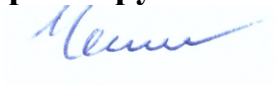
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
Институт компьютерных технологий и информационной безопасности
Кафедра интеллектуальных и многопроцессорных систем

На правах рукописи



ЧЕКИНА МАРИЯ ДМИТРИЕВНА

МЕТОДЫ И СРЕДСТВА ОБРАБОТКИ ФРАКТАЛОВ НА РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Специальность: 2.3.5 – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей, технические науки

ДИССЕРТАЦИЯ НА СОИСКАНИЕ УЧЕНОЙ СТЕПЕНИ КАНДИДАТА ТЕХНИЧЕСКИХ НАУК

Научный руководитель:
доктор технических наук,
профессор **И.И. Левин**

Таганрог – 2023 г.

СПИСОК СОКРАЩЕНИЙ

ПЛИС – программируемая логическая интегральная схема

РВС – реконфигурируемая вычислительная система

ЦП – центральный процессор

МВС – многопроцессорная вычислительная система

ЭВМ – электронная вычислительная машина

СИФ – система итерируемых функций

CPU – central processing unit, центральное процессорное устройство

GPU – graphics processing unit, графический процессор

RAM – Random Access Memory, память с произвольным доступом

ОЗУ – оперативное запоминающее устройство

СЛАУ – система линейных алгебраических уравнений

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы. Геометрия объектов различных размеров - от атомов и субатомных частиц до галактик - занимает одно из центральных мест в моделях, создаваемых для изучения окружающего мира. Траектории элементарных частиц, форма морских течений и очертания береговых линий, взаимное расположение молекул в кристаллической решётке - геометрия природы рассматривается и является одним из важнейших факторов во многих областях естествознания.

Долгое время классическая наука для описания природных объектов и процессов использовала только объекты евклидовой геометрии. В то же время очевидно, что объекты реальной природы только в исключительных случаях обладают простой геометрической формой. Реальные горы не являются конусами, облака - сферами, а береговые линии - набором дуг. При попытке же их детального описания в виде набора объектов евклидовой геометрии, полученная модель будет переусложнённой и малоприменимой для практических вычислений.

При решении задач, связанных с описанием реальных природных объектов и не только, возникает необходимость оперировать самоподобными структурами — объектами, в точности ли приближённо совпадающими с частью себя самого (то есть целое имеет ту же форму, что и одна или более частей). В 1975 году Бенуа Мандельброт ввел термин «фрактал», который образован от латинского причастия fractus, соответствующий которому глагол frangere переводится как ломать, разламывать. В работах Мандельброта придается особое значение тому, что речь тут идёт об образовании фрагментов неправильной формы.

Фракталами называют объекты, в точности или полностью совпадающие с частью самих себя, то есть обладающие свойством самоподобия. Математический аппарат, основанный на фракталах и их свойствах, находит применение в различных областях, таких как, математическое моделирование, радиотехника и информационные технологии.

При помощи фрактальных алгоритмов решают такие актуальные задачи, как анализ финансовых рынков, фрактальное сжатие изображений, моделирование пористых веществ, фильтрации жидкостей и природных объектов, обладающих сложной геометрией.

Применение фрактального принципа при конструировании активной части антенны позволяет значительно уменьшить её габариты, а также расширить полосу рабочих частот [1]. При этом активная часть имеет вид какой-либо самоподобной кривой, либо состоящей из подобных сегментов фигуры [2].

Фрактальное сжатие изображений — алгоритм сжатия изображений с потерями, основанный на применении систем итерируемых функций (как правило, являющихся аффинными преобразованиями) к изображениям. Данный алгоритм известен тем, что в некоторых случаях позволяет получить очень высокие коэффициенты сжатия при приемлемом визуальном качестве для реальных фотографий природных объектов. Особенную актуальность приобретает применение фрактального сжатия изображений на фоне постоянно растущего количества графической информации, обрабатывающейся в цифровых системах, передающейся по линиям связи и хранящейся в банках данных.

Фракталы широко применяются в компьютерной графике для построения изображений природных объектов [3, 4]. Многие природные явления демонстрируют некоторую форму статистического самоподобия, которая может моделироваться фрактальными поверхностями. Кроме того, вариации текстуры поверхности обеспечивают важные визуальные ориентиры для ориентации и уклонов поверхностей, а использование почти самоподобных фрактальных узоров может помочь создать естественные визуальные эффекты. Моделирование шероховатых поверхностей Земли с помощью дробного броуновского движения впервые было предложено Б. Мандельбротом, и было в последствие развито Л. Карпентером. Из-за особенностей задачи компьютерной генерации ландшафтов фрактальные методы стали первым эффективным способом ее решения [5].

Применение фракталов позволяет моделировать сложные структуры неупорядоченных сред и протекающие в них процессы. Примерами неупорядоченных сред являются пористые тела. При этом фракталами могут быть поровое пространство, скелет породы, поверхность скелета породы и т.д. В сплошной среде средний квадрат расстояния, на которое удаляется от исходной точки случайно блуждающая частица пропорционален времени. Во фрактальной среде случайно блуждающая частица будет удаляться от места старта медленнее, т. к. не все направления для нее доступны. Средний квадрат расстояния для фрактальной среды пропорционален некоторой дробной степени времени, показатель которой связан с размерностью Хаусдорфа-Безиковича для соответствующего среде фрактала. Множество препятствий (узких мест, крутых поворотов и тупиков) затрудняют продвижение частиц и замедляют диффузию, следовательно картина распространения вещества во фрактальной среде не соответствует аналогичному процессу для сплошной среды.

Аномальная диффузия описывает процессы переноса на стационарных процессах, характеризующиеся нестационарным распределением частиц в пространстве. Если степень дробного оператора больше единицы, то такой случай называют супердиффузией (ускоренное блуждание частиц), при значении этого параметра меньше единицы говорят о субдиффузии (замедленное блуждание частиц). Для описания этого явления используется дифференциальное уравнение дробного порядка, при этом порядок дробной производной связан с размерностью фрактала. К процессам супердиффузии относится перенос техногенных (захоронение отходов) и естественных (радон) радионуклидов, миграция природного газа по трещинам в горных породах и другие фрактальные процессы.

В отличие от теоретических фрактальных кривых, которые легко поддаются измерению, природные системы являются источниками неоднородности и генерируют сложные пространственно-временные структуры, которые могут демонстрировать только частичное самоподобие. Используя фрактальный анализ, можно анализировать и распознавать изменения характеристик сложных

экологических систем, поскольку фракталы способны характеризовать их естественную сложность. Фрактальный анализ широко используется для изучения структур, а также пространственной и временной сложности в биологических системах, и его широко применяют в экологических исследованиях.

Пространственные паттерны и последовательности поведения животных во фрактальном времени [6] имеют оптимальный диапазон сложности, который можно рассматривать как гомеостатическое состояние в спектре, где последовательность сложности должна регулярно падать. Используя фрактальный анализ, можно изучить сложную последовательность поведения животных. Важным преимуществом фрактального анализа является возможность мониторинга состояния популяций диких животных в их естественной среде обитания без инвазивных измерений. Также фрактальный анализ позволял выявлять поведенческие последствия различных стресс-факторов там, где классические частотные методы показали свою несостоятельность [7]. Паттерны в поведении животных также проявляют фрактальные свойства в пространственном и временном масштабе. Фрактальный анализ помогает понять поведение животных и то, как они взаимодействуют с окружающей средой в различных масштабах (количества, пространства и времени). Было обнаружено, что различные сигнатуры движения животных в их соответствующих средах демонстрируют пространственно-нелинейные фрактальные структуры.

Методы фрактального анализа применимы не только в изучении природных систем, но и при прогнозировании изменений курсов валют или ценных бумаг на финансовом рынке [8]. Современный финансовый рынок характеризуется значительной сложностью протекающих на нем процессов. С одной стороны финансовый рынок достаточно хаотичен, поскольку его эволюция определяется волей большого количества людей, а с другой в нем действуют устойчивые механизмы, определяемые коллективным поведением участников [9]. Стандартные методы моделирования временных рядов для анализа и

прогнозирования процессов, происходящих на финансовых рынках, часто дают неудовлетворительные результаты.

Благодаря своим свойствам фракталы находят широкое применение при описании и моделировании природных и техногенных явлений, сложных физических и социально-экономических систем. А с ростом производительности современных вычислительных систем появляются новые области применения фрактальных алгоритмов.

Главной особенностью фрактальных алгоритмов является обработка больших объемов данных, при этом возникает необходимость обмена данными между параллельно выполняемыми процессами, что мешает производить эффективное масштабирование алгоритмов при увеличении количества вычислительных узлов.

Степень разработанности темы исследования. Первым трудом, посвященным изучению фракталов является «Фрактальная геометрия природы» Б. Мандельброта. Именно здесь вводится понятие фрактала так, как оно будет использоваться в дальнейшем. Также именно Мандельбротом было проведено первое компьютерное моделирование фракталов. Ранее рядом исследователей выделялись объекты, обладающие фрактальными свойствами: функция Вейерштрасса, кривая Леви, кривая Коха – но ни один из предшественников Мандельброта не выделял самоподобные множества в отдельный класс.

Идеи Мандельброта продолжил развивать Е. Федер. В его работе «Фракталы» описаны приложения теории фракталов к различным областям математического моделирования – океанологии, гидродинамике и исследованию перколяционных процессов т.д. Помимо перечисленного в данной работе предложены методы построения фрактальных ландшафтов, заложившие основы компьютерной графики, использующейся сейчас для создания мультимедийной продукции.

Способы применения теории фракталов в компьютерной графике активно разрабатывались соучредителем компании Pixar Animation Studios Л.

Карпентером, который создал метод рендеринга фотореалистичных изображений Reyes. М. Барнсли в 1987 году предложил идею сжатия данных, основанную на самоподобии отдельных участков, что впоследствии заложило фундамент алгоритмов фрактального сжатия изображений.

Реализация фрактальных алгоритмов возможна на устройствах различного типа - от персональных компьютеров до многопроцессорных вычислительных систем. Однако приложения, реализующие фрактальные алгоритмы для традиционных многопроцессорных систем, обладают рядом недостатков. Для систем на универсальных процессорах характерны большие накладные расходы на передачу данных между процессами, и возникает проблема с синхронизацией процессов, приводящая к простою части вычислительного оборудования [10, 11]. Графические ускорители обладают большой вычислительной мощностью, но не могут функционировать как самостоятельное устройство. При выполнении фрактального алгоритма на системе, основным вычислителем которой является графический ускоритель, требуется постоянный обмен данными между ним, центральным процессором и оперативной памятью. Пропускная способность этих каналов существенно ниже, чем частота работы графического ускорителя, что приводит к снижению производительности системы.

Программируемые логические интегральные схемы потенциально имеют большие возможности для решения задач фрактального типа, благодаря возможности адаптировать структуру вычислительного узла для каждого типа задач. Однако существующие реализации фрактальных алгоритмов предназначены для ускорителей на основе только одной ПЛИС, которые не являются независимыми устройствами и для их работы требуется персональный компьютер. Разработчики не предусматривают возможность масштабирования алгоритмов для системы, содержащей более одного кристалла.

В отличие от выше описанных устройств реконфигурируемые вычислительные системы (РВС), объединяющие множество ПЛИС в единое вычислительное поле, используют структурную парадигму вычислений.

Благодаря этому РВС успешно позволяет ре-шить проблему «бутылочного горла», изменяя гибкую логику ПЛИС и выстраивая пространственные коммутации между вычислительными блоками. На данный момент не существует для РВС методов представления и обработки данных, описывающих самоподобные структуры, для которых характерна нерегулярность данных, а также необходимость хранения и обработки помимо полезных данных управляющей информации, которая определяет связность блоков и может динамически меняться по ходу решения задачи. Существующие методы обработки данных на РВС не позволяют эффективно задействовать вычислительный ресурс при решении задачи фрактального типа. Таким образом, возникает необходимость в создании новых методов и средств для эффективной работы с задачами фрактального типа на РВС, обеспечивающими масштабирование приложений.

В диссертации был выполнен обзор существующих методов и средств решения сильносвязных задач на РВС. Метод распараллеливания по итерациям на РВС, реализованный Левиным И.И., позволяет сократить количество вычислительного ресурса, затрачиваемого на подсистему коммутации, и количество памяти, используемой для хранения промежуточных данных. Данный метод в дальнейшем для решения задач на РВС использовали: Пелипец А.В. – для задач линейной алгебры, Коваленко А.Г. – для задач математической физики, Семерников Е.А. – для задач цифровой обработки сигналов, Михайлов Д.В. - для решения задач с последовательным информационным графом. Однако при использовании метода распараллеливания по итерациям для решения задач фрактального типа невозможно получить изоморфные информационные графы для эффективного масштабирования вычислительной структуры, т.к. для изоморфных графов характерны как постоянные вычислительные структуры, так и постоянные потоки данных. В данной работе впервые решается задача создания эффективных параллельно-конвейерных программ для решения задач фрактального типа на реконфигурируемых вычислительных системах.

Цель исследования: повышение реальной производительности РВС при решении задач фрактального типа.

Объект исследования: задачи фрактального типа.

Предмет исследования: методы, алгоритмы и программные средства для решения задач фрактального типа на РВС.

Научная задача, решаемая в диссертации, состоит в разработке методов и средств обработки самоподобных структур, повышающих реальную производительность реконфигурируемых вычислительных систем при решении задач фрактального типа.

Для достижения сформулированной цели и решения научной задачи необходимо:

1) провести анализ существующих вычислительных средств и методов, позволяющих решать задачи фрактального типа, и существующих способов представления фрактальных данных в вычислительных системах;

2) разработать методы решения на РВС параллельно-конвейерным способом задач фрактального сжатия и декомпрессии изображений, обеспечивающие максимальное задействование вычислительного ресурса системы;

3) разработать метод синтеза вычислительной структуры для решения на РВС задачи распространения газа во фрактальной среде, использующий подобные подграфы;

4) разработать программные средства для решения задач фрактального сжатия и декомпрессии изображений и задачи распространения газа во фрактальной среде.

Методы исследований. В ходе исследований были использованы методы структурной организации вычислений на РВС, основы линейной алгебры, методы дискретизации математических моделей, методы решения систем линейных алгебраических уравнений. Практические исследования проведены на РВС последних поколений.

Положения, выносимые на защиту:

1) при решении задач фрактального типа на традиционных многопроцессорных вычислительных системах рост производительности замедляется с увеличением числа задействованных узлов, вследствие увеличения объемов данных, пересылаемых между процессорами и памятью устройства, а необходимость синхронизации параллельных процессов приводит к простою оборудования;

2) разработанные методы параллельно-конвейерной реализации фрактального сжатия и декомпрессии изображений для РВС обеспечивают значительный рост реальной производительности при увеличении вычислительного ресурса системы по сравнению с многопроцессорными приложениями.

3) разработанный метод синтеза вычислительной структуры для решения на РВС задачи распространения газа во фрактальной среде обеспечивает увеличение реальной производительности по сравнению с классическими способами решения СЛАУ на РВС.

Результаты, выносимые на защиту:

1) метод решения на РВС параллельно-конвейерным способом задачи фрактального сжатия изображений, **отличающийся** от известных побитовой обработкой данных, обеспечивающей максимальное задействование вычислительного ресурса системы, и сортировкой структур, содержащих выходные данные, по номеру рангового блока;

2) метод решения на РВС параллельно-конвейерным способом задачи декомпрессии сжатых изображений, **отличающийся** от известных использованием одинарной косвенной адресации для блоков памяти, содержащих доменные блоки.

3) метод синтеза вычислительной структуры для решения на РВС задачи распространения газа во фрактальной среде, **отличающийся** от известных возможностью оптимизации вычислительной структуры для конкретной СЛАУ на

основе оценки ее параметров и использованием подобных вычислительных подграфов в конвейере.

Теоретическая значимость научных результатов. Результаты, полученные в ходе работы над диссертацией, являются важными в развитии теории решения задач фрактального типа. Автором сформулированы принципы организации эффективной обработки фрактальных структур на РВС, и показано, что при решении задач согласно этим принципам обеспечивается рост реальной производительности при увеличении вычислительных ресурсов системы по сравнению с известными многопроцессорными реализациями.

Автором доказано, что реальная производительность традиционных вычислительных систем при решении задач фрактального типа при линейном увеличении вычислительного ресурса снижается.

Практическая значимость.

Разработаны программные средства на языке программирования COLAMO для решения на РВС задач фрактального сжатия и декомпрессии изображений и задачи супер-диффузии (Свидетельства о государственной регистрации программ для ЭВМ № 2023614463, № 2023617300, № 2023617550, РФ), и синтезированы соответствующие вычислительные структуры

Разработанные программные средства обеспечивают ускорение в 15000 раз при выполнении на РВС фрактального сжатия по сравнению с аналогичной реализацией для многоядерного универсального процессора. Сравнение времени решения с другими существующими реализациями на ПЛИС показало повышение реальной производительности в 10-40 раз для реализаций на одном кристалле, и более чем в 50 раз по сравнению с графическими ускорителями. При декомпрессии изображения обеспечивается ускорение в 380 раз по сравнению с аналогичной реализацией для многоядерного универсального процессора.

Применение предложенного метода модификации вычислительной структуры метода Якоби для решения задачи распространения газа во

фрактальной среде позволяет повысить эффективность вычислений на 40% по сравнению с известными способами реализации этой задачи на РВС.

Степень достоверности результатов, полученных соискателем, подтверждена корректностью, непротиворечивостью математических выкладок и результатами машинных экспериментов. Результаты диссертации докладывались и обсуждались на всероссийских научно-технических конференциях, где соискатель выступал с докладами по данной проблематике и получил положительный отзыв научной общественности.

Апробация работы. Основные результаты, представленные в диссертации, докладывались и обсуждались на всероссийских научно-технических конференциях: 3-й Всероссийской научно-технической конференции «Суперкомпьютерные технологии (СКТ-2014)» с. Дивноморское, Геленджик, 2014 г., XVI ежегодной молодежной научной конференции «Юг России: вызовы времени, открытия, перспективы», Таганрог, 2020 г., XVII Ежегодной молодежной научной конференции "Наука и технологии Юга России", Таганрог, 2021 г., XIV Всероссийской Мультиконференции по проблемам управления (МКПУ-2021) с. Дивноморское, Геленджик, 2021 г., XVIII ежегодной молодежной научной конференции «Наука Юга России: достижения и перспективы» , Таганрог, 2022 г., Всероссийской научно-технической конференции «Многопроцессорные вычислительные системы» (МВУС-2022) Таганрог, 2022 г., XIX ежегодной молодежной научной конференции «Достижения и перспективы научных исследований молодых ученых Юга России», Таганрог, 2023 г.

В 2022 г. цикл работ «Методы обработки фракталов на реконфигурируемых вычислительных системах» был удостоен премии для молодых ученых имени академика А.В. Каляева.

Результаты диссертации внедрены в ООО «НИЦ СЭ и НК», акт о внедрении от 21 апреля 2023 г., утвержден техническим директором. Материалы диссертации использовались:

- в рамках НИОКР «Разработка нового поколения программного обеспечения РВС для решения ресурсоёмких научно-технических задач различных предметных областей», шифр «Поколение» при реализации модельных параллельно-конвейерных программ, выполняющих обработку данных;
- в рамках НИР «Выбор направления исследований и разработки высокопроизводительных вычислительных систем «СКИФ-ГЕО» второй базовой конфигурации, обеспечивающих формирование оптимальных конфигураций аппаратных средств и программного обеспечения для решения расчетных геолого-геофизических задач», шифр «Нефть» № гос. рег. 115092410004 по договору № 2-26-06-СКИФ/НИР/15 от 22.06.2015.

Результаты диссертации внедрены в учебном процессе кафедры интеллектуальных и многопроцессорных систем (ИМС) Института компьютерных технологий и информационной безопасности (ИКТИБ) Южного федерального университета (ЮФУ), акт от 20 апреля 2023 г., утвержден директором ИКТИБ ЮФУ. Материалы диссертации используются в лекционных курсах по дисциплинам «Современные проблемы прикладной математики и информатики» (тема № 11 «Фракталы. Дробные производные») и «Математическое моделирование процессов гидрофизики на высокопроизводительных вычислительных системах» (модуль 2 «Численная реализация математических моделей гидрофизики»), для подготовки магистров направления подготовки 01.04.02 Прикладная математика и информатика (образовательные программы «Прикладная математика для высокопроизводительных вычислительных систем», «Математическое моделирование в инженерных науках»).

Личный вклад автора. Все представленные в диссертации результаты получены лично автором.

Наиболее значительными публикациями по теме диссертации являются:

1. Чекина, М.Д. Математическое моделирование задач геофильтрации в почвогрунтах с фрактальной структурой на многопроцессорных вычислительных системах // Известия ЮФУ. Технические науки. – 2014. – № 12 (161). – С. 242-251. **(ведущий рецензируемый журнал, входит в перечень ВАК).**

В работе Чекиной М.Д. получен дискретный аналог для математической модели, описывающей процессы аномальной диффузии с использованием дробного оператора дифференцирования, разработана параллельный алгоритм решения, полученной СЛАУ, для моделирования на сетках с большим разрешением.

2. Чекина, М.Д. Решение разреженных СЛАУ большой и сверхбольшой размерности многосеточным методом на РВС // М.Д. Чекина, А.В. Подопригора / Известия ЮФУ. Технические науки. – 2018. – № 8 (202). – С. 212-221. DOI: 10.23683/2311-3103-2018-8-212-221 **(ведущий рецензируемый журнал, входит в перечень ВАК).**

В работе Чекиной М.Д. проведен анализ возможности применения многосеточного метода к решению на РВС больших разреженных матриц.

3. Чекина, М.Д. Реализация фрактального сжатия и декомпрессии изображений параллельно-конвейерным способом на реконфигурируемых вычислительных системах // Известия ЮФУ. Технические науки. – 2020. – № 7 (217). – С. 130-142. DOI: 10.18522/2311-3103-2020-7-130-142 **(ведущий рецензируемый журнал, входит в перечень ВАК).**

В работе Чекиной М.Д. предложен подход к реализации фрактального сжатия и декомпрессии изображений, позволяющих получать параллельно-конвейерные программы для РВС различных конфигураций.

4. Чекина, М.Д. Параллельно-конвейерная реализация фрактального сжатия изображений на реконфигурируемых вычислительных системах // М.Д. Чекина, И.И. Левин / Вестник компьютерных и информационных технологий. – 2021. – Т. 18. – № 4 (202). – С. 37-44. DOI: 10.14489/vkit.2021.04.pp.037-044 **(ведущий рецензируемый журнал, входит в перечень ВАК).**

В работе Чекиной М.Д. предложен метод создания параллельно-конвейерных структур для фрактального сжатия изображений.

5. Чекина, М.Д. Модификация реализации метода Якоби при моделировании супердиффузии радона на реконфигурируемых вычислительных системах // Известия ЮФУ. Технические науки. – 2021. – № 7 (224). – С. 198-206. DOI: 10.18522/2311-3103-2021-7-198-206 **(ведущий рецензируемый журнал, входит в перечень ВАК).**

В работе Чекиной М.Д. предложена модификация метода Якоби для решения задачи супердиффузии на РВС, которая позволяет строить вычислительные структуры, исходя из параметров входной СЛАУ.

6. **Свидетельство о государственной регистрации программ для ЭВМ** № 2023614463, РФ. Программа фрактального сжатия изображений // Чекина М.Д., Левин И.И. Зарегистр. в Реестре программ для ЭВМ 01.03.2023 г. Правообладатель: ООО «НИЦ супер-ЭВМ и нейрокомпьютеров».

Вклад Чекиной М.Д. в создание зарегистрированной программы для ЭВМ состоит в разработке алгоритма создания параллельно-конвейерных структур для фрактального сжатия изображений.

7. **Свидетельство о государственной регистрации программ для ЭВМ** № 2023617300, РФ. Программа декомпрессии изображений, сжатых фрактальным способом // Чекина М.Д., Левин И.И. Зарегистр. в Реестре программ для ЭВМ 07.04.2023 г. Правообладатель: ООО «НИЦ супер-ЭВМ и нейрокомпьютеров».

Вклад Чекиной М.Д. в создание зарегистрированной программы для ЭВМ состоит в разработке алгоритмов декомпрессии изображения, сжатого фрактальным способом, для применения на реконфигурируемых вычислительных системах.

8. **Свидетельство о государственной регистрации программ для ЭВМ** № 2023617550, РФ. Программа для решения задачи супердиффузии //

Чекина М.Д., Левин И.И. Зарегистр. в Реестре программ для ЭВМ 11.04.2023 г. Правообладатель: ООО «НИЦ супер-ЭВМ и нейрокомпьютеров».

Вклад Чекиной М.Д. в создание зарегистрированной программы для ЭВМ состоит в разработке алгоритмов решения задачи супердиффузии для применения на реконфигурируемых вычислительных системах.

9. Чекина, М.Д. Математическое моделирование задач геофильтрации в почвогрунтах с фрактальной структурой на многопроцессорных вычислительных системах // Материалы 3-й Всероссийской научно-технической конференции «Суперкомпьютерные технологии (СКТ-2014)». – 2014. – С. 258-262.

10. Чекина, М.Д. Решение больших и сверхбольших разреженных СЛАУ на реконфигурируемых вычислительных системах // М.Д. Чекина, А.В. Подопригора / Суперкомпьютерные технологии (СКТ-2018). Материалы 5-й Всероссийской научно-технической конференции. – 2018. – Т. 1. – С. 201-204.

В работе Чекиной М.Д. проведен анализ необходимых вычислительных ресурсов для обработки больших разреженных матриц.

11. Чекина, М.Д. Реализация фрактальных алгоритмов на реконфигурируемых вычислительных системах // Суперкомпьютерные технологии: сборник трудов молодых ученых. – 2020. – С. 129-133.

12. Чекина, М.Д. Управление анализом сейсмической активности с применением модели супердиффузии радона на реконфигурируемых вычислительных системах // Материалы XIV Всероссийской Мультиконференции по проблемам управления (МКПУ-2021): в 4 т. – 2021. – Т. 2. – С. 278-280.

13. Чекина, М.Д. Особенности решения задачи супердиффузии с использованием адаптивных сеток на РВС // XVII ежегодная молодежная научная конференция «Наука и технологии Юга России». Материалы XVII Ежегодной молодежной научной конференции. – 2021. – С. 306.

14. Чекина, М.Д. Особенности определения параметров для решения задачи супердиффузии с использованием адаптивных сеток на РВС // XVIII Ежегодная

молодежная научная конференция «Наука Юга России: достижения и перспективы». Материалы XVIII ежегодной молодежной научной конференции. – 2022. – С. 267.

15. Чекина, М.Д. Решение задачи супердиффузии на реконфигурируемых вычислительных системах // Всероссийская научно-техническая конференция «Многопроцессорные вычислительные системы» (МВУС-2022). Материалы всероссийской научно-технической конференции. – 2022. – С. 155-160.

16. Чекина, М.Д. Повышение эффективности решения задачи супердиффузии с использованием адаптивных сеток на РВС // Материалы XIX Ежегодной молодежной научной конференции «Достижения и перспективы научных исследований молодых ученых Юга России». – 2023. – С. 295.

Во всех работах, выполненных в соавторстве, определен личный вклад автора.

Структура работы. Диссертация состоит из введения, пяти глав, заключения, списка использованных источников и приложений.

Во введении обосновывается актуальность диссертации, формулируется цель и обосновывается научная новизна исследования, показана практическая значимость результатов, представлены научные положения, выносимые на защиту, а также приводится краткое содержание каждой главы.

В первой главе приводится анализ применения фракталов в различных предметных областях, и показано, что с ростом производительности современных вычислительных систем появляются новые области применения фрактальных алгоритмов.

Показано, что фрактальные алгоритмы обладают большой вычислительной сложностью, и требуют больших вычислительных мощностей, но при этом фрактальные алгоритмы требуют постоянного обмена данными между вычислительными узлами, т.к. обладают высокой связностью, что затрудняет их масштабирование.

Показано, что существующие реализации фрактальных алгоритмов для многопроцессорных вычислительных систем, построенных на основе

графических ускорителей или универсальных процессоров, достигают меньшей производительности, чем реализации для устройств на основе ПЛИС.

Выполнен обзор существующих реализаций фрактальных алгоритмов на ПЛИС. Выявлены недостатки существующих реализаций, основным из которых является отсутствие возможности их масштабирования на несколько ПЛИС.

Произведен обзор структур данных, используемых для обработки фракталов в классических вычислительных системах. Рассмотрены двоичные, четверичные и восьмеричные деревья, хранение систем итерируемых функций и их коэффициентов. Указан основной недостаток данных структур – требования динамического выделения оперативной памяти.

Проведен анализ описанных выше структур на предмет возможности их применения при реализации фрактальных алгоритмов на РВС. Описаны способы представления данных, применяющиеся в РВС (линейная и косвенная адресация памяти, множества, пакеты данных), и показана необходимость создания специальных структур данных для работы с фрактальными структурами на РВС.

Сформулированы принципы синтеза эффективных параллельно-конвейерных программ для обработки самоподобных структур на РВС: необходимость использования структурного подхода организации вычислений для работы с самоподобными структурами на РВС; предпочтение распараллеливанию по итерациям, а не по слоям; объединение в вычислительные структуры подобных подграфов, а не изоморфных для масштабирования вычислений при реализации фрактальных алгоритмов на РВС; использование предварительной сортировки данных для увеличения эффективности алгоритма на РВС; уменьшение скважности потока операндов методом конвейера в конвейере или автоподстановки для увеличения быстродействия вычислительной структуры, содержащей обратные связи.

Во второй главе предложен метод создания масштабируемых вычислительных структур для фрактального сжатия изображений. Для этого созданы параллельные и последовательные реализации базовых операций

фрактального сжатия изображений, выделен базовый подграф алгоритма, проведена редукция по критическим вычислительным ресурсам и обоснован выбор редукции по разрядности входных данных. Предложенный метод отличается от известных побитовой обработкой данных, обеспечивающей максимальное использование вычислительного ресурса системы, и сортировкой структур, выходных данных, по номеру рангового блока.

Введена новая структура данных, содержащая информацию о соответствии между конкретными ранговым и доменным блоками, что значительно облегчает последующее восстановление изображения по коэффициентам системы итерируемых функций. Данная структура отличается от известных использованием только одной косвенной адресации для блоков памяти, содержащих доменные блоки. Применение предложенной структуры облегчает хранение данных и последующую работу с результатами архивации.

Выполнено сравнение производительности реализации фрактального сжатия изображений на РВС по сравнению с аналогичной для универсальных процессоров. Показано, что полученная реализация фрактального сжатия изображений показывает ускорение в 15000 раз по сравнению с аналогичной реализацией для многоядерного универсального процессора.

Проведено сравнение подхода, предложенного в диссертации, с существующими однокристальными реализациями фрактального сжатия [12-14], при этом ускорение составило 10-40 раз.

Предложен метод создания масштабируемых вычислительных структур для процедуры декомпрессии изображения, сжатого фрактальным способом, на РВС. Для этого выполнена оптимизация процесса формирования структур, содержащих информацию о сжатом изображении, что позволяет отказаться от использования обратной связи и обеспечивает возможность параллельного выполнения разархивации.

Выполнено сравнение производительности декомпрессии изображения, сжатого фрактальным способом, на РВС с аналогичной реализацией для

многоядерного универсального процессора, полученное ускорение составило более 50 раз.

В третьей главе предложен метод создания масштабируемых вычислительных структур для решения задачи фрактального распространения газов (супердиффузии) на РВС. Показано, что в результате дискретизации математической модели, описывающей супердиффузию газа, на адаптивных сетках, получается матрица с нерегулярной структурой, принципиально отличающейся от ленточных матриц, характерных для задач математической физики. В этом случае стандартные подходы к решению СЛАУ показывают недостаточную эффективность и приводят к низкой реальной производительности вычислительной системы.

Исходя из данных особенностей распределения элементов в системе линейных уравнений, предложен способ оценки оптимального количества каналов в вычислительном слое для максимизации использования оборудования. При синтезе программ приложенным методом учитываются особенности решаемых СЛАУ, и на основе таких параметров как математическое ожидание и среднеквадратичное отклонение количества элементов в строке матрицы определяются оптимальные параметры вычислительной структуры. Полученные таким способом оценки могут быть использованы для получения точных вычислительных структур и для выбора приближенной вычислительной структуры из списка готовых решений.

Проведено сравнение эффективности модифицированной вычислительной структуры решения задачи супердиффузии с классическими реализациями. Сравнение показало, что разработанная вычислительная структура обладает эффективностью, на 40% превышающей классические решения.

Предложен способ предварительной перестановки строк решаемой СЛАУ, обеспечивающий равномерную загрузку всех вычислительных слоев. Строки матрицы загружаются не в естественном порядке, а исходя из загруженности вычислительного слоя, благодаря чему сохраняется естественный порядок

получения новых значений неизвестных, и уменьшаются задержки вычислений в конвейере, обусловленные наличием информационных связей между строками СЛАУ. Данный способ дополнительно повышает эффективность работы алгоритма на 5-7%.

В заключении изложен основной научный результат диссертации, а также сформулированы теоретические и прикладные результаты, полученные в диссертационной работе, рекомендации и перспективы дальнейшей разработки темы.

В приложениях приведены:

1) в приложении 1 приведены описания программных реализаций задач фрактального сжатия и декомпрессии на языке COLAMO;

2) в приложении 2 представлены копии актов о внедрении и использовании результатов диссертационной работы от следующих организаций: НИЦ СЭ и НК (г. Таганрог), ИКТИБ ЮФУ (г. Таганрог);

3) в приложении 3 представлены копии свидетельств о государственной регистрации программ для ЭВМ.

1 АНАЛИЗ СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ ОБРАБОТКИ САМОПОДОБНЫХ СТРУКТУР

1.1 Области приложения фракталов

В настоящее время фрактальные алгоритмы используются в самых разнообразных классах задач. Например, они находят широкое применение в фрактальной графике, которая используется для создания красивых и сложных изображений, а также в фрактальном сжатии изображений, которое позволяет сохранять качество изображения при сжатии. Кроме того, фрактальный анализ финансовых рынков является одним из наиболее перспективных направлений, который позволяет предсказывать поведение рынков на основе анализа их фрактальной структуры. Фрактальные антенны используются в радиосвязи и обладают лучшими характеристиками, чем классические антенны. Фильтрация жидкостей в пористых средах – это еще одна область, в которой фрактальные алгоритмы могут применяться с большой пользой. С развитием технологий и увеличением вычислительной мощности, фрактальные алгоритмы все больше и больше используются в различных областях науки и техники. Несмотря на все преимущества, фрактальные алгоритмы обладают большой вычислительной сложностью, что усложняет их применение в реальных задачах. Требуется высокая производительность от вычислительной системы, чтобы обеспечить эффективное функционирование.

В диссертации рассматриваются перспективы реализации фрактальных алгоритмов на многопроцессорных системах различной архитектуры.

Некоторые классы задач требуют для своего решения использования самоподобных структур – объектов, в точности или приближённо совпадающих с частью себя самого в различных масштабах. Термин «фрактал» был введен Б. Мандельбротом в 1975 году.

Определение. Фракталом называется множество, размерность Хаусдорфа-Безиковича для которого строго больше его топологической размерности [15].

Также фракталами называют предметы, удовлетворяющие хотя бы одному из следующих свойств:

- Нетривиальная структура на всех масштабах. Для фрактала увеличение масштаба не ведёт к упрощению структуры, в отличие от регулярных фигур, где при увеличении масштаба в конечном итоге фрагмент фигуры можно представить в виде дуги или отрезка прямой.
- Самоподобие или приближённое самоподобие.
- Дробная метрическая размерность или метрическая размерность, превосходящая топологическую.

На рубеже XIX и XX веков изучение фракталов носило скорее эпизодический, нежели систематический характер. Тем не менее, некоторые известные тогда математические объекты обладали свойствами фракталов. В 1872 году немецкий математик Карл Вейерштрасс построил пример непрерывной функции, которая нигде не дифференцируема [16]. Также, в 1904 году Хельге фон Кох [17] придумал непрерывную кривую, которая нигде не имеет касательной. Эти два математических объекта имели свои особенности, которые существенно отличали их от привычных геометрических фигур. Таким образом, можно утверждать, что фракталы были известны математикам задолго до того, как они получили своё название.

Идеи самоподобия фигур далее разрабатывал французский математик П. П. Леви. В 1938 году вышла его статья «Плоские и пространственные кривые и поверхности, состоящие из частей, подобных целому», в которой описан еще один фрактал — С-кривая Леви. Впрочем, первые упоминания подобных кривых были сделаны еще раньше Э. Чезаро и Г. Фабером [18]. Все эти вышеперечисленные фракталы можно условно отнести к одному классу конструктивных (геометрических) фракталов.

Другой вид фракталов – это динамические (алгебраические) фракталы. Они были исследованы в начале XX века французскими математиками Г. Жюлиа и П. Фату [19], которые изучали голоморфную динамику и описали самоподобные объекты - множество Жюлиа и множество Фату (дополнение к первому). Множество Фату и множество Жюлиа имеют множество общих признаков, но различаются своими свойствами. Например, множество Жюлиа всегда замкнуто, а множество Фату открыто. Этот класс фракталов тесно связан с множеством Мандельброта (рис. 1.1) и представляет собой целое семейство фракталов.

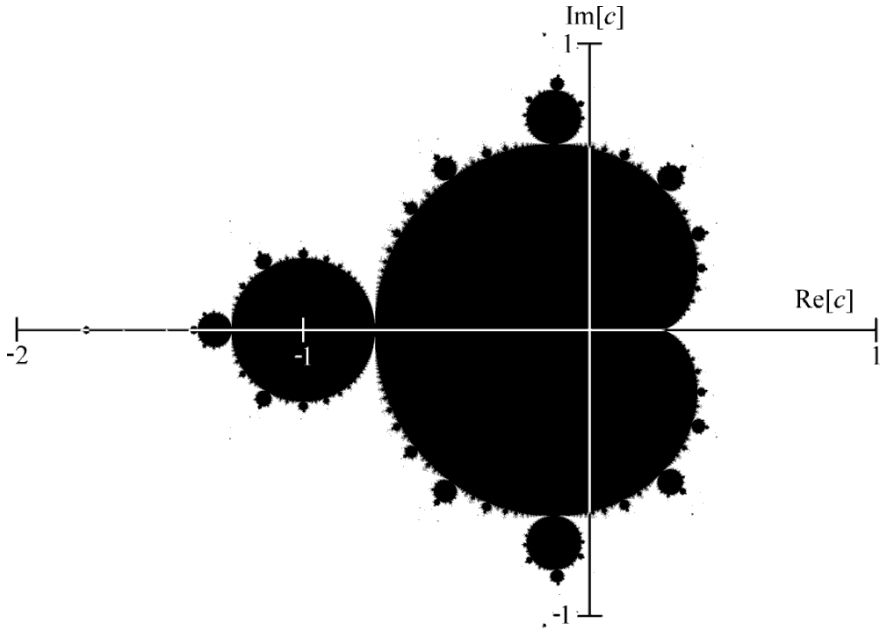


Рис. 1.1 – Множество Мандельброта

Следующий класс – стохастические фракталы. При их построении применяют случайное изменение некоторых параметров в итерационном процессе. Полученные таким образом фракталы очень похожи на природные объекты: несимметричные деревья, кораллы, особенности ландшафта [20]. Например, траектория броуновского движения на плоскости и в пространстве также является примером стохастических фракталов. Стохастические фракталы активно используются для моделирования различных процессов в экономике, физике, биологии [21] и т.д. Благодаря своей нетривиальности, стохастические

фракталы помогают улучшить точность и реалистичность моделей, а также дают возможность предсказывать поведение систем при различных условиях.

В 1977 году Б. Мандельброт в своей книге «Фрактальная геометрия природы» собрал и систематизировал практически всю имевшуюся на тот момент информацию о фракталах. Мандельброт первым применил ЭВМ для вычисления и визуализации классического фрактала, чем заложил основу для дальнейшего компьютерного моделирования самоподобных объектов.

Фракталы являются очень полезным математическим инструментом, который позволяет более точно описывать многие явления в реальном мире и находить применение в различных областях науки и техники. Такие объекты реального мира как береговые линии, облака, пористые вещества и др. обладают свойством статистического самоподобия: их части статистически однородны в разных шкалах измерения. Для описания таких объектов лучше других подходят математические модели, построенные на основе фракталов. Кроме этого, существует множество других областей приложения фракталов: фрактальное сжатие изображений, компьютерная графика, фрактальный анализ, фрактальные антенны и построение децентрализованных одноранговых компьютерных сетей. Рассмотрим данные области применения самоподобных структур более подробно.

Самоподобные объекты представляют собой важный инструмент для разработки более эффективных и компактных устройств в различных областях науки и техники. Фрактальные структуры применяются при проектировании антенных устройств, что дает возможность максимизировать эффективную длину антенны.

Показано, что выбор для формы приемного провода антенны трехсторонней кривой Коха расширяет принимаемый диапазон, и такие антенны могут быть использованы в качестве многодиапазонных [1, 22]. Также при миниатюризации антенны улучшаются показатели приема. Помимо кривой Коха для также применяются топологии в виде треугольника Серпинского и кривой Пеано. Подобные антенны находят применение в микроволновой технике и сотовой

связи [23], в том числе и для соединений типа 5G. В работе [24] показано, что применение фрактальных антенн коэффициенты усиления в 2-3 раза выше по сравнению с дипольными антеннами при тех же параметрах.

На принципе поиска самоподобных участков в изображении основан метод фрактального сжатия. Впервые возможность применения теории систем итерируемых функций (СИФ) к проблеме сжатия изображения была исследована М. Барнсли и А. Слоуном. А. Жакен представил метод фрактального кодирования, в котором используются системы доменных и ранговых блоков изображения, блоков квадратной формы, покрывающих всё изображение. Этот подход стал основой для большинства методов фрактального кодирования. Он был усовершенствован Ю. Фишером и рядом других исследователей [25]. Первый алгоритм для фрактального сжатия изображений появился в 1991 году [26]. Одним из главных его преимуществ является возможность произвольно менять разрешение изображения без появления пикселизации при распаковке. Также данный алгоритм имеет возможности для работы не только со статической графикой, но и с видео.

При общих коэффициентах сжатия, примерно до 50:1, фрактальное сжатие дает результаты, аналогичные алгоритмам на основе DCT, таким как JPEG [27]. При высоких степенях сжатия фрактальная компрессия может предложить превосходное качество. Для спутниковых снимков были получены отношения 170:1 [28]. Фрактальные коэффициенты сжатия видео 25:1–244:1 были достигнуты за разумное время сжатия (от 2,4 до 66 с/кадр). Эффективность сжатия увеличивается с более высокой сложностью изображения и глубиной цвета по сравнению с простыми изображениями в градациях серого.

Многие природные явления, такие как ландшафты и облака, могут быть описаны с помощью статистического самоподобия и моделироваться фрактальными поверхностями. Использование таких моделей может помочь улучшить наши представления о природе и легче анализировать сложные явления. Кроме того, структура поверхности может играть важную роль в

визуальной ориентации и оценке уклона поверхности. Вариации текстуры поверхности и использование стохастических фрактальных узоров могут помочь создать естественные визуальные эффекты, что может быть полезным при создании компьютерных игр и графических изображений. Моделирование шероховатых поверхностей Земли с помощью дробного броуновского движения впервые было предложено Б. Мандельбротом, и было в последствие развито Л. Карпентером. Из-за особенностей задачи компьютерной генерации ландшафтов фрактальные методы стали первым эффективным способом ее решения [5]. В качестве примера можно привести алгоритм diamond-square, который позволяет моделировать не только статические объекты ландшафта, но и динамические такие как течение жидкостей. Для этого при генерации используются случайные параметры смещения вершин сетки, такой прием получил название «эффект плазмы».

Теоретические фрактальные кривые легко поддаются измерению и не требуют больших усилий при моделировании, в то время как природные системы представляют собой сложные неоднородные и в предельном случае дискретные структуры, демонстрирующие только частичное самоподобие. Используя фрактальный анализ, можно моделировать изменения характеристик сложных экологических систем, благодаря таким свойствам фракталов как нерегулярность и тонкая структура, позволяющая работать в очень малых масштабах. Кроме того, фрактальный анализ позволяет изучать множество других свойств сложных систем, таких как разнообразие их форм, а также изменения этих форм во времени. В биологических системах фрактальный анализ применяется для изучения структур, обладающих пространственной и временной сложности. Он также находит широкое применение в экологических исследованиях, позволяя более глубоко изучать взаимодействия между различными видами и их окружением.

Пористые среды, такие как почва, горные породы, керамика и др., обладают свойством самоподобия. При моделировании фильтрации нефти, газов или их

смесей следует учитывать, что картина их просачивания через поры горной породы также будет иметь фрактальный характер. Перколяционные процессы, происходящие при движении воды в капиллярах и порах в почве также лучше описываются уравнениями, использующими дробные операторы дифференцирования, которые являются естественным математическим инструментом, для описания фрактальных процессов [29, 30]. Пример растекания воды в почве показан на рисунке 1.2.

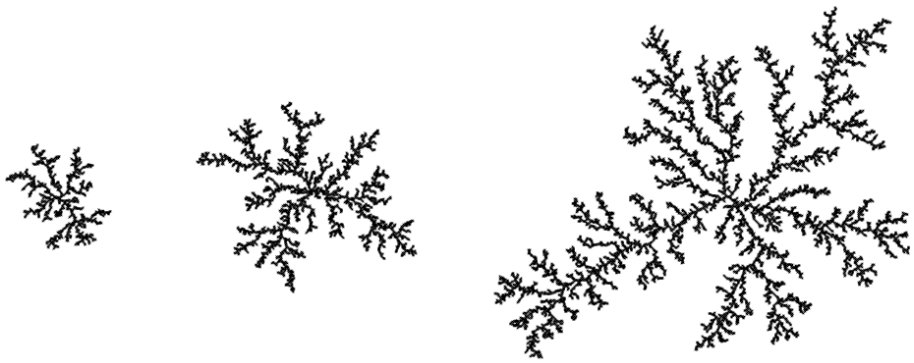


Рис. 1.2 – Фрактальная картина растекания жидкости в пористой среде

В работе [31] было показано, что применение фрактального подхода к моделированию процесса растекания жидкости в пористой среде является более точным, чем классическая модель, и это было подтверждено экспериментально. Кроме того, в последующих исследованиях была показана важность учета геометрических особенностей пористой среды для точного моделирования процессов, связанных с жидкостью. Использование фрактального подхода для моделирования процессов в пористой среде является перспективным направлением исследований, которое может привести к более точным результатам и более глубокому пониманию происходящих процессов.

Фрактальный характер распространения в пористых средах имеют не только жидкости, но и газы. Этот процесс также описывается посредством аппарата аномальной диффузии, но другим его ответвлением – супердиффузией. В работе [32] показано, что модель переноса радона в неоднородной среде, использующая супердиффузию, а не классическую, наиболее точно отражает реальные данные.

На рисунке 1.3 показаны траектории движения молекул радона в порах горной породы.

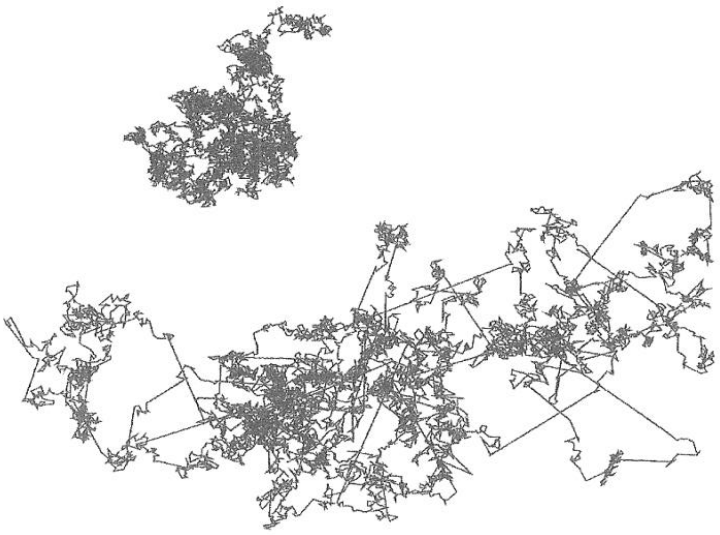


Рис. 1.3 – Траектории движения молекул газа

В работе [33] для моделирования микропористой структуры угольных коллекторов были взяты за основу структуры аналогичные губке Менгера. Фрактальные размерности полученных структур составили 2,616-3,0927. Фрактальный анализ позволяет получить более полные характеристики структуры коллектора, количественно отражает структуру пор, волнистость и шероховатость внутренней поверхности. При этом фрактальные параметры могут быть использованы в качестве важного показателя для описания характеристик структуры пор высокосортных угольных коллекторов.

Также фрактальных характер имеет растекание жидкостей в порах и капиллярах стеблей и корней растений, или костях животных [34].

Помимо пористых сред можно рассматривать как фрактальную среду снежный покров. При этом модель снежного покрова представляется в виде сферических фрактальных структур, взаимодействующих друг с другом [35] как показано на рисунке 1.4. Подобный подход позволяет добиться большей предсказательной точности по сравнению с классическими методами, и применяется при моделировании схода лавин [36].

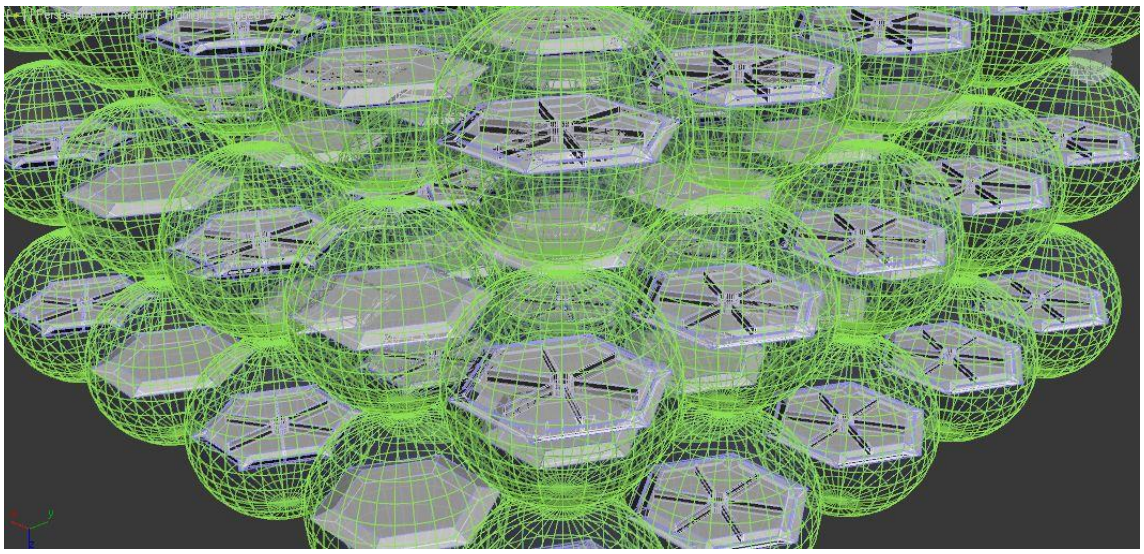


Рис. 1.4 – Модель снежного покрова

В работе [37] показано, что существует зависимость между фрактальной размерностью частиц руды и возможностью ее обогащения. В частности, было проведено изменение морфорлогии частиц медной руды путем их измельчения посредством шаровой мельницы, что позволило уменьшить фрактальную размерность частиц, приблизив их к форме шара, и улучшить параметры восстановления руды.

Инструменты фрактального анализа позволяют не только моделировать объекты неживой природы, но и предсказывать поведение животных – их взаимодействие с окружающей средой по параметрам количества, пространства и времени. Показано, что различные сигнатуры движения животных в их средах обитания демонстрируют пространственно нелинейные фрактальные структуры. Отсюда следуют такие экологические интерпретации, как гипотеза Леви, которая является более точным описанием движения животных для некоторых видов [7]. На рисунке 1.5 показана траектория перелетов чернобрового альбатроса у островов Керлеген в разных масштабах, демонстрирующих сходные закономерности сложности на всех размерах.

