На правах рукописи

Попов Алексей Юрьевич

Принципы организации и методология применения вычислительных систем с набором команд дискретной математики

2.3.2. Вычислительные системы и их элементы

Автореферат
диссертации на соискание учёной степени
доктора технических наук

Москва — 2023

Работа выполнена в Федеральном государственном бюджетном образовательном учреждении высшего образования «Московский государственный технический университет имени Н.Э.Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э.Баумана).

Официальные оппоненты: **Фельдман Владимир Марткович,**
д.т.н., профессор, Публичное акционерное общество «Институт электронных управляющих машин имени И.С. Брука», Заместитель генерального директора

Бобков Сергей Геннадьевич,
д.т.н., старший научный сотрудник, Федеральное государственное бюджетное учреждение науки Институт проблем проектирования в микроэлектронике Российской академии наук, Заместитель директора

Логовский Алексей Станиславович,
д.т.н., Акционерное общество «Радиотехнический институт имени академика А.Л. Минца», Заместитель генерального конструктора

Ведущая организация: Общество с ограниченной ответственностью «РТСофт-Смарт Грид»

Защита состоится 14 марта 2024 года в 14:00 часов на заседании диссертационного совета 24.2.331.19 при МГТУ им. Н.Э.Баумана по адресу: 105005, г. Москва, 2-ая Бауманская ул., д.5, стр.1, зал Ученого совета ГУК МГТУ им. Н.Э.Баумана.

С диссертацией можно ознакомиться в библиотеке МГТУ им. Н.Э.Баумана и на сайте https://bmstu.ru.

Отзывы на автореферат в двух экземплярах, заверенные печатью учреждения, просьба направлять по адресу: 105005, г. Москва, 2-ая Бауманская ул., д.5, стр.1, МГТУ им. Н.Э.Баумана, кафедра ИУ3, Ученому секретарю диссертационного совета 24.2.331.19 Сакулину С.А.

Автореферат разослан «____»_____2024 года.

Ученый секретарь
диссертационного совета
к.т.н., доцент С.А. Сакулин

Общая характеристика работы

Актуальность темы исследования. Современные вычислительные машины обладают не только достоинствами, вытекающими из принципов унификации и универсальности, но и рядом недостатков. Основными из них можно считать: малую производительность и степень параллельности в сравнении с теоретически достижимой, неудовлетворительные масс-габаритные характеристики, сложность проектирования высокопроизводительных и высоко-нагруженных программных решений, высокое энергопотребление, наличие так называемого семантического разрыва для различных уровней организации вычислительного процесса.

Одним из перспективных способов повышения производительности ЭВМ является применение аппаратных ускорителей вычислений (акселераторов). Целью применения акселераторов является увеличение скорости обработки данных благодаря применению более производительных и параллельных аппаратных механизмов. Одновременная обработка данных позволяет снять часть вычислительной нагрузки микропроцессоров и перенести их на программное и аппаратное обеспечение ускорителей. В данной работе предпринята попытка создания принципов и их воплощения в виде новой вычислительной технологии (т.е. совокупности программно-аппаратного и организационно-методического видов обеспечения), которые бы позволили ускорить обработку больших объемов информации за счет повышения эффективности операций над множествами и структурами данных. Разработанная в ходе диссертационного исследования система предназначена для более эффективного решения оптимизационных задач дискретной математики, широко распространенных во многих важных областях науки и техники. Было разработано и реализовано принципиально новое вычислительное устройство: *Процессор обработки структур*, реализующее набор команд высокого уровня над множествами и структурами данных, а также вычислительная система на основе данного устройства.

Актуальность темы исследования обусловлена востребованностью в науке и технике действующих образцов вычислительной техники, обладающих более высокой производительностью, меньшим энергопотреблением, меньшими масс-габаритными характеристиками, меньшей стоимостью. Также актуальность диссертационного исследования подтверждается важностью и востребованностью создания эффективных средств вычислительной техники, основанных на объектах отечественной интеллектуальной собственности.

Степень разработанности темы исследования. Перспективным направлением развития вычислительной технологии является создание гетерогенных систем, однако набор применяемых в таких системах ускорителей вычислений существенно ограничен. До данной работы не существовало средств, на аппаратном уровне выполняющих набор операций дискретной математики, достаточный для реализации большинства алгоритмов оптимизации, а современные универсальные вычислительные системы выполняют операции обработки множеств лишь посредством последовательности повторяющихся арифметически-логических

операций. В применяемых в настоящий момент универсальных принципах организации вычислений, как сами алгоритмы, неструктурированные данные, так и структуры данных, и алгоритмы их обработки хранятся и обрабатываются одними и теми же универсальными микропроцессорами, не удовлетворяющими растущим техническим и функциональным требованиям.

Цель работы:

Разработка и практическая апробация теоретических принципов функционирования аппаратного, лингвистического, программного и организационно-методического обеспечения вычислительных систем, выполняющих хранение и обработку множеств, структур данных и графов на основе набора команд дискретной математики.

Задачи исследования:

- Исследование эффективности существующих средств вычислительной техники, подходов к ускорению вычислений и обработке больших объемов данных.
- Анализ современной элементной базы и выбор вариантов реализации устройств ускорения вычислений.
- Разработка моделей и принципов функционирования вычислительных систем с аппаратной поддержкой операций дискретной математики.
- Разработка принципов функционирования аппаратного устройства обработки множеств и структур данных на основе набора команд дискретной математики: *Процессора обработки структур*.
- Определение архитектурных параметров системы с набором команд дискретной математики для решения задач различной размерности.
- Разработка моделей устройств и их верификация, а также разработка опытного образца вычислительной системы на основе *Процессора обработки структур*.
- Разработка лингвистического обеспечения вычислительной системы для решения задач дискретной оптимизации.
- Создание методики и средств проектирования программ, функционирующих в вычислительной системе с набором команд дискретной математики.
- Исследование эффективности и области применения вычислительной системы с набором команд дискретной математики.

Методы исследований. При решении задач данной диссертационной работы были использованы методы теории графов, методы решения задач комбинаторной оптимизации (поиска в глубину, поиск в ширину), методы комбинаторного анализа, методы алгебры логики, методы теории формальных языков, методы синтеза и верификации комбинационных схем и цифровых автоматов.

Научная новизна работы:

- На основе результатов анализа архитектуры современных вычислительных машин предложены варианты ускорения вычислений при обработке множеств и структур данных.

– Получена уточненная формальная модель вычислителя, выполняющего обработку структур данных, что позволило сформулировать принципы раздельной обработки данных и их отношений в алгоритмах дискретной оптимизации.
– Предложена функциональная модель устройства обработки отношений данных, позволившая разработать и реализовать принципы функционирования вычислительной системы с аппаратной поддержкой операций над структурами данных.
– Разработаны принципы функционирования вычислительной системы с аппаратной поддержкой операций над структурами данных, обеспечивающие возможность их параллельной обработки.
– Разработана архитектура специализированного *Процессора обработки структур*, включающего ряд уникальных аппаратных устройств.
– Разработан набор команд *Процессора обработки структур*, выполняющего аппаратную обработку множеств и структур данных. Новизна устройства подтверждена патентом РФ на полезную модель.
– Разработаны параметризованные модели устройств: *Каталога*, *Устройства загрузки команд*, *Операционного буфера*, устройства микропрограммного управления, что позволило реализовать *Процессор обработки структур* и вычислительную систему с набором команд дискретной математики.
– Реализован прототип вычислительной системы с набором команд дискретной математики, язык ассемблера и средства компиляции программ, что позволило разработать лингвистическое обеспечение системы.
– Разработаны алгоритмы оптимизации для системы с набором команд дискретной математики, для которых проведено тестирование и сравнение полученного решения с существующими универсальными вычислительными системами.
– Определены общие принципы применения систем с набором команд дискретной математики для реализации СУБД и систем обработки информации, геоинформационных и робототехнических систем.
– Создана методика преобразования алгоритмов дискретной оптимизации, составляющая организационно-методическое обеспечение системы.
– Выполнено проектирование алгоритмов комбинаторной оптимизации для системы с набором команд дискретной математики, экспериментально исследована архитектурная эффективность исполнения команд в *Процессоре обработки структур*, что позволило провести тестирование и сравнение полученного решения с существующими универсальными вычислительными системами.

Положения, выносимые на защиту:

– Уточненная формальная модель вычислителя, выполняющего обработку структур данных.

– Функциональная модель устройства обработки отношений данных, определяющая выполняемые им операции.
– Принципы функционирования вычислительной системы с аппаратной поддержкой операций над множествами и структурами данных.
– Принципы функционирования и архитектура устройства обработки множеств и структур данных, позволяющие реализовать *Процессор обработки структур* с набором команд дискретной математики.
– Принципы аппаратной обработки структуры B+дерева в составе *Процессора обработки структур*, позволившие впервые реализовать устройства: *Каталог*, *Операционный буфер* и *Устройство загрузки команд*.
– Методика преобразования алгоритмов дискретной оптимизации, позволяющая применять их в вычислительной системе с набором команд дискретной математики.
– Методика преобразования декомпозиции информационного графа программы для вычислительной системы с аппаратной поддержкой операций над множествами и структурами данных.
– Алгоритмы дискретной оптимизации на сетях и графах, оптимизированные для системы с набором команд дискретной математики: алгоритмы Крускала, Прима, Форда-Фалкерсона, Дейкстры, алгоритмы обхода графа в ширину и глубину.

Практическая значимость работы. Наибольшую практическую значимость имеют следующие результаты работы:

– Действующий образец устройства *Процессора обработки структур*, в отличие от существующих микропроцессоров, выполняющий обработку множеств и структур данных аппаратно и параллельно, занимающий в 30 раз меньше аппаратных ресурсов.
– Действующий образец вычислительной системы с набором команд дискретной математики, методика модификации алгоритмов и методика преобразования декомпозиции информационного графа программы, позволяющие разработать и реализовать параллельные варианты алгоритмов дискретной оптимизации.
– Транслятор ассемблера *Процессора обработки структур*.
– Действующие программы реализации алгоритмов Форда-Фалкерсона, Дейкстры, Крускала, Прима, и других алгоритмов для системы с набором команд дискретной математики.

Апробация работы. Основные положения диссертационной работы обсуждены на второй международной научно-технической конференции, посвященной 95-летию со дня рождения академика В.Н.Челомея (г. Москва 2009), международной конференции ICACON 2015 (Будапешт, 2015), международной научной конференции Hawaii International Conference on System Science «HICSS50» (Гавайи, 2017), международной выставке «Армия 2017» (Москва, 2017), международной выставке «ChipExpo 2017» (Москва, 2017), международной конфренции IEEE MMET 2018 (Санкт-Петербург, 2018), международной конфренции IEEE

EIConRus 2019 (Москва, 2019), международной конференции «Математические методы в технике и технологиях 2019 (Санкт-Петербург, 2019), международной конференции IEEE EIConRus 2020 (Москва, 2020), международной конфренции IEEE EIConRus 2021 (Москва, 2021), международной выставке «ЦИПР 2022» (Нижний Новгород, 2022). международной выставке «Армия 2022» (Москва, 2022), XI форуме по цифровизации оборонно-промышленного комплекса – «ИТОПК-2022» (Пермь, 2022), международной конференции «Суперкомпьютерные дни в России», RuSCDays2022 (Москва, 2022), международном форуме «Микроэлектроника 2023» (Сочи, 2023).

