Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Направление: 09.04.04 – Программная инженерия

Профиль: Разработка программно-информационных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ НЕЙРОСИМУЛЯТОРА GRAS

Студент 2 курса

группы 11-821

«___»_____2020 г. Леухин А.Д.

Научный руководитель

канд. техн. наук, зав.лаб.

«___»_____2020 г. Таланов М.О.

Директор Высшей школы ИТИС КФУ

канд. техн. наук

«___»_____2020 г. Абрамский М.М.

Казань – 2020

ОГЛАВЛЕНИЕ

Список сокращений и условных обозначений 3
Введение 4
1. Проект Memristive Spinal Cord 6
2. Анализ нейросимуляторов NCS6, NeuroGPU и NEST 8
3. Реализация GRAS (GPU Reflex Arc Simulator) 10
   3.1. Сравнение стандартов и фреймворков параллелизации программного кода нейросимулятора 10
      3.1.1. CPU архитектура: OpenMP, MPI и OpenCL 10
      3.1.2. GPU архитектура: OpenCL, OpenACC и CUDA 13
      3.1.3. Структуры AoS, AoaS и SoA вычислительного принципа SIMT 16
   3.2. Соответствие нейросимулятором био-правдоподобности 18
      3.2.1. Разработка упрощенной модели нейрона ESRN для расчетов в реальном времени 18
      3.2.2. Создание модели синапса для передачи сигнала 25
   3.3. Архитектура GRAS 28
   3.4. Воссоздание работы центрального генератора упорядоченной активности отдела спинного мозга в вычислительной модели GRAS 33
   3.5. Ускорение работы нейросимулятора для видеокарты Nvidia K80 36
4. Результаты GRAS симуляции модели отдела спинного мозга крысы . 41
Заключение 44
Список литературы 45

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

1. ЦГУА (англ. CPG, central pattern generator) – центральный генератор упорядоченной активности. Сложная структура нейронов спинного мозга, которая генерирует активность моторного выхода в зависимости от кожных входов и растяжения мышц [2].
2. SIMT (англ. single instruction, multiple threads) – вычислительный подход, в котором одиночный поток команд применяется для множества потоков обработки данных. Это принцип вычислений, который обеспечивает параллелизм на уровне данных [59].
3. Варп (англ. warp) – минимальная единица вычисления в аппаратной архитектуре CUDA (содержащая 32 потока), которая обрабатывает данные SIMT-способом в мультипроцессорах CUDA [59].
4. Спайк (потенциал действия) – резкое увеличение мембранного потенциала нейрона, вызывающий каскад процессов внутри клетки и передающий импульс во внешнюю среду [23].
5. Нейротрансмиссия (нейропередача) – распространение нервных импульсов от одного нейрона к другому (различают химическую и электрическую передачу) [23].
6. STDP (англ. Spike timing dependent plasticity) – регуляция силы синапсов на основе их пластичности, зависимой от времени между приходящим спайком пресинаптического нейрона и постсинаптического [5].
7. Потенциал реверсии – напряжение мембраны нейрона, при котором нейротрансмиттер не вызывает поток ионов через ионный канал [20].
8. Мотонейрон – крупная, по сравнению с другими, нервная клетка спинного мозга. Нейрон имеет выход на мышечные волокна, обеспечивая их поддержание тонуса и локомоторную координацию [17].
9. Интернейрон – нейрон, меньшего размера по сравнению с мотонейроном, который соединен только с другими нейронами (к примеру без выхода на мышечные волокна) [17].

ВВЕДЕНИЕ

По данным Всемирной организации здравоохранения (ВОЗ) ежегодно в мире получают травму спинного мозга от 250 000 до 500 000 человек. Подобное повреждение в разной степени приводит к общему ухудшению здоровья и даже к угрозе жизни (смертность с травмой выше в 2–5 раз, чем у не травмированных людей) [73]. В крупных городах России (на 2011 год) частота получения травмы спинного мозга примерно составляет 0.58–0.6 случаев на 10 000 человек [75], в то время как в США (на 2012 год) частота случаев приблизительно равна 0.54 на 10 000 человек [55].

Восстановление способности к самостоятельной ходьбе является одной из приоритетных задач реабилитации пациентов со спинальными травмами. Существуют два способа восстановления: (1) полная операция на спинной мозг с трансплантацией нервных клеток с дальнейшей послеоперационной физиотерапией в клинике, и (2) мало-инвазивные операции или без-операционные вмешательства с упором на физиотерапии с системами для восстановления ходьбы [22]. Процесс восстановления двигательной активности дорогостоящий, занимает много времени (от одного года), требует постоянных тренировок и обследования специалистов [47].

Исследования и работы в области травм спинного мозга очень актуальны. Американский медицинский центр Mayo Clinic (Рочестер, штат Миннесота) активно проводит испытания по компенсации двигательных нарушений пациентов при травме спинного мозга [22]. Их исследования [19; 37; 39] базируются на постоянной электростимуляции отдела спинного мозга без изменения паттерна сигнала. В данном подходе отсутствует адаптация сигнала в зависимости от кожного входа стопы, что не позволяет пациенту носить стимулятор в повседневной жизни, что сэкономило бы денежные средства пациента при нахождении в стационаре или реабилитационном центре.

В связи с этим, возникает проблема отсутствия протезного устройства, которое может в режиме реального времени адаптироваться под шаг пациента и заменять работу отдела спинного мозга для восстановления двигательной активности (ходьбы) после травмы без круглосуточного надзора специалистов.

Цель – создать нейросимулятор, который в режиме реального времени может обрабатывать кожные входы, перестраивать паттерн шага и выводить результат работы на внешний стимулятор, чтобы заменить работу отдела спинного мозга, отвечающего за ходьбу.

Для достижения цели были поставлены следующие задачи:

- сравнить существующие нейросимуляторы, найти их плюсы и минусы, выделить особенности для создания нового инструмента;
- подобрать оптимальный инструмент распараллеливания и тип структур данных для SIMT (single instruction, multiple threads) подхода;
- написать код нейросимулятора и модели центрального генератора упорядоченной активности (ЦГУА) на языке C++;
- провести эксперимент по работе нейросимулятора в аппаратном комплексе с цифро-аналоговым преобразователем для будущей стимуляции нервов сгибательной мышцы (руки или ноги) человека.

Объект исследования: био-совместимая симуляция функционального отдела мозга, состоящего из рефлекторной дуги и ЦГУА, способная заменить его поврежденную активность (нейропротезирование).

Предмет исследования: разработка био-правдоподобного нейросимулятора, работающего в режиме реального времени, для симуляции центрального генератора упорядоченной активности.

ГЛАВА 1. ПРОЕКТ MEMRISTIVE SPINAL CORD

Проект мемристивного спинного мозга (memristive spinal cord) лаборатории нейроморфных вычислений и нейросимуляций КФУ направлен на решение задачи моделирования нейрональных контуров спинного мозга, отвечающих за локомоцию, и создание протеза функционального отдела спинного мозга на основе мемристорной технологии (резисторы с памятью).

Первый этап проекта направлен на моделирование функционального отдела спинного мозга на основе ранее проведенных исследований и экспериментальных данных [19; 37; 39] и описанных решений модели нейрональных цепей спинного мозга. На основе анализа была предложена двухуровневая архитектура: первичный уровень с рефлекторной организацией с участием мотонейронов и первичных интернейронов и центрального генератора упорядоченной активности (ЦГУА) описанных в параграфе 3.4.

Моделирование данной архитектуры позволит создать протез, контролирующий движение человека с парализованными конечностями. Это является следующим шагом проекта. Идея по воссозданию активности базовых локомоторных функций с применением естественной топологии нейронных сетей (выведенной на основе миограмм мышечной активности крысы), с биоправдоподобным поведением нейронов и синапсов является оригинальной и новой. Уже сейчас существуют научные работы по аналоговой реализации отдельных частей нервной системы млекопитающего с низкой биореалистичностью и биосовместимостью [6; 42]. В 2018 году учеными Стэнфордского университета было создано небольшое протезное устройство для таракана [36] с биологической совместимостью, что может указывать на более простой подход в реализации аналогового направления протезирования.

На сегодняшний день нейроморфными вычислениями в мире занимаются несколько крупных проектов:

- проект Human Brain Project (humanbrainproject.eu), который занимается симуляцией человеческого мозга на базе больших вычислительных кластеров с использованием биологических данных. Деятельность в этом направлении предполагает создание новейших основ для изучения и по-

нимания работы мозга млекопитающего, создания современных компьютерных технологий, а также выяснения источников нейродегенеративных заболеваний. Для моделирования таких объемов требуется суперкомпьютер, который должен быть во много раз более мощным, чем есть сегодня. Так, научное сообщество подходит к пониманию необходимости создания нейроморфных процессоров (NPU).

- Европейский проект Blue Brain Project вовлечен в создание биологически правдоподобных и сверх детализированных нейросимуляций мозга крыс/мышей с постепенным переходом на более сложный человеческий мозг (epfl.ch/research/domains/bluebrain/). Специально для таких вычислений был спроектирован суперкомпьютер Blue Gene.
- Американский проект SyNAPSE (Systems of Neuromorphic Adaptive Plastic Scalable Electronics initiative) финансируемый DARPA по программе создания технологий электронных нейроморфных процессоров. Основными исполнителями проекта являются подразделение IBM Research и исследовательский центр HRL Laboratories. Конечной целью проекта является создание новой электронной микропроцессорной системы, соответствующей по размеру, потребляемой мощности и функциям мозгу небольшим млекопитающим (крыса/кошка).

Результатом проекта Memristive spinal cord лаборатории является рабочий и носимый парализованным пациентом протез, выполняющий роль генератора паттерна ходьбы. Для достижения этого результата, в рамках дипломной работы, необходимо создать нейросимулятор с упрощенной цифровой моделью нейрона для моделирования работы упрощенной био-совместимой нейронной архитектуры отдела спинного мозга, которая должна состоять из тысяч нейронов и сотни тысяч синапсов. Требуется обработка данных в реальном времени, так как время расчета является критическим для био-совместимых носимых устройств. Для модели нейрона выдвигаются следующие требования: обработка в реальном времени на одноплатном компьютере с учетом биологически достоверных форм спайка, рефрактерного периода (периода восстановления мембранного потенциала), ингибирующего синаптического воздействия на сому и инициация спайка при достижении пред-порогового состояния (threshold).

ГЛАВА 2. АНАЛИЗ НЕЙРОСИМУЛЯТОРОВ NCS6, NEUROGPU И NEST

Основными характеристиками при выборе нейросимулятора для научных разработок являются: время вычисления, занимаемое пространство ОЗУ объектами симуляции, возможность распределенных вычислений на кластере и возможность симулировать топологии нейронный сетей в режиме реального времени (принимая во внимание количество нейронов и синапсов). Руководителем лаборатории нейроморфных вычислений и нейросимуляций был проведен анализ возможностей нейросимуляторов, которые поддерживаются сообществом и пользуются популярностью. Были выставлены оценки по наличию нейромодулирующих систем, интегрированных моделей нейронов, детализированности. На основе оценок были взяты для сравнения: NCS, NEST, NEURON [67].

Большинство нейросимуляторов используют либо ресурсы центрального процессора (с помощью интерфейсов MPI и OpenMP), либо ресурсы GPU для лучшей производительности. Все симуляторы, без исключений, демонстрируют ограничения по времени вычислений, когда используются биореалистичные модели [32]. GPGPU (универсальные вычисления на графических процессорах) быстро завоевывает популярность в мире благодаря растущему числу вычислительных ядер: сотни ядер в видеокартах ноутбуков и тысячи в более мощных моделях [60]. Проведенные исследования [4; 26] описали потенциал применения подхода GPU для нейросимуляторов, которые нацелены на расчет крупномасштабных моделей (от 100 000 до 1 000 000 нейронов) с использованием нескольких видеокарт (через комбинацию интерфейсов MPI + CUDA), что значительно сокращает общее время вычислений. Одним из таких известных и поддерживаемых нейросимуляторов является NCS6 (NeoCortical Simulator version 6) [26]. Однако моделирование топологии отдела спинного мозга, который управляет сложной координацией двигательных паттернов вместе с интеграцией сенсорной информации, поступающей с кожи, может быть представлено с меньшей популяцией нервных клеток. Кроме того, подобное моделирование требует более мелкого шага симуляции (в NCS6 это 1 мс, когда необходимо вычислять с шагом 0.025–0.1 мс) для увеличения вариабельности спайковой активности.

Вторым нейросимулятором с наибольшей детализированностью является NeuroGPU, написанный на базе существующего симулятора NEURON (Европейского проекта Blue Brain) [4; 15]. Он имитирует работу нервных клеток с помощью био-правдоподобных моделей, имеющих пространственную форму тела нейронов, аксонов и дендритов. NeuroGPU имеет отличительную от NCS6 масштабируемость: построение небольших топологий до 50 000 нейронов с высокой точностью биологических процессов. Высокая степень детализации требует большего времени для моделирования (к примеру, для расчета 1 секунды модели ЦГУА NEURON требуется 14 минут, когда как NeuroGPU в 7.7 раз меньше – 108 секунд при использовании NVIDIA® TITAN Xp) [4], из-за чего NeuroGPU не может быть использован в проекте лаборатории.

Балансом между описанными выше нейросимуляторами является NEST (NEural Simulation Tool). Данный симулятор, написанный командой разработчиков из Blue Brain Project [15], используется для построения топологий “точечных” нейронов, то есть моделей нейронов, которые не имеют морфологии дендритов, аксонов и сомы. Стоит отметить, что доступна возможность запуска только на CPU ядрах, что увеличивает время вычислений, которое может нивелироваться упрощёнными моделями нейронов. Изучив код нейросимулятора и написав свой дополнительный функционал внутри него, выяснилось, что архитектура NEST сильно перегружена и имеет очень большой порог вхождения для новых разработчиков [16].

На базе вычислительного узла КФУ (Intel® Xeon® CPU E5-2640 v3) проводилась оценка эффективности NEST и NEURON (CPU версия NeuroGPU). Была просимулирована одинаковая топология ЦГУА на базе модели нейрона Ходжкина–Хаксли [32] длительностью 1 000 мс, NEURON потребовалось 14 минут с использованием 50 процессов MPI, в то время как NEST справился за 16 секунд с использованием 50 процессов OpenMP. Таким образом, стало очевидно, что снижение в разумных пределах био-реалистичности модели с отсечением перегруженного функционала и классов способствует уменьшению времени вычислений и нагрузки на центральный процессор. В рамках проекта memristive spinal cord все три известных, развиваемых и поддерживаемых нейросимулятора не удовлетворяют основным требованиям: работы в режиме реального времени и возможности запуска на одноплатных системах.

ГЛАВА 3. РЕАЛИЗАЦИЯ GRAS (GPU REFLEX ARC SIMULATOR)

3.1. Сравнение стандартов и фреймворков параллелизации программного кода нейросимулятора

3.1.1. CPU архитектура: OpenMP, MPI и OpenCL

При любом распараллеливании (на CPU или GPU) необходимо учитывать в программном коде не только формальный параллелизм (структуру алгоритма и хранения данных), но и то, что операции обмена данными между процессами происходят, как правило, значительно медленнее арифметических операций [29]. С этим связано существование больших накладных расходов на организацию параллелизма. Можно выделить следующие задачи в параллельном программировании: балансировка нагрузки, сокращение времени выполнения и простоя потоков в ожидании других процессов [1]. Перед написанием кода GRAS стояла задача выбора оптимального инструмента для расчета моделей в режиме реального времени.