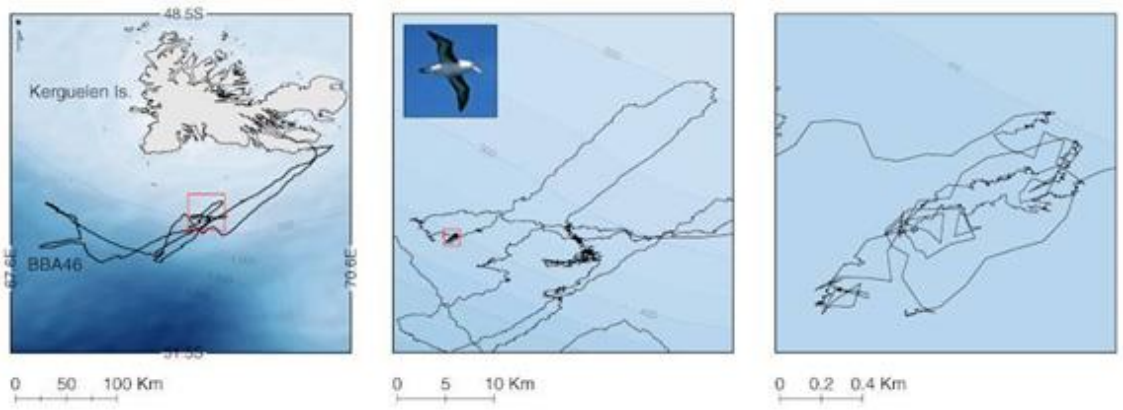


Рис. 1.5 – Траектория перемещений альбатроса

В работе [38] показано, что сообщества насекомых можно рассматривать как самоподобные объекты, и свойство фрактальности можно использовать в качестве характеристики структуры сообщества, которая отражает его адаптивные свойства как целостного эволюционного образования. На практике это означает, что, применяя свойства фракталов к имеющейся части информации об объекте, можно сделать заключение о целом. Таким образом, появляется возможность осуществлять оценки результатов процесса изменения природных и искусственных биологических систем.

Дробные дифференциальные операторы также нашли свое применение при моделировании динамики заболевания сахарным диабетом. В частности, применение фрактально-дробной производной Капуто позволяет позволяет формулировать математические модели с эффектами памяти [39]. Проведенные численные эксперименты при сравнении с реальными данными показали большую точность, нежели классический подход.

Фрактальные свойства также проявляются и на микроуровне организации живых существ, например, морфология различных клеток, пространственная организация мембран, распределение молекул клеточной адгезии. При моделировании компартментов – участков клетки, в которых происходят специфические биохимические процессы – в фармакинетическом анализе применение фрактальных скоростей при описании модели обеспечивает лучшее описание фармакокинетики без потери механистического описания лекарства. В

работе [40] показано, что использование фрактальной кинетики повысило производительность, а полученные данные полностью совпали с наблюдаемыми.

Методы фрактального анализа являются важным инструментом не только в моделировании природных объектов и систем, но также в области прогнозирования поведения курсов валют или ценных бумаг [8]. Финансовый рынок представляет собой сложную систему, на которую влияет множество факторов, что делает его достаточно хаотичным. Однако на рынке существуют и различные устойчивые механизмы, определяемые этими факторами [9]. Методы моделирования временных рядов, которые принято использовать для анализа и прогнозирования процессов, происходящих на финансовых рынках, часто дают неудовлетворительные результаты. Кроме того, можно отметить заметный разрыв между реальными экономическими данными и экономическими теориями. Для более точного прогнозирования и анализа финансового рынка, необходимо использовать более сложные методы, такие как методы фрактального анализа. Эти методы могут помочь выявить скрытые закономерности и особенности, которые не учитываются в стандартных моделях анализа временных рядов.

Рынки финансовых активов являются сложными системами, функционирование которых не всегда может быть правильно описано с помощью линейных моделей. Именно поэтому были созданы альтернативные подходы для исследования и моделирования финансовых рынков. В частности, современные исследователи рассматривают рынок как нелинейную динамическую систему, где все цены на активы зависят от своих предыдущих значений, а реакция на новую информацию не всегда происходит немедленно и симметрично. Одной из наиболее перспективных гипотез для анализа процесса ценообразования является идея о фрактальном рынке. В рамках этой гипотезы рынок рассматривается как самоподобная система, где поведение цен имеет фрактальную структуру и может быть описано с помощью математических моделей. Такой подход к исследованию финансовых рынков позволяет получать более точные прогнозы и предсказания,

что является особенно важным в условиях быстро меняющейся экономической ситуации [41].

Методы фрактального анализа временных рядов также активно применялись при прогнозировании распространения COVID-19 в небольших географических единицах [42]. В силу того, что кривые заболеваемости демонстрируют ежедневные значительные колебания, в них можно выделить фрактальные паттерны, которые в дальнейшем будут использованы для предсказательного моделирования.

Свойства фракталов позволяют использовать их для описания и моделирования сложных природных и техногенных явлений, а также социально-экономических систем. Долгое время фракталы представляли из себя лишь теоретические изыскания, пока с ростом производительности вычислительных систем не появилась возможность практического применения фрактальных алгоритмов. С каждым днем находятся все новые области приложения фракталов, которые требуют высокопроизводительных вычислений, что обосновывает актуальность данного исследования.

1.2 Вычислительная сложность фрактальных алгоритмов

Долгое время фрактальные алгоритмы не находили практического применения в силу их высокой вычислительной сложности, требующей больших мощностей от ЭВМ. Развитие компьютерной техники позволило ускорить работу с итерационными процессами и сделало возможным моделирование фракталов. Так множество Мандельброта впервые было описано аналитически в 1905 году Пьером Фату, но компьютерная визуализация была осуществлена только в 1978 [43].

Для определения порядка сложности вычислений фрактальных алгоритмов можно взять построение множества Мандельброта на координатной плоскости и сжатие изображений фрактальным способом.

Алгоритм фрактальной архивации изображений состоит из пяти основных пунктов [26]:

1. Кодируемое изображение разбивается на множество ранговых блоков $\{R_i\}$, которые не пересекаются между собой.

2. На ранговые блоки накладывается множество доменных блоков $\{D_i\}$, элементы которого могут пересекаться друг с другом.

3. Над доменными блоками выполняют аффинные преобразования [44]: сжатие до размеров рангового блока, изменение ориентации в восьми направлениях (повороты на 90°, 180°, 270°, 360°, а также выполнение ротации с отзеркаленным доменным блоком). После этого вычисляют значения коэффициентов преобразования доменного блока для его наилучшего соответствия ранговому:

$$w_{ij}\left(d_{xy}\right) = a_{ij} d_{xy} + b_{ij},$$

где a_{ij}, b_{ij} - коэффициенты СИФ, $d_{xy} \in D_{ij}$, D_{ij} - i-й доменный блок в j-й ориентации.

4. Для выявления соответствия доменного блока ранговому вычисляется расстояние между ними по формуле:

$$E(R,D) = \sum_{i=1}^{m}\sum_{j=1}^{m}\left(ud_{ij} + v - r_{ij}\right)^2,$$

где N_{R_i} - количество пикселей в ранговом блоке, $d_{xy} \in D_{ij}$, $r_{xy} \in R_i$.

5. Если для пары ранговый-доменный блок значение $E(R,D)$ не превышает заданной допустимой погрешности, то процедура перебора доменов для этого рангового блока останавливается, и начинается обработка следующего рангового блока. Если во время перебора не нашлось таких доменных блоков, для которых значение расстояния не превысило бы допустимой погрешности, то выполняется проверка максимального уровня разбиения ранговых блоков.

• Если ранговый блок не находится на максимальном уровне разбиения, то его разбивают на меньшие блоки и для них повторяют сравнение с доменными блоками.

• Если ранговый блок уже находится на максимальном уровне разбиения, то из всех доменных блоков выбирают тот, значение $E(R,D)$ которого является минимальным среди остальных, и считают рассматриваемый ранговый блок покрытым этим доменом.

Для поиска доменных блоков, наилучшим образом соответствующих ранговым, требуется полный перебор всех вариантов, что, в свою очередь, приводит к необходимости выполнить большое количество операций над массивами данных. Произведем оценку вычислительной сложности фрактального сжатия изображений относительно количества пикселей в изображении. Для удобства будем считать, что работаем с квадратным изображением со стороной N пикселей. При разбиении изображения получим M ранговых блоков и L доменных, при этом M>L, $M \sim N^2$, $L \sim N^2$. Выполняется сравнение каждого рангового блока со всеми доменными, при этом количество повторений цикла будет пропорционально $M \cdot L$, если не появится необходимость дальнейшего разбиения ранговых блоков. Исходя из выше сказанного, можно оценить порядок вычислительной сложности фрактального сжатия изображений как $O(N^4)$.

Множество Мандельброта является одним из самых известных фракталов, т.к. его визуализации часто приводят в качестве примера производительности вычислительной системы. Множество Мандельброта определяется как множество точек на комплексной плоскости, для которых рекуррентное соотношение $z_{n+1} = z_n^2 + c$ при $z_0 = 0$ задаёт ограниченную последовательность. То есть, это множество таких c, для которых существует такое действительное R, что неравенство $|z_n|<R$ выполняется при всех натуральных n. модуль z_n окажется больше 2, последовательность станет стремиться к бесконечности. В случае $|c|\leq 2$ это можно доказать с помощью метода математической индукции. При $|c|>2$ точка

c заведомо не принадлежит множеству Мандельброта, что также можно вывести методом индукции, используя равенство z_0=0 [45].

При визуализации множества Мандельброта можно получить лишь приближение к реальному множеству, т. к. количество итераций конечно. С увеличением максимального числа итераций результаты становятся все более точными, но также и возрастает время расчетов.

Ниже представлен алгоритм построения множества Мандельброта:

1. Выбирают точку *c* на комплексной плоскости.

2 Устанавливают $z_0 = 0$.

3 Вычисляют значение $z_{n+1} = z_n^2 + c$, и проверяют условие принадлежности точки множеству Мандельброта: если $z_{n+1}^2 > 4$, то точка является внешней.

4 Если условие из шага 2 не выполнилось заданное количество раз, то последовательность z_n сходится, и точка *c* принадлежит множеству Мандельброта.

5 Выбирают следующую точку *c* на комплексной плоскости и возвращаются в шаг 2.

Для построения множества Мандельброта следует перебрать все точки на комплексной плоскости в диапазоне $[-2;1]\times[-1;1]$. Поскольку цифровое изображение дискретно, то перебор точек будет осуществляться не бесконечно, а только в пределах заданных разрешением результирующего изображения. Увеличение числа итераций во время вычисления $z_{n+1} = z_n^2 + c$ приведет к более точному представлению визуализации множества Мандельброта. В предельном случае число итераций стремится к бесконечности. Если задать точность M для построения фрактала, то для генерации изображения размером $N \times N$ количество необходимых итераций будет пропорционально $N^2 \cdot M$.

Таким образом, вычислительная сложность фрактальных алгоритмов может варьироваться в зависимости от области приложения и реализуемой задачи, но остается достаточно высокой от $O(N^2)$ и выше.

1.3 Обзор вычислительных устройств, применяемых для реализации фрактальных алгоритмов

При работе с фрактальными алгоритмами часто возникает необходимость работы с большими массивами данных. Кроме того, большинство фрактальных алгоритмов используют рекурсивные и итерационные процедуры, поэтому заданная точность вычислений достигается за счет многократного повторения вычислительной части алгоритма. Эти два фактора приводят к необходимости использования больших вычислительных мощностей при работе с фрактальными алгоритмами.

В настоящее время существует множество реализаций фрактальных алгоритмов для вычислительных систем различных конфигураций, среди них можно выделить однопроцессорные и многопроцессорные устройства. Также можно классифицировать вычислительные системы по типу используемых устройств: универсальные процессоры, графические процессоры и программируемые логические интегральные схемы.

В начале проведем обзор вычислительных систем, базирующихся на универсальных процессорах (CPU). Одной из главных особенностей вычислителей данного типа считается эффективная работа с числами, представленными в формате с плавающей запятой, в частности современные процессоры отличаются оптимизированными реализациями операций над операндами длиной от 64 разрядов [46]. Пиковая производительность универсальных процессоров может достигать сотен Гфлопс. Например, четырехъядерные процессоры компании Intel серии Core i7–4470, обладающие тактовой частотой 3.9 ГГц и выполняющие до 16 операций над 64-разрядными числами с плавающей запятой за один такт на каждом из ядер, показывают при тестах производительность 249.6 Гфлопс [47]. Серверные процессоры линейки Intel Xeon E5 обладают увеличенным объемом кэш-памяти по сравнению с

другими линейками, поддерживают многопроцессорные системы большей производительности, имеют возможность работы с большим объемом памяти, поддерживают память с коррекцией ошибок. Процессоры данной линейки достигают пиковой производительности более тысячи Гфлопс.

В работах [11, 48] описывается, как были реализованы фрактальные алгоритмы на суперкомпьютере «МВС-100К», который построен на базе универсальных процессоров Intel Xeon (рис. 1.6).

Рис. 1.6 – Суперкомпьютер (СК) "МВС-100К"

Программные и аппаратные средства СК "МВС-100К" предоставляют множество возможностей для решения задач, не зависимо от их сложности. Они могут использоваться для решения одной задачи, при этом используя всю вычислительную мощность, а также для разделения решающего поля на несколько частей и предоставления их нескольким пользователям одновременно. Вычислительные модули, которые входят в состав СК "МВС-100К", базируются на серверах HP Proliant, что обеспечивает их высокую производительность. Основное решающее поле состоит из 1275 модулей, каждый из которых содержит 2550 четырехядерных процессора Intel Xeon (E5450, X5365, X5670 и X5675), работающих на частоте 3 ГГц.

Реализация фрактальных алгоритмов на вычислительных системах, построенных на базе универсальных процессоров, имеет ряд недостатков. В частности, это проявляется при параллельных реализациях фрактальных алгоритмов, которые требуют интенсивных взаимодействий типа процессор-память и процессор-процессор. В итоге большие накладные расходы снижают эффективность параллельного алгоритма, что негативно сказывается на общей производительности системы. К сожалению, с увеличением масштаба многопроцессорной системы данная проблема становится более актуальной [49]. Поэтому при выборе вычислительной системы для реализации фрактальных алгоритмов следует обращать внимание на возможности эффективно использовать ресурсы системы и минимизировать накладные расходы.

Помимо выше сказанного, можно отметить, что существует еще одна актуальная проблема - отставание производительности памяти относительно процессора, т.е. частота работы процессора во много раз превосходит частоту работы памяти. Например, процессор Intel Core i9-7900X может генерировать поток обращений в память до 528 Гбайт/с, тогда как пиковая полоса пропускания оперативной памяти DDR5 SDRAM составляет 51,2 Гбайт/с, что составляет менее 10% от возможного потока данных [50].

Проблема синхронизации рабочих процессов на МВС возникает довольно часто и уже хорошо известна. Она возникает в результате необходимости обмена данными между процессами, что приводит к простоям оборудования, если запрашиваемые данные не были отправлены процессами-источниками. Так в работе [49] показано, что время выполнения одного и того же алгоритма (фрактальное сжатие RGB-изображений) может отличаться более чем в 3 раза, разброс выполнения задачи на 64-х ядрах составил от 129 до 443 секунд. При учете специфики задачи возможно уменьшить рассинхронизацию между вычислительными узлами, но решение этой проблемы требует дополнительных ресурсов на ее реализацию, что может повысить стоимость проекта и затянуть сроки его выполнения.

В работах [51-54] описаны различные реализации фрактальных алгоритмов на графических процессорах (GPU). Графические ускорители сегодня являются неотъемлемой частью вычислительной технологии, их специализированная конвейерная архитектура позволяет эффективно обрабатывать и отображать компьютерную графику. GPU обладает высокой вычислительной мощностью, которая объясняется особенностями архитектуры. В отличие от обычного центрального процессора, графический процессор был создан как многопоточная структура с множеством ядер, что позволяет ему эффективно обрабатывать информацию. Это обуславливает и разницу в принципах работы: если архитектура CPU предполагает последовательную обработку информации, то GPU был исторически предназначен для обработки компьютерной графики и рассчитан на параллельные вычисления.

Графические процессоры фирмы AMD, использующие архитектуру RDNA, наследуют широко известную архитектуру GCN, имеющую ряд улучшений. Несмотря на то, что эта архитектура базируется на архитектуре GCN, она была изменена на базовом уровне и оптимизирована с использованием множества технологических решений, которые позволяют ей достичь более высокой эффективности и улучшить существующие слабые места GCN. RDNA была разработана с учетом гибкого масштабирования, поскольку эта архитектура будет использоваться в широком спектре устройств, включая следующее поколение игровых консолей, серверы Google Stadia и даже мобильные устройства Samsung с лицензированной графикой AMD. Кроме того, RDNA обладает новыми функциями, которые улучшают производительность, такие как улучшенный механизм масштабирования и новые функции распределения нагрузки. Так, видеокарта AMD Radeon VII фирмы AMD на базе архитектуры Vega второго поколения [55] с максимальной тактовой частотой 1,8 ГГц содержит 60 вычислительных блоков.

Видеокарта NVIDIA GeForce RTX 3080 Ti, внешний вид которой приведен на рис. 1.7, построена на чипах архитектуры Ampere [56]. Эта архитектура разработана NVIDIA как наследник архитектур Volta и Turing.

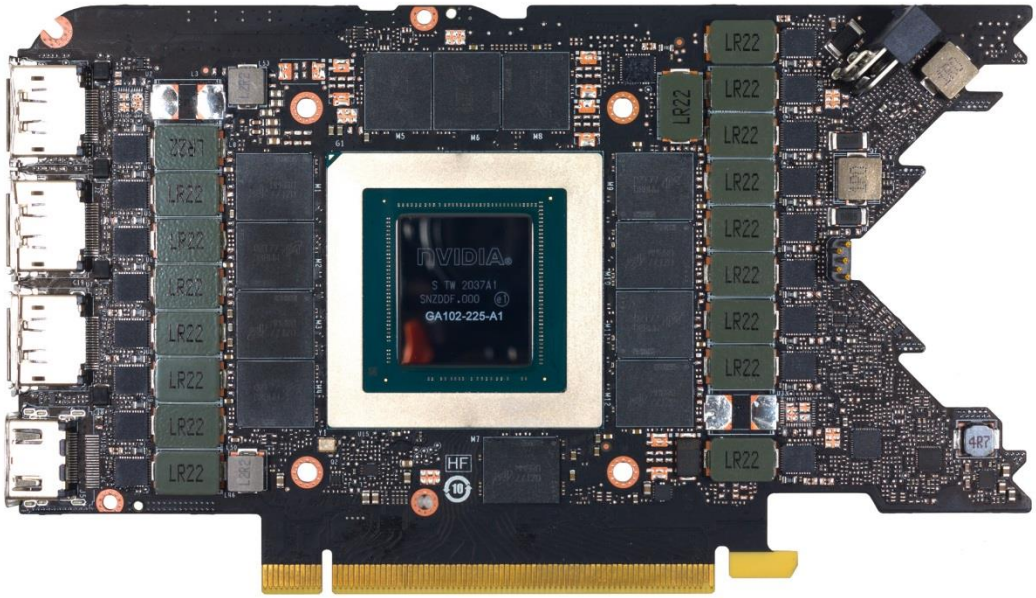


Рис. 1.7 – Внешний вид видеокарты NVIDIA GeForce RTX 3080 Ti

Графический процессор GA 102-225-A1, выполненный по технологии 8 нм, содержит 28,3 миллиардов транзисторов при площади кристалла 628 кв. мм. Также графический ускоритель включает в себя 80 потоковых мультипроцессоров (SM), 10240 шейдерных ALU FP32, 320 блоков наложения текстур (TMU), 112 блоков операции растеризации (ROP), 320 тензорных ядер и 80 RT-ядер. Тензорные ядра третьего поколения поддерживают работу с форматами чисел FP16, bfloat16, TensorFloat-32 (TF32) и FP64, а также обеспечивают ускорение при работе с разреженными данными.

Общий объем блока памяти GDDR6X, размещенного на видеокарте, составляет 12 Гбайт. Доступ к памяти организуется посредством 384-битного интерфейса. Тактовая частота работы графического процессора составляет 1365 МГц, а памяти – 1188 МГц.

Хотя кластеры из GPU позволяют получить высокую пиковую производительность, на практике она проявляется лишь при решении небольшого

процента задач. Например, при умножении плотных матриц, кластеры GPU способны достигать наиболее высокой производительности. Однако в случае задач с более сложной структурой данных, таких как обработка изображений или звуковых файлов, производительность кластеров GPU может быть ниже 5% от пиковой производительности.

Наиболее существенным физическим ограничением, которое необходимо учитывать при работе с GPU, является низкая пропускная способность каналов, связывающая графический процессор с источником данных, что существенно снижает производительность при решении сильно связанных задач, требующих обработки больших объемов данных. Наилучшую производительность GPU показывают при обработке массивов данных, полностью помещающихся в видеопамять, которая исчисляется десятками гигабайт. Т.е. при увеличении количества данных до сотен гигабайт, а также при необходимости обмена данными между вычислительными узлами скорость решения задачи существенно упадет. Важно помнить, что GPU является специализированным процессором, который требует для своей работы наличие управляющего узла, оснащенного не только CPU, но и оперативной памятью. Характеристики этих компонентов тоже оказывают значительное влияние на общую производительность системы [57]. Выбор компонентов для управляющего узлов ограничивается характеристиками GPU, и не всегда гарантирует высокую производительность всей системы.

В работе [14] проведено сравнение девяти реализаций фрактальных алгоритмов на различных вычислительных системах. Результаты сравнения приведены в Таблице 1.1.

Таблица 1.1. Сравнение реализаций фрактальных алгоритмов для CPU, GPU и ПЛИС

Публикация	Saad et al. [13]	Кулбаев [48]	Erra [58]	Samavi et al. [59]	Jackson et al. [60]	Panigrahy et al. [61]	Haque et al. [62]	da Rosa Righi et al. [63]	Ismail et al. [64]
Тип	ПЛИС	МВС	GPU	ПЛИС	ПЛИС	ПЛИС	GPU	CPU	GPU
Частота (МГц)	395	21300	300	240	32.05	75.52	1301	2900	700
Размер изображения	1024x1024	512x512	256x256	256x256	512x512	256x256	1024x1024	512x512	256x256
Тип перебора	Частичный	Полный	Полный	Частичный	-	Полный	Полный	Полный	Частичный
Размер рангового блока	Фикс. (8x8)	Фикс. (8x8)	Фикс. (4x4)	Фикс. (8x8)	Плавающий	Фикс. (4x4)	Фикс. x4)	Фикс. (8x8)	Фикс. (8x8)
Коэффициент сжатия	34,1	10	5	26	12.8	5.1	4.1	-	-
Время работы	10.8 мс	14.1 с	1 с	0.8 мс	8.4 мс	531 мс	45 с	8.5 с	455 мс

В работе [12] было проведено сравнение реализаций фрактального сжатия изображений для универсальных процессоров, графических ускорителей и ПЛИС. Показано, что один и тот же алгоритм при выполнении на ПЛИС показал в 2-3 раза большее быстродействие по сравнению с универсальным процессором и GPU.

Такие особенности многопроцессорных вычислительных систем, построенных на основе универсальных процессоров и графических ускорителей,

как рассинхронизация передачи обрабатываемых данных между процессорами, низкая пропускная способность каналов, связывающих источник данных и решающий узел, мешают росту реальной производительности при решении задач фрактального типа при увеличении аппаратных затрат.

Эту проблему можно успешно разрешить, используя устройства, где главным решателем являются микросхемы с программно-изменяемой вычислительной структурой [65]. Такие микросхемы позволяют конфигурировать структуру вычислительной системы, подстраивая ее наиболее оптимальным образом под алгоритм решаемой задачи. Для задач фрактального типа это означает возможность учитывать и использовать самоподобие структур данных для увеличения производительности системы и повышения точности решения. К таким микросхемам относятся ПЛИС [66, 67], которые представляют собой программно-конфигурируемое полупроводниковое устройство, базовая архитектура которого включает в себя:

– конфигурируемые логические блоки, известные как логические вентили (gates), реализующие требуемую логическую функцию (AND, NAND, OR, NOR, XOR, MULT_AND);

– программируемые электронные связи между логическими блоками;

– программируемые блоки ввода/вывода, обеспечивающие связи внутренней логики с внешними выводами микросхемы;

– дополнительные функциональные ресурсы (встроенные блоки памяти, умножители, схемы управления синхросигналами и др.).

Использование ПЛИС может значительно повысить реальную производительность многопроцессорных вычислительных систем при работе с задачами фрактального типа.

Основными производителями ПЛИС на данный момент являются фирмы Xilinx и Intel, которая в 2015 году приобрела компанию Altera, занимавшую до этого момента второе место на рынке производителей ПЛИС [47, 68]. Данные

компании выпускают разнообразные устройства на базе ПЛИС, применяемые в различных сферах.

В настоящее время производители ПЛИС предлагают широкий ассортимент изделий, имеющих различный вычислительный ресурс, наиболее известные среди них изделия фирм Xilinx и Intel. В 2015 году Intel приобрела компанию Altera, которая до этого занимала второе место на рынке производителей ПЛИС [47, 68]. Помимо ПЛИС Xilinx и Intel ускорители вычислительные платы, ориентированные на различное применение.

К самым передовым семействам ПЛИС, выпускаемым Xilinx, относятся Virtex и Kintex. Семейство Virtex UltraScale предназначено для создания систем высокой производительности, где требуется большой аппаратный ресурс, повышенные коммуникационные возможности и большой объем памяти. Состав семейства включает ПЛИС, содержащие от 626 тысяч до 4,432 миллионов логических блоков, объем встроенной памяти - от 44.3 до 115.2 мегабит, до 100 GTX приемо-передатчиков на частоте до 12.5Gb/s, до 6 IP ядер PCIe с поддержкой стандартов PCI Express Gen3 и Gen4. Пиковая производительность старшей ПЛИС семейства на частоте 700 МГц превышает $12*10^{12}$ операций с фиксированной запятой в секунду или до $2*10^{12}$ флопс (2 Тфлопс) при обработке данных в плавающем формате одинарной точности.

Семейство Kintex UltraScale рекомендуется для использования в задачах цифровой обработки сигналов, где требуется минимизировать показатель «цена/производительность» на один ватт потребляемой мощности. Эти ПЛИС содержат от 355 тысяч до 1,160 миллионов логических блоков и до 75.9 мегабит встроенной памяти. Кроме того, они обладают до 64 GTX приемо-передатчиков на частоте до 12.5Gb/s и до 32 GTY – на частоте до 16.3Gb/s, несколько IP ядер PCIe с поддержкой стандартов PCI Express Gen3. Пиковая производительность старшей ПЛИС семейства на частоте 700 МГц превышает 23*1012 операций с фиксированной запятой в секунду или до 3,8*1012 флопс (3,8 Тфлопс) при обработке данных в плавающем формате одинарной точности.

Важно отметить, что устройства, содержащие ПЛИС, могут использоваться в широком спектре сфер, включая медицину, телекоммуникации, аэрокосмическую и автомобильную промышленности. Также, благодаря высокой производительности, ПЛИС могут быть использованы для создания сложных систем искусственного интеллекта и глубокого обучения. Большой объем встроенной памяти обеспечивает быстрое и эффективное выполнение задач обработки данных, а возможность использования IP ядер PCIe способствует интеграции с другими устройствами и системами.

Фирма Intel выпускает ПЛИС семейств Stratix, Arria, Cyclone и Agilex.

ПЛИС семейства Arria 10 GX – это уникальные кристаллы, обладающие высокой производительностью и множеством функций. В зависимости от конкретной модели, они содержат от 160 тысяч до 1,150 миллионов логических блоков, что позволяет выполнять сложные задачи в самых разных областях - от авиации до медицины. ПЛИС Arria также имеют до 64 мегабит встроенной памяти, что обеспечивает быстрое чтение и запись данных. Кроме того, они могут быть оснащены до 96 трансиверов, работающих на частоте до 17,4 Gb/s, что позволяет передавать большие объемы информации в режиме реального времени. Они являются единственными в мире кристаллами, у которых DSP блоки могут выполнять операции с плавающей запятой одинарной точности. Это означает, что пиковая производительность на операциях с фиксированной и плавающей запятой одинакова и достигает 4,7 Тфлопс на кристалл. Таким образом, ПЛИС Arria могут быть использованы в самых разных областях, где требуется высокая вычислительная мощность и точность. В состав ПЛИС Arria 10 SX SoC включен аппаратный двухъядерный процессор ARM Cortex-A9 MPCore, функционирующий на частоте до 1,5 ГГц. Это позволяет обрабатывать большие объемы данных и выполнять сложные вычисления прямо на кристалле, что повышает эффективность работы системы в целом.

Семейство Stratix 10 обладает еще более впечатляющими характеристиками Например, эти ПЛИС выпускаются по технологии 14 нанометров, что

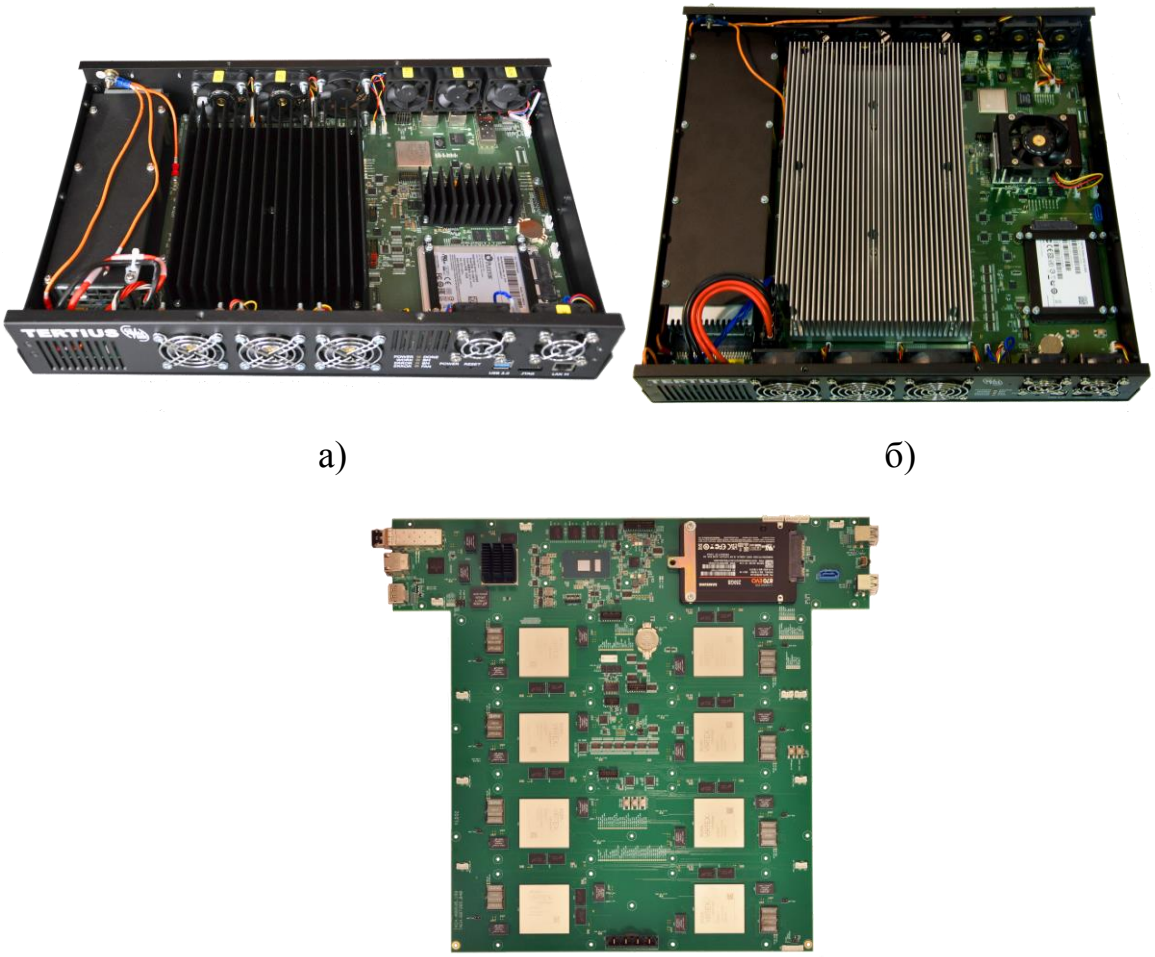
обеспечивает высокую эффективность и производительность. Кроме того, они содержат до 4 миллионов логических блоков (LEs), что позволяет обрабатывать большие объемы данных. Приемо-передатчики в семействе Stratix 10 обладают скоростью передачи до 28 Гбит/с для обмена между платами и 32 Гбит/с для обменов между микросхемами внутри платы. Это значительно повышает скорость передачи данных, что особенно важно для больших и сложных проектов. Пиковая производительность на операциях с фиксированной и плавающей запятой одинарной точности в семействе Stratix 10 составляет до 10 Тфлопс на кристалл. Это означает, что эти ПЛИС могут обрабатывать огромные объемы данных с высокой точностью и скоростью. Таким образом, семейство Stratix 10 представляет собой мощное и эффективное решение для различных задач, которые требуют высокой производительности и скорости обработки данных.

В настоящее время в Российской Федерации существует несколько организаций, которые занимаются созданием реконфигурируемых вычислительных систем на основе ПЛИС: ООО «НИЦ СЭ и НК» [69], НИИ МВС ЮФУ [70], НПО «Роста» [71], ФГУП «НИИ «Квант» [72].

В Научно-исследовательском центре супер-ЭВМ и нейрокомпьютеров (ООО «НИЦ СЭ и НК», г. Таганрог) занимаются разработкой реконфигурируемых вычислительных систем суперкомпьютерной производительности на базе ПЛИС семейств Virtex и Kintex фирмы Xilinx и серийно выпускают их. Эти системы содержат до нескольких сотен ПЛИС, объединенных в единое поле посредством скоростных каналов передачи данных – LVDS и Rocket GTX. Такой подход обеспечивает высокую производительность, возрастающую с увеличением задействованного аппаратного ресурса. РВС на основе больших полей ПЛИС являются развитием концепции многопроцессорных вычислительных систем с программируемой архитектурой, предложенной академиком РАН А.В. Каляевым [73]. Разработка таких систем позволяет решать сложные задачи, требующие большой вычислительной мощности, в различных отраслях науки.

В настоящее время ООО «НИЦ СЭ и НК» [69] осуществляет выпуск РВС на ПЛИС нового поколения фирмы Xilinx серии UltraScale, выполненных по технологии 20нм и обладающих повышенным быстродействием в сравнении с ПЛИС Xilinx 7-й серии.

Примером данных РВС являются реконфигурируемые компьютеры семейства Терциус (рис. 1.8).



а) б)

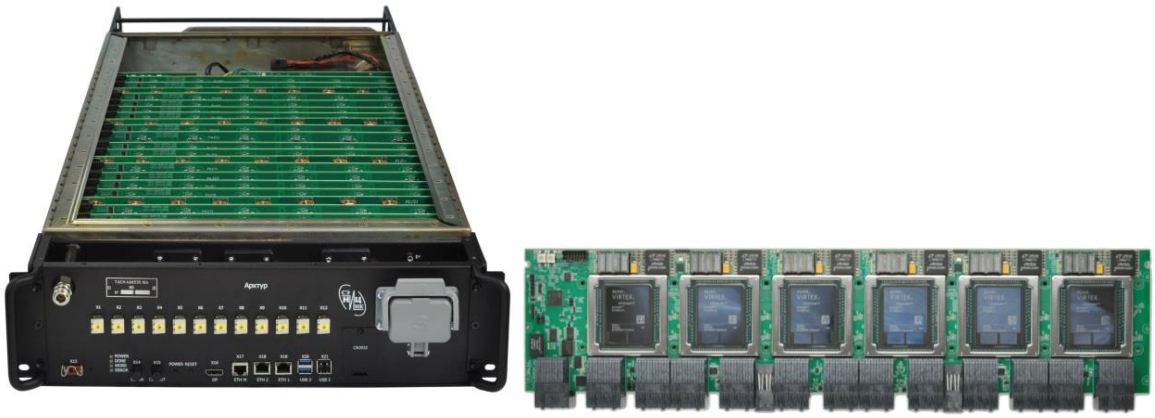
в)

Рис. 1.8 – Вычислительные блоки на основе ПЛИС Kintex UltraScale: а) Терциус; б) Терциус-2 в) плата Терциус-3

Вычислительный блок Терциус обладает уникальной конфигурацией, состоящей из 4 ПЛИС Kintex UltraScale XCKU095, что обеспечивает производительность вычислительного поля в 2,5 Тфлопс. Реконфигурируемый компьютер Терциус-2 содержит 8 ПЛИС Kintex UltraScale XCKU095 и его

производительность составляет 5 Тфлопс. Тактовая частота работы обоих блоков составляет 550 МГц. Чтобы обеспечить быстрый обмен данными, все ПЛИС внутри блока объединяются LVDS-каналами со скоростью передачи 1,25 Гбит/с, а также высокоскоростными GTY/GTH каналами со скоростью передачи вплоть до 12,5 Гбит/с. Скорость обмена данными по каналу Ethernet составляет 1 Гбит/с.

Перспективной разработкой НИЦ «СЭ и НК» для решения сильносвязанных задач является реконфигурируемый вычислительный блок «Арктур» (рис 1.9) [74]. Он представляет собой функционально законченное изделие с 12-ю вычислительными модулями (ВМ) на основе ПЛИС XCVU35P (или XCVU45P). В качестве основных связей между ПЛИС предполагается использование дифференциальных линий с подключенными к ним мульти-гигабитными трансиверами (MGT); вспомогательными связями являются дифференциальные линии, подключенные к HP-банкам.



а) б)

Рис.1.9 – а) вычислительный блок Арктур, б) вычислительный модуль Арктур

В вычислительном модуле Актур ПЛИС связаны «горизонтальными» и «вертикальными» линиями связей. «Горизонтальные» связи между ПЛИС представлены 28-ю дифференциальными парами со скоростью передачи данных до 25 Гбит/с по одной паре, что обеспечивает пропускную способность до 700 Гбит/с независимо в каждом направлении.

Также имеется канал, связывающий по кольцу первую и последнюю ПЛИС в ВМ с помощью 12 дифференциальных пар и со скоростью до 300 Гбит/с независимо в каждом направлении. Дополнительно имеются 48 дифференциальных линий с максимальной скоростью до 1,2 Гбит/с, каждая из которых может работать в любом направлении.

К «вертикальным» связям относятся 192 дифференциальные пары с допустимой скоростью передачи данных до 20 Гбит/с по одной паре, что обеспечивает пропускную способность до 3840 Гбит/с независимо в каждом направлении.

Кроме того, между платами существуют связи, предназначенные для служебных целей, таких как загрузка конфигурации ПЛИС, передача команд и данных от процессора к любому вычислительному модулю. Пропускная способность данного канала – 64 Гбит/с для обоих направлений независимо.

Общая пропускная способность каналов связи ВМ – 13 Тбит/с, в том числе между ВМ – 7,68 Тбит/с.

Разработанные в ООО «НИЦ СЭ и НК» и НИИ МВС ЮФУ реконфигурируемые вычислительные системы [75, 76] обладают высокой реальной производительностью. Это стало возможным благодаря созданию адекватной вычислительной структуры информационного графа для каждой решаемой задачи при использовании структурно-процедурной организации вычислений.

В настоящее время становится все более популярной реализация фрактальных алгоритмов на программируемых логических интегральных схемах (ПЛИС) [77, 78]. Особенности устройств на базе ПЛИС позволяют добиться более высокой производительности при решении задач фрактального типа по сравнению с системами на универсальных процессорах и графических процессорах. В частности, структура ПЛИС позволяет эффективно использовать распараллеливание операций на уровне логических блоков. Такой подход обеспечивает реальную производительность в сотни Гфлопс на одном кристалле

ПЛИС [79]. Например, в работе [12] показано, что при реализации фрактального сжатия изображений на Altera Stratix IV 530 и Altera Stratix V 5SGXA7, Intel Xeon W3690, NVIDIA Fermi C2075 устройства на базе ПЛИС показывают скорость обработки в три раза большую, чем GPU и в 114 раз большую, чем CPU. Это свидетельствует о том, что ПЛИС позволяют значительно ускорить процессы обработки данных и являются более эффективным решением в данной области.

ПЛИС нашли широкое применение благодаря своим уникальным свойствам, которые позволяют использовать их как вспомогательные ускорители, так и в качестве основных вычислительных узлов. Они используются во многих областях, включая встраиваемую технику и персональные компьютеры.

Например, фирма Nallatech выпускает многоцелевые FPGA-акселераторы на базе кристаллов Xilinx и Intel для повышения производительности встраиваемой техники и персональных компьютеров. Эти устройства могут использоваться в качестве вспомогательных ускорителей, и их эффективность уже доказана в работе [12], где была успешно реализована фрактальный алгоритм на ускорителе Nallatech PCIE 385n (рис. 1.10), который основан на кристалле Stratix V 5SGXA7.

Рис. 1.10 – Ускоритель Nallatech PCIE 385n

Продукция фирмы Xilinx охватывает различные типы устройств, которые широко применяются для реализации разнообразных фрактальных алгоритмов.

Например, в работах [12, 77, 78] приведены описания высокопроизводительных реализаций фрактального сжатия изображений для ускорителей на базе ПЛИС семейств Virtex и Spartan (рис. 1.11). Эти решения обеспечивают значительное ускорение вычислений и позволяют экономить время на обработке изображений.

а)

б)

Рис. 1.11 – Ускорители на базе ПЛИС семейств Virtex и Spartan а) ускоритель XUPV5 Development Board (ML509), б) ускоритель Spartan-6 FPGA Embedded Kit

Использование ПЛИС позволяет разработчику оптимизировать задачи и создавать свои методы распараллеливания для реализации высокопроизводительных фрактальных алгоритмов. Благодаря применению ПЛИС, можно ускорить процесс обработки данных на порядки, сделав вычисления более эффективными. При этом задействование нескольких ПЛИС в вычислительной системе дает возможность увеличивать производительность системы пропорционально количеству кристаллов. Таким образом, использование ПЛИС является наилучшим инструментом для реализации сложных алгоритмов в вычислительных системах.

1.4 Обзор современных реализаций фрактальных алгоритмов на ПЛИС

В работах [13, 14] рассмотрена реализация стандартного алгоритма фрактального сжатия изображений с полным перебором на ПЛИС Altera Cyclone II 2C70. Изображение разбивается на равные ранговые блоки. В каждую ячейку памяти помещается по два таких блока. Такой подход позволяет увеличить скорость обработки изображения. Процесс разбиения изображения и его укладки в память устройства продемонстрирован на рисунке 1.12.

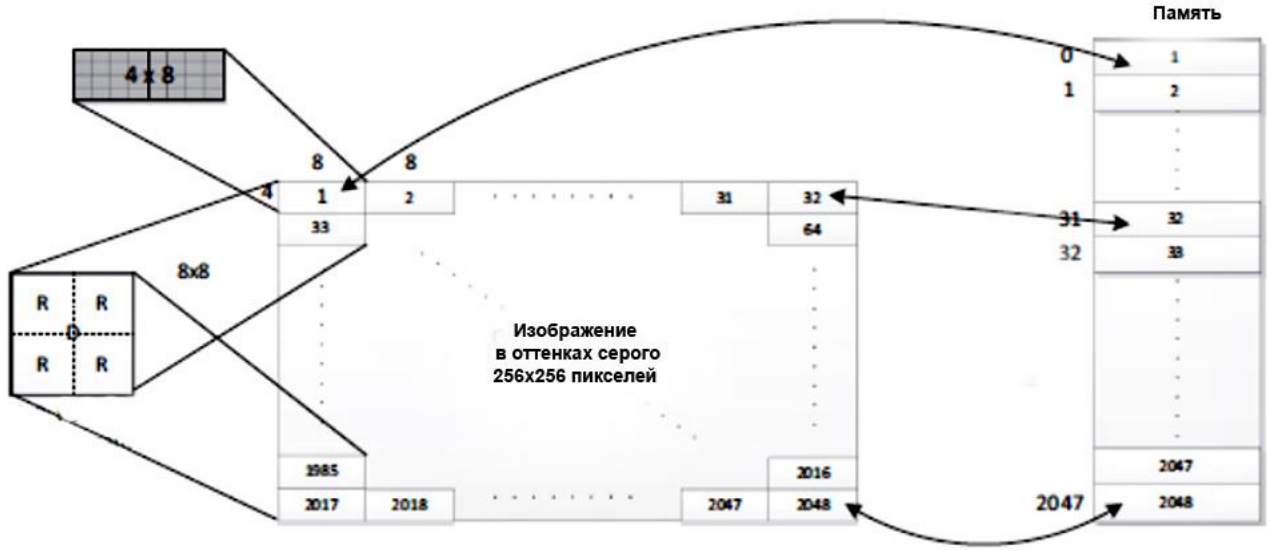


Рис.1.12 – Представление изображения в памяти устройства

Общая работа системы описывается диаграммой конечного автомата, показанной на рисунке 1.13. Данная диаграмма демонстрирует работу системы по сжатию данных, которая включает в себя несколько этапов. Первый этап - это состояние бездействия, обозначенное S0. Затем, при получении сигнала начала, система переходит в состояние S1, где начинается извлечение ранговых блоков. После этого система переходит в состояние S2, где происходит дополнительная обработка ранговых блоков. Далее система переходит в состояние S3, где начинается извлечение доменных блоков, а затем в состояние S4, где происходит выполнение сопоставления. Для перехода между описанными выше состояниями система использует сигналы «Следующий домен», «Следующий ранг» и «Готово». Когда сигнал сброса становится неактивным, система ожидает сигнала начала, чтобы начать операцию сжатия, передав четыре блока диапазона из

памяти и сохраняя их в регистрах. Таким образом, общая работа системы включает в себя несколько этапов, и каждый этап выполняется при помощи определенных состояний и сигналов, что обеспечивает эффективность и точность сжатия данных.

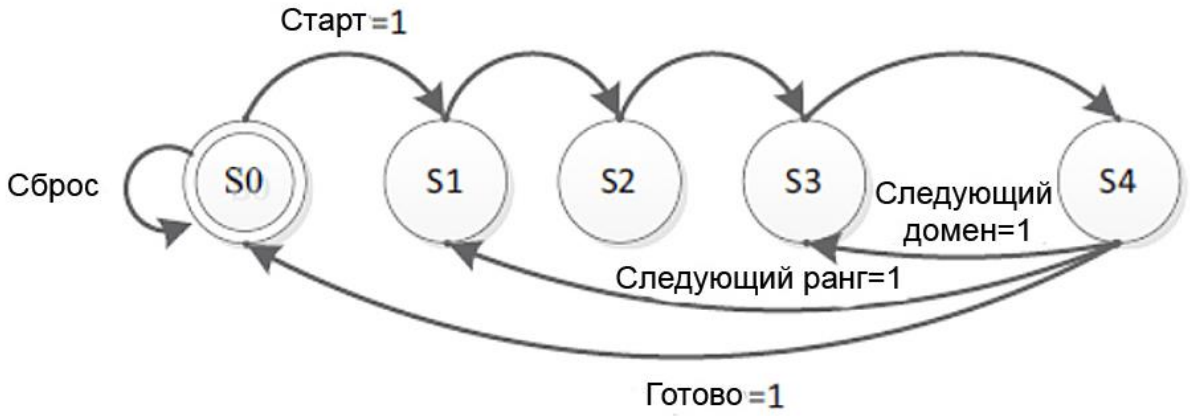


Рис.1.13 – Конечный автомат общей работы системы

На рисунке 1.14 показана параллельная архитектура решения. Ранговые блоки загружаются в регистры RangeBlockXX за один такт. Доменный блок преобразуется до размера рангового и после этого также загружается в регистр. Выполняется параллельное сравнение сжатого доменного блока с четырьмя предзагруженными ранговыми блоками. После этого результаты сравниваются с минимально допустимыми и сохраняются в случае успешного сравнения вместе с номером текущего доменного блока. Если следующий доменный блок покажет лучший результат, то значение будет перезаписано. После перебора всех доменных блоков загружаются следующие четыре ранговых блока и процесс повторяется.

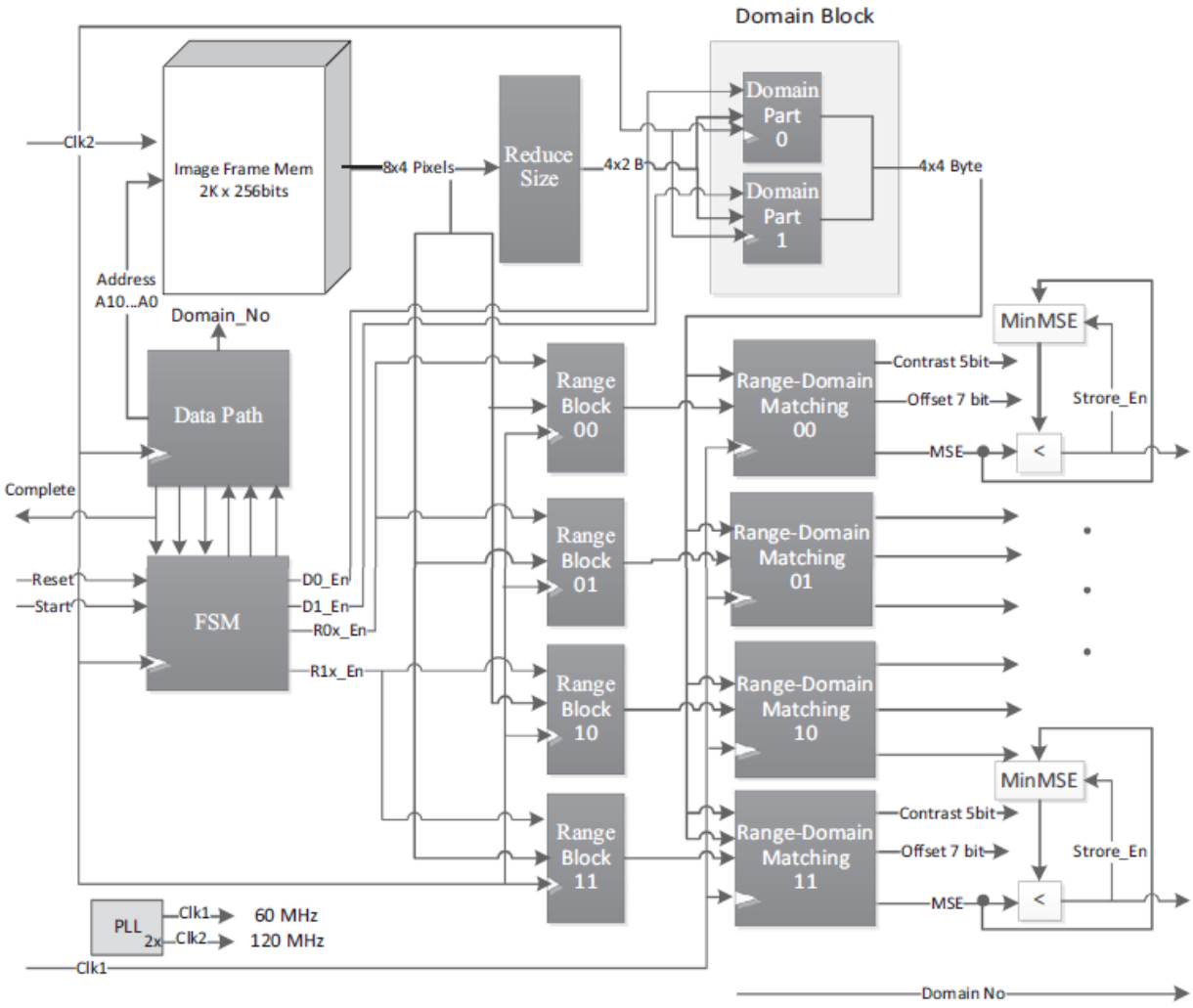


Рис.1.14 – Параллельная архитектура фрактального сжатия изображений

Неоспоримым достоинством рассмотренного метода является то, что весь алгоритм полностью реализован на ПЛИС, а не использует их в качестве ускорителей. Такой подход позволяет полностью использовать возможности ПЛИС, что увеличивает производительность за счет уменьшения обменов между ней и процессором управляющей машины. Однако серьезным недостатком данной реализации является отсутствие возможности масштабирования параллельной архитектуры фрактального алгоритма на несколько ПЛИС для увеличения производительности. Если необходимо обработать изображение высокого качества, которое невозможно загрузить во внутреннюю память одной ПЛИС, то для сохранения той же скорости обработки данных следует разбить

выполнение задачи на несколько кристаллов. Однако это не предусмотрено в рассматриваемой реализации, что ограничивает возможности использования данного метода в более широком спектре задач. Разделение задачи на несколько ускорителей приведет к проблеме обмена данными между ними. Низкая скорость каналов обмена данными нивелирует большую часть выигрыша по скорости решения задачи.

В работе [80] также описана однокристальная реализация фрактального сжатия изображений, для которой использовалась плата Xilinx Spartan 6 (xc6slx45t-3fgg484). В данном случае использовалось адаптивное разбиение изображения методом квадродерева, как показано на рисунке 1.15.

