**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# РЕФЕРАТ

**по дисциплине «Системы параллельной обработки данных»**

**Тема: Параллельные вычисления на GPU**

Студент гр. 9303 _____ Колованов Р.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург

2023

**ЗАДАНИЕ**

**НА РЕФЕРАТ**

Студент Колованов Р.А.

Группа 9303

Тема реферата: Параллельные вычисления на GPU

Исходные данные:

- Темы рефератов выбирается студентами самостоятельно и утверждаются преподавателем;
- Для начала необходимо согласовать задание на реферат, которое включает в себя введение, источники и содержание. После согласования задания необходимо предоставить окончательный реферат;
- Число источников, используемых и цитируемых в работе, должно быть не менее семи, при этом не менее двух из них англоязычных.

Предполагаемый объем реферата:
Не менее 16 страниц.

Дата выдачи задания: 24.09.2023

Дата сдачи реферата: 08.12.2023

Дата защиты реферата: 08.12.2023

Студент _____ Колованов Р.А.

Преподаватель _____ Татаринов Ю.С.

## АННОТАЦИЯ

В данной работе были рассмотрены параллельные вычисления на GPU. Была рассмотрена архитектура вычислительной системы с GPU и ее классификация среди параллельных вычислительных систем. Были рассмотрены области применения параллельных вычислительных систем с GPU. Были рассмотрены история развития и примеры реализации вычислительных систем с GPU. Была рассмотрена программная модель CUDA. В завершение были рассмотрены перспективы развития параллельного программирования на GPU.



## SUMMARY

In this paper, parallel computing on the GPU was considered. The architecture of a GPU computing system and its classification among parallel computing systems were considered. The areas of application of parallel computing systems with GPU were considered. The history of development and examples of implementation of computing systems with GPU were considered. The CUDA software model was considered. In conclusion, the prospects for the development of parallel programming on the GPU were considered.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Параллельные вычисления на графическом процессоре (GPU) стали неотъемлемой частью современных вычислительных систем. В последние годы наблюдается стремительный рост интереса к их использованию для решения широкого спектра вычислительных задач, включая научные и инженерные расчеты, анализ данных, глубокое обучение и многое другое [4,5]. Эта тенденция объясняется несколькими причинами.

Во-первых, с ростом объема данных и сложности вычислений стандартные центральные процессоры (CPU) сталкиваются с ограничениями в производительности. В таких ситуациях графические процессоры (GPU) предлагают значительное преимущество за счет своего параллельного устройства и способности обрабатывать одновременно большой объем данных.

Во-вторых, графические процессоры (GPU) широко используются в области научных исследований, где требуются высокопроизводительные вычисления для моделирования физических процессов, анализа данных и выполнения сложных математических расчетов. В таких случаях параллельные вычисления на GPU обеспечивают высокую производительность и снижение времени выполнения задач. Еще одной важной областью применения GPU является машинное обучение. Алгоритмы глубокого машинного обучения требуют обработки больших объемов данных и выполнения множества параллельных вычислений, для чего как раз подходит GPU, обеспечивающий ускорение процесса обучения моделей.

На протяжении последних лет параллельные вычисления на GPU являются активным объектом исследований и разработок. Существуют широкодоступные специализированные программные фреймворки, такие как CUDA и OpenCL, которые предоставляют разработчикам инструменты для эффективного использования GPU в параллельных вычислениях. Тем не менее, продолжаются исследования и разработка новых алгоритмов и методов, которые бы еще больше оптимизировали параллельные вычисления на GPU.

Возникают новые задачи и проблемы, такие как оптимальное распределение вычислительных задач, синхронизация потоков и управление памятью. Исследователи и разработчики активно работают над решением этих проблем, чтобы повысить производительность и эффективность параллельных вычислений на GPU [4,5].

Целью данного исследования является рассмотрение параллельного программирования на GPU с использование технологии CUDA.

Для достижения поставленной цели были решены следующие задачи:

1. Рассмотрение архитектуры вычислительной системы с GPU и ее классификация среди параллельных вычислительных систем;
2. Рассмотрение области применения параллельных вычислительных систем с GPU;
3. Рассмотрение истории развития и примеров реализации вычислительных систем с GPU;
4. Рассмотрение программной модели CUDA;
5. Рассмотрение перспектив развития параллельного программирования на GPU.

# 1. ОПИСАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С GPU

## 1.1. Архитектура вычислительной системы с GPU

### 1.1.1. Архитектура GPU

Конструктивно графический процессор (GPU) представляет собой вычислительное устройство, работающее параллельно с центральным процессором (CPU), отдельно от него. Вместе с GPU на графической карте расположена видеопамять – специализированная оперативная память, в которой хранятся обрабатываемые графическим процессором массивы данных.

Главной характеристикой архитектуры GPU является то, что GPU представляет собой систему из параллельных вычислительных устройств, каждое из которых применяет заданную, единую для всех устройств, программу к различным элементам входных массивов данных, расположенных в общей памяти. В качестве примера на рис. 1.1 представлена архитектура GPU NVIDIA G80 [3].