OpenMP – популярный инструмент написания параллельных программ для архитектуры процессоров с общей памятью. Он включает в себя набор директив компилятора и встроенных библиотечных функций [8]. Для кода, написанного с использованием OpenMP, требуется многоядерный процессор, поэтому их количество, базовая частота и объем памяти являются важными факторами, которые влияют на эффективность ускорения программы [3; 74].

Достижение высокой производительности и скорости работы программы обуславливается тщательной оптимизацией кода, которая может быть достигнута следующими путями:

1. автоматическим распараллеливанием с помощью директив компилятора;
2. профилированием выявить участки кода, которые сильнее всего подвержены нагрузке и требуют большее количество процессорного времени;
3. выполнить проверку времени на создание параллельных процессов и их доступа к оперативной памяти;
4. сбалансировать нагрузку между потоками и выявить ненужную синхронизацию между ними [3].

Таким образом, в целях увеличения быстродействия программы требуется эффективно использовать локальную память, кэш и минимизировать обращения к удаленной памяти [3; 7].

Основными плюсами OpenMP являются простота в изучении, разработке и отладке программы, а так же то, что изначальная архитектура программы не подстраивается специально под параллелизацию, а остается прежней. Из минусов можно выделить следующее: код OpenMP запускается только под архитектуры с общей (shared) памятью, то есть нельзя распределить вычисления программы на другие узлы [3]. В рамках проекта, где планируется выбор одноплатных компьютеров для протезного устройства, данное замечание не является недостатком, так как в данный момент отсутствует идея комбинации одноплатных компьютеров в мини-кластер.

Более известным инструментом для распределенных и параллельных вычислений является MPI (Message Passing Interface) [3]. Основные достоинства MPI это: возможность запуска одной и той же программы как для распределенных вычислений на разных узлах, так и на одном с общей разделяемой памятью [3; 74]. MPI позволяет разработчику более тонко настраивать передачу данных и их обработку в каждом процессе, который имеет собственный локальный экземпляр переменной (несомненный плюс в отсутствии блокировки доступа другим потоком) [3]. Сложность в отладке и ограничение скорости передачи данных между узлами в сети являются основными минусами данного инструмента. Кроме того, архитектура программы сильно видоизменяется при использовании конструкций MPI.

Идея разработки облегченного нейросимулятора GRAS подразумевает использование одного вычислительного устройства, что избавит пользователя от настройки передачи данных между вычислительными узлами. Таким образом MPI как инструмент параллелизации отпадает.

Современные процессоры Intel поддерживают SIMD-параллелизм на относительно небольших векторах данных [11]. Данный подход в распараллеливании активно используется в OpenCL (Open Computing Language) – с открытым стандартом для программирования на гетерогенных платформах, таких как процессоры, графические процессоры, FPGA и процессоры других типов [46]. Другими словами, стандарт OpenCL решил проблему портирования кода – про-

Тест	Вычислительный узел Broomway CPU (2xIntel Xeon E5-2680) Пиковая достижимая производительность 345.6 ГФлопс, Пиковая достижимая пропускная способность 102.4 ГБ/с					
	32 OpenMP		32 MPI		OpenCL	
	ГФлопс	ГБайт/с	ГФлопс	ГБайт/с	ГФлопс	ГБайт/с
viscosity	81.79	23.72	86.48	20.14	81.86	23.66
accelerate	25.67	43.45	30.42	50.81	21.45	36.17
pdv	48.92	53.55	42.46	47.28	38.59	42.1
ideal_gas	43.92	50.19	57.79	63.29	25.52	29.06
flux_calc	9.44	50.33	12.19	64.27	5.18	27.52
advec_mom	17.04	46.69	21.39	63.79	10.44	28.51
advec_cell	25.7	44.02	30.04	54.88	18.52	31.61

Таблица 1. Сравнение производительности разных стандартов на базе различных бенчмарк-тестов. Пиковая достижимая производительность вычисляется на основе числа ядер × средняя частота × количество операций за такт для процессоров Intel Xeon E5-2680. Пиковый показатель пропускной способности вычисляется по формуле: частота передачи x разрядность шины x количество каналов памяти ОЗУ [53]. Чем выше значение, тем насыщеннее цвет ячейки.

граммы, написанные на нем, будут корректно работать на любом совместимом с OpenCL устройстве: от многоядерных процессоров с наборами команд SIMD до многоядерных графических процессоров, но вместе с этим отсутствует гарантия переносимости производительности [50].

В Таблице 1 видно, как OpenCL на различных тестах производительности на базе CPU в основном проигрывает двум другим стандартам. Так же его пропускная способность заметно ниже. Это может быть обусловлено тем, что тесты не были написаны под текущую модель Xeon, так как драйвера OpenCL играют решающую роль в том, насколько производительным код окажется для той или иной вычислительной модели. Для платформы CPU явным лидером в простоте работы с кодом, производительности и пропускной способности является OpenMP. При необходимости переноса кода нейросимулятора на CPU будет использована именно эта архитектура. Однако имеется другая, более мощная архитектура для SIMT вычислений данных – графические процессоры (GPU).

3.1.2. GPU архитектура: OpenCL, OpenACC и CUDA

Использование графических процессоров в высокопроизводительных вычислениях (GPGPU) стало популярным, начиная с 2010-х годов, и эта популярность растет [11; 13; 65]. Популярность обусловлена высокой пропускной способностью памяти и высокой вычислительной мощностью видеокарт в совокупности доступности использования языков программирования высокого уровня. Существуют нейросимуляторы, которые написаны на OpenMP, MPI, OpenCL, GLSL, CUDA, OpenACC и т.д., все они имеют свои особенности, ограничения, плюсы и минусы при различных типах задач. Одной из задач дипломной работы является выяснение лучшей для проекта лаборатории мемристивного отдела спинного мозга технологии. Описанные подходы в параграфе 3.1.1 и их тесты показали, что центрального процессора уже недостаточно для обработки данных в режиме реального времени.

Одно из архитектурных отличий CPU от GPU заключается в том, что центральный процессор исполняет 1–2 потока (при поддержке гиперпоточности) вычислений на одно ядро (их может быть десятки), а видеокарты поддерживают от 1 024 до 2 048 потоков на каждый мультипроцессор (MP), которых в зависимости от модели [60] в видеочипе может быть несколько штук (физических ядер может быть более 4 000) (Рисунок 1). Возможных запущенных потоков обычно больше, чем физических ядер на видеокарте, благодаря чему планировщиками задач (warp schedulers) выполняется быстрое переключение между потоками для сокрытия задержки чтения/записи ядрами [59]. Это означает, что высокопроизводительные системы на одном GPU вычислят задачу быстрее (при правильном выборе структур данных), чем сотня ядер на OpenMP/MPI [53].

OpenCL, описанный в параграфе 3.1.1, теперь сравнивается на базе архитектуры видеокарт. Здесь он показывает лучшие характеристики наряду с OpenACC и CUDA (Таблица 2), нежели при CPU подходе (Таблица 1).

Написание кода на OpenCL и CUDA выполняется одинаково [3; 14] с использованием следующих шагов: (1) инициализируется GPU с использованием встроенных функций API; (2) выделяется необходимый объем видеопамяти; (3) входные данные переносятся в память устройства; (4) графический процессор выполняет функцию ядра; (5) переносится результат из видеопамяти в оперативную [14; 65]. Здесь можно выделить следующие плюсы данного подхода:

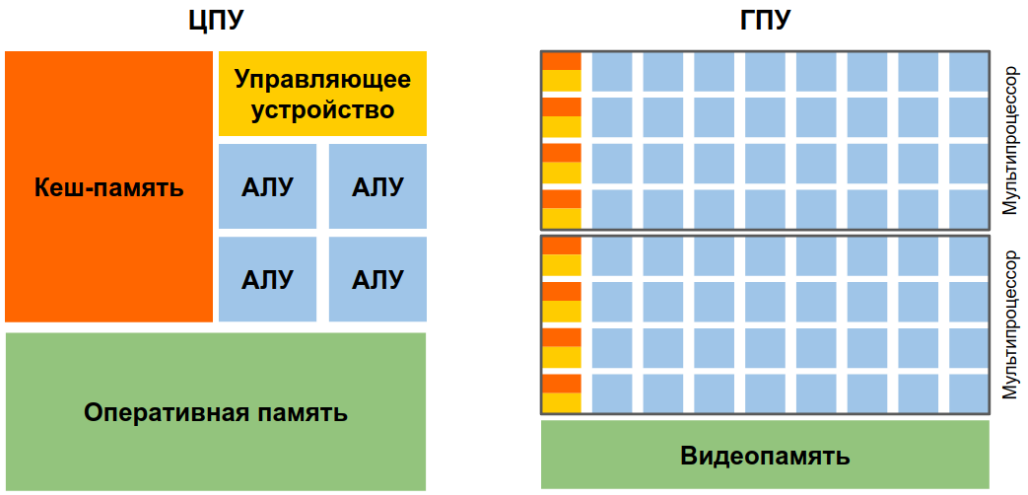

Рисунок 1. Упрощенное изображение архитектур CPU и GPU. Архитектура видеокарт спроектирована для высокой пропускной способности, предназначенной для одновременного выполнения множества задач (значения указаны в Таблице 1 и 2). На плате размещены сотни небольших слабых по тактовой частоте ядер (синего цвета), когда как места для устройств управления (желтый цвет) и кеш-памяти (оранжевый цвет) выделяется мало [56; 59]. CPU архитектура разработана для получения малой задержки исполнения команд для быстрого одновременного выполнения одной операции. Здесь наоборот выделяется много места для кеш-памяти и управляющего устройства, когда как количество физических ядер с высокой тактовой частотой на один процессор может составляет до 72 штук (Intel® Xeon® Phi).

программисту даётся свобода действий в тонкой настройке вычислительного процесса [11; 70], явно указаны действия для видеокарты. Из минусов: высокий порог вхождения для начинающих (требуется хорошая теоретическая база, чтобы обучиться правильной работе с API [59]), для каждой видеокарты нужно подбирать свои комбинации размера блока и их количества (параграф 3.5), а в случае использования CUDA подходят только видеокарты от NVidia [57; 59].

Программный стандарт OpenACC в отличие от OpenCL/CUDA работает схожим образом с OpenMP [46]. Появление OpenACC обусловлено желанием разрабатывать программы, используя знакомую модель распараллеливания OpenMP (который появился в 1997 году [8]), обеспечивающую высокую производительность и портируемость для различного спектра вычислительных архитектур (имеющих компилятор, который поддерживает OpenACC) [61].

Одним из важнейших преимуществ OpenACC (как и OpenMP) является то, что разработчику не нужно полностью переписывать существующий код под API инструмента, чтобы перенести вычисления на видеокарту [70; 74]. Чаще всего программы для научных вычислений громоздки [16; 52; 63] и переписывание всей архитектуры требует больших трудозатрат. Использование специ-

Тест	K20 GPU (NVIDIA K20c) Пиковая достижимая производительность 1.17 ТФлопс, Пиковая достижимая пропускная способность 208 ГБ/с					
	CUDA		OpenCL		OpenACC	
	ГФлопс	ГБайт/с	ГФлопс	ГБайт/с	ГФлопс	ГБайт/с
viscosity	248.55	72.08	250.26	72.57	176.35	51.16
accelerate	90.84	153.75	69.37	117.41	39.36	66.64
pdv	126.78	138.8	122.2	133.78	131.52	103.08
ideal_gas	127.43	145.64	125.71	143.67	52.85	122.82
flux_calc	29.11	155.27	23.66	126.18	25.1	133.93
advec_mom	47.08	129.01	42.52	116.54	39.43	108.1
advec_cell	76.09	130.33	74.53	127.66	64.7	110.85

Таблица 2. Сравнение производительности разных стандартов на базе различных бенчмарк-тестов. Пиковая достижимая производительность вычисляется на основе числа ядер × средняя частота × количество операций за такт для видеокарты NVIDIA K20c [53]. Пиковый показатель пропускной способности предоставляется производителем видеокарты [60]. Чем выше значение, тем насыщеннее цвет ячейки.

альных команд OpenACC (директив) над участками кода, которые нужно распараллелить, упрощает процесс разработки и не требует такого высокого порога вхождения как в CUDA или OpenCL. Вторым преимуществом является компилирование программы с директивами или без них. Другими словами программа может по-прежнему выполняться последовательно и даст такой же результат. Главным недостатком OpenACC является в основном низкая производительность работы программы по сравнению с CUDA или OpenCL (Таблица 2).

Имея навыки и знания в работе с CUDA, а также в связи с хорошими показателями ее производительности при разных тестах (Таблица 2), была выбрана именно эта технология. Стоит отметить, что умелая комбинация CUDA и OpenMP может дать дополнительный прирост к производительности по сравнению с их отдельным друг от друга использованием [34]. В дипломной работе на начальном этапе разработки нейросимулятора предполагается использование одного фреймворка для распараллеливания с дальнейшим расширением функционала для повышения био-реалистичности с сохранением высокой скорости вычислений на однобортных устройствах (личные ноутбуки, компьютеры, одноплатные системы).

3.1.3. Структуры AoS, AoaS и SoA вычислительного принципа SIMT

Модель доступа к памяти имеет большое влияние на производительность программы и должна обязательно учитываться при использовании параллелизма. AoS (англ. Array of Structs, массив структур), AoaS (англ. Array of aligned Structs, массив выровненных структур) и SoA (англ. Struct of Arrays, структура массивов) описывают различные способы расположения данных в оперативной памяти и являются важными элементами вычислительного принципа параллелизма SIMT (Single Instruction, Multiple Threads), в котором одна команда применяется к большому числу потоков обработки данных.

В нейросимуляторах обновление состояний всех нейронов и синапсов выполняется каждую итерацию. В это время различными потоками центрального процессора происходит одновременный доступ к ячейкам памяти различных нейронов/синапсов, поэтому логично предположить, что оптимальным и производительным для процессоров с большим кэшем, является использование AoS в качестве хранения данных [10]. Графические процессоры наоборот имеют небольшой по размеру кэш, что снижает прирост к производительности при обработке данных с помощью AoS, которое требует больше кэша для обработки одной транзакции [66]. Реализация и работа с AoS проще для программистов, где нейрон представлен как объект (в объектно-ориентированном подходе), но AoS не очень эффективен при использовании SIMT принципа [59].

Возможным решением к ускорению AoS является использование AoaS, которое подразумевает выравнивание структур по 4, 8 или 16 байт [59]. Следует отметить, что в GRAS размер структур данных нейрона и синапса превышают 16 байт, что означает, что данные не могут быть эффективно прочитаны или записаны с использованием выравнивания (Рисунок 2). Подход хранения данных в виде структуры массивов может дать лучшую производительность.

Структура массивов (SoA) потенциально может быть более эффективной для SIMT подхода благодаря группировке одинаковых типов данных. Кроме того, SoA всегда объединяет доступ к глобальной (global) памяти [52]. Использование общей (shared) памяти вместо глобальной нецелесообразно, так как вызов ядра происходит многократно, и расходы на постоянный перенос памяти перекроют быстродействие общей памяти. Результаты проведенных тестов на

вычислительном узле КФУ показывают, что структура массивов (SoA) в GRAS дает лучшую производительность, чем AoS (Рисунок 2).

Неструктурированные данные (когда параметры нейрона или синапса передаются в виде отдельных массивов без каких-либо структур), как и SoA, имеют лучшие результаты по времени. Бесструктурный подход быстрее SoA на 6% (Рисунок 2). Удобство использования подхода SoA растет с увеличением количества переменных – каждая передача переменной в ядро функции GPU делает код негибким и трудно читаемым.