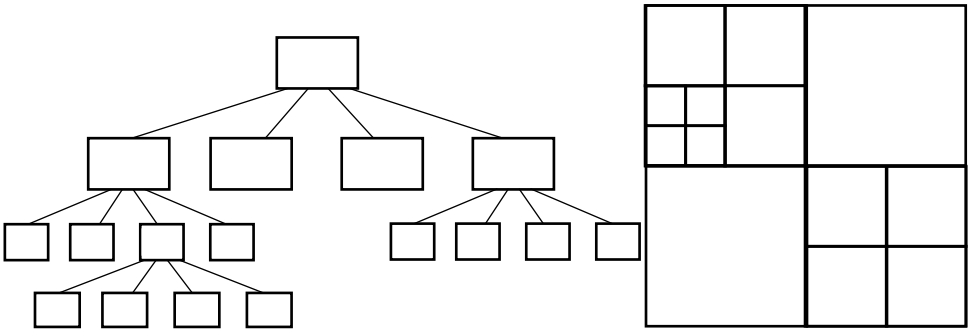


Рис.1.15 – Адаптивное разбиение квадродеревом

Первоначально исходное изображение представляется в виде массива пикселей, который сохраняется в 64 x 8 блоках оперативной памяти, где каждый блок представляет собой небольшую часть изображения. Затем эти блоки вычитываются в однопортовом блоке ОЗУ. После этого полученное изображение делится на более мелкие блоки с использованием фрактального разложения по квадродереву с определенным пороговым значением. Это позволяет получить средние значения, значения координат x и y, номер размера блока определяется в результате разложения квадродерева. Фаза кодирования изображения завершается чтением и записью коэффициентов сжатия фрактального изображения. Предлагаемая архитектура фрактального сжатия изображения с квадродеревом показана на рисунке 1.16 ниже.

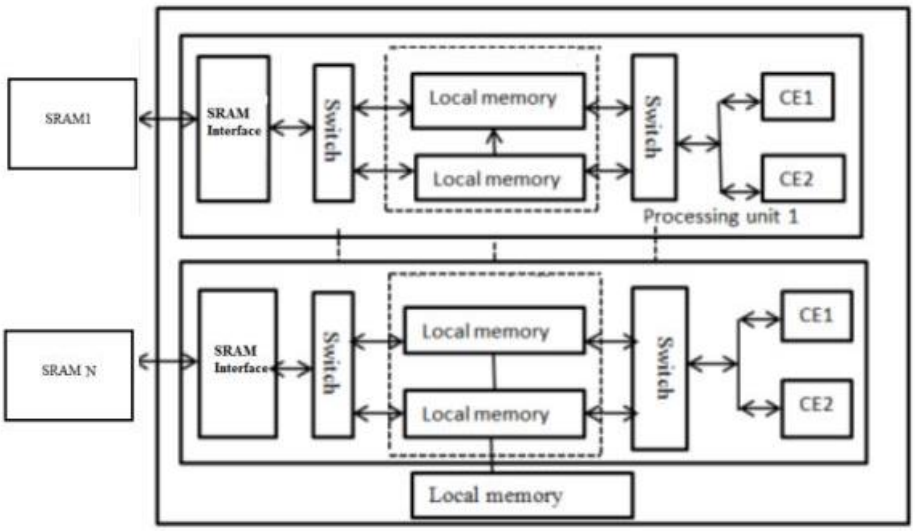


Рис.1.16 – Архитектура фрактального алгоритма с адаптивным разбиением

Достоинством данной реализации является адаптивное разбиение изображения на более мелкие фрагменты, что помогает сэкономить вычислительный ресурс и получить качественный результат. Но при этом требуется более внимательный подход к каждому набору входных данных для поиска оптимального разбиения каждого отдельного изображения. Это, в свою очередь, приводит к невозможности прогнозирования времени работы алгоритма при одинаковом объеме, но различном характере входных данных. Также недостатком данной реализации является отсутствие возможности масштабирования решения на несколько ПЛИС, что может быть критично для задач, требующих обработки больших объемов данных.

В работе [81] предложено улучшение алгоритма шифрования AES для кодирования изображений с помощью фрактальных методов. Это позволило добиться лучшей непредсказуемости и более широкого динамического поведения результата шифрования. Реализация данного модифицированного алгоритма AES была выполнена на ПЛИС Altera Cyclone IV, архитектура решения строится на основе восьмикрылого хаотического осциллятора (рис. 1.17).

Схема содержит 6 D-триггеров FF0 - FF5, делитель/умножитель чисел с фиксированной точкой MD0, сумматор чисел с фиксированной точкой ADD0, мультиплексоры M0 - M7 и блок статической памяти FSM0.

FF0 – FF4 образуют блок памяти, использующийся для хранения промежуточных значений переменных и состояний автомата. MD0 и ADD0 используются для проведения операций для фиксированной запятой. FF5 и FSM0 составляют конечный автомат, который генерирует последовательности управляющих сигналов. В блоке FSM0 осуществляется изменение состояний автомата в зависимости от значения стартового сигнала BT. Регистр FF5 отправляет значение текущего состояния автомата на блок FSM0 и получает значение следующего состояния автомата каждый такт синхронизирующего сигнала. Мультиплексоры M0 - M7 позволяют управляющим сигналам из FSM0 динамически изменять путь потока данных, переключая его между регистрами и операционными блоками для выполнения специфических операций. Схема представляет собой детерминированный конечный автомат.

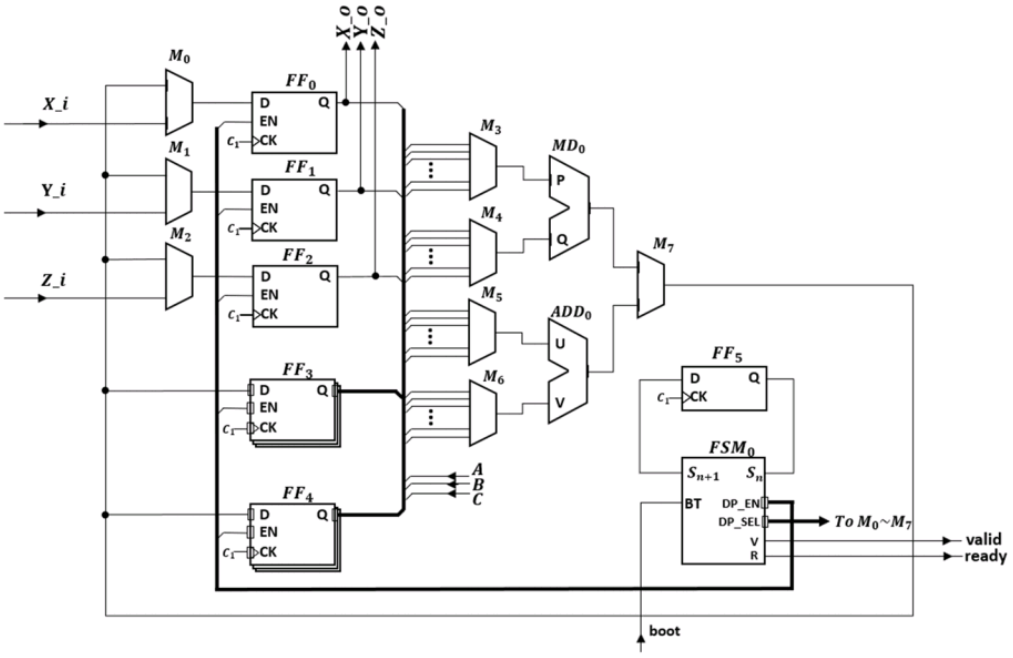


Рисунок 1.17 Архитектура AES с фрактальными модификациями

Достоинствами данной реализации являются низкое потребление энергии и использование небольшого вычислительного ресурса в сравнении с другими реализациями данного алгоритма на ПЛИС [82-84]. Однако полностью отсутствует возможность масштабирования этой архитектуры не только на

несколько ПЛИС, но и на устройства, обладающие большим вычислительным ресурсом, чем рассмотренное в статье.

Все описанные выше реализации фрактальных алгоритмов как для традиционных вычислительных систем, так и для ускорителей на основе ПЛИС, не подразумевают возможность масштабирования алгоритмов без их существенной доработки. Также увеличение количества вычислительных узлов приведет к увеличению потока пересылаемых данных между ними, что влечет за собой уменьшение реальной производительности. Таким образом, доказано первое положение, выносимое на защиту.

1.5 Представление самоподобных структур в традиционных вычислительных системах

Фракталы применяются в различных отраслях техники и при решении разнообразных задач, поэтому существует множество способов работы с данными в конкретных случаях, и многие из них зависят от особенностей задачи и используемых инструментов. Кроме традиционного способа представления фракталов в виде массива в прямоугольном виде, существуют и другие подходы, которые могут быть более эффективными в определенных задачах. Например, для решения задач, связанных с анализом изображений, часто используются методы, основанные на статистической обработке пикселей. Эти методы позволяют извлекать изображения из фракталов с большой точностью и эффективностью, что делает их полезными в таких областях, как медицинская диагностика и научные исследования. Кроме того, существуют и методы, которые позволяют работать с фракталами как с непрерывными объектами, не требуя их дискретизации. Эти методы могут быть особенно полезны в задачах, связанных с анализом физических процессов и моделированием сложных систем. Несмотря на то, что традиционный способ представления фракталов в виде массива имеет свои недостатки, существуют множество других подходов, которые могут быть более

эффективными в различных задачах. Важно выбирать подход, который наилучшим образом соответствует конкретной задаче, чтобы получить наилучший результат.

Один из способов представления фракталов в памяти устройств – это хранение системы итерируемых функций (СИФ) и ее коэффициентов, кодирующих фрактальное множество. СИФ представляет собой систему функций из некоторого фиксированного класса функций, отображающих одно многомерное множество на другое. Общий вид системы таких функций представлен формулой (1.1):

$$\{f_i : X \to X \mid i = 1, 2, \dots, N\}, \qquad N \in \mathbb{N} \qquad (1.1)$$

В общем случае f_i могут быть и нелинейными функциями, но чаще всего используются системы линейных итерируемых функций, которые по сути являются аффинными преобразованиями [26]. В том случае для хранения массива данных используется не сам массив, а набор функций и коэффициентов. В результате выполнения операций над этим набором можно получить исходный массив. Этот метод позволяет существенно сократить объем необходимой памяти, но требует специализированного распаковщика для работы с данными. Использование данного подхода может быть целесообразным для хранения фракталов в выходных файлах. Данный метод может быть использован для обработки данных в вычислительной системе, при условии наличия соответствующего распаковщика, что может быть реализовано в качестве отдельного модуля.

Подобный способ хранения фракталов находит применение в компьютерной графике, при реализации сложных неевклидовых объектов, образы которых похожи на природные, например, облаков, гор, поверхности моря и т.д., т.к. по сути своей является алгоритмом построение объекта с заданными параметрами. Кроме того, такой подход может быть использован для уменьшения объема данных, используемых в обработке изображений, что позволяет сократить затраты на компьютерные ресурсы и ускорить процесс обработки. Подобную

упаковку также можно применять в случаях, когда необходимо передавать изображения по сети с низкой пропускной способностью, так как она позволяет существенно сократить объем передаваемых данных. Для получения такой упаковки используются фрактальные алгоритмы сжатия, при которых исходная информация кодируется в виде СИФ, что обеспечивает высокую эффективность сжатия при сохранении качества изображения.

При обработке фракталов часто используются древовидные структуры данных. Деревья – это структуры данных, которые состоят из узлов. Деревья являются связными графами, не содержащими циклы. Узлы могут быть разных типов, в зависимости от их функциональной роли в дереве. Некоторые узлы могут содержать данные, а другие – ссылки на узлы в дереве. Таким образом, при обходе дерева можно получить доступ к различным данным, которые хранятся в узлах. Каждый узел в дереве имеет свой тип. Например, листья – это узлы, которые не имеют потомков. Они являются конечными узлами в структуре дерева. Внутренние узлы, с другой стороны, связывают листья между собой и могут содержать адреса других внутренних узлов. Пример структуры дерева можно увидеть на рисунке 1.18.

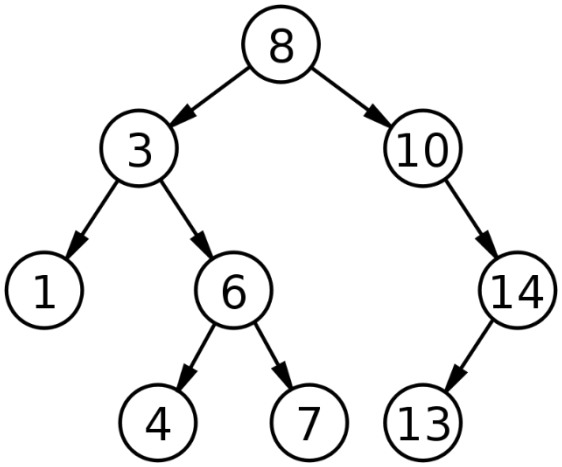


Рис. 1.18 – Двоичное трехуровневое дерево

Существует множество разных типов деревьев, каждый из которых специализирован для определенных задач. Некоторые из них, например, двоичное дерево, префиксное дерево, расширяющееся дерево и танцующее дерево, могут быть использованы в разных областях программирования. Деревья, в которых

число потомков узла кратно степени двойки, такие как двоичные, четверичные или восьмеричные, часто применяются для работы с фракталами. Это связано с тем, что фракталы, как правило, имеют рекурсивную структуру, и деревья хорошо подходят для работы с такими структурами.

Двоичные деревья – это тип структур данных, которые имеют не более двух потомков у каждого узла. Они используются в различных областях, в том числе в работе с фрактальными структурами, которые могут представлять всевозможные ломаные линии. Например, такие деревья могут быть применены для функций, описывающих процессы, происходящие на финансовых рынках. В то же время двоичные деревья могут быть использованы для многих других целей, таких как хранение и поиск данных, сортировка и многое другое. С их помощью можно эффективно решать различные задачи и оптимизировать работу системы в целом.

В задачах фрактального типа часто используются четверичные деревья, которые также называют деревьями квадрантов или квадродеревьями. В отличие от двоичных деревьев у четверичных каждый внутренний узел имеет либо ровно четыре потомка, либо не имеет вовсе. Четверичные деревья используются в различных областях, например, в графических редакторах для хранения изображений, в компьютерной графике для рендеринга изображений и в криптографии для хранения и обработки криптографических ключей.

С помощью четверичных деревьев рекурсивно разбивают двухмерное пространства по четыре области в общем случае произвольной формы, чем и заслужили особую популярность при реализации фрактального сжатия и анализа фрагментов фрактальных изображений на различных вычислительных системах [78, 85-87].

Узел четверичного дерева содержит следующую информацию:

- четыре указателя на потомков узла;

- точка, содержащая координаты x и y и данные.

Пример структуры четверичного дерева, разбивающего область размером 8x8 пикселей, приведён на рисунке 1.19.

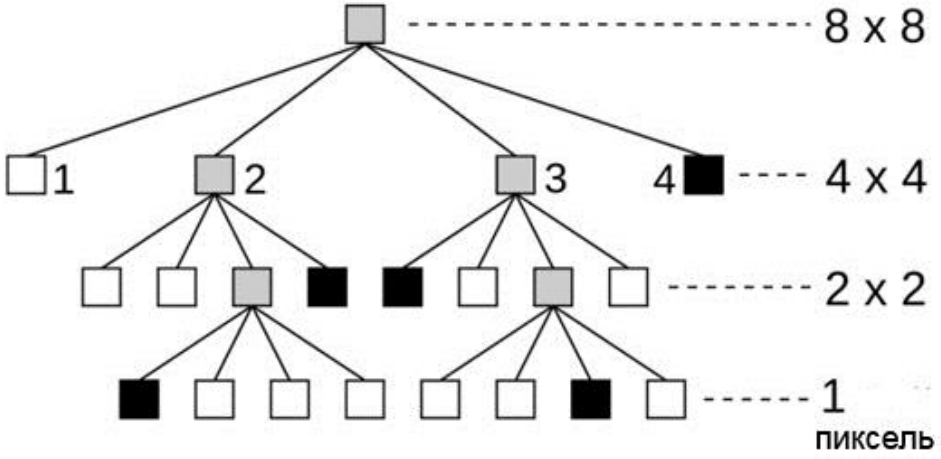


Рис. 1.19 – Четверичное дерево

Аналогично четверичным восьмеричные деревья являются структурами данных, каждый внутренний узел которой имеет ровно восемь дочерних элементов. Эти деревья широко используются для разделения трехмерного пространства. Принцип их работы заключается в рекурсивном разделении пространства на восемь равных ячеек, как показано на рисунке 1.20, что позволяет эффективно организовать хранение и поиск трехмерных объектов.

Каждый узел восьмеричного дерева представляет собой региональную точку и хранит явную трёхмерную координату, которая является центром разделения пространства для данного узла. Помимо этого, данная точка хранит один из углов дочерних пространств, что необходимо для последующего быстрого поиска объектов внутри дерева.

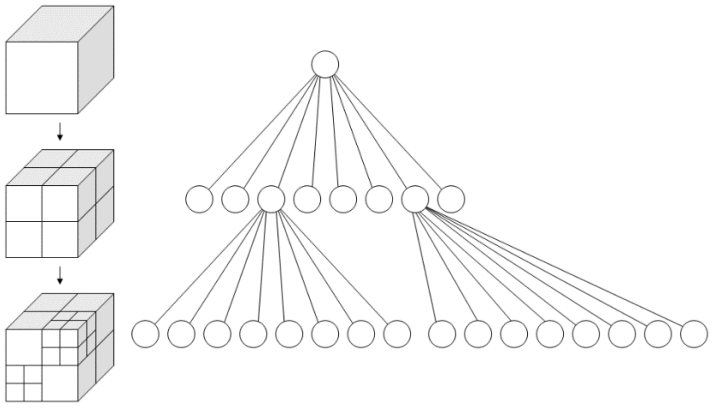


Рис. 1.20 – Разделение трехмерного пространства и соответствующее восьмеричное дерево

Восьмеричные деревья широко применяются в компьютерной графике, компьютерном зрении, а также в других областях, связанных с обработкой трехмерных данных, например, при построении подробных трехмерных изображений для компьютерных игр, моделировании столкновений физических объектов, воксельном представлении 3D-графики.

При работе с древовидными структурами данных возникают определенные трудности при распараллеливании вычислений. Динамическое представление таких структур данных в памяти устройства при распределении между вычислительными узлами многопроцессорной вычислительной системы приводит к неравномерной нагрузке на параллельно выполняемые процессы. Чтобы решить эту проблему и более равномерно распределить данные, перед их раздачей рекомендуется выполнить обход дерева, чтобы определить глубину каждой ветви. Данная операция требует дополнительных временных затрат, что снижает реальную производительность вычислительной системы.

Решением этой проблемы является линеаризация древовидных структур посредством списков с пропусками. Список с пропусками является вероятностной структурой данных, состоящей из нескольких отсортированных связных списков, распределенных по уровням. На самом нижнем (первом) уровне располагаются все элементы. Дальше около половины элементов в таком же порядке располагаются на втором, почти четверть — на третьем и так далее, но при этом известно, что если элемент расположен на уровне *i*, то он также расположен на всех уровнях, номера которых меньше *i* (рис. 1.21). Это позволяет осуществлять быстрый поиск элементов в списке и ускоряет процесс работы с данными.

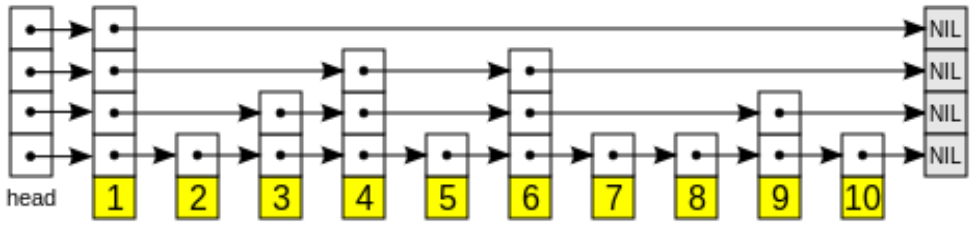


Рис. 1.21 – Структура списка с пропусками

Список с пропусками позволяет представить древовидную структуру в виде линейного массива (рис. 1.22), где за каждым элементом следуют его потомки, но сохраняются связи между одноуровневыми точками, что улучшает возможность обращения к конкретной части фрактала [87].

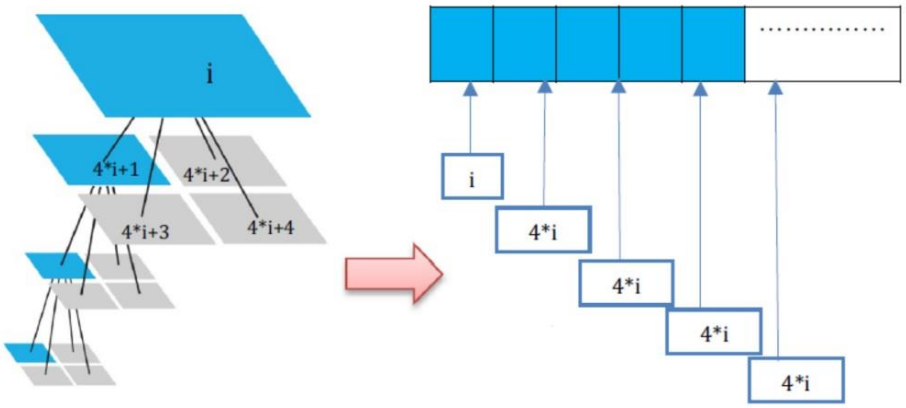


Рис. 1.22 – Соответствие дерева списку с пропусками

Данный способ требует динамического выделения оперативной памяти, что может привести к ряду проблем. Во-первых, помимо основных данных, которые необходимо хранить, нужно учитывать сопутствующую информацию, такую как адреса памяти и другие метаданные. Во-вторых, необходимость своевременного освобождения памяти приводит к наложению дополнительных ограничений на разбиваемую область, что может затруднять работу с данными. Кроме того, проблема фрагментации памяти может привести к невозможности использования всего потенциала оперативной памяти, т.к. при мелких разбиениях отдельные данные все равно будут занимать целые блоки памяти.

Из выше сказанного можно сделать вывод, что существующие способы представления данных для традиционных вычислительных систем хотя и позволяют обрабатывать самоподобные структуры, но в некоторых случаях могут приводить к чрезмерному расходу памяти устройства, требуют предварительной подготовки данных перед работой и высокой связности памяти.

1.6 Существующие способы представления данных на реконфигурируемых вычислительных системах

Ранее ПЛИС рассматривались в вычислительных системах лишь как вспомогательное оборудование, выполняющее рутинные задачи. Они увеличивали производительность основного процессора, но их использование было ограничено. Однако в настоящее время ПЛИС можно эффективно использовать во многих областях благодаря увеличению степени интеграции и быстродействия этих устройств. Например, они могут применяться в сетевых коммуникациях, биомедицинской технике, промышленных автоматических системах и автомобильной промышленности. В сфере науки и исследований ПЛИС также нашли свое применение, например, в многопроцессорных системах и в машинном обучении. Продолжающийся рост эффективности этих устройств может привести к еще большему расширению их области применения в будущем. При решении задач, связанных с обработкой больших потоков данных, ПЛИС показывают высокую эффективность благодаря своим свойствам. Более того, при необходимости обновления оборудования и его настройки для лучшего соответствия решаемой задаче важную роль играет способность ПЛИС к реконфигурации в рабочих условиях в сжатые сроки. В целом, можно сказать, что ПЛИС сегодня являются незаменимым инструментом во многих областях, где необходимо эффективно обрабатывать данные и решать сложные задачи.

В значительной степени свойства вычислительной системы определяются тем, в каком виде в ней представлены данные. Процесс представления данных в вычислительной системе является одним из ключевых, так как он определяет многие свойства системы. Различные типы адресации, которые используются при хранении данных в памяти, могут существенно влиять на ее производительность и эффективность. В большинстве современных реконфигурируемых вычислительных систем (РВС) данные представляются в форме прямоугольных

массивов, отличающихся друг от друга различными типами адресации, применяемыми при хранении их в памяти.

В настоящее время существует четыре способа представления массивов данных, применяющихся на РВС, - линейная адресация памяти линейная адресация памяти, косвенная адресация, множества и пакеты [88-90]. Особенности работы с РВС дают возможность применять те способы адресации, которые следуют из специфики решаемой задачи.

Линейная адресация памяти. При этом способе организации памяти адрес для каждого данного формируется из адресов вычислительного блока, блока памяти и адреса ячейки памяти, где содержится данное.

На рисунке 1.23 показан пример формирования адреса, первые N старших битов указывают на адрес вычислительного блока, затем идут K битов, определяющие адрес блока памяти, и оставшиеся L битов указывают на ячейку памяти, содержащую данное. Таким образом, длина адреса данного составит $N+K+L$ бит, а максимальный объём данных, который может быть представлен в данном адресном пространстве, равен 2^{N*K*L}.

Внутренняя память современных ПЛИС фирмы Xilinx поддерживает разбиение на страницы, содержащие более восьми миллионов слов, с длиной слова до 4608 бит.

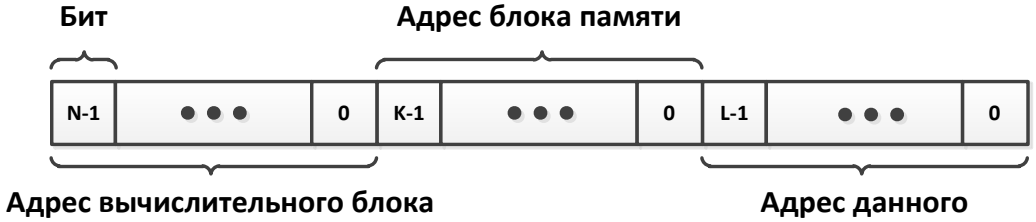


Рис. 1.23 – Пример формирования адреса при линейной адресации памяти

Данный тип адресации применяется в задачах, требующих обработки больших объёмов данных, которые могут быть последовательно обработаны, например, в задачах цифровой обработки сигналов или при обработке изображений.

Также стоит отметить, что линейная адресация памяти используется в широком спектре задач адаптивной фильтрации, в том числе при вычислении автокорреляционных функций. В таких случаях, данные, над которыми осуществляется данная процедура, последовательно записываются в память, а затем вычитываются по двум каналам. Первый канал осуществляет последовательное вычитывание данных, которое происходит N раз, где N равно количеству элементов в наборе. Второй канал вычитывает набор данных только один раз, но каждое значение вычитывается раз в N тактов.

Косвенная адресация памяти. Этот тип организации памяти отличается тем, что данные и их адреса находятся в разных блока памяти ПЛИС, т.е. блок RAM1 содержит адреса данных, которые участвуют в процессе вычислений, а в блоке RAM2 находятся искомые данные (рис. 1.24). Перед доступом к требуемому данному пользователь должен отправить запрос в блок RAM, содержащий адрес искомого данного, и после его получения обратиться в блок RAM, где находится данное. Адрес данного может храниться в блоках памяти в линейной форме.

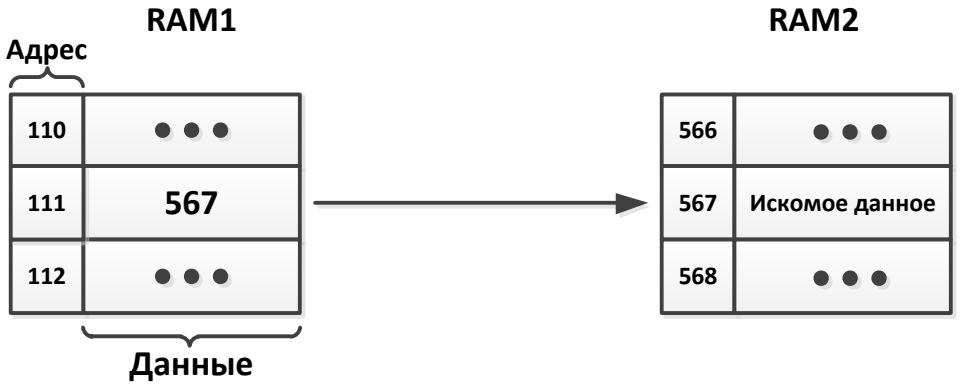


Рис. 1.24 – Организация косвенной адресации памяти

В качестве примера применения данного типа адресации можно привести использование заранее загруженных в блоки памяти таблиц значений для нелинейных функций – логарифмических, тригонометрических, интегральных и других. Данные функции трудоемки в вычислениях, и для оптимизации работы РВС их значения, вычисленные заранее с требуемой погрешностью, сохраняют

ПЗУ так, чтобы входной адрес соответствовал нужному аргументу функции. При подаче аргумента на вход ПЗУ он выдаст соответствующее значение функции, что делает его использование более эффективным и быстрым, нежели непосредственное вычисление каждого значения заданной функции.

Множества. Данный способ организации данных на РВС является самым новым, он был предложен А.В. Касаркиным в 2020 году [90]. Множество представлено в памяти в виде двух массивов. В первом хранятся значимые данные, принимающие участие в вычислениях. Второй массив содержит столько же элементов, что и первый, но является битовым, где каждый бит поставлен в соответствие конкретному элементу из первого массива. По своей сути битовый массив является массивом признаков, устанавливающим является ли элемент первого массива частью множества (рис. 1.25). В зависимости от размера множества оно может занимать разное количество блоков RAM.

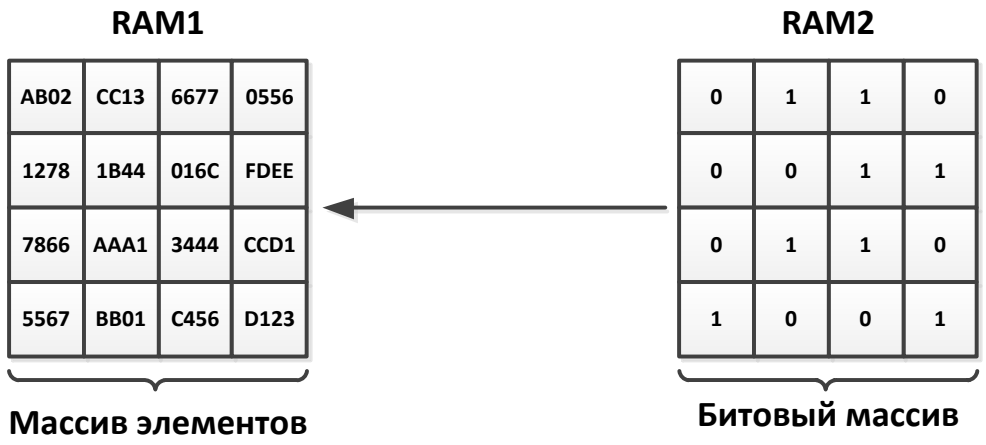


Рис. 1.25 – Способ хранения множества в памяти

Такой способ представления данных применяется, например, в задаче поиска всех максимальных клик графа [89, 91]. Эта задача часто встречается в химии, логистике, социологии и т.д. В неориентированном графе клика – это подмножество вершин, каждые две из которых соединены ребром графа. Максимальной кликой принято называть клику, которая не включена ни в какую другую большего размера. Кроме того, можно отметить, что задача поиска максимальных клик графа имеет важное практическое значение, так как может

быть использована, например, для оптимизации работы социальных сетей, ускорения поиска в больших базах данных, а также для решения задач анализа данных в различных областях науки и техники.

Пакеты. Данный способ представления данных применяется для передачи данных между отдельными ПЛИС или между вычислительными блоками внутри ПЛИС. Пакет – это специально оформленный блок данных, состоящий из двух частей**:** управляющей информации и полезных данных (рис. 1.26). Управляющая информация содержит данные, необходимые для доставки пакета, такие как адреса отправителя и получателя, коды обнаружения ошибок, например, контрольные суммы, и информация об очередности. В заголовке пакета содержится информация о размере пакета, типе данных, адресе отправителя и получателя и других параметрах. В хвосте пакета содержится информация о контрольной сумме и других кодах обнаружения ошибок. Все эти данные необходимы для правильной обработки и доставки пакета. Между заголовком и хвостом пакета размещаются полезные данные. Эти данные могут быть любого типа и формата, в зависимости от того, какую информацию нужно передать.

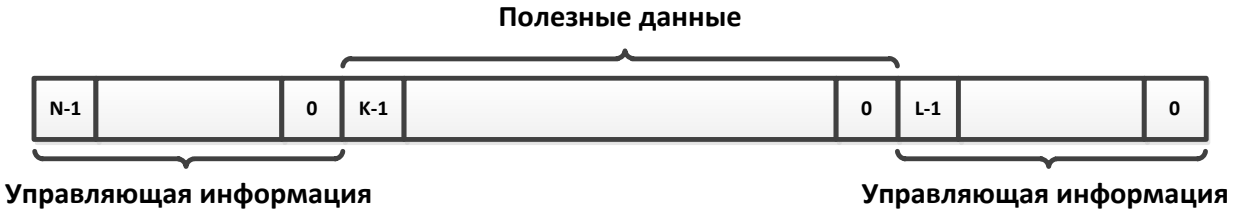


Рис. 1.26 – Структура пакета данных

На практике пакетная организация данных в РВС применяется в различных сценариях, таких как обмен информацией с различными сетевыми устройствами или передача данных между разными ПЛИС., но очень редко для передачи данных внутри одной ПЛИС, т.к. вероятность повреждения данных в этом случае ничтожно мала. Кроме того, пакетная организация данных позволяет улучшить скорость передачи, снизить нагрузку на канал связи и повысить производительность вычислительной системы в целом. Таким образом, используя

пакетную организацию данных, можно гарантировать более эффективный и безопасный обмен информацией между устройствами. Например, в радиолокационных системах (РЛС) пакеты данных передаются между ПЛИС по каналу Ethernet или посредством LVDS. Результат обработки входящей информации в каждой ПЛИС преобразуется в форму пакета и передаётся в следующую ПЛИС, где этот пакет дополняется данными из неё, и так далее до тех пор, пока не будет пройдена вся РВС [88].

На данный момент не существуют эффективных способов представления фрактальных структур для РВС. Это создает проблему при работе с фрактальными алгоритмами, которые могут иметь множество практических применений. В связи с этим, необходимо создать специально оптимизированные для РВС структуры данных, которые могли бы обеспечить удобство и эффективность работы. Среди описанных выше структур данных множества наиболее близки к фрактальным задачам, но и они не обеспечат достаточного удобства и эффективности. Разнообразные ветвящиеся структуры данных требуют больших затрат памяти устройства, а специфика распределения блоков памяти в ПЛИС может привести к тому, что данные могут быть разделены внутри кристалла, и потребуются дополнительные затраты на их синхронизацию. Таким образом, для эффективной работы с различными фрактальными алгоритмами возникает необходимость создания специально оптимизированных для РВС структур данных.

1.7 Принципы организации эффективной обработки фрактальных структур на РВС

Для синтеза параллельно-конвейерных программ для РВС с высокой реальной производительностью, предназначенных для работы с самоподобными структурами, предлагается подход, позволяющий создавать сбалансированную

вычислительную структуру задачи, обеспечивающую минимальный простой и высокий коэффициент использования оборудования.

Специфика работы с фрактальными структурами связана с обработкой больших массивов данных, где для вычислений необходимо осуществлять многократные обмены между процессором и источником данных. В этом случае уровень реальной производительности вычислительных систем, заточенных на выполнение мультипроцедурных параллельных вычислений, снижается [8, 92]. Альтернативой является структурный подход организации вычислений [73].

Для структурной реализации вычислительного процесса характерно распараллеливание по операционным вершинам информационного графа задачи. При этом вершины соответствуют множеству функциональных устройств, соединенных между собой информационными связями для обмена данными. При структурном подходе к организации вычислений отсутствуют такие, присущие мультипроцедурному подходу недостатки, как потеря времени за счет синхронизации процессов, потеря времени на передачу данных через промежуточные процессы, очереди при обращении к данным, хранящимся в памяти. При структурной организации вычислений все процессы выполняются с одинаковым быстродействием, включая пересылки данных и обращения к памяти.

Структурный подход более предпочтителен при решении задач, связанных с фрактальной обработкой данных, поскольку возможность привести в соответствие вычислительную структуру и информационный граф задачи позволяет эффективно организовать вычисления и избежать потерь, связанных с обменом данными. Таким образом, можно сформулировать первый принцип организации эффективной работы с самоподобными структурами на РВС: *для работы с самоподобными структурами необходимо использовать структурный подход организации вычислений.*

Минимальным ресурсом для структурной организации вычислений является аппаратная реализация базового подграфа, которому соответствуют вычисления в узле сетки. При работе с самоподобными структурами между такими узлами

происходит обмен данными. При этом итерационно вершины графа связаны между собой, а внутри одного слоя информационные связи между вершинами отсутствуют, что дает возможность эффективно распараллелить вычислительную структуру по слоям (рис. 1.27).

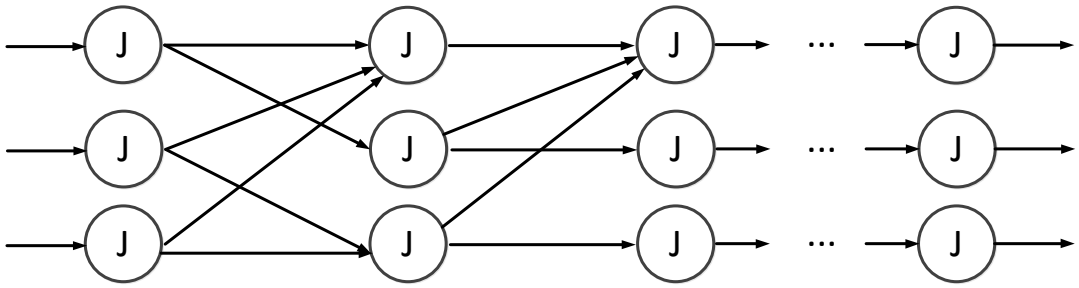


Рис. 1.27 – Распараллеливание по слоям

Количество каналов подачи данных в современных РВС является критическим ресурсом, тогда как потенциал вычислительного ресурса несоизмеримо выше. Например, реконфигурируемый компьютер Терциус-2, содержит 8 ПЛИС Kintex UltraScale XCKU095, каждая из которых обладает ресурсом в 1176000 LUT, но при этом для работы с каждой из ПЛИС доступно только по 2 контроллера распределенной памяти с шириной канала 32 бита. Для максимального задействования вычислительного ресурса, а значит получения более высокой реальной производительности, следует производить распараллеливание вычислений не только по слоям, но и по итерациям (рис. 1.28).

В пределах каждой итерации для вычислительных вершин характерна последовательная информационная зависимость, где каждый следующий узел будет иметь информационные связи с предыдущими.

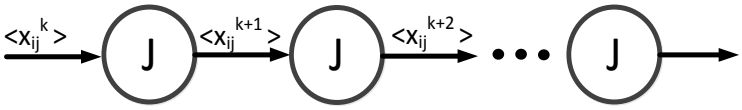
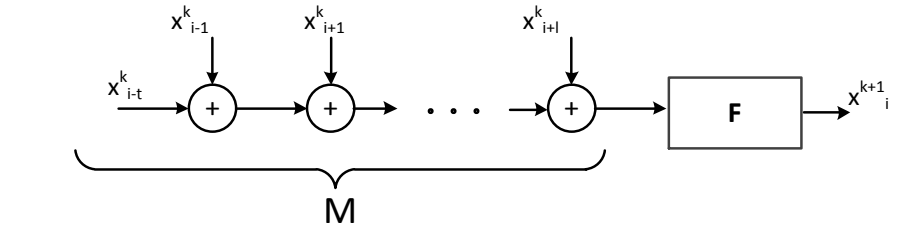


Рис. 1.28 – Распараллеливание по итерациям

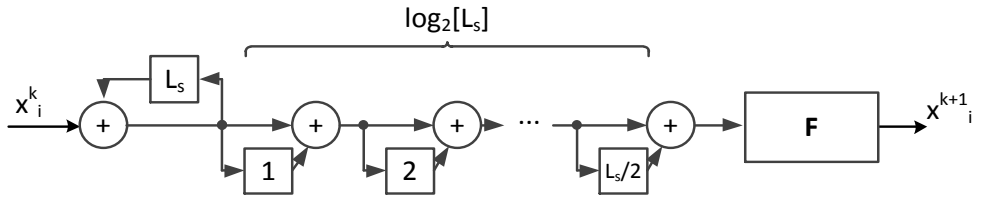
При неограниченном количестве каналов для подачи данных в вычислительную структуру можно было бы получить очень высокие показатели производительности, если бы была возможность задействовать при

распараллеливании по слоям весь доступный вычислительный ресурс. Однако в реальных РВС распараллеливание только по слоям позволит задействовать лишь небольшой процент вычислительного ресурса, что приведет к низкой удельной производительности системы. Для повышения реальной производительности и увеличения эффективности алгоритма следует использовать распараллеливание по итерациям, т.е. выстраивать вычислительные блоки в форме конвейера, где каждый следующий блок принимает поток данных, выходящий из предыдущего. Таким образом, следующий принцип эффективной обработки самоподобных структур на РВС формулируется так: *при реализации данного класса задач следует отдавать предпочтение распараллеливанию по итерациям, а не по слоям.*

При решении фрактальных задач возникает необходимость обработки самоподобных структур, что в свою очередь приводит к работе с системами итерируемых функций и их коэффициентами, упакованными в массивы, отличные от прямоугольных. Для удобства масштабирования вычислительной структуры классические методы работы с потоками данных подразумевают объединение в подкадры изоморфных структур, при этом в зависимости от специфики задачи за основу берется либо максимальная вычислительная структура (рис.1.29 а), либо при нехватке вычислительного ресурса – минимальная (рис. 1.29 б).



а)



б)

Рис. 1.29 – Максимальная (а) и минимальная (б) вычислительные структуры

Использование максимального подграфа приведет к простою оборудования, т.к. особенности работы с самоподобными структурами таковы, что необходимость обработки маленьких порций данных возникает значительно чаще, чем больших. При использовании второго варианта снижается скорость выполнения вычислений. При работе с самоподобными структурами для обеспечения загрузки всех входов вычислительной структуры пришлось бы создать отдельные структуры для всех возможных вариантов подачи данных, что не представляется возможным, и помешает эффективному масштабированию вычислительной структуры. В связи с этим необходимо создать постоянные вычислительные структуры, оставив переменными только потоки данных, т.е. перейти от изоморфных структур к подобным (рис. 1.30).

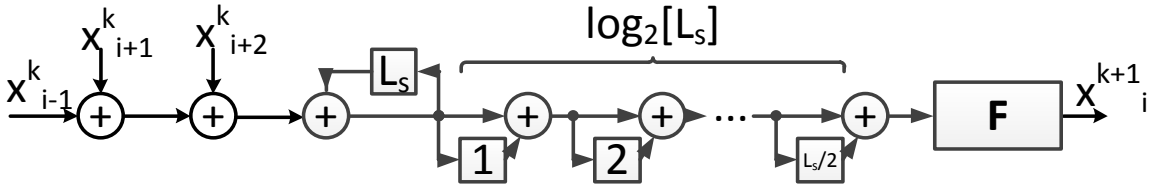


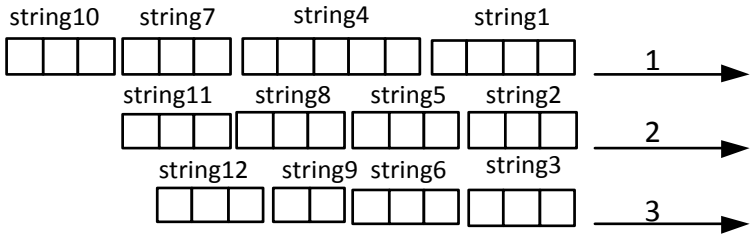
Рис. 1.30 – Вычислительная структура для неравномерного потока данных

Такой подход обеспечивает возможность масштабирования вычислительных структур, сокращая излишние затраты оборудования и не снижая скорость вычислений. Отсюда следует третий принцип эффективной обработки самоподобных структур на РВС: *для масштабирования вычислений необходимо объединять в вычислительные структуры подобные подграфы, а не изоморфные.*

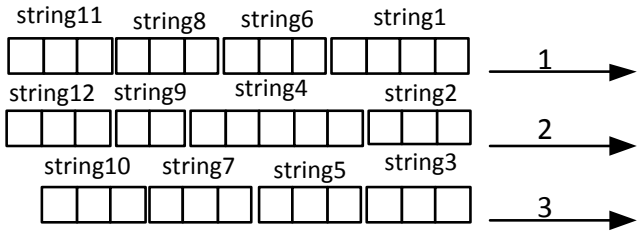
Ключевой задачей при поиске оптимальной структуры является определение необходимого количества каналов для подачи данных, связанного с параметрами входного потока и влияющего на затраты оборудования. Эффективность реализации алгоритма оценивается как отношение числа полезных вычислительных блоков к общему их количеству, которое включает в себя вспомогательные вычислительные блоки, например, дополнительные

сумматоры, уменьшающие скважность потока данных. Поэтому при определении параметров вычислительной структуры следует также руководствоваться принципом максимизации коэффициента использования оборудования. При повышении этого коэффициента увеличивается полезная нагрузка на оборудование, что при увеличении вычислительного ресурса ведет к росту производительности, близкому к линейному.

Одним из факторов, влияющих на скорость обработки самоподобных структур, является неравномерное распределение данных в массивах, что может приводить к простою из-за различной загруженности вычислительных конвейеров (рис. 1.31), что приводит к уменьшению реальной производительности вычислительной системы.



а)



б)

Рис. 1.31 – Поступление данных на вычислительные конвейеры: (а) неотсортированные данные, (б) отсортированные данные

Для уменьшения простоя и более равномерной загрузки конвейеров предлагается выполнять предварительную подготовку входных данных до их загрузки в РВС, т.е. распределять данные между вычислительными слоями не в естественном порядке, а таким образом, чтобы учитывалось не только количество подаваемых порций данных, но и их длина. Такая сортировка выполняется один

раз, не требует коррекции на этапе вычислений, и обеспечивает минимальное время прохождения потоков данных. Таким образом, можно сформулировать четвертый эффективной обработки самоподобных структур на РВС: *для увеличения эффективности алгоритма необходимо учитывать особенности входных данных и выполнять предварительную их сортировку.*

Работа с самоподобными структурами производится с помощью рекурсивных процедур и итерационных процессов, при этом блоки в вычислительной структуры становятся охвачены обратной связью и неизбежно возникает проблема скважности потока данных. Для начала обработки следующего пакета данных необходимо дождаться окончания обработки предыдущего – и в случае, если латентность вычислительного блока больше единицы, поток данных необходимо будет задержать до этого момента. В результате снижается быстродействие вычислительной структуры, т.к. уменьшается коэффициент использования оборудования, поскольку часть времени схема простаивает, не выполняя полезных вычислений. Одним из способов раскрытия обратных связей является применение схемы «конвейер в конвейере». Данная схема предполагает параллельную обработку независимых данных, которые последовательно подаются на вход вычислительной структуры. Суть метода заключается в следующем. Пусть имеется вычислительный блок, охваченный обратной связью и обладающий латентностью L. В случае если на вход вычислительного блока подаётся непрерывный поток данных, мы можем для сохранения корректности вычислений вместо увеличения скважности одного потока до L, подавать на вход схемы L потоков данных со скважностью 1. Таким образом, за L тактов мы обработаем L независимых потоков данных. На рисунке 1.32 показаны схемы и диаграммы работы исходного вычислительного блока с обратной связью (а) и конвейера в конвейере (б).

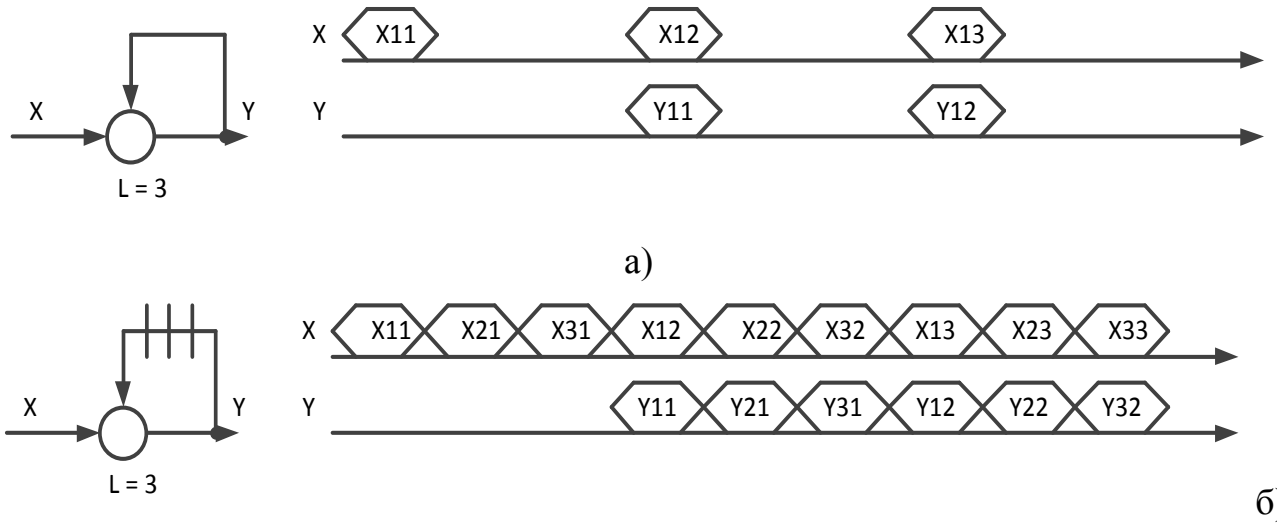


Рис. 1.32 – Схема и диаграмма работы вычислительного блока, охваченного обратной связью (а) и вычислительного блока, распараллеленного по схеме «конвейер в конвейере» (б)

Другим способом ускорения вычислений в вычислительных структурах, реализующих рекурсивные процедуры и итерационные процессы, является автоподстановка. Суть метода автоподстановки заключается в следующем. Пусть имеется некоторый итерационный процесс, описываемый уравнением вида:

$$x_{k+1} = ax_k + b. \quad (1.2)$$

В этом случае, не нарушая общности рассуждений, мы можем выразить x_k через x_{k-1}.

$$x_k = ax_{k-1} + b \quad (1.3)$$

Теперь подставим выражение (1.3) в выражение (1.2):

$$x_{k+1} = a(ax_{k-1} + b) + b.$$

Продолжая подстановку выражений для x_{k-2}, x_{k-3} и т.д., мы можем получить формулу для выражения x_{k+1} из любого существующего x_{k-r}:

$$x_{k+1} = a^{r-1}x_{k-r} + c \quad (1.4).$$

где

$$c = b(a^{r-2} + a^{r-3} + \ldots + 1).$$

В случае если выражение (1.2) реализуется вычислительной схемой, латентность которой равна r тактов, поток выходных данных будет иметь

скважность, также равную r, поскольку для вычисления k+1-го данного необходимо дождаться появления k-го. Но в том случае, если есть возможность заранее вычислить значения a^{r-1} и c, можно реализовать схему, соответствующую выражению (1.4). Латентность этой схемы также будет равна r, однако на выходе мы получим значение не следующего x, а отстоящего от входного на r. Поскольку значение r как правило невелико, проведя некоторую предварительную подготовку данных и обеспечив тем самым предварительное заполнение вычислительного конвейера, мы можем добиться плотного потока данных через вычислительную схему, увеличив её быстродействие примерно в r раз по сравнению с исходным вариантом.

Описанные выше способы обеспечивают ускорение вычислений, пропорциональное значению латентности вычислительного блока. Таким образом, сформулируем пятый принцип эффективной обработки самоподобных структур на РВС: *для увеличения быстродействия вычислительной структуры, содержащей обратные связи, следует произвести их раскрытие методом конвейера в конвейере или автоподстановкой.*

1.8 Выводы

1) Фрактальные алгоритмы находят применение в различных областях науки и техники – математическом моделировании, компьютерной графике, медицине, геологии и др. С их помощью улучшаются предсказательные возможности различных математических моделей, качество компьютерной графики, показатели сжатия данных.

2) Из-за своих особенностей фрактальные алгоритмы обладают значительной вычислительной сложностью, поэтому их компьютерная реализация требует больших вычислительных мощностей. Также важным нюансом является необходимость обмена данными между вычислительными ядрами, что мешает

производить эффективное масштабирование алгоритмов при увеличении количества вычислительных узлов.

3) Стандартные методы и средства решения проблем, возникающих при реализации фрактальных алгоритмов на многопроцессорных вычислительных системах, не обеспечивают высокую реальную производительность. Существующие реализации фрактальных алгоритмов для многопроцессорных вычислительных систем, построенных на основе универсальных процессоров или графических ускорителей, достигают меньшей производительности, нежели системы, использующие ПЛИС.

4) Особенности архитектуры ПЛИС обладают большим потенциалом для решения задач фрактального типа, за счет возможности адаптировать структуру вычислителя для каждого типа задач. Однако существующие реализации фрактальных алгоритмов разработаны для акселераторов на основе только одной ПЛИС, которое не являются самостоятельными устройствами и требуют для своего функционирования наличия персонального компьютера. Также разработчики не предусматривают возможность масштабирования алгоритмов на систему, содержащую больше количество кристаллов ПЛИС.

5) В настоящее время существуют реконфигурируемые вычислительные системы, содержащие в себе большое количество кристаллов ПЛИС и работающие как самостоятельное вычислительное устройство, а также обладающие высокой реальной производительностью. Однако для этих систем не существует методов работы с фрактальными алгоритмами, т.к. данные вычислительные системы оперируют прямоугольными массивами данных.