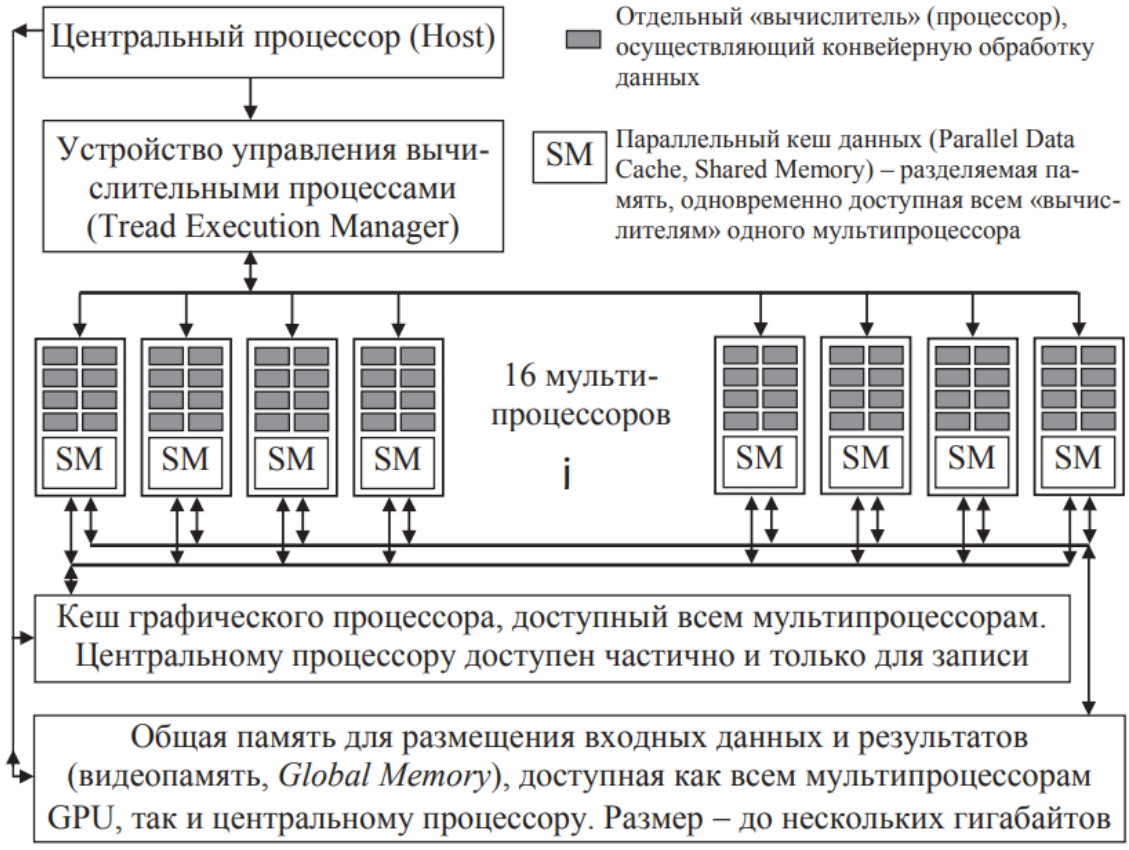


Рисунок 1.1 – Архитектура GPU NVIDIA G80.

Параллельная архитектура GPU ориентирована на исполнение алгоритмов, в которых элементы больших входных массивов обрабатываются одинаковым образом независимо или почти независимо друг от друга, то есть использующих распараллеливание вычислений по данных. Множества элементов, подвергаемых однотипной независимой обработке, называют потоками данных, так что графические процессоры осуществляют поточно-параллельную обработку данных (рис 1.2) [3].

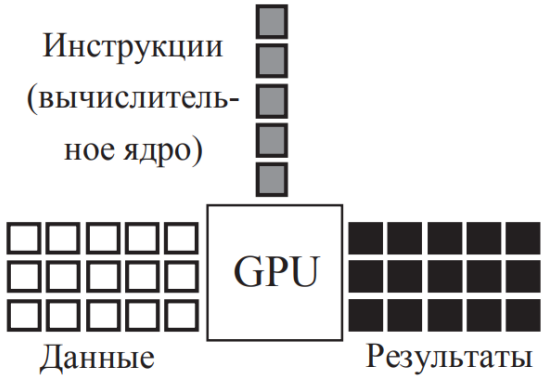


Рисунок 1.2 – Поточно-параллельная обработка данных в GPU.

Выбор такой архитектуры для GPU обусловлен тем, что она обеспечивает параллельное использование большого количества «вычислителей» без явного управления ими: распределения задач, синхронизации вычислений и коммуникации между параллельными расчетами. Разработчикам GPU это позволяет за счет упрощения архитектуры добиваться большей производительности, а при программировании сокращает работу [3].