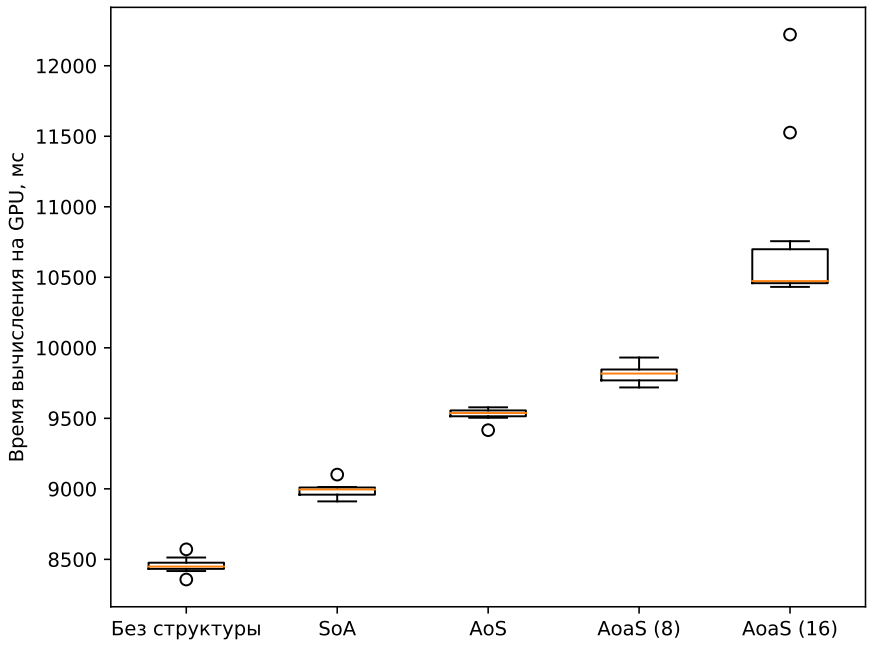

Рисунок 2. Результаты тестов различных реализаций структур данных одной и той же симуляционной модели на базе модели нейрона Ходжкина–Хаксли в среде GRAS. Симулировалось 1 100 мс активности 3 500 нейронов и 600 000 синапсов. Использовались 15-й и 85-й процентиль для “биржевой” диаграммы. Ось Y показывает общее время симуляции в миллисекундах. AoS – Array of Structs, AoaS – Array of aligned, Structs SoA – Structure of Arrays. Тест проведен на базе Intel® Xeon® CPU E5-2640 v3 (2.6 GHz) и видеокарты NVIDIA® Tesla® K80 (2 x 2 496 CUDA ядер с частотой 562 MHz, CUDA v7.5) вычислительного узла кластера КФУ.

Для симуляции в реальном времени ЦГУА и рефлекторной дуги оптимальным будет использование бесструктурного подхода и упрощенной модели нейрона. При расширении функционала GRAS и отсутствии жёстких требований по скорости работы можно перейти к структуре массивов (SoA) и модели нейрона Ходжкина – Хаксли.

3.2. Соответствие нейросимулятором био-правдоподобности

3.2.1. Разработка упрощенной модели нейрона ESRN для расчетов в реальном времени

На данный момент в научном сообществе уже создано большое количество моделей нейронов, и их число растет. Первые модели появились в начале 50-х годов благодаря работам Маккаллока и Питтса [48], Розенблата [64] и Ходжкина-Хаксли [27]. Существует база данных под названием “modelDB” Йельского университета, где в настоящее время опубликовано несколько тысяч моделей [49]. Одной из критических проблем в области вычислительной нейробиологии является время вычислений, это особенно актуально с учетом того, что модели синапсов и нейронов оказывают основную нагрузку на ЦП.

Особенно сильные требования к скорости предъявляются автономным роботам [44; 45] или носимым нейропротезам [12; 21; 72], где специалисты сталкиваются с задачей обработки данных или имитации нейрональных структур, состоящих из сотен тысяч синапсов, в режиме реального времени на одноплатных компьютерах с ограниченным количеством вычислительных ядер (не более 6 для самого мощного одноплатного компьютера RockPro64 [35]).

В рамках дипломной работы в нейросимуляторе GRAS были созданы три модели нейрона для гибкости работы при разных условиях использования (повышенная био-реалистичность или производительность):

1. модель нейрона Ходжкина–Хаксли [27] (или HH модель) является наиболее био-правдоподобной и требует большого количества вычислительной мощности ЦП (Рисунок 3);
2. модель Ижикевича [32] (или IZH модель), является балансом между скоростью и био-реалистичностью (Рисунок 3);
3. модель упрощенного нейрона для расчета в реальном времени (или ESRN модель [40]) имитирует работу минимального набора клеточных механизмов, что позволяет рассчитать работу нейрональных структур свыше нескольких тысяч нейронов в режиме реального времени (Рисунок 3). Что позволит использовать её при симуляции в робототехнических системах или протезных устройствах.

Упрощенная модель нейрона, должна соответствовать следующим требованиям: обработке данных в реальном времени на одноплатном компьюте-

ре с учетом био-правдоподобной формы спайковой активности нейрона (Рисунок 4a), периода восстановления мембранного потенциала (refractory period, Рисунок 5a, 3–15 мс), ингибирующего синаптического воздействия на сому и инициации спайка при достижении порогового состояния (threshold).

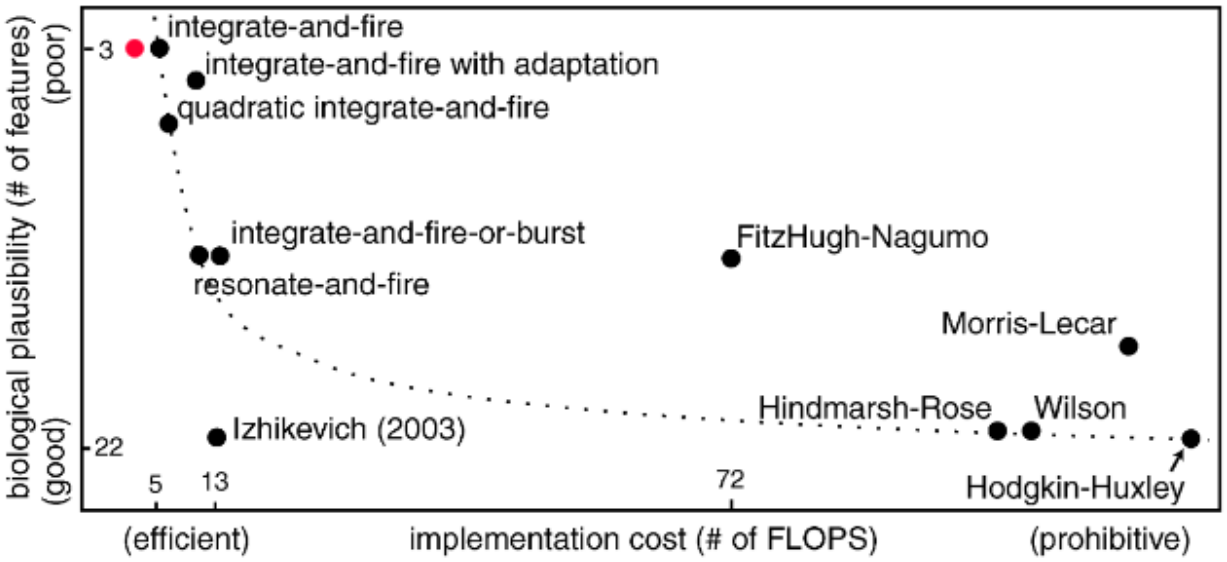

Рисунок 3. Ожидаемое расположение ESRN (красный цвет) на координатной плоскости, предложенной Ижикевичем [32] для сравнения моделей на основе био-правдоподобности (ось Y) и требуемых Флопс (единица измерения вычислительной мощности компьютеров) (ось X).

Для создания новой модели использовался обширный обзор 11 моделей, проделанный Ижикевичем (Рисунок 3) [32]. Были взяты две наиболее близко стоящие модели нейронов LIF [69] и IZH [33] для сравнения производительности с моделью нейрона ESRN. Эти модели довольно близки, с точки зрения производительности (5–13 Флопс). Новая модель ESRN разрабатывалась, чтобы ещё сильнее уменьшить время вычислений (затрачивая меньше 5 Флопс).

Модель IZH является упрощением модели HH [27] и не имитирует работу ионных каналов (Рисунок 4), рассчитывая вероятность их активации/инактивации. Спайк нейрона вызывается превышением порогового состояния мембранного потенциала (Рисунок 4, 1 мс). Модель может работать в разных режимах в зависимости от типа нейрона и заданных параметров поведения (bursting, tonic, phasic, mixed).

Модель LIF использует входной синаптический ток для обновления мембранного потенциала, по аналогии с зарядкой конденсатора. LIF считается более простой, чем IZH [69]. Параметры модели LIF, такие как скорость появле-

ния спайка или период восстановления мембранного потенциала (Рисунок 5 а2), соответствуют параметрам биологического нейрона, что делает её более реалистичной, чем простая модель Розенблата.

Перед созданием модели ESRN рассматривались разумные ограничения расчетов, таких как урезание амплитуды после порогового состояния (Рисунок 4а, 1мс), так как факта появления спайка достаточно, чтобы передать сигнал по синапсу другим нейронам без расчета фазы реполяризации, то есть снижения мембранного потенциала после спайка (Рисунок 4а) и активации калиевых каналов в модели HH (Рисунок 4б). Спайк ESRN является единичным импульсом, который умножается на синаптический вес, чтобы сэкономить время вычисления, когда как в других более реалистичных моделях затрачиваются ресурсы на формирование более био-правдоподобного сигнала.

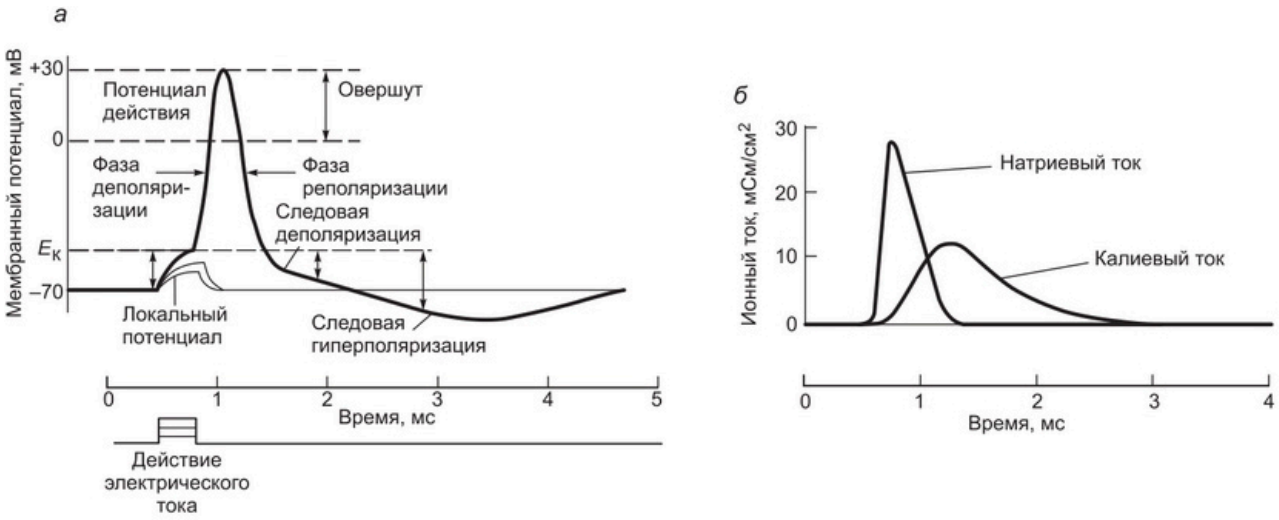

Рисунок 4. Потенциал действия (схематическое изображение) пирамидального нейрона гиппокампа крысы. Стоит учитывать тот факт, что форма реального потенциала действия при экспериментах обычно отличается от схематично-идеализированной [23].

Для расчета в реальном времени использовалось упрощенное уравнение (Уравнение 1) для формирования потенциала, который внешне напоминает мембранный потенциал реального нейрона (Рисунок 4а), где этот уровень представляет собой сумму синаптических весов, утечки тока (плюс при восстановлении мембраны и минус при деполяризации на Рисунке 5 а3) и генератор шума (спонтанная активность и влияние внешней среды на нейрон).

Когда потенциал превышает пороговое значение (threshold), модель выдает сигнал, который представлен единичным спайковым импульсом.

$$L = \Sigma W \pm leakage + noise \quad (1)$$

Так же были реализованы био-правдоподобные: период восстановления мембранного потенциала и длительность спайка. Форма спайка схожа с LIF, но период восстановления не сглажен и является линейным (Рисунок 5 а3). Синаптический ток не используется для расчета уровня мембранного потенциала в ESRN, вместо этого он представлен суммой проекционных весов синапсов, тока утечки и генератора шума для повышения производительности модели. Сигналы, поступающие от возбуждающих синапсов, увеличивают мембранный потенциал, тогда как ингибирующие (или тормозные) синапсы уменьшают (из-за отрицательного веса) (Рисунок 6). В течение рефракторного периода нейрона, приходящие на него сигналы игнорируются и не могут повлиять на восстанавливающийся потенциал мембраны.

В модели ESRN используются следующие допущения для оптимизации кода нейрональной модели и ускоренной обработки нескольких тысяч нейронов и сотен тысяч синапсов:

1. использовались характеристики биологических нейронов спинного мозга (мотонейроны и интернейроны) меньших по размеру вместо классических параметров для головного мозга;
2. игнорируется расчет амплитуды мембранного потенциала после достижения порогового значения (в HH модели наоборот, проводится расчет, что сильнее нагружает центральный процессор или видеокарту);
3. ингибирование напрямую через снижение напряжение мембранного потенциала для уменьшения вероятности нейрона сгенерировать спайк.