Личный вклад соискателя. В диссертационной работе приведены научные положения и практические результаты, полученные лично автором. Из совместных публикаций в диссертацию включен лишь тот материал, который непосредственно принадлежит соискателю, заимствованный материал обозначен в работе ссылками.

Достоверность результатов. Достоверность полученных в диссертационной работе результатов подтверждается проведенными натурными испытаниями действующего образца системы с набором команд дискретной математики. Функциональность, работоспособность и технические характеристики всей системы и *Процессора обработки структур* продемонстрированы в ходе проведения выставок, приведены автором в периодических изданиях и открыто демонстрируются на действующих образцах в Центре обработки данных МГТУ им. Н.Э.Баумана.

Реализация и внедрения. Теоретические и практические результаты работы, полученные автором, нашли применение при исследовании факторов активного долголетия в ФГБУ ФНКЦ ФХМ ФМБА России, при проведении научно-исследовательской деятельности АО «БАЙКАЛ ЭЛЕКТРОНИКС», в научно-исследовательской и коммерческой деятельности компании ООО «ИБС Софт», при реализации программно-технического комплекса по предотвращению угроз, реализуемом УИ-ВЦ МГТУ им. Н.Э. Баумана, при создании высокопроизводительного вычислительного комплекса Тераграф в ходе совместной исследовательской и коммерческой деятельности с АО «ТРИНИТИ СОЛЮШНС», в научно-исследовательской деятельности и учебном процессе научно-учебного комплекса «Информатика и системы управления» МГТУ им. Н.Э.Баумана.

Публикации автора по теме диссертации. По результатам диссертационной работы автором опубликовано 19 работ, в том числе 15 работ в журналах, входящих в список научных журналов ВАК Минобрнауки России и патент Российской Федерации. Общий объем: 11.0 п.л.

Объем и структура диссертации. Диссертационная работа состоит из введения, пяти глав, заключения, списка литературы и приложения, занимающих 400 страниц текста, в том числе 81 рисунок и 56 таблиц, список литературы из 164 наименований на 16 страницах.

Содержание работы

Во введении обоснована актуальность темы диссертации, сформулированы основные задачи и научные результаты, дано краткое содержание глав.

В первой главе «Определение предметной области и постановка задачи» описана проблемная область и определены объекты исследования. Проведенный в разделе 1.1 анализ развития систем обработки больших объемов данных позволил сделать вывод о том, что проблема ускорения вычислительного процесса в настоящее время не менее актуальна, чем десять или двадцать лет назад. Показано, что ряд подходов к ускорению вычислений, таких как использование графических ускорителей, имеют ограниченную область применения, а их архитектура не адекватна многим важным моделям данных (например, графам) . В разделе 1.2 выполнен анализ путей развития вычислительной техники, продемонстрирована противоречивость требований к перспективным образцам вычислительной техники: обработка данных в потоке и выполнение сложных аналитических функций; универсальность и высокое быстродействие.

В разделе 1.3 рассматривается одно из важнейших понятий вычислительной техники: структуры данных. Информация представляется в оперативной памяти ЭВМ как в виде отдельных хранимых операндов (скалярных данных), так и в виде структур данных, для которых в коде программ (или в виде информации в оперативной памяти) задано множество отношений. Структура данных определяется, как совокупность двух множеств:

$$S = (I,R), \tag{1}$$

где I – множество элементов данных; R – множество отношений между элементами данных.

В зависимости от способа их представления в памяти, структуры данных делятся на векторные, элементы которых расположены в памяти последовательно (массивы, матрицы), и ссылочные (также используется термин *списковые*), для которых множество отношений задано при помощи более сложного соглашения об адресации (хэш-таблицы, кучи) или осуществляется при помощи указателей (списки, деревья). Разработаны и активно применяются несколько десятков структур, которые позволяют сократить вычислительную сложность алгоритмов. Подходы к оптимизации алгоритмов с использованием структур данных продемонстрированы в работах таких ученых, как Кормен, Лейзерсон, Ахо, Хопкрофт, Кнут и многих других. Примером таких подходов могут служить реализации большинства алгоритмов дискретной оптимизации, в том числе алгоритмов на сетях и графах. Так, для алгоритма Прима применение структур данных позволяет снизить вычислительную сложность с $O(V^2)$ до $O(E + V \log V)$, а вычислительную сложность алгоритма дихатомического разрезания гиперграфа удается снизить с $O(V^3)$ до $O(V \log V)$. Вместе с этим, архитектурные решения, заложенные в современные ЭВМ, направлены на ускорение обработки векторных структур и, напротив, замедляют обработку ссылочных структур данных.

В связи с конструктивной неоднородностью полупроводниковой памяти, обращение процессора к данным в оперативной памяти требует различного времени в зависимости от их расположения. Проведенные в диссертации исследования показали, что время доступа к данным, находящимся на большом адресном расстоянии, существенно выше, чем к данным, расположенным по последовательным адресам. Это негативно сказывается на времени обработки ссылочных структур, так как их элементы могут располагаться в памяти на большом адресном расстоянии. Доступ к оперативной памяти замедляется также вследствие пакетной передачи, и в результате работы механизмов виртуализации памяти.

Свой вклад в замедление вносит длинный конвейер универсальных процессоров, так как при обработке ссылочных структур используются указатели, которые поступают на конвейер в качестве операндов. Только после их обработки становится возможным передать физический адрес на шину памяти, в результате чего транзакция выполняется только после завершения предшествующих команд, а упреждающая загрузка (предвыборка) операндов становится невозможной. Таким образом, обработка зависимых данных происходит в тех случаях, когда результат работы одной команды используется в качестве адреса операнда другой. Эта ситуация называется *проблемой обработки зависимых данных* и приводит к существенному замедлению обработки ссылочных структур. В диссертации проведены экспериментальные исследования проблем обработки зависимостей, результаты которых показаны на Рисунке 1.

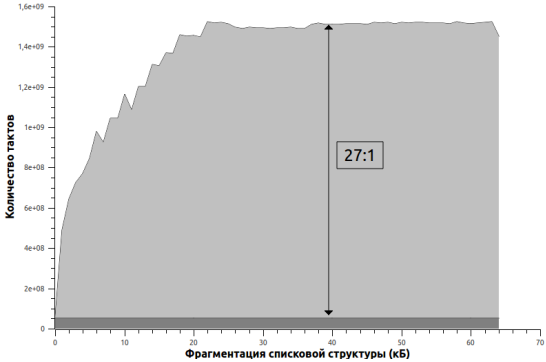

Рисунок 1 — Сравнение времени доступа к ссылочным и векторным структурам данных

При увеличении фрагментации структур данных существенно возрастает время доступа к ним. Обработка больших массивов информации сопряжена с открытием большого числа физических страниц памяти. При первом обращении к странице наблюдается увеличение времени доступа. Это связано с необходимостью преобразования логического адреса в физический адрес памяти, а также с открытием страницы динамической памяти и сохранением данных в кэш-памяти. Преобразование выполняется на основе информации об использованных ранее

страницах, содержащейся в TLB-буфере (Буфере быстрого страничного преобразования) процессора. Первое обращение к странице при отсутствии информации в TLB-буфер вызывает двойное обращение к оперативной памяти: сначала за информацией из таблицы страниц, а далее - за востребованными данными. Предвыборка заключается в заблаговременном проведении всех указанных действий благодаря дополнительному запросу небольшого количества данных из оперативной памяти. Приведенные эксперименты позволили выявить архитектурные принципы, существенно замедляющие обработку структур данных. Для дальнейшего совершенствования архитектуры вычислительных машин необходимо разработать и реализовать более совершенные механизмы для поддержки операций над структурами данных и множествами, устраняющие указанные недостатки при сохранении универсальности.

На основании модели вычислительной системы, представленной в работах В.Г. Хорошевского, определена уточненную модель вычислителя, выполняющего обработку структур данных:

$$c_S = \langle U_I, U_R, g, a_I(p_I(I)), a_R(p_R(R)) \rangle, \tag{2}$$

где U_I - множество устройств обработки скалярных данных, U_R - множество устройств обработки отношений данных ($U = U_I \cup U_R$), g - структура связей между устройствами, a_I и a_R алгоритмы управления вычислительными процессами в U_I и U_R ($a = a_I \cup a_R$), p_I программа обработки данных I, p_R программа обработки отношений R ($p = p_I \cup p_R$).

Уточненная модель вычислителя позволяет сформулировать задачи исследования диссертационной работы: выполнить структурную декомпозицию архитектуры вычислительной системы и алгоритмов обработки структур данных, создать специализированные аппаратные и программные средства независимой обработки отношений R и скалярных данных I ($U_I \cap U_R = \varnothing$; $a_I \cap a_R = \varnothing$, $p_I \cap p_R = \varnothing$), а также определить топологию и характеристики связности устройств системы g. Для достижения этих целей были сформулированы 14 принципов, на основе которых в главах 2-5 была решена задача разработки вычислительной системы с аппаратной поддержкой операций над структурами данных и множествами.

Принцип гетерогенности вычислений. *Для разработки вычислительной системы с аппаратной поддержкой операций дискретной математики необходимо реализовать аппаратные средства обработки множеств и структур данных, функционирующие независимо от аппаратных средств арифметической обработки. Это позволит построить архитектуру вычислительной системы таким образом, чтобы множества отношений и множества данных обрабатывались параллельно.*

На основе анализа результатов экспериментов делается вывод о том, что дальнейшее совершенствование архитектуры вычислительных машин в рамках существующих принципов ведет к противоречивым результатам: попытки ускорения работы одних алгоритмов могут приводить к замедлению других. Это

противоречие может быть частично устранено благодаря применению аппаратных средств, реализующих более совершенные механизмы для поддержки операций над структурами данных, которые работают параллельно и одновременно с существующей подсистемой памяти. Это не снизит универсальности системы, но позволит ускорить решение ряда задач, основанных на использовании ссылочных структур данных.

В разделе 1.4 рассматриваются перспективы применения аппаратных ускорителей вычислений (акселераторов). Целью применения ускорителей является увеличение скорости обработки данных благодаря применению более производительных и параллельных аппаратных механизмов. Очевидно, что параллельная обработка данных на различных устройствах позволяет снять часть вычислительной нагрузки с универсальных процессоров и перенести их на программное и аппаратное обеспечение ускорителей. В разделе 1.5 проводится обоснование фундаментальной темы исследования: поиск и реализация принципов хранения и эффективной обработки множеств и структур в электронных вычислительных машинах и системах.