6) В настоящее время не существует способов представления фрактальных структур для вычислительных систем со структурным способом вычислений, обладающих удовлетворительной эффективностью. Реализация ветвящихся структур памяти на ПЛИС может привести к повышению требований к синхронизации, в случае, если данные будут разделены внутри кристалла. Таким

образом, для эффективной работы с различными фрактальными алгоритмами возникает необходимость создания специальных для РВС структур данных.

7) Сформулированы принципы синтеза эффективных параллельно-конвейерных программ для обработки самоподобных структур на РВС:

- для работы с самоподобными структурами на РВС необходимо использовать структурный подход организации вычислений;

- при реализации фрактальных алгоритмов на РВС следует отдавать предпочтение распараллеливанию по итерациям, а не по слоям;

- для масштабирования вычислений при реализации фрактальных алгоритмов на РВС необходимо объединять в вычислительные структуры подобные подграфы, а не изоморфные;

- для увеличения эффективности алгоритма на РВС необходимо выполнять предварительную сортировку входных данных с учетом их особенностей;

- для увеличения быстродействия вычислительной структуры, содержащей обратные связи, следует уменьшить скважность потока операндов методами конвейера в конвейере или автоподстановки.

2 МЕТОДЫ ПАРАЛЛЕЛЬНОЙ РЕАЛИЗАЦИИ ФРАКТАЛЬНОГО СЖАТИЯ И ДЕКОМПРЕССИИ ИЗОБРАЖЕНИЙ

В настоящее время количество графической информации, обрабатываемой в цифровых системах, передаваемой по линиям связи и хранящейся в банках данных, продолжает увеличиваться с каждым годом. Это связано не только с ростом популярности дистанционного обучения, но и с развитием медицинской техники, научных исследований, а также других отраслей, где используется графическая информация. Чтобы обеспечить наращивание объема хранилищ данных и скорость передачи каналов, требуются большие финансовые затраты и физические ресурсы. В этом случае создание и применение более эффективных алгоритмов сжатия графической информации, которые позволят сократить затраты на хранение и передачу данных, будет являться более выгодным решением. Одним из наиболее перспективных методов архивации изображений является фрактальное сжатие. Этот метод обеспечивает высокие коэффициенты компрессии данных, сохраняя при этом высокое качество распакованного изображения.

2.1 Общие принципы фрактального сжатия изображений

Фрактальная архивация основана на том, что на основе поиска самоподобных участков, изображение переводится в коэффициенты системы итерируемых функций (СИФ). При этом пиксели изображения рассматриваются в качестве точек трехмерного пространства (x-координата, y-координата, яркость).

$$w\begin{pmatrix} x^* \\ y^* \\ I^* \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & u \end{pmatrix}\begin{pmatrix} x \\ y \\ I \end{pmatrix} + \begin{pmatrix} e \\ f \\ v \end{pmatrix} \qquad (2.1)$$

Здесь x^*, y^* , I^* - координаты и яркость рангового блока, x, y , I - координаты и яркость доменного блока, a ,b, c, d, e, f - коэффициенты

преобразований координат на плоскости, u – коэффициент сжатия яркости, v – сдвиг по яркости.

На рисунке 2.1 показано, как выглядит применение данного преобразования к изображению.

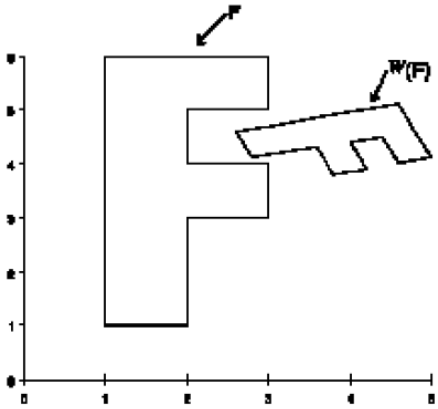


Рис. 2.1 – Аффинные преобразования над изображением

Процесс преобразования фрагментов изображения с помощью аффинных преобразований описан в книге М. Барнсли «Фрактальное сжатие изображения» [26]. Там вводится понятие фотокопировальной машины, состоящей из экрана с исходным изображением и системы линз, которые проецируют искаженное изображение на другой экран. По-разному расставляя линзы и меняя такие их параметры как, например, фокусное расстояние и степень искажения, можно получить различные наборы изображений. Также в мысленном эксперименте линзы могут менять яркость изображения и проецировать области с произвольной границей. В результате таких преобразований обнаруживаются самоподобные участки в изображении, что приводит к увеличению детализации изображения и позволяет сохранять его в более подробном виде. В соответствии с данным методом, изображение одновременно разбивается на два множества: неперекрывающихся ранговых блоков и перекрывающихся доменных блоков, пример подобного разбиения показан на рисунке 2.2.

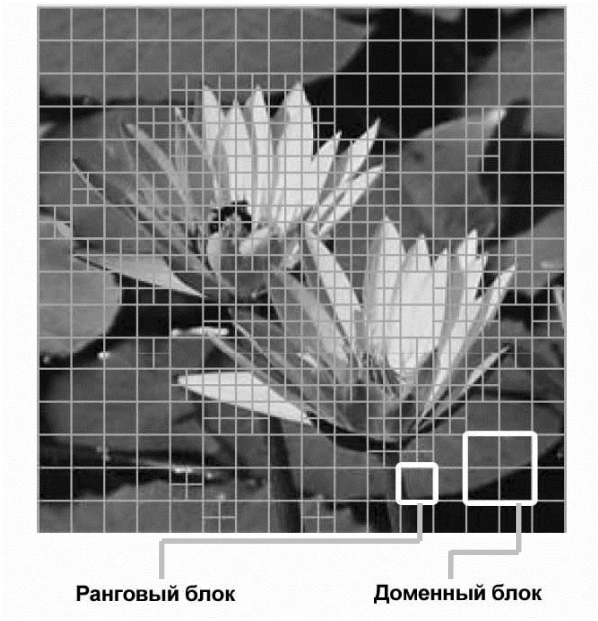


Рис. 2.2 – Пример адаптивного разбиения на доменные и ранговые блоки

Существуют различные способы разбиения изображений на ранговые блоки, наиболее популярными являются разбиение на блоки одинакового размера и адаптивное разбиение квадродеревом. Второй вариант дает возможность плотно заполнять ранговыми блоками меньшего размера части изображения, содержащие мелкие детали, благодаря чему улучшается качество кодирования изображения и сохраняется максимальная детализация при восстановлении сжатого изображения. Следует заметить, что чрезмерное измельчение ранговых блоков может привести к большим потерям качества декодированного изображения [92].

В пункте 1.2 приведен алгоритм фрактального сжатия изображений, поэтому в данной главе приведем только суть фрактального сжатия. После разбиения изображения на ранговые блоки его покрывают сеткой доменных блоков. Все доменные блоки сжимают до размеров ранговых и осуществляют их повороты по восьми направлениям. После этого вычисляют сдвиг по яркости для каждой пары ранговый-доменный блок по формуле

$$v = \left(\sum_{i=1}^{m} \sum_{j=1}^{m} r_{ij} - u \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} \right) / m^2, \qquad (2.2)$$

где m - количество пикселей в ранговом и усредненном доменном блоках, $d_{ij} \in D$ - пиксель усредненного доменного блока, $r_{ij} \in R$ - пиксель рангового блока.

Далее для определения оптимального соответствия доменного блока ранговому необходимо вычислять расстояние между ними по формуле

$$E(R,D) = \sum_{i=1}^{m}\sum_{j=1}^{m}\left(ud_{ij} + v - r_{ij}\right)^2 \qquad (2.3)$$

где m - количество пикселей в ранговом и усредненном доменном блоках, u – коэффициент сжатия яркости, $d_{ij} \in D$, $r_{ij} \in R$.

Если полученное значение не превышает заранее заданной допустимой погрешности, то считается, что пара ранговый-доменный блок сформирована, производится сохранение коэффициентов СИФ. Данный ранговый блок исключается из перебора, и начинается обработка следующего блока.

В случае если после перебора всех доменных блоков не нашлось таких, которые могли бы составить пару ранговому блоку, то в качестве пары берется тот домен значение E(R,D) для которого является минимальным.

Одна из основных сложностей фрактального сжатия заключается в необходимости полного перебора для поиска соответствующих доменных блоков. Каждый раз при переборе должны сравниваться два массива, что делает данную операцию достаточно длительной и ресурсоемкой. Существуют различные варианты оптимизации этой процедуры, которые позволяют ускорить ее выполнение и сократить затраты ресурсов.

2.2 Решение задачи фрактального сжатия изображений на РВС

При осуществлении фрактального сжатия изображения над доменными блоками выполняются четыре базовые операции: сжатие доменного блока до размеров рангового (W), поворот сжатого блока (F_i, где i=0..7), сдвиг по яркости (L), вычисление расстояния между преобразованным доменным блоком и ранговым (E) [93] (работа [93] выполнена в соавторстве с Левиным И.И.). При

этом для выполнения первых двух операций требуется только доменный блок, тогда как сдвиг по яркости и вычисление расстояния выполняются для каждой пары ранговый-доменный блок. На рисунке 2.3 показано выполнение операций для одной пары доменного и рангового блока при параллельной подаче пикселей. Для удобства обозначим этот граф буквой λ .

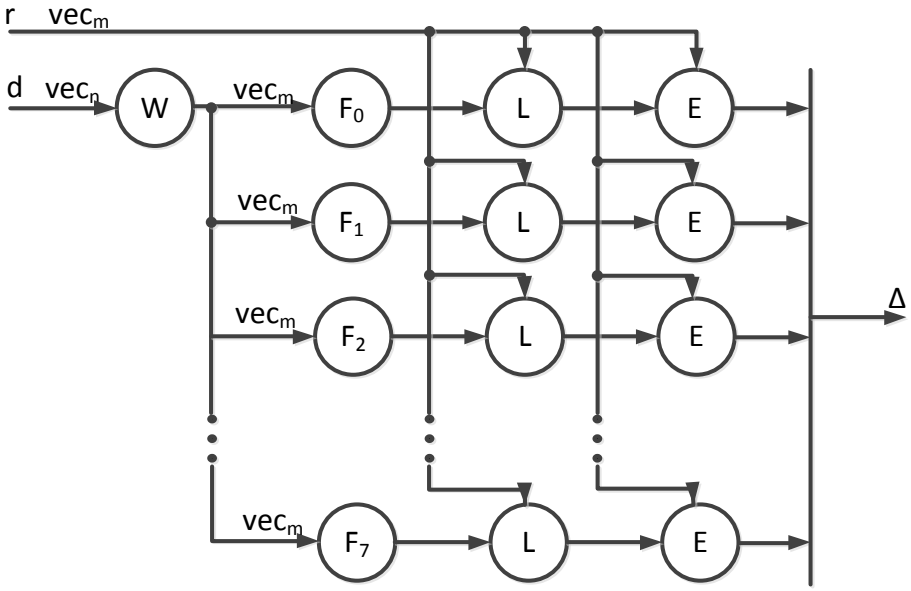


Рис. 2.3 – Базовые операции. Граф λ

В этом случае время работы блока составит сумму времени предварительной подготовки данных t_{prep}, времени заполнения конвейера t_{conv} для блоков сдвига по яркости и вычисления расстояния между преобразованным доменным блоком и ранговым. Также учитывается скважность S для конвейерной подачи данных.

$$t_\lambda = t_{prep} + t_{conv} + S \cdot \tau ,$$

где τ – длительность одного такта.

Далее раскроем все составляющие графа λ.

Пусть размер доменного блока – n пикселей, а размер рангового – m, тогда получим $k = n / m$ - количество пикселей доменного блока, сжимаемых до одного. Процедуру усреднения необходимо произвести для каждых k пикселей доменного блока на входе в вычислительную структуру. Эти участки не пересекаются между собой, и возможно параллельное выполнение данной процедуры для всего

доменного блока. Вычислительная структура для описанной выше процедуры приведена на рисунке 2.4.

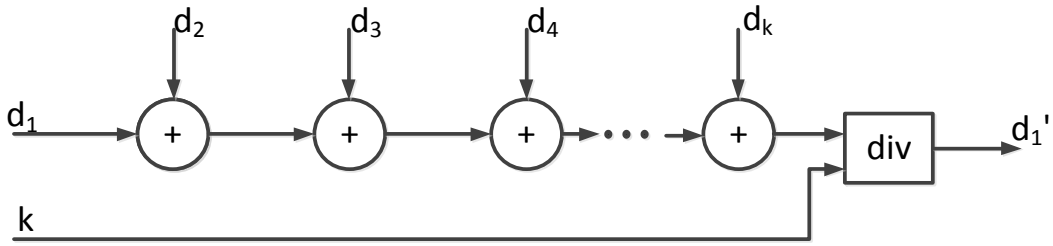


Рис. 2.4 – Процедура усреднения пикселей

Для уменьшения латентности вычислений можно реализовать процедуру усреднения пикселей доменного блока посредством пирамиды сумматоров, осуществляя параллельную подачу значения каждого пикселя (рис. 2.5).

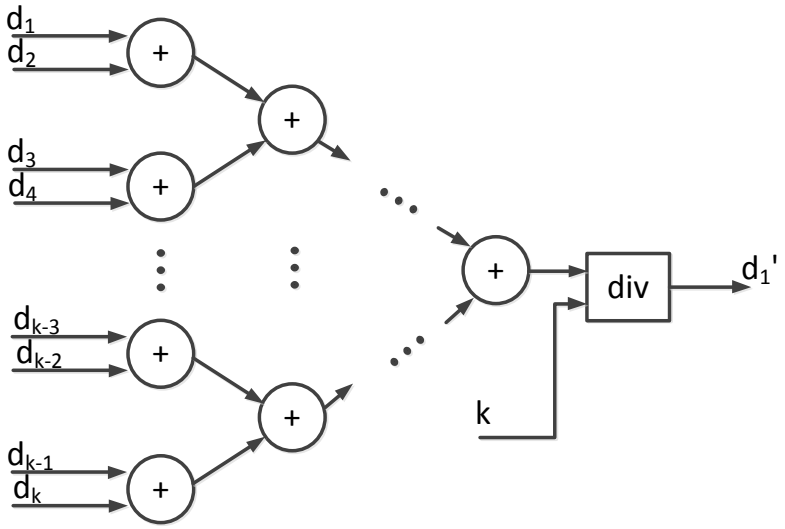


Рис. 2.5 – Сжатие пикселей доменного блока с помощью пирамиды сумматоров

Время заполнения конвейера вычислительной структуры, показанной на рисунке 2.5, составит

$$t_{wp} = \left(\right]\log_2 k\left[L_s + L_d \right)\tau ,$$

где L_s – латентность сумматора, L_d – латентность делителя. При такой подаче данных скважность их потока будет равна единице. Однако, поскольку в одном доменном блоке может быть от 64 и более пикселей, нецелесообразно

использование пирамиды сумматоров, т.к. количество каналов для подачи входных данных в большинстве современных РВС является критическим ресурсом. В этом случае использование пирамиды сумматоров нецелесообразно и следует производить данную процедуру при помощи сумматора с накоплением, как показано на рисунке 2.6.

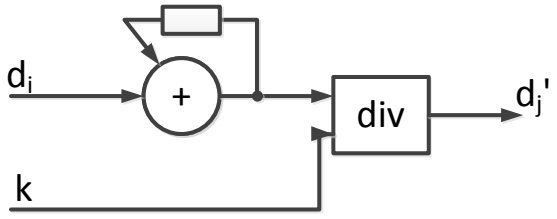


Рис. 2.6 – Выполнение сжатия пикселей доменного блока посредством сумматора с накоплением

При последовательной подаче данных время работы блока увеличится, однако это позволит сэкономить входные каналы данных, и затем нивелировать снижение скорости работы посредством установки нескольких таких блоков, работающих параллельно. При этом время заполнения конвейера составит

$$t_{wa} = \left(L_s + L_d\right)\tau .$$

В этом случае скважность потока данных будет равна k. Хотя скорость работы данной вычислительной структуры уступает варианту с пирамидой сумматоров, за счет экономии каналов возможно разместить несколько вычислительных слоев, что в конечном итоге приведет к увеличению быстродействия.

При конвейерной подаче данных результат из блока W будет выходить каждые k тактов, формируя во всем вычислительном конвейере поток данных с задержками в k тактов. В соответствии с принципами эффективной реализации фрактальных алгоритмов для решения этой проблемы можно поставить k параллельно работающих блоков W, принимающих на вход части одного доменного блока, из которого будет сформирован блок, сжатый до размеров

рангового, при этом поток результатов, выходящий из обновленного блока W’, не будет иметь задержек k (рис. 2.7).

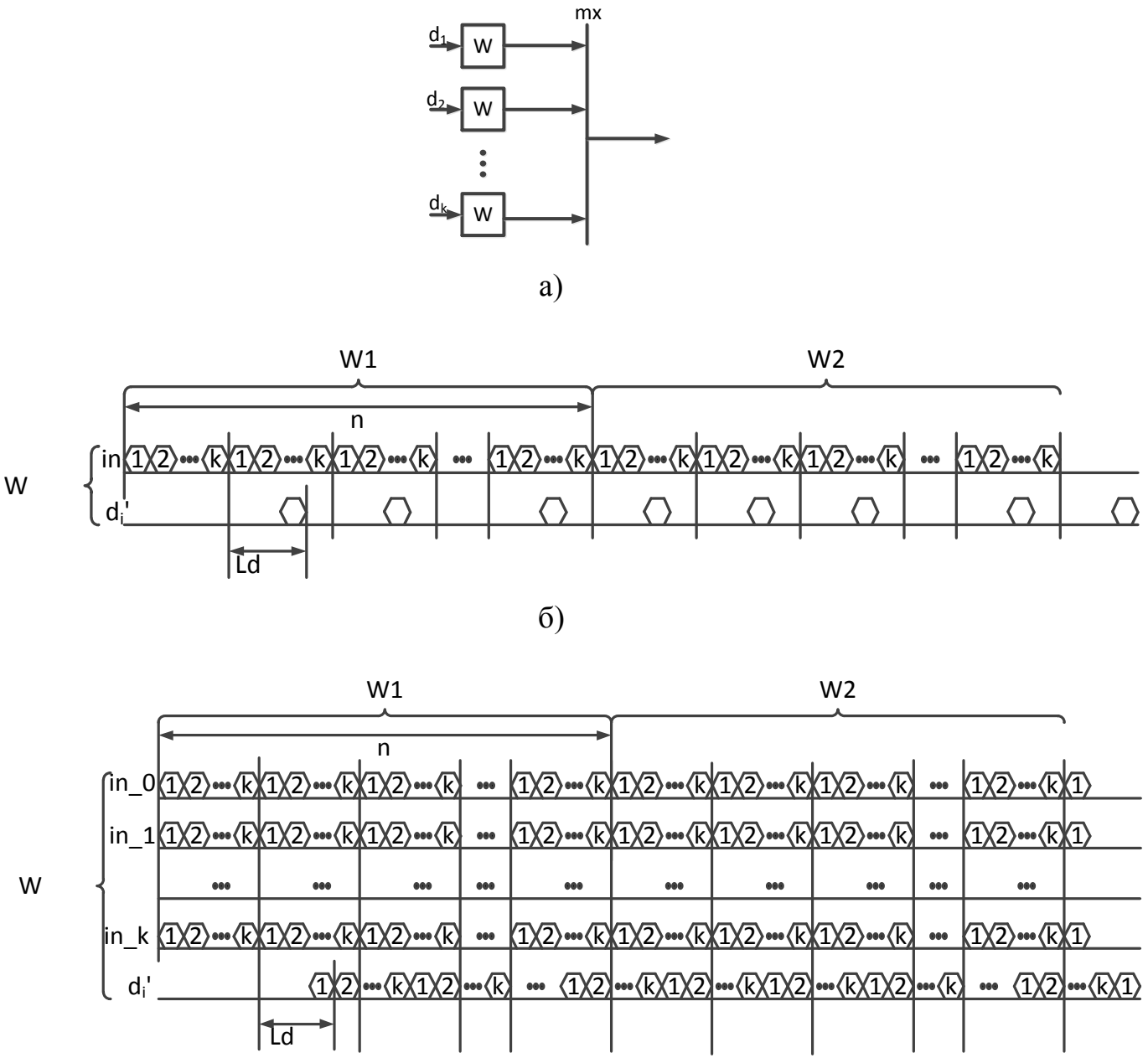
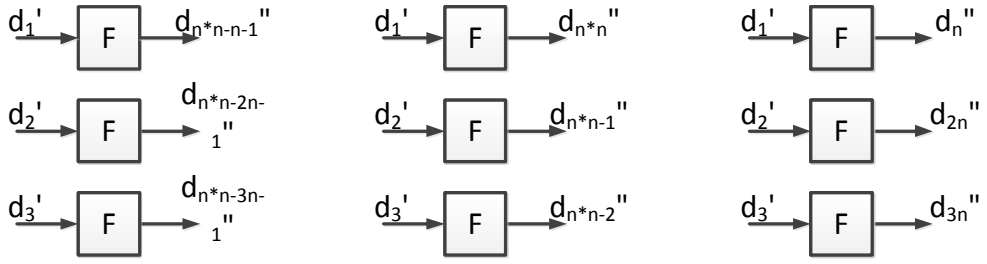


Рис. 2.7 – Уплотнение потока данных а) вычислительная структура, б) временная диаграмма для одного блока W, в) временная диаграмма для k блоков W

После сжатия доменного блока выполняется следующее аффинное преобразование – изменение угла поворота F. В общем случае угол может быть

любым, но исходя из прямоугольной формы удобнее всего использовать 8 вариантов ротации - 0°, 90°, 180°, 270° и аналогичные параметры для зеркального отображения доменного блока. Эти операции могут быть реализованы на РВС с помощью простых перекоммутаций потоков данных (рис. 2.8).



а) б) в)

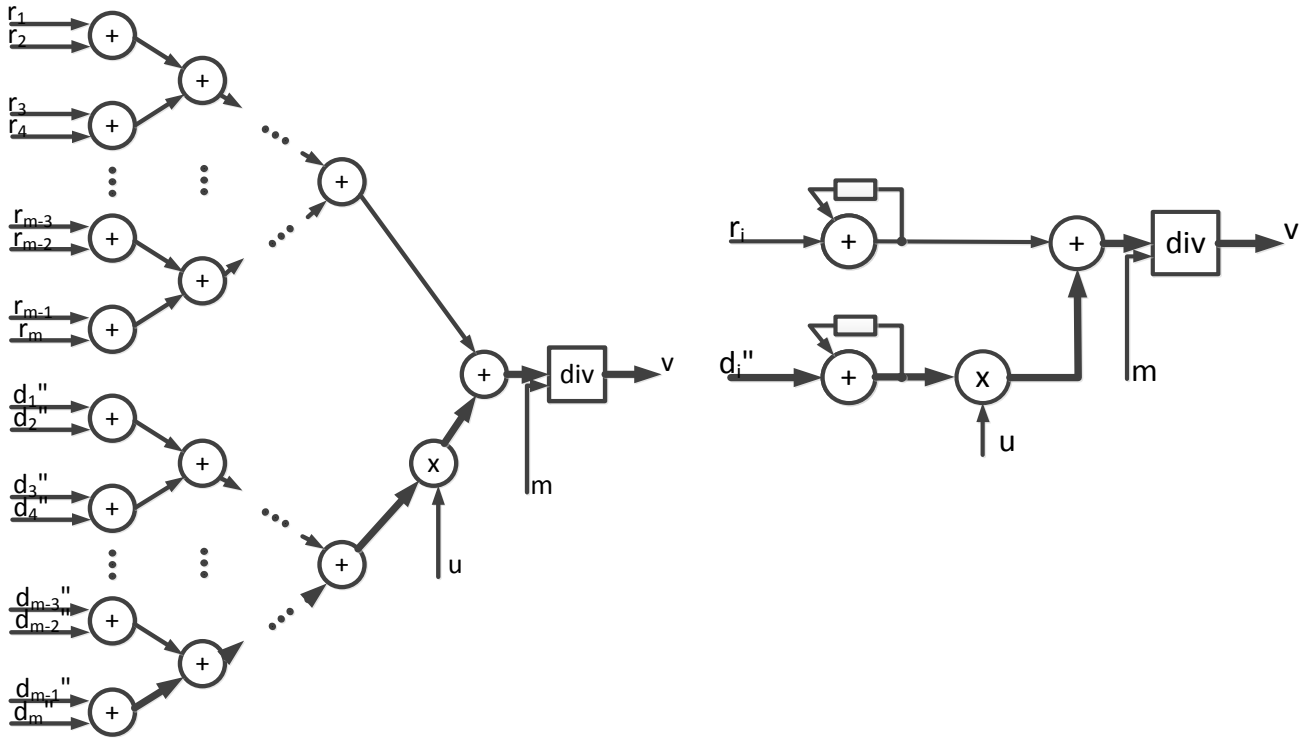
Рис. 2.8 – Поворот массива данных на а) 90°, б) 180° и в) 270°

После процедуры поворота образуется 8 потоков данных, в которых перекоммутированы исходные значения, после этого вычисляется по формуле (2.2) значение сдвига по яркости L. Данная операция имеет возможность параллельного выполнения, т.к. сложение значений пикселей рангового и доменного блока выполняется независимо друг от друга. Следует заметить, что результат сложения пикселей доменного блока в отличие от результата сложения пикселей рангового блока умножается на коэффициент сжатия яркости u, что даёт дополнительную задержку и делает данный участок информационного графа вычислительной структуры частью критического пути. В результате работы вычисляется сдвиг по яркости v, который в дальнейшем будет сохранен, как часть вектора коэффициентов СИФ в случае успешного покрытия рангового блока доменным.

При выполнении операции сдвига по яркости возможны различные варианты подачи данных. При использовании пирамиды сумматоров (рис. 2.9 а) время заполнения вычислительного конвейера составит

$$t_{Lp} = \left(\right]\log_2 m\left[L_s + L_s + L_m + L_d \right)\tau ,$$

где L_m – латентность умножителя, m – число пикселей в ранговом блоке. Скважность потока данных будет равна единице, что позволит добиться большой скорости вычислений, но затраты каналов входных данных будут значительными.



а) б)

Рис. 2.9 – Вычисление сдвига по яркости а) параллельная подача данных, б) конвейерная подача данных

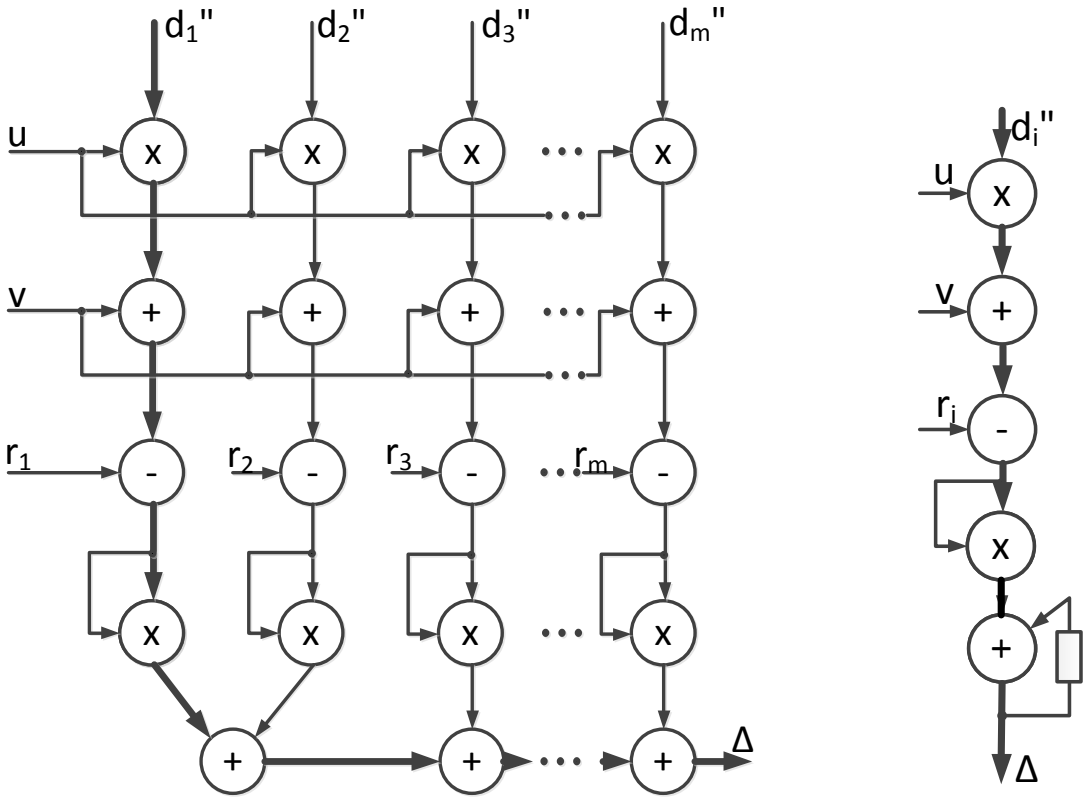
В силу вышеупомянутого дефицита входных каналов, характерного для большинства современных РВС, предпочтительным представляется отказ от пирамиды сумматоров в пользу сумматоров с накоплением (рис. 2.9 б). При этом время заполнения конвейера составит

$$t_{La} = \left(2L_s + L_m + L_d\right)\tau .$$

Значение скважности потока данных через данную вычислительную структуру будет равно m. Это замедление можно компенсировать за счет возможности расположения на РВС параллельно работающих вычислительных слоев.

Последней выполняется процедура E, определяющая расстояние между конкретной парой ранговый-доменный блок по формуле (2.3). Выходное

значение, полученное в результате данной операции, не является результирующим коэффициентом СИФ, оно сравнивается с заданным ранее параметром, определяющим требуемую точность вычислений. При этом возможно параллельное вычисление значения каждого слагаемого в формуле (2.3). На рисунке 2.10 показаны варианты вычислительных структур для вычисления расстояния между ранговым и доменным блоками с параллельной и конвейерной подачей данный, критический путь информационного графа выделен жирным.



а) б)

Рис. 2.10 – Блок вычисления расстояния а) с параллельной подачей данных, б) с конвейерной подачей данных

Время заполнения конвейера в этом случае составит

$$t_{Ep} = \left(2L_m + mL_s + L_{sub}\right)\tau ,$$

где L_{sub} – латентность вычитателя. Значение скважности будет равно единице, но так же, как и в предыдущих вариантах для обработки всего доменного блока одновременно требуется большое количество каналов.

На рисунке 2.10 б показан вариант реализации блока вычисления расстояния с последовательной подачей данных. Время заполнения вычислительного конвейера в этом случае составит

$$t_{Ea} = \left(2L_m + 2L_s + L_{sub}\right)\tau .$$

Значение скважности потока данных также будет равно m, однако экономия входных каналов данных позволит разместить больше устройств на кристалле, что в конечном итоге приведет к увеличению быстродействия.

В соответствии с принципами организации эффективной обработки фрактальных структур на РВС выполнение операций над доменными и ранговыми блоками можно распараллелить по слоям, т.к. между этими вершинам графа отсутствуют информационные связи. Этот способ позволяет значительно повысить производительность вычислительной системы, но подходит только для ограниченного количества РВС, обладающих большим ресурсом входных каналов для данных. Поэтому, согласно второму принципу, следует отдать предпочтение распараллеливанию по итерациям, при котором существует последовательная информационная зависимость между вычислительными узлами, т.е. данные из первого узла поступают во второй, потом в третий и так далее. Сочетание распараллеливания по слоям и итерациям, где большая часть вычислительного ресурса выделена для итераций, приведет к увеличению реальной производительности вычислительной системы.

При попытке добиться максимально возможного параллелизма для РВС с ограниченным количеством каналов подачи данных, можно осуществить предзагрузку ранговых блоков в FIFO и получить структуру, представленную на рисунке 2.11.

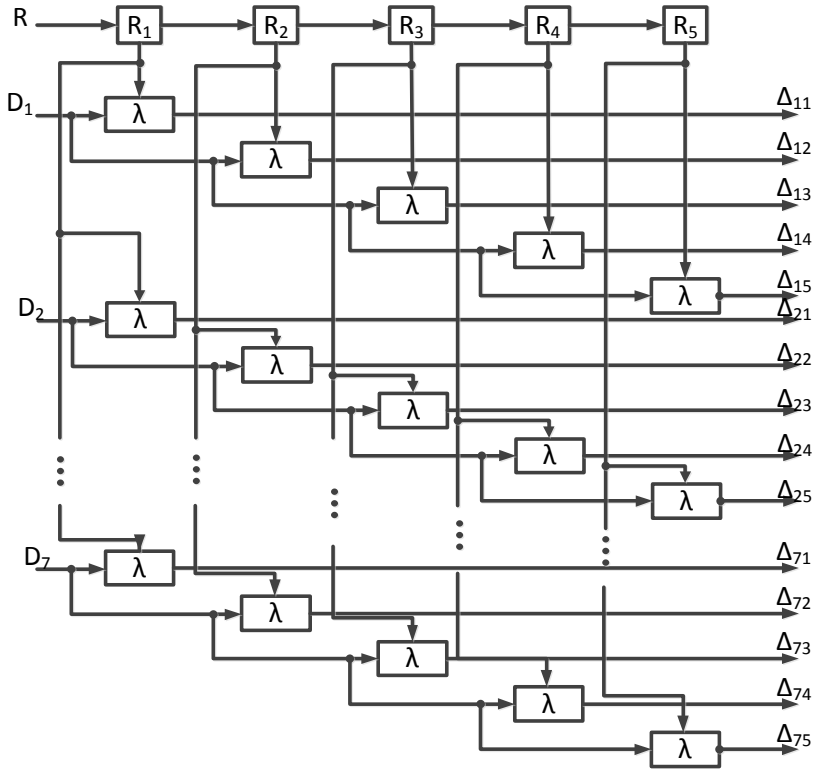


Рис. 2.11 – Максимальное количество входов

Такое решение обладает существенным недостатком – после завершения работы вычислительной схемы последует долгое ожидание выгрузки всех результатов в единственное отведенное для вывода данных КРП.

Из вышесказанного следует, что для загрузки данных лучше использовать половину имеющихся каналов, а вторую половину использовать для выгрузки данных, тогда получим структуру, показанную на рисунке 2.12. Для ее реализации необходимо обеспечить транзит данных и результатов вычислений через конвейер. Полученный после модификации блок обозначим λ*.

Время заполнения конвейера блока λ* будет равно

$$t_\lambda = t_w + t_L + t_E .$$

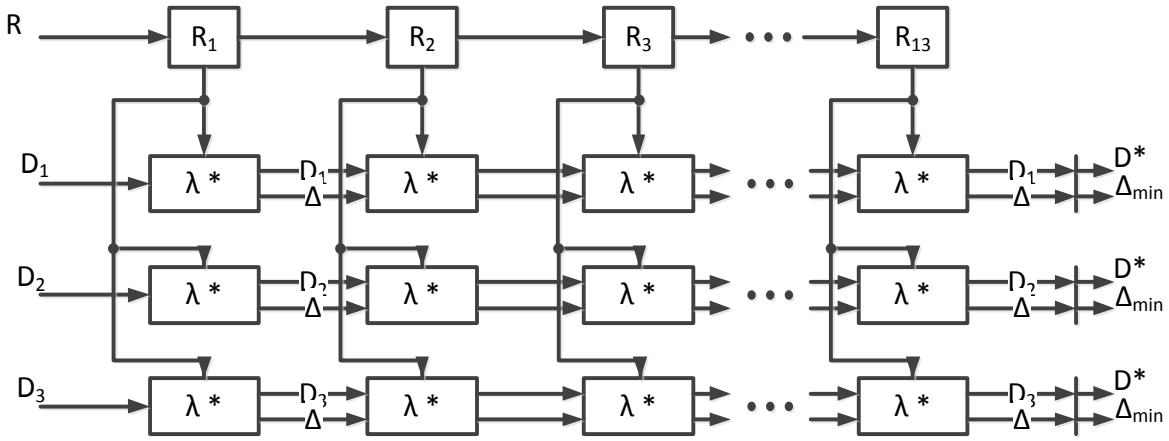


Рис. 2.12 – Равное количество входов и выходов

Время работы конвейера при последовательной подаче данных одного цветового канала изображения в формате RGB для такого расположения блоков будет равно

$$T_{conv\lambda} = \left(t_w + t_L + t_E\right)N + MS\tau,$$

где S – значение скважности потока данных, при параллельной подаче данных равное единице, при конвейерной подаче данных равное m (число пикселей в исходном доменном блоке), M – количество подаваемых доменных блоков, N – глубина конвейера.

Однако при данной реализации возникает проблема с глубиной конвейера, которая может привести к неоптимальному распределению ресурсов. Когда коэффициенты СИФ удовлетворяют заданной погрешности, следует остановить процесс сравнивания рангового блока с доменными, и перейти к работе со следующим ранговым блоком. Таким образом, для улучшения производительности следует рассмотреть возможность оптимизации глубины конвейера. Например, можно уменьшить глубину конвейера, разделив его на несколько промежуточных этапов, что позволит более эффективно распределить ресурсы и улучшить общую производительность системы.

При конвейерном способе размещения блоков λ* происходит повторение операций сжатия и поворота для доменных блоков. Эти операции можно провести

один раз перед началом конвейера, и для дальнейших вычислений подавать модифицированные доменные блоки D_i'' (рис. 2.13). Таким образом, блок λ* будет разделен на блок WF, который будет выполнять сжатие доменного блока до размеров рангового и восемь вариантов поворота, и блок LE, вычисляющий коэффициент сдвига по яркости и расстояние между ранговым и доменным блоками для каждой из восьми модификаций сжатого блока.

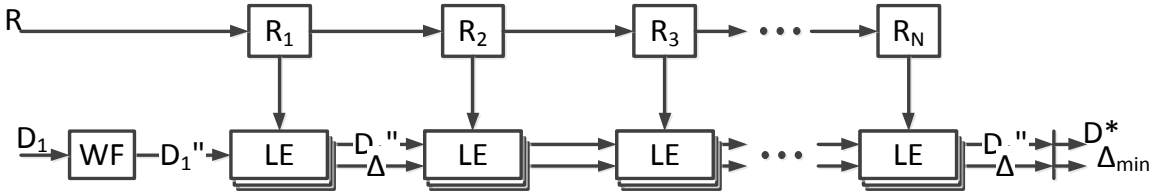


Рис. 2.13 – Вынос в начало конвейера операций сжатия и поворотов

При такой реализации время работы конвейера составит

$$T_{conv} = t_{WF} + \left(t_L + t_E\right)N + MS\tau .$$

Таким образом, экономия времени составит (L_s + L_d)(N - 1) тактов по сравнению с предыдущим вариантом. Параллельная подача данных в вычислительный конвейер позволит уменьшить скважность, однако количество входных каналов является критическим ресурсом, поэтому согласно второму принципу эффективной организации вычислений на РВС, следует сочетать распараллеливание по слоям и итерациям, при этом выделяя больший вычислительный ресурс на увеличение количества итераций в конвейере (рис. 2.14).

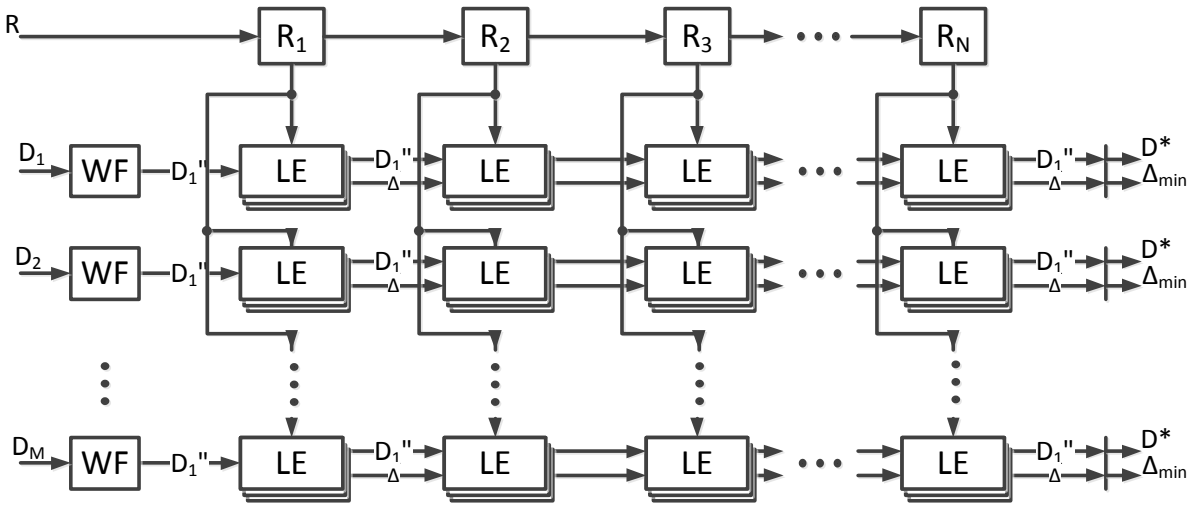


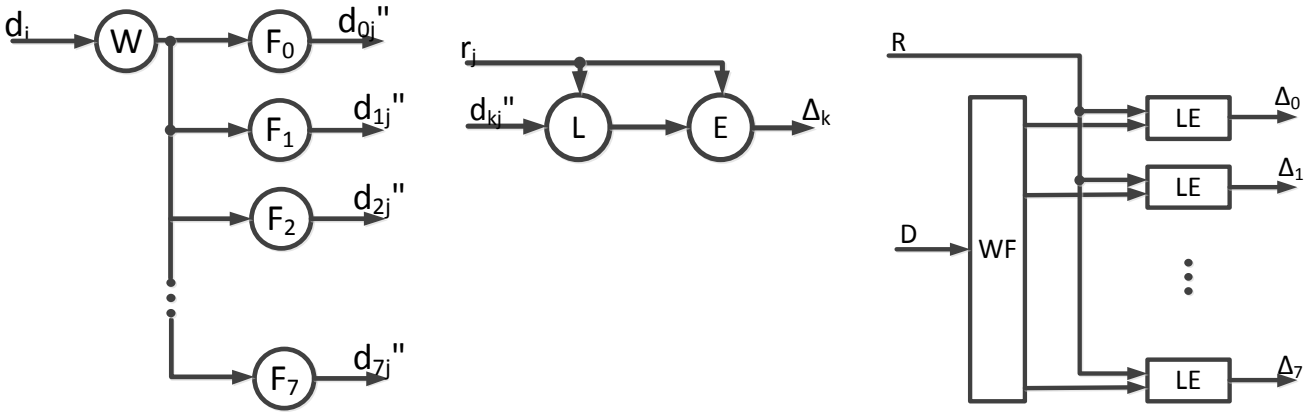
Рис. 2.14 – Вынос в начало конвейера операций сжатия и поворотов

Для K слоев время работы вычислительной схемы составит

$$T_{convK} = t_{WF} + (t_L + t_E)N + \frac{MS}{K}\tau .$$

Количество вычислительных слоев зависит от конкретной используемой РВС и может быть увеличено или уменьшено в зависимости от характеристик задачи.

На рисунке 2.15 показаны структура блоков WF и LE, а также их коммутация в вычислительной схеме.



а) б) в)

Рис. 2.15 – Блоки WF и LE, составлявшие блок λ*: а) блок WF, б) блок LE, в) коммутация блоков WF и LE

Для осуществления попарного сравнения всех доменных и ранговых блоков, необходимо произвести количество прогонов доменных блоков через конвейер, равное общему числу ранговых блоков, деленному на число уникальных ранговых блоков в одном прогоне.

Подача доменных блоков в вычислительную схему осуществляется посредством загрузки их в распределенную память, откуда осуществляется их вычитывание в виде непрерывного потока данных. Ранговые блоки загружаются во внутреннюю память ПЛИС. Поскольку ресурс внутренней памяти ограничен при обработке изображений больших объемов, возможна порционная загрузка ранговых блоков в память.

При частичном сравнении доменных и ранговых блоков между собой есть возможность избежать повторного прогона доменных блоков через конвейер при дозагрузке ранговых блоков. Если выполняется процедура полного перебора ранговых и доменных блоков, то следует исключать из перебора ранговые блоки, для которых найдена пара, и далее сравнивать доменные блоки только с непокрытыми ранговыми. Для этого также следует сохранить коэффициенты СИФ полученной пары.

Номера доменных и ранговых блоков также следует сохранять вместе с советующими им коэффициентами СИФ – номером поворота и значением сдвига по яркости. Без такого задания однозначного соответствия невозможно будет восстановить исходное изображение во время процедуры декомпрессии. Номера ранговых или доменных блоков добавляются перед данными, участвующими в вычислениях. Также этот номер сопровождает доменный блок и после аффинных преобразований – сжатия и поворотов. Номера, маркирующие доменные и ранговые блоки, не участвуют в вычислениях, поэтому их следует отделить от данных перед началом выполнения каждой процедуры. Также необходимо сохранять номера рангов и доменов, находящихся в работе, для последующего их сохранения в результирующую структуру. На рисунке 2.16 показано, как номер

доменного блока сохраняется в регистре, а данные идут дальше в вычислительную структуру. Для ранговых блоков операция отделения номера от данных будет проводиться аналогично.

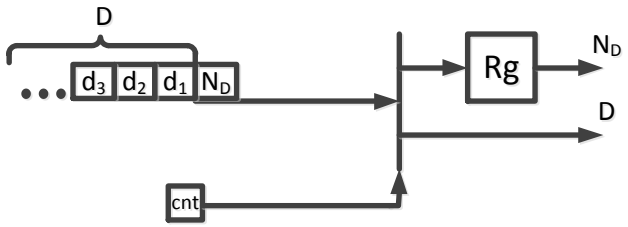


Рис. 2.16 – Отделение номера доменного блока от полезных данных

Каждый доменный блок имеет после процедуры ротации F восемь ориентаций, сдвиг про яркости v и расстояние Δ вычисляются для всех полученных ориентаций. Чтобы избежать хранений лишней информации, значения Δ для разных ориентаций одного доменного блока сравниваются между собой, после этого выбирается наименьшее из расстояний, и уже оно сравнивается с заданным заранее значением точности ε. На рисунке 2.17 показана схема сравнений результатов работы блоков E между собой, и последующее сохранение лучшего результата [94].

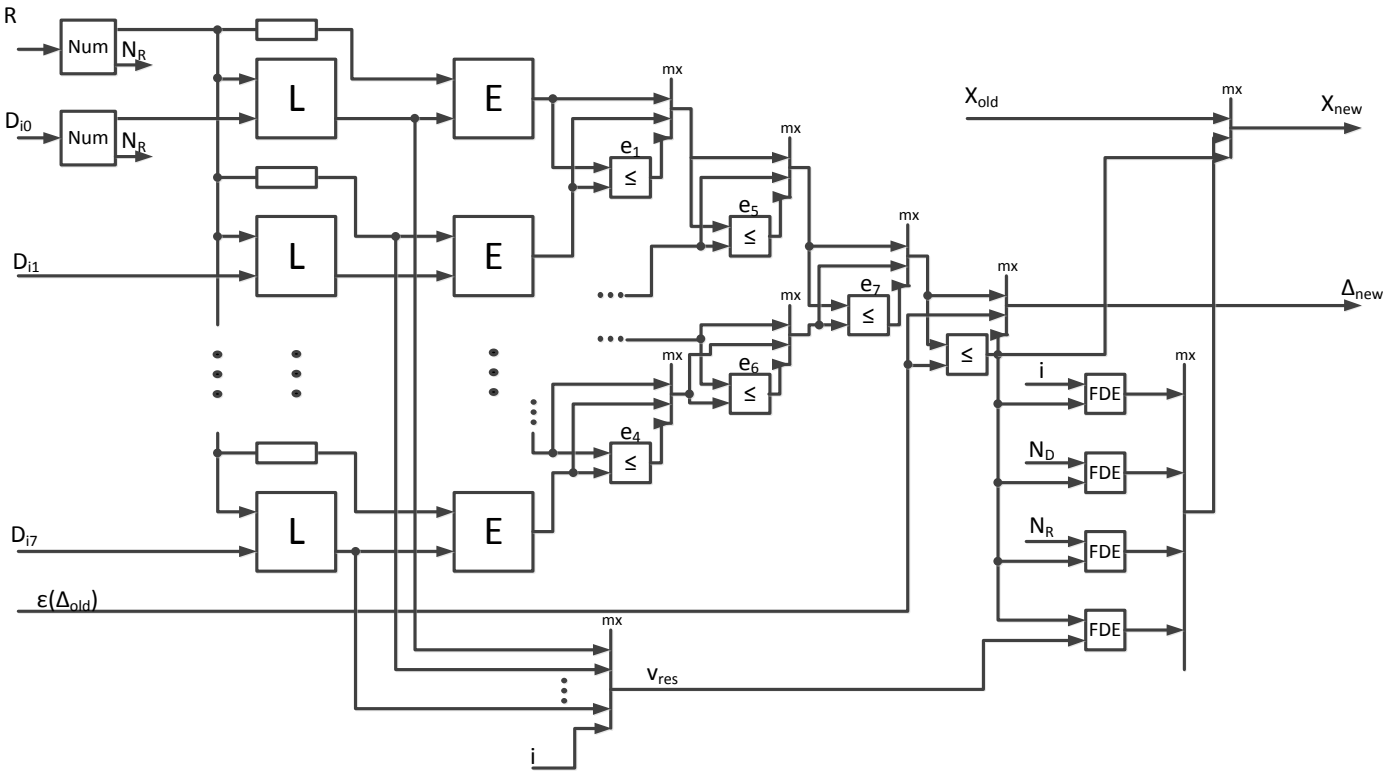


Рис. 2.17 – Формирование результирующей структуры, содержащей коэффициенты СИФ

Значения расстояния для разных ориентаций доменного блока сравниваются между собой без привязки к номеру поворота. Поскольку номер поворота i является одним из коэффициентов СИФ, составляющих результат кодирования изображения, необходимо определить какая именно ориентация доменного блока имеет наилучший результат после вычисления расстояния. Для этого можно вывести правило определения коэффициента поворота *i*:

$$i = \overline{e_7} b a;$$
$$b = e_7 e_5 \vee \overline{e_7} e_6;$$
$$a = e_1 e_5 e_7 \vee e_2 \overline{e_5} e_7 \vee e_3 e_6 \overline{e_7} \vee e_4 \overline{e_6} \overline{e_7},$$

где $e_1 \ldots e_7$ – значения компараторов, показанных на рисунке 2.17.

Если значение $E(R,D)$ меньше заданной погрешности, то будет сформирована новая структура $X = (N_r, N_d, i, v)$ для хранения результатов кодирования изображения, где N_r – номер рангового блока, N_d – номер доменного блока, i – поворотный коэффициент, v – сдвиг по яркости. Информация,

содержащаяся в одной ячейки, введенной структуры данных, устанавливает однозначное соответствие между ранговыми и доменным блоками, образующими пару, и полученными для них коэффициентами СИФ. Описанная структура данных имеет общие черты с множествами [89], которые описаны в пункте 1.6 первой главы. В отличие от множеств данная структура данных ставит в соответствие четыре массива данных разного формата, а не два. Следует отметить, что полученное соответствие не является взаимно однозначным, т.к. одному доменному блоку может соответствовать несколько ранговых.

Таким образом, в результате работы последовательность ранговых блоков (R_1,R_2,R_3,…) преобразуется в последовательность результатов (X_1,X_2,X_3..), которая может заполняться не по порядку, т.к. неизвестно точное время появления доменного блока, при сравнении с которым будут получены коэффициенты СИФ, удовлетворяющие заданной точности. Например, после прогона первой порции доменных блоков через вычислительный конвейер может получиться, что результат будет найден для ранговых блоков R_3 и R_5. Т.е. соответствие между ранговыми блоками и последовательностью результатов X будет выглядеть следующим образом $\left(R_1, R_2, \begin{bmatrix} R_3 \\ X_3 \end{bmatrix}, R_4, \begin{bmatrix} R_5 \\ X_5 \end{bmatrix}, R_6, \ldots \right)$.

Перед подачей следующей порции доменных блоков ранговые блоки R_3 и R_5 будут исключены из дальнейшего сравнения.

Подробно формирование структуры X для хранения результатов показано на рисунке 3.15. Там же показано, что при получении лучшего результата значения коэффициентов СИФ и номер соответствующего доменного блока могут быть заменены в процессе прохода данных через вычислительный конвейер. Перед каждой сменой порции доменных блоков и обновлением списка актуальных ранговых блоков каждый ранговый блок сравнивается с некоторым количеством доменов. При этом в первом вычислительном блоке конвейера полученное значение расстояния $E(R,D)=\Delta$ будет сравниваться с ε – установленным заранее значение точности, а последующих блоках конвейера

новое значение Δ будет сравниваться либо с ε, либо с предыдущим Δ, если подходящее было найдено ранее. Если новое Δ меньше предыдущего, то параметры, содержащиеся в соответствующей ячейке X будут обновлены.