Блок-схема модели ESRN показана на Рисунке 6. Сигналы приходящих спайков обрабатываются через маршрутизатор (router), который передает их соответствующему постсинаптическому нейрону. Все синапсы ESRN, представленные в виде массивов параметров, то есть они сохраняются на протяжении всей симуляции (синаптическая пластичность STDP также отсутствует в силу экономии ресурсов ЦП). Так как каждый спайк представлен как единица, то нет необходимости умножать синаптический вес на входной стимул. Динамика мембранного потенциала клетки нейрона (сомы) поддерживается с помощью

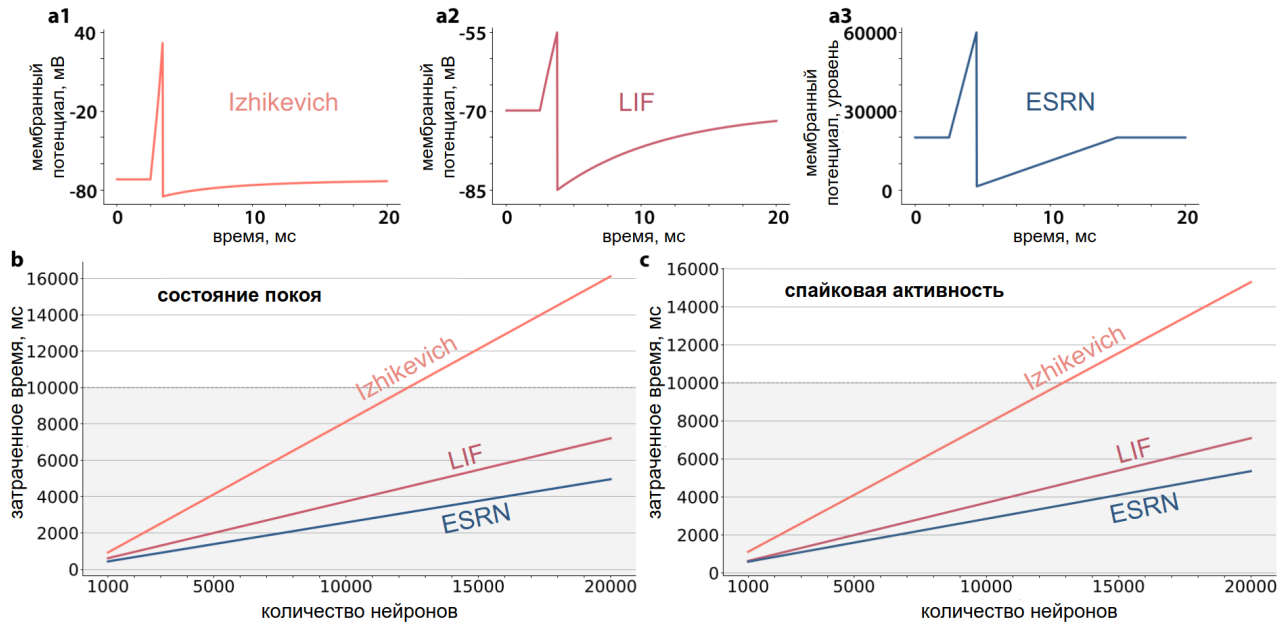

Рисунок 5. Форма спайков: **a1** - модель Ижикевича (IZH), **a2** – модель нейрона с утечкой LIF (leaky integrate and fire), **a3** – модель упрощенного цифрового нейрона (ESRN). **b.** Тесты зависимости числа нейронов от времени вычислений в состоянии покоя и **c.** спайковой активности (генерация спайков каждые 7.5 мс). Продолжительность моделирования составляет 10 секунд, поэтому от моделей ожидается, что вычисления будут не дольше 10 секунд с коэффициентом реального времени выше 1 (симуляционное время / затраченное время).

параметра тока утечки, который представляет собой работу натриево-калиевых насосов и утечку калия через мембрану (Рисунок 4б). Было обнаружено, что шум важен даже в этой примитивной модели из-за стохастичности всех нейрональных каналов (даже в биологических простых нейрональных структурах спинного мозга). Параметры тока утечки и шума устанавливаются для каждого нейрона индивидуально для повышения вариабельности спайковой активности и контроля уровня стохастичности.

Функция порогового состояния (threshold, Рисунок 6) запускает выходной импульсный генератор с заданной продолжительностью формы сигнала, когда мембранный потенциал достигает порогового значения. Период восстановления мембранного потенциала клетки (Рисунок 4а) реализуется как отрицательная обратная связь с “soma integrator” (Рисунок 6), которая снижает уровень потенциала индивидуально для каждого нейрона. В “synapse table” хранятся номера ячеек массивов соответствующие порядковому номеру нейрона и синаптические задержки для адресации сигнала (Параграф 3.2.2).

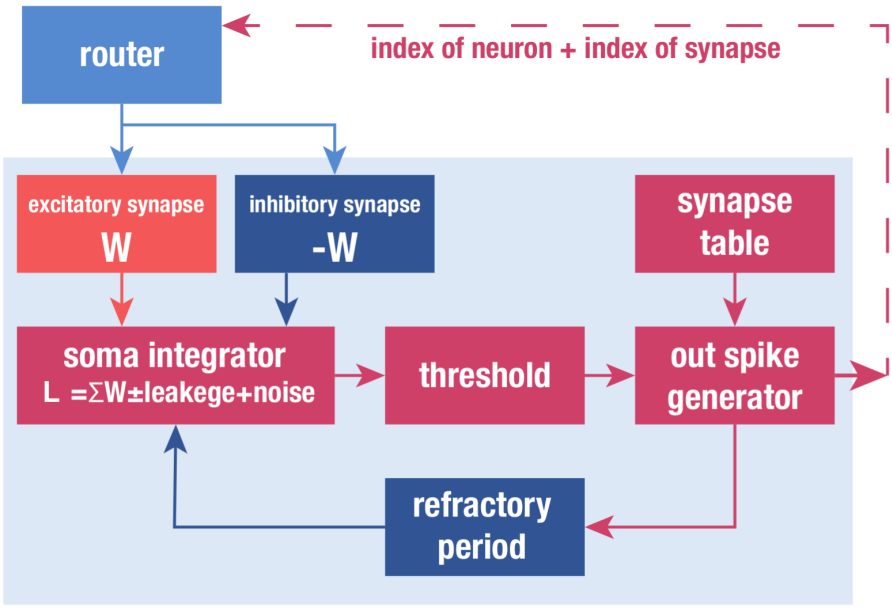

Рисунок 6. Схематичное представление работы упрощенной модели нейрона (ESRN). Маршрутизатор (router) направляет входные импульсы от соответствующего синапса (возбуждающего или ингибирующего) постсинаптическому нейрону (“адресату”). Вес возбуждающего (excitatory) синапса положителен, и, следовательно, импульс, поступающий через этот входной канал, увеличивает вероятность того, что нейрон сгенерирует спайковую активность, тогда как подавляющий (inhibitory) сигнал снижает эту вероятность. В “soma integrator” используется простая формула для расчета уровня мембранного потенциала (уравнение 1). Если уровень достигает порогового значения (threshold), то генератор (out spike generator) передает сигнал маршрутизатору, что произошёл спайк у пресинаптического нейрона. Данный генератор также блокирует поступление новых сигналов в период рефрактерности (восстановления).

Для уменьшения потребления оперативной памяти и сокращения времени расчета в ESRN используется тип данных unsigned short (uint16_t) (2 байта) для переменной мембранного потенциала, поскольку операции с плавающей запятой выполняются медленнее (но полностью зависят от архитектуры ЦП), чем целочисленные операции [43]. Использование unsigned short, а не char или unsigned int, позволяет эффективнее (с точки зрения вариабельности и оперативной памяти) использовать диапазон значений от 0 до 65 535. Появляется возможность в сэкономленную память добавлять дополнительные ESRN-нейроны в 2 раза больше, чем LIF моделей, и около 4 раз больше, чем моделей Ижикевича (в 4 и 8 раз соответственно если используется тип данных double). Это может быть важно при размещении вычислений на одноплатных компьютерах.

Для сравнения трех моделей нейронов (ESRN, LIF и IZH) использовался изолированный вычислительный узел кластера КФУ, где проводилась OpenMP симуляция одного и того же размера нейронной сети на процессоре Intel® Xeon®

E5-2650 v2 (8 ядер, 2.6 ГГц) с 96 ГБ ОЗУ. Во время симуляции все модели нейронов имели постоянные, неизменяемые параметры (без рандомизатора), за исключением мембранного потенциала (и переменной U_m [33] для модели IZH), эти параметры были реализованы как массивы данных (по размеру равные количеству нейронов).

Имитировалось два состояния нейронов: состояние покоя и постоянная спайковая активность. В симуляции покоя нет входного тока ни в одном нейроне. В симуляции постоянной активности генерировались спайки каждого нейрона с интервалом 7.5мс (периода спайка плюс рефрактерный период как на Рисунке 4a), чтобы гарантировать достижение потенциала покоя. В данном эксперименте не были использованы синапсы или записывающие устройства (виртуальные мультиметры или детекторы спайков), чтобы минимизировать их влияние на производительность.

Моделирование включало в себя 10 тестов для каждой модели нейрона для топологий от 1 000 до 20 000 нейронов с размером шага 1 000. Итоговое количество симуляция одной модели составило 200 запусков. Полученные данные представлены в виде линии тренда (с использованием функции polyfit пакета NumPy). В режиме покоя (Рисунок 5b) модели ESRN и LIF работают быстрее, чем реальное время при симуляции 20 000 нейронов. Модель IZH работает быстро, если число нейронов меньше 13 000 (Рисунок 5b). В режиме постоянной спайковой активности (Рисунок 5c) ESRN и LIF соответствуют требованиям лаборатории по работе в реальном времени, в то время как IZH намного выше этой отметки.

Все три проверенные модели имеют линейные зависимости от времени с разными наклонами (Рисунок 5b и Рисунок 5c). Таким образом была создана новая модель нейрона, которая позволит нейросимулятору GRAS (и другим) достичь повышенной производительности и проводить расчеты в режиме реального времени. Код модели опубликован в GitHub репозитории проекта лаборатории вместе с нейросимулятором GRAS: github.com/research-team/memristive-spinal-cord/tree/master/GRAS.

3.2.2. Создание модели синапса для передачи сигнала

Синапс – это соединение между нейронами, которые служат для передачи сигналов [20]. В био-правдоподобных нейросимуляторах, которые учитывают морфологию нейрона, имитируют также пре-синаптическую часть, синаптическую щель между нейронами и пост-синаптическую часть (Рисунок 7) [4; 15; 26]. Такая детализированность увеличивает время расчета состояний всех синапсов в несколько раз.

Повысить скорость вычислений можно за счет уменьшения биологической правдоподобности в разумных пределах. Под этим подразумевается сохранение основных механизмов синапса без использования морфологии аксонов/дендритов и отсутствии расчетов открытия ионных каналов. Активацию рецепторами, приводящую к открытию определенных ионных каналов, можно заменить упрощенной формулой (Формула 2) расчета возбуждающего (EPSC) или ингибирующего (IPSC) [62] постсинаптического трансмембранного тока и перенести этот расчет на пост-синаптический нейрон (Формула 3).

$$I_{syn}(t) = g_{ion}(t) * (membrane\_voltage(t) - E_{ion}), \tag{2}$$

$$g_{ion}(t) = g_{ion}(t-1) * (1 - \frac{sim\_step}{\tau_{syn}}) \tag{3}$$

В формуле 2 для получения постсинаптического тока ионная проводимость умножается на разницу между мембранным потенциалом нейрона и синаптическим потенциалом равновесия ионов (E_{ion}), представляющим собой напряжение мембраны, при котором данный нейротрансмиттер не вызывает поток ионов через ионный канал [20]. Изменение ионной проводимости в момент времени t в формуле 3 вычисляется путем умножения предыдущего значения на константу в виде τ_{syn} синаптической постоянной времени, характеризующей время нарастания синаптической проводимости ($\frac{\text{нСм}}{\text{мкм}^2}$) с поправкой на шаг симуляции (при учете площади поверхности нейрона в квадратных микрометрах).

Были использованы два основных механизма передачи сигнала для синапсов: возбуждающей через AMPA/NMDA-рецепторы для глутамата и ингибирующей через $GABA_A$/$GABA_B$-рецепторов для GABA. Основная роль возбуждающей и тормозящей нейротрансмиссии заключается в том, чтобы иметь

возможность балансировать между увеличением и уменьшением вероятности генерации выходного спайка. Это поведение является критическим для моделирования схем спинного мозга.

Для дополнительного повышения производительности в GRAS использовались статические синапсы без их пластичности (Алгоритм 1) (STDP, spike-timing-dependent plasticity), то есть изменение веса в течение симуляции не происходит. Для STDP требуются буферы спайковой активности нейронов, которые не позволят выполнять вычисления в режиме реального времени, что очень важно с точки зрения цели создания GRAS. Данный шаг оправдан еще тем, что механизмы адаптации через STDP не всегда играют важную роль, например, в случае краткосрочных симуляций (менее суток симуляционного времени), когда STDP не может существенно влиять на синаптические веса [5; 9; 54].

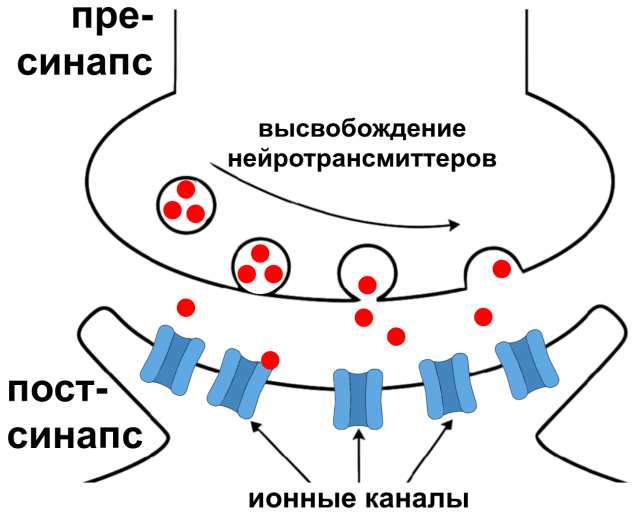

Рисунок 7. Упрощенная схема работы химического синапса [20], когда сигнал между нейронами передается через высвобождение в синаптическую щель нейротрансмиттеров, которые, прикрепляясь к рецепторам ионных каналов, открывают их. Нейротрансмиттеры указаны красным цветом, синим – ионные каналы. Открытые ионные каналы (в зависимости от типа) увеличивают/понижают мембранный потенциал клетки (Рисунок 4).

Псевдокод обновления состояния синапсов представлен в Алгоритме 1. Изначально происходит проверка на существование спайка у пресинаптического нейрона и состояния таймера (-1 означает таймер сброшен, синапс готов к новой передаче сигнала) синаптической задержки (строка 2, Алгоритм 1). Теоретически, при возникновении последующего спайка, когда таймер уже ведет отсчет, должен существовать вектор спайков, чтобы создать очередь из передаваемых сигналов постсинаптическому нейрону. Существует очень малая вероят-

ность (только когда время рефрактерного периода меньше, чем синаптическая задержка), что у нейрона вновь образуется спайк прежде, чем истечет таймер синаптической задержки. Поэтому, чтобы увеличить производительность, подобный вектор таймеров не используется, и маловероятный одиночный спайк проигнорируется.

Если условие строки 2 Алгоритма 1 истинно, устанавливается новое значение таймера, которое в дальнейшем будет уменьшаться на один шаг (строка 9, Алгоритм 1). Когда таймер принимает нулевое значение (строка 5, Алгоритм 1), запускается механизм увеличения проводимости постсинаптического нейрона на уровень, указанный в синаптическом весе (положительный для возбуждения и отрицательный для торможения), а постсинаптический нейрон, при достижении порогового состояния, уже генерирует исходящий спайк.

```
1  for tid ∈ grid stride loop thread IDs do
2      if synaptic_delay_timer[tid] == −1 and pre_neuron has spike then
3          synaptic_delay_timer[tid] ← synaptic_delay[tid]
4      end
5      if synaptic_delay_timer[tid] == 0 then
6          post_neuron_conductivity += synaptic_weight_value[tid];
7          synaptic_delay_timer[tid] ← −1
8      end
9      if synaptic_delay_timer[tid] > 0 then
10         synaptic_delay_timer[tid] − −
11     end
12 end
```

Алгоритм 1: Функция обновления состояния синапсов во время каждой итерации. Таймер синаптической задержки определен тремя состояниями (строки 2, 5 и 9): если значение равно 0, синапс изменяет проводимость нейрона; если значение больше 0, таймер уменьшается на один шаг итерации; когда значение равно -1 и происходит спайк пресинаптического нейрона, выставляется новое время синаптической задержки.

