, \bar{b}_{43} = 0, \bar{b}_{44} = 4, \bar{b}_{45} = -\\
&\bar{b}_{51} = 0, \bar{b}_{52} = 0, \bar{b}_{53} = 0, \bar{b}_{54} = -, \bar{b}_{55} = 7
\end{aligned} \qquad (4.3)$$

где $\bar{b}_{45}$ и $\bar{b}_{54}$ в расширении не используются.

Пусть дана система оснований $p_1 = 2, p_2 = 3, p_3 = 5$. Расширим систему путем добавления оснований $p_4 = 7$ и $p_5 = 11$, и представим число $A = (1,2,2)$ в расширенной системе как $A = (1,2,2,|A|_7,|A|_{11})$.

Процесс решения приведен в таблице 4.3.

Таблица 4.3 Расширение на два основания

Вычеты числа A по модулю p_i	Модули				
	$p_1 = 2$	$p_2 = 3$	$p_3 = 5$	$p_4 = 7$	$p_5 = 11$
$\alpha_1 = 1$	1	1	2	3	5
$\alpha_2 = 2$	0	4	2	4	3
$\alpha_3 = 2$	0	0	2	8	4

Продолжение таблицы 4.3

$\alpha_4 = \lvert A \rvert_{p4=7}$	0	0	0	$4\lvert A \rvert_7$	–
$\alpha_5 = \lvert A \rvert_{p5=11}$	0	0	0	–	$7\lvert A \rvert_{11}$
Коэффициенты ОПСС числа A	$a_1 = 1$	$a_2 = 2$	$a_3 = 2$	$a_4 = 2 + 4\lvert A \rvert_7$	$a_5 = 2 + 7\lvert A \rvert_{11}$

Между расширяющими модулями переноса нет по причине отсутствия связи между ними.

После сложения цифр по модулю p_i получим $A = \left[1,\ 2,\ 4,\ 2 + 4\lvert A \rvert_7,\ 2 + 7\lvert A \rvert_{11}\right]$, но так как $a_4 = 2 + 4\lvert A \rvert_7$, $a_5 = 2 + 7\lvert A \rvert_{11}$ и по условию $a_4 = 0$ и $a_5 = 0$, тогда

$$\left|4\lvert A \rvert_7\right|_7 = p_4 - 2 = 5,\ \text{откуда}\ \lvert A \rvert_7 = \left|5 \cdot \left|\frac{1}{4}\right|_7\right|_7 = \lvert 5 \cdot 2 \rvert_7 = 3,$$

$$\left|2 + 7\lvert A \rvert_{11}\right|_{11} = p_4 - 2 = 9\ \text{и}\ \lvert A \rvert_{11} = \left|9 \cdot \frac{1}{7}\right|_{11} = \lvert 9 \cdot 8 \rvert_{11} = 6.$$

Тогда расширенное представление числа будет равно $A = (1,\ 2,\ 2,\ 3,\ 6)$.

4.2.2. Численный метод коррекции ошибок на основе расширения модулей СОК

Рассмотрим пример, иллюстрирующий обнаружение, локализацию и исправление одиночной ошибки в системе с двумя контрольными основаниями. Возьмем систему оснований как в примере 4.4 $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11$. Основания p_4 и p_5 будем считать избыточными (контрольными). Рабочий диапазон системы определяется как $P = p_1 \cdot p_2 \cdot p_3 = 2 \cdot 3 \cdot 5 = 30$, и полный диапазон как $\mathrm{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2310$.

Пример 4.5. Передано число $A=(1,2,2,3,6)$. Принято вместо него число $\overline{A}=(1,2,3,3,6)$, т.е. произошла ошибка по третьему информационному основанию. Задача состоит в обнаружении, локализации и исправлении одиночной ошибки.

Избыточные разряды по четвертому и пятому основаниям равны, соответственно, 3 и 6, и допустим, что они абсолютно верны.

На основании информационных остатков определим синдромы ошибок δ_1 и δ_2. Метод обнаружения, исправления и локализации ошибок в СОК представляется следующим образом.

Контролируемое число $a_1, a_2, ..., a_n, a_{n+1}, a_{n+2}$ делится на две части: информационную, в которую входят остатки по информационным основаниям $a_1, a_2, ..., a_n$, и контрольную часть, в которую входят остатки по избыточным (контрольным) основаниям a_{n+1}, a_{n+2}.

Далее по остаткам информационных каналов $A=(1,2,3)$ определяем остатки по избыточным основаниям p_4 и p_5 на основе расширения системы оснований. Воспользуемся таблицей 4.3, тогда третья строка будет иметь вид не $[0, 0, 2, 8, 4]$, а $[0, 0, 3, 12, 6]$, так как $\alpha_3=3$.

При выполнении операции согласно таблице 4.3 аналогично примеру 4.5 получим

$$4|A|_7+6=0,\ 7|A|_{11}+4=0.$$

Отсюда

$$\left|4|A|_7\right|_7=1,\ \alpha'_{n+1}=|A|_7=\left|\frac{1}{4}\right|_7=2.$$

$$\left|7|A|_{11}\right|_{11}=7,\ \alpha'_{n+2}=1.$$

Используя найденные остатки, определяем синдром ошибки:

$$\delta_1\equiv(a_{n+1}-a'_{n+1})(\mathrm{mod}\ p_{n+1})=3-2=1, \tag{4.4}$$

$$\delta_2\equiv(a_{n+2}-a'_{n+2})(\mathrm{mod}\ p_{n+2})=6-1=5 \tag{4.5}$$

Так как δ_1 и δ_2 не равны нулю, то произошла ошибка Δ_i в информационных основаниях.

По величинам δ_1, δ_2 формируется константа ошибок таким образом, чтобы при ее сложении с информационным разрядом контролируемого числа A, имевшая место ошибка в числе устранялась. Заметим, что если контролируемое число не будет содержать ошибки, то величины δ_1 и δ_2 равны нулю.

Для локализации ошибки необходимо провести сравнение разрядов исходного числа с исправленным числом и по тому разряду, где сравнение не выполняется, определяется ошибочный разряд.

Константы ошибок для СОК с модулями $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, $p_5 = 11$ приведем в таблице 4.4.

Таблица 4.4 Константы ошибок для СОК с основаниями $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, $p_5 = 11$

Ошибка Δ_1 по основанию p_1	δ_1, δ_2	Ошибка Δ_2 по основанию p_2	δ_1, δ_2	Ошибка Δ_3 по основанию p_3	δ_1, δ_2
0, 0, 0	0, 0	0, 0, 0	0, 0	0, 0, 0	0, 0
1, 0, 0	6, 7	0, 1, 0	1, 2	0, 0, 1	4, 9 6, 6
		0, 2, 0	4, 1	0, 0, 2	3, 4 5, 1
				0, 0, 3	4, 7 2, 10
				0, 0, 4	1, 5 3, 2

Таблица 4.4 содержит две колонки: колонка Δ_i - «ошибка по основанию p_i» отражает глубину ошибки, равной величине p_i, и колонка «δ_1, δ_2» отражает синдром одиночной ошибки.

В соответствии с $\delta_1 = 1,\ \delta_2 = 5$ нижней строки таблицы 4.4 определяется величина ошибки $\Delta_3 = (0,\ 0,\ 4)$, которая складывается с контролируемым числом $\overline{A} = (1,\ 2,\ 3) + (0,\ 0,\ 4) = (1,\ 2,\ 2)$.

По величине ошибки $(0,\ 0,\ 4)$ локализуется ошибочный разряд, им будет разряд по $\bmod p_3$. Использование синдрома ошибки позволяет все процедуры по обнаружению, локализации и исправлению ошибок объединить в одну процедуру, что позволяет уменьшить время коррекции ошибок и повысить эффективность.

4.2.3 Модель отказоустойчивого модулярного нейрокомпьютера с коррекцией одиночной ошибки на основе расширения системы оснований СОК с использованием синдрома ошибок

На рисунке 4.3 представлена модель отказоустойчивого модулярного нейрокомпьютера с обнаружением, локализацией и коррекцией ошибок в СОК, состоящего из двух частей: модулярного нейрокомпьютера в традиционном исполнении и нейронной сети для обнаружения, локализации и коррекции ошибок в СОК.

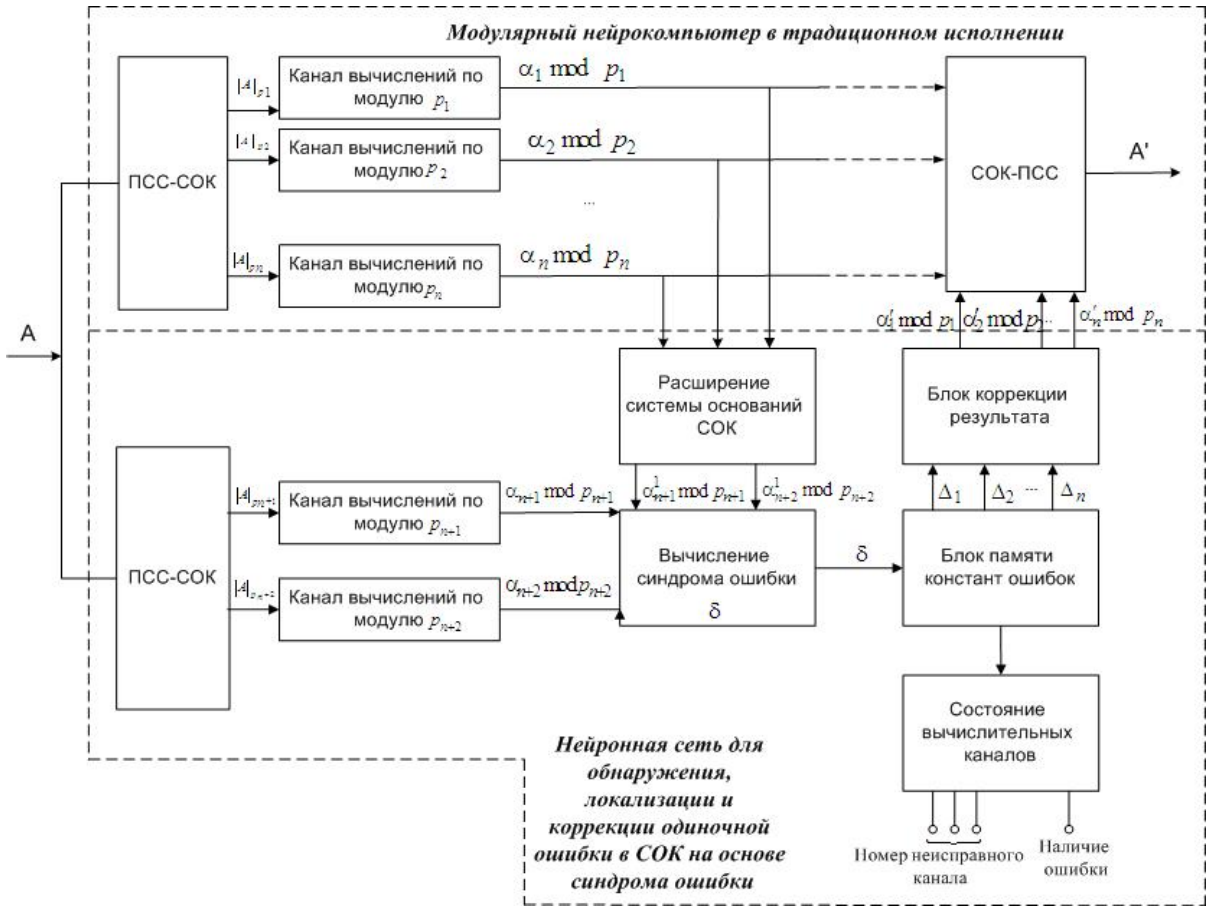


Рисунок 4.3 – Модель отказоустойчивого модулярного нейрокомпьютера на основе синдрома ошибок

В свою очередь модулярный нейрокомпьютер в традиционном исполнении состоит из входного преобразователя ПСС–СОК, вычислительных каналов мультинейропроцессора по модулю p_i и выходного преобразователя СОК–ПСС, где A – входные данные, и $A^{'}$ – выходные данные. Вычислительные каналы представляют собой элементарные нейропроцессоры.

Для достижения требуемого уровня отказоустойчивости нейрокомпьютера в общую схему введена нейронная сеть для обнаружения, локализации и коррекции ошибок, состоящая из преобразователей ПСС–СОК и вычислительных каналов по контрольным модулям p_{n+1} и p_{n+2}, блока расширения СОК, блока вычисления синдрома ошибок δ, блока памяти констант ошибок Δ_i, блока коррекции результата и блока, сигнализирующего состояние вычислительных каналов.

Входной информацией для нейронной сети являются остатки по модулям СОК $p_1, p_2, ..., p_{n+1}, p_{n+2}$,а выходной информацией является корректирующее слово для коррекции ошибок в виде констант ошибок Δ_i по модулям $p_1, p_2, ..., p_n$.

Реализация нейронной сети для обнаружения, локализации и коррекции ошибок основана на базе НСКК и приведена на рисунке 4.4, где: блок расширения СОК состоит из НСКК 4, 5, 6; блок вычисления синдрома ошибок состоит из НСКК 10, 11,27 и 28; блок памяти констант ошибок НСКК 14; блок коррекции результатов состоит из НСКК 21 и блок, сигнализирующий состояние вычислительных каналов, состоит из шины 15 выхода блока констант ошибок и элемента ИЛИ – 19.

Нейронная сеть состоит из входного слоя нейронов 2, 23, входного слоя 24, предназначенного для хранения остатков числа по рабочим и контрольным основаниям в течение времени обнаружения ошибки, вход которой соединен со входами $1(a_1, a_2, ..., a_n)$ и входами $25(a_{n+1}, a_{n+2})$ нейронной сети; нейронной сети расширения системы оснований 8, предназначенной для вычисления остатков чисел по контрольным основаниям, входы которой соединены с выходами нейронов 2, хранящими остатки по рабочим основаниям; нейронных сетей конечного кольца 10, 11, предназначенных для вычисления синдромов ошибки по контрольным основаниям, первые входы которых соединены соответственно с выходами нейронной сети вычисления остатков по контрольным основаниям, а вторые – соответственно с выходами нейронов 23, хранящие остатки по контрольным основаниям, входы которых подключены к шине 25; блок памяти 14, предназначенный для хранения констант, вход которого соединен с выходами нейронных сетей конечного кольца 10, 11; нейронных сетей конечного кольца 21, предназначенных для получения исправленного числа путем суммирования неправильного числа с константой ошибки, первые входы которых соединены соответственно с выходами блока памяти, шины 15, которые являются выходными шинами 16, 17 и 18, формирующие номера неисправных модулей и входом элемента ИЛИ 19, выход 20 которого формирует сигнал "есть ошибка", а

вторые - с выходами нейронов 2, а выходные сигналы 22 нейронных сетей конечного кольца 21 являются выходами нейронной сети исправления ошибок.