В графическом процессоре используется конвейерная обработка потоков данных. Она заключается в том, что вся обработка потока данных разбивается на последовательные блоки. Каждый блок выполняет некоторую конкретную задачу, для которой выделена нужная часть аппаратуры. Последовательное же выполнение блоков позволяет решать исходную и более крупную задачу. Программа для выполнения в рамках некоторого блока называется шейдером. В качестве иллюстрации принципа конвейерной обработки данных на рис. 1.3

показаны основные этапы графического конвейера, используемого GPU для визуализации трехмерных сцен [3].

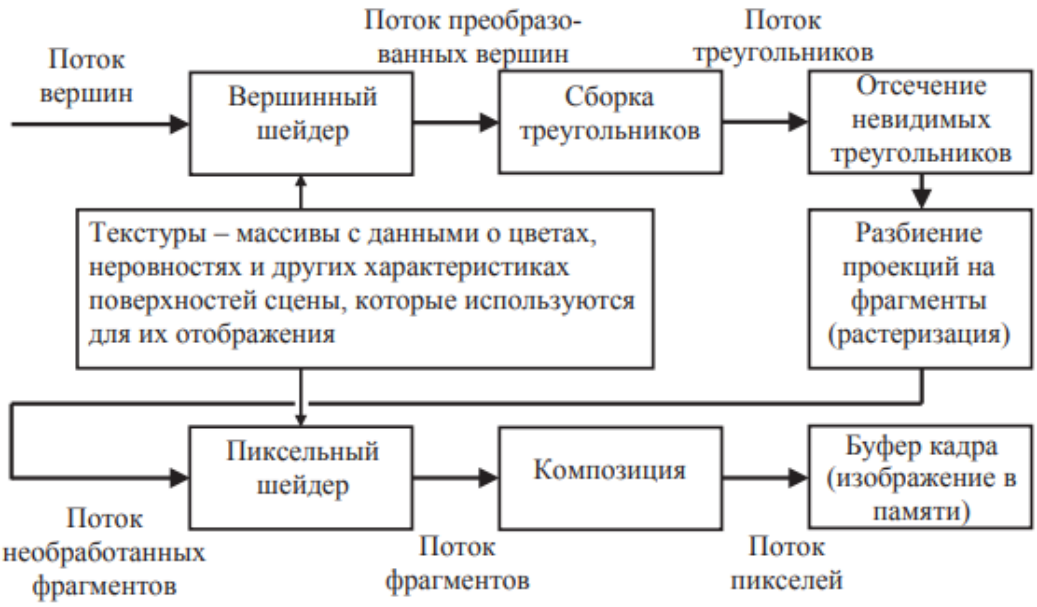


Рисунок 1.3 – Пример графического конвейера для визуализации трехмерных сцен.

### 1.1.2. Взаимодействие с GPU

Графический процессор не имеет средств прямого взаимодействия с устройствами ввода-вывода (кроме монитора), а также доступа к оперативной памяти компьютера. Поэтому управление GPU осуществляется только через центральный процессор. Схема взаимодействия центрального и графического процессоров приведена на рис. 1.4 [3].

GPU подключаются к системной плате персонального компьютера через высокоскоростную шину данных. Посредством этой шины CPU получает доступ к видеопамяти, а также к некоторым разделам кэш-памяти, расположенной на самом графическом процессоре, загружает в GPU программу (набор шейдеров) и запускает ее [3].

Перед запуском программы, исполняемой на GPU, CPU передает графическому процессору данные двух видов:

- Значения констант (записываются в кэш-память);

- Один или несколько больших массивов данных для потоковой обработки (записываются в видеопамять).