Для сохранения потокобезопасности в строке 6 Алгоритма 1 используется CUDA функция $atomicAdd$. В некоторых реализациях программного кода [71] функция $atomicAdd$ замедляет работу, однако благодаря планировщику задач CUDA, возможное замедление скрывается одновременной работой сотней других потоков в блоке. В GRAS при тестировании подходов без/с $atomicAdd$ было выяснено, что задержка не оказывает видимого влияния на симуляции, время которых превышает 800 мс (в тесте использовался профилировщик производительности от Nvidia).

3.3. Архитектура GRAS

Нейросимулятор GRAS проектировался так, чтобы он был гибким и удобным для новых пользователей (студентов и сотрудников, работающих с симуляциями). Основными особенностями нового нейросимулятора являются скорость вычислений и простота написания нового функционала на языке C++. Достаточно низкий порог вхождения понимания кода позволяет расширять функционал инструмента собственными модулями.

Архитектуру GRAS можно представить в виде высокоуровневой диаграммы компонентов (Рисунок 8). Компонент нейросимулятора “Simulation” является основным и начальным в симуляции – он запускает все циклы, инициализирует модель ЦГУА (Параграф 3.4) и контролирует вычисления. Вспомогательный модуль “Generator” инициализирует группы нейронов в структуры данных, переводит вектор метаданных синапсов в массивы параметров, формирует соединения групп нейронов в соответствии с написанной топологией ЦГУА. Для отслеживания времени симуляции, переноса результатов из видеокарты в оперативную память, с дальнейшей генерацией результирующих файлов для их дальнейшего заполнения используется модуль “Tracker”.

Компонент “GPU” представляет собой интерфейс для работы с платформой CUDA (“GPU kernel”) и выделения видеопамяти для структур данных, находящихся в оперативной памяти (Параграф 3.1.2). “WARPUtils” обеспечивает оптимальное использование grid-stride цикла на основе данных о размере блоков и их количества.

Для описания клеточных связей и управления ими используется компонент “Bio models” (биологических моделей симулятора). Гибкость нейросимулятора GRAS обеспечивается возможностью использования различных моделей нейронов, которые содержат в себе разную степень биологической детализированности и формул расчетов (“Neuron models”). В данный момент реализовано три модели для различных задач лаборатории (повышенная биоправдоподобность или повышенная производительность): модель Ижикевича, Ходжкина – Хаксли и ESRN (Параграф 3.2.1). Параметры соединения клеток (синапса) и его структуры данных (Параграф 3.2.2) находятся в модуле “Synapse metadata”, который в дальнейшем конвертируется в читаемый CUDA формат с помощью компонента “Generator”. “ModelUtils” предоставляет вспомогатель-

ные функции вычислений для операций с нейронами и синапсами (конвертация времени в шаги итерации и обратно, интерпретация групп нейронов в их порядковые номера, генерация случайных параметров с разными законами распределения).

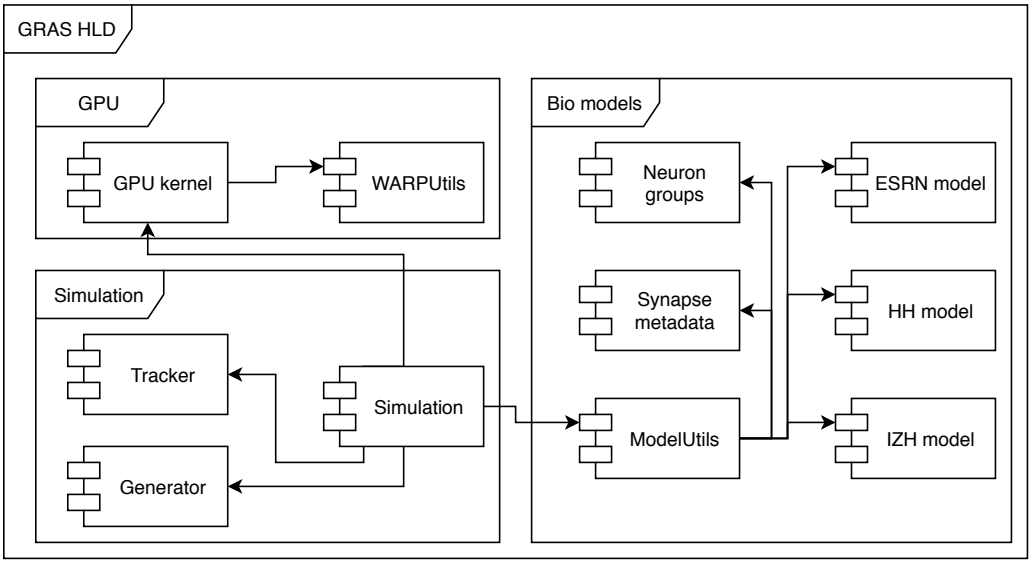

Рисунок 8. Высокоуровневое представление (диаграмма компонентов) архитектуры GRAS.

Био-правдоподобность симуляций может быть достигнута путем использования реализованной модели нейрона Ходжкина – Хаксли [28] с уменьшением дельты шага моделирования. Таким образом, realtime-фактор становится намного выше, чем 1 (отношение затраченного времени к симуляционному). Дополнительная гибкость симулятора реализуется путем настройки симуляционной модели. В случае, когда вычисления в режиме реального времени важны (например для нейропротезов или робототехнических систем) можно использовать модель нейрона Ижикевича [32] или предложенные в параграфах 3.2.1 и 3.2.2 дипломной работы упрощенную модель нейрона реального времени (ESRN) [40] и упрощенную модель синапса с увеличением дельты времени шага симуляции.

Вызов функции ядра GPU для обновления состояния нейронов и синапсов (на базе “grid-stride” подхода) происходит каждую итерацию в основном теле цикла нейросимуляции, расположенного на стороне CPU. Расположение основного цикла (строка 2 в Алгоритме 2) вне GPU имеет несколько причин.

Во-первых, это позволяет проще синхронизировать ядра для согласованной симуляции, требующей последовательности итераций: потоки CUDA не

должны обрабатывать N-ю итерацию, пока предыдущая $N-1$ не завершена. Размещение всего цикла в GPU может ускорить симуляцию в несколько раз, но в этом случае появляется необходимость в контролировании порядка выполнения блоков, что является нетривиальной задачей. Под нетривиальностью подразумевается внедрение системы глобальных флагов в видеопамяти (сложная реализация/отладка/дополнительные вычислительные расходы) для контроля последовательности вычислений или использование динамической вложенности потоков с авто-синхронизацией при завершении дочерних процессов (только для видеокарт с версией спецификации выше 3.5 [59]). Использование “grid-stride” для графического процессора с итерационным циклом извне позволяет менять количество блоков и их размер, чтобы добиться лучшей производительности для разных видеокарт [70]. Подход с использованием “grid-stride” также сохраняет когерентность кэша и свойства объединения памяти для SoA и неструктурированных данных [25]. Кроме того, пользователь может легко переключиться на последовательную (однопоточную) обработку данных для отладки функций графического процессора. Таким образом, использование ядра CUDA становятся более гибкими, масштабируемыми и отлаживаемыми [24]. Оптимальное количество блоков и их размер важны для достижения высокой загрузки графического процессора и уменьшения задержки вычислений [10]. CUDA калькулятор загруженности ядер на официальном сайте предоставляет данную информацию (возможен вариант подбора экспериментальным путем).

Во-вторых, чтобы получить результаты симуляции, необходимо дождаться её полного окончания, что приводит к двум проблемам: (1) если симуляция останавливается из-за переполнения видеопамяти или ошибки в одном из потоков, все данные теряются; (2) сохранять внутри ограниченной по размеру видеопамяти большие объёмы данных о состоянии нейронов в каждый момент времени (для отладки или получения полного обзора активности) не является возможным. Увеличение времени симуляции только ухудшает ситуацию в данном случае. Хранение данных в оперативной памяти (которая обычно больше по размеру) лучше защищает пользователя от переполнений, но увеличивает общее время симуляции из-за многократного копирования данных из видеопамяти (строка 7, Алгоритм 2).

```
1  (цикл на стороне CPU)
2  for iter = 0; iter < T_sim; iter++ do
3      (вызов функции видеокарты)
4      for tid = blockIdx.x * blockDim.x + threadIdx.x; tid < N; tid+=
         blockDim.x * gridDim.x do
5          обновление состояния нейронов под индексом tid
6      end
7      сохранение состояния нейронов через функцию cudamemcpy(...)
8      (вызов функции видеокарты)
9      for tid = blockIdx.x * blockDim.x + threadIdx.x; tid < M; tid+=
         blockDim.x * gridDim.x do
10         обновление состояния синапсов под индексом tid
11     end
12 end
```

Алгоритм 2: Псевдокод основной части симуляции. Цикл в строке 2 расположен на стороне процессора, циклы в строках 4–6 и 9–11 выполняются на стороне GPU. Переменная N обозначает общее количество нейронов, M – общее количество синапсов. Информация о размерности блока находится в переменной $blockDim$, порядковый номер блока в сетке указан в $blockIdx$, вызов $threadIdx$ возвращает номер потока в блоке. Размерность сетки (пространственная совокупность блоков) указана в $gridDim.x$.

Продолжительность передачи данных (от видеокарты к ОЗУ) и ее последующей обработки зависит от объема данных, например, копирование четырех состояний 3 500 нейронов приблизительно занимает 6% от общего времени симуляции (подсчет производился с помощью профилировщика NVidia).

Псевдокод главного цикла представлен в Алгоритме 2, который включает в себя “grid-stride” подход и вызов ядер видеокарты. Благодаря высокой эффективности расчетов на GPU было решено перенести расчеты мембранного потенциала нейронов и передачу сигналов между ними в ядро CUDA. Другими словами, каждую итерацию симуляции в основном цикле программы центрального процессора будет вызываться расчет данных нейронов и синапсов в памяти видеокарты.

Параметры нейронов представлены в виде нескольких одномерных массивов, где их длина зависит от количества нейронов и N-ая ячейка массива параметров соответствует N-му нейрону (Рисунок 9A). Такая же структура хранения данных используется для обновления состояний синапсов: синаптическая задержка, счетчик задержки, синаптический вес (сила сигнала) в нано Сименсах (увеличение/снижение электрической проводимости мембраны постсинап-

тического нейрона), порядковые номера пре- и постсинаптических нейронов для считывания/изменения их состояний (Рисунок 9Б).

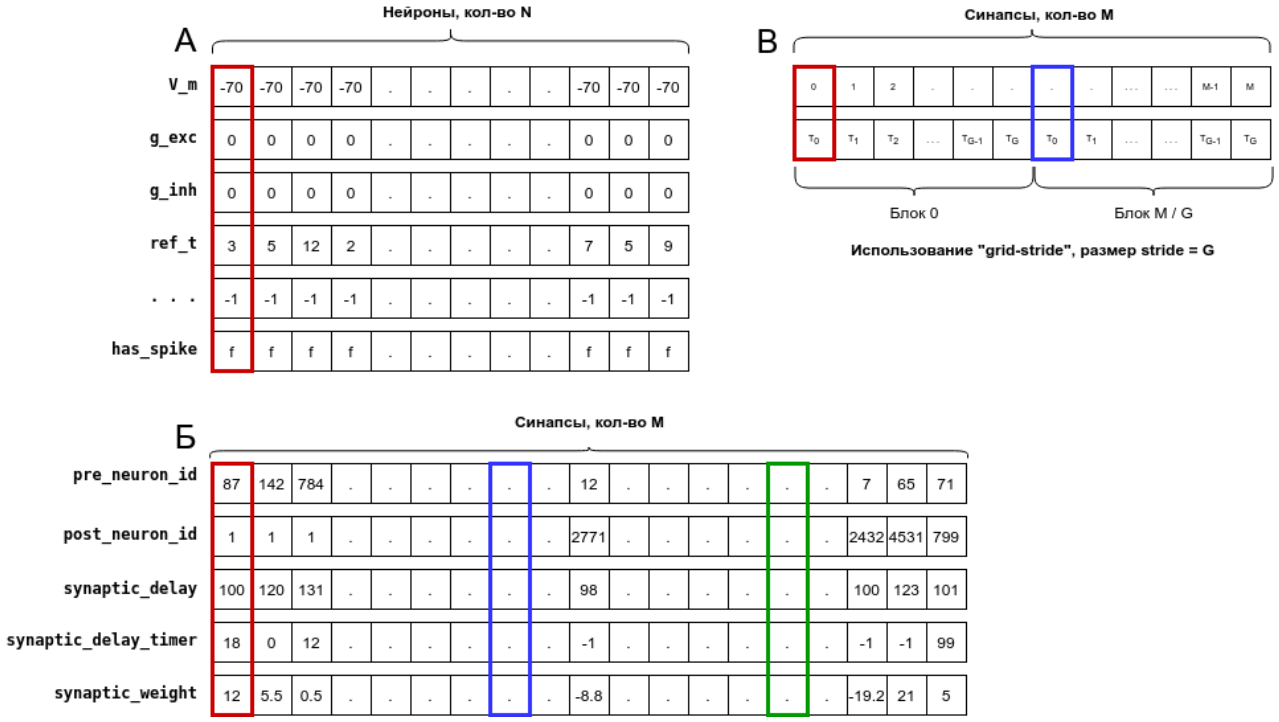

Рисунок 9. А. представление хранения данных нейронов: параметров и изменяемых состояний (в зависимости от используемой модели количество переменных может быть разным). Красным цветом показан поток, который обрабатывает один нейрон. Строки это параметры нейронов, столбцы – разные нейроны. **Б.** представление хранения данных синапсов: параметров и изменяемых состояний (без морфологии синапсов/STDP параметры, характеризующие синапсы, обычно, схожи). Разными цветами (красным, синим, зеленым) показан пример обработки одним потоком нескольких столбцов данных в рамках одной итерации симуляции. **В.** красным цветом показан поток, который обрабатывает на начальном этапе цикла “grid-stride” один синапс, когда как на следующей итерации блока (синий цвет) обработка данным потоком синапса сдвинется на индекс G вправо (количество потоков в блоке “stride”).

Для дополнительного ускорения [43] работы нейросимулятора на уровне обработки данных все переменные, где используется плавающая запятая, представлены в виде float (с одинарной точностью). В техническом описании [60] видеокарты K80 NVidia сказано, что производительность float вычислений может достичь до 8.73 ТФлопс, в то время как с double гораздо ниже – 2.91 ТФлопс. Ускорение работоспособности не ограничено только способом хранения данных (Параграф 3.1.3), использованием упрощенной модели нейрона ESRN (Параграф 3.2.1) или float переменных для минимизации времени вычислений. Был проведен дополнительный анализ по нахождению оптимального числа CUDA блоков и их размеров (Параграф 3.5).

3.4. Воссоздание работы центрального генератора упорядоченной активности отдела спинного мозга в вычислительной модели GRAS

Модель ЦГУА была разработана сотрудниками лаборатории нейроморфных вычислений и нейросимуляции на основе ранее проведенных исследований [37; 38] в клинике Mayo (Рочестер, США). Итоговый вариант модели представлен на рисунке 10. Модель состоит из двух основных компонентов: первый уровень моносинаптического ответа (снизу) и второй уровень формирования полисинаптического ответа (сверху).

В среде GRAS нейроны модели представлены в виде массивов данных с параметрами клеток и их состояниями (Параграф 3.3), а клеточная группировка является структурой данных “Group” с переменными имени группы, первого индекса нейрона и последнего. В примере Алгоритма 3 показано, как формируется группа 120-ти нейронов ингибирующего интернейронального пула мышцы разгибателя (“iIP_E”, строка 1) и 50-ти нейронов-осцилляторов, генерирующих паттерн на втором уровне модели (“OM0_1”, строка 2). Соединение нейронов производится путем вызова функции, аргументами которой являются: объект пресинаптической группы нейронов, постсинаптическая группа, время синаптической задержки (к примеру 1.5 мс) и значение силы изменения проводимости мембраны нейрона (к примеру $-9.5\frac{\text{нСм}}{\text{мкм}^2}$).