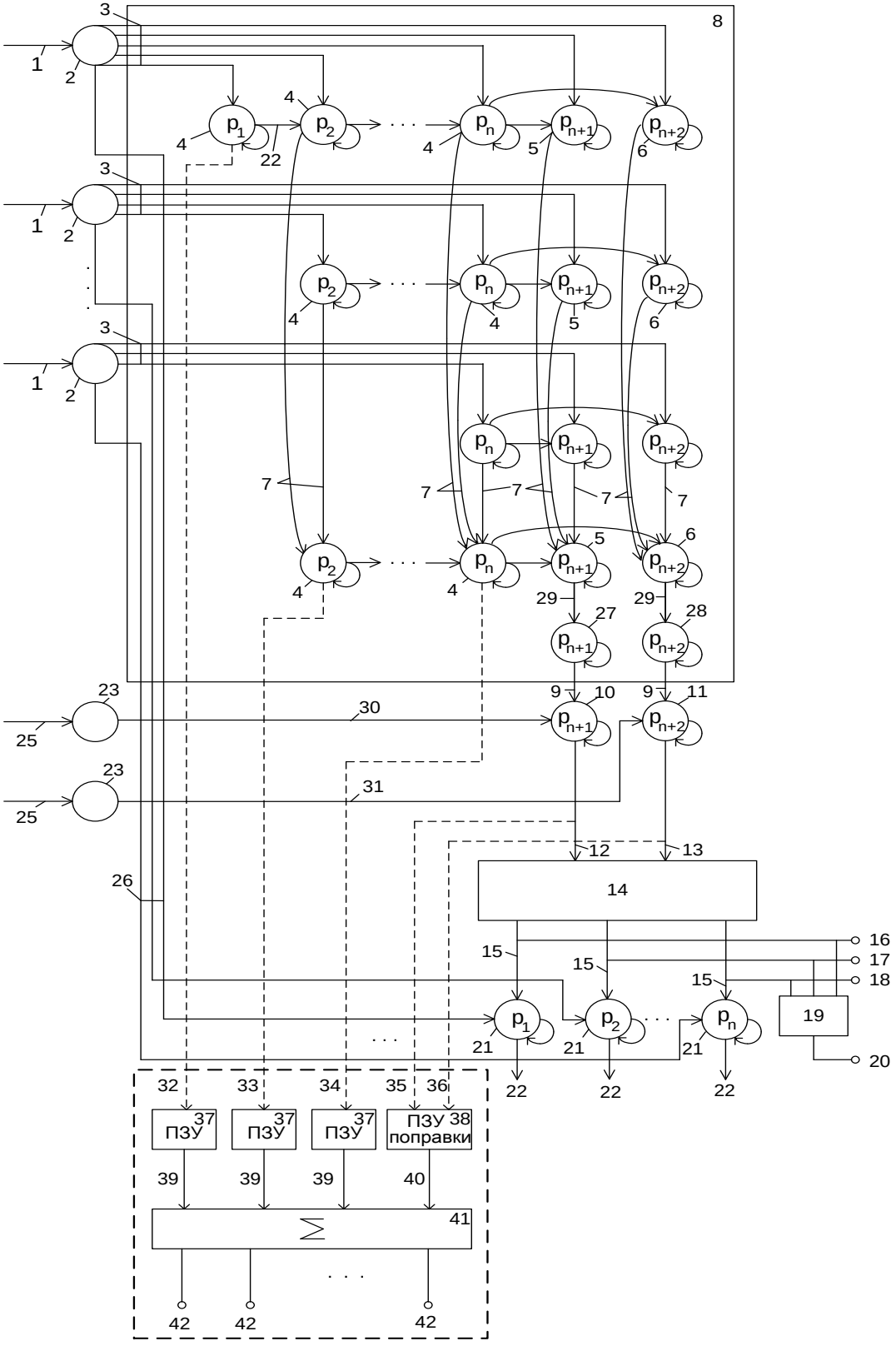


Рисунок 4.4 – Нейронная сеть для обнаружения, локализации и коррекции ошибок

Работа нейронной сети для обнаружения, локализации и исправления ошибок в системе остаточных классов осуществляется следующим образом.

На входы 1, 25 нейронов 2 и 23 нейронной сети для обнаружения, локализации и исправления ошибки в системе остаточных классов подается контролируемое число $A = (\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n, \alpha_{n+1}, \alpha_{n+2})$. С выходов нейронов 2 остатки по рабочим основаниям с весовыми коэффициентами w_{ij} 3 поступают на вход нейронной сети 8 вычисления остатков по контрольным модулям. Нейронная сеть 8 состоит из совокупности нейронных сетей конечного кольца 4, 5, 6. Весовые коэффициенты w_{ij} 3 и w_{ij} 7 нейронов нейронных сетей конечного кольца, выполняющие роль распределенной памяти, определяются, соответственно $w_{ij} = \bar{b}_{ij}$ и $w_{jk} = 1$. Нейронные сети конечного кольца 4, 5 и 6 реализуют вычислительную модель, представленную в таблице 3.4.

Выходные сигналы НСКК 5 и 6 последней строки поступают на вход НСКК 27 и 26, с весовыми коэффициентами w_{kl} 29, равные $\left|\frac{1}{\bar{b}_{i(n+l)}}\right|_{p_{n+l}}$, где l - количество расширяемых модулей и принимает значение 1, 2, …

Выходные сигналы 9 НСКК 27 и 28 будут отрицательными значениями: $-\alpha'_{n+1}$ и $-\alpha'_{n+2}$, которые поступают, соответственно, на входы 9 НСКК 10 и 11, а на вторые входы поступают значения α_{n+1} и α_{n+2} по шинам, соответственно 30 и 31. НСКК 10 и 11 реализуют вычислительную модель:

$$\delta_1 = \alpha_{n+1} - (-\alpha'_{n+1});$$

$$\delta_2 = \alpha_{n+2} - (-\alpha'_{n+2}).$$

Выходные значения НСКК 10 и 11 по шинам 12 и 13, соответствующим синдромам ошибки поступают на входы блока памяти 14 и выбирают оттуда соответствующую константу, согласно таблице 3.5. Эти константы с выхода блока памяти 14 по шинам 15 поступают на соответствующую шину 16, 17 и 18, которые показывают номер неисправного модуля, а также на вход элемента ИЛИ 19, выход 20 которого показывает наличие ошибки. На входы НСКК 21 поступает сигнал с выхода блока памяти, а на вторые входы поступают, соответственно,

выходные сигналы рабочих каналов, нейроны 2, выходная шина 26, где суммируются с константами ошибок, подобранными таким образом, что при ее сложении с контролируемым числом *A*, имевшая место ошибка в числе устраняется. На выходе НСКК 21 выходная шина 22 формирует исправленное число.

Для преобразования остаточного кода в позиционный с коррекцией ошибок используются коэффициенты ОПСС, которые поступают по шинам 32-34, соответственно, по модулям $p_1 p_2 ... p_n$, а на выходах 35, 36 – синдромы ошибок, которые, соответственно, являются адресами ПЗУ 37 для хранения произведений $a_i p_1 p_2 ... p_{i-1}$ и ПЗУ 38 для хранения констант поправок, представленных в двоичном коде. На рисунке 4.4 введенные шины 32-34 показаны пунктирными линиями. С выхода ПЗУ выбранные значения по шинам 39, 40 поступают на сумматор 41, на выходе которого формируется исправленное число, представленное в позиционной системе счисления.

Для реализации такого решения в нейронную сеть для обнаружения, локализации и исправления ошибок в системе остаточных классов введены *n* – постоянных запоминающих устройств 37, где *n* – число модулей системы остаточных классов, в которых хранятся произведения $a_i p_1 p_2 ... p_{i-1}$, определяемые формулой обобщенной позиционной системы счисления и постоянное запоминающее устройство 38 для хранения констант поправок, определяемые синдромом ошибки, равной разности цифр $\delta_1 = (a_{n+1} - a'_{n+1}) \bmod p_{n+1}$ и $\delta_2 = (a_{n+2} - a'_{n+2}) \bmod p_{n+2}$, где a_{n+1} и a_{n+2} - известные значения по избыточным основаниям p_{n+1} и p_{n+2}, a'_{n+1} и a'_{n+2} - вычисленные значения по избыточным основаниям, которые могут содержать ошибочные разряды.

Адресом запоминающих устройств 37 и 38 являются коэффициенты обобщенной позиционной системы счисления, которые вычисляются при расширении оснований системы остаточных классов с целью определения остатков по контрольным модулям. В режиме обнаружения, локализации и исправления ошибок используются только переносы из младших разрядов в

старшие и разряды ОПСС по контрольным модулям для вычисления синдрома ошибок, а в режиме преобразования кода системы остаточных классов в позиционный код будут использованы разряды ОПСС по рабочим и контрольным основаниям.

Итак, исходными данными для получения позиционного исправленного представления числа являются коэффициенты обобщенной позиционной системы счисления $a_1, a_2, ..., a_n$ и синдромы ошибок δ_1 и δ_2. Рассмотрим метод коррекции ошибок.

Пусть имеем набор модулей $p_1, p_2, ..., p_n, p_{n+1}, p_{n+2}$, в котором $p_1, p_2, ..., p_n$ - рабочие основания, а p_{n+1}, p_{n+2} - контрольные основания. Рассмотрим число $A = (\alpha_1, \alpha_2, ..., \alpha_n, \alpha_{n+1}, \alpha_{n+2})$, в котором остатки $\alpha_1, \alpha_2, ..., \alpha_n$ - остатки по рабочим основаниям. Пусть по одному из рабочих оснований произошла ошибка. Тогда полученное число $\tilde{A} = (\alpha_1, \alpha_2, ..., \tilde{\alpha}_i, ..., \alpha_n, \alpha_{n+1}, \alpha_{n+2}), \quad i = 1,2,..,n+2$ будет отличаться от исходного числа одним из остатков $\tilde{\alpha}_i \neq a_i$. Отбросим в числе $\tilde{A}$ все остатки по контрольным основаниям (сужение диапазона): $\tilde{A}_n = (\alpha_1, \alpha_2, ..., \tilde{\alpha}_i, ..., \alpha_n)$, а затем произведем расширение диапазона по значениям информационных остатков: $\tilde{A}' = (\alpha_1, \alpha_2, ..., \tilde{\alpha}_i, ..., \alpha_n, \alpha'_{n+1}, \alpha'_{n+2})$. Очевидно, что остатки (α'_{n+1}, α'_{n+2}), определенные по остаткам рабочих оснований в представлении числа $\tilde{A}_n$, в случае наличия ошибки $\tilde{\alpha}_i \neq a_i$ не будут равны остаткам (α_{n+1}, α_{n+2}) в представлении числа $\tilde{A}$, что обеспечивается избыточностью СОК. Более того, это гарантирует однозначную определенность при различных величинах и местоположениях ошибок.

Для пояснения, рассмотрим пример.

Пример 4.6. Пусть задана система оснований $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, $p_5 = 11$, в которой представлено число $A = (1, 2, 2, 3, 6)$. Два последних основания будем считать контрольными.

По основанию $p_1 = 2$ может иметь только одна ошибка, в результате которой будет получено число $\tilde{A} = (0, 2, 2, 3, 6)$. Сократив основания до информационных, получим $\tilde{A}_n = (0, 2, 2)$. При расширении диапазона вышеуказанным методом получим

$\tilde{A}_n^{'}$ = (0, 2, 2, 2, 2). Аналогичным способом определим остатки по расширенным основаниям по последним контрольным модулям при возникновении ошибок по остальным рабочим основаниям и результаты сведем в таблицу 4.5. Из полученной таблицы видно, что, для каждой пары значений остатков по контрольным основаниям (α_4, α_5) и ($\alpha_4^{'}$, $\alpha_5^{'}$), можно однозначно указать местоположение ошибки. Т.к. рабочий диапазон представления чисел $P_k = 30$, а число всевозможных ошибок по информационным основаниям $N_{\Delta\alpha} = 7$, то необходимо помнить $P_k N_{\Delta\alpha} = 210$ корректирующих чисел вида $\Delta A = (0,0,...,\Delta\alpha_i,...,0)$ для каждой возможной комбинации ошибок во всем основном диапазоне представления чисел. Аргументами выбора того или иного корректирующего числа ΔA будут являться пары коэффициентов контрольных оснований (α_4, α_5) и ($\alpha_4^{'}$, $\alpha_5^{'}$).

Таблица 4.5 Расширенные основания по контрольным основаниям

Величина ошибки	Исходные (α_4, α_5)	Полученные при расширении диапазона ($\alpha_4^{'}$, $\alpha_5^{'}$).
$\Delta\alpha_1 = 1$, $\alpha_1 = 0$	(3, 6)	(2, 2)
$\Delta\alpha_2 = 1$, $\alpha_2 = 0$	(3, 6)	(6, 5)
$\Delta\alpha_2 = 2$, $\alpha_2 = 1$	(3, 6)	(0, 7)
$\Delta\alpha_3 = 1$, $\alpha_3 = 3$	(3, 6)	(5, 5)
$\Delta\alpha_3 = 2$, $\alpha_3 = 4$	(3, 6)	(4, 0)
$\Delta\alpha_3 = 3$, $\alpha_3 = 0$	(3, 6)	(2, 1)
$\Delta\alpha_3 = 4$, $\alpha_3 = 1$	(3, 6)	(1, 7)

В случае возникновения ошибки по одному из контрольных оснований расширение диапазона автоматически исправит ошибку, т.к. восстановление дополнительных оснований производятся по информационным, которые по условию

являются безошибочными. Таким образом, в данном случае пара (α_4, α_5), (α_4', α_5') также однозначно определит местоположение ошибки (таблица 4.6). Число $P_k N_{\Delta\alpha} = 30 \cdot 16 = 480$, т.е. необходимо помнить 480 корректирующих чисел вида $\Delta A = (0,0,...,\Delta\alpha_i,...,0)$ при выбранной системе оснований для коррекции одиночных ошибок по контрольным основаниям.

Таблица 4.6 Местоположение ошибки

Величина ошибки	Исходные (α_4, α_5)	Полученные при расширении диапазона (α_4', α_5')	Величина ошибки	Исходные (α_4, α_5)	Полученные при расширении диапазона (α_4', α_5')
$\Delta\alpha_4 = 1,\ \alpha_4 = 4$	(4, 6)	(3, 6)	$\Delta\alpha_5 = 1,\ \alpha_4 = 7$	(3, 7)	(3, 6)
$\Delta\alpha_4 = 2, \alpha_4 = 5$	(5, 6)	(3, 6)	$\Delta\alpha_5 = 2, \alpha_4 = 8$	(3, 8)	(3, 6)
$\Delta\alpha_4 = 3,$ $\alpha_4 = 6$	(6, 6)	(3, 6)	$\Delta\alpha_5 = 3,\ \alpha_4 = 9$	(3, 9)	(3, 6)
$\Delta\alpha_4 = 4, \alpha_4 = 0$	(0, 6)	(3, 6)	$\Delta\alpha_5 = 4, \alpha_4 = 10$	(3, 10)	(3, 6)
$\Delta\alpha_4 = 5, \alpha_4 = 1$	(1, 6)	(3, 6)	$\Delta\alpha_5 = 5, \alpha_4 = 0$	(3, 0)	(3, 6)
$\Delta\alpha_4 = 6, \alpha_4 = 2$	(2, 6)	(3, 6)	$\Delta\alpha_5 = 6, \alpha_4 = 1$	(3, 1)	(3, 6)
			$\Delta\alpha_5 = 7, \alpha_4 = 2$	(3, 2)	(3, 6)
			$\Delta\alpha_5 = 8, \alpha_4 = 3$	(3, 3)	(3, 6)
			$\Delta\alpha_5 = 9, \alpha_4 = 4$	(3, 4)	(3, 6)
			$\Delta\alpha_5 = 10, \alpha_4 = 5$	(3, 5)	(3, 6)

Пример 4.7. Коррекция преобразованного остаточного кода в позиционный двоичный код для набора оснований $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11, p_6 = 13$, где $p_1 – p_4$ – информационные (рабочие) основания, а $p_5 – p_6$ – контрольные.

Предположим, что в результате сбоя или отказа вместо правильного числа A = (1, 1, 2, 5, 0, 5) было получено $\tilde{A}$ = (1, 1, 0, 5, 0, 5), т.е. ошибка произошла по основанию p_3. Тогда при переводе из СОК в ОПСС коэффициенты ОПСС (адреса ПЗУ) будут равны $a_1 = 1, a_2 = 0, a_3 = 4, a_4 = 4$, а остатки СОК по контрольным модулям - $\alpha_4 = 2$ и $\alpha_5 = 2$.