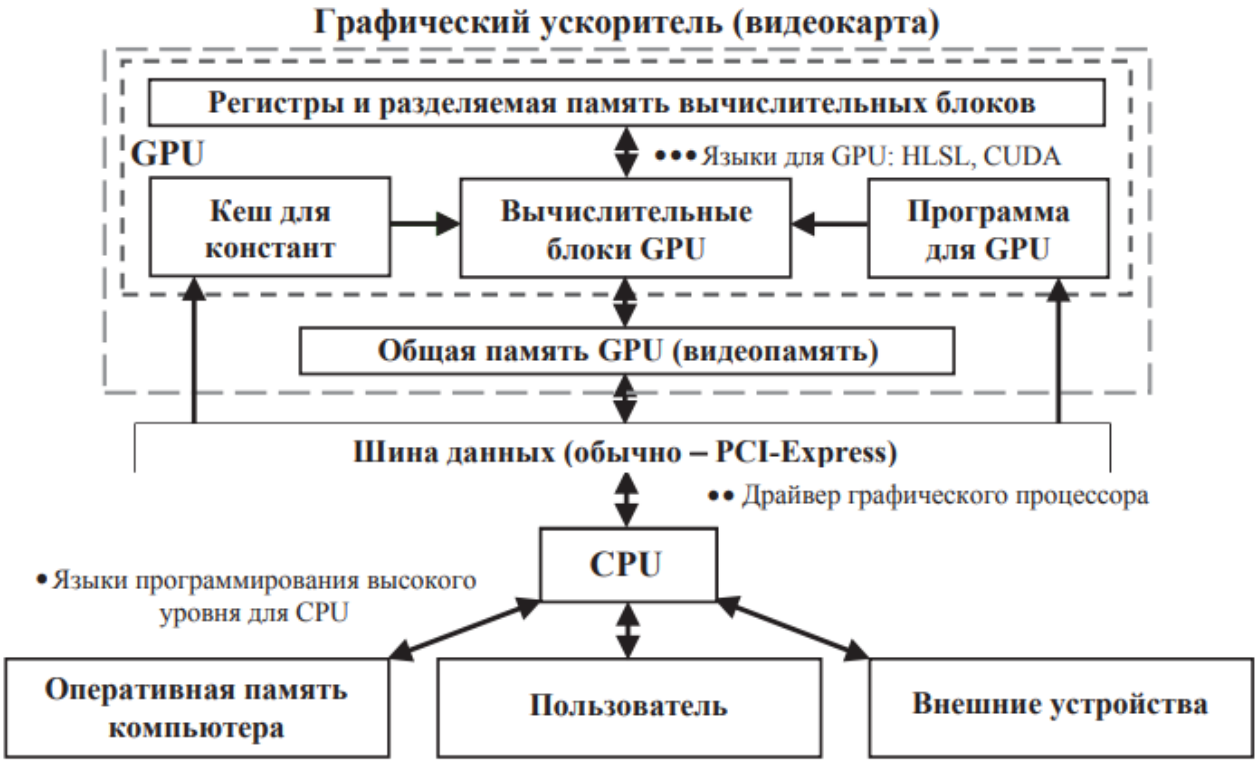


Рисунок 1.4 – Схема взаимодействия CPU и GPU с памятью и между собой.

Кэш-память и видеопамять отличается тем, что кэш-память обладает большой скоростью доступа к данным, но маленьким объемом, а видеопамять – наоборот – обладает медленной (в сравнении с кэш-памятью) скоростью доступа к данным, но большим объемом [3].

Результаты своей работы графический процессор может сразу записывать в раздел видеопамяти, называемый буфером кадра, откуда они передаются на монитор. Но существует также возможность вообще не отображать расчет на экране, а копировать результаты из видеопамяти в оперативную память компьютера, где они становятся доступными для дальнейшей обработки центральным процессором. На этом и основано использование GPU для вычислений общего назначения, не связанных с обработкой графики [3].

### 1.2. Место в классификации параллельных вычислительных систем

Вычислительная система с GPU в классификации параллельных архитектур по Флинну относится к SIMD (Single Instruction, Multiple Data) – вычислительная система с одиночным потоком команд и множественным потоком данных [1,3,6].

### 1.3. Область применения

Современные графические процессоры могут эффективно решать очень многие задачи. Основное требование к таким задачам — возможность распараллеливания по данным или по задачам, а также алгоритм, в котором вычисления преобладают над условными переходами и обменом данными между параллельными процессами. Вычисления на графических процессорах уже сегодня применяются в следующий областях [3,8]:

- Научное моделирование и высокопроизводительные вычисления
- Квантовая механика и химия;
- Астрономические и астрофизические расчеты;
- Биология и геномика;
- Кодирование видео;
- Визуализация (3D-графика, томография, нанотехнологии);
- Нейронные сети;
- Биология и геномика.

Наряду с симуляцией реальных процессов, GPU используются и для визуализации (отображение результатов томографии, представление сложных молекул и больших систем), матричных преобразований, преобразований Фурье [3].

### 1.4. История развития и примеры реализации

#### 1.4.1 История развития и примеры реализации GPU

На начальных этапах развития компьютерной графики за все вычислительные задачи, включая рендеринг графики, отвечал центральный процессор. Но по мере усложнения задач компьютерной графики CPU уже не мог справляться с растущими требованиями к качеству и производительности. Это послужило толчком для начала разработки специального оборудования и создания первых графических карт или видеокарт (плата с GPU и видеопамятью) [8].

Первые видеокарты, такие как IBM Monochrome Display Adapter (MDA) и Color Graphics Adapter (CGA) могли поддерживать только текстовый режим отображения или базовую цветную графику. Позже появились более совершенные графические карты, например, Video Graphics Array (VGA). Они обеспечивали более высокое разрешение и глубину цвета [8].

Термин «GPU» впервые ввела компания Nvidia в 1999 году с выпуском GeForce 256. Это была первая видеокарта, которую продавали как GPU, поскольку она интегрировала механизмы трансформации, освещения, настройки/склейки треугольников и рендеринга на одном чипе. В то время как предыдущие видеокарты для некоторых из этих задач использовали CPU [8].

С 2001 по 2006 годы графические процессоры включали в себя «вычислители» двух типов: вершинные и пиксельные конвейеры [3]. Первые были предназначены для проектирования на плоскость экрана вершин, задающих отображаемые поверхности, а вторые — для расчета цветов пикселей на экране.