Ранговые блоки загружаются во внутреннюю память перед началом вычислений и подаются на схему частями. Когда для конкретного рангового блока найдено его покрытие доменным, он исключается из вычислений, и на его место подаются данные, относящиеся к другому ранговому блоку. Поскольку один доменный блок может покрыть несколько ранговых, доменные блоки проходят через вычислительный конвейер несколько раз. Для этого необходимо иметь возможность подавать в вычислительную схему поток доменных блоков, который уже один раз прошел через нее. Для решения этой проблемы можно использовать контроллеры распределенной памяти (КРП), благодаря их свойству быть не только источниками, но и приемниками обрабатываемых данных. Применение КРП дает возможность организовывать множество синхронных потоков данных.

На рисунке 2.18 показано, как встраиваются КРП в вычислительную схему для осуществления многократных прогонов доменных блоков через конвейер. В каждом из блоков LE формируется маркер для записи данных в память КРП. Во время первого прогона из RAM1 происходит чтение данных, в это время RAM2 накапливает данные, которые уже прошли через конвейер. После того как RAM1 выгрузило все данные, произойдет переключение, и теперь RAM2 будет выгружать данные, а RAM1 накапливать.

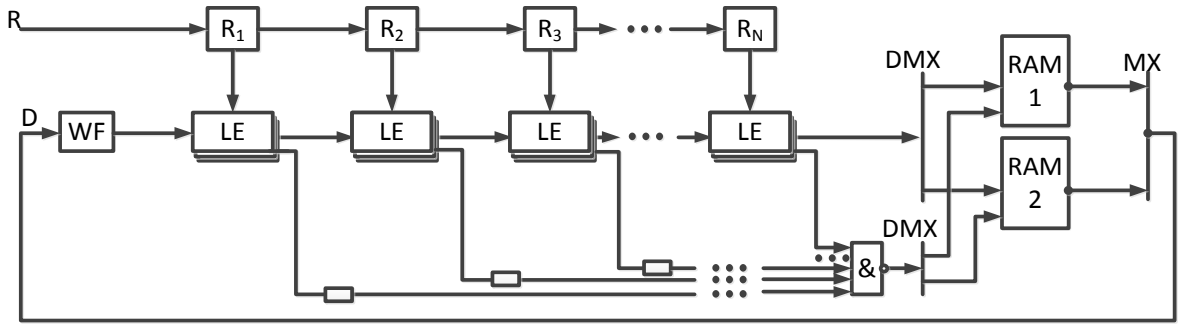


Рис. 2.18 – Многократный прогон доменных блоков через вычислительный конвейер

Проведем сравнение предложенной схемы с существующими реализациями фрактального сжатия изображений на РВС. В реализации, предложенной в работе [14], выполняется фрактальное сжатие изображений с полным перебором доменных и ранговых блоков, а в реализации, предложенной в работе [13], выполняется сжатие с частичным перебором доменных и ранговых блоков. Блок-схема одной из реализаций представлена на рисунке 2.19.

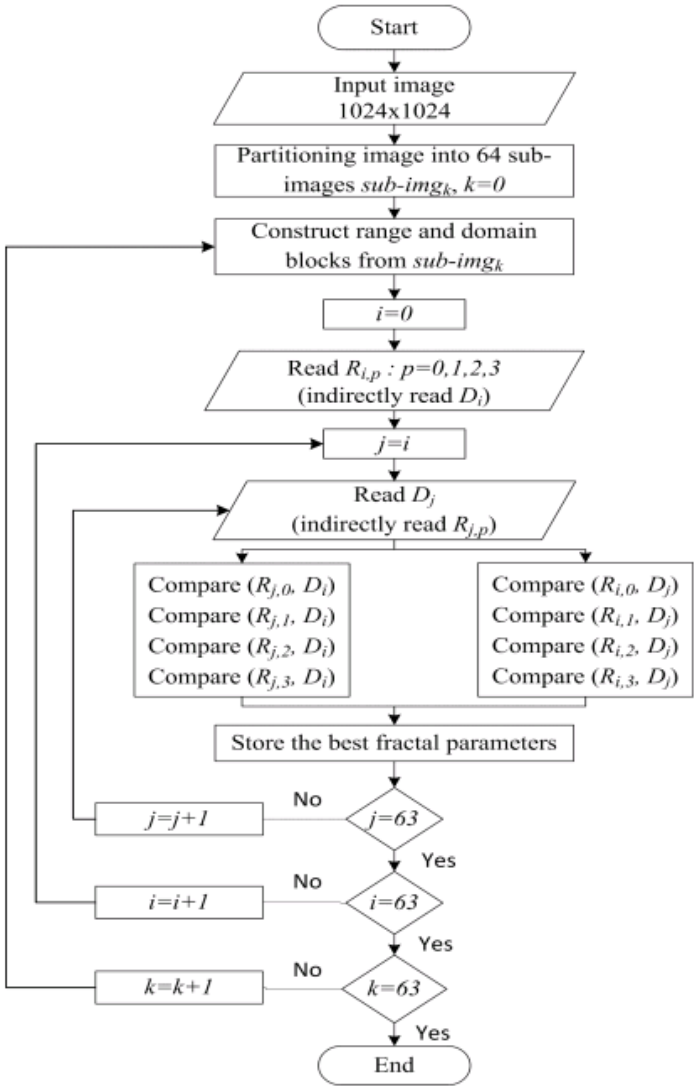


Рис. 2.19 – Блок-схема реализации фрактального сжатия изображений на ПЛИС в работе [13]

Существующие реализации отличаются следующими характеристиками: небольшая глубина конвейера (в приведенном выше примере она равняется

четырем), небольшое количество параллельно работающих вычислительных слоев K', большое количество обратных связей, увеличивающих скважность потока данных. Время работы вычислительной схемы в этом случае можно оценить по формуле

$$T_* = t_{wf*} + t_{le*}\frac{NMS_{le*}}{K'}, \qquad (8)$$

где t_{le*} – время заполнения конвейера блоков вычисления сдвига по яркости и расстояния, вычисленное с учётом значения глубины конвейера.

Скважность потока данных при такой реализации будет пропорциональна количеству прогонов одного доменного блока для сравнения его с ранговыми. Слагаемое t_{wf*} пренебрежимо мало по сравнению со вторым слагаемым, поэтому для сравнения времени работы схем мы можем им пренебречь:

$$T_* \approx t_{le*}\frac{NMS_{le*}}{K'}, \qquad T_{convK} \approx \left(t_{LE}N + \frac{Mn}{K}\right)\tau.$$

Сравним время работы предложенного в диссертации варианта и существующих реализаций

$$\frac{T_{convK}}{T_*} \approx \frac{t_{le}N + S_{le}M/K}{t_{le*}NMS_{le*}}\cdot K' = \left(\frac{1}{t_{le}NK} + \frac{1}{MS_{le*}}\right)\cdot K'$$

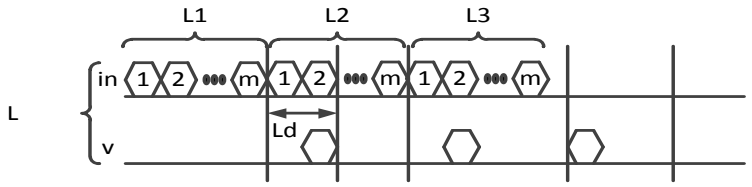
В силу того, что $K' << t_{le*}NK$ и $K' << MS_{le*}$, время работы предложенной в диссертации схемы будет значительно меньше времени работы рассмотренной.

В работах [59-61, 95] приведены параметры работы для существующих реализаций фрактального сжатия изображений на ПЛИС и GPU. Большинство публикаций приводят результаты сжатия для изображений в серых тонах, один пиксель которого занимает 8 бит, поэтому при реализации предложенного выше подхода на одном кристалле возможно разместить 4 параллельно работающих конвейера, т.к. входной канал в современных ПЛИС составляет 32 бит. Таким образом, по сравнению с существующими реализациями на ПЛИС мы получим ускорение в 15-40 раз на одном кристалле, и более чем в 50 раз по сравнению с реализациями для GPU. Таким образом, доказано второе положение, выносимое на защиту.

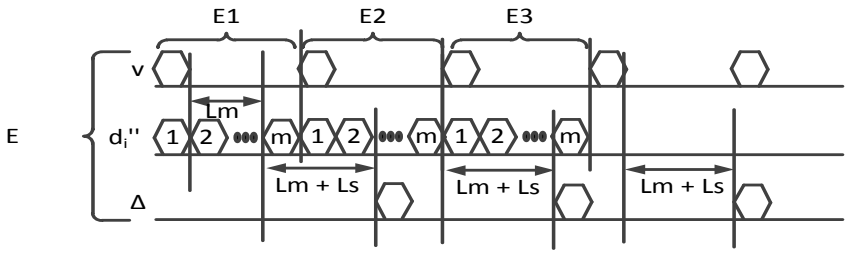
2.3 Реализация фрактального сжатия изображений на РВС с учетом особенностей адаптивного разбиения

Существуют различные способы разбиения изображения на доменные и ранговые блоки: покрытие равномерной квадратной сеткой [96], покрытие правильными многоугольниками [97] и покрытие адаптивными сетками [98], в частности при помощи четверичных деревьев. В зависимости от того, какой способ разбиения исходного изображения на доменные и ранговые блоки применяется, будет изменяться и функционирование вычислительной системы.

При работе с равномерными порциями данных поток результатов из вычислительной структуры также поступает с равномерными интервалами, как показано на рисунке 2.20.



а)



б)

Рис. 2.20 – Временная диаграмма входных данных и результатов при равномерном разбиении изображения а) для блока L, б) для блока E

На временных диаграммах показано, что скважность схемы будет зависеть от размера рангового блока. При этом в существующих реализациях [13-14] выдача результата будет зависеть в первую очередь от размеров доменного блока.

На рисунке 2.21 приведена временная диаграмма реализации фрактального сжатия изображения, описанная в работе [13], где также приведено сравнение предложенной реализации с другими реализациями для ПЛИС, универсальных и графических процессоров.

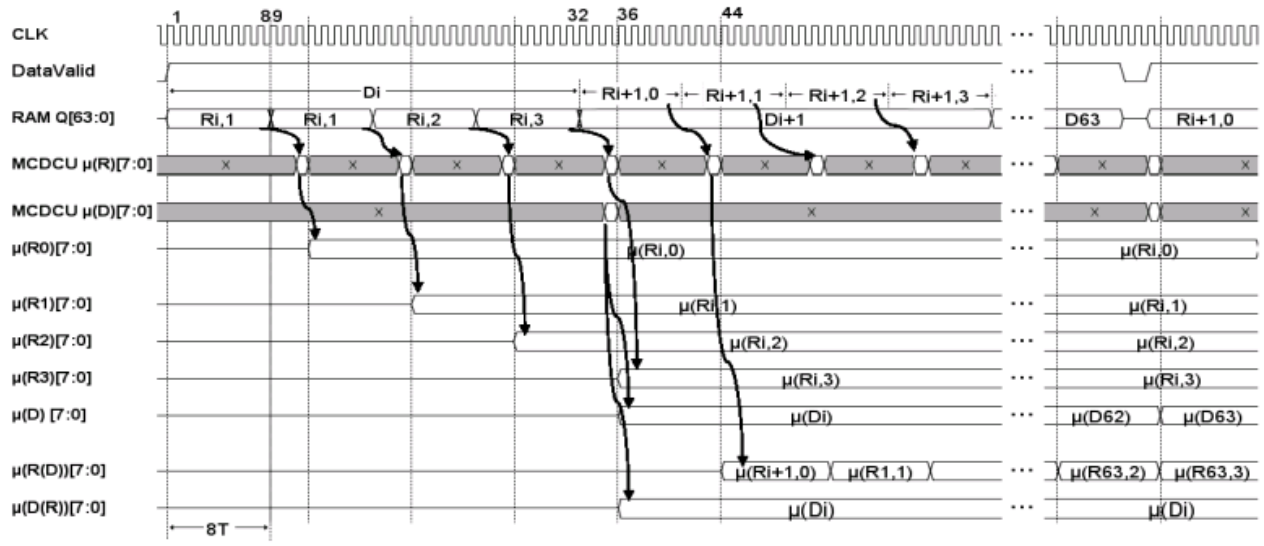


Рис. 2.21 – Временная диаграмма для реализации [13]

На диаграмме выше можно увидеть, что результат усреднения доменного блока появляется раз в 32 такта, и соответственно все результаты работы схемы, где задействованы данные сжатого доменного блока будут выходить с подобной задержкой, что существенно снижает реальную производительность вычислительной системы.

В реализации, предложенной в данной диссертации, эта проблема решается за счет установки k блоков W, работающих параллельно, как описано выше. Благодаря этому вычисления распараллеливаются по принципу "конвейер в конвейере" и результаты усреднения пикселей очередного доменного блока выходят каждый такт. Данный вариант позволяет избежать простоя делителя, который в противном случае не выполняет полезных вычислений k-1 тактов из каждых k, что увеличивает коэффициент использования оборудования.

При покрытии изображения адаптивной сеткой ранговых блоков возможно сократить количество операций для однотонных частей изображений и более

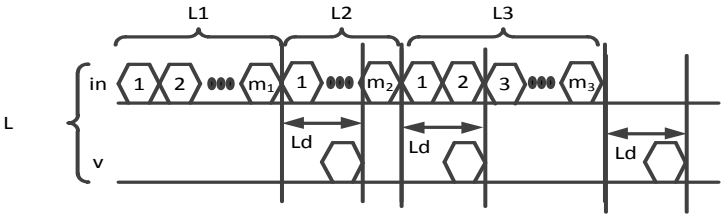
точно закодировать мелкие детали. При использовании адаптивного разбиения значение m в формулах 3.2 и 3.3 становится переменным и уникальным для каждой отдельной группы ранговых блоков.

$$v = \left(\sum_{i=1}^{m} \sum_{j=1}^{m} r_{ij} - u \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} \right) / m^2 , \qquad E(R,D) = \sum_{i=1}^{m} \sum_{j=1}^{m} \left(u d_{ij} + v - r_{ij} \right)^2 ,$$

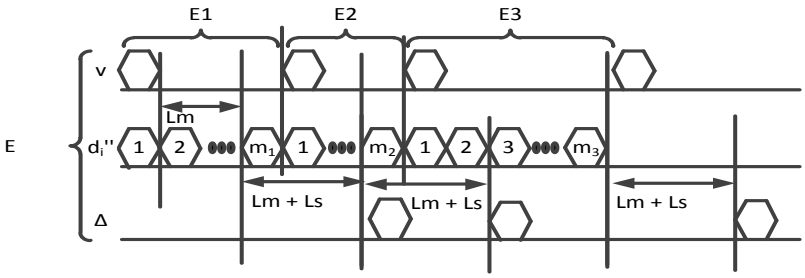
$m \in (a_1, a_2, \ldots, a_q)$ – множество возможных значений, полученных при разбиении изображения.

Покрытие доменными блоками может также быть адаптивным и использовать доменные блоки разных размеров скользящим окном покрывающих неодинаковое количество ранговых блоков для всего изображения.

При реализации фрактального сжатия изображений с адаптивным разбиением рабочей области необходимо учитывать переменные значения в потоке данных. Входной поток данных будет неравномерным и выдача результатов работы вычислительных блоков изменится, как показано на рисунке 2.22.



а)



б)

Рис. 2.22 – Временная диаграмма входных данных и результатов при адаптивном разбиении изображения а) для блока L, б) для блока E

При конвейерной подаче значений пикселей для доменных блоков в конвейер вычислительная структура не будет отличаться от аналогичной для постоянного размера сетки (рис. 2.23), но время работы конвейера изменится соответственно размерам подаваемых порций данных

$$T_{convK} = t_{WF} + t_{LE}N + \sum_{i=0}^{M}\frac{S_i}{K}\tau \approx t_{WF} + t_{LE}N + \frac{M\bar{S}}{K}\tau,$$

здесь $\bar{S}$ - среднее значение для скважности, равное $\bar{S} = \sum_{i=0}^{M}\frac{S_i}{M}$. Скважность потока будет меняться в зависимости от размера порции входных данных в вычислительную структуру.

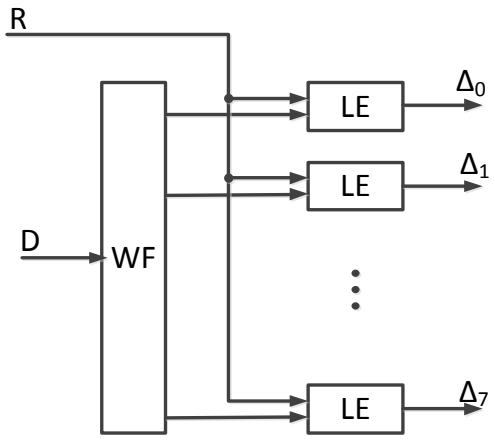


Рис. 2.23 – Выполнение базовых операций сжатия изображения

Произведем сравнение времени работы вычислительной схемы при сжатии изображения, покрытого адаптивными сетками ранговых и доменных блоков, для предложенной в диссертации реализации и существующих [13, 14, 59-61, 95].

В формуле (8) заменим скважность потока S_* средним значением $\bar{S}_*$, чтобы определить оценочное время для работы с адаптивным разбиением. Сравним полученный результат с предложенной реализацией

$$\frac{T_{convK}}{T_*} \approx \frac{t_{LE}N + \frac{M\bar{S}}{K}}{t_{le*}NM\bar{S}_*}\cdot K' \approx \left(\frac{1}{M\bar{S}_*} + \frac{1}{t_{le*}NK}\right)\cdot K'.$$

Поскольку $K' << M\bar{S}_*$ и $K' << NK$ отношение $\frac{T_{convK}}{T_*}$ будет в $\frac{M\bar{S}_* + t_{le*}NK}{M\bar{S}_* t_{le*}NK}$ меньше количества параллельно выполняемых операций K' для существующих реализаций. При сравнении экспериментальных данных получено ускорение в 12-35 раз по сравнению с существующими реализациями на ПЛИС.

2.4 Практическая реализация фрактального сжатия изображений на реконфигурируемом компьютере Терциус-2

Для практической реализации задачи фрактального сжатия изображений на РВС необходимо отталкиваться от имеющихся в наличии вычислительных ресурсов конкретной РВС. Современные РВС обладают большим запасом логических блоков и намного меньшим количеством каналов для ввода и вывода информации. Реконфигурируемый компьютер Терциус-2, описание которого приведено в главе 1, содержит 8 ПЛИС Kintex UltraScale XCKU095, каждая из которых обладает ресурсом в 1176000 LUT, помимо этого Терциус-2 содержит 16 КРП с шириной канала подачи данных 32 бит, т.е. во всем реконфигурируемом компьютере для работы доступно 512 битовых каналов данных.

При разработке вычислительной схемы ее параметры не привязываются к конкретным параметрам РВС для удобства реализации на РВС с любыми параметрами. В случае реализации на Терциус-2 количество каналов подачи данных является критическим ресурсом, поэтому следует определить степень редукции [99] по каналам для задачи фрактального сжатия изображений.

Процедура сжатия проводится над изображениями, представленными в формате цветовой модели RGB. В данном формате каждый пиксель изображения кодируются 24 битами – по восемь бит на каждый цветовой канал. Для удобства вычислений представим, что обрабатываемое изображение квадратное со стороной N пикселей и равномерно разделено на квадратные ранговые блоки со стороной m пикселей, которые в свою очередь покрыты сеткой равномерных

квадратных доменных блоков со стороной n пикселей. В этом случае количество ранговых блоков будет равно N^2/m^2, а доменных – $(N/m - 1)^2$, и для параллельной подачи всех доменных блоков потребовалось бы $24(N/m - 1)^2n^2$ каналов подачи данных. Для минимального стокового изображения со стороной 2000 пикселей и разбиением на квадратные ранговые блоки со стороной 4 пикселя и доменные – 8 пикселей потребовалось бы 95 233 536 каналов, чтобы одновременно подать все изображение. Исходя из выше сказанного для квадратного изображения с равномерным разбиением на квадратные ранговые и доменные блоки коэффициент редукции по каналам можно выразить формулой [100]:

$$K_c = 24\left(N / m - 1\right)^2 n^2 / C,$$

где N^2 – количество пикселей в исходном изображении, n^2 – количество пикселей в доменном блоке, m^2 - количество пикселей в ранговом блоке, C – количество доступных каналов. В случае прямоугольного разбиения все квадраты заменяются произведением длины на ширину, для других видов разбиений размеры ранговых и доменных блоков следует заменить значением их площади, выраженной в пикселях.

Для приведенного выше примера четырехмегапиксельного изображения в формате RGB коэффициент редукции по каналам составит K_c=372006 для реконфигурируемого компьютера Терциус-2, для загрузки данных в вычислительную схему используется половина имеющегося ресурса по каналам.

В рамках рассмотренного выше примера параметров обрабатываемого изображения каждый доменный блок содержит 64 пикселя, каждый из которых содержит 24 бита информации, т.е. весь доменных блок будет содержать 1536 бит. Очевидно, что параллельная подача даже одного доменного блока целиком невозможна. Можно разделить изображение на цветовые каналы и работать с каждым из них отдельно. В таком случае каждый пиксель будет содержать всего восемь бит информации, а весь доменный блок займет 512 бит. В этом случае также невозможна подача всего доменного блока одновременно. Данная проблема решается посредством редукции по разрядности входных данных, т.е.

предлагается выполнять побитовую подачу каждого пикселя последовательно [101]. Если осуществить такую редукцию, то для подачи одного пикселя потребуется только один канал данных, и при имеющемся количестве входных каналов станет возможна одновременная подача четырех доменных блоков.

На первый взгляд не очевидно, но далеко не всегда параллельная подача всех данных, принадлежащих доменному блоку, имеет смысл. Параллельная подача всего доменного блока удобна только в случае фиксированного разбиения изображения на ранговые и доменные блоки. При использовании адаптивного разбиения размер подаваемых порций данных, принадлежащих доменным блокам, может быть переменным, отсюда может возникнуть необходимость динамического выбора ширина канала для входных данных. При использовании последовательной побитовой подачи всего доменного блока отсутствует необходимость закладывать возможность динамической смены ширины канала данных, а значит такой способ будет более универсальным.

При побитовой подаче данных для каждого из доменных блоков возможно осуществлять одновременную загрузку 256 доменных блоков. Также при такой организации данных можно использовать одноразрядные арифметические операции, которые не требуют больших затрат оборудования. Расход вычислительного ресурса для описанных выше операций составит 60 LUT для блока сжатия-поворота WF и 220 LUT для восьми блоков LE, выполняющих вычисление сдвига по яркости и расстояния между ранговым и доменным блоками. На одной ПЛИС Kintex UltraScale XCKU095 с ресурсом 1176000 LUT можно будет разместить до 554 данных вычислительных блоков. Пример реализации задачи фрактального сжатия изображения на одной ПЛИС показан на рисунке 2.24. В этом случае используются 256 каналов подачи данных с побитовой загрузкой пикселей доменных блоков. При таких параметрах в конвейере помещается всего два последовательных блока LE.

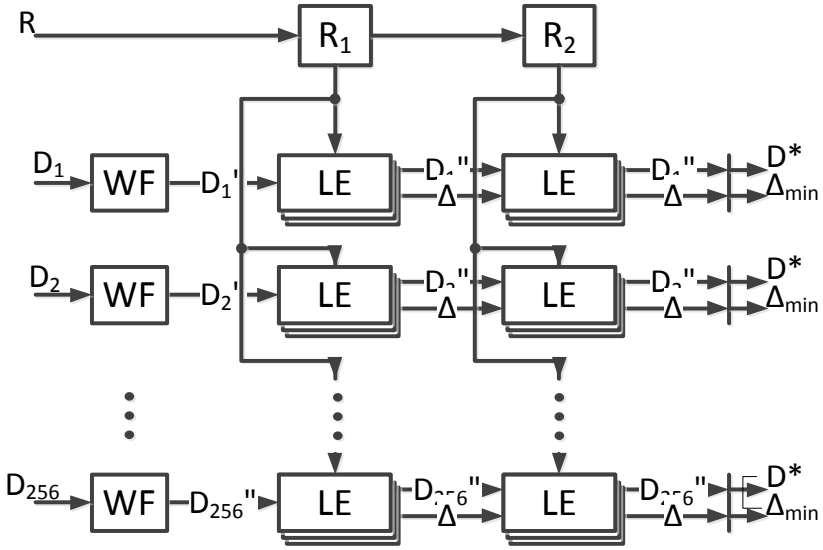


Рис. 2.24 – Реализация фрактального сжатия при использовании 256 каналов данных с побитовой подачей пикселей

Время выполнения фрактального сжатия изображения будет складываться из времени работы каждого блока, а также времени, которое требуется для первичной загрузки данных в вычислительный конвейер - Ld. Обозначим латентность блока сжатия-поворота WF как Lt_{WF}, латентность блока L, вычисляющего сдвиг по яркости – Lt_L, а для блока E, определяющего расстояние между ранговым и доменным блоками, - Lt_E. Поскольку блоки L и E объединены в один, то их суммарная латентность $Lt_{LE}= Lt_L+ Lt_E$. Также для вычисления времени работы понадобятся параметры сжимаемого изображения, такие как количество доменных блоков (N_D) и количество ранговых блоков (N_R), и параметры РВС, на которой будут производиться вычисления – количество доступных каналов (C), количество задействованных ПЛИС (F), и количества блоков LE, помещающихся в конвейер, (Q). Латентность конвейера, обрабатывающего один поток доменных блоков будет равна $Lt_{WF} + QLt_{LE}$, тогда прогон всех доменных блоков займет $\left(Lt_{WF} + QLt_{LE}\right)N_D$ тактов.

Для поиска коэффициентов СИФ, кодирующих сжимаемое изображение, попарно выполняется сравнение всех доменных блоков во всеми ранговыми. Количество одновременно обрабатываемых ранговых блоков равно Q количеству

установленных блоков LE. Для этого сравнения понадобится $\left(Lt_{WF}+QLt_{LE}\right)N_D\frac{N_R}{Q}$ тактов. При масштабировании данной вычислительной схемы на F ПЛИС и распараллеливании по C каналам получим следующее время работы:

$$T=\frac{t}{U}\left(Ld+\frac{\left(Lt_{WF}+QLt_{LE}\right)N_D\frac{N_R}{Q}}{CF}\right),$$

где t – величина обратная частоте работы ПЛИС, U – коэффициент использования оборудования, равный 3/4.

Для четырехмегапиксельного изображения, покрытого сетью ранговых и доменных блоков как описано выше [102], время сжатия на реконфигурируемом компьютере Терциус-2 составит 1,5 с.

Время выполнения сжатия для изображения с теми же параметрами на универсальном четырехъядерном процессоре Intel Core i7-4770 с тактовой частотой 3,50 GHz составит 23000 секунд [12]. Таким образом, реализация фрактального сжатия изображений на РВС показывает ускорение в 15000 раз по сравнению с аналогичной реализацией для многоядерного универсального процессора.

При проведении сравнения предложенного подхода с существующими реализациями фрактального сжатия [12-14] для ускорителей на основе одной ПЛИС увеличение реальной производительности составит 15-40 раз при тех же параметрах.

2.5 Методы параллельной реализации декомпрессии сжатого изображения

Декомпрессия изображения, сжатого фрактальным способом, является методом восстановления оригинального изображения из полученных ранее коэффициентов системы итерируемых функций. Этот процесс осуществляется

итерационно, применяя найденные коэффициенты к произвольному начальному изображению. Важно заметить, что начальное изображение может быть любым. Согласно теореме о сжимающем отображении [103], итерационный процесс будет сходиться к неподвижному изображению. Таким образом, декомпрессия изображения может быть применена для восстановления оригинального изображения, сохраняя тем самым качество и детали изображения. Ниже приведен алгоритм восстановления изображения, сжатого фрактальным способом:

1. Создаются два произвольных изображения одинакового размера A и B.
2. Сетка ранговых блоков, идентичная покрывающей сжатое изображение, накладывается на изображение A. На изображение B накладывается сетка доменных блоков, также идентичная сетке, покрывающей исходное изображение.
3. Над каждым доменным блоком, определенным в изображении B, производятся аффинные преобразования согласно полученным при сжатии коэффициентам СИФ для содержащихся в нем ранговых блоков, хранящихся в изображении A. Полученный результат записывается в B. После перебора всех доменных блоков получается новое изображение.
4. Пиксели из изображения B переносятся в изображение A.
5. Пункты 3 и 4 повторяются до тех пор, пока изображения A и B не станут одинаковыми.

Для дальнейшей работы следует более подробно раскрыть пункт 3 приведенного выше алгоритма декомпрессии:

1. Для каждого рангового блока восстанавливаемого необходимо найти соответствующий ему доменный в текущем изображении.
2. Выполнить усреднение пикселей доменного блока до размеров рангового.
3. Выполнить поворот доменного блока соответственно заданному коэффициенту СИФ.

4. Умножить значения всех пикселей преобразованного доменного блока на коэффициент сжатия яркости и добавить к получившемуся произведению сдвиг по яркости.
5. Перенести преобразованные пиксели в соответствующий ранговый блок.

Входными данными для работы процедуры по восстановлению сжатого изображения является структура для хранения результата X, описанная в пункте 2.3. Она содержит как коэффициенты СИФ, так и номера ранговых и доменных блоков, для которых установлено соответствие. Как видно из приведенного выше алгоритма декомпрессии для работы необходимо выделить области память, где будет итерационно производиться восстановление изображения.

Изначально входные данные, содержащиеся в структуре X, не упорядочены, и может возникнуть проблема обращения к требуемой области памяти устройства, а также проблема разметки рабочей области ранговыми и доменными блоками. В таких случаях принято использовать косвенную адресацию, подробное описание которой приведено в пункте 1.6. При выполнении декомпрессии изображения адреса используемых ранговых и доменных вычисляются исходя из их номеров, которые загружаются вместе с коэффициентами СИФ [104].

С целью экономии памяти устройства принято использовать только одну рабочую область для итерационного восстановления изображения [92]. Подобный подход не может быть реализован на РВС, т.к. неизбежно возникновение коллизий при одновременных запросах записи в память и чтения из нее. В таком случае невозможно добиться непрерывных потоков данных, или как-то сократить количество разрывов, т.к. разрывы в потоках данных будут необходимы для осуществления записи в память результата работы (рис. 2.25)

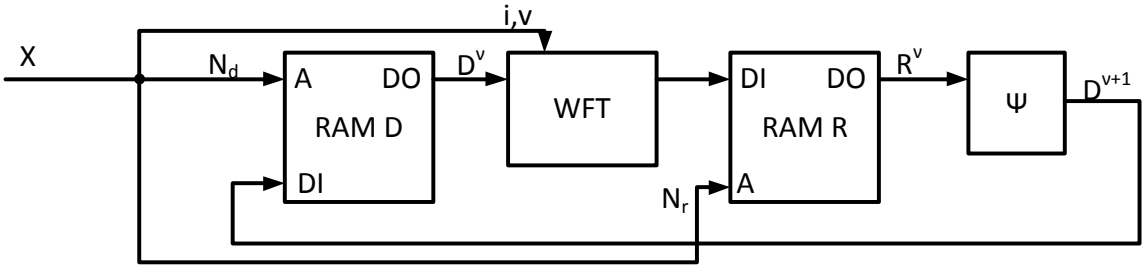


Рис. 2.25 – Тривиальная реализация процедуры декомпрессии изображения

Блок ψ выполняет выбор доменного блока в зависимости от соответствующего ему рангового $D^v = \Psi\left(X, R^v\right)$.

В этом случае время работы вычислительной структуры составит

$$T_{dec} = \left(t_{WFT} + t_{\psi}\right) N\tau,$$

здесь t_{WFT} и t_{ψ} – время работы блоков WFT и ψ соответственно, N – количество доменных блоков, которые должны пройти через вычислительную структуру.

При тривиальной реализации декомпрессии изображения блоки памяти, содержащие доменные RAM D и ранговые RAM R блоки, должны выступать как приемник и как источник данных. Из памяти RAM D данные, содержащиеся в доменных блоках, поступают в блок выполнения аффинных преобразования WFT (рис. 2.26).

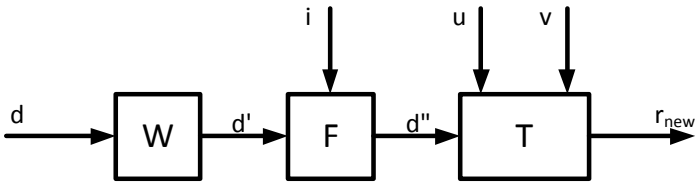


Рис. 2.26 – Элементы блока WFT

Блок W выполняет операцию сжатия доменного блока до размеров рангового, блок F выполняет ротацию доменного блока согласно полученному на входе номеру. Подробные описания работы этих блоков приведены в пункте 2.2. Блок T выполняет обратный сдвиг по яркости (рис. 2.27), на вход подаются значения усредненных пикселей сжатого доменного блока, коэффициент сжатия яркости u и значение сдвига по яркости v, на выходе получаются новые значения

для пикселей восстанавливаемого рангового блока, которые будут записаны в RAM R.

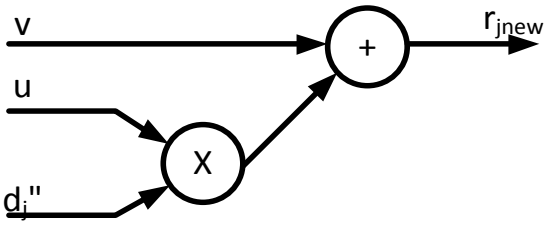


Рис. 2.27– Вычислительная структура блока T

Время работы для приведенной выше вычислительной структуры блока T составит

$$t_T = \left(L_s + L_m\right)\tau$$

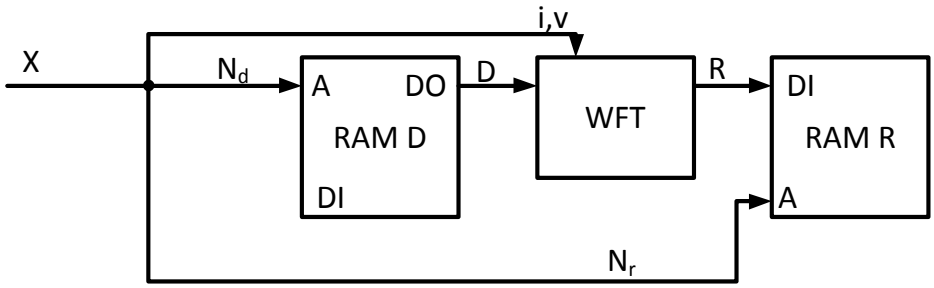
RAM R является источником данных для получения элементов доменных блоков $D^{\nu+1}$ нового итерационного слоя ν+1.

Вычислительная структура с обратной связью неизбежно повлечет за собой увеличение скважности и как следствие разрывы потока операндов, а также падение реальной производительности. Использование косвенной адресации приводит к невозможности параллельного доступа к памяти, из-за чего невозможно также и распараллеливание всей вычислительной структуры.

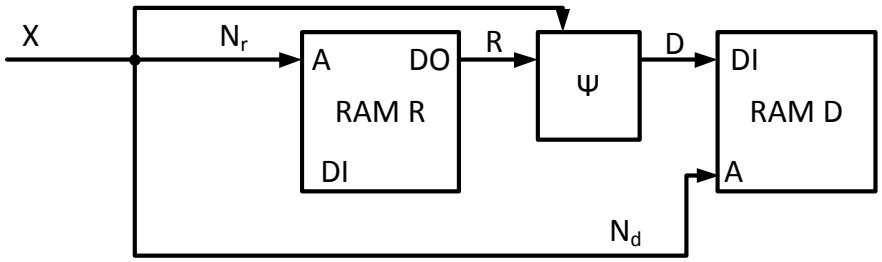
Для такой структуры характерна высокая скважность порядка нескольких сотен тактов, и скорость обработки данных снижается в соответствующее количество раз. Согласно принципам эффективной реализации фрактальных алгоритмов для РВС следует раскрыть обратную связь для снижения скважности потока данных.

Для этого можно разделить процедуру декомпрессии изображения на два кадра. В первом будет происходить поиск значений рангового блока на основе полученных значений доменного $R^{\nu} = F\left(X, D^{\nu-1}\right)$, а во втором выполняться сбор и обновление доменных блоков на основе новых значений ранговых

$D^{\nu} = \Psi\left(X, R^{\nu}\right)$. В полученной двухкадровой структуре блоки памяти RAM R и RAM D выполняют либо роль приемника данных, либо роль источника (рис.2.28)



а)



б)

Рис. 2.28– Вычислительная структура с двумя проходами: а) прямой ход – поиск элементов ранговых блоков, б) обратный ход – перенос элементов ранговых блоков в доменные

Время работы вычислительной структуры для прямого хода (рис. 2.28 а) составит $T_{WFT} = (2\mathrm{L_s} + \mathrm{L_d} + \mathrm{L_m} + \mathrm{Nn})\tau$, где n – число пикселей в доменном блоке на входе. Для обратного хода (рис. 2.8 б)) время работы составит $T_{\psi} = (2\mathrm{L_{read}} + \mathrm{Nn})\tau$, здесь $\mathrm{L_{read}}$ – время, требуемое для записи данных во внутреннюю память.

На вход вычислительной структуры, показанной на рисунке 2.28, поступают данные, хранящиеся в X. Номера доменных блоков $\mathrm{N_d}$ поступают на вход адреса чтения памяти RAM D, требуемые доменные блоки вычитываются и подаются в блок WFT, также в него приходят коэффициенты СИФ (i,v). После преобразования данные записываются в память RAM R по адресу, сформированному согласно поступившему на вход номеру рангового блока $\mathrm{N_r}$.

Однако наличие двойной косвенной адресации мешает эффективному распараллеливанию по данным описанной выше структуры из-за неизбежности одновременных запросов на чтение и запись к одному каналу памяти. Эту проблему можно решить, если, следуя принципам эффективной обработки самоподобных структур на РВС, выполнить предварительную подготовку данных. Если отсортировать элементы структуры X = (N_r, N_d, i, v) по номерам ранговых блоков N_r, то набор из четырех параметров уменьшится до трех $X_R = (N_d, i, v)$. Эта модификация позволит перейти от двойной косвенной адресации к одинарной и создает возможность улучшить распараллеливание по данным. Вычислительная структура, модифицированная для использования X_R, показана на рисунке 2.29.

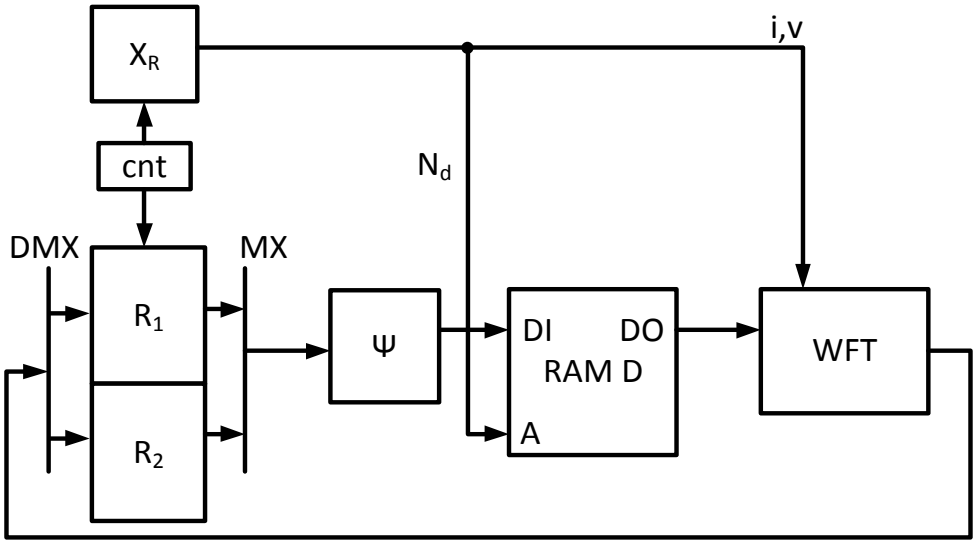


Рис. 2.29 – Использование упорядоченных значений X_R

Номера элементов X_R соответствуют номерам ранговых блоков, которые хранятся во внешней памяти. Аналогично предыдущему варианту адрес чтения из памяти RAM D формируется за счет подаваемого номера доменного блока N_D. Пиксели доменного блока обрабатываются в блоке WFT, и преобразованные данные записываются во внешнюю память, обновляя хранящиеся там ранговые блоки. Двухпортовая память дает возможность осуществлять чтение и запись

независимо. Данные обновленных ранговых блоков после установления соответствия в блоке Ψ записываются в RAM D.

Время работы оптимизированной таким образом системы составит

$$T_{opt} = (2\mathrm{L_s} + \mathrm{L_d} + \mathrm{L_m} + 2\mathrm{L}_{read} + \mathrm{Nn})\tau .$$

Таким образом выигрыш по сравнению с предыдущим вариантом составит величину Nnτ, т.е. время работы схемы сократится практически вдвое.

Возможно несколько способов сортировки элементов структуры X. Самым простым из них является предварительная сортировка данных на ПК, и подача уже отсортированного массива в РВС. Этот способ может подойти только для небольших массивов данных, иначе предварительная подготовка может занять больше времени, чем вычисления, поэтому наилучшим вариантом будет создание уже отсортированной структуры X_R во время сжатия изображения.

В пункте 2.2 приведено описание вычислительного блока, устанавливающего соответствие между парой ранговый-доменный блок и коэффицентами СИФ, полученными после преобразований, которые хранились в структуре X. Элементы результирующей структуры не были специально упорядочены, а располагались в порядке своего естественного появления.

Ранговые блоки загружаются во внутреннюю память по порядку, поэтому возможно привязать номер рангового блока к номеру каждого элемента структуры X (рис. 2.30).

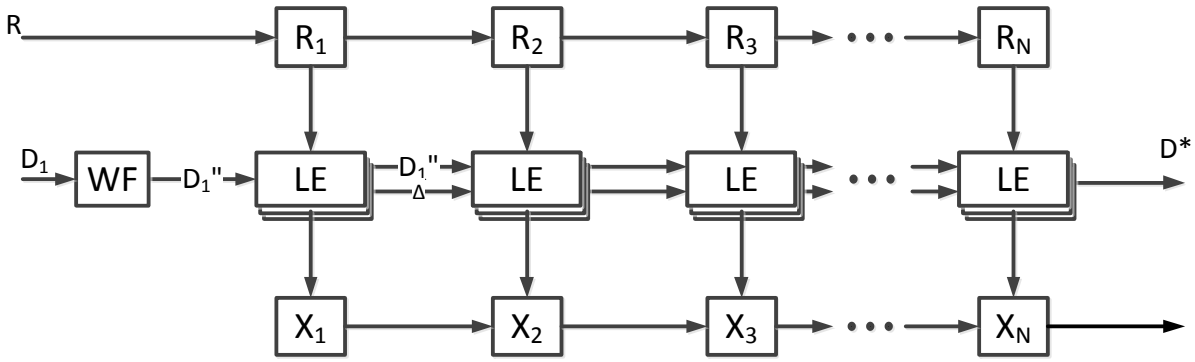


Рис. 2.30– Упорядочивание значений в структуре X_R согласно номерам ранговых блоков

При формировании результирующих данных для кодирования изображения элементы X_R записываются в отведенные для них регистры.

Как описано в пункте 2.4, при работе с реконфигурируемым компьютером Терциус-2 можно организовать до 256 потоков входных данных при побитовой подаче. Для работы с памятью RAM D необходимо осуществлять буферизацию подаваемых параллельно потоков данных. Пример вычислительной структуры с дополнительными буферами и коммутаторами данных приведен на рисунке 2.31.

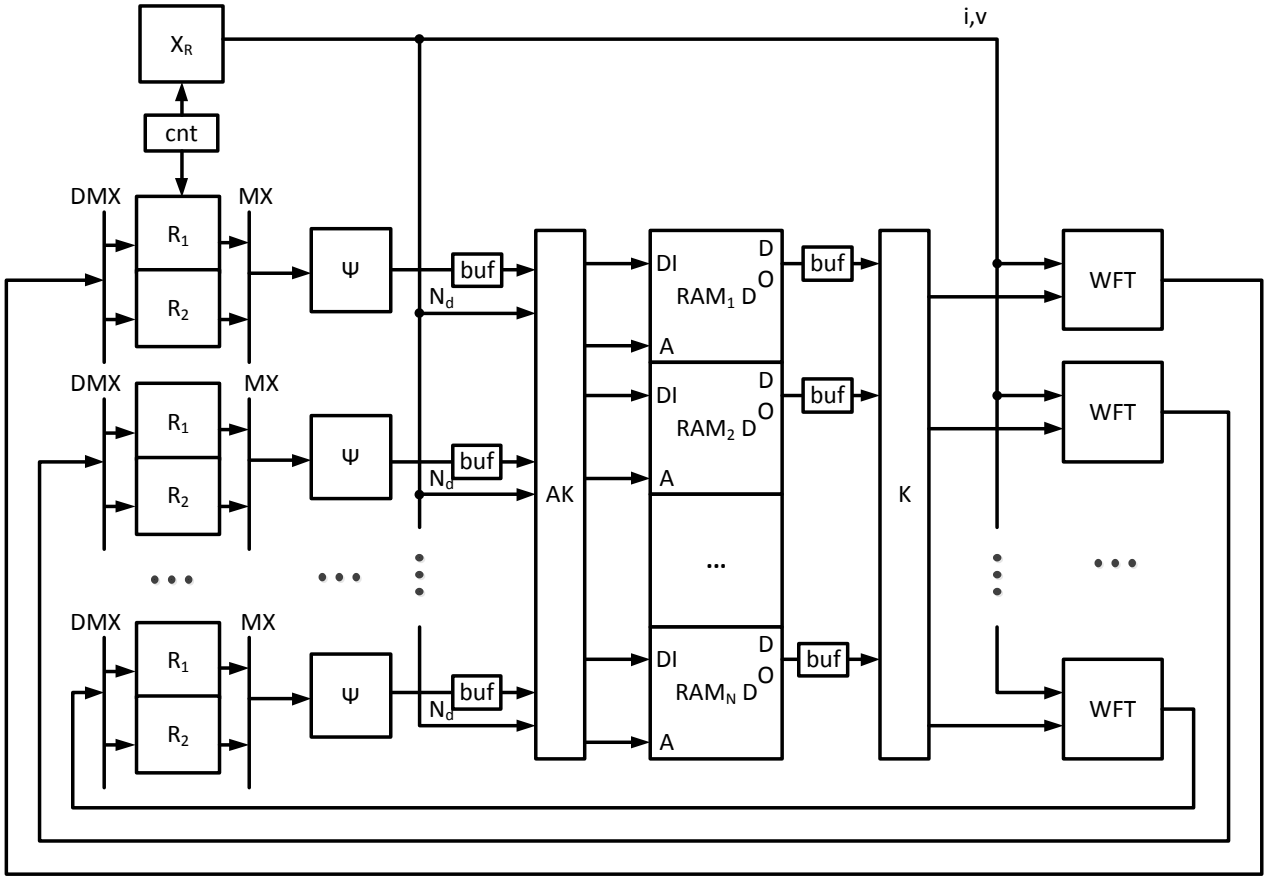


Рис. 2.31– Распараллеливание процедуры декомпрессии изображения

Аналогично предыдущему варианту номера доменных блоков N_d формируют адреса доменных блоков, которые обрабатываются в блоках WFT [94]. После обработки данные записываются в память ранговых блоков, после этого порядок чтения и записи меняется, и происходит выгрузка данных ранговых блоков и обновление доменных.

Время работы распараллеленной таким образом структуры составит

$$T = \left(t_{WFT} + t_{\psi}\right)\frac{Nn}{2K},$$

где K – количество параллельно работающих вычислительных слоев.

Процесс декомпрессии изображения после фрактального сжатия является задачей, с трудом поддающейся распараллеливанию на РВС. Описанный в данном параграфе подход не является оптимальным для данной задачи, и поиск наилучшего решения может стать предметом для дальнейших исследований.

Время выполнения декомпрессии четырехмегапиксельного изображения, представленного в формате RGB, можно вычислить по формуле:

$$T = 1{,}3\tau\left(Ld + \frac{3LtN_R I}{KF}\right),$$

где τ – величина обратная частоте работы ПЛИС, Ld – количество тактов, необходимое для загрузки одной порции входных данных, N_R – количество ранговых блоков, , Lt – латентность вычислительной схемы, I – количество итераций, требуемых для восстановления изображения, K – количество каналов для подачи данных, F – количество задействованных ПЛИС.

Для изображения, разбитого на 250 000 квадратных ранговых блоков, время работы на РВС Терциус-2 составит примерно 0,003 секунды. Тогда как время выполнения этой же задачи на универсальном четырехядерном процессоре Intel Core i7-4770 составит примерно 1,15 секунд [105]. Таким образом, реализация декомпрессии изображения на РВС показывает ускорение в 380 раз по сравнению с аналогичной реализацией для многоядерного универсального процессора.

2.6 Выводы

1) Разработан метод создания параллельно-конвейерных программ фрактального сжатия изображений для РВС с использованием распараллеливания по слоям и итерациям, отличающийся от известных побитовой обработкой

данных, обеспечивающей максимальное использование вычислительного ресурса системы, и сортировкой структур, выходных данных, по номеру рангового блока.

2) Разработан метод создания параллельно-конвейерных программ декомпрессии изображений для РВС, сжатых фрактальным способом, отличающийся от известных использованием только одной косвенной адресации для блоков памяти, содержащих доменные блоки.

3) Введена новая структура данных, позволяющая однозначно устанавливать соответствие между доменным и ранговым блоками и хранить полученные для них коэффициенты системы итерируемых функций (СИФ). Такой способ хранения данных имеет общие черты с множествами, но в отличие от них более универсален. Подобная упаковка коэффициентов СИФ вместе с номерами соответствующих им ранговых и доменных блоков облегчит последующую распаковку сжатого изображения.

4) Выполнены оценки времени работы для вычислительных структур, реализующих фрактальное сжатие и декомпрессию изображений, с учетом равномерного и адаптивного покрытий исходного изображения ранговыми и доменными блоками. Проведен сравнительный анализ разработанной параллельно-конвейерной программы с существующими реализациями приложений для фрактального сжатия изображений на ПЛИС, графических ускорителях и универсальных процессорах. Сравнение времени решения с другими существующими реализациями показало повышение реальной производительности в 10-40 раз для реализаций на ПЛИС и более чем в 50 раз по сравнению с графическими ускорителями, что было подтверждено экспериментально на образцах РВС «Терциус-2».

3 МЕТОД РЕШЕНИЯ ЗАДАЧИ АНОМАЛЬНОЙ ДИФФУЗИИ НА РВС

В точных и естественных науках возникают задачи, требующие моделирования сложных систем различного характера для их решения. К сожалению, для описания этих систем не всегда хватает инструментария, предоставляемого евклидовой геометрией, где любая даже самая сложная линия или поверхность при некотором приближении может быть заменена отрезком прямой. Фракталы же позволяют моделировать сложные структуры в неупорядоченных средах и процессы, протекающие в них. Фракталы могут быть использованы для моделирования пористых тел, таких как поровое пространство, скелет породы, поверхность скелета породы и т.д. Для примера, пористые тела могут быть использованы в медицине для имитации костной ткани. Фракталы могут использоваться для моделирования формы кости, а также для имитации ее внутренней структуры. Это может помочь в создании более точных моделей для исследования заболеваний костей и разработки новых методов лечения. Также фракталы могут быть полезны при моделировании других сложных систем, таких как погодные условия или экономические процессы. Их применение может помочь получить более точные результаты и прогнозы, что может иметь важное значение для многих областей науки и промышленности.

3.1 Аномальная диффузия. Суб- и супердиффузия

В сплошной среде средний квадрат расстояния, на которое удаляется от исходной точки случайно блуждающая частица, пропорционален времени [38]. Во фрактальной среде случайно блуждающая частица будет удаляться от места старта медленнее, т. к. не все направления для нее доступны. Средний квадрат расстояния для фрактальной среды пропорционален некоторой дробной степени времени, показатель которой связан с размерностью Хаусдорфа-Безиковича для соответствующего среде фрактала. Множество препятствий (узких мест, крутых

поворотов и тупиков) затрудняют продвижение частиц и замедляют диффузию, следовательно картина распространения вещества во фрактальной среде не соответствует аналогичному процессу для сплошной среды. Замедление диффузии в неупорядоченных средах столь существенно, что она перестает удовлетворять классическому закону Фика [38], и для описания этого процесса используется аномальная диффузия.

Аномальная диффузия описывает процессы переноса на стационарных процессах, характеризующиеся нестационарным распределением частиц в пространстве, где квадрат расстояния r, которое прошла частица за время t, пропорционален $r^2 \sim Dt^{\chi}$, где $\chi = 1$ для классической диффузии и $\chi \neq 1$ - для аномальной [106]. Если параметр χ больше единицы, то такой случай называют супердиффузией (ускоренное блуждание частиц), при χ меньше единицы говорят о субдиффузии (замедленное блуждание частиц). Для описания этого явления используется дифференциальное уравнение дробного порядка, при этом порядок дробной производной связан с размерностью фрактала. К процессам супердиффузии относится перенос техногенных (захоронение отходов) и естественных (радон) радионуклидов, миграция природного газа по трещинам в горных породах и другие фрактальные процессы [107, 108]. Обычно такие процессы описываются дробными производными по пространственным координатам. К процессам субдиффузии относится перколяция (просачивание), диффузия в дефектных средах с временным удержанием, диффузия в пористых средах с прилипанием и др. Субдиффузия обычно описывается уравнениями с дробными степенями по времени.