Определим синдром ошибки $\delta_1 = 0 - 2 = 9$, $\delta_2 = 5 - 2 = 3$.

В соответствии с выражением $A = a_1 + a_2 p_1 + a_3 p_1 p_2 + a_4 p_1 p_2 p_3$ на вход выходного сумматора будут поступать значения $a_1 = 1$, $a_2 = 0$, $a_3 p_1 p_2 = 4 \cdot 2 \cdot 3 = 24$ и $a_4 p_1 p_2 p_3 = 4 \cdot 2 \cdot 3 \cdot 5 = 120$.

В результате поступивших значений на выходе сумматора будет число, равное 1 + 24 + 120 = 145.

Величина поправки, представленная в СОК, определяет величину коррекции в ПСС, т.к. $\delta_1 = 9$, а $\delta_2 = 3$, то по таблице 6 находим $A_n = 42$ (в таблице 4.7 обведено контуром), тогда на выходном сумматоре по модулю $P = p_1 p_2 p_3 p_4 = 210$. Получим позиционное откорректированное число $A = |145 + 42|_{210} = 187$. Это соответствует правильному представлению в СОК, как (1, 1, 2, 5, 0, 5).

Таблица 4.7 Таблица поправок в СОК и в ПСС

Основание	Глубина ошибки	Разность минимального следа и остатка по контрольным основаниям		Поправка в ПСС по модулю 210
		$a_5 - a_5'$	$a_6 - a_6'$	
2	1	5 6	12 1	105 105
3	1	7 8	8 10	140 70

Продолжение таблицы 4.7

3	2	3 4	3 5	70 140
5	1	6 7	4 6	126 84
5	2	2 3	10 12	42 168
5	3	8 9	1 3	168 42
5	4	4 5	7 9	84 126
7	1	1 2	10 12	120 120
7	2	3 4	9 11	180 30
7	3	4 5	6 8	60 150
7	4	6 7	5 7	150 60
7	5	7 8	2 4	30 180
7	6	9 1	1 3	120 120
11	1	1 1	0 0	0 0

Продолжение таблицы 4.7

11	2	2 2	0 0	0 0
11	3	3 3	0 0	0 0
11	4	4 4	0 0	0 0
11	5	5 5	0 0	0 0
11	6	6 6	0 0	0 0
11	7	7 7	0 0	0 0
11	8	8 8	0 0	0 0
11	9	9 9	0 0	0 0
11	10	10 10	0 0	0 0
13	1	0 0	1 1	0 0
13	2	0 0	2 2	0 0
13	3	0	3 3	0 0

Продолжение таблицы 4.7

13	4	0	4 4	0 0
13	5	0	5 5	0 0
13	6	0	6 6	0 0
13	7	0	7 7	0 0
13	8	0	8 8	0 0
13	9	0	9 9	0 0
13	10	0	10 10	0 0
13	11	0	11 11	0 0
13	12	0	12 12	0 0

Таким образом, синтезированная нейронная сеть для преобразования кода системы остаточных классов в позиционный с коррекцией ошибок характеризуется минимальными аппаратурными затратами. Предложенный подход предполагает использование до 70 процентов аппаратной избыточности для обнаружения и исправления 100% одиночных ошибок и обнаружения всех двукратных ошибок. Аналогичный подход в позиционной системе счисления

потребует 200 % аппаратурной избыточности. Время преобразования в позиционный код определяется как

$$t_{ПСС} = (n+1)(t_p) + t_{ПЗУ} + T_{\sum} \ , \qquad (4.6)$$

где n – число каналов, t_p – время вычисления модульным сумматором, $t_{ПЗУ}$ – время выборки из ПЗУ коэффициентов ОПС, $T_{\sum}$ – время суммирования коэффициентов ОПС. Для 14 каналов время преобразования составит $t_{ПСС} = 88{,}8$ нс при аппаратных затратах 14 RAMB36, 14RAM18 и 1063 SLISEL.

4.2.4 Моделирование и сравнительный анализ предложенных моделей на основе проекций и расширения оснований СОК

Предложенная модель отказоустойчивого блока основана на нейронных сетях конечного кольца, состоящая из цифровых нейронов. Элементами цифрового нейрона являются умножители для формирования произведений синаптических весов на разрядные остатки модулярного кода и сумматоры для сложения полученных слагаемых с использованием функции активации. Реализация разработанной модели выполнена на ПЛИС Xilinx серии XC4000EC у которой время задержки на вентиле составляет не более одной наносекунды и ниже. Моделирование основных элементов блока произведено на языке HVDL.

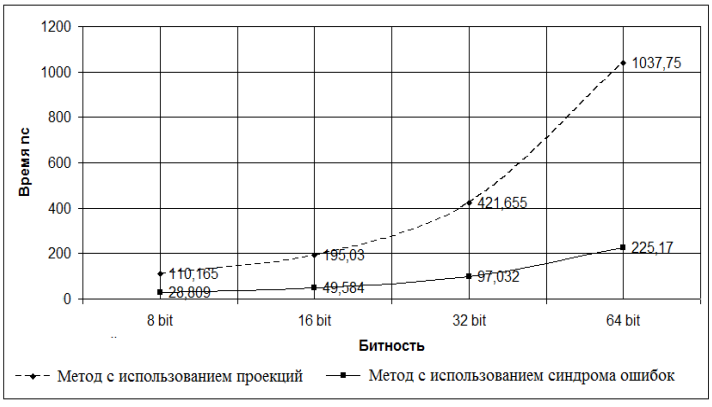


Рисунок 4.5 – Результаты моделирования методов коррекции ошибок.

На рисунке 4.5 приведены результаты моделирования синдромного метода и метода основанного на ОПСС с использованием проекций. Сравнение

полученных результатов говорит о значительном преимуществе метода с использованием синдрома ошибок.

Результаты моделирования вычислительных устройств в СОК показали, что они имеют более высокое быстродействие по сравнению с вычислительными устройствами в ПСС, и более гибки в возможностях реконфигурации и перепрограммирования. Так, для 16-разрядных входных двоичных чисел: скорость работы сумматора в СОК в 1.04 раза выше чем в ПСС, скорость работы умножителя в СОК в 2.27 раза выше чем в ПСС; для 64-разрядных входных двоичных чисел скорость работы сумматора в СОК в 1.31 раза выше чем в ПСС, скорость работы умножителя в СОК в 3.26 раза выше чем в ПСС. С увеличением разрядности входных чисел преимущество устройств в СОК будет возрастать.

Модель отказоустойчивого нейрокомпьютера на основе синдрома ошибок выгодно отличается от модели, основанной на методе проекций.

Рассмотренный метод значительно снижает вычислительную сложность и делает реализацию коррекции ошибок проще, который требует лишь одну процедуру преобразования из СОК в ОПСС, а в известном методе используется n+1 преобразование. По оценкам, с помощью этого метода, вычислительная сложность может быть уменьшена примерно на 80% по сравнению с методом на основе ОПСС с использованием ее проекций.

4.3 Многофункциональная модель для контроля и диагностики модулярных нейрокомпьютеров

На рисунке 4.6 приведена модель блока функционального контроля модулярного процессора. Входные регистры $RG(p_i)$ объемом $(n+r)\lceil \log p_i \rceil$ бит служат для временного хранения разрядов α_i СОК, поступающих по входным шинам α_i размером $q_i = \lceil \log_i \rceil$. Для формирования $\overline{A_i}$ проекций из принятого числа схем формирования проекций (СФП-А) за одно обращение вычеркивает один разряд α_i. Аналогичное назначение имеет и схема формирования проекций рабочего диапазона (СФП-М). СФП-А формирует проекции для избыточного

диапазона $P_{изб}$, а СФП-М – для рабочего диапазона $P=M$. Формирование $\overline{A_i}$ проекций осуществляет счетчик проекций (СТ) начиная с проекции $\overline{A_1}, \overline{A_2}, ..., \overline{A_{n+r}}$. Если счетчик находится в нулевом состоянии, тогда анализируется число $\overline{A}$. Запуском счетчика управляет «единственный» выход триггера (Т), который переходит в единичное состояние после анализа числа $\overline{A}$. Выходы схем формирования проекций СФП-А и СФП-М поступают на вход $LUT\ p_i$ – таблицы суммарным объемом $O\left(n 2^b \log n\right)$ бит. Для сравнения, процедура преобразования в ОПСС требует $O\left(n^2 2^b\right)$ бит при нежестком допущении когда $b_i = b$, где b_i – количество двоичных разрядов модулей p_i.

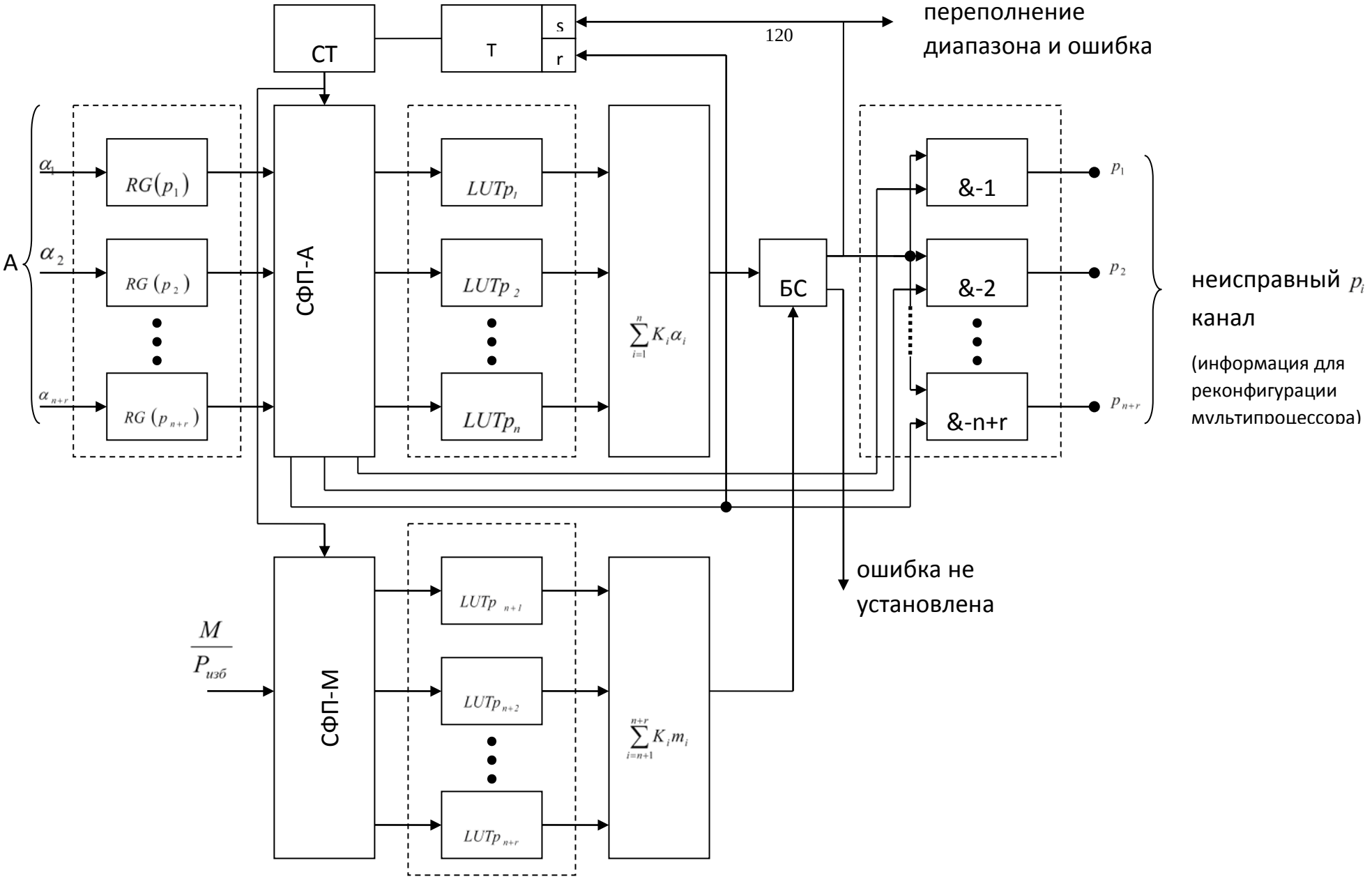


Рисунок 4.6 – Модель блока контроля модулярного нейрокомпьютера.

Для повышения эффективности алгоритма и удобства анализа его сложности, мы предполагаем, что величины модулей более-менее одинаковы. В LUT-таблицы записываются некоторые округленные дробные числа $[k_i\ \alpha_i]_{2^{-q}}$ и $[k_i\ m_i]_{2^{-q}}$, каждое из которых содержит $q' = nb + \log n$ бит, которые затем суммируются деревом сумматоров $\left[\sum_{i=1}^{n} k_i\ \alpha_i\right]_{2-q}$ и $\left[\sum_{i=n+1}^{n+2} k_i\ m_i\right]_{2-q}$ по модулю 1. Точные значения чисел определяются неравенством $[x]_{2-q} \le x \le [x]_{2-q+2-q}$.

Общая структура алгоритма и блока контроля похожи на известные алгоритмы и схемы, где для формирования проекций используется ОПСС. Предполагается, что СОК содержит *n* модулей, *n* модулярных процессоров и *b* бит для каждого из модулей. Предложенный блок контроля использует LUT-таблицы суммарным объемом $O\left(n2^b \log n\right)$ бит. Процедура преобразования в ОПСС требует таблицы размером $O\left(n^2 2^b\right)$ бит. По сравнению с наиболее эффективными практическими реализациями СОК [11], которые базируются на основе ОПСС, вычислительная сложность предложенной модели, которая базируется на приближенном методе, сокращает аппаратурные затраты в $\frac{n}{\log n}$ раз на выполнение основной части блока контроля. Элементы оставшейся части блока контроля однотипные, которые используются в известной и предложенной схемах. Сравнение вычислительной сложности предлагаемого блока проведем с известным, который реализован на улучшенной модели ОПСС.

Дальнейшие исследования будут направлены на адаптацию алгоритма к специальным наборам модулей и конкретных приложений и оптимального перераспределения данных между оставшимися работоспособными состояниями модулярного процессора.

4.4 Выводы по четвертой главе

1. Разработан алгоритм параллельного расширения модулей СОК, ориентированный на использовании нейронных сетей. Предложенный алгоритм сокращает число арифметических операций с $2(n-1)$ до $(n-1)$ по сравнению с итеративным методом Гарнера.

2. Предложена модель отказоустойчивого модулярного нейрокомпьютера на основе проекций числа с использованием обобщенного алгоритма , которая в отличии от известных имеет двойное назначение: первое – это традиционная система с обнаружением, локализацией и коррекцией ошибок, а второе – специализированное в пороговых структурах, например, в системах безопасности, которое основано на использовании реконфигурируемой в процессе функционирования искусственной нейронной сети. Предлагаемый подход позволяет достоверно преобразовывать остаточный код в двоичный при искажениях или зашумленных разрядах без их традиционного исправления за счет свойств пороговых схем. Таким образом, предлагаемый подход использует принцип обнаружения и локализации, а исправление и перевод из СОК в ПСС исключается, что позволяет оптимизировать производительность традиционных преобразований.