В результате исследований, проведенных **во второй главе** «Анализ методов и средств аппаратной поддержки обработки множеств и структур данных», был определен базис операций, кванторов и отношений, применяемых в дискретной математике ко множествам.

Устройство обработки отношений данных может быть представлено формальной моделью, определяющей выполняемые им функции и операции:

$A = \langle A_1, \ldots, A_n \rangle$ - функция хранения кортежа A из n множеств;
$R(A_i,x,y), x \in A_i, y \in A_i$ - функция определения отношения между элементами x и y множества A_i;
$\mid A_i \mid, i = \overline{1,n}$ - операция определения мощности множества;
$x \in A_i, i = \overline{1,n}$ - операция проверки принадлежности элемента x множеству;
$A_i \cup x, i = \overline{1,n}$ - операция добавления элемента в множество;
$A_i \setminus x, i = \overline{1,n}$ - операция удаления элемента из множества;
$A \setminus A_i$ - операция удаления множества A_i из кортежа A;
$A_i \subset A_j$ - операция отношения включения множества A_i в A_j;
$A_i \equiv A_j$ - операция отношения эквивалентности;
$A_i \cup A_j$ - операция объединения множеств;
$A_i \cap A_j$ - операция пересечения множеств;
$A_i \setminus A_j$ - операция разности множеств;
$A_i \triangle A_j$ - операция симметрической разности;
$\overline{A}$ - операция дополнения A_i;
$A_i \times A_j$ - операция Декартового произведения;
2^{A_i} - операция определения Булеана.

Функции хранения и отношения элементов являются необходимым базисом, который должен быть заложен в устройство для реализации остальных операций.

Модель основана на представленных базовых принципах обработки дискретной информации. Из модели следует, что в устройстве требуется обеспечивать упорядоченное хранение нескольких множеств в соответствии с выбранным вариантом отношения порядка хранимых элементов.

Была определена вычислительная сложность реализации операций для различных типов структур данных, известных из практики программирования. На основе этого, модель B+деревьев и алгоритмы их обработки были выбраны в качестве прототипа при аппаратной реализации функций обработки множеств и структур данных.

Рассмотренные в разделе 2.3 алгоритмы выполнения операций в B+дереве позволили выявить возможность повторения однотипных действий над различными уровнями дерева. Это показывает возможность разработки универсальных обрабатывающих устройств (вершинных процессоров), которые будут заниматься поиском и модификацией структуры дерева на разных его уровнях.

В разделе 2.4 определены принципы двойственности структур данных и структурной декомпозиции вычислительной системы для решения задач дискретной оптимизации, предложена иерархическая структура подсистемы памяти, обеспечивающая выполнение основных функций для структур данных больших объемов. Предложены четыре уровня памяти: внутренняя память ключей для нижнего уровня дерева; внутренняя память вершин для хранения структуры дерева; внешняя память структур; внешнее хранилище структур (Рисунок 2).

Принцип двойственности структур данных *Любая структура данных обладает двойственностью, так как состоит из информационной и структурной составляющих. Для возможности параллельной обработки структур данных в состав вычислительной системы вносится аппаратное устройство (Процессор обработки структур), который обрабатывает лишь ту часть информации, которая определяет взаимные отношения хранимых данных, т.е. структурную часть структур данных. Универсальный Центральный процессор обрабатывает информационную составляющую структур данных.*

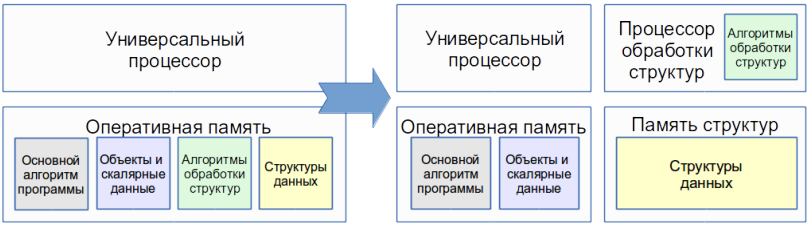

Рисунок 2 — Универсальная ЭВМ, построенная по базовым принципам Фон-Неймана (слева) и вычислительная система с аппаратной поддержкой операций над множествами и структурами данных (справа)

Принцип структурной декомпозиции вычислительной системы для решения задач дискретной оптимизации. *Процессор обработки структур имеет*

доступ к оперативной памяти, в которой хранятся структуры данных и команды их обработки. Результаты выполнения команд направляются в Центральный процессор для дальнейшего использования в ходе вычислительного алгоритма. Способ представления структур данных в локальной памяти Процессора обработки структур должен обеспечивать высокую производительность для всего набора операций.

В разделе 2.5 определен набор команд дискретной математики для вычислительной системы (*Discrete mathematics Instruction Set Computer, DISC*), которая обеспечивает на аппаратном уровне хранения и обработку структур данных в виде множеств ключей и значений.

В третьей главе «Архитектура вычислительной системы с набором команд дискретной математики» рассмотрена архитектура вычислительной системы, обеспечивающей параллельную обработку структур данных на основе множественного потока команд и одиночного потока данных. Функционирование системы соответствует уточненной модели вычислителя (2), формальной модели устройства обработки отношений, и основано на параллельной обработке двух взаимосвязанных частей структур данных: информационной (множество I) и структурной (множество R). Структурная составляющая определяет взаимосвязь данных, в то время как информационная составляющая состоит из самих данных, используемых в ходе вычислительного процесса. Вычислительная система со множественным потоком команд и одиночным потоком данных использует параллельную обработку структур данных на основе *Процессора обработки структур* (далее, СП), который осуществляет прием, хранение и обработку ключей и связанных с ними значений.

В разделе 3.1 приведены основные функции *Процессора обработки структур*, определен и обоснован четвертый принцип.

Принцип множественности потоков команд. *В вычислительной системе с аппаратной поддержкой операций над структурами данных один поток данных обрабатывается несколькими потоками команд.*

Для обработки каждого потока команд используется отдельное устройство: *Центральный процессор* применяется для обработки данных потоком команд из локальной оперативной памяти; *Процессор обработки структур* также связан с локальной памятью команд, а также памятью для хранения структур (Рисунок 3).

Для синхронизации потоков команд, оба процессора обмениваются сообщениями, содержащими данные и результаты. Так как способ синхронизации должен определяться на этапе компиляции кода потоков команд, требуется применение специальных средств разработки и оптимизации программного обеспечения, а также средств компиляции и отладки, входящих в базовое организационно-методическое и лингвистическое обеспечение МКОД системы с набором команд дискретной математики.

Можно привести следующий пример работы ЦП и СП в МКОД системе и их синхронизации (Таблица 1). Пусть ЦП необходимо из множества ключей структуры SOURCE получить все ключи (и ассоциированные с ними значения),

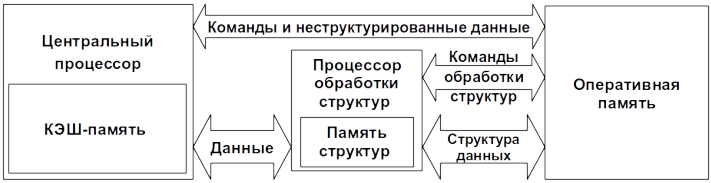

Рисунок 3 — Взаимодействие устройств в вычислительной системе с аппаратной поддержкой операций над множествами и структурами данных

большие некоторого i. Для этого в СП можно использовать команду среза для формирования нового множества RESULT, состоящему из ключей и значений из SOURCE, удовлетворяющих условию $key > i$. Далее, СП достаточно выполнить обход множества от первого до последнего элемента и передать их в ЦП. Отметим, что СП не требуется выполнения каких-либо арифметических действий и, следовательно, синхронизации с ЦП. Для команды GR необходимо получить ключ от ЦП, что обозначено в псевдокоде знаком «?» и командой PUT(j).

Таблица 1 — Пример обработки структур данных в МКОД системе: выборка всех (*key*,*value*) из SOURCE таких, что *key*>*i*

Поток ЦП:	**Поток СП:**
PUT(i)	GR(SOURCE,RESULT,?)
ПОВТОРЯТЬ	(key,$value$)=MIN(RESULT)
GET(key)	@1: PUT(key)
GET($value$)	PUT($value$)
ПОКА $key \neq$ NULL	(key,$value$)=NEXT(RESULT,key)
PUT(break)	JT(@1)
КОНЕЦ	@end:

Пример демонстрирует, что предложенная архитектура обладает возможностью выполнять независимо два потока команд, однако наличие таких потоков не гарантировано алгоритмами дискретной оптимизации. В ряде случаев количество независимых действий в двух потоках оказывается малым, в связи с чем оба процессора будут ожидать синхронизации. Тем не менее стоит отметить, что вынужденное время ожидания готовности данных процессоров (ЦП и СП) возможно использовать для обработки других задач. В главах 4 и 5 предложен ряд решений данной проблемы, позволяющих снизить зависимостей по данным двух потоков команд (арифметической обработки и команд дискретной математики): снижение количества пересылок между процессорами; реализация специализированных регистров для хранения составных ключей; снижение накладных расходов на передачу данных; конвейеризация СП; параллельная обработка нескольких потоков.

В работах Флинна предложена классификация вычислительных машин по количеству потоков команд и данных (предложены четыре класса, отличающиеся множественностью потоков команд и данных (классы ОКОД, ОКМД, МКОД и МКМД). На основе классификации и уточненной модели (2) в работе показано, что вычислительная система с аппаратной поддержкой операций над

множествами и структурами данных соответствует классу МКОД (множественный поток команд и одиночный поток данных). Анализ классификации Флинна и архитектурных особенностей МКОД системы позволил сформулировать принцип иерархической вложенности вычислительных систем различных классов (Рисунок 4).

Принцип иерархической вложенности архитектур вычислительных систем. *При структурной декомпозиции вычислительных систем для решения задач дискретной оптимизации в одной МКМД системе реализуется несколько МКОД систем, в одной МКОД системе реализуется несколько ОКМД систем, а в одной ОКМД системе реализуется несколько ОКОД устройств.*

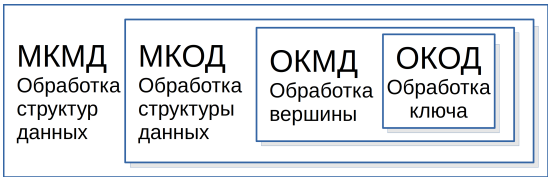

Рисунок 4 — Принцип иерархической вложенности архитектур вычислительных систем

Анализ способов взаимодействия устройств и вариантов обмена информацией в МКОД системе с набором команд дискретной математики, проведенный в разделе 3.2, позволил определить режимы работы *Процессора обработки структур*. Показаны и выявлены достоинства и недостатки трех вариантов организации топологии системы: общей шины, независимых шин и иерархии шин.