Математический аппарат, используемый для описания субдиффузии, можно рассмотреть на примере моделирования просачивания воды в почве. В данной работе для моделирования процессов фильтрации во фрактальных средах используется уравнение Букингема-Ричардса [31].

$$C \cdot D^{\chi}_{[0;t]} h = \left(k(h) h_x'\right)_x' + \left(k(h) h_y'\right)_y' + \tilde{Q}, \qquad (3.1)$$

здесь x, y – горизонтальные координаты, θ – влажность, h – потенциал почвенной влаги, $k(h)$ - коэффициент влагопроводности, z_0 – толщина распределения источников и потерь, F – источники (приток, осадки и др.), Q – потери (сток, испарение и др.), $D^{\chi}_{[0;t]} f(t) = \frac{1}{\Gamma(1-q)} \int_0^t \frac{f_\tau(\tau) d\tau}{(t-\tau)^{\chi}}$ – регуляризованная дробная производная Римана-Луивилля порядка χ ($0 < \chi < 1$) [109, 110].

Зависимость коэффициента влагопроводности $k(h)$ от потенциала h имеет вид:

$$\begin{cases} k(h) = k_f e^{\alpha h}, & h < 0 \\ k(h) = k_f, & h \geq 0, \end{cases}$$

где k_f - коэффициент фильтрации, α - эмпирический параметр.

Для влажности θ будет справедлива следующая зависимость от потенциала h:

$$\begin{cases} \theta(h) = (\theta_s - \theta_r) e^{\alpha h} + \theta_r, & h < 0 \\ \theta(h) = \theta_s, & h \geq 0, \end{cases}$$

где θ_s - максимальная влажность, θ_r - минимальная влажность.

Дополним уравнение (3.1) граничными условиями, для этого воспользуемся законом Дарси:

$$v = -k\,\mathrm{grad}(h), \qquad (3.2)$$

где v - скорость фильтрации флюида.

В случае непроницаемой стенки должно выполняться требование отсутствия потока флюида через эту границу $v \cdot \mathbf{n} = 0$. Это требование в совокупности с законом Дарси (3.2) дает следующее граничное условие: $\frac{\partial h}{\partial \mathbf{n}} = 0$.

В случае если через границу задан поток жидкости S, то, исходя из тех же соображений, будем иметь следующее граничное условие: $\frac{\partial h}{\partial \mathbf{n}} = S$.

Во многих случаях логично предположить, что поток жидкости зависит от

потенциала линейно, тогда получим следующее граничное условие: $\frac{\partial h}{\partial \mathbf{n}} = \alpha h + \beta$.

Для дискретизации описанной выше математической модели введем равномерную прямоугольную сетку, покрывающую расчетную область, с шагами h_x, h_y по соответствующим трем пространственным направлениям:

$$\bar{\omega} = \left\{ \left(x_i, y_j \right), x_i = i \cdot h_x, y_j = j \cdot h_y, i = 1..n_1, j = 1..n_2 \right\},$$

где $h_x = \frac{l_x}{n_1}$, $h_y = \frac{l_y}{n_2}$.

Обозначим внутренние узлы сетки через ω:

$$\omega = \left\{ \left(x_i, y_j \right), x_i = i \cdot h_x, y_j = j \cdot h_y, \; i = 1..n_1 - 1, \; j = 1..n_2 - 1 \right\}.$$

Граница сетки γ задается соотношением:

$$\gamma = \left\{ \bar{\omega} \setminus \omega \right\}.$$

Для аппроксимации уравнения (3.1) дробную производную $D^{\chi}_{[-\infty;t]} h$ заменяют дискретным аналогом [111]:

$$D^{\chi}_{[-\infty;t]} h \cong \frac{1}{\Gamma\left(2-\chi\right)} \sum_{s=1}^{n} \left(t_{n-s+1}^{1-\chi} - t_{n-s}^{1-\chi} \right) h_{t,i,j}^{s},$$

Частные производные в уравнении (3.1) заменим на их конечно-разностные аналоги, полученные при помощи интегро-интерполяционного метода. Для удобства полученную разностную схему лучше представить в каноническом виде для сеточных уравнений [112]:

$$h_{i,j}^{n+1} \left(\frac{C_{i,j}}{\Gamma(2-\chi)\tau^{\chi}} + \frac{k_{i+\frac{1}{2},j}}{h_x^2} + \frac{k_{i-\frac{1}{2},j}}{h_x^2} + \frac{k_{i,j+\frac{1}{2}}}{h_y^2} + \frac{k_{i,j-\frac{1}{2}}}{h_y^2} \right) -$$

$$- h_{i+1,j}^{n+1} \frac{k_{i+\frac{1}{2},j}}{h_x^2} - h_{i-1,j}^{n+1} \frac{k_{i-\frac{1}{2},j}}{h_x^2} - h_{i,j+1}^{n+1} \frac{k_{i,j+\frac{1}{2}}}{h_y^2} - h_{i,j-1}^{n+1} \frac{k_{i,j-\frac{1}{2}}}{h_y^2} = C_{i,j} \frac{2-2^{1-\chi}}{\Gamma\left(2-\chi\right)\tau^{\chi}} h_{i,j}^{n} +$$

$$+ \frac{C_{i,j}}{\Gamma\left(2-\chi\right)\tau} \left[\left(t_{n+1}^{1-\chi} - t_{n}^{1-\chi} \right) h_{i,j}^{0} + \left(-t_{n+1}^{1-\chi} + 2t_{n}^{1-\chi} - t_{n-1}^{1-\chi} \right) h_{i,j}^{1} + \right.$$

$$\left. + \ldots + \left(-t_{3}^{1-\chi} + 2t_{2}^{1-\chi} - t_{1}^{1-\chi} \right) h_{i,j}^{n-1} \right] + k_{i,j} + Q_{i,j}$$

Здесь $k_{i+\frac{1}{2},j} = \frac{k_{ij}+k_{i+1j}}{2}$ и $k_{i-\frac{1}{2}j} = \frac{k_{i-1\,j}+k_{ij}}{2}$, аналогичное справедливо для вертикальной координаты.

Несложно увидеть, что при дискретизации непрерывной математической модели, описывающей явление субдиффузии и содержащей дробную производную только по времени, получается разностная схема, соответствующая СЛАУ с пятидиагональной матрицей [113]. Методы решения таких СЛАУ для реконфигурируемых вычислительных систем приведены в работах [114, 115].

В работах [32, 116] было установлено экспериментально, что модель переноса радона в неоднородной среде, использующая супердиффузию, а не классическую, наиболее точно отражает реальные данные. Этот газ, который является радиоактивным, может повышать концентрацию в воздухе при сейсмической активности земной коры, что является одним из признаков приближающихся землетрясений. Из-за этого возникает необходимость улучшения точности и скорости моделирования распространения радона. Для решения этой задачи в реальном времени критически важно создать эффективную параллельную реализацию. Она должна быть готовой к обработке больших объемов данных и быстрой реакции на изменения в окружающей среде. Таким образом, разработка эффективной параллельной реализации является необходимым условием для точного и быстрого моделирования распространения радона в реальном времени.

Для описания процессов супердиффузии используют дифференциальные уравнения в частных производных дробных порядков по пространственным координатам. Рассмотрим соответствующий математический аппарат на примере модели переноса радона во фрактальной пористой среде:

$$u_t' = C \cdot D^{\chi}_{[0;x]} u + C \cdot D^{\chi}_{[0;y]} u - \lambda u + Q. \qquad (3.3)$$

Здесь C – коэффициент диффузии радона в грунте; Q – объемный источник радона, численно равный концентрации радона, образующегося в единицу времени в единице объема пористой среды на заданной глубине u – поровая

активность радона в единицу объема порового пространства; η – пористость среды; λ – постоянная распада радона. Дробный дифференциальный оператор $D^{\chi}_{[0;x]}$ соответствует правилу:

$$D^{\chi}_{[0;x]}u(x)=\frac{1}{\Gamma(2-\chi)}\frac{d^2}{dx^2}\int_0^x\frac{u(\xi)d\xi}{(x-\xi)^{\chi-1}},\ 1<\chi<2. \qquad (3.4)$$

Выражение (3.4) можно дискретизировать на равномерной сетке ω следующим способом [117]:

$$I_m=\frac{1}{\Gamma(2-\chi)}\int_{x_1}^{x_m}\frac{f(\xi)d\xi}{(x-\xi)^{\chi-1}}=\sum_{k=2}^{m}\int_{x_{k-1}}^{x_k}\frac{(x_m-\xi)^{\chi-1}}{\Gamma(2-\chi)}\left[u_{k-1}\frac{(\xi-x_k)}{h_x}+u_k\frac{(\xi-x_{k-1})}{h_x}\right]d\xi= \qquad (3.5)$$
$$=\sum_{k=2}^{m}\left(u_{k-1}\frac{(x_kP_{mk}-Q_{mk})}{h_x}+u_k\frac{(Q_{mk}-x_{k-1}P_{mk})}{h_x}\right).$$

Здесь

$$P_{mk}=\frac{(x_m-x_{k-1})^{2-\chi}-(x_m-x_k)^{2-\chi}}{h_x\Gamma(3-\chi)},$$

$$Q_{mk}=\frac{x_{k-1}(x_m-x_{k-1})^{2-\chi}-x_k(x_m-x_k)^{2-\chi}}{h_x\Gamma(3-\chi)}+\frac{(x_m-x_{k-1})^{3-\chi}-(x_m-x_k)^{3-\chi}}{h_x\Gamma(4-\chi)}.$$

Здесь x_k - координата сетки, равная $x_k=i\cdot h_x$ при $i=k$.

Вторая производная в выражении (3.4) заменяется центральной разностью второго порядка:

$$\frac{1}{\Gamma(2-\chi)}\frac{d^2}{dx^2}\int_0^x\frac{u(\xi)d\xi}{(x-\xi)^{\chi-1}}=\frac{I_{m-1}-2I_m+I_{m+1}}{h_x^2}.$$

Предполагается, что в выражение (3.5) записывается на верхнем временном слое. Применяя аналогичные рассуждения к дробной производной по второй пространственной координате, получим следующую разностную схему:

$$\frac{u^{n+1}-u^n}{\tau}=C_{ij}\left(\frac{I^{n+1}_{i-1j}-2I^{n+1}_{ij}+I^{n+1}_{i+1j}}{h_x^2}+\frac{I^{n+1}_{ij}-2I^{n+1}_{ij}+I^{n+1}_{ij+1}}{h_y^2}\right)-\lambda_{ij}u^n_{ij}+Q_{ij}. \qquad (3.6)$$

Согласно выражению (3.5) представим I_{ij} для производной по горизонтальной координате следующим образом:

$$I_{ij} = \sum_{k=2}^{i} \left(V_{ik} u_{k-1\,j} + W_{ik} u_{kj} \right), \; V_{ik} = \frac{\left(x_k P_{ik} - Q_{ik} \right)}{h_x}, \; W_{ik} \frac{\left(Q_{ik} - x_{k-1} P_{ik} \right)}{h_x}. \qquad (3.7)$$

Для вертикальной координаты также справедливы предыдущие рассуждения. С учетом (3.7) перепишем выражение (3.6):

$$\frac{u^{n+1} - u^n}{\tau} = C_{ij} \frac{\sum_{k=2}^{i-1} \left(V_{ik} u_{k-1\,j}^{n+1} + W_{ik} u_{kj}^{n+1} \right) - 2 \sum_{k=2}^{i} \left(V_{ik} u_{k-1\,j}^{n+1} + W_{ik} u_{kj}^{n+1} \right) + \sum_{k=2}^{i+1} \left(V_{ik} u_{k-1\,j}^{n+1} + W_{ik} u_{kj}^{n+1} \right)}{h_x^2} +$$

$$+ C_{ij} \frac{\sum_{k=2}^{j-1} \left(V_{jk} u_{ik-1}^{n+1} + W_{jk} u_{jk}^{n+1} \right) - 2 \sum_{k=2}^{j} \left(V_{jk} u_{ik-1}^{n+1} + W_{jk} u_{jk}^{n+1} \right) + \sum_{k=2}^{j+1} \left(V_{jk} u_{ik-1}^{n+1} + W_{jk} u_{jk}^{n+1} \right)}{h_y^2} - \lambda_{ij} u_{ij}^n + Q_{ij}$$

Полученной разностной схеме будет соответствовать СЛАУ специального вида, представленной на рисунке:

Рис. 3.1 – Вид матрицы для полученной разностной схемы

При увеличении расчетной сетки будет увеличиваться количество уравнений в СЛАУ, и будет расти количество заполненных нижних диагоналей в матрице.

Несложно увидеть, что дискретные аналоги производных содержат в себе повторяющиеся члены, которые можно сократить и привести СЛАУ к диагональному виду.

$$\frac{u^{n+1}-u^n}{\tau}=C_{ij}\frac{V_{ii+1}u_{ij}^{n+1}+W_{ii+1}u_{i+1j}^{n+1}-V_{ii}u_{i-1j}^{n+1}-W_{ii}u_{ij}^{n+1}}{h_x^2}+$$
$$+C_{ij}\frac{V_{ji+1}u_{ij}^{n+1}+W_{jj+1}u_{jj+1}^{n+1}-V_{jj}u_{ij-1}^{n+1}-W_{jj}u_{jj}^{n+1}}{h_y^2}-\lambda_{ij}u_{ij}^n+Q_{ij}$$

Методы решения таких СЛАУ для РВС уже существуют и описаны в работах [79, 115, 118].

В настоящее время способ решения задач аномальной диффузии на равномерных сетках в значительной степени утратил свою актуальность [119, 120]. Для максимально точного описания геометрии сложных областей повсеместно применяются технологии адаптации расчетной сетки.

Рассмотрим построение двумерной адаптивной сетки из прямоугольных ячеек. На первом этапе такая сетка состоит из единственной прямоугольной ячейки C_0, покрывающей всю расчетную область. Такую ячейку называют корневой ячейкой сетки. Ячейка может быть разбита на четыре ячейки одинакового размера. Эти меньшие ячейки называются дочерними ячейками или потомками разделённой ячейки. Сама разделённая ячейка называется родителем своих потомков. Разделение ячейки означает создание её потомков без удаления самой ячейки. Существуют также алгоритмы, в которых ячейка разбивается не на четыре, а на две ячейки вдоль одной из координатных осей [121].

Каждая ячейка C_i $(0\le i<N)$ имеет размеры (h_i^x, h_i^y) и хранит величину V_i, описывающую среднее значение вычисляемой функции в пределах ячейки. В качестве окрестности ячейки возьмем прямоугольник размером $(3h_i^x, 3h_i^y)$ (рис. 3.2). В некоторых алгоритмах построения адаптивных сеток на окрестность ячейки накладывают дополнительные ограничения, позволяющие получить более точный результат для ряда задач [122].

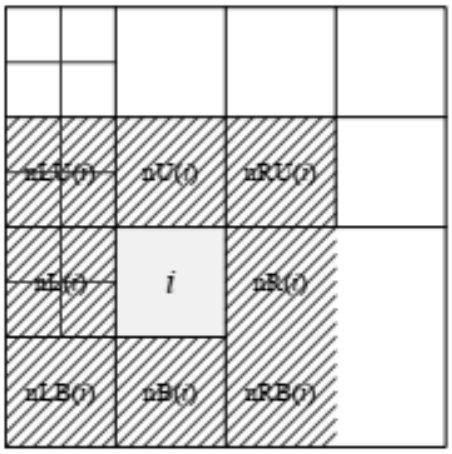


Рис. 3.2 – Окрестность ячейки C_i

Двумерную адаптивную сетку удобно хранить в виде четверичного дерева (рис. 3.3), где вычисления, связанные с задачей, производятся в листовых ячейках дерева. Листовая ячейка C_i может быть разделена на 4 дочерних ячейки: $C_{cLU(i)}$, $C_{cRU(i)}$, $C_{cLB(i)}$, $C_{cRB(i)}$. Для поддержания структуры дерева каждая ячейка C_i кроме величины V_i хранит индексы cLU(i), cRU(i), cLB(i), cRB(i) и индекс родительской ячейки $P(i)$.

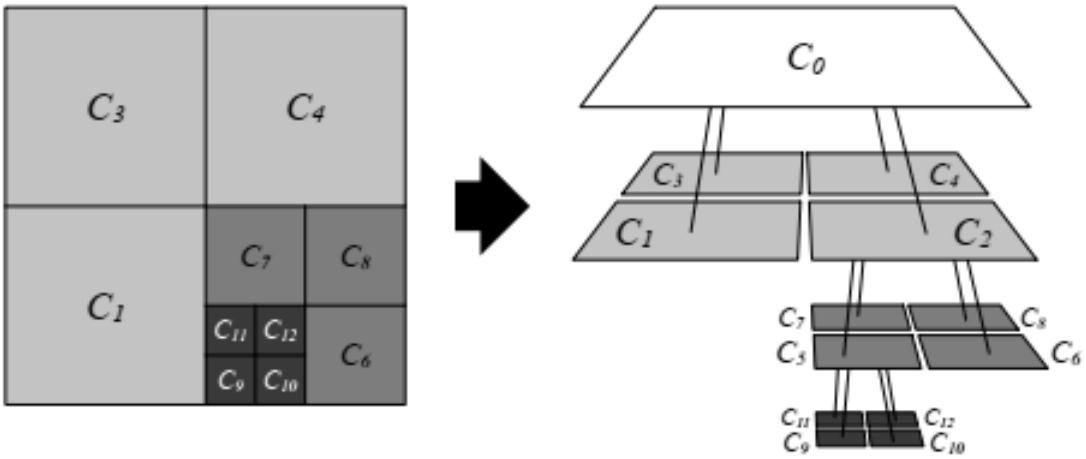


Рис. 3.3 – Представление сетки в виде четверичного дерева

Для работы с содержимым ячейки, а также для определения ее соседей, необходимых для построения шаблонов расчета сеточной функции, следует определить ее адрес, т.е. путь от корневой ячейки C_0 до ячейки C_i. Корневая ячейка будет иметь адрес, содержащий две пустые строки $(X_0,Y_0)=(\{\},\{\})$. Адрес

ячейки C_i получается из адреса её родителя с помощью следующей формулы: $(X_i,Y_i)=(\{X_{P(i)}\cdot x_i\},\{Y_{P(i)}\cdot y_i\})$, где $\cdot$ - оператор конкатенации, $x_i=0$, если C_i - один из левых потомков $C_{P(i)}$, и $x_i=1$, если C_i является правым потомком, $y_i=0$, если C_i - один из нижних потомков $C_{P(i)}$ и 1 – в противоположном случае. Например, адрес ячейки C_{11} на рисунке 3.3 будет равен $(\{100\},\{001\})$. Глубина d_i нахождения ячейки в дереве равна $d_i = | X_i | = | Y_i |$ где X_i − число разрядов в строке. Ячейка может быть найдена по её адресу путём перемещения по соответствующему пути от корня дерева.

Соседними с ячейкой C_i назовем 8 возможно совпадающих ячеек $C_{nLU(i)}$, $C_{nU(i)}$, $C_{nL(i)}$, $C_{nR(i)}$, $C_{nLB(i)}$, $C_{nB(i)}$, $C_{nRB(i)}$, которые расположены в окрестности ячейки. В примере, показанном на рисунке $C_{nR(i)}$ и $C_{nRB(i)}$ являются одной и той же ячейкой. Соседи могут иметь потомков, а также могут быть больше по размерам, чем сама ячейка. Кроме того, ячейка может иметь вплоть до 12 малых соседей, которые являются потомками восьми перечисленных ячеек.

Адреса соседних ячеек могут быть найдены с помощью двоичного сложения и вычитания единицы из компонент адреса ячейки. Например, $C_{nLU(i)}$ имеет адрес (X_i-1,Y_i+1), где под «-» и «+» понимаются двоичные вычитание и сложение. Для нахождения адресов малых соседей нужно взять потомков, если они существуют, найденных больших соседей.

Размеры каждой ячейки можно найти по формуле: $(h_i^x, h_i^y) = (h_0^x / 2^{d_i}, h_0^y / 2^{d_i})$. Координаты левого нижнего угла ячейки определяются выражениями $x_i = h_0^x \cdot X_i / 2^{d_i}$ и $y_i = h_0^y \cdot Y_i / 2^{d_i}$.

При построении линейно независимых шаблонов для расчета сеточной функции рассматривается конфигурация сетки в окрестности текущей ячейки (рис. 3.4).

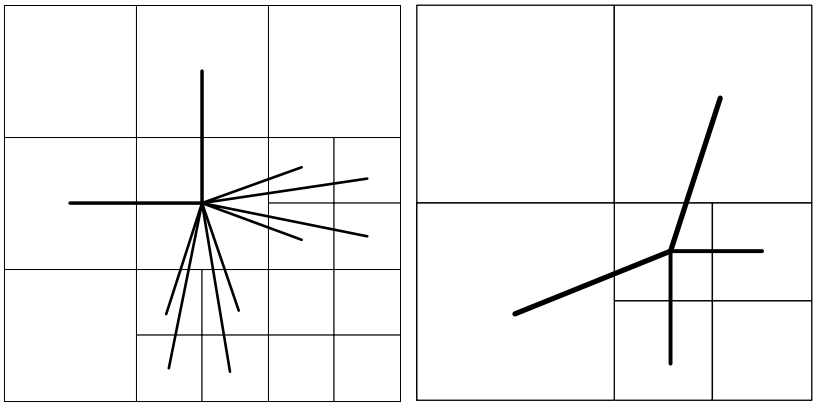


Рис. 3.4 – Построение линейно независимых шаблонов

При этом в случае крупной ячейки и мелких соседей берется усредненное значение функции в малых ячейках сетки, а в случае мелкой ячейки учитываются значения только непосредственных соседей. Т.е. если ячейка C имеет значение V, то при разбиении ее дочерние ячейки C_{cLU}, C_{cRU}, C_{cLB}, C_{cRB} получат такие значения V_{cLU}, V_{cRU}, V_{cLB}, V_{cRB}, чтобы выполнялось равенство $V = \frac{V_{cLU} + V_{cRU} + V_{cLB} + V_{cRD}}{4}$.

Для каждой расчетной области строится своя адаптивная сетка, отвечающая ограничением поставленной задачи (рис. 3.5) [123, 124] (работы [123, 124] выполнены в соавторстве с Подопригорой А.В.).

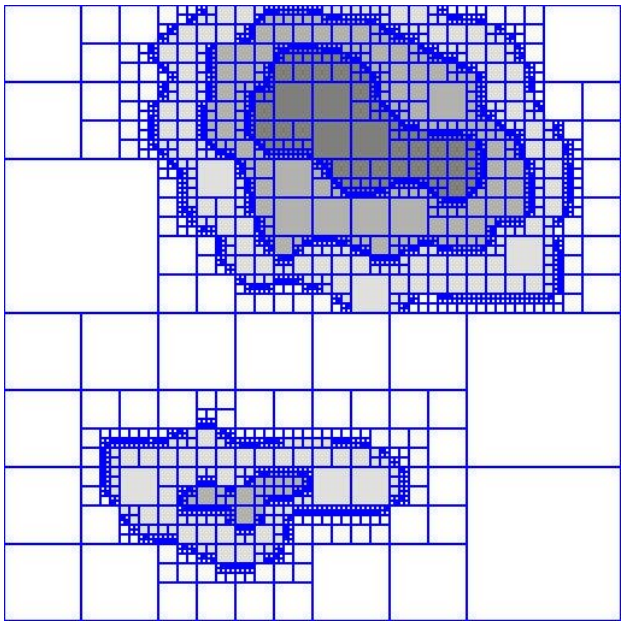


Рис. 3.5 – Полученная для расчетной области адаптивная сетка

При изменении входных данных или ограничений на построение вид адаптивной сетки будет меняться, а с ним и вид полученной СЛАУ даже при использовании одной и той же исходной математической модели. На рисунке 3.6 показано заполнение коэффициентами СЛАУ, соответствующей адаптивной сетке на рисунке 3.5.

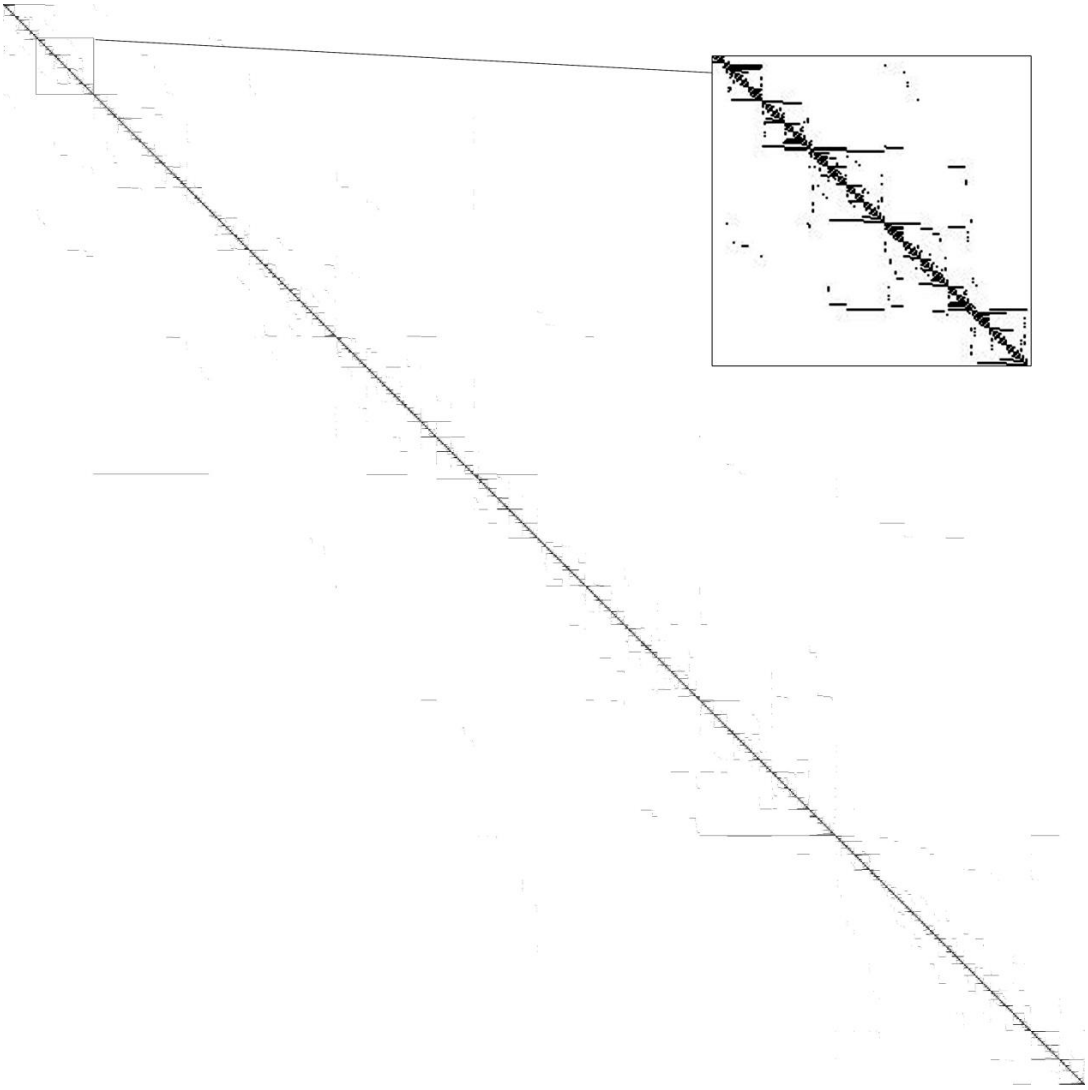


Рис. 3.6 – СЛАУ, полученная с использованием адаптивной сетки

Использование адаптивных сеток позволяет получить более точное решение задачи супердиффузии, однако в настоящее время не существует подходящих методов и средств для реализации на ПЛИС решения СЛАУ, полученных посредством применения адаптивных сеток. Таким образом, возникает необходимость создания таких методов и средств.

3.2 Вычислительные структуры для обработки ленточных матриц

Задачи математической физики, требующие решения СЛАУ, имеющих пять диагоналей со значимыми коэффициентами, успешно решаются на РВС посредством метода Якоби [125], который возможно распараллелить как по итерациям, так и по слоям. В этом случае значения вектора неизвестных вычисляются по формуле:

$$x_{ij}^{k+1} = \frac{1}{a_{ij}}\left(b_i - \left(a_{i-1\,j}x_{i-1\,j}^{k} + a_{i+1\,j}x_{i+1\,j}^{k} + a_{ij-1}x_{ij-1}^{k} + a_{ij+1}x_{ij+1}^{k}\right)\right). \qquad (3.8)$$

Таким образом, для поиска неизвестного на новом итерационном слое необходимы значения четырёх элементов вектора *x* из предыдущей итерации.

Информационный граф, соответствующий формуле (3.8), приведён на рисунке 3.7.

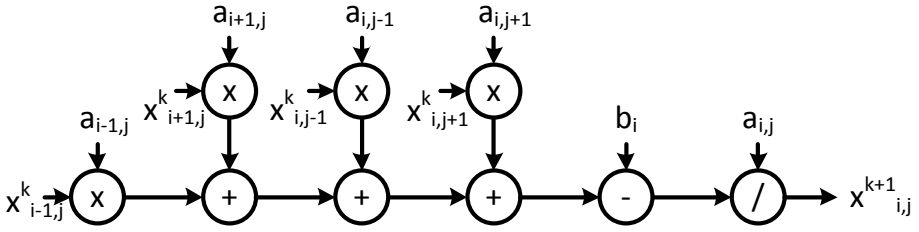


Рис. 3.7 – Информационный граф одной итерации метода Якоби для пятидиагональной СЛАУ

Поскольку операция сложения обладает свойствами ассоциативности и коммутативности, мы можем преобразовать исходный информационный граф к пирамидальному виду, представленному на рисунке 3.8.

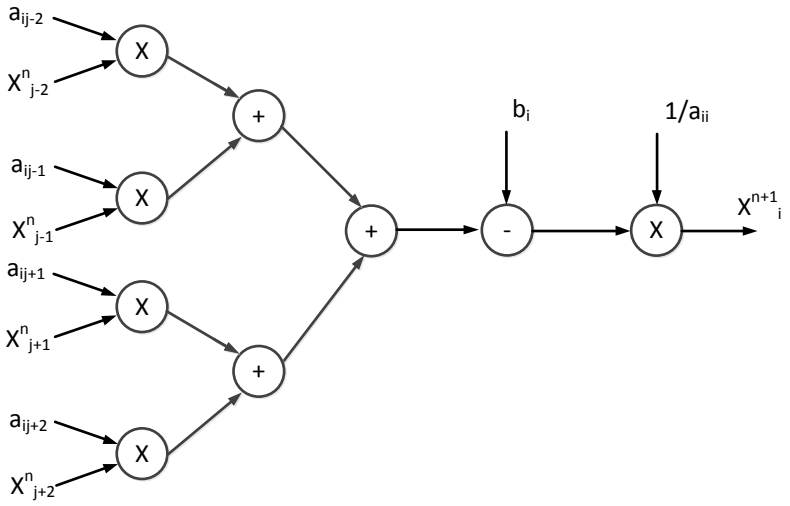


Рис. 3.8 – Параллельная форма одной итерации метода Якоби для пятидиагональной СЛАУ

Показанную выше структуру можно минимизировать посредством редукции числа входных каналов, после чего получим структуру, показанную на рисунке 3.9.

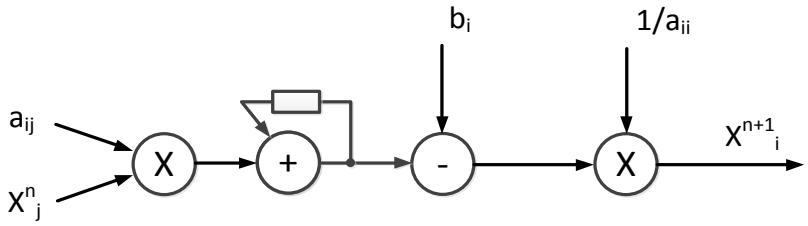


Рис. 3.9 – Базовый подграф метода Якоби, редуцированный по каналам

Для одновременного вычисления нескольких ступеней метода Якоби можно применить распараллеливание по итерациям, последовательно соединяя несколько базовых подграфов (рис. 3.10). На каждой итерации на вход ступени поступает поток операндов a_{ij} и x_j^k, вычисленные на предыдущей ступени. Выходными данными ступени являются новые значения x_j^{k+1}, которые в свою очередь подаются на следующую итерацию.

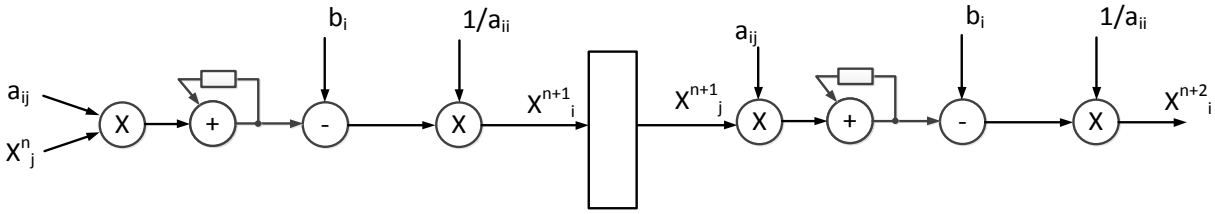


Рис. 3.10 – Распараллеливание по итерациям

Подобный подход становится малоэффективным, если СЛАУ содержит матрицу с нерегулярной структурой, как в случае дискретизации математической модели, описывающей супердиффузию газа, на адаптивных сетках. Специфика распределения ненулевых элементов в матрице полученной СЛАУ такова, что решения ее на реконфигурируемых вычислительных системах известными методами не эффективны и ведут к чрезмерным расходам оборудования [115]. Например, количество диагоналей, содержащих значимые элементы может приближаться к количеству строк в матрице. При этом только пять центральных диагоналей плотно заполнены коэффициентами и содержат 70-95% всех ненулевых элементов матрицы [126, 127]. Если для решения подобной СЛАУ применить метод Якоби, ориентированный на специфику ленточных матриц, то большую часть времени вычисления будут производиться над нулевыми элементами, что фактически означает простой оборудования. Этот случай проиллюстрирован временной диаграммой на рисунке 3.11. Таким образом, возникает необходимость создания эффективных методов и средств для решения СЛАУ, содержащих матрицы подобного типа.

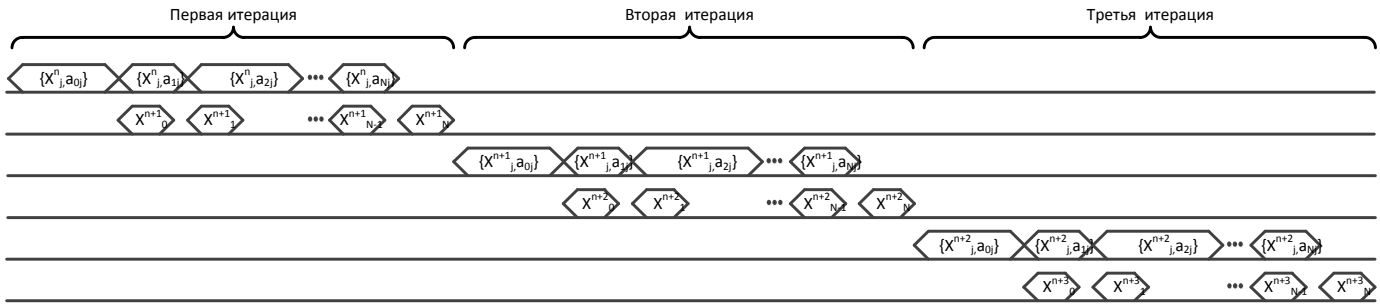


Рис. 3.11 – Временная диаграмма работы трех итераций метода Якоби при обработке матрицы с нерегулярной структурой

Если применить описанный выше подход для нерегулярной матрицы, то в предельном случае для начала вычислений в новой ступени необходимо будет вычисление всех значений x_j^{k+1} на предыдущей ступени.

Исходя из выше сказанного требуется модификация реализации метода Якоби с учетом специфики задачи. Современные ПЛИС обладают значительным вычислительным ресурсом, но очень сильно ограничены по числу входных каналов. В связи с этим целью модификации реализации метода Якоби становится достижение наиболее плотного потока данных при максимальном использовании вычислительного ресурса ПЛИС.

3.3 Способ оценки оптимального количества каналов

При решении СЛАУ, содержащей матрицу с нерегулярной структурой, как в случае дискретизации математической модели, описывающей супердиффузию газа, на адаптивных сетках, возникает необходимость обработки строк разной длины. При этом минимальное количество данных в строке равно трем, а максимальное зависит от способа дискретизации и может быть на порядок или два больше.

На рисунке 3.12 представлены вычислительные структуры, специально адаптированные под каждую строку. Как видно, они отличаются друг от друга по количеству входных данных и вычислительных блоков. Тем не менее, для того, чтобы обеспечить удобство масштабирования согласно принципам эффективной обработки самоподобных структур на РВС необходимо получить подобные вычислительные структуры, которые обеспечивают оптимальное соотношение между каналами подачи данных и используемым оборудованием.

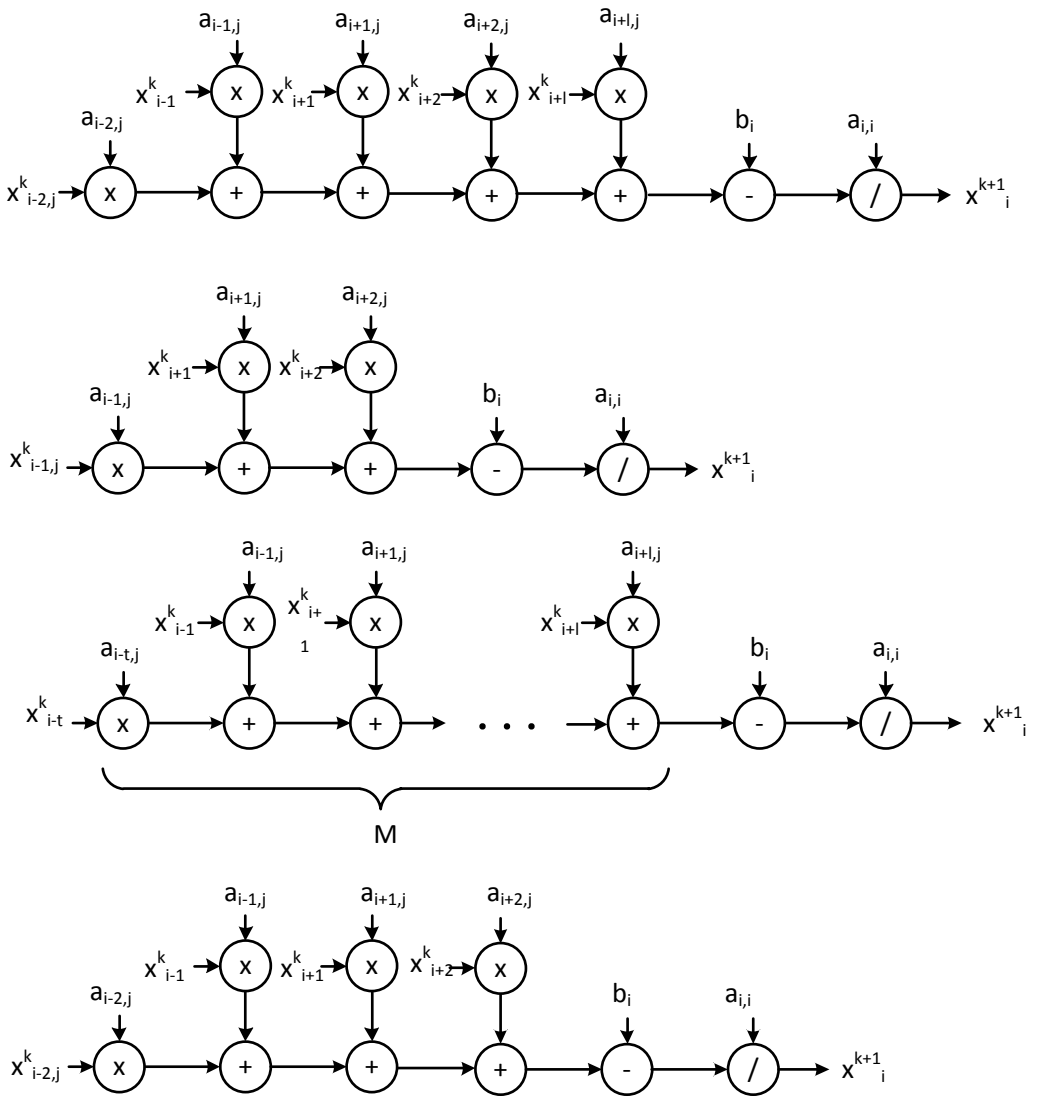
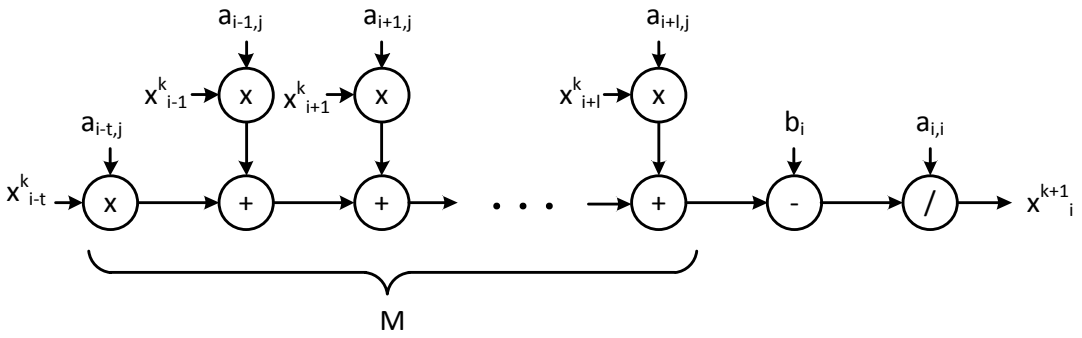


Рис. 3.12 – Отдельные вычислительные структуры для разных строк матрицы

На практике при реализации вычислительных структур обычно выбирают один из двух вариантов реализации: в случае наличия достаточного ресурса реализуют максимальную вычислительную структуру, а в противном случае реализуют минимальную. Но при реализации вычислительных структур для решения СЛАУ с нерегулярной структурой оба этих варианта обладают значительными недостатками.

Реализация максимальной вычислительной структуры будет малоэффективна, т.к. количество строк, содержащих максимальное количество

ненулевых элементов, менее 0,5% от общего размера матрицы. Таким образом, при подобной реализации значительная часть оборудования будет простаивать 99% времени вычисления (рис. 3.13 б).



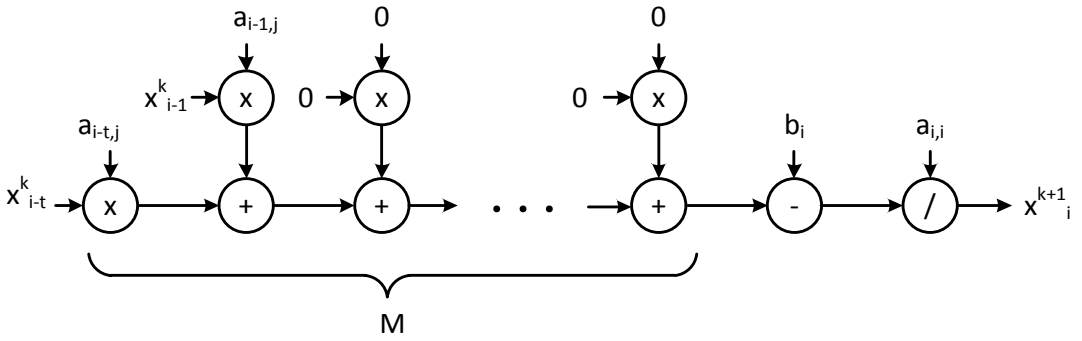


Рис. 3.13 – Подача строки с максимальной (а) и минимальной (б) загрузкой

При реализации минимальной вычислительной структуры мы избежим простоя оборудования, однако вынуждены будем столкнуться с необходимостью провести модификацию вычислительной структуры, обеспечивающей корректное подсуммирование результатов при вычислении строк, имеющих длину больше минимальной (рис. 3.14).

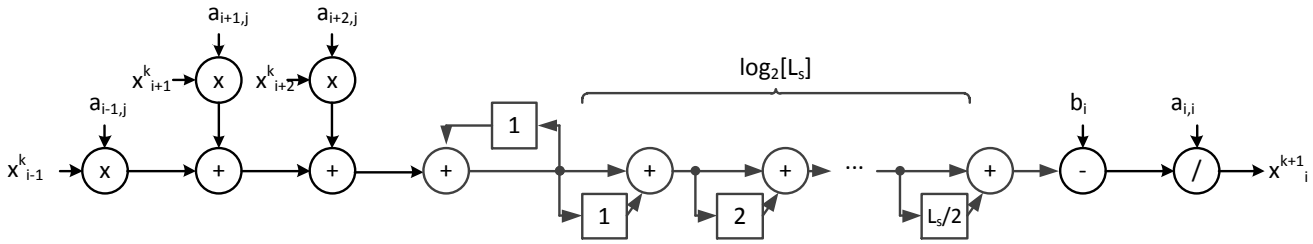


Рис. 3.14 – Модификация вычислительной структуры, уменьшающая скважность потока данных

При этом количество оборудования, потраченного на реализацию этой дополнительной структуры, будет соизмеримо или даже равно количеству оборудования, затраченного на реализацию исходной вычислительной структуры. Количество сумматоров, осуществляющих подсуммирование результатов, зависит от латентности одного сумматора и равно $\log_2 L_S + 1$, где L_s – латентность сумматора. Для большинства реальных значений L_s эта величина будет больше минимальной длины строки.

Из выше сказанного следует, что оба рассмотренных случая, не являются оптимальными для задачи решения СЛАУ с нерегулярной структурой. В работе [128] показано, что при реализации некоторых вычислительных структур оптимальным реализуемым размером является промежуточный между максимальным и минимальным. Таким образом, следует найти оптимальную структуру для базового изоморфного подграфа.

Ключевой задачей при поиске оптимальной структуры является определение необходимого количества каналов для подачи данных, связанного со средним количеством ненулевых элементов в строке и влияющее на затраты оборудования.

Рассматриваемые СЛАУ имеют большой разброс по количеству элементов в строках, пример такого распределения для матрицы размером 1441x1441 показан на рисунке 3.15. Математическое ожидание количества элементов в строке $M(Q) \approx 6,5$, дисперсия $D(Q) \approx 8,5$, а среднеквадратичное отклонение $\Delta Q \approx 2.9$

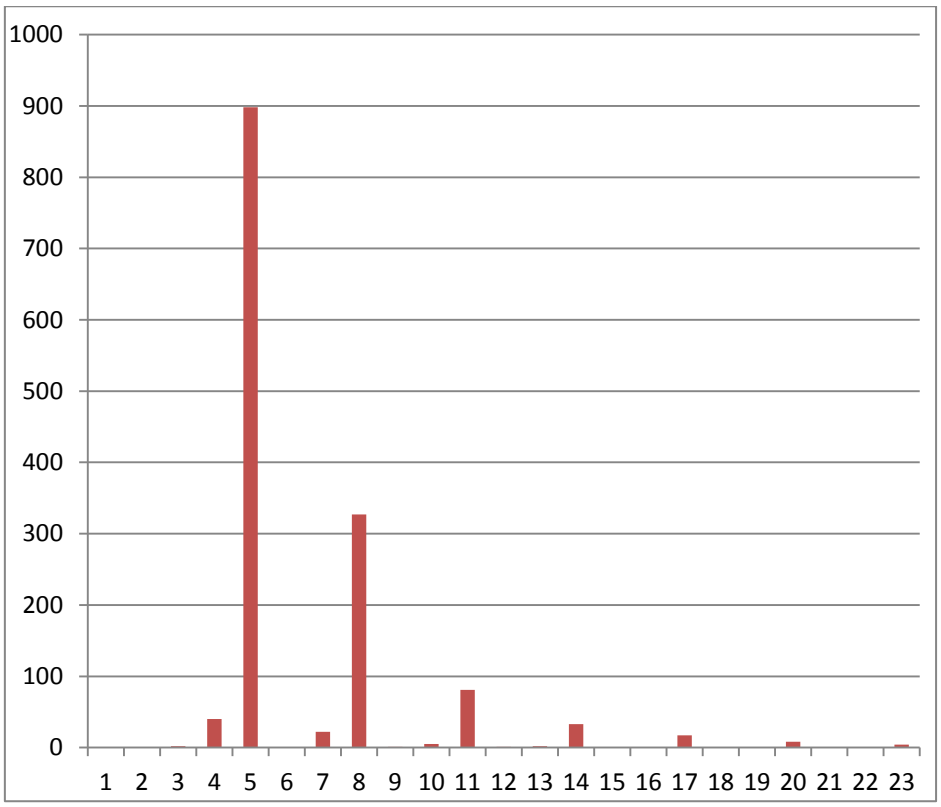


Рис. 3.15 – Гистограмма распределения количества элементов в строке матрицы

Если количество каналов для подачи данных будет равно максимально возможному количеству элементов в строке Q_{max}, то коэффициент использования оборудования U для каждой отдельной строки будет равен $U = Q_i / Q_{\max}$, где i – номер строки, и для всей матрицы соответственно $U = \frac{1}{N}\sum_{i=1}^{N} Q_i / Q_{\max}$, где N – количество строк в матрице. Очевидно, что в этом случае коэффициент использования оборудования для прохода всей матрицы через одну итерацию будет примерно равен ¼.

Рассмотрим случай, при котором количество каналов $x < Q_{max}$. В этом случае коэффициент использования оборудования для прохода всей матрицы будет равен $U = \frac{1}{N}\sum_{i=1}^{N} \frac{Q_i / x}{\lceil Q_i / x \rceil}$. В этом случае целесообразно выполнить оценку для этого

параметра $U^* \approx \frac{(M(Q)+\Delta Q)/x}{\lceil (M(Q)+\Delta Q)/x \rceil}$. Найдем оптимальное значение для количества каналов *x*. Для этого возьмем производную от U* и приравняем ее к нулю:

$$\frac{((M(Q)+\Delta Q)/x)'\lceil (M(Q)+\Delta Q)/x \rceil - (M(Q)+\Delta Q)/x\left(\lceil (M(Q)+\Delta Q)/x \rceil\right)'}{\lceil (M(Q)+\Delta Q)/x \rceil^2} = 0. \quad (3.9)$$

Функция $y = \lceil (M(Q)+\Delta Q)/x \rceil$ не имеет производной, поэтому заменим ее непрерывным аналогом, полученным с помощью метода наименьших квадратов. Для этого подставим в *y(x)* значение $(M(Q)+\Delta Q) \approx 9.4$, получим приближенную функцию $y = 8.9123x^{-0.8}$ (рис. 3.16).

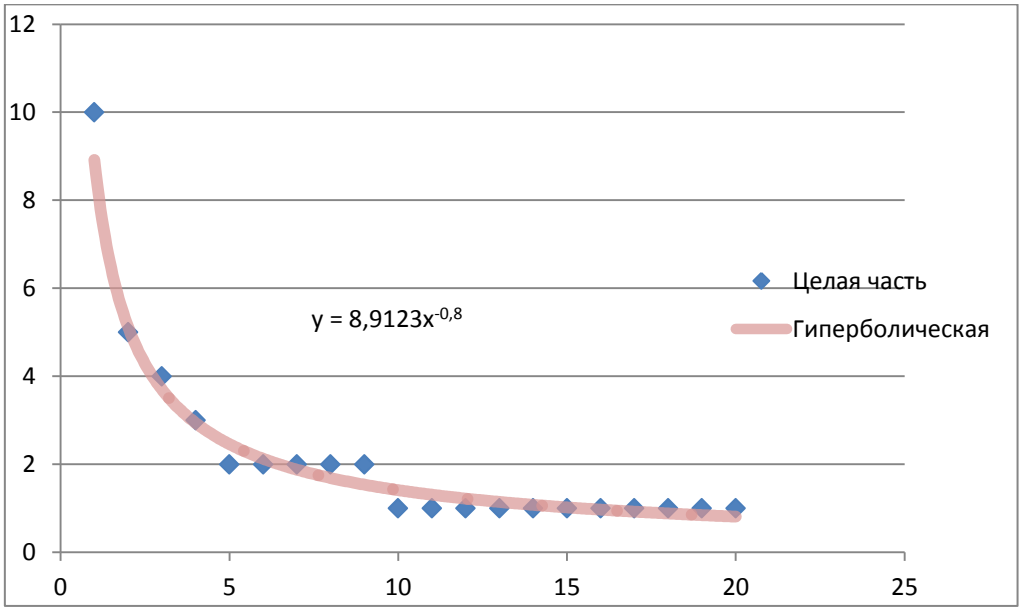


Рис. 3.16 – Интерполяция точечной функции

Подставим полученную функцию в уравнение (3.9), получим следующее выражение:

$$(9.4/x)'\lceil 9.4/x \rceil - (9.4/x)\left(8.9123x^{-0.8}\right)' = 0.$$

После алгебраических преобразований, получим

$$\lceil 9.4/x \rceil = 7.12984x^{-0.8}. \quad (3.10)$$

Параметр x отражает количество каналов для передачи данных, а значит является натуральным числом. Подберем значение x, наилучшим образом удовлетворяющее выражению (3.10). Результаты показаны в таблице 3.1.