3. Разработан метод, алгоритм и модель отказоустойчивого модулярного нейрокомпьютера с коррекцией одиночной ошибки на основе расширения системы оснований СОК с использованием синдрома ошибок. Показано, что предложенный подход значительно снижает вычислительную сложность, который требует лишь одну процедуру преобразования из СОК в ОПСС, а в известных схемах используются $(n+1)$ преобразование. Моделирование, проведенное на созданной библиотеке программ на языке VHDL для реализации модулярных нейросетевых устройств с использованием ресурсов ПЛИС FPGA показало снижение вычислительной сложности на 80% по сравнению с методом на основе ОПСС с проекциями.

4. Предложен высокоэффективный подход, который способен оптимизировать производительность традиционных моделей отказоустойчивых модулярных нейрокомпьютеров за счет двойного назначения вычислительных коэффициентов ОПСС: одно – для формирования данных в формате СОК, а второе для данных – в формате ПСС без дополнительных временных затрат, так как обе операции выполняются параллельно. Предложена многофункциональная модель для контроля и диагностики модулярных нейрокомпьютеров на основе адаптации корректирующих свойств кодов СОК к дробным значениям метода приближенного вычисления позиционных характеристик. Проведены доказательства корректности использования КТО с дробными числами при использовании средств … работоспособности

5. Проведено экспериментальное исследование разработанных моделей с использованием средств среды ISE Design Suite 14.7 на плате Kintex-7. Результаты моделирования показали снижение затрат ресурсов платы при использовании модулей СОК разной разрядности, примерно в 1,3 раза по сравнению с ОПСС, а повышение скорости преобразования увеличилось примерно в 3 раза.

6. Создан комплекс программ для моделирования нейросетевых структур модулярного нейрокомпьютера. На основе анализа моделирования, синтеза из нейросетевых элементов модулярного нейрокомпьютера можно сделать вывод о том, что интеграция СОК и искусственных нейронных сетей, параллельное построение вычислений малых форматов данных, реализация метода приближенного вычисления позиционных характеристик – все это вместе дает возможность построить высокопроизводительный, отказоустойчивый модулярный нейрокомпьютер обработки данных большой размерности, характеристики которого превышают характеристики нейрокомпьютера, функционирующего в ПСС, причем разрыв в лучшую сторону характеристик будет увеличиваться при увеличении разрядности обрабатываемых данных.

Заключение

В диссертации проведено решение актуальной научной задачи по разработке модели отказоустойчивого модулярного нейрокомпьютера для обработки данных большой размерности на основе нового подхода с использованием КТО с дробными числами. В отличие от стандартной компьютерной арифметики, базирующейся на КТО с целыми числами, в работе получены новые технологии, которые способны оптимизировать производительность и отказоустойчивость вычислительных средств.

Проведенное в диссертации исследование, а также результаты экспериментов и моделирование в базисе ПЛИС дали возможность получить следующие основные научные и практические результаты.

1. Проведенный в работе анализ методов создания высокопроизводительных и надежных вычислительных устройств показал, что одним из перспективных является метод придания вычислительным устройствам устойчивости к отказам на основе применения СОК, обладающей высоким потенциалом по коррекции ошибок. Параметры СОК и вычислительной базы в виде ИНС позволяют создавать модулярные нейрокомпьютеры с высокой отказоустойчивостью.

2. Результаты проведенных исследований математических моделей повышения отказоустойчивости модулярных нейрокомпьютеров показали, что существующие и перспективные пути решения данной научной задачи базируются на использовании трудновыполнимых немодульных операций модулярного представления и обработки данных, которые существенным образом влияют на модель и принципы их функционирования.

В работе показано, что выбор двух позиционных характеристик в виде ОПСС и КТО с дробными числами, определяющих модели вычисления немодульных операций позволяют значительно повысить надежность и производительность обработки информации, что подтверждается приведенными расчетами и математическим моделированием.

3. Исследования КТО с дробными числами позволили разработать эффективный метод вычисления позиционной характеристики, имеющий линейную алгоритмическую сложность, а также новые методы выполнения немодульных операций на этой основе.

4. На основе разработанных моделей корректирующих свойств кодов СОК с дробными числами проведена адаптация и доказана корректность применения их при вычислении немодульных процедур, используемых при синтезе отказоустойчивых модулярных нейрокомпьютеров и показана их высокая эффективность, что подтверждено математическим моделированием основных узлов нейрокомпьютера, приведенных в выводах по главам.

5. Разработаны модели и алгоритмы реализации схем сравнения и определения знака модулярных чисел, основанных на КТО с дробными числами. Сравнительная оценка применения приближенного метода с точным методом показал высокую эффективность приближенного метода. Так, при разрядности 80 бит выигрыш в slices и задержка снижена более, чем в 4 раза.

6. Разработанные впервые модели, алгоритмы и технологии реализации модулярного деления на основе приближенного метода является лучшим на сегодняшний день. Предложенные алгоритмы и аппаратная их реализация значительно снижают ресурсы использования плат FPGA и увеличивают скорость преобразования.

Экспериментальные исследования математических методов моделирования алгоритмов деления в среде ISE Design Suite 14.7 при использовании четырех модулей, покрывающие диапазон в 64 бита, показали следующие результаты: предложенный метод требует всего 689 блоков slice, в то время как метод деления на основе ортогональных базисов и улучшенный метод ОПСС 1457 и 865 блоков slice, соответственно; частота – 65,5 МГц, что в 7,6 раза больше, чем КТО, и в 10,1 раз больше, чем ОПСС.

7. Разработаны эффективные численные методы: модулярного деления; вычисления частного при использовании нового алгоритма деления; коррекции ошибок на основе метода расширения модулей СОК; определения неисправного

модуля СОК реализованы в виде комплекса программ для проведения вычислительных экспериментов при аппаратной реализации метода деления в формате СОК и высокоскоростного общего деления с использованием метода приближенного вычисления позиционной характеристики, а также при разработке технических реализаций схем коррекции ошибок на основе расширения модулей СОК и схем определения неисправного канала СОК.

8. Разработан перспективный метод и алгоритм, на основе которого предложена модель модулярного высокопроизводительного и отказоустойчивого нейрокомпьютера с коррекцией ошибки на основе расширения модулей СОК. Показано, что предложенный подход снижает вычислительную сложность по сравнению с традиционной реализацией на основе проекций числа в $n+1$ раз, где n - число модулей СОК.

9. Предложена многофункциональная модель для диагностики модулярных нейрокомпьютеров на основе корректирующих свойств кодов СОК. Проведена адаптация корректирующих свойств кодов СОК к дробным числам метода приближенного вычисления позиционных характеристик. Доказана корректность использования КТО с дробями для целей контроля и диагностики модулярных нейрокомпьютеров. На основании результатов моделирования проведена сравнительная оценка предложенного подхода с известным, которая показала преимущество предложенного метода с известным ОПСС по затратам ресурсов платы Kintex 7 в 1,3 раза, а по скорости, примерно, в 3 раза.

10. Создан комплекс программ для моделирования нейросетевых структур основных блоков модулярного нейрокомпьютера, который позволил провести вычислительные эксперименты на базе платформы FPGA управляемый пользователем, который может перенастраивать на различные конфигурации при решении проблем разработки схем, устойчивых к отказам.

11. Показано, что интеграция СОК и ПСС, параллелизм, малые форматы за счет модулярного разделения, реализация метода КТО с дробями является основой перспективной разработки модулярных отказоустойчивых

нейрокомпьютеров, обладающих лучшими характеристиками при обработке данных большой размерности.

12. Полученные результаты имеют высокое практическое значение для развития высокопроизводительных и отказоустойчивых модулярных нейрокомпьютеров, и их применение в современных системах обработки данных позволяет реализовать обработку данных большой размерности в сотни двоичных разрядов, например, при разработке систем безопасности.

Обозначения и сокращения

СОК – система остаточных классов

КТО – китайская теорема об остатках

ОПСС – обобщенная позиционная система счисления

ПСС – позиционная система счисления

ЦОС – цифровая обработка сигналов

ИНС – искусственные нейронные сети

ПЛИС – программируемые логические интегральные схемы

МА – модулярная арифметика

БИС – большие интегральные схемы

СБИС – сверхбольшая интегральная схема

VHDL – язык моделирования

BRAM – блочная память

LUT – просмотровая таблица

КЛБ – конфигурируемый логический блок

НК – нейрокомпьютеры

НСКК – нейронная сеть конечного кольца

ВО – вычислительные операции

ВУ – вычислительные устройства

DSP – процессоры цифровой обработки сигналов

РМВ – реальный масштаб времени

ОЗУ – оперативное запоминающие устройство

ОУ – операционное устройство

ММ – математическая модель

ПХ – позиционная характеристика

УПХ – универсальная позиционная характеристика

УППХ – универсальная приближенная позиционная характеристика

FPGA – программируемые вентильные матрицы

Slice – элементарные ячейки ПЛИС

Список литературы

1. Айерленд, К. Классическое введение в современную теорию чисел. – М.: Мир, 1987. – 416 с.
2. Акушский, И.Я. Машинная арифметика в остаточных классах / И.Я. Акушский, Д.И. Юдицкий – М.: Советское радио, 1968. – 440 с.
3. Бухштаб, А.А. Теория чисел. – Лань, 2008. - 384 с.
4. Амербаев, В.М. Модулярной арифметике – 50 лет / В.М. Амербаев, И.Т. Пак // 50 лет модулярной арифметики: труды юбилейной Международной научной конференции, Зеленоград. – МИЭТ, 2006. – С. 5-21.
5. Виноградов, М.М. Основы теории чисел. – М.: Наука. – 1981. – 176 с.
6. Долгов, А.И. Диагностика устройств, функционирующих в системе остаточных классов. – М.: Радио и связь, 1982. – 64 с.
7. Дадев, Ю.Г. Арифметические коды, исправляющие ошибки. – М.: Советское радио. – 1968. – 68 с.
8. Дзегелёнок, И.И. Подход к решению проблемы безошибочных вычислений с использованием ускоренного алгоритма отображения дробей Фарея / И.И. Дзегелёнок, Ш.А. Оцоков // Труды научной конференции, посвященной 75-летию со дня рождения академика В.А. Меньшикова. РАН. – М., 2004. – С. 25-31.
9. Жихарев, В.Я. Пути повышения производительности и отказоустойчивости ЭВМ / В.Я. Жихарев, Юнес Эль Хандасси, В.А. Краснобаев // Открытые информационные и компьютерные технологии. – Х.: НАКУ (ХАИ). – 2003. – Вып. 19. – С. 269 - 282.
10. Журавлев, Ю.П. Надежность и контроль ЭВМ / Ю.П. Журавлев, Л.А. Кателюк, Н.И. Циклинский; под ред. Ю.П. Журавлева – М.: Радио и связь, 1978. – 416 с.
11. Жуков, О.Д. Модулярные вычисления в системах защиты информации / О.Д. Жуков // Информационные технологии. – 2002. – №1. – С. 26 – 29.
12. Жуков, О.Д. Коррекция двойных ошибок и обнаружения многократных ошибок модулярных вычислений / О.Д. Жуков // Информационные технологии. – 2002. – №7. – С. 15 -20.
13. Жуков, О.Д. Обнаружение и коррекция ошибок компьютерных вычислений на основе модулярной алгебры / О.Д. Жуков // Информационные технологии. – 2009. – №6. – С. 15 – 24.

14. Жуков, О.Д. Обработка числовых данных с повышенной точностью в модулярной алгебре / О.Д. Жуков // Информационные технологии. – 2004. – №2. – С. 10 -15.

15. Зотов, В.Ю. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPACK ISE / В.Ю. Зотов. – М.: горячая линия-Телеком, 2003. – 624 с.

16. Ирхин, В.П. Проектирование непозиционных специализированных процессоров / В.П. Ирхин. – Воронеж, 1999. – 118 с.

17. Ирхин, В.П. Алгоритмы табличной реализации остаточной арифметики / В.П. Ирхин // Воронеж: ВИРЭ. Труды института: Методические основы развития способов и средств радиоэлектронной борьбы, 1995. – С. 37–39.

18. Ирхин, В.П. Проектирование непозиционных специализированных процессов. / В.П. Ирхин. – Воронеж: Издательство Воронежского государственного университета, 1999. – 136 с.

19. Ирхин, В.П. Табличная реализация операций модулярной арифметики / В.П. Ирхин // 50 лет модулярной арифметики: труды юбилейной международной научной конференции (Москва, Зеленоград, 23–25 ноября 2005). – М.: издательство МИЭТ, 2005. – С. 261–266.

20. Ирхин В.П. Улучшение основных характеристик операционных устройств спецпроцессоров. / В.П. Ирхин// Харьков: ХВКИУРВ. Тематический научно-технический сборник, 1992. - № 337 - С. 31–33.

21. Ирхин, В.П. Разработка тренажёров на базе непозиционных вычислительных устройств / В.П. Ирхин, В.М. Коровин // Научно-методический сборник МО РФ № 49. – М.: Воениздат, 2000. – С. 94–98.

22. Ирхин, В.П. Алгоритм приведение двоичного числа по простому модулю / В.П. Ирхин, В.А. Табуненко. – Системы информационного взаимодействия. Сборник научных трудов. – Харьков: НАНУ, ПАНИ, ХВУ, 1996. – С. 43–46.

23. Инютин, С.А. Основы многоразрядной алгоритмики / С.А. Инютин. – Сургут: РИО, 2002. – 137 с.

24. Исупов, К.С. Способ представления чисел с плавающей точкой большой разрядности, ориентированный на параллельную обработку / К.С. Исупов, А.Н. Мальцев // Вычислительные методы и программирование. – 2014. – Т. 15. – № 4. – С. 631–643.

25. Исупов, К.С. Библиотека параллельной арифметики многократной точности для высокопроизводительных систем / К.С. Исупов, В.С. Князьков // Суперкомпьютерные дни в России: труды международной конференции. Суперкомпьютерный консорциум университетов России, Федеральное агентство научных организаций России. – М., 2015. – С. 110-121.

26. Исупов, К.С. Высокоточное вычисление интервально-позиционной характеристики модулярной арифметики / К.С. Исупов // Всероссийская ежегодная научно-практическая конференция: сборник материалов: общеуниверситетская секция, БФ, ГФ, ФЭМ, ФАВТ, ФАМ, ФПМТ, ФСА, ХФ, ЭТФ. Вятский государственный университет. – Киров, 2014. – С. 1173-1178.

27. Исупов, К.С. Библиотека высокоточных вычислений для многоядерных процессоров / К.С. Исупов, А.Н. Мальцев // Всероссийская ежегодная научно-практическая конференция: сборник материалов: общеуниверситетская секция, БФ, ГФ, ФЭМ, ФАВТ, ФАМ, ФПМТ, ФСА, ХФ, ЭТФ. Вятский государственный университет. – Киров, 2014. –С. 1185-1189.

28. Калашников, В.С. Исследование и разработка методов проектирования быстродействующих вычислительных узлов для реализации отказоустойчивых систем на основе модулярной арифметики: дис. ... канд. техн. наук: 05.13.05/ Калашников Вячеслав Сергеевич. – М., 2007. – 180 с.

29. Калмыков, И.А. Математическая модель нейронной сети для исправления ошибок непозиционного кода поля Галуа в частотной области / И.А. Калмыков // Нейрокомпьютеры: разработка и применение. – 2004. – № 5-6. – С. 71-78.

30. Калмыков, И.А. Разработка метода контроля и коррекции ошибок для непозиционного спецпроцессора с деградируемой структурой / И.А. Калмыков // Збірник наукових праць: Київ, Національна Академия Наук Украіни. – 2004. – №25, С. 65 – 78.

31. Калмыков, И.А. Архитектура отказоустойчивой нейронной сети для цифровой обработки сигналов / И.А. Калмыков, Н.И. Червяков, Ю.О. Щелкунова, В.В. Бережной // Нейрокомпьютеры: разработка, применение. – 2004. – №12. – С. 51-60.