Принцип работы Процессора обработки структур. *Процессор обработки структур в вычислительной системе с аппаратной поддержкой операций над структурами данных может работать в режиме МКОД, режиме сопроцессора и в комбинированном режиме.*

Топология, использующая иерархию шин, является универсальной и позволяет подключать СП в виде ускорителя вычислений к существующим вычислительным системам, что позволит на первой стадии внедрения в наибольшей степени использовать возможности предлагаемой архитектуры системы с набором команд дискретной математики. Топология независимой шины (Рисунок 5) хорошо подходит для тех проектов, где необходимо достичь максимальной параллельности и быстродействия при обработке структур, так как такая схема обеспечивает высокую скорость загрузки команд и данных в СП параллельно с работой ЦП. Топология общей шины обеспечивает максимальную унификацию доступа ко всем ресурсам системы со стороны нескольких процессоров, но приводит к необходимости реализации высокоскоростных интерфейсов системной шины, а также порождает большую конкуренцию при доступе к шине.

В разделе 3.3 представлен состав, принципы взаимодействия и функциональное назначение блоков, входящих в СП (Рисунок 6). Согласно предложенным принципам, *Процессор обработки структур* данных подключён к локальным

ЗУ, в которых хранятся структуры данных и команды в соответствии с топологией В+дерева.

Принцип аппаратного поиска в В+дереве. *Последовательность вершин В+дерева, составляющая путь для аппаратного поиска ключа от корня до листа, называется* ***трассой****. Трасса сохраняется во внутренней памяти Процессора обработки структур для возможности оперативного доступа.*

ЗУ команд представляет собой адресную память, в которой располагаются отдельные процедуры управления СП, составляющие поток команд обработки структурной информации. Доступ к содержимому адресного ЗУ структур выполняется с помощью уникальных ключей.

Принцип параллельной аппаратной обработки В+дерева. *Вершина нижнего уровня В+дерева, состоящая из некоторого количества ключей и значений, хранится в специальной памяти внутри Процессора обработки структур (Памяти структур) и называется* ***линейкой****. Информация одной линейки может обрабатываться в СП параллельно.*

Машинные команды СП являются командами высокого уровня. В рассматриваемой версии реализованы команды: поиска, добавления, удаления, минимума/максимума, мощность, И-ИЛИ-НЕ команды, срезы, обход структуры, поиск ближайшего большего и меньшего, удаление и сжатие структур, а также команда ветвления. Обработка команд начинается со стадии загрузки из ОЗУ в *Блок выборки команд* кодов операции, операндов и тегов. Тег — это разряд машинной команды *Процессора обработки структур*, указывающий на достоверность операнда. Команда может быть выполнена только в том случае, когда все ее теги установлены в единичное состояние.

Различные команды содержат различное количество операндов и соответственно, тегов, причем допускается от 1 до 3 операндов. Команда может быть

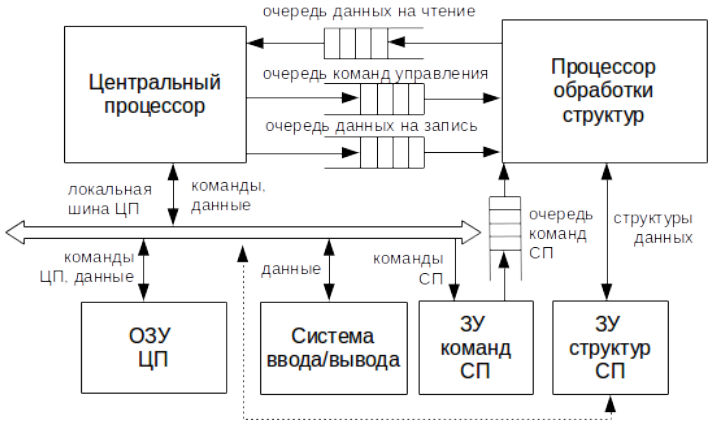

Рисунок 5 — Взаимодействие устройств в системе МКОД с набором команд дискретной математики при топологии независимых шин

выполнена в том случае, когда все ее операнды достоверны, т.е. их теги установлены (в единичное состояние). Такой вариант команды с размещением операндов в ее составе традиционно называется непосредственным.

Если же какой-либо операнд не является валидным, СП ожидает его поступления из ЦП. Последовательность передаваемых тегов для одной команды не должна иметь значения, так как, в общем случае, последовательность поступления операндов из ЦП заранее неизвестна. Получение операнда из ЦП, в этом случае, является способом синхронизации двух процессов в СП и ЦП. При использовании **внешней адресации** *Процессор обработки структур* ожидает поступления операнда от Центрального процессора.

Предложен способ адресации операндов в командах *Процессора обработки структур*, позволяющий повторно использовать уже загруженные операнды. Способ состоит в конструировании составных ключей из частей использованных операндов, что позволяет использовать аппаратную логику для сдвига и маскирования частей ключей.

Принцип составной регистровой адресации. *При* ***составной регистровой адресации*** *операнд формируется из разрядов внутренних регистров Процессора обработки структур на основе маски.*

В разделе 3.4 представлены варианты реализации уникальных устройств: *Каталога, Операционного буфера, Устройства загрузки команд* и др. (Рисунок 6). В основе реализации этих устройств лежит ряд новых архитектурных принципов. Последовательность вершин B+дерева, составляющая путь для аппаратного поиска ключа от корня до листа (трасса) обрабатывается в *Каталоге* (структура *Каталога* показана на Рисунке 7). Совместно с очередной командой, из *Буфера команд* поступает номер структуры и ключ, которые используются при поиске сегмента трассы на каждом уровне B+дерева.

Трасса сохраняется во внутренней памяти *Процессора обработки структур* для возможности оперативного доступа. Вершина нижнего уровня B+дерева, состоящая из некоторого количества ключей и значений, хранится в специальной памяти внутри *Процессора обработки структур* (*Памяти структур*) и называется линейкой. Информация одной линейки может обрабатываться в СП параллельно.

Предлагается следующий вариант представления дерева во внутренней памяти СП. Каждый уровень дерева, кроме нижнего, реализован в СП в виде набора взаимосвязанных вершинных процессоров, которые осуществляют хранение и обработку информации о структуре дерева. Информация для этих уровней загружается по мере необходимости в *Каталог* СП. Для ускоренного поиска и модификации информации в вершинах *Каталога*, они объединены в независимые списки, которые автоматически модифицируются по мере увеличения или сокращения размеров структур. Максимальное количество структур, хранимых в СП, соответствует количеству блоков на верхнем уровне дерева, но не более количества структур, ограниченного атрибутом номера структуры за вычетом

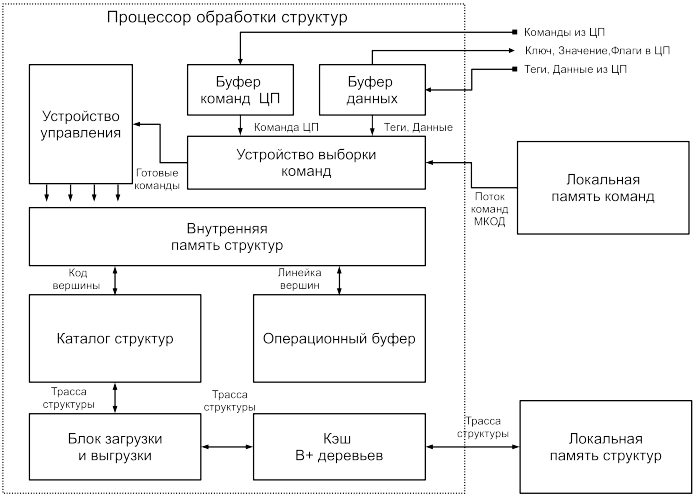

Рисунок 6 — Подсистемы процессора обработки структур

единицы (нулевой, служебной структуры). *Каталог* является уникальным аппаратным устройством, реализованным впервые. В каждой вершине *Каталога* хранится следующая информация:

- Параметры *Low* и *High* – нижняя и верхняя границы ключей в поддереве. Эти параметры используются для определения вершины *Каталога*, в котором может находиться искомый ключ.
- Номер структуры - число, указывающее на принадлежность вершины к определенной структуре. При поиске параметр учитывается для выделения вершин запрашиваемой структуры данных. Нулевая структура является служебной и объединяет все свободные вершины.
- Номера предшествующей и последующей вершины - позволяют выполнять связанную обработку информации на уровне дерева;
- Адрес дочерней вершины *Каталога* в *Памяти структур* СП;
- Атрибуты вершины *Каталога*: флаг переполнения вершины; некоторые дополнительные атрибуты, ускоряющие обработку.

Принцип функционирования Каталога. *Для выполнения вставки, удаления и последовательного обхода поддеревьев в вершине B+дерева все записи одного уровня, принадлежащие одной структуре, объединяются в связанные цепочки записей.*

Цепочки позволяют осуществить не только обход, но и упрощают такие внутренние команды как: выборка поддерева, добавление и удаление. За хранение и обработку одного поддерева отвечает *Вершинный процессор* (Рисунок 7), представляющий собой примитивное исполнительное устройство, выполняющее ряд простых команд обработки информации. Каждый *Вершинный процессор* связан со своей локальной памятью, в которой для всех уровней дерева сохраняются трассы, позволяющие вести поиск в поддеревьях.

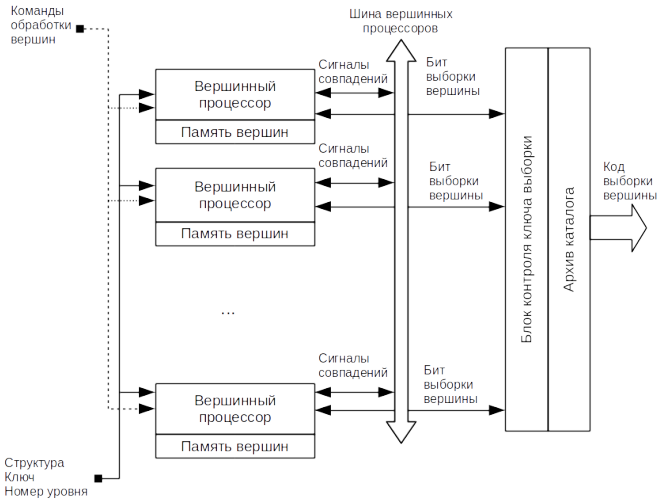

Рисунок 7 — Структура *Каталога*

Операционный буфер (Рисунок 8) предназначен для обработки информации нижнего уровня дерева (линеек). Для этого из *Памяти структур* загружается *B* ключей и значений, причем только часть из них могут быть заняты информацией. Для указания занятых и свободных позиций в линейках для каждой пары сохраняется бит достоверности, указывающий на наличие реальных данных. *Операционный буфер* позволяет выполнять не только основные действия над одной линейкой, но и обеспечивает обработку нескольких линеек в ходе И-ИЛИ-НЕ операций.