Таблица 3.1 Сравнение значений для разных x

	$\lceil 9.4/x \rceil$	$7.12984x^{-0.8}$
	10	7.13
	5	4.1
	4	2.96
	3	2.35
	2	1.97
	2	1.7
	2	1.5
	2	1.35

Как видно из таблицы 3.1 при x=5 левая и правая часть выражения (3.10) наиболее близки по значениям, а значит при таком количестве каналов будет достигнуто оптимальное использование оборудования $U^* \approx 0.94$ [129].

Из выше сказанного можно сделать вывод, что определение оптимального количества каналов для подачи элементов строки СЛАУ может быть не совсем очевидным, и оптимальным может являться не среднее значение, как кажется на первый взгляд. Полученное выше значение x=5 также может меняться при существенных изменениях вида матрицы, например, если адаптивная сетка строится по правилам, отличным от описанных ранее. В этом случае оптимальное значение каналов подачи данных можно найти способом, описанным выше.

Полученные выше оценки могут быть использованы как для получения точных вычислительных структур, например, реализованных на языке высокого уровня COLAMO [130], так и для выбора приближенной вычислительной структуры из списка готовых решений.

3.4 Сравнение производительности для различных реализаций

При создании вычислительной структуры принято использовать два подхода. Если количество оборудования достаточно для размещения всего графа

целиком, то синтезируется вычислительная структура, соответствующая полному графу. В противном случае согласно теореме Иванова [131] используется минимальный граф задачи.

Представленные варианты могут быть использованы при решении задачи аномальной супердиффузии методом Якоби на адаптивных сетках, однако они обладают рядом недостатков.

В случае реализации единственного минимального изоморфного подграфа время работы вычислительной схемы для одного полного прогона СЛАУ составит

$$T_{\min} = \left(2 \cdot L_{mul} + L_{sum} + L_{sub} + L_{sum} \cdot \hat{Q} \cdot n\right)\tau,$$

где $\hat{Q}$ – среднее количество значащих элементов в строке матрицы, n – количество строк в СЛАУ, L_{mul} – латентность умножителя, L_{sum} – латентность сумматора, τ – длительность одного такта.

На рисунке 3.17 показана реализация минимального информационного графа для решения разреженной СЛАУ специального вида методом Якоби.

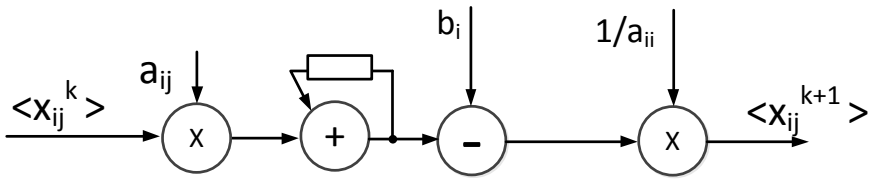


Рис. 3.17 – Минимальный информационный граф

Применение описанного выше подхода обеспечит минимальный простой оборудования, однако для выполнения корректных вычислений необходимо подавать данные со значением скважности, равным латентности сумматора с накоплением, что приводит к большим временным затратам.

Описанный выше вариант получил развитие в работах [128, 132], при этом информационный граф задачи принимает вид, показанный на рисунке 3.18.

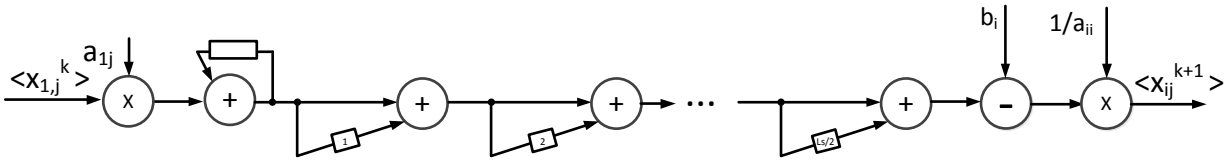


Рис. 3.18 – Модификация минимального информационного графа

В данной реализации сумматор с накоплением снабжается дополнительными сумматорами, которые осуществляют корректное подсуммирование частичных сумм за счет подачи операндов на их входы с задержками, которые определяются специальным образом [128] в зависимости от латентности сумматора с накоплением – число тактов задержки каждой следующей линии равно очередной степени двойки вплоть до значения, равного половине латентности сумматора. В случае если значение латентности сумматора не равно целой степени двойки, необходимо дополнить поток операндов нулевыми элементами, а цепь обратной связи – регистрами до ближайшей целой степени двойки, что приводит к усложнению организации вычислений.

В этом случае время работы вычислительной схемы для одного полного прогона СЛАУ составит

$$T_{mod} = \left(2 \cdot L_{mul} + \hat{Q} \cdot n + L_{sum} + L_{sub} + \lceil \log_2 L_{sum} \rceil L_{sum}\right)\tau$$

Применение такой модификации уменьшит скважность и соответственно увеличит скорость работы схемы по сравнению с предыдущим вариантом, но также возрастут накладные расходы оборудования, что уменьшит эффективность его использования.

Ещё одним вариантом реализации является вычислительная структура, рассчитанная на одновременную обработку всех элементов одной строки (включая нулевые). Пример такой реализации представлен в работах [73, 133]. Количество входных каналов, по которым подаются операнды, в этом случае будет равно m, где m – ширина ленты, покрывающей собой все диагонали, содержащие хотя бы один значащий элемент. Информационный граф, соответствующий такой реализации, представлен на рисунке 3.19.

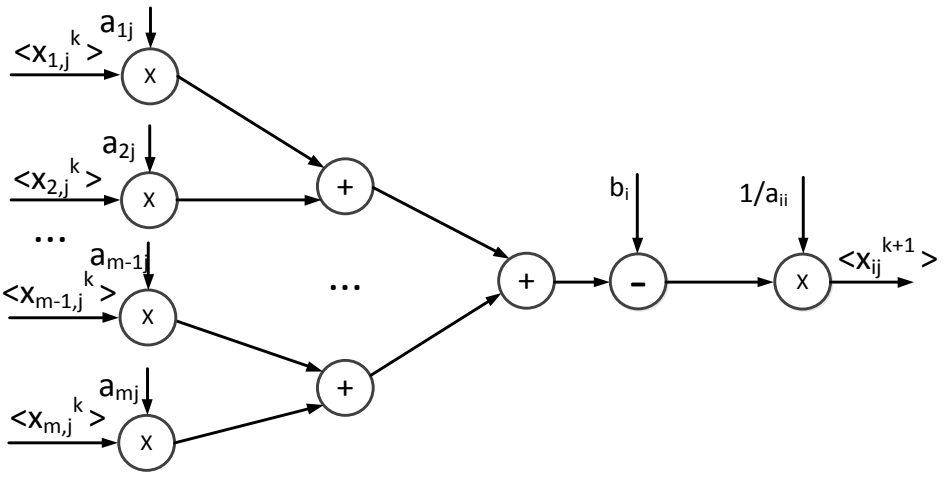


Рис. 3.19 – Информационный граф, соответствующий вычислительной структуре для обработки матрицы с широкой лентой

Отличием данной вычислительной структуры от минимального варианта является то, что входной умножитель заменяется на блок, включающий m умножителей, а сумматор – на пирамиду сумматоров с m входами (состоящую из m-1 сумматора). Благодаря этому время работы вычислительной схемы для одного полного прогона СЛАУ значительно уменьшится и составит

$$T_{\max} = \left(U \cdot n + L_{mul} + L_{sub} + L_{sum} \log_2 m\right)\tau ,$$

где U – коэффициент использования оборудования $U = \frac{1}{N}\sum_{i=1}^{N} Q_i / Q_{\max}$.

Недостатком данного варианта являются значительные затраты оборудования (поскольку m может составлять около половины от n, при том, что n может равняться 10^6 и более) и низкий коэффициент использования оборудования при работе с матрицами специального вида, для которых Q<<m. Большую часть времени вычислительная структура будет обрабатывать нулевые коэффициенты.

После анализа достоинств и недостатков существующих реализаций представляется целесообразным реализовать структуру, объединяющую в себе черты второго и третьего вариантов. Информационный граф, соответствующий данной структуре, представлен на рисунке 3.20 [134, 135].

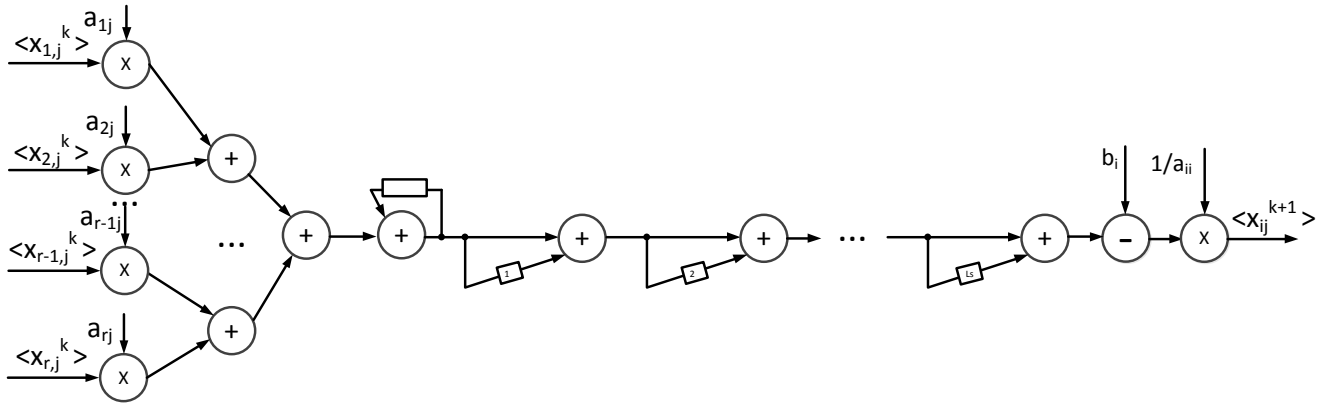


Рис. 3.20 – Информационный граф модифицированной вычислительной структуры

Размер пирамиды сумматоров (и количество умножителей на входе) r определяется в зависимости от параметра Q. Эта зависимость описана ранее в параграфе 3.3. Время работы вычислительной схемы в данном случае будет равно

$$T_{opt} = \left(n + L_{mul} + \left\rceil \frac{Q}{r} \right\lceil \cdot U' + L_{sum}\left(\log_2 r + \log_2 L_{sum} + 1\right) + L_{sub} \right)\tau ,$$

где $U' \approx \frac{\left(M(Q) + \Delta Q\right)/K}{\left\lceil \left(M(Q) + \Delta Q\right)/K \right\rceil}$. Данное выражение будет верно для случая $r \le K - 1$.

Соответственно, ускорение по сравнению с реализацией минимального информационного графа, составит примерно

$$S_{o/\min} = \frac{T_{\min}}{T_{opt}} = \frac{L_{mul} + L_{sum} \cdot Q \cdot n}{n + L_{mul} + \left\rceil \frac{Q}{r} \right\lceil \cdot U' + L_{sum}\left(\log_2 r + \log_2 L_{sum} + 1\right) + L_{sub}} \approx L_{sum} \cdot Q .$$

Ускорение по сравнению с модификацией минимального информационного графа составит

$$S_{o/\mathrm{mod}} = \frac{T_{mix}}{T_{opt}} = \frac{2 \cdot L_{mul} + \hat{Q} \cdot n + L_{sum} + L_{sub} + \left\lceil \log_2 L_{sum} \right\rceil L_{sum}}{n + L_{mul} + \left\rceil \frac{Q}{r} \right\lceil \cdot U' + L_{sum}\left(\log_2 r + \log_2 L_{sum} + 1\right) + L_{sub}} \approx \hat{Q} .$$

Ускорение по сравнению с реализацией максимального информационного графа составит

$$S_{o/\max} = \frac{T_{\max}}{T_{opt}} = \frac{U \cdot n + L_{mul} + L_{sub} + L_{sum} \log_2 m}{n + L_{mul} + \left\lceil \frac{Q}{r} \right\rceil \cdot U' + L_{sum}\left(\log_2 r + \log_2 L_{sum} + 1\right) + L_{sub}} \approx U.$$

Значение U лежит в пределах (0,1), таким образом оптимальная реализация проиграет по скорости максимальной, но будет более эффективной по сравнению с ней.

Полученные выше соотношения справедливы для случая, когда реализуется один вычислительный блок. Однако решение задачи может быть ускорено в случае, если вычислительная структура будет включать большее количество таких блоков. Рассмотрим случай реализации вычислительной структуры, включающей в себя некоторое количество вычислительных блоков, ограниченное доступным вычислительным ресурсом. Примем количество входных каналов неограниченным, а доступное количество вычислительных элементов обозначим как A.

В этом случае для реализации модификации минимального информационного графа затраты оборудования составят $R_{mod} = 2 \cdot A_{mul} + A_{sub} + A_{sum}(\log_2 L_{sum})$. Таким образом мы сможем реализовать $H_{mod} = \left\lfloor \frac{A}{R_{mod}} \right\rfloor$ вычислительных блоков, а общие затраты оборудования составят $A_{mod} = R_{mod} \cdot H_{mod}$. Для такой вычислительной системы время работы для одного полного прогона СЛАУ составит

$$T_{mod} = \left(2 \cdot L_{mul} + \frac{\hat{Q} \cdot n}{H_{mikh}} + L_{sum} + L_{sub} + \lceil \log_2 L_{sum} \rceil L_{sum} \right) \tau$$

Для реализации вычислительного графа для обработки матриц с широкой лентой затраты оборудования составят $R_{\max} = (m+1) \cdot A_{mul} + A_{sub} + A_{sum} \cdot (m-1)$. Количество реализованных вычислительных блоков будет равно $H_{\max} = \left\lfloor \frac{A}{R_{\max}} \right\rfloor$.

Время выполнения одного полного прогона СЛАУ в такой системе составит

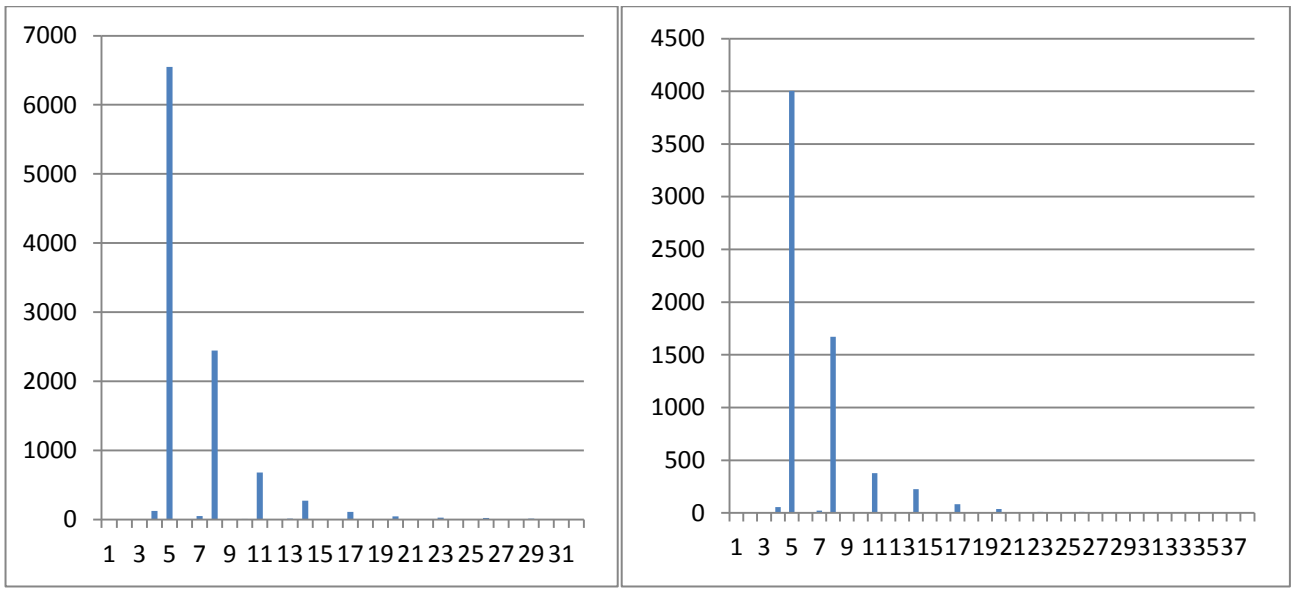
$$T_{\max} = \left(U \cdot \frac{n}{H_{\max}} + L_{mul} + L_{sub} + L_{sum} \log_2 m \right) \tau .$$

Для реализации модифицированного информационного графа затраты оборудования составят $R_{opt} = (r+1) \cdot A_{mul} + A_{sub} + A_{sum} \cdot (r + \lceil \log_2 L_{sum} \rceil)$. Таким образом мы сможем реализовать $H_{opt} = \left\lfloor \frac{A}{R_{opt}} \right\rfloor$ вычислительных блоков. Время выполнения одного полного прогона СЛАУ в такой системе составит

$$T_{opt} = \left(\frac{n}{H_{opt}} + L_{mul} + \right\rceil \frac{Q}{r} \left[\cdot U' + L_{sum} \left(\log_2 r + \log_2 L_{sum} + 1 \right) + L_{sub} \right) \tau .$$

3.5 Реальная производительность РВС при решении задачи супердиффузии

Распределение ненулевых элементов в СЛАУ меняется в зависимости от решаемой задачи, наибольший разброс имеет максимальное количество элементов в строке. На рисунке 3.21 представлены гистограммы для двух различных матриц, в первом случае максимальное количество элементов в строке достигает 32, во втором – 38. При этом также меняются и математическое ожидание $M(Q) \approx 6,3 - 6,7$, дисперсия $D(Q) \approx 8,3 - 9,2$, среднеквадратическое отклонение $\Delta Q \approx 2.7 - 3$.



а) б)

Рис. 3.21 – Гистограммы распределения элементов в матрицах 10000*10000 (а) и 6500*6500 (б)

Рассмотрим общую формулу для времени работы вычислительной схемы:

$$T = \frac{IN\hat{Q}}{P_{it}P_qP_s}S + T_{конв}, \tag{3.11}$$

где I – количество итераций, N – количество строк в матрице, $\hat{Q}$ - среднее количество ненулевых элементов в строке, S – скважность потока данных, $T_{конв}$ – время заполнения конвейера, P_{it} – количество реализованных итераций, P_q – количество затраченного оборудования, P_s – количество реализованных слоев.

При решении реальных задач количество строк в матрицах, как правило, имеет большие значения (N > 10000), поэтому временем заполнения конвейера $T_{конв}$ можно пренебречь, и формула (3.11) приобретет вид

$$T \approx \frac{IN\hat{Q}}{P_{it}P_qP_s}S. \tag{3.12}$$

Чтобы получить время работы для минимальной реализации вычислительного графа задачи [131], подставим значения соответствующие данному случаю: $P_q = 4$; $S = l_+$, где l_+ - латентность сумматора, для РВС Арктур $l_+=12$ для чисел с плавающей запятой одинарной точности; $P_s = K/2$, где K –

количество доступных каналов; P_{it} = A/(P_q P_s), где A – количество доступного оборудования. Таким образом, время работы может быть вычислено по формуле

$$T_{\min} = \frac{IN\hat{Q}}{A} S .$$

Для модификации минимального информационного графа [128] также можно вычислить время по указанному методу, но в отличие от предыдущего варианта скважность S = 1, но помимо количества полезных вычислителей P_q = 4, следует также учитывать вспомогательные, поэтому введем параметр M_q= P_q+]$\log_2 l_+$[. Количество итераций также будет зависеть от M_q и вычисляться по формуле P_{it} = A/(M_q P_s). Тогда время работы будет равно

$$T_{mod} = \frac{IN\hat{Q}}{A} \frac{M_q}{P_q} .$$

При максимальной реализации [73] в формуле (3.12) среднее количество элементов в строке $\hat{Q}$ заменяется на Q_{max}, т.к. количество входных каналов должно соответствовать максимально возможному количеству ненулевых элементов в строке. При этом необходимо учитывать эффективность использования каналов, поэтому формула (3.12) примет следующий вид

$$T_{\max} = \frac{INQ_{\max}}{A\hat{Q}} .$$

Для реализации оптимизированного информационного графа количества полезных вычислителей $P_q = 4+2\hat{Q}$, при этом параметр, учитывающий вспомогательные вычислители будет равен M_q= P_q+]$\log_2 l_+$[. Тогда формула (3.12) примет вид

$$T_{opt} = \frac{IN\hat{Q}}{A} \frac{P_q}{M_q} .$$

Также необходимо учитывать коэффициент использования каналов $K_L = \frac{\sum_{i=i_{\min}}^{i_{\max}} m_i \frac{Q_i}{\hat{Q}}}{\sum_{i=i_{\min}}^{i_{\max}} \left\lceil \frac{Q_i}{\hat{Q}} \right\rceil m_i}$, где Q_i – количество значащих элементов в строке, m_i – количество строк в матрице, содержащих Q_i элементов соответственно, i_{max} – максимальное количество значащих элементов в строке, i_{min} – минимальное количество значащих элементов в строке.

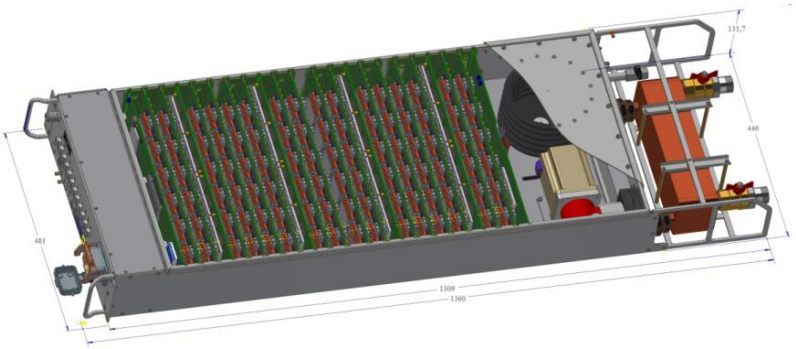
Количество данных в строке не всегда кратно среднему, с учетом этого формула (3.12) примет вид

$$T_{opt} = \frac{IN\hat{Q}}{A} \frac{P_q}{M_q} K_L . \quad (3.13)$$

Также возможен случай, когда количества доступных каналов не хватает для реализации оптимального графа, и их необходимо сократить, в этом случае время будет вычисляться по формуле аналогичной формуле (3.13), но абсолютное значение коэффициента K_L будет меньше, чем соответствующее для оптимального варианта. Также $\hat{Q}$ будет заменено на Q* – количество доступных каналов:

$$T_{opt} = \frac{INQ^*}{A} \frac{P_q}{M_q} K_L^* . \quad (3.14)$$

При работе с современными высокопроизводительными РВС, такими как Арктур [74] (рис. 3.22), в доступе имеется большое количество каналов передачи данных (в случае вычислительного блока Арктур доступно 3072 32-хбитных канала для работы), что обеспечивает возможность реализации нескольких параллельно работающих вычислительных конвейеров, соответствующих оптимальному вычислительному графу.



а)

б)

Рис. 3.22 – РВС Арктур - а) вычислительный блок, б) вычислительный модуль.

В случае если вычисления производятся на РВС, где число каналов данных является критическим ресурсом, например вычислительный блок Терциус-2 [69], содержащий только 16 доступных 32хбитных каналов, то необходимо уменьшить количество входных каналов в вычислительном графе, тогда эффективность подобной реализации будет рассчитываться по формуле (3.14) (рис. 3.23).

Рис. 3.23 – Вычислительный блок Терциус-2

Эффективность реализации алгоритма оценивается как отношение числа полезных вычислительных блоков к общему их количеству, которое включает в себя вспомогательные вычислительные блоки, например, дополнительные сумматоры, уменьшающие скважность потока данных. В таблице 3.2 представлены значения уровня реальной производительности для описанных выше случаев при обработке матриц с разными параметрами с учётом вычислений для чисел с плавающей запятой одинарной точности. В таблице 3.3 представлены аналогичные параметры, но для чисел с двойной точностью [136].

Таблица 3.2. Значения уровня реальной производительности реализаций для одинарной точности.

	$Q_{max} = 26$	$Q_{max} = 38$	$Q_{max} = 41$
E_{min}	0,08	0,08	0,08
E_{mod}	0,5	0,5	0,5
E_{max}	0,25	0,17	0,16
E	0,64	0,64	0,64
E*	0,53	0,53	0,53

Таблица 3.3. Значения уровня реальной производительности реализаций для двойной точности

	$Q_{max} = 26$	$Q_{max} = 38$	$Q_{max} = 41$
E_{min}	0,03	0,03	0,03
E_{mod}	0,44	0,44	0,44
E_{max}	0,25	0,17	0,16
E	0,62	0,62	0,62
E*	0,50	0,50	0,50

При моделировании реальных процессов супердиффузии, как и в других задачах математической физики используются вычисления с двойной точностью. Это обусловлено необходимостью обработки больших объемов данных, т.к. вычисления с меньшей точностью приводят к накоплению ошибки округления, что в свою очередь влияет на предсказательную точность модели.

Приведенные выше таблицы с экспериментальными данными подтверждают теоретические выводы. Предложенный метод решения задачи распространения газа в фрактальной среде обеспечит повышение реальной производительности на 40% по сравнению с известными способами решения. Таким образом, доказано третье положение, выносимое на защиту.

3.6 Повышение реальной производительности посредством перестановки строк в СЛАУ

Можно дополнительно увеличить производительность вычислений, если специально подготовить матрицу, содержащую обрабатываемые значения. Для этого необходимо произвести сортировку строк матрицы с учетом количества значимых данных в каждой строке.

Очевидными способами сортировки является перемещение наиболее длинных строк вверх, или вниз, группируя их вместе (рис. 3.24).

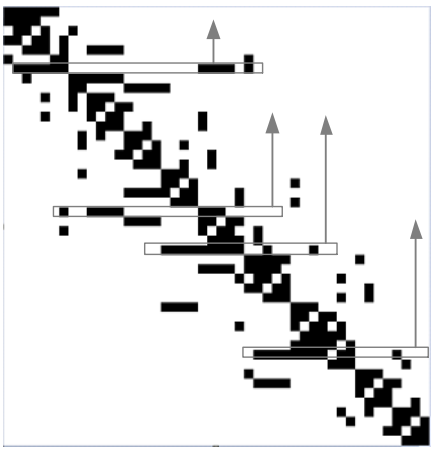


Рис. 3.24 – Сортировка матрицы «большие строки вверх»

Также возможно производить аналогичные сортировки не во всей матрице, а внутри кластеров различного размера (рис. 3.25).

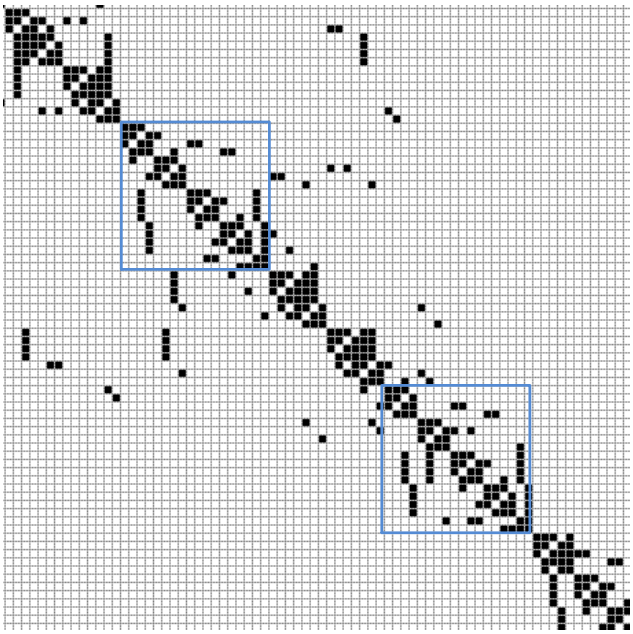


Рис. 3.25 – Пример выделения кластеров в матрице

Следует отметить, что в случае применения только распараллеливания по итерациям изменение порядка следования строк в матрице может привести к выигрышу в скорости вычислений только на первой итерации по сравнению с естественным порядком следования строк, в последующих итерациях скорость выравнивается, или может быть даже хуже, чем у неотсортированной матрицы, из-за наличия большого количества информационных связей между строками. На

рисунке 3.26 показано, что для вычисления элемента номер 3 необходимо дождаться значения седьмого элемента с предыдущей итерации.

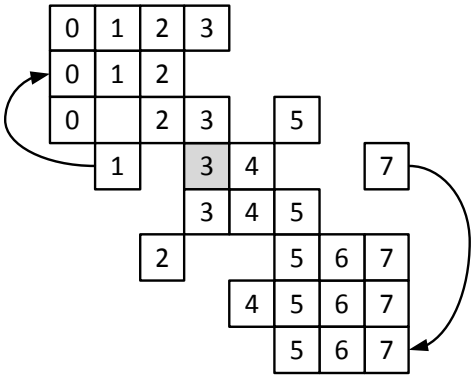


Рис. 3.26 – Информационные связи между строками СЛАУ

При распараллеливании задачи по слоям строки матрицы подаются в каждый слой согласно их порядку (рис. 3.27).

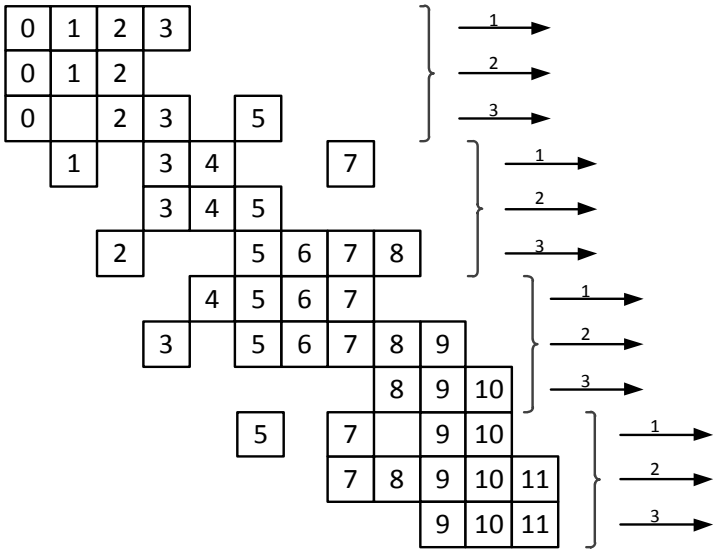


Рис. 3.27 – Распределение строк матрицы в вычислительные слои

Матрицы, отсортированные кластерным способом, показывают лучший результат, чем матрицы, где сортировка производилась по всем строкам. При этом в некоторых случаях скорость вычислений на 10% выше, чем у неотсортированных матриц. Однако этот способ не показывает устойчивого ускорения для всех возможных матриц СЛАУ, образованных при дискретизации задачи супердиффузии на адаптивных сетках. Это может происходить из-за

задержек, обусловленных информационными зависимостями между строками. Также при равномерном распределении строк между параллельными слоями может произойти неравномерное заполнение данными вычислительной системы, т.е. в одном из слоев могут накопиться более длинные строки, которые тормозят вычисления в остальных слоях.

Из выше сказанного следует, что при работе с несколькими параллельными слоями необходимо в сортировке матрицы и ее последующем распределении отталкиваться от количества слоев.

Пусть у нас M слоев, в каждый из которых поступают строки матрицы по порядку 0,2 ..M; M+1, M+2,..2M и т.д., таким образом, каждый из слоев получит одинаковое количество строк в обработку, и могут возникнуть описанные выше проблемы с задержками (рис. 3.28).

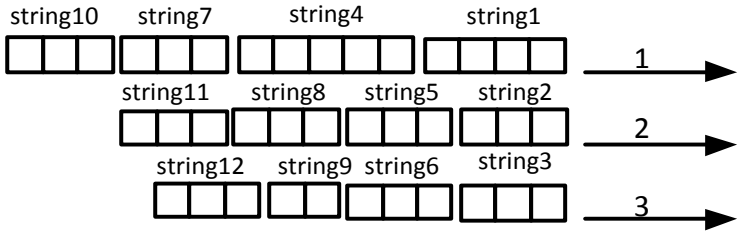


Рис. 3.28 – Поступление данных на слои в прямом порядке

Каждый слой имеет K – входных каналов, данные с которых поступают на вычислители. Для некоторых строк пакет данных имеет длину большую, чем количество каналов K, и будет подаваться на обработку за 2 такта. В это время слой, получивший более короткие строки, сможет принять в обработку 2 строки. Отсюда следует, что необходимо распределять строки на обработку между слоями не по порядку, а согласно занятости слоя.

При подготовке данных к загрузке матрица разделяется на подматрицы, которые будут подаваться в параллельную обработку. Для этого при чтении строк матрицы необходимо ввести признак занятости слоя, который покажет, как быстро освободится для приема следующих данных конкретный слой. В первый проход в каждый слой загружаются строки по порядку, затем производится

анализ занятости слоев. Занятые в следующий такт слои не получат новые строки, пока не освободятся (рис. 3.29).

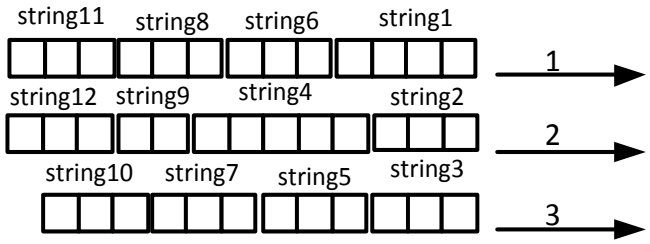


Рис. 3.29 – Поступление данных на слои после пересортировки

Такой способ распределения строк между слоями обеспечивает увеличение скорости работы конвейера на 5-7%, работает для любых распределений ненулевых элементов в СЛАУ, полученных при дискретизации задачи супердиффузии на адаптивных сетках.

Примеры времени работы конвейера приведены в таблице 3.4 [137].

Таблица 3.4 Время работы конвейера для матриц с естественным порядком и отсортированных

СЛАУ	Количество тактов работы
M_1 прямой порядок	$374*10^9$
M_1 оптимизированный порядок	$343*10^9$
M_2 прямой порядок	$622*10^9$
M_2 оптимизированный порядок	$518*10^9$
M_3 прямой порядок	$547*10^9$
M_3 оптимизированный порядок	$510*10^9$

3.7 Выводы

1) Стандартные подходы к решению СЛАУ малоэффективны, если СЛАУ содержит матрицу с нерегулярной структурой, как в случае дискретизации математической модели, описывающей супердиффузию газа, на адаптивных

сетках. Решение ее на реконфигурируемых вычислительных системах известными методами не эффективны и ведут к простою оборудования и излишним его расходам.

2) Разработан метод создания параллельно-конвейерных программ для РВС на основе принципа распараллеливания по слоям и итерациям для эффективной обработки фрактальных структур. При синтезе программ приложенным методом учитываются особенности решаемых СЛАУ, и на основе таких параметров как математическое ожидание и среднеквадратичное отклонение количества элементов в строке матрицы определяется оптимальные параметры вычислительной структуры. Такой подход позволяет увеличить реальную производительность вычислительной системы на 40% по сравнению с классическими.

3) Разработан метод предварительной сортировки строк решаемой СЛАУ, выполняемой на этапе подготовки данных к загрузке в РВС. Строки матрицы загружаются не в естественном порядке, а исходя и загруженности вычислительного слоя, благодаря чему сохраняется естественный порядок получения новых значений неизвестных, и уменьшаются задержки вычислений в конвейере, обусловленные наличием информационных связей между строками СЛАУ. Таким образом, достигается равномерная загрузка всех вычислительных слоев и уменьшаются простои оборудования., и скорость работы конвейера увеличивается на 5-7% по сравнению с подачей неподготовленной СЛАУ.

ЗАКЛЮЧЕНИЕ

Основной результат диссертационной работы заключается в разработке методов и средств обработки самоподобных структур для реализации фрактальных алгоритмов на высокопроизводительных реконфигурируемых вычислительных системах, позволяющих повысить их производительность.

При проведении исследований и разработок в диссертации получены следующие теоретические и прикладные результаты:

- разработан метод решения на РВС параллельно-конвейерным способом задачи фрактального сжатия изображений, отличающийся от известных побитовой обработкой данных, обеспечивающей максимальное задействование вычислительного ресурса системы, и сортировкой структур, содержащих выходные данные, по номеру рангового блока, что позволяет получить ускорение при выполнении на РВС в 15000 раз по сравнению с аналогичной реализацией для многоядерного универсального процессора и в 10-40 раз по сравнению с выполнением на устройстве, содержащем одну ПЛИС. Масштабирование данного метода обеспечит близкий к линейному рост производительности при увеличении вычислительного ресурса.

- разработан метод решения на РВС параллельно-конвейерным способом задачи декомпрессии сжатых изображений, отличающийся от известных использованием одинарной косвенной адресации для блоков памяти, содержащих доменные блоки. Данный метод позволяет получить на РВС ускорение более чем в 50 раз по сравнению с аналогичной реализацией для многоядерного универсального процессора.

- разработан метод синтеза вычислительной структуры для решения на РВС задачи распространения газа во фрактальной среде, отличающийся от известных возможностью оптимизации вычислительной структуры для конкретной СЛАУ на основе оценки ее параметров и использующий подобные подграфы, обеспечивающий увеличение эффективности на 40% по сравнению с классическими способами решения СЛАУ на РВС.

- разработаны программные средства на языке программирования COLAMO для решения на РВС задач фрактального сжатия и декомпрессии изображений и задачи супердиффузии (на программные продукты получены свидетельства о государственной регистрации программ для ЭВМ [102, 105, 137]).

Разработанные при решении научной задачи методы и средства позволили достигнуть поставленной цели, заключающейся в повышении реальной производительности РВС при решении задач фрактального типа, что подтверждено как теоретическими выкладками, так и экспериментальными исследованиями.

Основные научные результаты диссертации опубликованы в работах [93, 100, 102, 104, 105, 112, 123, 124, 126, 127, 129, 134, 135, 136, 137].

Предложенные в диссертации научные результаты аргументированы и оценены по сравнению с известными работами в области создания программного обеспечения высокопроизводительных реконфигурируемых вычислительных систем.

Таким образом, в диссертации решена актуальная научная задача, которая обладает новизной и имеет заметное научно-практическое значение для программирования РВС. Внедрение полученных в диссертации результатов может внести существенный вклад в повышение производительности высокопроизводительных реконфигурируемых вычислительных систем при обработке самоподобных структур.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Cohen, N. Fractal Antennas: Part 1/ N. Cohen // Communications Quarterly, Summer 1995. – 1995. – p. 7–22.

2. Cohen, N. Self-similarity and the geometric requirements for frequency independence in antennae / N. Cohen, R. G. Hohlfeld // Fractals, 1999. – Vol. 7, No. 1. – pp. 79-84.

3. Cook, R. L. The Reyes Image Rendering Architecture. Computer Graphics/ R. L. Cook, L. Carpenter, E. Catmull // ACM SIGGRAPH Computer Graphics, 1987. – N. 4, V. 21. – pp. 95-102.

4. Mayfield, W.D. Fractal Art Generation using GPUs / W.D. Mayfield, J.C. Eiland, T.J. Hutyra, M.C. Paulsen, B.M. Wyatt // 2016. 8 Nov. URL:https://arxiv.org/ftp/arxiv/papers/1611/1611.03079.pdf (дата обращения 15.03.2021).

5. Briggs, J. Fractals: The Patterns of Chaos: a New Aesthetic of Art, Science, and Nature / J. Briggs – Simon and Schuster, 2014. – p. 84.

6. Короткина, М.Р. Фрактальность пространства и времени / М.Р. Короткина // Лесной вестник (1997-2002), 2000. – № 2. – С. 158–162.

7. MacIntosh, A.J.J. The fractal primate: interdisciplinary science and the math behind the monkey / A.J.J. MacIntosh // Primate Research. – 2014. – V. 30(1), Jan. – pp. 95-119.

8. Петерс, Э. Фрактальный анализ финансовых рынков: применение теории хаоса в инвестициях и экономике/ Пер. с англ.- М.: Интернет-трейдинг, 2004. – 304 с.

9. Мандельброт, Б. (Не)послушные рынки: фрактальная революция в финансах / Б. Мандельброт, Ричард Л. Хадсон // Пер. с англ. – М.: Вильямс, 2006. – 400 с.

10. Морозов, Е. Почему частоты процессоров не растут выше нескольких гигагерц / Е. Морозов // 2017. – 26 мая.

URL:https://www.iguides.ru/main/other/pochemu_chastoty_protsessorov_ne_rastut_vyshe_neskolkikh_gigagerts/ (дата обращения 12.04.2021).

11. Кулбаев, С.С. Оценка эффективности сервиса сжатия цифровых изображений на высокопроизводительной вычислительной системе / С.С. Кулбаев, А.А. Немеров, А.С. Крупский // Доклады ТУСУР. – 2013. – № 4(30). – С. 142–146.

12. Chen, D. Fractal Video Compression in OpenCL: An Evaluation of CPUs, GPUs, and FPGAs as Acceleration Platforms / D. Chen, D. Singh // 18th Asia and South Pacific Design Automation Conference 2013, IEEE, Yokohama, Japan, 22-25 Jan., 2013. – pp. 297-304.

13. Saad, A-M.H.Y. High-speed implementation of fractal image compression in low cost FPGA / A-M.H.Y. Saad, M.Z. Abdullah // Microprocessors and Microsystems. – 2016. – V. 46 (part B). – pp. 429-440.

14. Saad, A.-M.H.Y. High-Speed Fractal Image Compression Featuring Deep Data Pipelining Strategy / A.-M.H.Y. Saad, M.Z. Abdullah // IEEE Access – 2018. – v. 10. – pp. 9-24. DOI: 10.1109/ACCESS.2018.2880480.

15. Мандельброт, Б. Фрактальная геометрия природы/ Пер. с англ. А.Р. Логунова – Москва: Институт компьютерных исследований, 2002. – 656 с.

16. Hardy, G. H. Weierstrass’s nondifferentiable function / G.H. Hardy // Trans — Amer. Math. Soc, 1916. – V. 17. – pp. 301-325.

17. Koch, H. Une méthode géométrique élémentaire pour l’étude de certaines questions de la théorie des courbes planes / H. Koch //Acta Mathematica,1906. – Band 30. – pp. 145-174.

18. Edgar, G.A. Classics on Fractals / G.A. Edgar. – NY Routledge, 2018. – 365 p.

19. Julia, G. Mémoire sur l'iteration des fonctions rationnelles / G. Julia // Journal de Mathématiques Pures et Appliquées, 1918. – V. 8. – pp. 47-245.

20. Van Lawick van Pabst, J. Dynamic Terrain Generation Based on Multifractal Techniques / J. Van Lawick van Pabst, H. Jense // 2001.

URL:https://web.archive.org/web/20110724160327/http://www.lawick.nl/publications/paperft.pdf (дата обращения 15.04.2021).

21. Cribb, N. Changes in the behavioural complexity of bottlenose dolphins along a gradient of anthropogenically-impacted environments in South Australian coastal waters: Implications for conservation and management strategies / N. Cribb; L. Seuront. // Journal of Experimental Marine Biology and Ecology 482: Sep. – 2016. – pp. 118-127.

22. Karaboikis, M. A Printed Folded Koch Monopole Antenna for Wireless Devices / G. Tsachtsiris, C. Soras, M. Karaboikis, V. Makios // Microwave and Optical Technology Letters. – 2004. – v.40. – № 5. – p.374–378.

23. Salim, N. Comparison study of fractal and slot antennas challenges for cellular band communications / N. Salim, Mohammed Omar Ali // International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING. – 2023. – V. 11, № 2. – pp. 478-483.

24. Nichita, M.-V. On the 5G Communications: Fractal-Shaped Antennas for PPDR Applications / M.-V. Nichita, M.-A. Paun, V.-A. Paun, V.-P. Paun // Complexity. – 2021. – Special Issue: Nonlinear Dynamics of Complex Systems. – pp. 1-12. DOI: 10.1155/2021/9451730.

25. Fisher, Y. Fractal Image Compression: Theory and Application / Y. Fisher // Springer-Verlag, New York, 1995. – 342 p.

26. Barnsley, M., Hurd, L.P. Fractal Image Compression / M. Barnsley, L.P. Hurd; A.K. Peters Ltd. – Wellsley Massachusetts, 1993. – 253 p.

27. Sayood, Khalid. Introduction to Data Compression / Khalid Sayood. – Third Edition. –Morgan Kaufmann Publishers, 2006. – 680 p.

28. Woon, W.M. Achieving High Data Compression of Self-similar Satellite Images using Fractal / W.M. Woon; A.T. Shuen Ho; T. Yu; S.C. Tam; S.C. Tan; L.T. Yap // IEEE 2000 International Geoscience and Remote Sensing Symposium. Taking the Pulse of the Planet: The Role of Remote Sensing in Managing the Environment.

Proceedings, Geoscience and Remote Sensing Symposium paper, IGARSS. –2000. – №2. – pp. 609–611.

29. Rieu, M. Fractal Fragmentation, Soil Porosity, and Soil Water Properties: I. Theory / M. Rieu, G. Sposito // SOIL SCI. SOC. AM. J., – 1991. 1 Sept. – v. 55. – pp. 1231-1241.

30. Zhou, A. Fractal Model to Interpret Porosity-Dependent Hydraulic Properties for Unsaturated Soils / A. Zhou, Y. Fan, W.-C. Cheng, J. Zhang // Advances in Civil Engineering. – 2019. – V. 2019, Article ID 3965803. – p. 1-13.

31. Кащенко, Н.М. Моделирование эффектов аномальной диффузии для дренажных систем / Н.М. Кащенко, М.А. Никитин // Матем. Моделирование. – 2013. – Т. 25. – № 12 – С. 44–49.

32. Паровик, Р.И. Модель нестационарной диффузии-адвекции радона в системе грунт-атмосфера / Р.И. Паровик // Вестник КРАУНЦ. Физ.-мат. науки. – 2010. – № 1(1). – С. 39–45.

33. Guan, X. Fractal Analysis and Classification of Pore Structures of High-Rank Coal in Qinshui Basin, China / D. Zhao, Y. Guo, G. Wang, X. Guan, X. Zhou, J. Liu // Energies. – 2022. – № 15. – pp. 1-22. DOI: 10.3390/en15186766.

34. Bouchendouka, A. A Generalization of Poiseuille's Law for the Flow of a Self-Similar (Fractal) Fluid through a Tube Having a Fractal Rough Surface / A. Bouchendouka, Z.E.A. Fellah, Z. Larbi, N.O. Ongwen, E. Ogam, M. Fellah, C. Depollier // Fractal and Fractional. – 2023. – v. 7, № 61. – pp. 1-17. DOI: 10.3390/fractalfract7010061.

35. Смирнов, Б.М. Фрактальный клубок – новое состояние вещества / Б.М. Смирнов // Успехи физических наук, Август 1991. – Т. 161, № 8. – с. 141-153.

36. Жуков, А.Л. Моделирование экстремальных природных процессов с использованием ГИС-технологий / А.Л. Жуков // Сборник научных трудов СЕВКАВГИПРОВОДХОЗ. – 2009. – В. 18. – с. 83-84.

37. Mehdi, I. Effect of wet ball milling on copper ore flotation by fractal geometry / I. Moradi, M. Irannajad // Separation Science and Technology. – 2023. – № 58(9). – pp. 1-18. DOI: 10.1080/01496395.2023.2189063.

38. Гелашвили, Д.Б. Фракталы и мультифракталы в биоэкологи : монография / Д.Б. Гелашвили, Д.И. Иудин, Г.С. Розенберг, В.Н. Якимов, Л.А. Солнцев. // Нижний Новгород : Издательство Нижегородского госуниверситета, 2013. – С. 373.

39. Karaagac, B. A computational technique for the Caputo fractal-fractional diabetes mellitus model without genetic factors / B. Karaagac, K.M. Owolabi, E. Pindza // International Journal of Dynamics and Control. – 2023. – № 3. – pp. 1-18. DOI: 10.1007/s40435-023-01131-7.

40. Jung, W. Fractal Kinetic Implementation in Population Pharmacokinetic Modeling / W. Jung, H. Ryu, J. Chae, H. Yun // Pharmaceutics. Special Issue: The Role of Pharmacometrics in Drug Discovery and Development Process (Volume II). – 2023. – № 15(1). – pp. 1-13. DOI: 10.3390/pharmaceutics15010304.

41. Александровская, Ю.П. Использование фрактальных методов для анализа финансовых рядов / Ю.П. Александровская // Вестник технологического университета. – 2014. – т.17, в.18. – с.257-262.

42. Faes, C. Fractal dimension based geographical clustering of COVID19 time series data / Y.A. Natalia, C. Faes, T. Neyens, P. Chys, N. Hammami, G. Molenberghs // 2022. – v. 1. – pp. 1-10. DOI:10.21203/rs.3.rs-1901254/v1.

43. Brooks, R. The Dynamics of 2-Generator Subgroups of PSL(2, ℂ) / R. Brooks, P.J. Matelski // Riemann Surfaces Related Topics (AM-97), Proceedings of the 1978 Stony Brook Conference. (AM-97), Princeton: Princeton University Press. – 1981. – V. 97. – pp. 65-72. DOI: 10.1515/9781400881550-007.

44. Беклемишев, Д.В. Курс аналитической геометрии и линейной алгебры / Д.В. Беклемишев. – М: Высш. шк.,1998. – 320 с.

45. Мандельброт, Б. Фракталы и Хаос. Множество Мандельброта и другие чудеса / Б. Мандельброт. // М. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2009. – 392 с.

46. Механов, В.Б. Особенности архитектуры универсальных микропроцессоров: учебное пособие / В.Б. Механов. – Пенза : Изд-во ПГУ, 2010. – 176 с.

47. Intel | Data Center Solutions, IoT, and PC Innovation : сайт. – URL: http://www.intel.com (дата обращения: 20.03.2021).

48. Бойченко, И.В. Эксперимент по фрактальному сжатию RGB-изображений на вычислительном кластере / И.В. Бойченко, С.С. Кулбаев, А.А. Немеров, В.В. Голенков // Известия Томского политехнического университета. – 2012. – Т. 321. – № 5. – с. 87-92.

49. Бухановский, А.В. Проектирование прикладного математического обеспечения параллельной обработки данных / А.В. Бухановский, С.В. Иванов // Учебные материалы Зимней школы-практикума «Технологии параллельного программирования, 2006». – 38 с. URL: http://window.edu.ru/resource/365/62365/files/book-2006.pdf (дата обращения 12.04.2021).

50. DDR5 will boost bandwidth and lower power consumption – The Tech Report : сайт. – URL: https://techreport.com/news/31673/ddr5-will-boost-bandwidth-and-lower-power-consumption (дата обращения: 30.09.2022).

51. Chauhan, M.S. Fractal Image Compression using Dynamically Pipelined GPU Clusters / M.S. Chauhan, A. Negi, P.S. Rana // Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012). Advances in Intelligent Systems and Computing. – 2014. – V. 236. – pp. 1317-1331.

52. Ismail, B.M. Cuckoo inspired fast search algorithm for fractal image encoding / B.M. Ismail, B.E. Reddy, T.B. Reddy // Journal of King Saud University – Computer and Information Sciences. – 2016. – № 30(4). – pp. 462-469.

53. Haque, E. GPU Accelerated Fractal Image Compression for Medical Imaging in Parallel Computing Platform / E. Haque, A.A. Kaisan, M.R. Saniat, A. Rahman // arXiv:1404.0774. – 2014. – v1. – pp. 1-7.

54. Chauhan, M.S. Fractals Image Rendering and Compression using GPUs / M.S. Chauhan, A. Negi // International Journal of Digital Information and Wireless Communications (IJDIWC) 2(1): 1-6 The Society of Digital Information and Wireless Communications. – 2012. – pp. 813-818.

55. AMD | together we advance_ : сайт: – URL: https://www.amd.com (дата обращения: 20.03.2021).

56. Мировой лидер в области вычислений для искусственного интеллекта | NVIDIA : сайт: – URL: https://www.nvidia.com (дата обращения: 20.03.2021).

57. Графический вызов суперкомпьютерам | Открытые системы. СУБД | Издательство «Открытые системы» : сайт: – URL: https://www.osp.ru/os/2008/04/5114497 (дата обращения: 21.04.2021).

58. Erra, U. Toward real time fractal image compression using graphics hardware / U. Erra // ISVC 2005: Advances in Visual Computing. – 2005. – v. 3804. – pp. 723–728.