32. Калмыков, И.А. Математические модели нейросетевых отказоустойчивых вычислительных средств, функционирующих в полиномиальной системе классов вычетов / И.А. Калмыков: под ред. Н.И. Червякова. – М.: Физматлит, 2005. – 276 с.

33. Кнут, Д. Искусство программирования / Дональд Эрвин Кнут . - Том 2. Получисленные алгоритмы, 3-е издание. Перевод с англ. – М.: Издательский дом «Вильямс», 2003. – 832 с.

34. Краснобаев, В.А. Надежностная модель ЭВМ в системе остаточных классов / В.А. Краснобаев // Электрон. моделирование. – 1985. – №4. – С. 44-46.

35. Коляда, А.А. Модулярные вычислительные структуры: вчера, сегодня, завтра / А.А. Коляда, А.Ф. Чернявский // Труды Юбилейной Международной научной конференции «50 лет модулярной арифметики», Зеленоград. – МИЭТ, 2006. – С. 23-34.

36. Коломейко, В.В. Вопросы упрощения немодульных операций в специализированных ЭВМ, работающих в СОК / В.В. Коломейко, В.Д. Петищак // – М.: Кибернетика, 1986. – С. 104–106.

37. Лавриненко, А.В. Принципы построения нейрокомпьютеров. / А.В. Лавриненко // Сборник научных трудов. – 2007. – №24. – С. 179-182.

38. Лавриненко, А.В. Метод преобразования кода системы остаточных классов в позиционный с коррекцией ошибок на основе искусственных нейронных сетей / А.В. Лавриненко // Наука. Инновации. Технологии. Научный журнал Северо-Кавказского федерального университета. – 2015. – №3. – С. 7-36.

39. Лавриненко, А.В. Проблемные вопросы реализации специализированных модулярных нейрокомпьютеров / А.В. Лавриненко, Н.И. Червяков, А.Н. Головко, А.В. Кондрашов, В.В. Сляднев // Научно-техническая конференция преподавателей и студентов СГУ. Научно-инновационные достижения ФМФ в области физико-математических и технических дисциплин: материалы конференции. – Ставрополь: СГУ, 2007. – С.149-152.

40. Лавриненко, А.В. Применение принципа синхронизации нейронных сетей в криптографии / А.В. Лавриненко, Н.И. Червяков, А.Н. Головко, А.В. Кондрашов, В.В. Сляднев, С.С. Кириевский // Научно-техническая конференция преподавателей и студентов СГУ. Научно-инновационные достижения ФМФ в области физико-математических и технических дисциплин: материалы конференции. – Ставрополь: СГУ, 2007. – С.142-145.

41. Лавриненко, А.В. Применение распределенной арифметики для реализации быстродействующих КИХ-фильтров / А.В. Лавриненко, Н.И. Червяков, А.Н. Головко, А.В. Кондрашов, В.В. Сляднев // Научно-техническая конференция преподавателей и студентов СГУ. Научно-инновационные

достижения ФМФ в области физико-математических и технических дисциплин: материалы конференции. – Ставрополь: СГУ, 2007. – С.145-149.

42. Лавриненко, А.В. Нейронная сеть для определения координат точки на эллиптической кривой / А.В. Лавриненко, Н.И. Червяков, А.Н. Головко, И.Н. Лавриненко, С.С. Кириевский // Инфокомунникационные технологии в науке, производстве и образовании: Третья международная научно-практическая конференция, 1-5 мая 2008 года. – Ставрополь, 2008. – С. 252 - 258.

43. Лавриненко, А.В. Нейронная сеть для обнаружения ошибок в минимально-избыточной симметричной системе остаточных классов / А.В. Лавриненко, А.Н. Головко, И.Н. Лавриненко, С.В, Лавриненко, В.В. Сляднев, М.А. Оспищев // Инфокомунникационные технологии в науке, производстве и образовании: Третья международная научно-практическая конференция, 1-5 мая 2008 года. – Ставрополь, 2008. – С.258 - 262.

44. Лавриненко, А.В. Новые информационные технологии безопасного хранения ключевой информации на принципе пороговых криптосистем / А.В. Лавриненко, Н.И. Червяков, А.Н. Головко, М.А. Оспищев, С.С. Кириевский, О.С. Мезенцева // Инфокомунникационные технологии в науке, производстве и образовании: Третья международная научно-практическая конференция, 1-5 мая 2008 года. – Ставрополь, 2008. – С.263 - 276.

45. Лавриненко, А.В. Новые технологии криптографической защиты данных на основе нейронных сетей и системы остаточных классов / А.В. Лавриненко, Н.И. Червяков, А.Н. Головко, А.В. Кондрашов В.В. Сляднев, С.С. Кириевский // Информационные системы, технологии и модели управления производством: материалы 3 международной научно-практической конференции. 12-13 марта 2007. – Ставрополь, 2007. – С. 10-13.

46. Лавриненко, А.В. Модулярные ПИД-регуляторы на базе нейронных сетей конечного конца / А.В. Лавриненко, Н.И. Червяков, Д.Н. Югов // Нейрокомпьютеры: разработка, применение. – 2007. – №5. – С. 40-48.

47. Лавриненко, А.В. Эллиптические кривые и криптография / А.В. Лавриненко, А.Н. Головко, В.В. Сляднев // Сборник научных трудов. – 2008. – №25. – С. 24 - 28.

48. Лавриненко, А.В. Определение подлинности объекта по внешним признакам / А.В. Лавриненко, А.Н. Головко, С.С. Кириевский // Сборник научных трудов. – 2008. – №25. – С. 35 - 38.

49. Лавриненко, А.В. Алгоритм с открытым ключом с многослойными персептронами (МСП) / А.В. Лавриненко, Н.И. Червяков, А.В. Кондрашов, Ю.В. Кондрашов, В.В. Сляднев // Проблемы и перспективы развития инфотелекоммуникационных технологий в системах связи военного назначения. – 2008. – №3. – С.12-13.

50. Лавриненко, А.В. Взаимодействующие нейронные сети / А.В. Лавриненко, Н.И. Червяков, А.В. Кондрашов, Ю.В. Кондрашов, В.В. Сляднев // Проблемы и перспективы развития инфотелекоммуникационных технологий в системах связи военного назначения. – 2008. – №3. – С.13-14.

51. Лавриненко, А.В. Исследование нейронного прогнозирующего вейвлет-фильтра / А.В. Лавриненко, А.И. Колдаев, С.С. Кириевский // Нейрокомпьютеры: разработка, применение. – 2009. – №7. – С. 35-40.

52. Лавриненко, А.В. Применение искусственных нейронных сетей и системы остаточных классов в криптографии /А.В. Лавриненко, Н.И. Червяков, А.А. Евдокимов, А.И, Галушкин. – М.: ФИЗМАТЛИТ, 2012. – 370 с.

53. Лавриненко, А.В. Пат. 2318239 Российская Федерация, МПК G06N 3/06, G06F 7/72. Нейронная сеть для деления чисел, представленных в системе остаточных классов / А.В. Лавриненко, Н.И. Червяков, А.В. Кондрашов, В.В. Сляднев; заявитель и патентообладатель Ставропольский военный институт связи. - № 2006124114/09; заявл. 05.07.2006; опубл. 27.02.2008, Бюл. № 6. – 9 с.

54. Лавриненко, А.В. Пат. 2359325 Российская Федерация, МПК G06N 3/04, G06F 7/72. Нейронная сеть ускоренного масштабирования модулярных чисел / Н.И. Червяков, А.Н. Головко; А.В. Лавриненко, В.В. Сляднев; заявитель и патентообладатель Ставропольский военный институт связи. - № 2007122192/09; заявл. 13.06.2007; опубл. 20.08.2009, Бюл. № 17. – 11 с.

55. Лавриненко, А.В. Пат. 2374678 Российская Федерация, МПК G06F 11/07, G06N 3/06, H03M 7/18. Нейронная сеть для обнаружения ошибок в симметричной системе остаточных классов / Н.И. Червяков, С.В. Лавриненко, И.Н. Лавриненко, А.Н. Головко, А.В. Лавриненко; заявитель и патентообладатель Ставропольский военный институт связи. - № 2007139960/09; заявл. 29.10.2007; опубл. 27.11.2009, Бюл. № 33. – 7 с.

56. Лавриненко, А.В. Пат. 2380751 Российская Федерация, МПК G06N 3/04. Нейронная сеть с пороговой (k,t) структурой для преобразования остаточного кода в двоичный позиционный код / Н.И. Червяков, А.Н. Головко, А.В. Лавриненко, Ю.В. Кондрашов, В.А. Козлов, С.В. Назаренко, М.А. Оспищев;

заявитель и патентообладатель Ставропольский военный институт связи. - № 2008121788/09; заявл. 30.05.2008; опубл. 27.01.2010, Бюл. № 3. – 13 с.

57. Лавриненко, А.В. Пат. 2400813 Российская Федерация, МПК G06N 3/02, G06F 7/72. Нейронная сеть основного деления модулярных чисел / Н.И. Червяков, И.Н. Лавриненко, А.В Лавриненко, А.Н. Головко; заявитель и патентообладатель Ставропольский военный институт связи. - № 2008150458/09; заявл. 22.12.2008; опубл. 27.06.2010, Бюл. № 27. – 10 с.

58. Лавриненко, А.В. Пат. 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 / Н.И. Червяков, М.Г. Бабенко, П.А. Ляхов, И.Н. Лавриненко, А.В Лавриненко; заявитель и патентообладатель Северо-Кавказский федеральный университет. - № 2011145755/08; заявл. 10.11.2011; опубл. 27.05.2013, Бюл. № 15. – 10 с.

59. Лавриненко, А.В. Пат. 2503995 Российская Федерация, МПК G06F 7/72. Устройство для определения знака модульного числа / Н.И. Червяков, М.Г. Бабенко, П.А. Ляхов, И.Н. Лавриненко, А.В Лавриненко; заявитель и патентообладатель Северо-Кавказский федеральный университет. - № 2011139278/08; заявл. 26.09.2011; опубл. 10.01.2014, Бюл. № 1. – 8 с.

60. Лавриненко, А.В. Пат. 2503992 Российская Федерация, МПК G06F 7/02, G06F 7/72. Устройство для сравнения чисел, представленных в системе остаточных классов / Н.И. Червяков, М.Г. Бабенко, П.А. Ляхов, И.Н. Лавриненко, А.В Лавриненко; заявитель и патентообладатель Северо-Кавказский федеральный университет. - № 2011139397/08; заявл. 27.09.2011; опубл. 10.01.2014, Бюл. № 1. – 11 с.

61. Лавриненко, А.В. Компьютерные вычисления на основе модулярной алгебры / А.В, Лавриненко, Червяков Н.И. Бабенко М.Г. Ляхов П.А. Лавриненко И.Н. – Ставрополь: Издательско - информационный центр «Фабула», 2015. – 210 с.

62. Лавриненко, А.В. Ускоренный метод вычисления остатка от деления с использованием распределенной арифметики / А.В. Лавриненко, Н.И. Червяков, М.Г. Бабенко, М.А. Дерябин, А.С. Назаров // Свидетельство об официальной регистрации программы для ЭВМ № 2016612432, РФ. Зарегистрировано в Реестре программ для ЭВМ 26.02.2016 г.

63. Нейрокомпьютеры в системах обработки изображений. Кн. 7. Коллективная монография (серия «Нейрокомпьютеры и их применение») / Общая ред. А.И. Галушкина. – М.: Радиотехника, 2003. – 192 с.

64. Нейрокомпьютеры в системах обработки сигналов. Кн. 9. Коллективная монография (серия «Нейрокомпьютеры и их применение») / Под ред. Ю.В. Гуляева и А.И.Галушкина. – М.: Радиотехника, 2003. – 224 с.

65. Овчаренко, Л.А. Оптимизация структуры ЭВМ в системе остаточных классов / Л.А. Овчаренко, А.А. Болкунов // Шестая межвузовская научно – техническая конференция. Тезисы докладов. Труды института. – Воронеж: ВИРЭ., 2000. – вып. 6. – С. 77-81.

66. Овчаренко, Л.А. Обнаружение ошибок в модулярном арифметическом устройстве на основе применения контрольного основания / Л.А. Овчаренко, С.С, Чекалин // Сборник научных трудов. – Харьков: НАНУ, ПАНМ, ХВУ, 2002. – вып. 5(21). – С. 55 -61.

67. Полард, Дж. Быстрые преобразования Фурье в конечном поле / Дж. Полард // Применение теории чисел в цифровой обработке сигналов. – 1983. – №2. – С. 147-156.

68. Потапов, Н.В. Отказоустойчивые функционально избыточные нейронные сети индивидуальной адаптацией к отказам нейронов / Н.В. Потапов // Нейрокомпьютеры. – 2010. – №5. – С. 5–10.

69. Стемпковский, А.Л. Особенности реализации устройств цифровой обработки сигналов в интегральном исполнении с применением модулярной арифметики / А.Л. Стемпковский, А.Н. Корнилов, М.Ю. Семенов // Инфокоммуникационные технологии. – 2004. – № 2. – С. 2-9.

70. Тынчеров, К.Т. Основы теории и принципы построения отказоустойчивых структур на основе нейронных сетей / дис.... док. техн. наук: 05.13.15./ Тынчеров КамильТалятович. – М., 2012. –325 с.

71. Торгашев, В.А. Система остаточных классов и надежность ЦВМ. / В.А. Торгашев. – М.: Сов. радио, 1973. – 120 с.

72. Финько, О.А. Модулярная арифметика параллельных вычислений: монография / Финько О.А.; под редакцией В.Д. Малюгина. – М.: Институт проблем управления им. В.А. Трапезникова РАН: Краснодар : Краснодарский военный институт, 2003. – 224 с.

73. Финько, О.А. Методика защиты модулярных криптопроцессоров от аппаратных ошибок / О.А. Финько // Межвузовский сборник научных трудов. – Краснодар: Краснодарский военный институт, 2001. – С. 171-175.

74. Финько, О.А. Методы обработки больших массивов информации на основе арифметики в остаточных классах / О.А. Финько // Третья научно – техническая конф. «перспективы использования новых технологий и научно-технических решений в изделиях ракетно-технической техники разработки ГКНПЦ им. М.В. Хруничева». Москва, 16-18 декабря 2003. Сборник трудов. – М.: Ин-т проблем управления им. В.А. Трапезникова РАН, 2003. – С. 211-216.

75. Финько, О.А. Многоканальные модулярные системы, устойчивые к искажениям криптограмм / О.А. Финько // Труды Юбилейной Международной научно - технической конференции «50 лет модулярной арифметике», Россия, Москва, Зеленоград, 23 -25 ноября 2005. – МИЭТ, 2005. – С. 545 -551.

76. Хайкин, С. Нейронные сети: полный курс / Саймон Хакин, 2-е изд. Пер. с англ. – М.: ООО «И. Д. Вильямс», 2006. – 1104 с.

77. Червяков, Н.И. Модулярные параллельные вычислительные структуры нейропроцессорных систем / Н.И.Червяков, А.В. Шапошников , С.А. Ряднов. – М.: Физматлит, 2002. – 288 с.

78. Червяков, Н.И. Нейрокомпьютеры в остаточных классах / Н.И. Червяков, П.А. Сахнюк, А.Н. Макоха. – М.: ИПРЖР, 2003. – 272 с.

79. Червяков, Н. И Модель и структура нейронной сети для реализации арифметики системы остаточных классов / Н. И. Червяков, А. В. Шапошников, П. А. Сахнюк // Нейрокомпьютеры: разработка, применение. – 2001. – № 10. – С. 6-12.

80. Червяков, Н.И. Структуры нейронных сетей конечного кольца. / Н.И. Червяков, С.Л. Ремизов // Нейрокомпьютеры: разработка, применение. – 2004. – №12. – С. 21-30.