1 $Group\, iIP\_E = form\_group('iIP\_E', 120);$
2 $Group\, OM0\_1 = form\_group('OM0\_1', 50);$
3 $connect\_fixed\_outdegree(iIP\_E, OM0\_1, 1.5, -9.5);$

Алгоритм 3: Псевдокод создания групп нейронов и генерации коннектомов между ними по правилу фиксированного количества синапсов от пре-синаптического нейрона (значение задается разработчиком топологии).

Топология включает в себя разные типы нейронов, отличающиеся по своей форме/размеру/свойствам возбуждения. Для воссоздания того же поведения, что и в спинном мозге млекопитающего (крысы), были использованы следующие параметры для модели Ходжкина – Хаксли:

- C_m – ёмкость мембраны, сгенерированная с использованием закона нормального распределения для создания большей вариабельности спайковой активности интернейронов и законом бимодального распределения

для мотонейронов согласно проведенным исследованиям [58] ($0.02\frac{\text{пФ}}{\text{мкм}^2}$ для мотонейронов и $0.01\frac{\text{пФ}}{\text{мкм}^2}$ для интернейронов [18]);

- E_{Na} – потенциал реверсии для натриевого тока (50 мВ для интернейронов и мотонейронов) (Рисунок 4) с максимальной проводимостью тока $g_{Na} = 0.5\frac{\text{нСм}}{\text{мкм}^2}$ мембраны клетки;
- E_K – потенциал реверсии для калиевого тока (-90 мВ для интернейронов и мотонейронов) (Рисунок 4) с максимальной проводимостью тока $g_K = 3\frac{\text{нСм}}{\text{мкм}^2}$ мембраны клетки;
- E_L – потенциал реверсии для тока утечки (-72 мВ для интернейронов и мотонейронов) и проводимостью тока утечки $g_L = 0.02\frac{\text{нСм}}{\text{мкм}^2}$;
- переменные E_{exc} и E_{inh} отвечают за возбуждающий/ингибирующий синаптический потенциал равновесия ионов (0 мВ и -80 мВ соответственно для всех типов нейронов);
- τ_{exc} и τ_{inh} – описывают синаптическую постоянную времени, характеризующую время нарастания синаптической проводимости для возбуждающего и ингибирующего синапсов (0.2 мс и 2 мс соответственно для всех типов нейронов) [51].

ЭЭС представлена в виде генератора спайковой активности, работающего с частотой 40 Гц (частота изменялась при разных экспериментах с моделью). C1–C5 являются группами нейронов, соединенных с генераторами, имитирующих кожный вход со стопы (Рисунок 10). Их активация происходит последовательно с длительностью, соответствующей скорости передвижения лап крысы (125 мс для скорости 6 см/с, 50 мс для 13.5 см/с и 25 мс для 21 см/с).

Датчики спайков и мультиметры подключаются ко всем нейронам всех групп при настройке модели, если необходимо повысить производительность, датчики ставятся только на мотонейроны. Результирующим внутренним мембранным потенциалом (intracellular) для групп нейронов является их усредненное значение по количеству нейронов. Итоговая модель содержит в себе 3 500 нейронов и более 500 000 синапсов (на один нейрон в среднем приходится 142 синапса). Модель разработана для симуляции двух групп мышц (сгибатель/разгибатель). Результаты симуляции описаны в Главе 4. В перспективе планируется расширение топологии для новых групп мышц.

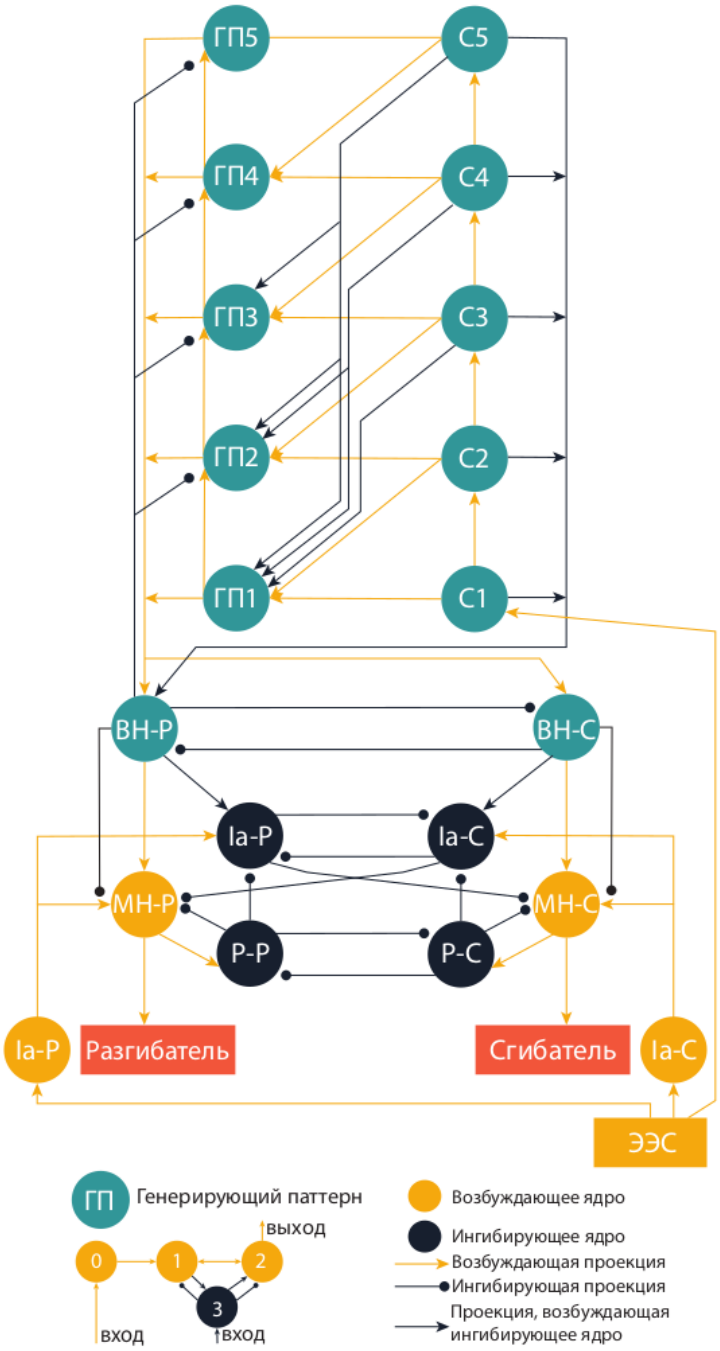

Рисунок 10. Схематичное изображение модели центрального генератора упорядоченной активности и рефлекторной дуги, которые симулировались в среде нейросимулятора GRAS. Эпидуральная электростимуляция спинного мозга (ЭЭС), помимо кожных входов, является входным сигналом, который активизирует генерацию паттерна ходьбы. Голубым цветом указаны сенсорные волокна, ведущие из стопы С1–С5, и группы нейронов, генерирующие паттерн ходьбы (ГП1–ГП5). Жёлтым цветом показаны группы нейронов, взаимодействующие с мышцами млекопитающего: мотонейроны сгибающей мышцы (МН-С), мотонейроны разгибающей мышцы (МН-Р), Ia-афференты сгибающей мышцы (Ia-С) и Ia-афференты разгибающей мышцы (Ia-Р). Тёмно-синим цветом показаны ингибирующие группы нейронов рефлекторной дуги: клетки Реншоу мышцы разгибателя (Р-Р) и мышцы-сгибателя (Р-С), Ia-интернейроны мышцы-разгибателя (Ia-Р) и Ia-интернейроны мышцы-сгибателя (Ia-С) [76].

3.5. Ускорение работы нейросимулятора для видеокарты Nvidia K80

Помимо тестов производительности в параграфах 3.1 и 3.2.1 был выполнен тест по поиску оптимального количества блоков и их размеров для модели реализованной в среде GRAS с использованием видеокарты NVIDIA® Tesla® K80 (2 x 2 496 CUDA ядер с частотой 562 MHz, CUDA v7.5). Для других видеокарт NVidia результат тестов будет отличаться при различных характеристиках (количества ядер, тактовой частоты и скорости доступа к памяти), что означает архитектурную зависимость оптимизированного кода.

По рекомендациям онлайн справочника CUDA [59] размер блока всегда должен быть кратен 32 для достижения максимальной производительности. Это обусловлено размером варпа (warp), представленный группой из 32 потоков. В аппаратной архитектуре CUDA варпом является минимальная единица исполнения, которая обрабатывает данные SIMT-способом в мультипроцессорах CUDA. Мультипроцессор (SM) исполняет одновременно один варп [59]. Стоит отметить, что на SM может (перемежаясь по инструкциям) почти одновременно выполняться несколько варпов. С точки зрения производительности, выставлять размер блока больше, чем размер варпа (несколько варпов в одном блоке с соблюдением кратности 32) выгодно [59]. Благодаря планировщику варпов в CUDA и использованию нескольких варпов в одном блоке в симуляционных вычислениях GRAS уменьшается время на обращение к глобальной памяти – пока одни группы потоков ожидают данные, над другими исполняются инструкции и кешируются прочитанные данные (для соседних варпов одного SM), что придает дополнительную скорость к вычислениям (Рисунок 12).

В зависимости от архитектуры используемой видеокарты, её мощностей и написанного CUDA кода зависит оптимальное соотношение размера блока и их количества. Очень маленький размер блока (к примеру от 32 до 64 потоков на блок) может ограничить производительность из-за занятости SM другими инструкциями варпов (представлен зубчатыми всплесками на рисунке 12). Большое количество потоков на блок (к примеру 1 024) также ограничивает производительность если существуют ограничения по видео-ресурсам. Например, лимит регистров выделенных на поток или одновременно работающих варпов на SM) [59], которые не позволяют на одном SM выполнить сразу несколько инструкций, создавая простои по времени (Рисунок 13).

При выделении от 128 до 512 потоков (универсальный диапазон) на блок вероятность возникновения вышеупомянутых проблем меньше. Обычно [30; 57] для кода не существует больших различий в производительности между 128 или 256 потоками на блок (как видно на рисунке 12 и 13).

Результатом тестов производительности работы GRAS стало выводом оптимального размера блоков и их количества для функций расчета состояний нейронов и синапсов: 230 блоков в каждом по 224 потоков и 200 блоков в каждом по 128 потоков соответственно. Один поток grid-stride обрабатывает 1 нейрон (3 500 сгенерированных нейронов) и 10 синапсов (из примерно 250 000 сгенерированных синапсов и 25 600 потоков). Количество потоков, превышающее количество нейронов, позволяет планировщику варпов в CUDA оперативнее (Рисунок 13) рассчитывать состояния нейронов (“лишние” потоки обрезаются по количеству сгенерированных нейронов), то есть от расположения планировщиком задач CUDA варпов на мультипроцессорах и их количества снижается задержка доступа к памяти (количество обращений к массивам данных у нейронов выше, чем у синапсов) [59; 63].

В конце работы по оптимизации было проведено итоговое сравнение работы трёх нейросимуляторов. Нейросимулятор NEURON имеет самое большое время вычислений (указан синей линией на рисунке 11), так как он имеет самую большую биологическую правдоподобность, где проводится расчет мембранного потенциала с учетом морфологии, размера клетки, толщины мембраны и температуры. Заметно резкое увеличение времени вычислений при вычислении 10 000 нейронов. Стоит отметить, что данный нейросимулятор не рассчитан на большое количество нейронов, и архитектура возможно не приспособлена к быстрой работе с такими объемами данных. Для NEST и NEURON были выделены ресурсы Intel® Xeon® E5-2650 v2 (2.6 ГГц), а именно 24 потока OpenMP и 24 процесса MPI соответственно. NEST (указан оранжевой линией на рисунке 11) имеет лучшее время по сравнению с NEURON, но недостаточное для задач лаборатории. Для симуляций GRAS использовались ресурсы NVIDIA® Tesla® K80 (2 x 2 496 CUDA ядер с частотой 562 MHz, CUDA v7.5). Не оптимизированный GRAS (указан зеленой линией на рисунке 11) при симуляции до 4 000 нейронов затрачивает время меньше, чем время самой симуляции (с учетом синапсов время вырастет), но в дальнейшем показывает ступенчатый

рост (влияние комбинации размера блока и вычисляемых нейронов). Использование новой модели нейрона ESRN (параграф 3.2.1) с оптимизацией количества блоков и их размеров даёт лучший результат – при симуляции 10 000 нейронов затраченное время ниже симулированного (указан красной линией на рисунке 11). Так как в модели ЦГУА используется небольшое количество нейронов (от 3 000 до 4 000), то остается достаточно “свободного” времени, в которое можно уместить расчет синапсов (при сохранении шага симуляции в 0.025 мс). Для увеличения производительности можно сократить количество вызова ядра GPU и переноса данных из видеопамяти в оперативную путём увеличения шага симуляции до 0.1 мс с сохранением достаточной вариабельности спайковой активности для воссоздания работы ЦГУА и рефлекторной дуги человека.

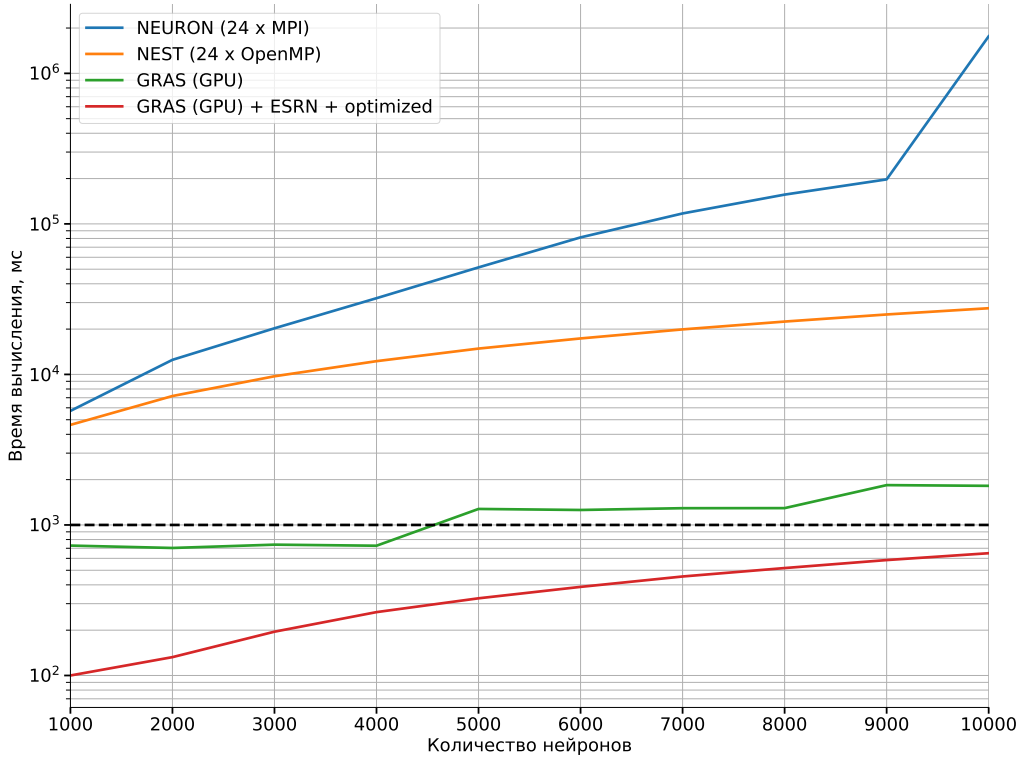

Рисунок 11. Итоговое сравнение работы трёх нейросимуляторов NEURON, NEST, GRAS. Проводились вычисления нейронов находящихся в состоянии покоя на протяжении 1 000мс (черная пунктирная линия). Результаты, находящиеся выше данной линии, не являются вычислениями в режиме реального времени. На оси X отражено количество нейронов. Были созданы топологии без соединений, где каждый нейрон был изолирован друг от друга. Использовалась одинаковая модель нейрона Ходжкина – Хаксли и одинаковые формулы расчета мембранного потенциала (кроме ESRN реализации). Для каждого числа нейронов проводился пятикратный замер времени работы только основного цикла расчета состояний нейронов (так называемых “полезных” вычислений), чтобы не учитывать влияние служебных функций нейросимуляторов и затрат на генерацию групп нейронов. Использовалось среднее значение времени тестов.