В 2007 году производители GPU перешли от различающихся вершинных и пиксельных конвейеров к унифицированным потоковым процессорам, заменяющим как вершинные, так и пиксельные конвейеры [3]. Это позволило использовать GPU не только в задачах, связанных с компьютерной графикой, но и в любых других задачах, где требуются высокопроизводительные вычисления.

### 1.4.2 Интерфейсы программирования приложений

Для работы с GPU было разработано множество библиотек специализированных функций, также называемых программным интерфейсом приложения (API) для работы с GPU [3].

В качестве разработчиков API для графических процессоров выступают как сами производители GPU, так и компании, специализирующиеся на программном обеспечении. Под управлением Windows наиболее широко используется библиотека API DirectX от Microsoft, а для других операционных систем такие библиотеки разрабатываются в основном по спецификациям OpenCL и OpenGL [23].

Средства программирования GPU, описываемые выше, применимы для программирования графических процессоров всех производителей, в частности, компаний NVIDIA и AMD. Однако внутренние архитектуры GPU разных производителей и поколений существенно различаются между собой, а возможности использования особенностей каждой из архитектур универсальными системами программирования ограничены [3]. Преимущества конкретных GPU доступны при использовании специализированных инструментов программирования, одним из которых является NVIDIA CUDA [2,7], который и будет рассматриваться далее.

# 2. ПРОГРАММНАЯ МОДЕЛЬ CUDA

## 2.1. Описание программной модели CUDA

Платформа для параллельных вычислений NVIDIA CUDA (Compute Unified Device Architecture — архитектура единого вычислительного устройства) фактически представляет собой программно-аппаратный комплекс, в котором физическое устройство графического процессора и его программная модель соответствуют подходам к организации параллельных вычислений, разрабатываемым NVIDIA [2,3,7].

CUDA позволяет программировать только совместимые GPU (производства NVIDIA), однако позволяет управлять ими на более низком уровне, чем API DirectX и OpenCL. В частности, преимуществами CUDA являются [2,3,7]:

- Доступ к программируемой разделяемой памяти;
- Произвольная адресация при записи в память;
- Ускоренное взаимодействие CPU и GPU.

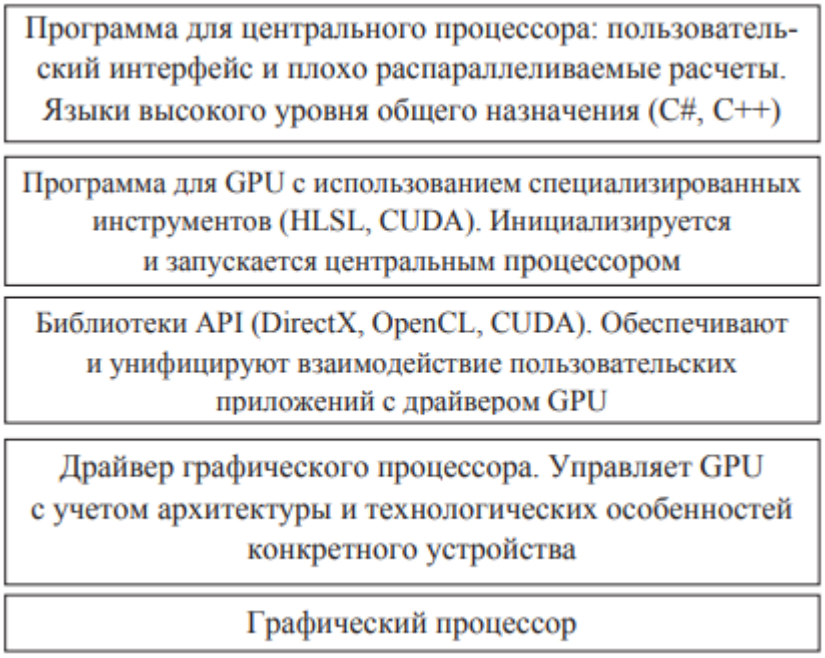


Рисунок 2.1 – Иерархия приложений, составляющих программный комплекс для расчетов на графических процессорах.

Общая схема, иллюстрирующая взаимодействие приложений в составе программного комплекса, для расчетов с использованием GPU приведена на рис. 2.1 [3].

Система программирования CUDA основана на расширении языка C путем добавления в него новых типов данных и процедур для управления графическим процессором. CUDA включает в себя возможности языков, предназначенных для написания вычислительного ядра (HLSL), и графических библиотек (DirectX, XNA, OpenCL). Практически программирование на CUDA сводится к использованию новых операций, процедур и типов данных в «обычных» программах на языках C, C++ и C#.

### 2.1.1. Концепция CUDA

Выпускаемые с 2007 года графические процессоры очередного поколения характеризуются многими нововведениями, как уже было отмечено ранее в пункте 1.4.1, важными для неграфического программирования, среди которых:

- Замена пиксельных и вершинных конвейеров (имевших уже практически одинаковые возможности) универсальными по- токовыми процессорами;
- Поддержка целочисленных типов данных и побитовые операции с целыми числами;
- Работа с вещественными числами двойной точности.