Принцип обработки линеек B+дерева. *При добавлении новых ключей и значений в заполненную линейку Операционного буфера возникает ее переполнение, что приводит к выборке следующей линейки при помощи Каталога и повторении операции добавления.*

Для работы *Операционного буфера* необходимо подать на его вход команду управления и ключ, а для операции добавления также передается значение. Результаты поиска записываются в регистры ключа и значения. После добавления или удаления могут измениться границы линейки в блоке R, что требует чтения граничных значений (первого и последнего ключей). Для хранения этих значений предусмотрены регистры нижней границы и верхней границы. Выбор конкретного *Процессорного элемента* осуществляется с помощью установки и анализа битов маски, которые влияют на его поведение в ходе выполнения команд.

Принцип функционирования Операционного буфера. *В каждом процессорном элементе блока результата в Операционном буфере предусмотрено специальное поле маски, которое влияет на его поведение в ходе выполнения команд.*

Устройство выборки команд (Рисунок 9) осуществляет не только выборку кодов команд из Локальной памяти команд (Рисунок 6), но и их подготовку

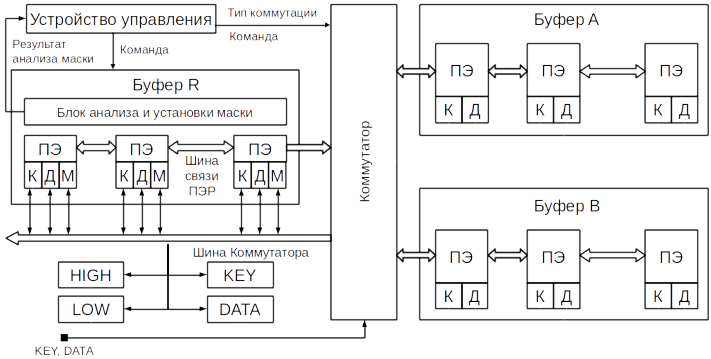

Рисунок 8 — Структура *Операционного буфера*

к исполнению. Были определены 6 форматов команд *Процессора обработки структур* и набор из 17 машинных команд высокого уровня.

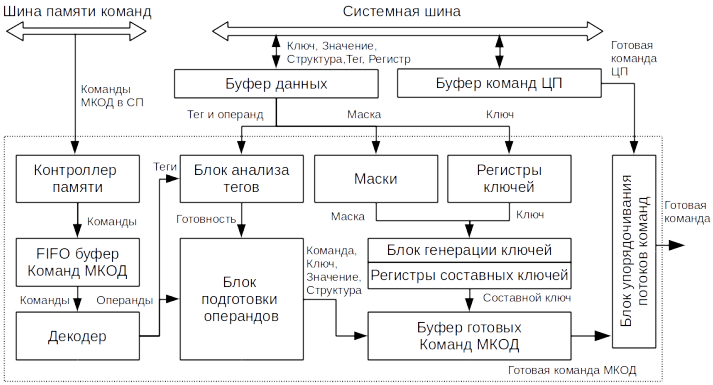

Рисунок 9 — Структура устройства выборки команд

– **Поиск (мнемокод SEARCH)**. СП принимает на вход номер структуры и ключ поиска. По команде СП выполняет поиск по ключу в указанной структуре. Если соответствующая запись найдена, ключ и данные выдаются в качестве результата.
– **Добавление (мнемокод INSERT)**. СП принимает на вход номер структуры, ключ и значение. По команде производится добавление информации в указанную структуру. Если таковой структуры нет, то она создается. Если добавление невозможно (память переполнена), то устанавливается флаг ошибки.
– **Удаление (мнемокод DELETE)**. СП принимает на вход номер структуры и ключ. СП выполняет поиск по указанному ключу, и если такая запись найдена, она удаляется. Если запись не найдена, то устанавливается флаг ошибки.

– **Удаление структуры (мнемокод DELSTR)**. СП принимает на вход номер структуры. По команде СП выполняет удаление всех данных из указанной структуры за O(1).
– **Максимум/Минимум (мнемокод MAX/MIN)**. СП принимает на вход номер структуры. Если такая структура отсутствует, СП устанавливает флаг ошибки. В противном случае выбирается максимальный/минимальный ключ и его значение.
– **Мощность (мнемокод POWER)**. СП принимает на вход номер структуры. По команде выдается количество ключей в указанной структуре.
– **И/ИЛИ/НЕ (мнемокод AND/OR/NOT)**. СП принимает на вход два номера исходных структур (структур-операндов), а также номер структуры результата, в которую помещается результат операций И/ИЛИ/НЕ над всеми элементами структур-операндов. Если элементов, соответствующих условиям команд AND/OR/NOT нет, флаг ошибки не устанавливается, а результирующая структура не создается.
– **Срез «меньше»/«больше»/«меньше или равно»/«больше или равно»/«больше и меньше» (мнемокод LS/GR/LSEQ/GREQ/GRLS)**. СП принимает на вход номер структуры-операнда, номер структуры результата и ключ для указания границы среза. Если структура результата уже существует, устанавливается флаг ошибки и работа команды прерывается. В противном случае в структуру результата добавляются только те элементы из структуры-операнда, ключи которых меньше/больше/меньше или равны/больше или равны границы среза. Для команды GRLS задаются две границы: нижняя для отсечения по условию «больше», верхняя для отсечения по условию «меньше». Если таких элементов нет, флаг ошибки не устанавливается, а результирующая структура не создается.
– **Следующий (мнемокод NEXT) и Предыдущий (мнемокод PREV)**. СП принимает номер структуры. В результате работы команды СП выдает ключ и значение элемента структуры, следующего по порядку ключей за элементом, на который указывает курсор. Если следующего элемента в структуре нет, то команда прерывается и устанавливается флаг ошибки. В последних версиях *Процессора обработки структур* реализованы модификации команды поиска: поиск ближайшего большего (мнемокод *NGR*) и меньшего (мнемокод *NSM*).
– **Сжатие (мнемокод SQUEEZE)**. СП принимает номер структуры. В результате работы команды СП выполняет размещение структуры таким образом, чтобы она занимала минимальное место в *Локальной памяти структур*.
– **Переход по тегу (мнемокод JT)**. СП принимает на вход адрес перехода и тег. Команда доступна только в режиме МКОД, так как служит для управления потоком команд из локальной памяти команд СП.

В разделе 3.4.5 рассмотрены вопросы организации управления блоками СП. Выполнена декомпозиция функций управления на четыре иерархически подчиненных уровня: центральное устройство управления, устройство управление уровнями дерева, управление *Каталогом* и управление *Операционным буфером*. Показано, что при реализации микропрограммных управляющих автоматов на ПЛИС типа FPGA возможности языковых средств и САПР существенно ограничены. В связи с этим в разделе 3.5 была разработана специальная технология динамического микропрограммирования, направленная на ускоренное создание, модификацию и отладку таких устройств. Разработана грамматика языка микропрограммирования, средства генерации и оптимизации синтезируемых описаний микропрограммных устройств на языке VHDL, а также средства трансляции микрокоманд в карту блочной памяти ПЛИС FPGA фирмы Xilinx и изменения конфигурационных файлов. Формальная грамматика языка динамического микропрограммирования относится к регулярным (тип 3 по классификации Хомского).

В разделе 3.6 рассмотрены вопросы построения вычислительных комплексов на основе гибридных устройств, включающих как узлы с Фон-Неймановской архитектурой на основе арифметически-логического набора команд, так и узлы с набором команд дискретной математики. Рассмотрены достоинства и недостатки возможных вариантов интеграции DISC в вычислительные комплексы: в виде ускорителей вычислений, в виде специальных блоков на одном кристалле с универсальным микропроцессором, а также в виде отдельного узла в кластерной вычислительной системе. Также рассмотрены перспективы применения DISC в более далекой перспективе: при реализации масштабной концепции Интернета вещей; для аппаратной поддержки различных сервисов в рамках архитектуры SOA; для реализации функций в цепочках устройств.

В четвертой главе «Оптимизация алгоритмов в вычислительной системе с набором команд дискретной математики» предложены варианты реализации решений на основе новой архитектуры для построения аналитических систем, систем хранения и обработки данных, робототехнических систем, геоинформационных систем. Рассмотрены общие принципы организации программного обеспечения, учитывающего особенности архитектуры со множественным потоком команд и одиночным потоком данных.

Существенной особенностью разработанной архитектуры МКОД DISC является необходимость адаптации известных последовательных алгоритмов с учётом разделения одного потока команд, в рамках которого традиционно описываются алгоритмы, на два потока (команд обработки отношений структур R и команд обработки информационной составляющей данных I). Параллельные ветви алгоритма связаны между собой, так как результаты и операнды передаются между процессорами.

Показаны особенности функционирования программного обеспечения: гетерогенная структура вычислительной системы с различными наборами команд; функционирование *Процессора обработки структур* в режиме сопроцессора

и МКОД режиме; необходимость раздельной компиляции кода *Процессора обработки структур* и *Центрального процессора*; обеспечение распределенного хранения скалярных данных и структур данных. На основе этого делается вывод о необходимости создания специальной технологии разработки программного обеспечения в МКОД системе.

В разделе 4.1.3 выполнен анализ ускорения вычислительного процесса при совмещении операций арифметической обработки и обработки структур данных. Получено выражение коэффициента повышения производительности системы с набором команд дискретной математики при совмещении операций. Показано, что как для алгоритмов с доминирующей арифметической обработкой, так и для алгоритмов с доминирующей обработкой структур данных, для увеличения производительности необходимо:

- сокращение времени обработки структур данных при аппаратной реализации в СП;
- сокращение временных затрат на прием и передачу данных;
- поиск алгоритмов с меньшими зависимостями между арифметической обработкой и обработкой структур данных.

В разделе 4.1.4 приведены примеры описания алгоритмов в параллельном псевдокоде и в виде информационного графа. Сформирована методика преобразования существующих алгоритмов, представленных в парадигме исполнения одного потока команд над одиночным потоком данных, в алгоритмы со множественным потоком команд и одиночным потоком данных.

В разделе 4.2 предложена формальная постановка задачи декомпозиции информационного графа программы, функционирующей в МКОД DISC системе. Любая последовательная программа (последовательный алгоритм) может быть представлена при помощи особой интегральной модели, объединяющей в себе взвешенный двудольный ориентированный граф $G_{\text{о.д.}}$ и управляющий граф $G_{\text{у}}$.

Инструкции и объекты данных представляются в графе вершинами, причём вершины потока управления связаны между собой непосредственно, а вершины данных связаны между собой через поток управления.