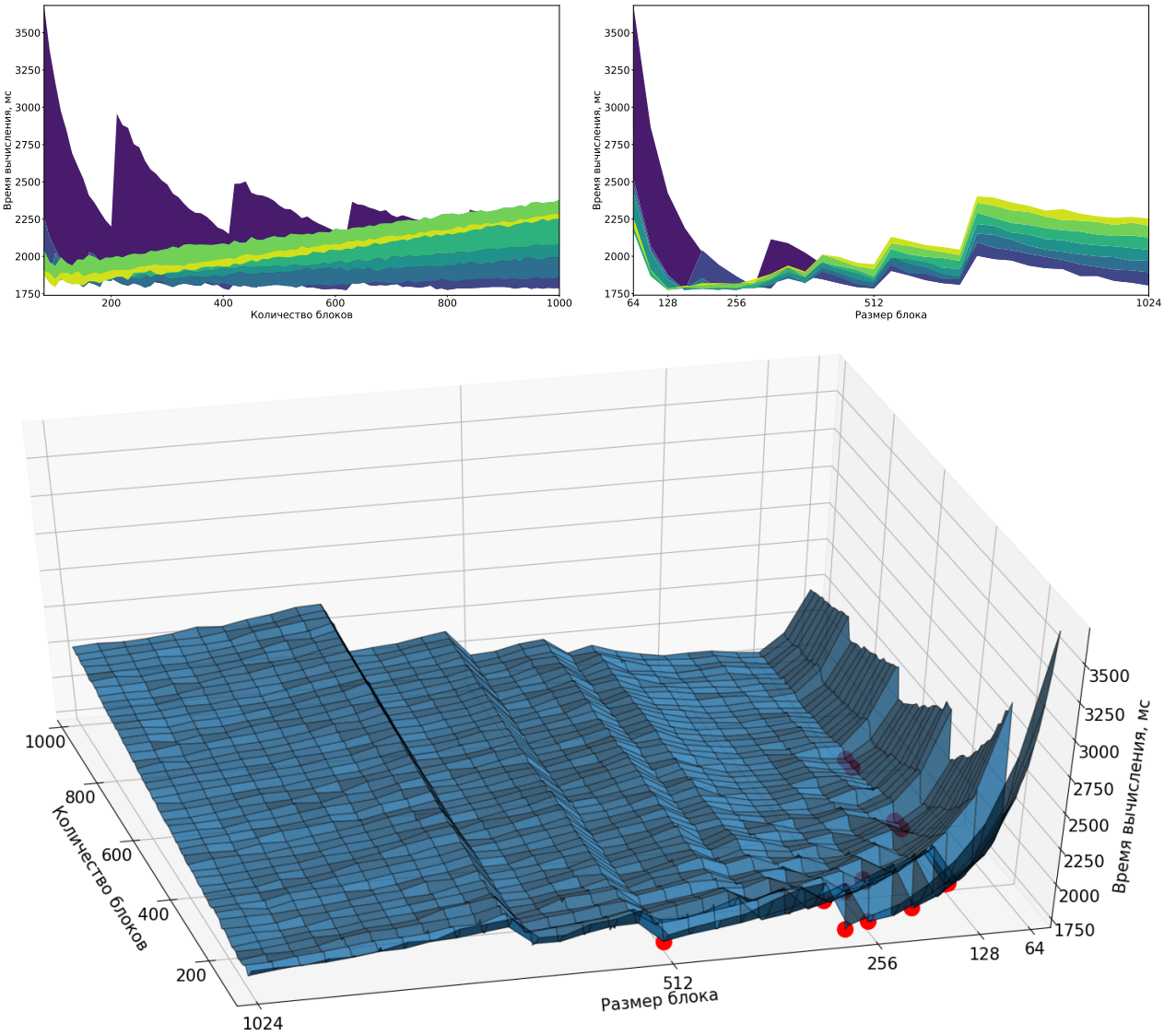

Рисунок 12. Оценка производительности функции ядра GPU обновления состояния синапсов в среде GRAS на базе вычислительного узла кластера КФУ. Симулировалось 1 000 мс работы модели ЦГУА и рефлекторной дуги на базе модели нейрона Ходжкина – Хаксли. Для нивелирования влияния системных процессов на вычисления было произведено 5-кратное повторение тестов с дальнейшим усреднением значений. Рисунок сверху слева демонстрирует зависимость времени вычисления от количества блоков. Чем ярче цвет контура плотности, тем больше вероятность нахождения в ней значений тестов (большее количество точек расположено в этом диапазоне). Зубчатость характерна при размере блока от 32 до 64 потоков. Оптимальным количеством блоков является диапазон от 90 до 410 с диапазоном размеров от 128 до 512 потоков. Рисунок сверху справа демонстрирует зависимость времени вычисления от размера блока. Заметно, что увеличение его размера не придает большей производительности. Изображение снизу является трёхмерным представлением результатов тестов. По сравнению с рисунком 13 поверхность более зубчатая, что может говорить о том, что выделение количества потоков больше, чем возможных ячеек памяти массива данных в функции обновления состояния нейронов, действительно позволяет планировщику задач лучше скрывать задержку чтения/записи при почти любых комбинациях размера блока и их количества. Лучшие 0.5% результатов по времени вычисления представлены в виде красных точек. Самая оптимальная комбинация: 200 блоков в каждом по 128 потоков.

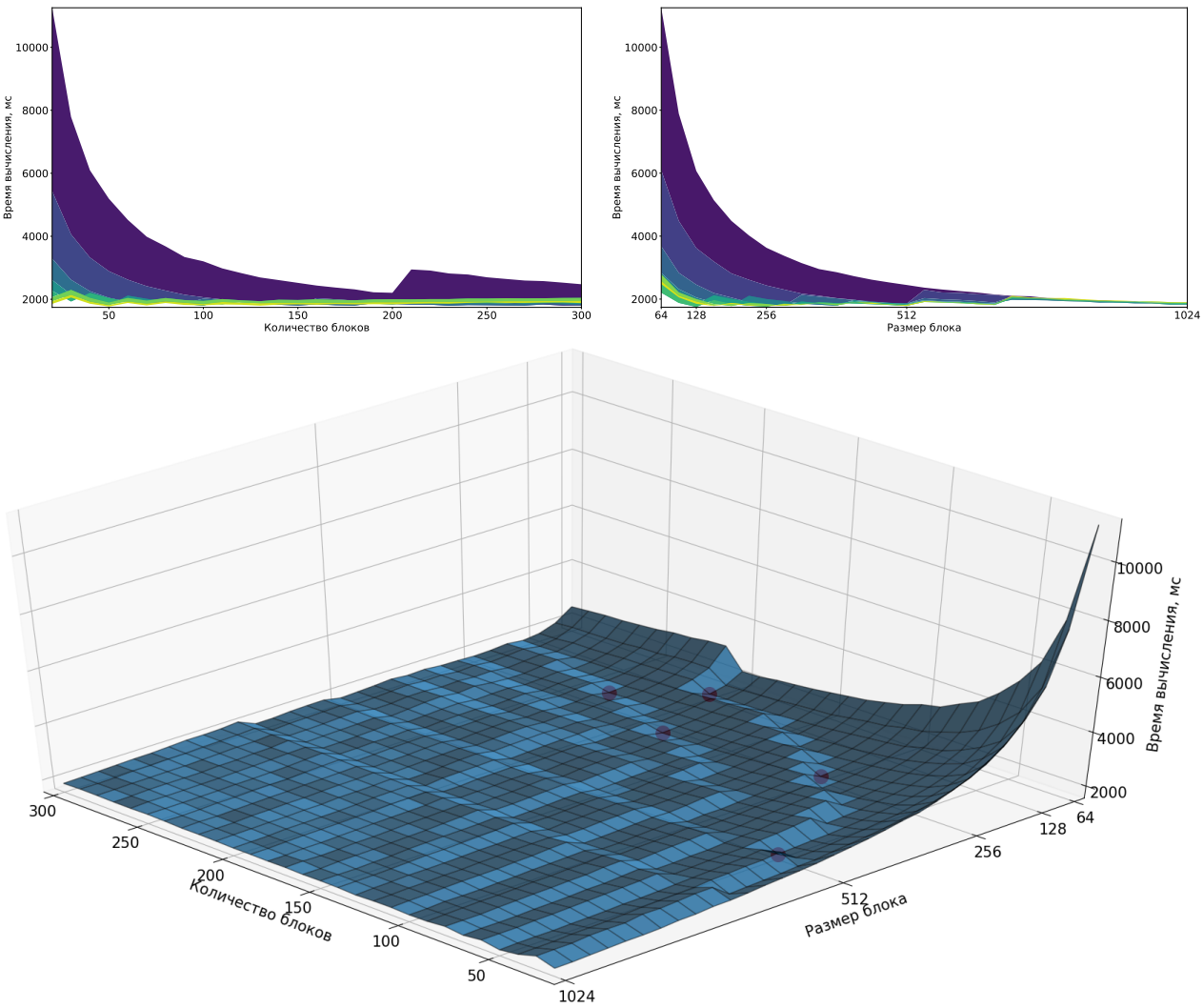

Рисунок 13. Оценка производительности функции ядра GPU обновления состояния нейронов в среде GRAS на базе вычислительного узла кластера КФУ. Симулировалось 1 000 мс работы модели ЦГУА и рефлекторной дуги на базе модели нейрона Ходжкина – Хаксли. Для нивелирования влияния системных процессов на вычисления было произведено 5-кратное повторение тестов с дальнейшим усреднением значений. Рисунок сверху слева демонстрирует зависимость времени вычисления от количества блоков. Чем ярче цвет контура плотности, тем больше вероятность нахождения в ней значений тестов (большее количество точек расположено в этом диапазоне). Рисунок сверху справа демонстрирует зависимость времени вычисления от размера блока. Заметно, что увеличение его размера не придает большей производительности (плавное увеличение времени вычислений). Изображение снизу является трёхмерным представлением результатов тестов. При grid-stride подходе “лишние” потоки обрезаются по количеству нейронов (то есть, выделив 10 000 потоков на 1 000 нейронов, будут задействованы только 1 000, но на разных мультипроцессорах). Однако как и в случае обновления состояния синапсов оптимальным количеством блоков является диапазон от 50 до 230 с диапазоном размеров от 128 до 512 потоков. Лучшие 0.5% результатов по времени вычисления представлены в виде красных точек. Самая оптимальная комбинация: 230 блоков в каждом по 224 потоков.

ГЛАВА 4. РЕЗУЛЬТАТЫ GRAS СИМУЛЯЦИИ МОДЕЛИ ОТДЕЛА СПИННОГО МОЗГА КРЫСЫ

Воссозданная по схеме (Рисунок 10, параграф 3.4) модель ЦГУА была симулирована в среде GRAS. На Рисунке 14 показаны результаты пяти моделирований трех шагов виртуальной крысы с частотой эпидуральной электростимуляции 40 Гц (результаты нарезаны по 25 мс для того, чтобы увидеть формируемый паттерн). Моносинаптический ответ как в биологических, так и в симуляционных данных, проявляется на 3–6 мс. Этот ответ постепенно ослабевает с каждым слайсом, что также удалось повторить в симулируемой модели – реализованный внутри афферент имеет дугообразную форму сигнала, которая передается на мотонейроны. Полисинаптическая активность мотонейронов находится в диапазоне 10–25 мс. Группы осциллирующих нейронов, которые генерируют паттерн, вызывают волнообразные возмущения. При увеличении разброса в рандомизации параметров нейронов можно добиться большего сходства с биологическими данными. Пример на Рисунке 14 демонстрируется возможность применения разработанного нейросимулятора GRAS, так как точность симулируемой модели зависит от точности её настройки специалистом.

Временная задержка между моносинаптическими и полисинаптическими ответами увеличивается с 1-го по 6-й срез (слайс), образуя трапециевидную фигуру полисинаптической активности (фигура розового цвета на Рисунке 14), что было показано в исследованиях [17; 31], с которыми сравниваются симуляционные данные. Распределение задержек визуально коррелирует с биологическими данными [17], где были использованы одинаковые экспериментальные параметры (скорость перемещения и частота стимуляции), что может говорит о том, что нейросимулятор GRAS может использоваться для создания био-правдоподобных симуляций отделов спинного мозга.