81. Червяков, Н.И. Отказоустойчивые непозиционные процессоры / Н.И. Червяков // Управляющие системы и машины. – 1988. – №3. – С. 3-7.

82. Akkal, M. A new Mixed Radix Conversion algorithm MRC-II / M. Akkal, P. Siy // Journal of Systems Architecture. – 2007. – №53. – P. 577–586

83. Alia, G.,. NEUROM: a ROM based RNS digital neuron / G. Alia, E. Martinelli // Neural Networks. – 2005. – №18. – P. 179-189.

84. Beckmann, P. Fast Fault-Tolerant Digital Convolution Using a Polynomial Reside Number System / P. Beckmann, R. Bruce // IEEE Transactions on Signal Processing, 1993. – P. 2300-2313.

85. Lavrinenko, A.V., Comparison of Modular Numbers Based on the Chinese Remainder Theorem with Fractional Values / N.I. Chervyakov, A.S. Molahosseini, P.A. Lyakhov, M.G. Babenko, I.N. Lavrinenko // Automatic control and computer scinces. – 2015. – Vol. 49. – № 6. – P. 354-365

86. Chang, C. A division algorithm for residue numbers / Chin-Chen Chang, Yeu-Pong Lai // Applied Mathematics and Computation. – 2006. – V. 172. – № 1. – P. 368-378.

87. Chervyakov, N.I. Digital rietering of images in a residue number system vsing rinite-rield wavelets / N.I. Chervyakov, P.A. Lyakhov, M.G. Babenko // Automatic control and computer scinces. – 2014. – Vol. 48. – № 3. – P. 180-189.

88. Goh, V.T. Multiple error detection and correction based on Redundant Residue Number System / V.T. Goh, M.U. Siddiqi // IEEE Transactions on Communications. – 2008. – Vol. 56. – № 3. – Pp. 325-330.

89. Gomathisankaran, M.. HORNS: A homomorphic encryption scheme for Cloud Computing using Residue Number System / M. Gomathisankaran, A. Tyagi, K. Namuduri // Information Sciences and Systems (CISS), 45th Annual Conference. – 2011. – № 2. – P. 1-5.

90. Hung, C.Y.. Fast RSN division algorithms for fixed divisors with application to RSA encryption / C.Y. Hung, B. Parhami // Information Processing Letters. 1994. V. 51, No. 4. P. 163–169. A. Number theoretic processor / Патент США №4281391 NDCK 13/24.

91. Hung, C.Y. An approximate sign detection method for residue numbers and its application to RNS division / C.Y. Hung, B. Parhami // Computers & Mathematics with Applications. – 1994. – Vol. 27. – № 4. – P. 23-25.

92. Lu, M. A novel deviation algorithm for residue number systems / M. Lu, J.S. Chiang // IEEE Transactions on Computers. – 1992. – Vol. 41. – № 8. – P. 1026–1032.

93. Howard, S. L. Error control coding in low-power wireless sensor networks: When is ECC energy-efficient. / S.L. Howard, C. Schlegel, K. Iniewski // EURASIP Journal on Wireless Communications and Networking. – 2006. – №3. – P. 28-29.

94. Jakop, Y. A robust symmetrical number system based parallel communication system with inherent error detection and correction / Y. Jakop,

A.S. Madhukumar, A.B. Premkumar // IEEE Transactions on Wireless Communications. – 2009. – № 6. – P. 2742-2747.

95. Jilu, J. A novel method for error correction using Redundant Residue Number System in digital communikation systems. / James Jilu, Pe Ameenudeen // IEEE 2015 International Conference on Advances in Computing, Communication and Informatics (ICACCI). – 2015. – № 9. – P. 1793-1798.

96. Chang, C.-H.. Residue Number Systems: A New Paradigm to Datapath Optimization for Low-Power and High-Performance Digital Signal Processing Applications / C.-H. Chang, A.S. Molahosseiqi, A.A.E. Zarandi, T.F. Tay // IEEE Circuits and Systems Magazine. – 2015. – Vol. 15. – № 4. – P. 26-44.

97. Zhengbing, H.. Increasing the data transmission robustness in wsn using the modified error correction codes on residue number system. / Hu Zhengbing, Vasyl Yatskiv, Anatoliy Sachenko // Eleltronika ir elektrotechnika, ISSN 1392 – 1215. – 2015. – Vol. 21. – №1. – P. 12-17.

98. Jun, S. Method and dedicated processor for image coding based on residue number system /Su Jun, Hu Zhengbing // Modern Problems of Radio Engineering Telecommunications and Computer Science (TCSET), International Conference. – 2012. – №4. – P. 406-407.

99. Chiang, J.S. A general division algorithm for residue number systems / J.S. Chiang, M. Lu // 10th IEEE symposium on computer arithmetic. – 1991. – P.14-23.

100. Mohan, P.V. Residue Number Systems – Algorithms and Architectures / P.V. Mohan. – Kluwer Academic Publishers, 2002. – 254 p.

101. Molahosseini, A.S. Research challenges in next-generation residue number system architectures / A.S Molahosseini, S. Sorouri, A.A.E. Zarandi // Computer Science & Education (ICCSE), International Conference. – 2012. – №7. – P. 1658-1661.

102. Haron, N.Z. Redundant Residue Number System Code for Fault-Tolerant Hybrid Memories / Nor Zaidi Haron and Said Hamdioui // ACM Journal on Emerging Technologies in Computing Systems. – 2011. – № 1. – P. 11-23.

103. Nykolaychuk, Ya. M. Theoretical Foundations for the Analytical Computation of Coefficients of Basic Numbers of Krestenson’s Transformation / Ya. M. Nykolaychuk, M. M. Kasianchuk, I. Z. Yakymenko // Cybernetics and Systems Analysis. – 2014. – № 50. – P. 649-654.

104. Omondi, A., Residue Number Systems / A. Omondi, B. Premkumar. Theory and Implementation. – London: Imperial College Press, 2007. – 296 p.

105. Pontarelli, G.C. Totally fault tolerant RNS based FIR filters. / G.C. Pontarelli, M.Re. Cardarilli, A. Salsano // 14th IEEE IOLTS. – 2008. – P. 192-194.

106. Hiasat, A.A. Design and implementation of an RSN division algorithm / A.A. Hiasat, H.S. Abdel-Aty-Zohdy // 13th IEEE symposium on computer arithmetic. – 1997. – P. 18-25.

107. Hiasat, A.A. Semi-custom VLSI design and implementation of a new efficient RSN division algorithm / A.A. Hiasat, H.S. Abdel-Aty-Zohdy // The computer journal. – 1999. – V. 42. – № 3. – P. 232–240.

108. Sachenco, A. Increasing the Data Transmission Robustness in Wsn Using the Modified Error Correction Codes on Residue Number System. / Hu Zhengbing and Vasyl Yatskiv // Elektronika ir Elektrotechnika. – 2015. – № 1. – P. 76-81.

109. Safiri, H. Design and FPGA Implementation of Systolic FIR Filters Using the Fermat ALU / H. Safiri, H. Ahamadi, V. Dimitrov // Proc. of the Asilomar Conference on Signals, Systems and Computers, Pacific Grove. – 1996. – № 7. – P. 56-64.

110. Szabo, N. Residue arithmetic and is applications to computer technology. / N. Szabo, R. I. Tanaka. – New York, 1967. – 238 p.

111. Talahmeh, S.. Aritmetic division in RSN using field GF(p) / S. Talahmeh, P. Siy // Computers & Mathematics with Aplications. – 2000. – № 39. – P. 227–238.

112. Vetterli, M. Wavelets and Subband Coding. / M. Vetterli and J. Kovacevic. – Englewood Cliffs, NJ: Prentice Hall, 1995. – 505 p.

113. Yang, L. Redundant Residue Number System Based Error Correction Codes. / Yang L-L. And Hanzo L // In VTC'2001. – 2001. – № 1. – P. 1472-1476.

114. Yatskiv, V. The Use of Modified Correction Code Based on Residue Number System in WSN. / V. Yatskiv, N. Yatskiv, Su Jun, A. Sachenko., Hu Zhengbing //Proceedings of the 7-th2013 IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems, IDAACS'2013, Berlin, Germany. – 2013. – №1. – P. 513-516.

115. Yang, Y.H. A high-speed division algorithm in residue number system using parity-checking technique / Y.H Yang, C.C. Chang, C.Y. Chen // International journal of computer mathematics. – 2004. – №6. – P. 775–780.

116. Chang, C.-C. A division algorithm for residue numbers / Chin-Chen Chang, Yeu-Pong Lai //Applied mathematics and computation. – 2006. – №1. – P 368-378.

117. Zhang, D. Parallel designs for Chinese remainder conversion / D. Zhang// Proc. Int. Conf. Parallel Process. University Park, Pa. – 1987. – №2. – P. 557-559.

118. Zheng, X.D.. Parallel DNA arithmetic operation based on n-moduli set / X.D. Zheng, J. Xu, W. Li // Applied Mathematics and Computation. – 2009. – №1. – P. 177-184.

ПРИЛОЖЕНИЕ 1

Программа управления устройством перевода чисел из системы остаточных классов в позиционную систему счисления на основе Китайской теоремы об остатках

Файл crt_converter.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

use work.rns_constants_pkg.all;
use work.pairing_package.all;


entity crt_converter is
	Port (
		rns : in  rns_number(0 to N-1);		-- число в СОК
		A   : out unsigned(pBC-1 downto 0)	-- результат восстановления
	);
end crt_converter;

architecture Behavioral of crt_converter is

	-- подключение компонента для сдваивания
	COMPONENT pairing
	GENERIC (N: NATURAL);
	PORT(
		input : IN input_block;
		output : OUT unsigned(pairBC-1 downto 0)
	);
	END COMPONENT;

	-- сигналы для сохранения результатов произведений на константы B
	signal ba : input_block(0 to N-1);

	-- результат рекурсивного сдваивания
	signal q : unsigned(pairBC - 1 downto 0);
begin
```

```
	-- генерация модулей умножения входов rns на константы B
	for1 : for I in 0 to N-1 generate
		ba(I) <= rns(I) * B(I);
	end generate for1;

	-- выполнение рекурсивного сдваивания
	inst_pairing: pairing
	GENERIC MAP (
		N => N
	)
	PORT MAP(
		input => ba,
		output => q
	);

	-- вычисление результата
	A <= q mod P;

end Behavioral;
```

Файл pairing.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

use work.pairing_package.all;

entity pairing is
	Generic (
		constant N: natural := CN
	-- размер входного массива
	);
	Port (
		input  : in  input_block(0 to N-1);
		output : out unsigned(BC-1 downto 0)
	);
end pairing;

architecture Behavioral of pairing is
```

```
	-- подключение компонента, отвечающего за один слой сдваивания
	COMPONENT pairing_layer
		GENERIC (N:NATURAL);				-- количество элеметов входного сигнала
		PORT(
			input : IN  input_block;
			output : OUT  input_block
		);
  END COMPONENT;


	------------------------------------------------------------
	-- блок функций для расчета основных констант

	-- получение количества элементов в слое по его номеру
	-- рекурсивный расчет до нужного слоя
	function getLayerSize(ln, n : in natural) return natural is
		variable res : natural;
	begin
		res := n;

		for i in 2 to ln loop
			-- количество элементов слоя равно половине количества элементов предыдущего слоя
			-- плюс 1 в случае, если предыдущий слой содержал нечетное количество элементов
			res := res/2 + (res mod 2);
		end loop;

		return res;
	end function getLayerSize;

	-- получение начальной позиции слоя
	function getLayerStart(ln, n : in natural) return natural is
		variable res : natural;
	begin
		res := 0;

		-- начальная позиция слоя равна сдвигу от нуля на количество всех элементов предыдущих слоев
		for i in 1 to ln-1 loop
			res := res + getLayerSize(i, n);
```

```
            end loop;

            return res;
        end function getLayerStart;

        -- получение количества всех компонентов во всех сллоях
        function getAllLayersSize(lc, n : in natural) return natural is
            variable res : natural;
        begin
            res := 0;

            -- складываются количества элеметов на всех слоях плюс выходной
элемент
            for ln in 1 to lc+1 loop
                res := res + getLayerSize(ln, n);
            end loop;

            return res;
        end function getAllLayersSize;

        -- получение количества слоев
        -- двоичный логарифм числа N с округлением вверх
        function getLayersCount(n: in natural) return natural is
            variable res : natural;
            variable vn : unsigned(31 downto 0);  -- число, учитывающее границы
типа natural
        begin
            res := 0;

            vn := to_unsigned(n-1, 32);              -- n-1 чтобы верно считать log(2^t)

            -- считаем количество бит в числе vn
            while to_integer(vn) /= 0 loop
                res := res + 1;
                vn := vn srl 1;
            end loop;

            return res;
        end function getLayersCount;
        ------------------------------------------------------------
```

```
	constant layers_count : natural := getLayersCount(N);
		-- количество слоев

	constant signal_size : natural := getAllLayersSize(layers_count, n);		-- размер массива сигналов
	signal signals: input_block (0 to signal_size-1);
		-- массив сигналов:

-- хранит все сигналы, связывающие слои

begin

	-- запись массива входов в массив сигналов
	f1 : for I in 0 to N-1 generate
		signals(I) <= input(I);
	end generate f1;

	-- генерация слоев
	f2 : for I in 1 to layers_count generate

		-- подключение слоя сдваивания
		-- на вход подается результат расчета предыдущего слоя
		layer : pairing_layer
		GENERIC MAP ( N => getLayerSize(I, N) )
		PORT MAP (
			input => signals(getLayerStart(I, N) to getLayerStart(I, N)+getLayerSize(I, N)-1),
			output => signals(getLayerStart(I+1, N) to getLayerStart(I+1, N)+getLayerSize(I+1, N)-1)
		);
	end generate f2;

	-- результат работы последнего слоя подается на выход
	output <= signals(signal_size-1);

end Behavioral;
```

Файл pairing_layer.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.NUMERIC_STD.ALL;


use work.pairing_package.all;

entity pairing_layer is
	Generic (
		constant N: natural := 8
	-- количество элементов слоя
	);
	Port (
		input  : in  input_block(0 to N-1);
	-- вход: массив размера N
		output : out input_block(0 to N/2 - 1 + (N mod 2) )	-- выход: массив
размера N/2 (плюс 1 в случае, если N нечетно)
	);
end pairing_layer;

architecture Behavioral of pairing_layer is

begin

	-- генерация блоков сложения для элементов входного массива
	-- сдваиваются элементы от начала с элементами с конца
	for1: for I in 0 to (N / 2 - 1) generate
		output(I) <= input(I) + input(N-I-1);
	end generate for1;

	-- вывод среднего элемента напрямую в случае, если N нечетно
	if1: if (N mod 2 = 1) generate
		output(N / 2) <= input(N / 2);
	end generate if1;

end Behavioral;
```

Файл pairing_layer.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

use work.rns_constants_pkg.all;
```

```
package pairing_package is

	constant BC : natural := pairBC;
	constant CN : natural := 9;

	type input_block is array (natural range <>) of unsigned(BC-1 downto 0);

end pairing_package;


package body pairing_package is

end pairing_package;
```