59. Habibi, M. Real time fractal image coder based on characteristic vector matching / M. Habibi, S. Samavi, S. Shirani, N. Rowshanbin // Image and Vision Computing. – 2010. – v. 28, i. 11. – pp. 1557–1568. DOI: 10.1016/j.imavis.2010.03.011.

60. Jackson, D.J. A hardware architecture for real-time image compression using a searchless fractal image coding method / D.J. Jackson, H. Ren, X. Wu, K.G. Ricks // Journal of Real-Time Image Processing. – 2007. – v. 1, № 3. – pp. 225–237.

61. Dhar, A.S. Low-delay parallel architecture for fractal image compression / M. Panigrahy, I. Chakrabarti, A.S. Dhar // Circuits, Systems, and Signal Processing. – 2016. – v. 35, № 3. – pp. 897–917.

62. Kaisan, A.A. GPU accelerated fractal image compression for medical imaging in parallel computing platform / M.E. Haque, A.A. Kaisan, M.R. Saniat, A.

Rahman // Computing Research Repository. – 2014. – v. 1404.0774. – pp. 1-7. DOI: 10.48550/arXiv.1404.0774.

63. Costa, C.A. Exploiting data-parallelism on multicore and SMT systems for implementing the fractal image compressing problem / R. da Rosa Righi, V.F. Rodrigues, C.A. Costa, R.Q. Gomes // Computer and Information Science. – 2016. – v. 10, № 1. – p. 34-47. DOI: 10.5539/cis.v10n1p34.

64. Ismail, B.M. Cuckoo inspired fast search algorithm for fractal image encoding / B.M. Ismail, B.E. Reddy, T.B. Reddy // Journal of King Saud University – Computer and Information Sciences. – 2018. – v. 30, № 4. – pp. 462–469. DOI: 10.1016/j.jksuci.2016.11.003.

65. Грушвицкий, Р.И. Проектирование систем на микросхемах с программируемой структурой. 2-е изд. / Р.И. Грушвицкий, А.Х. Мурсаев, Е.П. Угрюмов // СПб.: «БХВ-Петербург», 2008. – 608 с.

66. Программируемая пользователем вентильная матрица – Википедия : сайт: – URL: https://ru.wikipedia.org/wiki/FPGA (дата обращения: 14.04.2021).

67. Попов, А.Ю. Проектирование цифровых устройств с использованием ПЛИС: Учеб. Пособие / А.Ю. Попов // – М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. – 80 с. – ISBN 978-5-7038-3317-9.

68. Xilinx – Adaptable. Intelligent | together we advance_: сайт. – URL: http://www.xilinx.com (дата обращения: 14.05.2021).

69. Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров | НИЦ супер-ЭВМ и нейрокомпьютеров: сайт. – URL: http://www.superevm.ru (дата обращения: 14.05.2021).

70. Сайт НИИ МВС ЮФУ: сайт. – URL: http://www.mvs.sfedu.ru (дата обращения: 14.05.2021).

71. НПО «РОСТ» : сайт. – URL: https://ristnvo.com (дата обращения: 14.01.2022).

72. ФГУП «НИИ «КВАНТ»: сайт. – URL: http://www.rdi-kvant.ru (дата обращения: 02.03.2021).

73. Каляев, А.В. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений / А.В. Каляев, И.И. Левин. – Москва: Изд-во "Янус-К", 2003. – 380 с.

74. Доронченко, Ю.И. Перспективные высокопроизводительные реконфигурируемые вычислители с иммерсионным охлаждением / И.И. Левин, А.М. Федоров, Ю.И. Доронченко, М.К. Раскладкин // Известия ЮФУ. Технические науки – 2020. – №7. – С. 6-19.

75. «Разработка научно-технических основ создания многопроцессорных вычислительных систем сверхпетафлопсной производительности и подготовка кадров высшей квалификации в области распределенных вычислений», отчет о НИР, № гос. рег. 01200958384, инв. 05.НОЦ.2011, Таганрог, НИИ МВС ЮФУ, шифр «2009-1.1-215-002-013», 2011 г.

76. «Разработка реконфигурируемой вычислительной системы РВС-7 и организация на ее основе производства реконфигурируемых вычислительных систем с производительностью до 1015 операций в секунду в одностоечном конструктиве 47U», отчет об ОКР, № гос. рег. 49-15/259, Таганрог, НИИ МВС ЮФУ, шифр «Плеяда», 2013 г.

77. Son, T.N. Implementation of Fractal image compression on FPGA/ T.N. Son, O.M. Hung, D.T. Xuan, V.L. Tran, N.T. Dzung, T.M. Hoang // 4th International Conference on Communications and Electronics ICCE 2012, online IEEExplorer, 2012, 1-3 Aug. – pp. 339 – 344.

78. Son, T. N. Efficient implementation of a fractal color image compression on FPGA/ T. N. Son, O. M. Hung, D. T. Xuan, V. L. Tran, N. T. Dzung, T. M. Hoang // 2013 International Conference of Soft Computing and Pattern Recognition (SoCPaR). – 2013. – pp. 184-189.

79. Левин, И.И. Решение задачи LU-декомпозиции на реконфигурируемых вычислительных системах: оценка и перспективы / И.И. Левин, А.В. Пелипец, Д.А. Сорокин // Известия ЮФУ. Технические науки. – 2015. – № 7 (168). – С. 62-70.

80. Padmavati, S. FPGA Implementation for Fractal Quadtree Image Compression / S. Padmavati, M. Vaibhav // International Journal of Computer Sciences and Engineering. – 2018. – V. 6. – I. 10, Oct. – pp. 405-409.

81. Cai, H. A novel multiwing chaotic system with FPGA implementation and application in image encryption / H. Cai, J. Sun, Z. Gao, H. Zhang // Journal of Real-Time Image Processing. – 2000. – № 19. – pp. 775-790.

82. Amirtharajan, R. Optimal concurrency on FPGA for lightweight medical image encryption / V. Raj, S. Janakiraman, R. Amirtharajan // Journal of Intelligent and Fuzzy Systems. – 2021. – № 40(6). – pp. 10385–10400. DOI: 10.3233/JIFS-200203.

83. Gafsi, M. FPGA implementation of improved security approach for medical image encryption and decryption / A. Hafsa, M. Gafsi, J. Malek, M. Machhout, A. Vitiello // Scientific Programming. – 2021. – № 2. – pp. 1–20. DOI: 10.1155/2021/6610655.

84. Kishore, B. FPGA based simple and fast JPEG encryptor / B. Kishore, B.K. Shreyamsha Kumar, C.R. Patil // Journal of Real-Time Image Processing. – 2012. – № 10(3). – pp. 551–559. DOI: 10.1007/s11554-012-0282-5.

85. Son, T. N. Fast FPGA Implementation of YUV-based Fractal Image Compression/ T. N. Son T. M. Hoang, N. T. Dzung, N. H. Giang //2014 IEEE Fifth International Conference on Communications and Electronics (ICCE). – 2014. – pp. 440-445.

86. Ramirez, A.M. An architecture for fractal image compression using quad-tree multiresolution / A.M. Ramirez, A.D. Sanchez, M.L. Aranda, J.V. Pineda // IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512). – 2004. – pp. 878-897. doi: 10.1109/ISCAS.2004.1329417.

87. Alzeidi, A.A.S.N. CUDA implementation of fractal image compression / A.A.S.N. Alzeidi, M.A. Hammoshi, M.S. Chauhan, G. AlFarsi // Journal of Real-Time Image Processing. – 2019. – 3 July. – pp. 1375-1387.

88. Левин, И.И. Обработка сетевых потоков данных в реконфигурируемых вычислительных системах / И.И. Левин, А.С. Котляров // Известия ЮФУ. Технические науки. – Ростов/Д: Изд-во ЮФУ. – 2019. – №2.

89. Касаркин, А.В. Структурная реализация задачи нахождения всех максимальных клик графа на реконфигурируемых вычислительных системах / А.В. Касаркин, И.И. Левин // Вестник компьютерных и информационных технологий. – 2018. – № 10 (172). – С. 3-10.

90. Левин, И.И. Структурная реализация задачи нахождения всех максимальных клик графа на реконфигурируемых вычислительных системах / И.И. Левин, А.В. Касаркин // Вестник компьютерных и информационных технологий. – М.: Машиностроение, 2018. – № 10. – С. 3-10.

91. Касаркин, А.В. Метод решения графовых NP-полных задач на реконфигурируемых вычислительных системах на основе принципа распараллеливания по итерациям / А.В. Касаркин // Известия ЮФУ. Технические науки. – Ростов/Д: Изд-во ЮФУ. – 2020. – № 7(217). – С. 121–129.

92. Богданов, А.Ф. Фрактальное сжатие сигналов в многопозиционных спутниковых системах / А.Ф. Богданов, Н.А. Потемкин // РЕШЕТНЕВСКИЕ ЧТЕНИЯ. – 2016. – т. 1. – с. 250-252.

93. Чекина, М.Д. Параллельно-конвейерная реализация фрактального сжатия изображений на реконфигурируемых вычислительных системах // М.Д. Чекина, И.И. Левин / Вестник компьютерных и информационных технологий. – 2021. – Т. 18. – № 4 (202). – С. 37-44. DOI: 10.14489/vkit.2021.04.pp.037-044.

94. Чекина, М.Д. «Методы и средства обработки фракталов на реконфигурируемых вычислительных системах» : Направление 01.04.02 «Прикладная математика и информатика», образовательная программа «Прикладная математика для высокопроизводительных вычислительных систем» : выпускная квалификационная работа (диссертация на звание магистра техники и технологии) / Чекина Мария Дмитриевна; Южный Федеральный Университет. – Таганрог, 2020. – 122 с.

95. Bina, T.C. Architecture for fractal image compression / D. Vidya, R. Parthasarathy, T.C. Bina, N.G. Swaroopa // Journal of Systems Architecture. – 2000. – v. 46, i. 14. – pp. 1275–1291. DOI: 10.1016/S1383-7621(00)00018-7.

96. Кудрина, М.А. Исследование вариантов реализации и методов ускорения фрактального сжатия изображений / М.А. Кудрина, В.Б. Сахибназарова // Труды международного симпозиума "Надежность и качество". – 2018. – т. 1. – с. 333-337.

97. Фрактальное сжатие изображений : сайт. – URL: http://www.100byte.ru/stdntswrks/vrncd/vrncd.html (дата обращения: 10.01.2022).

98. Осокин, А.Н. Быстродействующий алгоритм фрактального сжатия изображений / М.П. Шарабайко, А.Н. Осокин // Известия Томского политехнического университета. – 2011. – т. 318, № 5. – с. 52-57.

99. Левин, И.И. Методы редукции вычислений для программирования гибридных реконфигурируемых вычислительных систем / А.И. Дордопуло, И.И. Левин // Материалы XII мультиконференции по проблемам управления (МКПУ-2019): в 4 т. – 2019. – т. 3. – С. 78-82.

100. Чекина, М.Д. Реализация фрактальных алгоритмов на реконфигурируемых вычислительных системах // Суперкомпьютерные технологии: сборник трудов молодых ученых; Южный Федеральный Университет. – Ростов-на-Дону; Таганрог, 2020. – с. 129-133.

101. Гудков, В.А. Организация битовой обработки данных для реконфигурируемых вычислительных систем на языке программирования высокого уровня / В.А. Гудков, И.И. Левин, Е.Е. Семерникова // Вестник компьютерных и информационных технологий. – М.: Машиностроение, 2015. – №5. – С. 3–9.

102. Свидетельство о государственной регистрации программ для ЭВМ № 2023614463, РФ. Программа фрактального сжатия изображений // Чекина М.Д., Левин И.И. Зарегистр. в Реестре программ для ЭВМ 01.03.2023 г. Правообладатель: ООО «НИЦ супер-ЭВМ и нейрокомпьютеров».

103. Колмогоров, А.Н. Элементы теории функций и функционального анализа / А.Н. Колмогоров, С.В. Фомин – 4-е изд. – М.: Наука, 1976. – 544 с.

104. Чекина, М.Д. Реализация фрактального сжатия и декомпрессии изображений параллельно-конвейерным способом на реконфигурируемых вычислительных системах // Известия ЮФУ. Технические науки. – 2020. – № 7 (217). – С. 130-142. DOI: 10.18522/2311-3103-2020-7-130-142.

105. Свидетельство о государственной регистрации программ для ЭВМ № 2023617300, РФ. Программа декомпрессии изображений, сжатых фрактальным способом // Чекина М.Д., Левин И.И. Зарегистр. в Реестре программ для ЭВМ 07.04.2023 г. Правообладатель: ООО «НИЦ супер-ЭВМ и нейрокомпьютеров».

106. Нахушева, В.А. Дифференциальные уравнения математических моделей нелокальных процессов / В.А. Нахушева // М.: Наука, 2006. – 173 с.

107. Zaslavsky, G.M. Chaos, fractional kinetics, and anomalous transport / G.M. Zaslavsky // Physics Reports. – 2002. – Т. 371. – P. 461–580.

108. Крылов, С.С. Фракталы в геофизике / С.С. Крылов, Н.Ю. Бобров // СПб.: Издательство С-Пб. Университета, 2004. – 138 с.

109. Нахушев, А.М. Дробное исчисление и его применение / А.М. Нахушев // М.: Физматлит, 2003. – 272 с.

110. Грачев, В.И. Фрактальные лабиринты / В.И. Грачев, А.А. Потапов, В.А. Потапов // Радиоэлектроника. Наносистемы. Информационные технологии. – 2011. – т. 3, № 2. – с. 103-109.

111. Такенова, Ф.И. Разностные методы решения краевых задач для дифференциальных уравнений дробного порядка / М.Х. Шхануков-Лафишев, Ф.И. Такенова // Журнал вычислительной математики и математической физики. – 2006. – Т. 46, №10. – С. 1871-1881.

112. Чекина, М.Д. Математическое моделирование задач геофильтрации в почвогрунтах с фрактальной структурой на многопроцессорных вычислительных системах // Известия ЮФУ. Технические науки. – 2014. – № 12 (161). – С. 242-251.

113. Чекина, М.Д. Математическое моделирование задач геофильтрации в почвогрунтах с фрактальной структурой на многопроцессорных вычислительных системах // Материалы 3-й Всероссийской научно-технической конференции «Суперкомпьютерные технологии (СКТ-2014)». - Ростов/Д: Изд-во ЮФУ, 2014. - С. 258-262.

114. Левин, И.И. Эффективная реализация распараллеливания на реконфигурируемых системах / И.И. Левин, А.В. Пелипец // Вестник компьютерных и информационных технологий. – М.: Машиностроение. – 2018. – № 8. – С. 11–16.

115. Левин, И.И. Решение задачи LU-декомпозиции на реконфигурируемых вычислительных системах: оценка и перспективы / И.И. Левин, А.В. Пелипец, Д.А. Сорокин // Известия ЮФУ. Технические науки. – 2015. – № 7 (168). – С. 62-70.

116. Паровик, Р.И. Процессы переноса радона в средах с фрактальной структурой / Р.И. Паровик, Б.М. Шевцов // Математическое моделирование. – 2009. – Т. 21. – № 8. – С. 30–36.

117. Литвинов, В.А. О влиянии способа аппроксимации неизвестной функции на устойчивость численных методов решения уравнения аномальной диффузии / В.А. Литвинов // Кибернетика и программирование. – 2019. – № 2. – С. 23-29. DOI: 10.25136/2306-4196.2019.2.29201.

118. Левин, И.И. Реконфигурируемые компьютеры на основе ПЛИС Xilinx Virtex Ultrascale / И.И. Левин, А.И. Дордопуло, Д.А. Сорокин, З.В. Каляев, Ю.И. Доронченко // Параллельные вычислительные технологии (ПаВТ'2019). Короткие статьи и описания плакатов XIII Международной научной конференции. – Челябинск: Издательский центр ЮУрГУ. – 2019. – С. 288-298.

119. Terekhov, K.M. Two-phase water flooding simulations on dynamic adaptive octree grids with two-point nonlinear fluxes / K.M. Terekhov, Yu.V. Vassilevski // Russian Journal of Numerical Analysis and Mathematical Modelling. – 2013. – V. 28, № 3. – P. 267–288.

120. Афендиков, А.Л. Метод адаптивных декартовых сеток для решения задач газовой динамики / А.Л. Афендиков, И.С. Меньшов, К.Д. Меркулов, П.В. Павлухин – Российская академия наук. – 2017. – С. 63.

121. Berger, M.J. Aspects (and Aspect Ratios) of Cartesian Mesh Methods / M.J. Berger, M.J. Aftosmis // Proceedings of the 16th International Conference on Numerical Methods in Fluid Dynamics, Springer-Verlag. – 1998. – pp. 1-12.

122. Сухинов, А.А. Построение декартовых сеток с динамической адаптацией к решению / А.А. Сухинов // Матем. моделирование. – 2010. – Т. 22. – № 1. – С. 86–98.

123. Чекина, М.Д. Решение разреженных СЛАУ большой и сверхбольшой размерности многосеточным методом на РВС // М.Д. Чекина, А.В. Подопригора / Известия ЮФУ. Технические науки. – 2018. – № 8 (202). – С. 212-221. DOI: 10.23683/2311-3103-2018-8-212-221.

124. Чекина, М.Д. Решение больших и сверхбольших разреженных СЛАУ на реконфигурируемых вычислительных системах // М.Д. Чекина, А.В. Подопригора / Суперкомпьютерные технологии (СКТ-2018). Материалы 5-й Всероссийской научно-технической конференции. – 2018. – Т. 1. – С. 201-204.

125. Самарский, А.А. Методы решения сеточных уравнений / А.А. Самарский, Е.С. Николаев // М.: Наука, 1978. – 592 с.

126. Чекина, М.Д. Управление анализом сейсмической активности с применением модели супердиффузии радона на реконфигурируемых вычислительных системах // XIV Мультиконференция по проблемам управления (МКПУ-2021). Материалы XIV Всероссийской Мультиконференции по проблемам управления (МКПУ-2021): в 4 томах. – 2021 г. – Т. 2. – С. 278-280.

127. Чекина, М.Д. Особенности решения задачи супердиффузии с использованием адаптивных сеток на РВС // XVII ежегодная молодежная научная конференция «Наука и технологии Юга России». Материалы XVII Ежегодной молодежной научной конференции. – 2021 г. – С. 306.

128. Михайлов, Д.В. Преобразование некоторых видов последовательных информационных графов в параллельно-конвейерную форму / Д.В. Михайлов // Известия ЮФУ. Технические науки. – 2020. – №7 (217). – С. 78–93.

129. Чекина, М.Д. Особенности определения параметров для решения задачи супердиффузии с использованием адаптивных сеток на РВС // XVIII Ежегодная молодежная научная конференция «Наука Юга России: достижения и перспективы». Материалы XVIII ежегодной молодежной научной конференции. – 2022. – С. 267.

130. Левин, И.И. Средства программирования реконфигурируемых и гибридных вычислительных систем на основе ПЛИС / А.И. Дордопуло, В.А. Гудков, А.А. Гуленок, А.В. Бовкун, Г.А. Евстафьев, М.И. Беззубова // Параллельные вычислительные технологии (ПаВТ'2019). Короткие статьи и описания плакатов XIII Международной научной конференции. – 2019. – С. 299-312.

131. Иванов, А.И. Методы организации параллельно-конвейерных вычислений для решения расчетоемких задач / А.И. Иванов, П.М. Коновальчик // Информационные технологии. Москва. – 2004. – № 12. – С. 38-43.

132. Михайлов, Д.В. Представление графов с ассоциативными операциями на языке программирования SET@L / Д.В. Михайлов, И.И. Левин, А.И. Дордопуло, И.В. Писаренко // Известия ЮФУ. Технические науки. – 2020. – № 3 (213). – С. 98–109.

133. Каляев, И.А. Реконфигурируемые мультиконвейерные вычислительные структуры: монография / И.А. Каляев, И.И. Левин, Е.А. Семерников, В.И. Шмойлов; под общ. ред. И.А. Каляева. - 2-е изд., перераб. и доп. - Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.

134. Чекина, М.Д. Модификация реализации метода Якоби при моделировании супердиффузии радона на реконфигурируемых вычислительных системах // Известия ЮФУ. Технические науки. – 2021. – № 7 (224). – С. 198-206. DOI: 10.18522/2311-3103-2021-7-198-206.

135. Чекина, М.Д. Решение задачи супердиффузии на реконфигурируемых вычислительных системах // Всероссийская научно-техническая конференция «Многопроцессорные вычислительные системы» (МВУС-2022). Материалы всероссийской научно-технической конференции. – 2022. – С. 155-160.

136. Чекина, М.Д. Повышение эффективности решения задачи супердиффузии с использованием адаптивных сеток на РВС // Материалы XIX Ежегодной молодежной научной конференции «Достижения и перспективы научных исследований молодых ученых Юга России», 17–28 апреля 2023 г., Ростов-на-Дону: Издательство ЮНЦ РАН. – с. 295.

137. Свидетельство о государственной регистрации программ для ЭВМ № 2023617550, РФ. Программа для решения задачи супердиффузии // Чекина М.Д., Левин И.И. Зарегистр. в Реестре программ для ЭВМ 11.04.2023 г. Правообладатель: ООО «НИЦ супер-ЭВМ и нейрокомпьютеров».

ПРИЛОЖЕНИЕ 1

ОПИСАНИЕ ПРОГРАММНЫХ РЕАЛИЗАЦИЙ

ЧЕКИНА МАРИЯ ДМИТРИЕВНА

МЕТОДЫ И СРЕДСТВА ОБРАБОТКИ ФРАКТАЛОВ НА РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

2.3.5 - Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:

доктор технических наук, профессор **Левин И.И.**

Таганрог – 2023

Язык высокого уровня COLAMO дает возможность создавать программы, описывающие вычислительные структуры в рамках структурно-процедурной парадигмы вычислений. Описание параллельно или последовательно будут выполняться операторы при программировании на COLAMO задается неявно, т.е. в код программы не внедряются специальные директивы, указывающие какой именно участок кода будет выполняться параллельно.

При объявлении переменных указывается способ доступа к массиву данных: параллельный (тип Vector) или последовательный (тип Stream). При параллельном доступе используются отдельные каналы распределенной памяти для подачи элементов массива. При последовательном элементы массива идут один за другим в одном канале памяти. Такой подход обеспечивает возможность смены типа доступа к массиву без изменений кода программы, отвечающего за вычисления. При этом после таких изменений параллельно выполняемый участок может стать последовательным и наоборот.

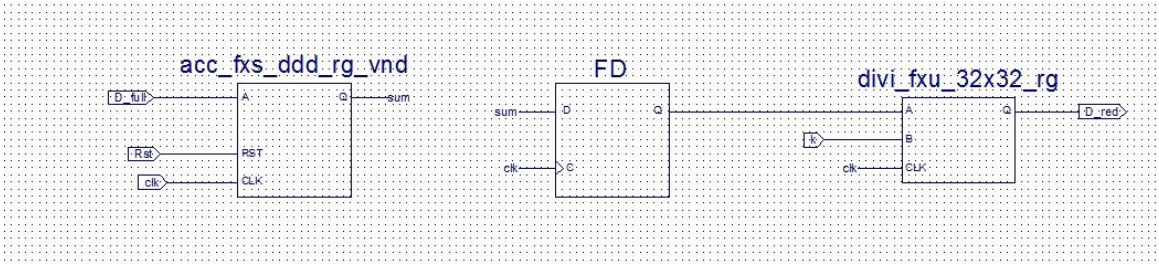
Для переменных в COLAMO также существуют несколько типов хранения:

- Mem - мемориальные переменные, хранятся во внешней распределенной памяти;
- Com – коммутационные переменные, применяются для информационных связей между вершинами информационного графа;
- Reg – регистровые переменные, в этом случае переменная хранится в аппаратно реализованном регистре;
- InterMem – внутренняя память, при этом переменные хранятся во внутренней памяти ПЛИС.

Фрактальное сжатие реализовано на языке COLAMO. Каждый вычислительный блок вынесен в отдельный подкадр (SubCadr), который является аналогом подпрограммы. На рисунке П.1.1 представлен текст подкадра (а) и синтезированная вычислительная структура на его основе, выполняющая операцию сжатия доменного блока до размеров рангового.

```
SubCadr W(In: D_full, k; Out: D_red);
Var D_full: array bit8 [N1: Stream] Com;
Var k: bit8 Com;
Var SumD: bit8 Com;
Var D_red: array bit8 Com;
Var i_w: Number;
for i_w:=0 to N1-1 do
    SumD := ACC(D_full[i_w]);
D_red := SumD/k;
EndSubCadr;
```

а)



б)

Рис. П.1.1 Текст (а) подкадра W и его вычислительная структура (б)

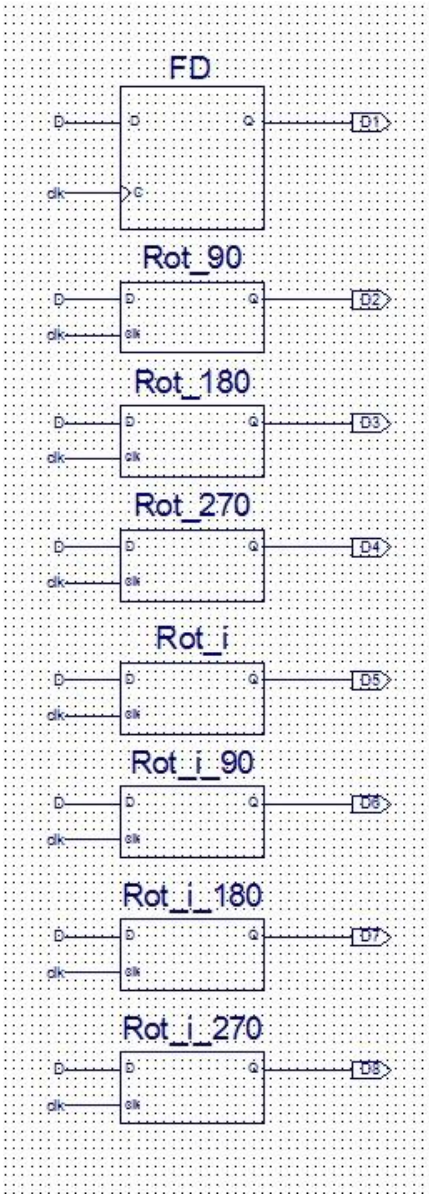
В подкадре F выполняются операции ротации сжатых доменных блоков, всего на выходе получается 8 положений и соответственно 8 потоков данных. Синтезированная вычислительная структура и текст подкадра представлены на рисунке П.1.2.

```
SubCadr F(In: D; Out: D0,D1,D2,D3,D4,D5,D6,D7);
Var D: array bit8 [M: Stream] Com;
Var D0,D1,D2,D3,D4,D5,D6,D7: array bit8 [M: Stream] Com;
Var i, j, k: Number;

for i:=0 to M1/2 do
    begin
    for j:=i to M1-1 do
        begin
        connect(i,j);
        D1[j+M1*(M1-i-1)] := D[i+M1*j];/// 90 градусов
        D1[i+M1*j] := D[M1-j-1+M1*i];
        D1[M1-j-1+M1*i] := D[M1-i-1+M1*(M1-j-1)];
        D1[M1-i-1+M1*(M1-j-1)] := D[j+M1(M1-i-1)]

        D3[i+M1*j] := D[j+M1*(M1-i-1)]; // 270 градусов
        D3[j+M1*(M1-i-1)] := D[M1-i-1+M1*(M1-j-1)];
        D3[M1-i-1+M1*(M1-j-1)] := D[M1-j-1+M1*i];
        D3[M1-j-1+M1*i] := D[i+M1*j];
        end:
    end;
for i:=0 to M1-1 do
    begin
    for j:=0 to M1-1 do
        begin
        connect(i,j);
        D4[i+M1*j] := D[i+M1*(M1-j)]; //отзеркаливание
        end;
end;
for k:=0 to M-1 do
    begin
        D0[k] := D[k]; //нулевой поворот
        D2[k] := D[M-k]; //поворот на 180 градусов
    end;

EndSubCadr;
```



а) б)

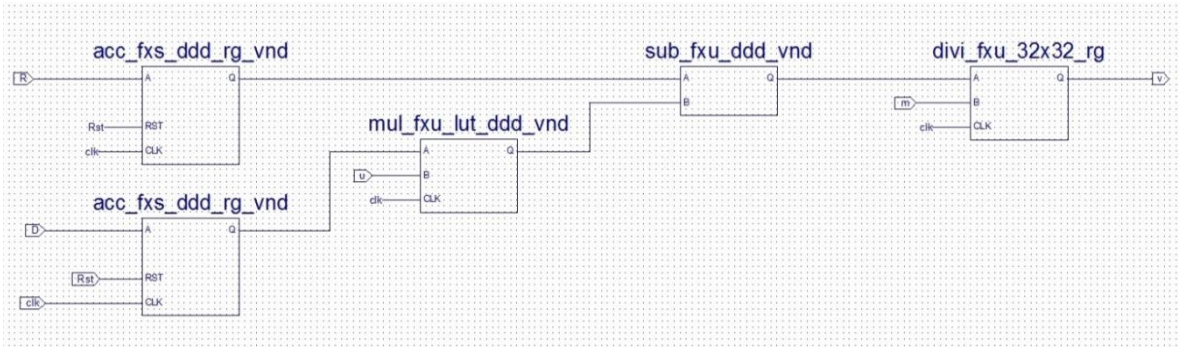
Рис. П.1.2. – Текст (а) подкадра F и его вычислительная структура (б)

При повороте доменных блоков выполняется перекоммутация потоков данных и соответственно изменение порядка следования элементов массива.

На рисунке П.1.3 приведен текст подкадра L и вычислительная структура, синтезированная на его основе.

```
SubCadr L(In: R, D, u, m; Out: v);
Var R: array bit8 [M: Stream] Com;
Var D: array bit8 [M: Stream] Com;
Var u: bit8 Com;
Var m: bit8 Com;
Var v: bit8 Com;
Var SumR, SumD: bit8 Com;
Var ComM: bit8 Com;
Var i_l: Number;
for i_l:=0 to M-1 do
    begin
        SumD := ACC(D[i_l]);
        SumR := ACC(R[i_l]);
    end;
ComM := SumD*u;
v := (ComM + SumR)/m;
EndSubCadr;
```

а)



б)

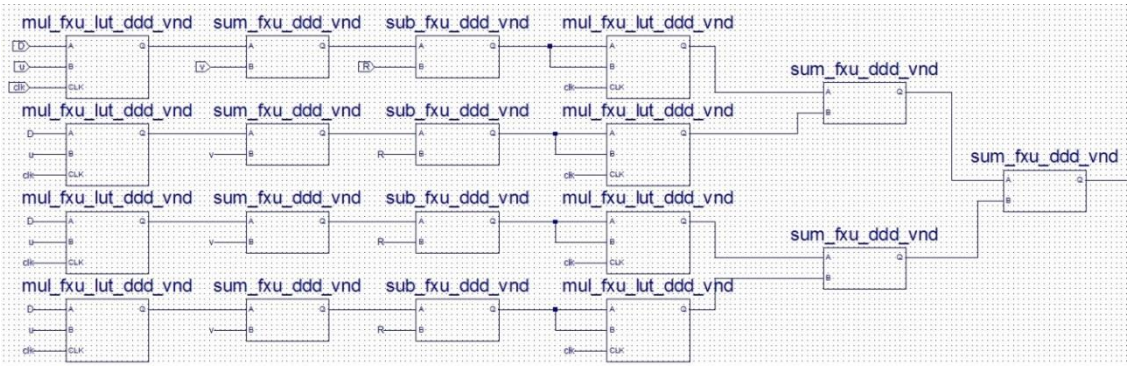
Рис. П.1.3. – Текст (а) подкадра L и его вычислительная структура (б)

В подкадре L осуществляется вычисление сдвига по яркости для одной из ориентаций доменного блока в паре с одним из ранговых блоков по формуле (2.2), приведенной во второй главе диссертации.

Подкадр E осуществляет вычисление расстояния между элементами трансформированного доменного блока и рангового по формуле (2.3). На рисунке П.1.4 приведены текст подкадра и его вычислительная структура.

```
SubCadr E(In: R, D, u, v; Out: e_);
Var R: array bit8 [M: Stream] Com;
Var D: array bit8 [M: Stream] Com;
Var u: bit8 Com;
Var v: bit8 Com;
Var e_: bit8 Com;
Var ComSqr, ComMul, ComSum: bit8 Com;
Var i_e: Number;
    for i_e:=0 to M-1 do
    begin
        ComMul := u*D[i_e];
        ComSum := ComMul + v - R[i_e];
        ComSqr := SQR(ComSum);
        e_ := ACC(ComSqr);
    end;
EndSubCadr;
```

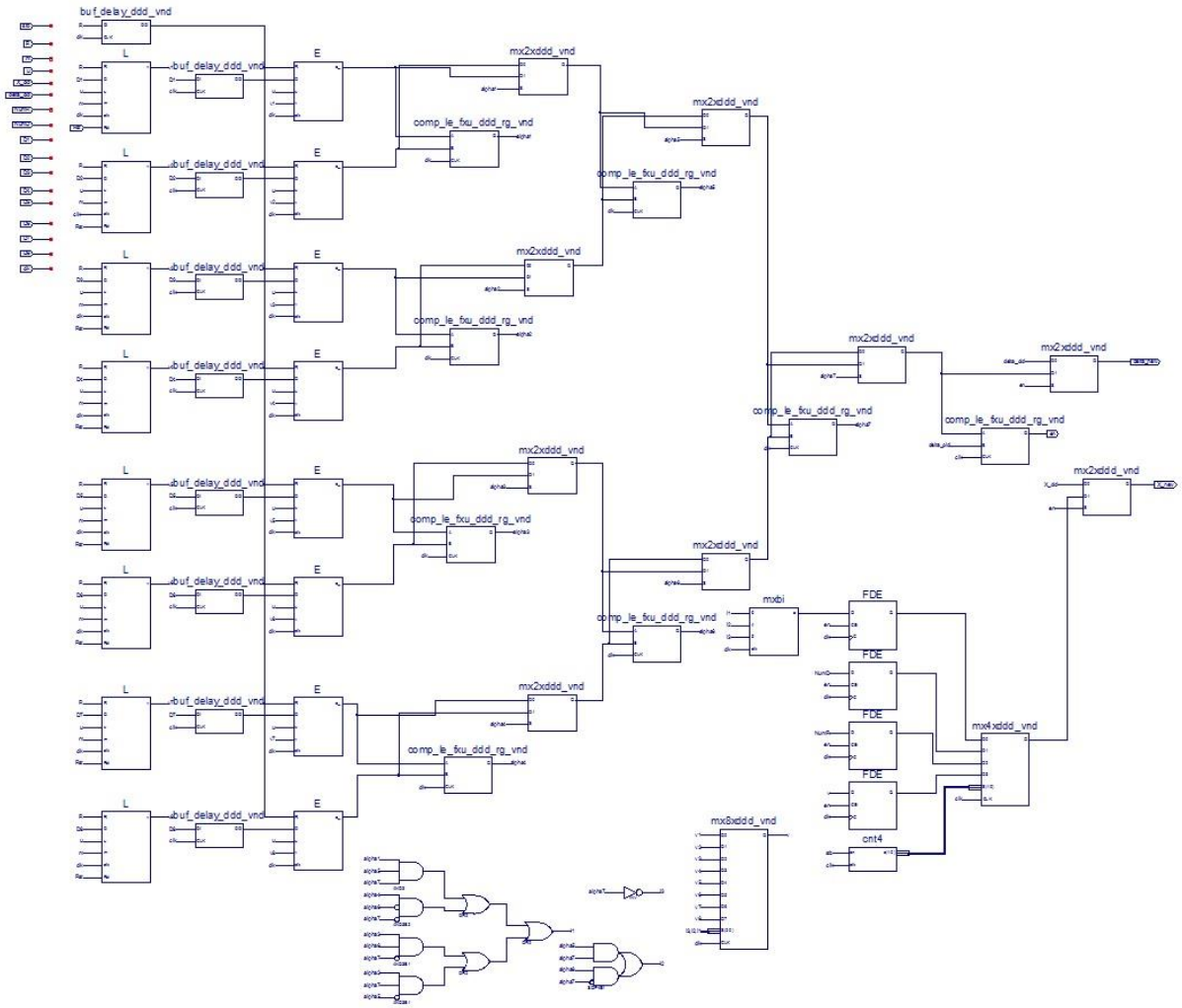
а)



б)

Рис. П.1.4 – Текст (а) подкадра E и его структура (б)

Подкадр LE8 объединяет работу подкадров L и E для восьми ориентаций одного доменного блока. Синетезированная вычислительная структура для этого подкадра показана на рисунке П.1.5 а). На рисунке П.1.5 б) приведен фрагмент подкадра LE8, в котором выполняется выбор наилучшего значения коэффициентов СИФ для дальнейшего их сохранения. На входы компаратора alpha8 приходят предыдущее наилучшее значение расстояния между доменным и ранговым блоками и наилучшее значение, полученное при работе с восемью ориентациями доменного блока, находящегося в работе. Если условие сравнения выполняется, то значения коэффициентов СИФ обвновляются.



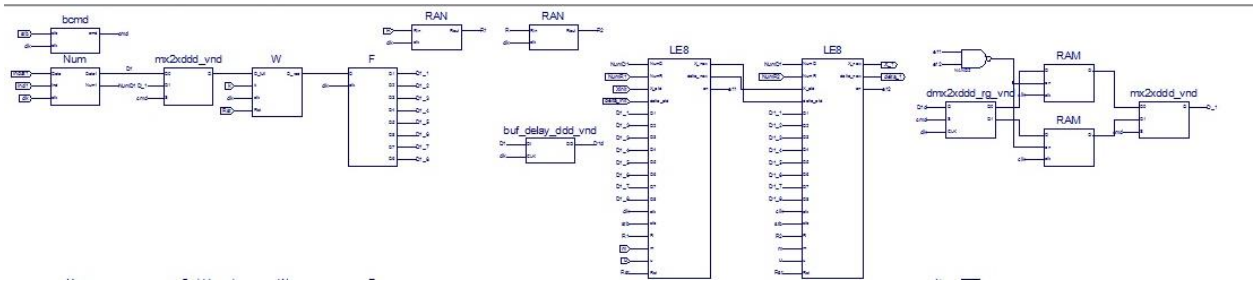
а)

```
alpha8 := buf7 <= delta_old;
if alpha8 then
   begin
      delta_new := buf7;
      X_new[0] := V_reg;
      X_new[1] := Nd_reg;
      X_new[2] := Nr_reg;
      X_new[3] := Num_F;
   end;
else
   begin
      delta_new := delta_old;
      X_new[*] := X_old[*];
   end;
```

б)

Рис. П.1.5 – Вычислительная структура (а) и фрагмент кода (б) подкадра LE8

Подкадр MainSIF объединяет вычислительные блоки для одного конвейера. В него входят подкадры W, F и LE8 (рис. П.1.6).



а)

```
SubCadr MainSIF(In: R_n1, R_n2, D_n, u, m, k_; Out: X, delta);
Var R_n1, R_n2: array bit8 [M_1: Stream] Com;
Var D_n: array bit8 [N_1: Stream] Com;
Var NumR1, NumR2, NumD: uinteger Com;
Var R1, R2: array bit8 [M: Stream] Com;
Var D_full: array bit8 [N: Stream] Com;
Var D_red: array bit8 [M: Stream] Com;
Var D0,D1,D2,D3,D4,D5,D6,D7: array bit8 [M: Stream] Com;
Var u: bit8 Com;
Var m: bit8 Com;
Var k_: bit8 Com;
Var delta, delta1,delta2,delta3: bits3 Com;
Var X, X1, X2, X3 : array [4: Vector] Com;
Var i,j: Number;

for i:=0 M_1-1 then
    Num(R_n1[i], i, NumR1, R1[i-1]);
for i:=0 M_1-1 then
    Num(R_n2[i], i, NumR2, R2[i-1]);
for j:=0 N_1-1 then
    Num(D_n[j], j, NumD, D_full[j-1]);

W(D_full, k_, D_red);
F(D_red, D0, D1, D2, D3, D4, D5, D6, D7);
LE8(NumR1, NumD, R1, u, m, D0,D1, D2, D3, D4, D5, D6, D7, delta1, delta2, X1, X2);
LE8(NumR2, NumD, R2, u, m, D0,D1, D2, D3, D4, D5, D6, D7, delta2, delta3, X2, X3);

X[*] := X3[*];
delta := delta3;

EndSubCadr;
```

б)

Рис. П.1.6 – Синтезированная вычислительная структура (а) и текст (б) подкадра MainSIF

После выполнения подкадра MainSIF полученные результаты выводятся во внешнюю память, а соответствующие ранговые блоки исключаются из вычислений. Те ранговые блоки, для который не были найдены коэффициенты СИФ сравниваются со следующей партией доменных блоков.

Для вычислений в качестве источника и приемника данных используются сдвоенные КРП, которые могут менять свою функцию, т.е. как вычитывать данные в схему, так и записывать результат работы в во внешнюю память. Для их описания в языке COLAMO существует конструкция Let, которая может менять источники и приемники данных при смене кадра без изменения ее кода. Пример текста такой программной конструкции, выполняющей фрактальное сжатие изображения, приведен на рисунке П.1.7

```
Let FullSIF(In: R1, R2,D1, D2, D3, u, m, k_; Out: X);
Var R1, R2: array bit8 [M_1: Stream] Com;
Var D1, D2, D3: array bit8 [N_1: Stream] Com;
Var u: bit8 Com;
Var m: bit8 Com;
Var k_: bit8 Com;
Var X:  array [4: Vector, NM: Stream] Com;
Var delta1,delta2,delta3: bits3 Com;
MainSIF(R1, R2, D1, u, m, k_; Out: X[*,1], delta1);
MainSIF(R1, R2, D2, u, m, k_; Out: X[*,2], delta2);
MainSIF(R1, R2, D3, u, m, k_; Out: X[*,3], delta3);
EndLet;
```

Рис. П. 1.7 – Пример программной конструкции Let

Программа декомпрессии изображения также реализована на языке COLAMO. На вход поступают полученные ранее коэффициенты СИФ. При этом блоки сжатия W и поворота F почти идентичны аналогичным для программы, реализующей фрактальное сжатие, за исключением того, что на выход подкадра F поступает только одна ориентация доменного блока согласно сохраненному параметру ротации во входных коэффициентах СИФ. Фрагмент подкадра с выбором выходной ориентации доменного блока приведен на рисунке П.1.8.

```
switch of N_f
    Case 0 :    for k:=0 to M do
                    begin
                        D_out[k] := D0[k];
                    end;
    Case 1 :    for k:=0 to M do
                    begin
                        D_out[k] := D1[k];
                    end;
    Case 2 :    for k:=0 to M do
                    begin
                        D_out[k] := D2[k];
                    end;
    Case 3 :    for k:=0 to M do
                    begin
                        D_out[k] := D3[k];
                    end;
    Case 4 :    for k:=0 to M do
                    begin
                        D_out[k] := D4[k];
                    end;
    Case 5 :    for k:=0 to M do
                    begin
                        D_out[k] := D5[k];
                    end;
    Case 6 :    for k:=0 to M do
                    begin
                        D_out[k] := D6[k];
                    end;
    Case 7 :    for k:=0 to M do
                    begin
                        D_out[k] := D7[k];
                    end;
    Default :   for k:=0 to M do
                    begin
                        D_out[k] := D8[k];
                    end;
```

Рис. П.1.8 – Фрагмент подкадра F

Подкадр T реализует применение сдвига по яркости для модифицируемого доменного блока (рис. П.1.9)

```
SubCadr T(In: D, u, v; Out: R);
Var D: array bit8 [M: Stream] Com;
Var R: array bit8 [M: Stream] Com;
Var u: bit8 Com;
Var m: bit8 Com;
Var v: bit8 Com;
Var MulD: bit8 Com;
Var ComS: bit8 Com;
Var i_l: Number;
for i_l:=0 to M-1 do
    begin
        MulD := D[i_l]*u;
        S := v+MulD;
        R[[i_l]] := S;
    end;
EndSubCadr;
```

Рис. П.1.9 – Подкадр T

ПРИЛОЖЕНИЕ 2

АКТЫ О ВНЕДРЕНИИ И ИСПОЛЬЗОВАНИИ РЕЗУЛЬТАТОВ ДИССЕРТАЦИИ

ЧЕКИНА МАРИЯ ДМИТРИЕВНА

МЕТОДЫ И СРЕДСТВА ОБРАБОТКИ ФРАКТАЛОВ НА РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

2.3.5 - Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:

доктор технических наук, профессор **Левин И.И.**

Таганрог – 2023

УТВЕРЖДАЮ
Директор
Института компьютерных технологий
и информационной безопасности
Южного федерального университета

_____ Г.Е. Веселов

«20» 04 2023 г.

АКТ

об использовании результатов диссертации на соискание ученой степени
кандидата технических наук
Чекиной Марии Дмитриевны
в учебном процессе кафедры интеллектуальных и многопроцессорных систем
Института компьютерных технологий и информационной безопасности
Южного федерального университета

Комиссия в составе:

председателя Никитиной А.В.

и членов комиссии Горбунова А.В., Куповых Г.В.

составили настоящий акт о том, что результаты диссертационной работы

«Методы и средства обработки фракталов на реконфигурируемых вычислительных системах»,

представленной на соискание ученой степени кандидата технических наук, использованы в учебном процессе кафедры интеллектуальных и многопроцессорных систем (ИМС) Института компьютерных технологий и информационной безопасности (ИКТИБ) Южного федерального университета (ЮФУ).

Настоящий акт подтверждает, что в учебном процессе кафедры ИМС ИКТИБ ЮФУ используются следующие научно-теоретические и практические результаты кандидатской диссертации Чекиной М.Д.:

- метод фрактального сжатия изображений, реализованный для РВС параллельно-конвейерным способом, с возможностью побитовой обработки данных, и сортировкой структур, содержащих выходные данные, по номеру рангового блока;
- метод декомпрессии сжатых изображений, реализованный для РВС параллельно-конвейерным способом, с использованием только одной косвенной адресации для блоков памяти, содержащих доменные блоки;

– метод Якоби, модифицированный для решения СЛАУ, образованной при дискретизации задачи супердиффузии, реализованный для РВС, с оптимизацей вычислительной структуры для конкретной СЛАУ на основе оценки ее параметров.

Указанные материалы используются в лекционных курсах по дисциплинам «Современные проблемы прикладной математики и информатики» (тема № 11 «Фракталы. Дробные производные») и «Математическое моделирование процессов гидрофизики на высокопроизводительных вычислительных системах» (модуль 2 «Численная реализация математических моделей гидрофизики») для подготовки магистров направления подготовки 01.04.02 Прикладная математика и информатика (образовательные программы «Прикладная математика для высокопроизводительных вычислительных систем», «Математическое моделирование в инженерных науках»).

Председатель комиссии:

Лектор дисциплин
«Современные проблемы прикладной
математики и информатики» и
«Математическое моделирование
процессов гидрофизики на
высокопроизводительных вычислительных системах»,
профессор ИКТИБ ИТА ЮФУ,
д.т.н., доцент _____ А.В. Никитина

Члены комиссии:

Зам. директора ИКТИБ ИТА ЮФУ
по учебной работе, к.т.н., доцент _____ А.В. Горбунов

Руководитель образовательной программы,
«Математическое моделирование в инженерных науках»
заведующий кафедрой
высшей математики ИКТИБ ИТА ЮФУ,
д.ф.-м.н., профессор _____ Г.В. Куповых

УТВЕРЖДАЮ
Технический директор
ООО «НИЦ СЭ и НК»,
к.т.н.

_____ Ю.И. Доронченко

"21" апреля 2023 г.

АКТ

о внедрении результатов кандидатской диссертации «Методы и средства обработки фракталов на реконфигурируемых вычислительных системах»
младшего научного сотрудника Общества с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров» (ООО «НИЦ СЭ и НК»)
Чекиной Марии Дмитриевны

Комиссия в составе председателя, заместителя директора по разработкам, к.т.н. Катаева О.В., и членов комиссии: начальника отдела, к.т.н. Сорокина Д.А., начальника отдела, к.т.н. Раскладкина М.К., составила настоящий акт о том, что в разработках Общества с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров» (ООО «НИЦ СЭ и НК») внедрены следующие результаты диссертации Чекиной Марии Дмитриевны на тему «Методы и средства обработки фракталов на реконфигурируемых вычислительных системах»:

1. Метод создания для РВС параллельно-конвейерных программ фрактального сжатия изображений с использованием распараллеливания по слоям и итерациям, с побитовой обработкой данных, которое обеспечивает максимальное использование вычислительного ресурса системы.

2. Метод создания для РВС параллельно-конвейерных программ декомпрессии изображений, сжатых фрактальным способом, с использованием единственной косвенной адресации памяти, содержащей доменные блоки.

3. Метод Якоби, модифицированный для решения СЛАУ, образованной при дискретизации задачи супердиффузии, реализованный для РВС параллельно-конвейерным способом, с оптимизацией вычислительной структуры для конкретной СЛАУ на основе оценки ее параметров.

Указанные результаты диссертации Чекиной М.Д. были внедрены при реализации параллельно-конвейерных программ, выполняющих обработку данных.

Основываясь на результатах диссертации совместно с Чекиной М.Д. созданы следующие программы:

- программа фрактального сжатия изображений (свидетельство о государственной регистрации программ для ЭВМ № 2023613157, РФ). Левин И.И., Чекина М.Д. Зарегистр. в Реестре программ для ЭВМ 01.03.2023 г.

Правообладатель: Общество с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров» (ООО «НИЦ СЭ и НК»);

- программа декомпрессии изображений, сжатых фрактальным способом (свидетельство о государственной регистрации программ для ЭВМ № 2023617300, РФ). Левин И.И., Чекина М.Д. Зарегистр. в Реестре программ для ЭВМ 07.04.2023 г. Правообладатель: Общество с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров» (ООО «НИЦ СЭ и НК»).

- программа для решения задачи супердиффузии (свидетельство о государственной регистрации программ для ЭВМ № 2023617550, РФ). Левин И.И., Чекина М.Д. Зарегистр. в Реестре программ для ЭВМ 11.04.2023 г. Правообладатель: Общество с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров» (ООО «НИЦ СЭ и НК»).

Результаты диссертации Чекиной М.Д. внедрены в следующих НИР и ОКР, выполненных в НИЦ СЭ и НК:

1) НИОКР «Разработка нового поколения программного обеспечения РВС для решения ресурсоёмких научно-технических задач различных предметных областей», шифр «Поколение»;

2) НИР «Выбор направления исследований и разработки высокопроизводительных вычислительных систем «СКИФ-ГЕО» второй базовой конфигурации, обеспечивающих формирование оптимальных конфигураций аппаратных средств и программного обеспечения для решения расчетных геолого-геофизических задач», шифр «Нефть».

Внедрение разработанных соискателем методов и программ позволило повысить реальную производительность РВС при обработке самоподобных структур, а именно выполнении фрактального сжатия изображения – более чем в 10 раз, декомпрессии изображения – в 2,5 раза, и при решении задачи супердиффузии на 40%.

ПРЕДСЕДАТЕЛЬ КОМИССИИ

Заместитель директора
по разработкам ООО «НИЦ СЭ и НК»,
к.т.н., О.В. Катаев

ЧЛЕНЫ КОМИССИИ

Начальник отдела ООО «НИЦ СЭ и НК»,
к.т.н. М.К. Раскладкин

Начальник отдела ООО «НИЦ СЭ и НК»,
к.т.н. Д.А. Сорокин

ПРИЛОЖЕНИЕ 3

СВИДЕТЕЛЬСТВА О ГОСУДАРСТВЕННОЙ РЕГИСТРАЦИИ ПРОГРАММ ДЛЯ ЭВМ

ЧЕКИНА МАРИЯ ДМИТРИЕВНА

МЕТОДЫ И СРЕДСТВА ОБРАБОТКИ ФРАКТАЛОВ НА РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

2.3.5 - Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:

доктор технических наук, профессор **Левин И.И.**

Таганрог – 2023

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023614463

Программа фрактального сжатия изображений

Правообладатель: ***Общество с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров» (ООО «НИЦ СЭ и НК») (RU)***

Авторы: ***Левин Илья Израилевич (RU), Чекина Мария Дмитриевна (RU)***

Заявка № **2023613157**
Дата поступления **21 февраля 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***01 марта 2023 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023617300

Программа декомпрессии изображений, сжатых фрактальным способом

Правообладатель: ***Общество с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров» (ООО «НИЦ СЭ и НК») (RU)***

Авторы: ***Левин Илья Израилевич (RU), Чекина Мария Дмитриевна (RU)***

Заявка № **2023615238**
Дата поступления **22 марта 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***07 апреля 2023 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023617550

Программа для решения задачи супердиффузии

Правообладатель: ***Общество с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров» (ООО «НИЦ СЭ и НК») (RU)***

Авторы: ***Левин Илья Израилевич (RU), Чекина Мария Дмитриевна (RU)***

Заявка № **2023615305**
Дата поступления **22 марта 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***11 апреля 2023 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Ю.С. Зубов