Дополнительные возможности появились в рамках концепции универсального вычислительного устройства компании NVIDIA – CUDA. На аппаратном уровне эта концепция выразилась в следующем [3]:

- Появилась возможность идентификации в ходе выполнения вычислительного ядра конкретных вычислительных процессов (threads), исполняемых данным потоковым процессором в данный момент;

- Стало реализуемым варьирование того, какие именно элементы потоков данных загружаются и обрабатываются конкретными потоковыми процессорами и конкретными вычислительными процессами;
- Первые две возможности, задействованные вместе, дают некоторую свободу ветвления алгоритмов, исполняемых различными вычислительными процессами;
- Вместо автоматической записи результатов расчета в цель рендера появилась возможность произвольной записи в видеопамять, что обеспечивает гибкое формирование массивов результатов;
- Потоковые процессоры, входящие в один и тот же вычислительный блок, получили доступ к общей разделяемой кеш-памяти, что позволяет вычислительным процессам обмениваться результатами работы на промежуточных этапах алгоритма.

Вычислительные блоки с общей разделяемой памятью названы компанией NVIDIA потоковыми мультипроцессорами. На рис. 2.2 показана структура такого мультипроцессора в составе графического процессора GTX 200 [3].

Каждый мультипроцессор состоит из 8 скалярных процессоров, каждому из которых доступны для чтения и записи, наряду с разделяемой памятью, еще и собственные регистры. В мультипроцессоре есть еще два типа кеш-памяти, доступной всем его скалярным процессорам, но только для чтения, это память для констант и память для текстур. Данные в память таких разновидностей загружает только CPU. Первая используется для хранения констант, одинаковых во всех вычислительных потоках, а вторая — для хранения текстур в графических приложениях. Всем мультипроцессорам целиком доступна общая память (видеопамять) большого объема [3].

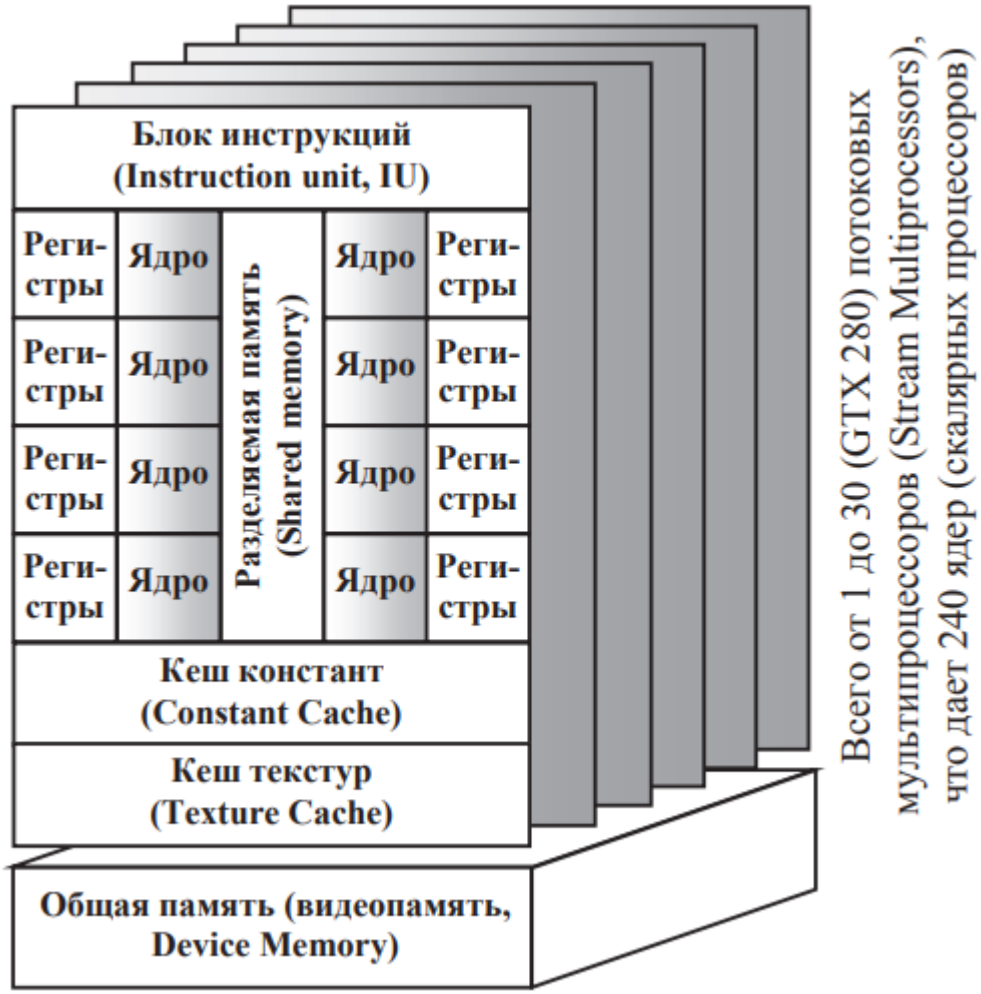


Рисунок 2.2 – Потоковые мультипроцессоры графических процессоров серии NVIDIA GTX 200.