Тогда модель программы, обрабатывающей данные, определяется следующим образом:

$$G_{\text{A}}\left(\{X,Y\},F_1X,F_1^{-1}X,F_2X,F_2^{-1}X,F_3Y,F_3^{-1}Y\right)=G_{\text{у}}\cup G_{\text{о.д.}} \qquad (3)$$

В представленной модели приняты следующие обозначения:

$X=\{x_1,x_2,\ldots x_n\}$ – множество вершин графа G_{A}, соответствующих инструкциям программы;

$Y=\{y_1,y_2,\ldots y_3\}$ – множество вершин графа G_{A}, соответствующих элементам данных, причём $X\cap Y=\emptyset$;

F_1X – множество вершин-образов вершин потока управления, причём: $F_1X\subset X$;

$F_1^{-1}X$ – множество вершин-прообразов вершин потока управления, причём: $F_1^{-1}X\subset X$;

F_2X – множество вершин-образов вершин потока управления во множестве вершин, соответствующих обрабатываемым данным, т.е. $F_2X \subset Y$.

$F_2^{-1}X$ – множество вершин-прообразов вершин потока управления во множестве вершин, соответствующих обрабатываемым данным, т.е. $F_2^{-1}X \subset Y$.

F_3Y - множество вершин-образов вершин данных во множестве вершин потока управления, т.е. $F_3Y \subset X$.

$F_3^{-1}Y$ - множество вершин-прообразов вершин данных во множестве вершин потока управления, т.е. $F_3^{-1}Y \subset X$.

Однако, существенным недостатком модели 3 является то, что различия между скалярными типами данных и структурами данных не могут быть учтены в соответствии с Принципом гетерогенности вычислений и (2). Требуется выделить часть вершин графа управления, соответствующего обработке структур данных, и далее детализировать модель для преобразования последовательных программ к виду МКОД.

В связи с этим обозначим O множество инструкций исходной программы, D – множество данных, принадлежащих области определения. При этом $O = O^* \cup O^{**}, O^* \cap O^{**} = \emptyset$, где O^* - множество операторов обработки данных, O^{**} - множество операторов вычисления условий. Тогда множество операторов обработки данных детализируется следующим образом:

$$O^* = O_S^* \cup O_I^*, \tag{4}$$

где O_S^* - множество операторов обработки структур данных, а O_I^* - множество операторов обработки данных примитивных типов, в том числе отдельных элементов структур. В результате разделим все операторы на две пересекающиеся категории:

– операторы, выполнение которых возможно в СП ($O_{\text{СП}} = O_S^*$);

– операторы, выполнение которых возможно в ЦП ($O_{\text{ЦП}} = O_I^* \cup O_S^* = O^*$).

Полагая, что аппаратная реализация механизмов обработки структур данных и множеств повышает степень параллельности системы, сокращает время обработки и/или требует меньших энергии, можно выделить ключевую задачу подготовки программы для ЭВМ МКОД DISC как задачу минимизации количества элементов множества вершин графа управления, выполняемых ЦП, и обратно, увеличение количества вершин графа управления, исполняемых в СП. Введем следующие обозначения:

$O^*_{S\text{возм.}}$ множество возможных значений операторов обработки структур.

$O_{\text{ЦП}}^{\text{действ.}}$ множество операторов, которые действительно выполняются ЦП после осуществления декомпозиции;

$O_{\text{СП}}^{\text{действ.}}$ - множество операторов, которые действительно выполняются СП после осуществления декомпозиции; $O_{\leftrightarrow} = O_{\rightarrow} \cup O_{\leftarrow}$ - множество операторов обмена данными между ЦП и СП, где $O_{\rightarrow}$ - множество операторов пересылки данных из ЦП в СП, $O_{\leftarrow}$ - множество операторов получения данных из СП в ЦП. Стоит отметить, что

$O_{\leftrightarrow} \cap O = \emptyset$, поэтому введём понятие полного множества операторов ЭВМ МКОД. Полное множество операторов ЭВМ МКОД

$O_{\text{МКОД}} = O \cup O_{\leftrightarrow}$.

Аналогичным образом выделим виды обрабатываемых программой данных. Множество обрабатываемых данных D представлено подмножествами $D_S \subset D$ и $D_P \subset D$ (индекс S далее указывает на применение структур данных, индекс P указывает на примитивные типы данных). Таким образом, получим $D = D_S \cup D_P$. Поскольку структуры данных состоят из элементов данных (примитивного типа) и отношений между ними, то каждая структура данных может быть представлена в виде $d_i^S = (DE_i, R_i)$, где DE_i – множество элементов данных примитивного типа i-ой структуры данных, R_i – множество отношений между отдельными элементами DE_i i- ой структуры данных, при этом $d_i^S \in D_S$, тогда как $DE_i \in D_P$.

Тогда детализированная модель программы в принятых обозначениях может быть представлена следующим образом:

$$Dec(G_A) = \begin{cases} G_{\text{AI}}\left(\{X_{IC\leftrightarrow}, Y_P\}, F_1 X_{IC\leftrightarrow}, F_1^{-1} X_{IC\leftrightarrow}, F_2 X_{IC\leftrightarrow}, F_2^{-1} X_{IC\leftrightarrow}, F_3 Y_P, F_3^{-1} Y_P\right) \\ G_{\text{AS}}\left(\{X_{S\leftrightarrow}, Y_S\}, F_1 X_{S\leftrightarrow}, F_1^{-1} X_{S\leftrightarrow}, F_2 X_{S\leftrightarrow}, F_2^{-1} X_{S\leftrightarrow}, F_3 Y_S, F_3^{-1} Y_S\right) \end{cases} \quad (5)$$

В представленной модели G_{AI} – информационный граф алгоритма арифметико-логической обработки данных примитивного типа; G_{AS} – информационный граф алгоритма обработки структур данных. Индекс S далее указывает на отношение оператора к потоку управления СП и применение структур данных, индексы P и I указывает на примитивный тип данных, индекс IC указывает на отношение оператора к потоку управления ЦП. Также приняты следующие обозначения:

X_S – множество вершин обработки структур данных, $O_{\text{СП}}^{\text{действ.}} \leftrightarrow X_S$, $X_S \subset X$;

X_I – множество вершин обработки данных примитивных типов, $O_I^* \leftrightarrow X_I$, $X_I \subset X$;

X_C – множество вершин вычисления условий, $O^{**} \leftrightarrow X_C$, $X_C \subset X$;

$X_{\text{МКОД}}$ – полное множество вершин информационного графа программы для ЭВМ МКОД, представляющее собой отображение $O \cup O_{\leftrightarrow} \leftrightarrow X_{\text{МКОД}}$, при этом $X \subset X_{\text{МКОД}}$;

$X_{\leftrightarrow}$ - множество операторов обмена данными между ЦП и СП, $O_{\leftrightarrow} \leftrightarrow X_{\leftrightarrow}$, $X_{\leftrightarrow} \subset X_{\text{МКОД}}$;

$X_{IC\leftrightarrow}$ – множество вершин обработки элементов данных, данных примитивных типов и вычисления условий, а также пересылки данных между ЦП и СП, $X_{IC\leftrightarrow} = X_I \cup X_C \cup X_{\leftrightarrow}$;

$X_{S\leftrightarrow}$ – множество вершин обработки структур данных, а также пересылки данных между ЦП и СП, $X_{S\leftrightarrow} = X_S \cup X_{\leftrightarrow}$;

Y_S – множество вершин структур данных, $D_S \leftrightarrow Y_S$, $Y_S \subset Y$;

Y_P – множество вершин данных примитивного типа $D_P \leftrightarrow Y_P$, $Y_P \subset Y$.

Остальные обозначения соответствуют принятым в (3).

В разделе 4.3 проведены исследования реализации алгоритмов Форда-Фалкерсона, Дейкстры, алгоритмов обхода графов в ширину и глубину, а также

алгоритмы Крускала и Прима. Исследования показали возможность решения задач на графах на основе параллельно и независимо работающих ЦП и СП в МКОД системе с набором команд дискретной математики. На основе общей схемы алгоритма удалось получить модифицированную версию, представленную в базисных операциях над структурами данных, а также версию с двумя потоками команд и одиночным потоком данных.

Анализ ряда алгоритмов дискретной оптимизации показал, что при их модификации по разработанной в Главе 4 методике преобразования алгоритмов становится возможным сократить зависимости потоков команд *Центрального процессора* и *Процессора обработки структур*. Для рассмотренных алгоритмов количество независимых команд обработки структур варьируется от 70% до 95%.

В разделе 4.4.1 рассмотрены варианты применения системы с набором команд дискретной математики для поддержки СУБД. Показано, что планы выполнения SQL запросов могут использовать аппаратные команды *Процессора обработки структур*, что приводит к большей параллельности обработки запросов в целом. В разделе 4.4.2 приводится структура распределенной информационно-аналитической системы, в которой применение СП позволяет реализовать механизмы защиты данных на основе ролевой, мандатной или дискреционной модели. В разделе 4.4.3 рассмотрен алгоритм построения графа видимости, используемый в робототехнических системах. Показана возможность реализации такого алгоритма на основе МКОД архитектуры. Вместе с этим выявлен сценарий, реализуемый неэффективно в разработанной версии DISC системы: реализация динамически переупорядочиваемого множества на основе статических индексов. Предложен вариант реализации такого алгоритма при размещении структуры в локальной памяти ЦП. Вместе с этим предложен вариант модификации *Процессора обработки структур* для поддержки подобного рода функциональности.

В пятой главе «Реализация и экспериментальные исследования эффективности вычислительной системы с набором команд дискретной математики» представлены результаты научно-исследовательской и опытно-конструкторской работ по созданию действующего образца вычислительной системы с аппаратной поддержкой операций над множествами и структурами данных. В разделе 5.1 приведено описание процесса проектирования и верификации *Процессора обработки структур* с набором команд DISC, включающего двадцать последовательных этапов. Показана трудоемкость проведенных автором научных исследований и опытно-конструкторских работ.

Рассмотренная в работе версия *Процессора обработки структур* была реализована на базе микросхем ПЛИС Xilinx Virtex II Pro, Virtex 6 и Ultrascale+. В разделе 5.1.3 представлено формальное описание правил грамматики языка ассемблера и структура программы *Процессора обработки структур*. Для каждой команды ассемблера показан способ кодирования, а также приведены примеры использования. Разработан транслятор ассемблера и технология сборки программного кода в DISC системе. Помимо этого представлены краткие сведения

о библиотеке обработки множеств и структур данных на языке C++ (включая библиотеку для обработки и визуализации графов), а также необходимое системное программное обеспечение (драйверы устройства для ОС Linux и ОС Windows).

В разделе 5.2 приводятся результаты комплекса процедур верификации модели *Процессора обработки структур* и опытного образца, а также представлен состав тестов. Были использованы методы функционального и структурного тестирования. В результате функционального тестирования были использованы методы анализа причинно-следственных связей и предположения об ошибке. Наибольшую сложность для СП представляют случаи переполнения или сильной фрагментации памяти. Структурное тестирование применялось для проверки корректности микропрограмм управления: был использован метод покрытия условий. Для каждой команды был составлен набор структурных тестов, учитывающих особенности аппаратной реализации и структуру микропрограммы. В результате все тесты были пройдены успешно.