Файл pairing_layer.vhd

```
-- константы для  СОК с модулями (31, 37, 41, 43, 47, 53, 59, 61)

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.ALL;

package rns_constants_pkg is

	-- количество бит модуля P
	constant pBC : natural := 45;

	function getAllBlocksCount(num: in natural) return natural;
	function getBlockStart(num: in natural) return natural;

	-- количество бит используемых чисел
	constant rnsBC : natural := 6;

	-- количество бит констант k
	constant kBC : natural := 53;

	constant pairBC : natural := pBC + rnsBC;                    --- crt original

	-- количество оснований
	constant N : natural := 8;
```

```
constant p1_pairBC : natural := pBC;


-- ортогональные базисы
type b_array is array (0 to N-1) of unsigned(pBC - 1 downto 0);
constant B : b_array := (
        0 => "010101010001100110011100010000001011000011100",
        1 => "100000000101011100001110001111011001010100100",
        2 => "010010011111111011011010100110000110011111011",
        3 => "011101001001000100111000010000110010110111010",
        4 => "011110001010110111011001011111101001100011001",
        5 => "010010101010100111001010010011000001001111110",
        6 => "011110001011101000000110111000110010011001010",
        7 => "010001110101101111000101000111101011111001001"
);

-- константы приближенного метода
type k_array is array (0 to N-1) of unsigned(kBC - 1 downto 0);
constant k : k_array := (
        0 => "101001010010100101001010010100101001010010100110",
        1 => "111110010001010011000001101110101100111110010001010",
        2 => "100011111001110000011000111110011100000110001111110100",
        3 => "111000100011101110001000111011100010001110111000010010",
        4 => "111010100011011001110111110101000110110011101111110110",
        5 => "100100001110011111011001010110111100011000001001101110",
        6 => "111010100100111000011010000010001010110110001111001110",
        7 => "100010100111110111100110110100011101011000001000011101"
);

--
constant basisSize : natural := getAllBlocksCount(N);
type basis_array is array (0 to basisSize-1) of unsigned(rnsBC - 1 downto 0);
constant basis : basis_array := (
        "011111",
        "000110",

        "010011",
        "010110",
        "101000",

        "001100",
```

```
            "100000",
            "010100",
            "010100",

            "011011",
            "011111",
            "010011",
            "000101",
            "001010",

            "110001",
            "100110",
            "110010",
            "011011",
            "100101",
            "001001",

            "010100",
            "010100",
            "001100",
            "110101",
            "101111",
            "011111",
            "110001",

            "100111",
            "111011",
            "100010",
            "110101",
            "110111",
            "100010",
            "110111",
            "100001"

        );



--      constant r1 :  := m(0);
--      constant r2 : unsigned(3*rnsBC-1 downto 0) := m(0)*m(1);
--      constant r3 : unsigned(3*rnsBC-1 downto 0) := m(0)*m(1)*m(2);
```

```
	-- основани¤
	type m_array is array (0 to N-1) of unsigned(rnsBC - 1 downto 0);
	constant m : m_array := (
		"011111",
		"100101",
		"101001",
		"101011",
		"101111",
		"110101",
		"111011",
		"111101"
	);

	type r_array is array (0 to N-1) of unsigned((N-1)*rnsBC-1 downto 0);
	constant r : r_array := (
		0 => (0 => '1', others => '0'),
		1 => resize(m(0), (N-1)*rnsBC),
		2 => resize(m(0)*m(1), (N-1)*rnsBC),
		3 => resize(m(0)*m(1)*m(2), (N-1)*rnsBC),
		4 => resize(m(0)*m(1)*m(2)*m(3), (N-1)*rnsBC),
		5 => resize(m(0)*m(1)*m(2)*m(3)*m(4), (N-1)*rnsBC),
		6 => resize(m(0)*m(1)*m(2)*m(3)*m(4)*m(5), (N-1)*rnsBC),
		7 => resize(m(0)*m(1)*m(2)*m(3)*m(4)*m(5)*m(6), (N-1)*rnsBC)
	);

	-- диапазон
	constant P : unsigned(pBC - 1 downto 0) :=
"1000001111100111101100100011000100010010001010101";

	--
	type rns_number is array (natural range <>) of unsigned(rnsBC-1 downto 0);

end rns_constants_pkg;

package body rns_constants_pkg is


	function getAllBlocksCount(num: in natural) return natural is
		variable res: natural;
	begin
		res := 0;
```

```
            if num > 1 then
                for i in 2 to num loop
                    res := res + i;
                end loop;
            end if;
            return res;
        end function getAllBlocksCount;

        function getBlockStart(num: in natural) return natural is
        begin
            return getAllBlocksCount(num-1);
        end function getBlockStart;

end rns_constants_pkg;
```

ПРИЛОЖЕНИЕ 2

Программа управления устройством перевода чисел из системы остаточных классов в позиционную систему счисления на основе перевода в обобщенную позиционную систему счисления

Файл mrc_converter.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

use work.rns_constants_pkg.all;
use work.pairing_package.all;
use work.p1_pairing_package.all;


entity mrc_converter is
	Port (
		rns : in  rns_number(0 to N-1);		-- число в СОК
		A   : out unsigned(pBC-1 downto 0)	-- результат восстановления
		--mrc : out rns_number(0 to N-1)
	);
end mrc_converter;


architecture Behavioral of mrc_converter is
	COMPONENT pairing
	GENERIC (N: NATURAL);
	PORT(
		input : IN input_block;
		output : OUT unsigned(pairBC-1 downto 0)
	);
	END COMPONENT;

	COMPONENT p1_pairing
	GENERIC (N: NATURAL);
	PORT(
		input : IN p1_input_block;
		output : OUT unsigned(p1_pairBC-1 downto 0)
```

```
    );
    END COMPONENT;

    COMPONENT divmod_e
    PORT(
        num    : in  unsigned(2*rnsBC-1 downto 0);
        m_num  : in  natural;
        quot   : out unsigned(rnsBC-1   downto 0);
        remain : out unsigned(rnsBC-1   downto 0)
    );
    END COMPONENT;

    signal mrc : rns_number(0 to N-1);

    --type signals_array is array (0 to basisSize) of unsigned (2*rnsBC-1 downto 0);
    signal signals: input_block(0 to basisSize);

    type sums_array is array (2 to N) of unsigned (2*rnsBC-1 downto 0);
    signal sums: sums_array;

    signal sums1: sums_array;

    type t_array is array (2 to N-1) of unsigned (rnsBC-1 downto 0);
    signal t: t_array;

    --type w_array is array (0 to N-1) of unsigned(pBC-1 downto 0);
    signal w : p1_input_block(0 to N-1);

begin

    mrc(0) <= rns(0);

    for1: for I in 2 to N generate
        for2 : for J in getBlockStart(I) to getBlockStart(I) + I-1 generate
            signals(J) <= resize(rns(I-1) * basis(J), pairBC);
        end generate for2;
    end generate for1;

    for3: for I in 2 to N generate
        inst_pairing: pairing
        generic map (
            N => I
```

```
        )
        port map (
                input => signals(getBlockStart(I) to getBlockStart(I) + I-1),
                output => sums(I)
        );
    end generate for3;


    sums1(2) <= sums(2);

    for4: for I in 2 to N-1 generate

        inst_divmod: divmod_e
        port map (
                num    => sums1(I),
                m_num  => I-1,
                quot   => t(I),
                remain => mrc(I-1)
        );

        if1: if I /= N generate
                sums1(I+1) <= sums(I+1) + t(I);
        end generate if1;
    end generate for4;

    mrc(N-1) <= sums1(N) mod m(N-1);

    for5: for I in 0 to N-1 generate
        w(I) <= r(I) * mrc(I);
    end generate for5;

    inst_p1_pairing: p1_pairing
    generic map (
        N => N
    )
    port map (
        input => w,
        output => A
    );


end Behavioral;
```

Файл divmod.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

use work.rns_constants_pkg.all;

entity divmod_e is
	Port (
		num    : in  unsigned(rnsBC*2-1 downto 0);
		m_num  : in  natural;
		quot   : out unsigned(rnsBC-1 downto 0);
		remain : out unsigned(rnsBC-1   downto 0)
	);
end divmod_e;

architecture Behavioral of divmod_e is

	function getTopBit(denom: unsigned) return integer is
	begin
	 for J in DENOM'RANGE loop
		if DENOM(J)='1' then
		  return J;
		end if;
	 end loop;
	 return -1;
	end function getTopBit;

	type topbits_array is array (0 to N-1) of integer;
	constant topbits : topbits_array := (
		0 => getTopBit(m(0)),
		1 => getTopBit(m(1)),
		2 => getTopBit(m(2)),
		3 => getTopBit(m(3))
	);


	-- this internal procedure computes UNSIGNED division
	-- giving the quotient and remainder.
```

```
        procedure DIVMOD (signal NUM : unsigned; I: natural; signal XQUOT,
XREMAIN: out UNSIGNED) is
         variable TEMP: UNSIGNED(NUM'LENGTH downto 0);
         variable QUOT: UNSIGNED(NUM'LENGTH-1 downto 0);
         --alias DENOM: UNSIGNED(m(I)'LENGTH-1 downto 0) is m(I);
         --variable TOPBIT: INTEGER;
        begin
         TEMP := "0"&NUM;
         QUOT := (others => '0');

         assert topbits(I) >= 0 report "DIV, MOD, or REM by zero" severity ERROR;

         for J in NUM'LENGTH-(topbits(I)+1) downto 0 loop
               if TEMP(topbits(I)+J+1 downto J) >= "0"&m(I)(topbits(I) downto 0) then
                 TEMP(topbits(I)+J+1 downto J) := (TEMP(topbits(I)+J+1 downto J))
                           -("0"&m(I)(topbits(I) downto 0));
                 QUOT(J) := '1';
               end if;
               assert TEMP(topbits(I)+J+1)='0'
                     report "internal error in the division algorithm"
                     severity ERROR;
         end loop;
         XQUOT <= RESIZE(QUOT, XQUOT'LENGTH);
         XREMAIN <= RESIZE(TEMP, XREMAIN'LENGTH);
        end DIVMOD;

begin

        DIVMOD(num, m_num, quot, remain);

end Behavioral;
```

Файл pairing.vhd и файл p1_pairing.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

use work.pairing_package.all;

entity pairing is
```

```
        Generic (
                constant N: natural := CN
        -- размер входного массива
        );
        Port (
                input  : in  input_block(0 to N-1);
                output : out unsigned(BC-1 downto 0)
        );
end pairing;

architecture Behavioral of pairing is

        -- подключение компонента, отвечающего за один слой сдваивания
        COMPONENT pairing_layer
                GENERIC (N:NATURAL);                    -- количество элеметов входного сигнала
                PORT(
                        input : IN  input_block;
                        output : OUT  input_block
                );
  END COMPONENT;


        ----------------------------------------------------------------
        -- блок функций для расчета основных констант

        -- получение количества элементов в слое по его номеру
        -- рекурсивный расчет до нужного слоя
        function getLayerSize(ln, n : in natural) return natural is
                variable res : natural;
        begin
                res := n;

                for i in 2 to ln loop
                        -- количество элементов слоя равно половине количества элементов предыдущего слоя
                        -- плюс 1 в случае, если предыдущий слой содержал нечетное количество элементов
                        res := res/2 + (res mod 2);
                end loop;

                return res;
```

```
    end function getLayerSize;

    -- получение начальной позиции слоя
    function getLayerStart(ln, n : in natural) return natural is
        variable res : natural;
    begin
        res := 0;

        -- начальная позиция слоя равна сдвигу от нуля на количество всех элементов предыдущих слоев
        for i in 1 to ln-1 loop
            res := res + getLayerSize(i, n);
        end loop;

        return res;
    end function getLayerStart;

    -- получение количества всех компонентов во всех сллоях
    function getAllLayersSize(lc, n : in natural) return natural is
        variable res : natural;
    begin
        res := 0;

        -- складываются количества элеметов на всех слоях плюс выходной элемент
        for ln in 1 to lc+1 loop
            res := res + getLayerSize(ln, n);
        end loop;

        return res;
    end function getAllLayersSize;

    -- получение количества слоев
    -- двоичный логарифм числа N с округлением вверх
    function getLayersCount(n: in natural) return natural is
        variable res : natural;
        variable vn : unsigned(31 downto 0); -- число, учитывающее границы типа natural
    begin
        res := 0;

        vn := to_unsigned(n-1, 32);             -- n-1 чтобы верно считать log(2^t)
```

```
		-- считаем количество бит в числе vn
		while to_integer(vn) /= 0 loop
			res := res + 1;
			vn := vn srl 1;
		end loop;

		return res;
	end function getLayersCount;
	-------------------------------------------------------------

	constant layers_count : natural := getLayersCount(N);
		-- количество слоев

	constant signal_size : natural := getAllLayersSize(layers_count, n);		-- размер массива сигналов
	signal signals: input_block (0 to signal_size-1);
		-- массив сигналов:

                                                                                -- хранит все сигналы, связывающие слои

begin

	-- запись массива входов в массив сигналов
	f1 : for I in 0 to N-1 generate
		signals(I) <= input(I);
	end generate f1;

	-- генерация слоев
	f2 : for I in 1 to layers_count generate

		-- подключение слоя сдваивания
		-- на вход подается результат расчета предыдущего слоя
		layer : pairing_layer
		GENERIC MAP ( N => getLayerSize(I, N) )
		PORT MAP (
			input => signals(getLayerStart(I, N) to getLayerStart(I, N)+getLayerSize(I, N)-1),
			output => signals(getLayerStart(I+1, N) to getLayerStart(I+1, N)+getLayerSize(I+1, N)-1)
		);
```

```
        end generate f2;

        -- результат работы последнего слоя подается на выход
        output <= signals(signal_size-1);

end Behavioral;
```

Файл pairing_layer.vhd и файл p1_pairing_layer.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;


use work.pairing_package.all;

entity pairing_layer is
        Generic (
                constant N: natural := 8
        -- количество элементов слоя
        );
        Port (
                input  : in  input_block(0 to N-1);
        -- вход: массив размера N
                output : out input_block(0 to N/2 - 1 + (N mod 2) )         -- выход: массив
размера N/2 (плюс 1 в случае, если N нечетно)
        );
end pairing_layer;

architecture Behavioral of pairing_layer is

begin

        -- генерация блоков сложения для элементов входного массива
        -- сдваиваются элементы от начала с элементами с конца
        for1: for I in 0 to (N / 2 - 1) generate
                output(I) <= input(I) + input(N-I-1);
        end generate for1;

        -- вывод среднего элемента напрямую в случае, если N нечетно
```

```
        if1: if (N mod 2 = 1) generate
                output(N / 2) <= input(N / 2);
        end generate if1;

end Behavioral;
```

Файл pairing_layer.vhd и ФАЙЛ p1_pairing_layer.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

use work.rns_constants_pkg.all;

package pairing_package is

        constant BC : natural := pairBC;
        constant CN : natural := 9;

        type input_block is array (natural range <>) of unsigned(BC-1 downto 0);

end pairing_package;


package body pairing_package is

end pairing_package;
```

Файл pairing_layer.vhd

```
-- константы для  СОК с модулями (31, 37, 41, 43, 47, 53, 59, 61)

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.ALL;

package rns_constants_pkg is

        -- количество бит модуля P
        constant pBC : natural := 45;
```

```
function getAllBlocksCount(num: in natural) return natural;
function getBlockStart(num: in natural) return natural;

-- количество бит используемых чисел
constant rnsBC : natural := 6;

-- количество бит констант k
constant kBC : natural := 53;

constant pairBC : natural := pBC + rnsBC;                    --- crt original

-- количество оснований
constant N : natural := 8;

constant p1_pairBC : natural := pBC;


-- ортогональные базисы
type b_array is array (0 to N-1) of unsigned(pBC - 1 downto 0);
constant B : b_array := (
        0 => "010101010001100110011100010000001011000011100",
        1 => "100000000101011100001110001111011001010100100",
        2 => "010010011111111011011010100110000110011111011",
        3 => "011101001001000100111000010000110010110111010",
        4 => "011110001010110111011001011111101001100011001",
        5 => "010010101010100111001010010011000001001111110",
        6 => "011110001011101000000110111000110010011001010",
        7 => "010001110101101111000101000111101011111001001"
);