Мультипроцессоры с разделяемой памятью явились новыми вычислительными единицами, модель программирования которых вышла за рамки шейдерных моделей в силу того, что потоковые процессоры в составе одного мультипроцессора могут совместно использовать разделяемую кеш-память [3].

### 2.1.2. Иерархия вычислительных процессов и памяти CUDA

Как и GPU предыдущих поколений, графические процессоры архитектуры CUDA представляют собой системы из параллельных «вычислителей» — потоковых процессоров (они же названы ядрами и скалярными процессорами), каждый из них применяет заданное

вычислительное ядро (программу) к некоторым элементам входных массивов данных. По иерархическому положению потоковые процессоры эквивалентны отдельным графическим конвейерам GPU предыдущих поколений, но могут быть идентифицированы и по-разному управляемы в ходе исполнения алгоритма [3].

Потоковые процессоры стали универсальными – они больше не подразделяются на вершинные и пиксельные конвейеры. Вместо этого потоковые процессоры физически и логически объединены в вычислительные блоки — мультипроцессоры, внутри которых они имеют общий доступ к разделяемой кеш-памяти, через которую могут обмениваться данными в ходе параллельных вычислений [3].

Все мультипроцессоры исполняют одно и то же вычислительное ядро, но могут использовать различные исходные данные, что в принципе позволяет реализовать параллелизм не только по данным, но и по задачам. Всем мультипроцессорам доступна общая память или видеопамять, в которой центральный процессор размещает исходные данные, а графический процессор — результаты расчетов, которые становятся доступными центральному процессору [3].

Каждому из мультипроцессоров доступен параллельный кеш данных (или разделяемая память Shared Memory на рис. 2.2), который работает со скоростью регистров процессора. Предназначена она для того, чтобы вычислительные процессы могли модифицировать общие данные и обмениваться информацией в ходе параллельного расчета [3].

Каждый блок разделяемой памяти доступен одновременно всем «вычислителям» в составе одного мультипроцессора (см. рис. 3.2) для чтения и для записи. Конструкцией GPU предусмотрена автоматическая синхронизация доступа «вычислителей» к разделяемой памяти. «Вычислители», принадлежащие к разным мультипроцессорам, общей кеш-памяти не имеют [3].

### 2.1.3. Конвейерная обработка данных в архитектуре CUDA

Потоковые процессоры имеют конвейерную архитектуру, то есть могут одновременно исполнять несколько вычислительных процессов, находящихся на разных стадиях алгоритма. Каждый мультипроцессор в составе GPU одновременно исполняет до 2048 вычислительных процессов в одной или двух «связках», где «связка» — логическое объединение вычислительных процессов с общей разделяемой памятью в программах на CUDA [3].

Применение «связок» (blocks) вычислительных потоков обусловлено тем, что в программах на CUDA должны быть логически разделены потоки, исполняемые на различных мультипроцессорах, поскольку только потоки, исполняемые на одном и том же мультипроцессоре, имеют общий доступ к разделяемой памяти [3].

Каждая «связка» исполняется на одном мультипроцессоре, а один процессор может исполнять две связки, что дополнительно сокращает время его простаивания при конвейерных вычислениях. В «связке» может быть до 1024 потоков. Оптимальное количество «связок» и вычислительных потоков должно быть таким, чтобы максимально задействовать ресурсы мультипроцессора (регистры и разделяемую память), но так, чтобы данные всех потоков размещались в регистрах и памяти одновременно. Предельно допустимое количество вычислительных потоков на один мультипроцессор в двух «связках» не более 2048 [3].

## 2.2. Программные средства для разработки на CUDA

Программные средства для разработки на CUDA предоставляют разработчикам инструменты для создания параллельных вычислений, использующих графические процессоры NVIDIA. Наиболее популярными из них являются следующие программные средства:

1. NVIDIA CUDA Toolkit. Это основной пакет разработки для программирования на CUDA. Он включает в себя компилятор,

библиотеки, отладчики и другие инструменты, необходимые для создания приложений, использующих вычисления на GPU;

2. NVIDIA Nsight Systems. Это инструмент профилирования и анализа производительности, позволяющий разработчикам получить глубокое понимание работы и производительности CUDA-приложений;
3. NVIDIA Nsight Visual Studio Edition. Это интегрированная среда разработки (IDE) в Visual Studio, предоставляющая возможности отладки, профилирования и анализа производительности для CUDA-приложений;
4. CUDA-GDB. Это отладчик, специально разработанный для разработки приложений на CUDA. Он предоставляет возможности отладки как на уровне CPU, так и на уровне GPU;
5. PGI CUDA Fortran. Это компилятор Fortran, который позволяет разработчикам использовать язык программирования Fortran для создания параллельных вычислений с использованием CUDA.