В разделе 5.3 приведены результаты экспериментальных исследований полученной DISC системы с помощью специально разработанных программных тестов. В качестве основного исследуемого параметра было выбрано количество тактов для реализации алгоритмов и отдельных операций, так как такой подход минимизирует влияние тактовой частоты и технологических ограничений технологии ПЛИС на результаты исследований.

Результаты экспериментов показывают, что *Процессор обработки структур* позволяет повысить эффективность вычислительной системы, так как в проведенных экспериментах результат был получен за меньшее количество тактов в сравнении с микропроцессорами Intel x86 (Intel Core i5,Intel Xeon Gold 7 серии, Intel Xeon Platinum 8 серии). Ряд исследований был направлен на применение высокоэффективных структур данных в универсальных системах, в частности: ART деревьев. При этом тактовая частота опытного образца для выбранной технологии ПЛИС составляет от 100 до 267 МГц, что в ряде экспериментов не позволило получить ускорения алгоритма по времени. Удалось существенно ускорить такие команды, как команда вставки и поиска случайных ключей для программной реализации ART деревьев (см. таблицу 3). Количество тактов выполнения аналогичных команд в *Процессоре обработки структур* оказалось несколько выше, что показывает необходимость дальнейшего совершенствования архитектуры микропроцессора DISC. Для всех остальных вариантов применения универсальных микропроцессоров DISC диск показал более высокую эффективность. При этом, аппаратная сложность *Процессора обработки структур* в 30 раз меньше, чем у одного ядра микропроцессора Intel Xeon Platinum.

Было проведено экспериментальное сравнения архитектурной эффективности обработки графов в вычислительной системе с СП и системах на основе GPGPU. Большинство команд *Процессора обработки структур* требуют $O(log_8 n)$ операций доступа к памяти (исключая операции AND/OR/NOT и операции срезов, которые требуют $O(n)$ операций). В следствие эффективной аппаратной реализации механизмов обработки B+деревьев, команды СП

выполняются за меньшее количество тактов по сравнению с универсальными процессорами и GPGPU. Результаты экспериментов продемонстрировали преимущества многоядерных систем на основе графических ускорителей: количество тактов для решения задачи оказывается в 10^2 раз меньше в сравнении с *Процессором обработки структур* и в 10^3 меньше Intel Xeon E5. Однако, следует принимать во внимание аппаратную сложность GPGPU, выраженную в количестве ресурсов кристалла (транзисторов или вентилей), необходимых для их реализации. Например, аппаратная сложность GPGPU NVidia Tesla K80 составляет порядка 7,1 млрд. транзисторов, в то время как аппаратная сложность *Процессора обработки структур* составляет от 1 до 2.5 млн. вентилей в зависимости от версии (вместе с микропроцессорными ядрами PowerPC, Microblaze или RISCV). В результате оказалось, что аппаратная эффективность графических ускорителей при решении задач на графах в 10^2 раз ниже специализированного СП (Рисунок 10).

Было также проведено сравнение количества тактов, необходимых для выполнения потоковых операций над программной и аппаратной реализаций B+деревьев. Такой тест дает представление об архитектурной эффективности не только микропроцессора в отдельности, но и всей системы, так как при этом учитывается скорость передачи данных и анализа результатов для действий над аналогичной структурой. Сравнение эффективности СП и универсальных микропроцессоров приведено в таблице 2. Показано, что высокий уровень производительности современных многоядерных микропроцессоров и ускорителей GPGPU достигается благодаря более высокой аппаратной сложности. При этом вклад в прирост производительности одного транзистора кристалла оказывается существенно ниже, чем у предшествующих моделей или менее параллельных микропроцессоров.

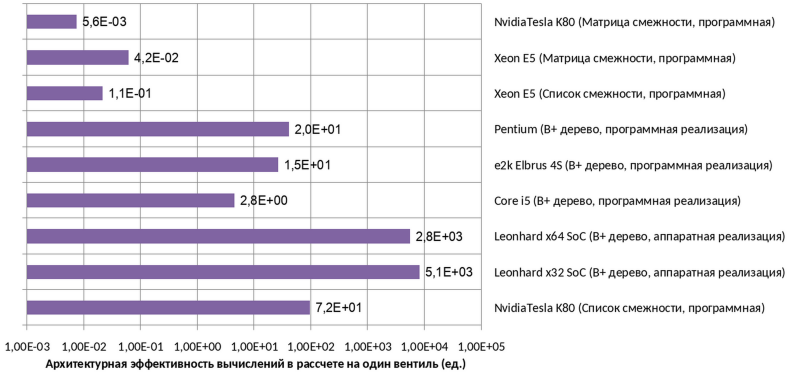

Рисунок 10 — Сравнение архитектурной эффективности вычислительных систем в расчете на один вентиль

Приведенные эксперименты позволили оценить производительность тестовых систем при выполнении базовых операций над множествами: вставка,

поиск, удаление, обход множеств и других. Эксперименты построены на основе формирования и обработки множества, состоящего из 1 миллиона элементов. Как видно из таблицы 2, во всех проведенных экспериментах в МКОД системе было достигнуто архитектурное ускорение при выполнении базовых операций дискретной математики.

Таблица 2 — Экспериментальное исследование архитектурного ускорения *Процессора обработки структур* для множества из 1 млн. элементов

Эксперимент	**Тактов на операцию (Intel Core i5)**	**Тактов на операцию (DISC)**	**Арх. ускорение DISC, разы**
Последовательная вставка	3201	284	11,3
Вставка случайных ключей	3298	441	7,5
Последовательный поиск	1903	30	62,6
Поиск по случайным ключам	3870	277	14,0
Последовательное удаление	2921	71	41,0
Удаление случайных ключей	6485	385	16,8
Обход множества	2017	45	45,1
Поиск ближайшего	3729	287	13,0

На основе результатов сравнения аппаратной эффективности разработанной системы с универсальными ЭВМ в главе делается вывод о том, что наиболее перспективными направлениями применения систем с набором команд дискретной математики являются: автономные роботизированные системы, средства аппаратной поддержки СУБД в распределенных системах, средства поддержки специальных алгоритмов оптимизации на графах, устройства сетевой маршрутизации.

Представленные в работе принципы были воплощены в высокопроизводительном вычислительном кластере «Тераграф» с набором команд дискретной математики DISC. Суперкомпьютер «Тераграф» предназначен для хранения и обработки графов сверх большой размерности и будет применяться для моделирования биологических систем, анализа финансовых потоков в режиме реального времени, для хранения знаний в системах искусственного интеллекта и в других прикладных задачах. Он способен обрабатывать графы сверхбольшой размерности до 10^{12} (одного триллиона) вершин и $2 * 10^{12}$ ребер. Комплекс состоит из 3-х однотипных гетерогенных узлов, которые взаимодействуют между собой через высокоскоростные сетевые подключения 100Gb Ethernet. Структурная схема одного узла представлена на Рисунке 11. Архитектура системы разработана по результатам анализа возможных вариантов реализации DISC систем, представленных в разделе 3.6. Выбраны следующие принципы организации комплекса:

– для ускорения обмена данными внутри DISC системы устройства обработки множеств и структур данных размещаются на одном кристалле с универсальным процессорным устройством, и образуют так называемое ядро обработки графов (Graph Processing Core), показанное на Рисунке 12;

– микропроцессор Леонард Эйлер представляет собой несколько DISC систем (групп ядер, Core Groups), которые имеют независимые каналы памяти и размещаются на одном кристалле;
– несколько вычислительных узлов объединены в комплекс высокоскоростными сетевыми интерфейсами.

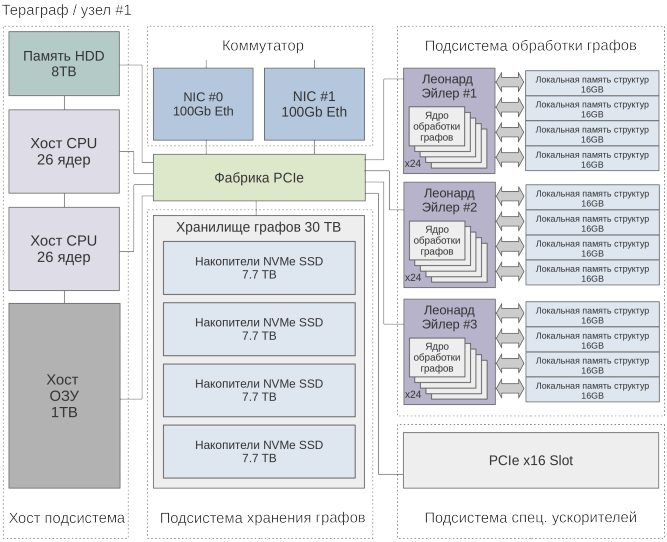

Рисунок 11 — Структура гетерогенного узла

Функция управления работой программных компонент системы реализованы в *Программном ядре хост-подсистемы* (*host software kernel*) – программном обеспечении, взаимодействующим с подсистемой обработки графов через шину PCIe. Со стороны подсистемы обработки графов реализовано до 24-х гетерогенных DISC ядер GPC (Рисунок 13), каждое из которых содержит обрабатывающие элементы SPE (Процессор обработки структур) и универсальный вычислительный элемент CPE. CPE реализован на базе микропроцессора с набором команд RISCV 64IM, на котором функционирует так называемое *Программное ядро ускорителя* (*software kernel*) – специальная программа, передаваемая в бинарном виде в локальную ОЗУ CPE в процессе инициализации и взаимодействующая с SPE.

В подсистему хранения графов входят основная память, состоящая из четырех твердотельных дисков. Подсистема коммутации узлов представляет собой два сетевых модуля связи по протоколу 100Gb Ethernet, позволяющих организовать соединение каждого гетерогенного узла с каждым другим узлом комплекса. Шина PCIe обеспечивает высокопроизводительное взаимодействие хост-подсистемы с процессорами Леонард Эйлер, а также последних с подсистемой хранения графов. Подсистема обработки графов состоит из 3 или 4 карт (в зависимости от версии) многоядерных процессоров Леонард Эйлер версии 4, каждая из которых включает 3 или 4 группы гетерогенных ядер. В группу входят от 2-х до 6-ти

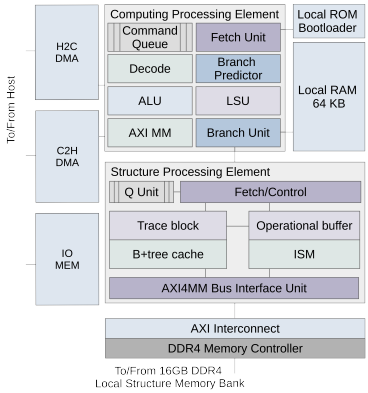

Рисунок 12 — Архитектура ядра обработки графов (GPC)

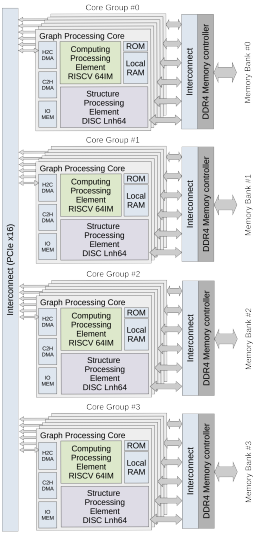

Рисунок 13 — Микроархитектура 24-х ядерного микропроцессора Леонард Эйлер

гетерогенных обрабатывающих ядра GPC, обладающие следующими характеристиками:

- Объем доступной Локальной памяти структур: до 2.5 ГБайт.
- Разрядность ключей и значений: 64 бита.
- Количество хранимых ключей и значений: до 117440512.
- Количество одновременно хранимых структур в Локальной памяти структур: до 7
- Объем ОЗУ CPE: 64 КБ.
- Взаимодействие DISC ядер и хост-подсистемы осуществляется через FIFO буферы и через адресуемую *Глобальную память* (*Global memory, GM*) размером 128 КБ.