-- константы приближенного метода
type k_array is array (0 to N-1) of unsigned(kBC - 1 downto 0);
constant k : k_array := (
        0 => "10100101001010010100101001010010100101001010010100110",
        1 => "11111001000101001100000110111010110011111001000101010",
        2 => "10001111100111000001100011111001110000011000111110100",
        3 => "11100010001110111000100011101110001000111011100010010",
        4 => "11101010001101100111011111010100011011001110111110110",
        5 => "10010000111001111101100101011011110001100000100110110",
        6 => "11101010010011100001101000001000101011011000111100110",
        7 => "10001010011111011110011011010001110101100000100001101"
);
```

```
--
constant basisSize : natural := getAllBlocksCount(N);
type basis_array is array (0 to basisSize-1) of unsigned(rnsBC - 1 downto 0);
constant basis : basis_array := (
        "011111",
        "000110",

        "010011",
        "010110",
        "101000",

        "001100",
        "100000",
        "010100",
        "010100",

        "011011",
        "011111",
        "010011",
        "000101",
        "001010",

        "110001",
        "100110",
        "110010",
        "011011",
        "100101",
        "001001",

        "010100",
        "010100",
        "001100",
        "110101",
        "101111",
        "011111",
        "110001",

        "100111",
        "111011",
        "100010",
        "110101",
```

```
            "110111",
            "100010",
            "110111",
            "100001"

        );



--      constant r1 :  := m(0);
--      constant r2 : unsigned(3*rnsBC-1 downto 0) := m(0)*m(1);
--      constant r3 : unsigned(3*rnsBC-1 downto 0) := m(0)*m(1)*m(2);


        -- основани¤
        type m_array is array (0 to N-1) of unsigned(rnsBC - 1 downto 0);
        constant m : m_array := (
            "011111",
            "100101",
            "101001",
            "101011",
            "101111",
            "110101",
            "111011",
            "111101"
        );

        type r_array is array (0 to N-1) of unsigned((N-1)*rnsBC-1 downto 0);
        constant r : r_array := (
            0 => (0 => '1', others => '0'),
            1 => resize(m(0), (N-1)*rnsBC),
            2 => resize(m(0)*m(1), (N-1)*rnsBC),
            3 => resize(m(0)*m(1)*m(2), (N-1)*rnsBC),
            4 => resize(m(0)*m(1)*m(2)*m(3), (N-1)*rnsBC),
            5 => resize(m(0)*m(1)*m(2)*m(3)*m(4), (N-1)*rnsBC),
            6 => resize(m(0)*m(1)*m(2)*m(3)*m(4)*m(5), (N-1)*rnsBC),
            7 => resize(m(0)*m(1)*m(2)*m(3)*m(4)*m(5)*m(6), (N-1)*rnsBC)
        );

        -- диапазон
        constant P : unsigned(pBC - 1 downto 0) :=
"1000001111100111101100100011000100010010001000101";
```

```
	--
	type rns_number is array (natural range <>) of unsigned(rnsBC-1 downto 0);

end rns_constants_pkg;

package body rns_constants_pkg is


	function getAllBlocksCount(num: in natural) return natural is
		variable res: natural;
	begin
		res := 0;
		if num > 1 then
			for i in 2 to num loop
				res := res + i;
			end loop;
		end if;
		return res;
	end function getAllBlocksCount;

	function getBlockStart(num: in natural) return natural is
	begin
		return getAllBlocksCount(num-1);
	end function getBlockStart;

end rns_constants_pkg;
```

ПРИЛОЖЕНИЕ 3

Программа управления устройством перевода чисел из системы остаточных классов в позиционную систему счисления на основе Китайской теоремы об остатках с дробными с числами

Файл crt_converter.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

use work.rns_constants_pkg.all;
use work.pairing_package.all;


entity crt_converter is
	Port (
		rns : in  rns_number(0 to N-1);		-- число в СОК
		A   : out unsigned(pBC-1 downto 0)	-- результат восстановления
	);
end crt_converter;

architecture Behavioral of crt_converter is
	COMPONENT pairing
	GENERIC (N: NATURAL);
	PORT(
		input : IN input_block;
		output : OUT unsigned(pairBC-1 downto 0)
	);
	END COMPONENT;

	signal ka : input_block(0 to N-1);

	signal q : unsigned(pairBC - 1 downto 0);
	signal A11 : unsigned(pairBC + pBC - 1 downto 0);
begin

	for1 : for I in 0 to N-1 generate
```

```
            ka(I) <= resize(rns(I) * k(I), pairBC);
        end generate for1;

        inst_pairing: pairing
        GENERIC MAP (
            N => N
        )
        PORT MAP(
            input => ka,
            output => q
        );

        A11 <= q * P;

        A <= A11(pairBC + pBC - 1 downto pairBC);

end Behavioral;
```

Файл pairing.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

use work.pairing_package.all;

entity pairing is
        Generic (
            constant N: natural := CN
        -- размер входного массива
        );
        Port (
            input  : in  input_block(0 to N-1);
            output : out unsigned(BC-1 downto 0)
        );
end pairing;

architecture Behavioral of pairing is

        -- подключение компонента, отвечающего за один слой сдваивания
        COMPONENT pairing_layer
```

```
        GENERIC (N:NATURAL);                    -- количество элеметов входного сигнала
        PORT(
            input : IN  input_block;
            output : OUT  input_block
        );
    END COMPONENT;
    ------------------------------------------------------------
    -- блок функций для расчета основных констант

    -- получение количества элементов в слое по его номеру
    -- рекурсивный расчет до нужного слоя
    function getLayerSize(ln, n : in natural) return natural is
        variable res : natural;
    begin
        res := n;

        for i in 2 to ln loop
            -- количество элементов слоя равно половине количества элементов предыдущего слоя
            -- плюс 1 в случае, если предыдущий слой содержал нечетное количество элементов
            res := res/2 + (res mod 2);
        end loop;

        return res;
    end function getLayerSize;

    -- получение начальной позиции слоя
    function getLayerStart(ln, n : in natural) return natural is
        variable res : natural;
    begin
        res := 0;

        -- начальная позиция слоя равна сдвигу от нуля на количество всех элементов предыдущих слоев
        for i in 1 to ln-1 loop
            res := res + getLayerSize(i, n);
        end loop;

        return res;
    end function getLayerStart;
```

```
	-- получение количества всех компонентов во всех сллоях
	function getAllLayersSize(lc, n : in natural) return natural is
		variable res : natural;
	begin
		res := 0;

		-- складываются количества элеметов на всех слоях плюс выходной элемент
		for ln in 1 to lc+1 loop
			res := res + getLayerSize(ln, n);
		end loop;

		return res;
	end function getAllLayersSize;

	-- получение количества слоев
	-- двоичный логарифм числа N с округлением вверх
	function getLayersCount(n: in natural) return natural is
		variable res : natural;
		variable vn : unsigned(31 downto 0);  -- число, учитывающее границы типа natural
	begin
		res := 0;

		vn := to_unsigned(n-1, 32);		-- n-1 чтобы верно считать log(2^t)

		-- считаем количество бит в числе vn
		while to_integer(vn) /= 0 loop
			res := res + 1;
			vn := vn srl 1;
		end loop;

		return res;
	end function getLayersCount;
	------------------------------------------------------------

	constant layers_count : natural := getLayersCount(N);
		-- количество слоев

	constant signal_size : natural := getAllLayersSize(layers_count, n);		-- размер массива сигналов
```

```
	signal signals: input_block (0 to signal_size-1);
		-- массив сигналов:
-- хранит все сигналы, связывающие слои
	begin

	-- запись массива входов в массив сигналов
	f1 : for I in 0 to N-1 generate
		signals(I) <= input(I);
	end generate f1;

	-- генерация слоев
	f2 : for I in 1 to layers_count generate

		-- подключение слоя сдваивания
		-- на вход подается результат расчета предыдущего слоя
		layer : pairing_layer
		GENERIC MAP ( N => getLayerSize(I, N) )
		PORT MAP (
			input => signals(getLayerStart(I, N) to getLayerStart(I,
N)+getLayerSize(I, N)-1),
			output => signals(getLayerStart(I+1, N) to getLayerStart(I+1,
N)+getLayerSize(I+1, N)-1)
		);
	end generate f2;

	-- результат работы последнего слоя подается на выход
	output <= signals(signal_size-1);

end Behavioral;
```

Файл pairing_layer.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;


use work.pairing_package.all;

entity pairing_layer is
	Generic (
```

```
            constant N: natural := 8
        -- количество элементов слоя
        );
        Port (
            input  : in  input_block(0 to N-1);
        -- вход: массив размера N
            output : out input_block(0 to N/2 - 1 + (N mod 2) )         -- выход: массив
размера N/2 (плюс 1 в случае, если N нечетно)
        );
end pairing_layer;

architecture Behavioral of pairing_layer is

begin
        -- генерация блоков сложения для элементов входного массива
        -- сдваиваются элементы от начала с элементами с конца
        for1: for I in 0 to (N / 2 - 1) generate
            output(I) <= input(I) + input(N-I-1);
        end generate for1;

        -- вывод среднего элемента напрямую в случае, если N нечетно
        if1: if (N mod 2 = 1) generate
            output(N / 2) <= input(N / 2);
        end generate if1;

end Behavioral;
```

Файл pairing_layer.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

use work.rns_constants_pkg.all;

package pairing_package is

        constant BC : natural := pairBC;
        constant CN : natural := 9;

        type input_block is array (natural range <>) of unsigned(BC-1 downto 0);
```

```
end pairing_package;


package body pairing_package is

end pairing_package;
```

Файл pairing_layer.vhd

```
-- константы для  СОК с модулями (31, 37, 41, 43, 47, 53, 59, 61)

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.ALL;

package rns_constants_pkg is

	-- количество бит модуля P
	constant pBC : natural := 45;

	function getAllBlocksCount(num: in natural) return natural;
	function getBlockStart(num: in natural) return natural;

	-- количество бит используемых чисел
	constant rnsBC : natural := 6;

	-- количество бит констант k
	constant kBC : natural := 53;

	constant pairBC : natural := pBC + rnsBC;		--- crt original

	-- количество оснований
	constant N : natural := 8;

	constant p1_pairBC : natural := pBC;


	-- ортогональные базисы
	type b_array is array (0 to N-1) of unsigned(pBC - 1 downto 0);
	constant B : b_array := (
		0 => "010101010001100110011100010000001011000011100",
```

```
        1 => "10000000010101110000111000111101100101010010 0",
        2 => "01001001111111101101101010011000011001111101 1",
        3 => "01110100100100010011100001000011001011011101 0",
        4 => "01111000101011011101100101111110100110001100 1",
        5 => "01001010101010011100101001001100000100111111 0",
        6 => "01111000101110100000011011100011001001100101 0",
        7 => "01000111010110111100010100011110101111100100 1"
);

-- константы приближенного метода
type k_array is array (0 to N-1) of unsigned(kBC - 1 downto 0);
constant k : k_array := (
        0 => "10100101001010010100101001010010100101001010010100110",
        1 => "11111001000101001100000110111010110011111100100010101 0",
        2 => "10001111100111000001100011111001110000011000111110100",
        3 => "11100010001110111000100011101110001000111011100010010",
        4 => "11101010001101100111011111010100011011001110111110110",
        5 => "10010000111001111101100101011011110001100000100110110",
        6 => "11101010010011100001101000001000101011011000111100110",
        7 => "10001010011111011110011011010001110101100000100001101"
);

--
constant basisSize : natural := getAllBlocksCount(N);
type basis_array is array (0 to basisSize-1) of unsigned(rnsBC - 1 downto 0);
constant basis : basis_array := (
        "011111",
        "000110",

        "010011",
        "010110",
        "101000",

        "001100",
        "100000",
        "010100",
        "010100",

        "011011",
        "011111",
        "010011",
        "000101",
```

```
        "001010",

        "110001",
        "100110",
        "110010",
        "011011",
        "100101",
        "001001",

        "010100",
        "010100",
        "001100",
        "110101",
        "101111",
        "011111",
        "110001",

        "100111",
        "111011",
        "100010",
        "110101",
        "110111",
        "100010",
        "110111",
        "100001"

    );

--  constant r1 :  := m(0);
--  constant r2 : unsigned(3*rnsBC-1 downto 0) := m(0)*m(1);
--  constant r3 : unsigned(3*rnsBC-1 downto 0) := m(0)*m(1)*m(2);

    -- основание¤
    type m_array is array (0 to N-1) of unsigned(rnsBC - 1 downto 0);
    constant m : m_array := (
        "011111",
        "100101",
        "101001",
        "101011",
        "101111",
        "110101",
        "111011",
```

```
            "111101"
        );
            type r_array is array (0 to N-1) of unsigned((N-1)*rnsBC-1 downto 0);
        constant r : r_array := (
            0 => (0 => '1', others => '0'),
            1 => resize(m(0), (N-1)*rnsBC),
            2 => resize(m(0)*m(1), (N-1)*rnsBC),
            3 => resize(m(0)*m(1)*m(2), (N-1)*rnsBC),
            4 => resize(m(0)*m(1)*m(2)*m(3), (N-1)*rnsBC),
            5 => resize(m(0)*m(1)*m(2)*m(3)*m(4), (N-1)*rnsBC),
            6 => resize(m(0)*m(1)*m(2)*m(3)*m(4)*m(5), (N-1)*rnsBC),
            7 => resize(m(0)*m(1)*m(2)*m(3)*m(4)*m(5)*m(6), (N-1)*rnsBC)
        );
        -- диапазон
        constant P : unsigned(pBC - 1 downto 0) :=
"10000011111001111011001000110001000100100010 1";

        --
        type rns_number is array (natural range <>) of unsigned(rnsBC-1 downto 0);
end rns_constants_pkg;

package body rns_constants_pkg is

        function getAllBlocksCount(num: in natural) return natural is
            variable res: natural;
        begin
            res := 0;
            if num > 1 then
                for i in 2 to num loop
                    res := res + i;
                end loop;
            end if;
            return res;
        end function getAllBlocksCount;

        function getBlockStart(num: in natural) return natural is
        begin
            return getAllBlocksCount(num-1);
        end function getBlockStart;

end rns_constants_pkg;
```

ПРИЛОЖЕНИЕ 4

XIII Московский международный
Салон изобретений и инновационных технологий

«АРХИМЕД-2010»

ДИПЛОМ

Решением Международного Жюри
награждается

ЗОЛОТОЙ МЕДАЛЬЮ

Ставропольский военный институт
связи ракетных войск
за разработку «Модулярный нейрокомпьютер RISC
структуры на базе ПЛИС серии Spartan-3E»
(Червяков Н.И., Малофей А.О., Евдокимов А.А.,
Лавриненко А.В.)

Председатель
Международного Жюри,
Президент Евразийской
патентной организации
А.Н. Григорьев

Президент Салона
Д.И. Зезюлин

Руководитель
Федеральной службы по
интеллектуальной собственност
патентам и товарным знакам
Б.П. Симонов

Россия, Москва, 30.03 - 02.04.2010 г.

ПРИЛОЖЕНИЕ 5

VII МОСКОВСКИЙ МЕЖДУНАРОДНЫЙ САЛОН ИННОВАЦИЙ И ИНВЕСТИЦИЙ

ДИПЛОМ

награждается

Серебряной медалью

Червяков Н.И., Сивоплясов Д.В., Кондрашов А.В., Лавриненко А.В.

за разработку

Инновационные технологии разработки нового класса модулярных нейропроцессоров цифровой обработки сигналов

Министр образования и науки Российской Федерации

А. А. Фурсенко

МОСКВА, ВВЦ, 2007