Эти программные средства предоставляют разработчикам большой выбор инструментов для создания высокопроизводительных приложений, использующих технологию CUDA для работы с графическими процессорами NVIDIA.

# 3. ПЕРСПЕКТИВЫ РАЗВИТИЯ

Перспективы развития параллельных вычислений на графических процессорах (GPU) остаются очень обнадеживающими. Можно выделить несколько ключевых направлений, которые могут формировать будущее параллельных вычислений на GPU [4,5]:

1. Исследования в области искусственного интеллекта и глубокого обучения. Параллельные вычисления на GPU имеют фундаментальное значение для развития искусственного интеллекта и глубокого обучения. С развитием более сложных нейронных сетей и моделей глубокого обучения, спрос на высокопроизводительные вычисления на GPU будет продолжать расти [4,5];
2. Обработка больших данных. С увеличением объема данных в мире появляется потребность в обработке этих данных. GPU обеспечивают параллельную обработку данных, что делает их идеальным выбором для анализа больших объемов информации;
3. Научные исследования. Многие научные исследования, такие как расчеты в области физики, химии и биологии, требуют высокопроизводительных вычислений. GPU предоставляют выгодное соотношение вычислительной мощности к стоимости, что делает их привлекательным выбором для ученых;
4. Развитие новых технологий. С появлением новых технологий, таких как виртуальная реальность, расширенная реальность и автономные автомобили, возникает потребность в мощных параллельных вычислениях. GPU могут обеспечить необходимую вычислительную мощность для реализации этих технологий;
5. Расширение возможностей облачных вычислений. Облачные вычисления продолжают развиваться, и GPU играют важную роль в обеспечении высокой производительности в облачных вычислениях. Перспективы развития параллельных вычислений на GPU связаны с

улучшением инфраструктуры и сервисов облачных вычислений для поддержки графических вычислений.

Эти направления указывают на постоянно растущий и важный характер развития параллельных вычислений на GPU в различных областях, что делает эту технологию одной из самых перспективных для будущего. Стоит также отметить, что производительность GPU растет и будет расти с каждым готом. Это также открывает для параллельных вычислений на GPU новые задачи и области.

# ЗАКЛЮЧЕНИЕ

В рамках данной работы была достигнута поставленная цель, а именно рассмотрено параллельное программирования на GPU с использование технологии CUDA. Были решены все поставленные задачи.

Была рассмотрена архитектура вычислительной системы с GPU, а также ее классификация среди параллельных вычислительных систем. Было выяснено, что параллельное программирования на GPU относится к классу SIMD в классификации параллельных архитектур по Флинну.

Были рассмотрены области применения параллельных вычислительных систем с GPU. Было выяснено, что параллельные вычисления на GPU используются во многих областях: от визуализации компьютерной графики до научного моделирования и расчетов.

Были рассмотрены история развития и примеры реализации вычислительных систем с GPU. Также были рассмотрены программные интерфейсы для программирования с использованием GPU. Было выяснено, что изначально GPU создавались для решения задач компьютерной графики, но с дальнейшим развитием множество решаемых задач значительно расширилось. Также было выяснено, что для работы GPU существуют множество программных интерфейсов, к которым относятся: DirectX, OpenCL, OpenGL и CUDA.

Была рассмотрена программная модели CUDA. Было выяснено, что данная модель позволила расширить класс решаемых при помощи GPU задач за счет более универсальных потоковых процессоров.

В завершение были рассмотрены перспективы развития параллельного программирования на GPU. Было выяснено, что у параллельного программирования на GPU множество перспектив с учетом дальнейшего развития GPU.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Акимова Е. Н. Параллельные вычисления. Екатеринбург: Изд-во Урал, 2023. 108 с.
2. Параллельные вычисления на GPU / А. В. Боресков, А. А. Харламов, Н. Д. Марковский и др. М.: Издательство Московского университет, 2015. 336 с.
3. Параллельные вычисления общего назначения на графических процессорах / К.А. Некрасов, С.И. Поташников, А.С. Боярченков, А.Я. Купряжкин. Екатеринбург: Изд-во Урал, 2016. 104 с.
4. GPUs and the Future of Parallel Computing / S. Keckler, W. Dally, B. Khailany, M. Garland и др. // Micro, IEEE. 2011, вып. (№) 31. С. 007-017.
5. GPU Computing / J. Owens, M. Houston, D. Luebke, S. Green и др. // Proceedings of the IEEE, IEEE. 2008, вып. (№) 96. С. 870-899.
6. Introduction – Parallel Programming // GitBook. URL: (дата обращения: 05.11.2023).
7. Understanding NVIDIA CUDA: The Basics of GPU Parallel Computing // TURING. URL: https://www.turing.com/kb/understanding-nvidia-cuda (дата обращения: 05.11.2023).
8. Все, что нужно знать про GPU: история технологии, архитектура графических процессоров и сферы их применения. URL: https://habr.com/ru/companies/itglobalcom/articles/746252 (дата обращения: 03.12.2023).