Таким образом, многоядерный процессор Леонард Эйлер версии 4 может содержать до 24-х гетерогенных обрабатывающих ядра GPC, а весь комплекс «Терапраф» может содержать до 288 гетерогенных обрабатывающих ядра GPC.

Были проведены тесты производительности многоядерного микропроцессора Леонард Эйлер в составе вычислительного комплекса «Терапраф», которые позволяют сравнить эффективность различных версий оборудования и производительность GPC с процессорами x86 (см. таблицы 3 и 4). В экспериментах использовались три вида программных структур данных: (i) красно-черное дерево (были использованы Red-Black-Tree из стандартного класса MAP библиотеки C++ STL), (ii) программная реализация B+деревьев и (iii) наиболее эффективные из найденных программных структур данных - так называемые Adaptive

Radix Tree деревья. Несмотря на низкую частоту и невысокую аппаратную сложность, скорость операции последовательного поиска на GPC оказалась выше, чем у программных Red-Black или B+ деревьев.

Таблица 3 — Сравнение производительности для основных команд дискретной математики

Характеристика	Леонард Эйлер	Intel Xeon Platinum 8168		
Тип структуры данных	B+ дерево	Красно-черное дерево	B+ дерево	ART дерево
DISC команды	hardware	software	software	software
Использовано ядер	1	1	1	1
Тактовая частота (MHz)	267	2700	2700	2700
Вентилей на ядро (млн)	2.5	83	83	83
Последовательная вставка (ops./sec.)	1179495	719948	2463484	7356619
Вставка случайных ключей (ops./sec.)	200607	667519	1358625	5278771
Последовательный поиск (ops./sec.)	7570905	1168116	6354895	15860176
Поиск по случайным ключам (ops./sec.)	329867	614191	1341165	4812921
Последовательное удаление (ops./sec.)	1649342	788151	2689176	11228889
Удаление случайных ключей (ops./sec.)	313392	491050	941129	3903154

Таблица 4 — Результаты измерения скорости выполнения основных операций многоядерным микропроцессором Леонард Эйлер v4

Характеристика	Значение			
Количество ядер	1	6	12	24
Тактовая частота Леонард Эйлер (MHz)	190	190	190	180
Последовательная вставка (ops./sec.)	875007	5221048	10471506	19840700
Вставка случайных ключей (ops./sec.)	159625	917561	1834893	3476651
Последовательный поиск (ops./sec.)	5427726	32566360	65135016	123413715
Поиск по случайным ключам (ops./sec.)	267749	1546293	3093292	5860974
Последовательное удаление (ops./sec.)	1240522	7257526	14510883	27494305
Удаление случайных ключей (ops./sec.)	253154	1441458	2883251	5463000
Длительность поиска LSM, 1M ключей (msec.)	271	45	23	13
Поиска ближайшего (ops./sec.)	271284	1553684	3104482	5882176
Операция AND (1M ключей) (msec.)	377	64	32	17
Операция OR (1M ключей) (msec.)	401	68	34	18
Операция NOT (1M ключей) (msec.)	396	68	34	18

Эффективность микроархитектуры следует считать выше в том случае, когда тратится меньше циклов и расходуется меньше вентилей для выполнения инструкций. Поэтому был принят следующий вариант расчета аппаратной эффективности на основе количества вентилей, необходимых для реализации устройств: эффективность $E_{HW} = 1/(TSC \cdot GATES)$, где TSC — количество тактов, необходимых для выполнения; GATES — это количество вентилей для аппаратной реализации. В результате оказалось, что аппаратная эффективность процессора Xeon Platinum 8168 от 160,2 раза (последовательный поиск в ART) до 12,75 раз ниже (случайная вставка ART) по сравнению с аппаратной реализацией B+ дерева в микропроцессоре обработки структур Леонард Эйлер.

Вычислительный комплекс был реализован и установлен в ЦОД МГТУ им. Н.Э.Баумана. На Рисунке 14 представлен внешний вид комплекса, использованного при анализе данных в ходе практикума, проведенного для 250 студентов МГТУ им. Н.Э.Баумана в 2022 году. Материалы практикума доступны по адресу: https://alexbmstu.github.io/2022/.

На Рисунке 15 представлен результат поиска сообществ вершин на основе модулярности (алгоритм Фрухтермана-Рейнгольда) и раскладки сообществ в пропорциональные области (алгоритм гильотинного деления).

Рисунок 14 — Конструктив высокопроизводительного вычислительного комплекса Тераграф

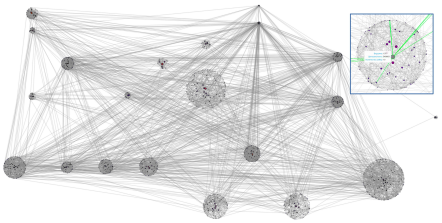

Рисунок 15 — Визуализация графа 4К вершин на вычислительном комплексе Тераграф

В **заключении** проведен анализ основных положений диссертационной работы, изложены основные практические и научные результаты, а также приведены перспективные направления внедрения результатов работы для решения практических задач.

Список публикаций автора по теме диссертации

1. Электронная вычислительная машина с многими потоками команд и одним потоком данных: пат. 71016 Рос. Федерация : МПК7 G 06 F 15/16 / А. Ю. Попов. № 2006115810 ; заявл. 06.05.2006 ; опубл. 20.02.2008, Бюл. № 5. 1 с.: ил. (0.05 п.л.)
2. Попов, А. Ю. Электронная вычислительная машина с аппаратной поддержкой операций над структурами данных // Аэрокосмические технологии : Тр. Второй Междунар. научно-техн. конф., посвященной 95-летию со дня рождения академика В.Н. Челомея (2009). Т. 1. ОАО ВПК «НПО машиностроения». МГТУ им. Н.Э. Баумана, 2012. С. 296—301. (0.4 п.л.)
3. Попов, А. Ю. Применение вычислительных систем с многими потоками команд и одним потоком данных для решения задач оптимизации // Инженерный журнал: наука и инновации. 2012. № 1. URL: http://engjournal.ru/catalog/it/hidden/80.html (дата обр. 27.02.2020). (1 п.л.)
4. Попов, А. Ю. Исследование производительности процессора обработки структур в системе с многими потоками команд и одним потоком

данных // Инженерный журнал: наука и инновации. 2013. № 11. URL: http://engjournal.ru/catalog/it/hidden/1048.html (дата обр. 27.02.2020). (1 п.л.)
5. Попов, А. Ю. О реализации алгоритма Форда-Фалкерсона в вычислительной системе с многими потоками команд и одним потоком данных // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2014. № 9. С. 162—180. (1.2 п.л.)
6. Попов, А. Ю. Реализация алгоритмов обхода графов в вычислительной системе с многими потоками команд и одним потоком данных // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2015. № 10. С. 453—472. (1 п.л.)
7. Попов, А. Ю. Исследование вариантов реализации алгоритмов Крускала и Прима в вычислительной системе с многими потоками команд и одним потоком данных // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2015. № 11. С. 505—527. (1 п.л.)
8. Подольский, В. Э., Попов, А. Ю. Методика декомпозиции информационного графа программы для организации параллельной обработки данных на ЭВМ МКОД // Вестник МГТУ им. Н.Э.Баумана. Сер. Приборостроение. 2016. № 1(106). С. 112—128. (1 п.л./0.5 п.л.)
9. Методы ситуационного анализа и графической визуализации потоков больших данных / А. Ю. Попов [и др.] // Вестник МГТУ им. Н.Э.Баумана. Сер. Приборостроение. 2018. № 2(119). С. 98—103. (1 п.л./0.15 п.л.)
10. Попов, А. Ю. Применение гетерогенной вычислительной системы с набором команд дискретной математики для решения задач на графах // Информационные технологии. 2019. Т. 25, № 11. С. 682——690. (0.6 п.л.)
11. Попов, А. Ю. Принципы организации гетерогенной вычислительной системы с набором команд дискретной математики // Информационные технологии. 2020. Т. 26, № 2. С. 67——79. (1 п.л.)
12. Popov, A. An Introduction to the MISD Technology // HICSS50. Proceedings of the 50th Hawaii International Conference on System Sciences, 2017. P. 1003—1012. (1 п.л.)
13. Abdymanapov, C., Popov, A.Y. Motion Planning Algorithms Using DISC // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). 01/2019. P. 1844—1847. (0.4 п.л./0.2 п.л.)
14. Rasheed, B., Popov, A.Y. Network Graph Datastore Using DISC Processor // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). 01/2019. P. 1582—1587. (0.4 п.л./0.2 п.л.)
15. Decision Support System to Prevent Crisis Situations in the Socio-political Sphere / A.Yu. Popov [et al.] // Studies in Systems, Decision and Control. Vol. 260. 2020. P. 301—314. (0.8 п.л./0.2 п.л.)

16. Li, J., Makarychev, M., Popov, A.Y. Alternative Approach to Solving Computer Vision Tasks Using Graph Structures // Studies in Systems, Decision and Control. Vol. 260. 2020. P. 63—78. (1 п.л./0.3 п.л.)
17. Dubrovin E., Popov A. Graph representation methods for the Discrete mathematics Instructions Set computer // 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Saint Petersburg and Moscow, Russia, 2020, pp. 1925-1930. (0.4 п.л./0.2 п.л.)
18. Multiple Objects Association System for the Smart City / A. Popov [et al.] // 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg, Moscow, Russia, 2021, pp. 2211-2216. (1 п.л./0.5 п.л.)
19. Popov, A., Ibragimov, S., Dubrovin, E. Teragraph Heterogeneous System for Ultra-large Graph Processing. / In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds) // Supercomputing. RuSCDays 2022. Lecture Notes in Computer Science, Springer, Cham. 2022. Vol. 13708, pp. 574-590. (1 п.л./0.5 п.л.)