Задачей дипломной работы также было проверить работу нейросимулятора в аппаратной связке с цифро-аналоговым преобразователем, чтобы симулируемая в режиме реального времени активность мотонейронов ЦГУА могла передаваться на электроды. На Рисунке 15 показан результат работы в аналоговом представлении. Усредненное значение мембранного потенциала выводилось из GRAS посредством библиотеки поставляемой с ЦАП L-Card E-502. Выходной

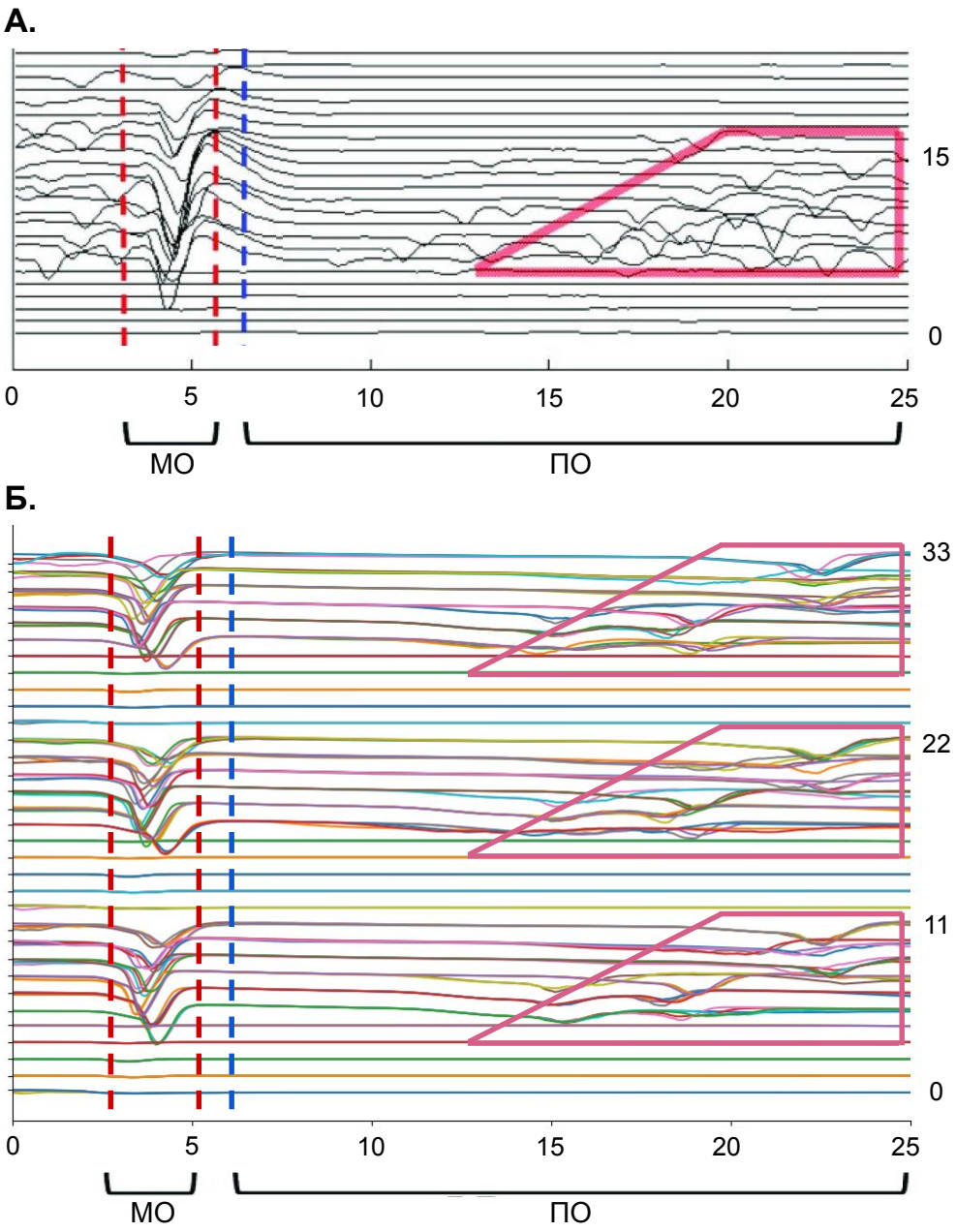

Рисунок 14. А. Миограмма биологической активности мышцы-разгибателя снятой с крысы, передвигающейся на беговой дорожке со скоростью 21 см/с под эпидуральной электростимуляцией спинного мозга с частотой 40 Герц. Данные были нарезаны на отрезки (слайсы) по 25 мс. Красной трапецией выделены поли-ответы (ПО), формирующие ступенчатый паттерн, где с каждой нарезкой увеличивается время задержки сигнала от начала моно-ответа (МО) [17]. Красным пунктиром показаны границы моно-ответа. Синей линией – начало поли-ответа.
Б. Результат работы нейросимулятора GRAS, имитирующего в реальном времени поведение мотонейронов ЦГУА, активность которых передается на мышечные волокна. Скорость передвижения для симулируемой модели отдела мозга крысы также составляет 21 см/с с такой же частотой ЭЭС. Было выполнено 5 запусков эксперимента для демонстрации вариабельности спайковой активности. Всего просимулировано в каждом эксперименте 3 шага крысы длительностью 825 мс (три шага по 150 мс работы мышцы-разгибателя и 125 мс мышцы-сгибателя).

сигнал может быть как усилен, так и ослаблен для различных целей экспериментов (к примеру меньшая сила тока для нервов рук и большая для частей тела с более толстой прослойкой между кожей и нервом). Шаг дискретизации (или скорости увеличения напряжения) с применением ЦАП был достигнут до 1 мс, что является хорошим результатом для стимуляции двигательной активности ног. Для стимуляции более сложных и точных двигательных процессов (движение пальцев рук) потребуется более высокая частота дискретизации, к примеру с шагом изменения напряжения в 0.1 мс. Так как нейросимулятор имеет возможность работать в режиме реального времени с упрощённой моделью нейрона в диапазонах шага симуляции 0.1–0.25 мс, данная задача может быть осуществима. В перспективе, архитектура GRAS может быть использована для различных нейропротезов по стимуляции мышечных/нервных волокон.

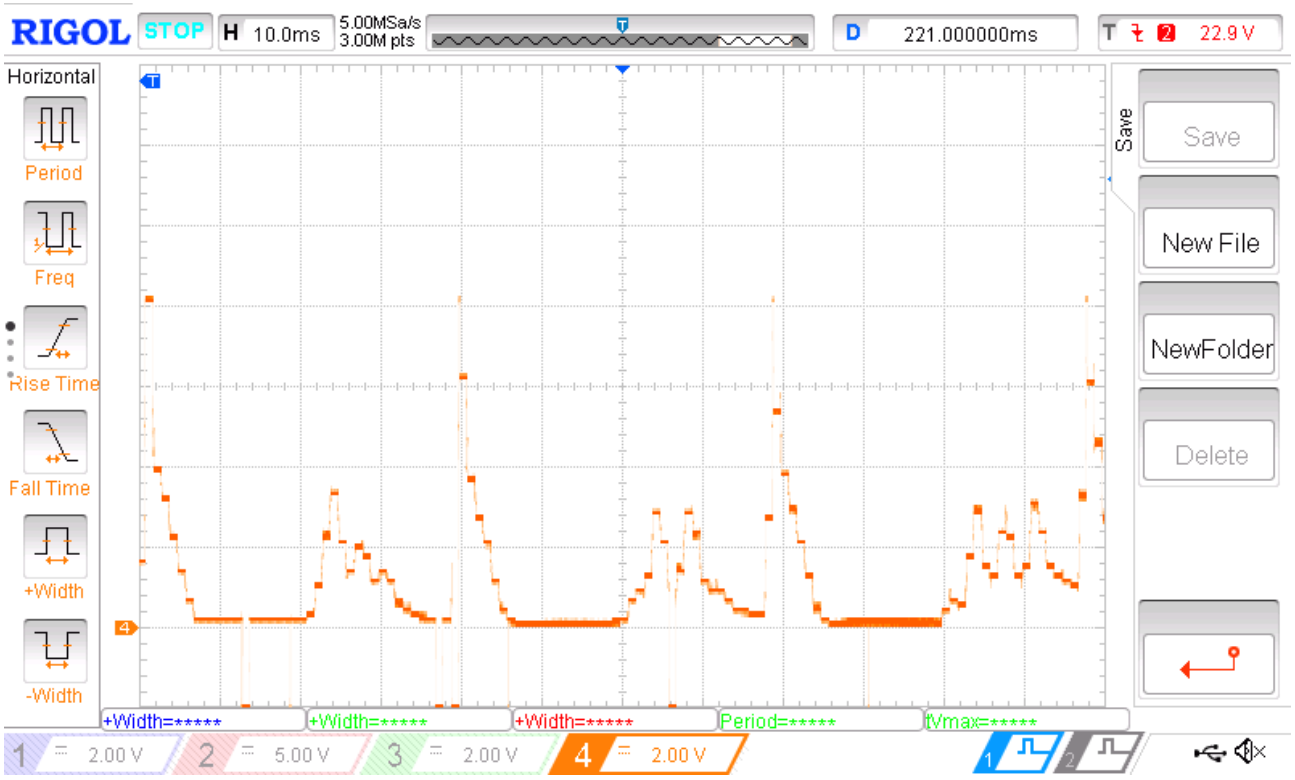

Рисунок 15. Скриншот результата работы GRAS в развертке (без нарезки по слайсам) с выводом на цифро-аналоговый преобразователь (использовался осциллограф RIGOL DS1054Z). На оси X отложена временная шкала (одна клетка = 10мс). На оси Y показано напряжение (одна клетка = 2 Вольта). Режим реального времени соблюдён – необходимое расстояние между моно-ответами должно быть не более 50мс при частоте ЭЭС 40 Герц (здесь демонстрируется 40мс, или 1.24 realtime-фактор, то есть необходимо небольшое искусственное замедление работы кода для синхронизации по частоте ЭЭС).

ЗАКЛЮЧЕНИЕ

Цель дипломной работы выполнена в полном объёме – реализован нейросимулятор GRAS на базе архитектуры GPU, работающий в режиме реального времени, имитирующий био-правдоподобную топологию ЦГУА и выводящий симулируемую активность мотонейронов через цифро-аналоговый преобразователь. На базе проведенного сравнения существующих нейросимуляторов появилось понимание трендов в разработке био-инспирированных систем. GRAS уже активно используется лабораторией в проекте по созданию протеза сегмента спинного мозга для реабилитации людей, которые получили травму данного отдела и стали парализованными. Для возможности симуляции в режиме реального времени были использованы имеющиеся средства ускорения работоспособности и проделана подробная работа по ее анализу. В будущих реализациях нейросимулятора планируется перенос архитектуры на одноплатные маломощные устройства, чтобы проводить вычисления для носимых нейропротезов или робототехнических систем. Простота нейросимулятора позволяет использовать дополнительную интеграцию с периферийными устройствами.

По результатам дипломной работы и проекта memristive spinal cord были написаны научные статьи для зарубежных журналов в соавторстве [40; 41; 68]. Кроме того, планируется к публикации в этом году статья в известном мировом журнале Nature Communication, аккумулирующая все результаты и наработки проекта лаборатории.

Код нейросимулятора GRAS и дополнительные Python-скрипты для визуализации данных и других экспериментальных наработок опубликованы в репозитории проекта лаборатории нейроморфных вычислений и нейросимуляций КФУ: github.com/research-team/memristive-spinal-cord/tree/master/GRAS.

СПИСОК ЛИТЕРАТУРЫ

1. *Akl S. G.*, *Nagy M.* Introduction to Parallel Computation // Parallel Computing: Numerics, Applications, and Trends / под ред. R. Trobec, M. Vajteršic, P. Zinterhof. — Springer London, 2009. — С. 43—80.
2. *Balaban P.*, *Vorontsov D.*, *Dyakonova V.* [и др.]. The Central Pattern Generators // Neuroscience and Behavioral Physiology. — 2015. — Дек. — Т. 45. — С. 42—57.
3. *Barlas G.* Chapter 2 - Multicore and parallel program design // Multicore and GPU Programming / под ред. G. Barlas. — Boston : Morgan Kaufmann, 2015. — С. 27—54.
4. *Ben-Shalom R.*, *Artherya N. S.*, *Cross C.* [и др.]. NeuroGPU, software for NEURON modeling in GPU-based hardware // bioRxiv. — 2019.
5. *Bocchio M.*, *Nabavi S.*, *Capogna M.* Synaptic Plasticity, Engrams, and Network Oscillations in Amygdala Circuits for Storage and Retrieval of Emotional Memories // Neuron. — 2017. — Т. 94, № 4. — С. 731—743.
6. *Boybat I.*, *Gallo M.*, *S.R. N.* [и др.]. Neuromorphic computing with multi-memristive synapses // Nature Communications. — 2017. — Нояб. — Т. 9. — С. 1—10.
7. *Chandra R.*, *Dagum L.*, *Kohr D.* [и др.]. Parallel programming in OpenMP. — San Diego, CA : Morgan, Kaufmann, 2001.
8. *Chapman B.*, *Jost G.*, *Pas R. v. d.* Overview of OpenMP // Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation). — The MIT Press, 2007. — С. 23—34.
9. *Citri A.*, *Malenka R. C.* Synaptic Plasticity: Multiple Forms, Functions, and Mechanisms // Neuropsychopharmacology. — 2008. — Т. 33, № 1. — С. 18—41 ; — Nature Publishing Group.
10. *Cook S.* Chapter 9 - Optimizing Your Application // CUDA Programming / под ред. S. Cook. — Boston : Morgan Kaufmann, 2013. — С. 305—440. — (Applications of GPU Computing Series).
11. *Corporation I.* Writing Optimal OpenCL™ Code with Intel® OpenCL SDK [Электронный ресурс] / Performance Guide. — 2011. — URL: software.intel.

com/sites/default/files/m/d/4/1/d/8/Writing_Optimal_OpenCL_28tm_29_Code_with_Intel_28R_29_OpenCL_SDK.pdf ; (дата обращения: 29.05.2020).

12. *Deska-Gauthier D.*, *Zhang Y.* The functional diversity of spinal interneurons and locomotor control // Current Opinion in Physiology. — 2019. — Апр. — Т. 8. — С. 99—108.
13. *Dimitrov M.*, *Mantor M.*, *Zhou H.* Understanding Software Approaches for GPGPU Reliability // Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units. — Association for Computing Machinery, 2009. — С. 94—104.
14. *Du P.*, *Weber R.*, *Luszczek P.* [и др.]. From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming // Parallel Computing. — 2012. — Т. 38, № 8. — С. 391—407.
15. *Einevoll G. T.*, *Destexhe A.*, *Diesmann M.* [и др.]. The Scientific Case for Brain Simulations // Neuron. — 2019. — Т. 102, № 4. — С. 735—744.
16. *Eppler J.*, *Helias M.*, *Muller E.* [и др.]. PyNEST: a convenient interface to the NEST simulator // Frontiers in Neuroinformatics. — 2009. — Т. 2. — С. 1—12.
17. *Gad P.*, *Lavrov I.*, *Shah P.* [и др.]. Neuromodulation of motor-evoked potentials during stepping in spinal rats // Journal of Neurophysiology. — 2013. — Т. 110, № 6. — С. 1311—1322.
18. *Gentet L. J.*, *Stuart G. J.*, *Clements J. D.* Direct Measurement of Specific Membrane Capacitance in Neurons // Biophysical Journal. — 2000. — Т. 79, № 1. — С. 314—320.
19. *Gerasimenko Y. P.*, *Lavrov I. A.*, *Courtine G.* [и др.]. Spinal cord reflexes induced by epidural spinal cord stimulation in normal awake rats // Journal of neuroscience methods. — 2006. — Т. 157, № 2. — С. 253—263.
20. *Gerstner W.*, *Kistler W.*, *Naud R.* [и др.]. Neuronal dynamics: From single neurons to networks and models of cognition. — 2014. — Гл. Dendrites and Synapses. С. 58—80.
21. *Gill M. L.*, *Grahn P. J.*, *Calvert J. S.* [и др.]. Neuromodulation of lumbosacral spinal networks enables independent stepping after complete paraplegia // Nature Medicine. — 2018.

22. *Grahn P. J.*, *Lavrov I. A.*, *Sayenko D. G.* [и др.]. Enabling Task-Specific Volitional Motor Functions via Spinal Cord Neuromodulation in a Human With Paraplegia // Mayo Clinic Proceedings. — 2017. — Т. 92, № 4. — С. 544—554.
23. *Grandars.ru*. Мембранный потенциал покоя и действия [Электронный ресурс] / Медицина и Физиология. — 2020. — URL: www.grandars.ru/college/medicina/membrannyy-potencial.html ; (дата обращения: 25.05.2020).
24. *Harris M.* CUDA Pro Tip: Write Flexible Kernels with Grid-Stride Loops [Электронный ресурс] / NVIDIA Developer Blog. — 2013. — URL: devblogs.nvidia.com/cuda-pro-tip-write-flexible-kernels-grid-stride-loops ; (дата обращения: 19.03.2020).
25. *Herten A.*, *Brömmel D.*, *Pleiter D.* GPU-Accelerated Particle-in-Cell Code on Minsky // High Performance Computing. — Cham : Springer International Publishing, 2017. — С. 205—219.
26. *Hoang R.*, *Tanna D.*, *Jayet Bray L.* [и др.]. A novel CPU/GPU simulation environment for large-scale biologically realistic neural modeling // Frontiers in Neuroinformatics. — 2013. — Т. 7. — С. 19.
27. *Hodgkin A. L.*, *Huxley A. F.* A quantitative description of membrane current and its application to conduction and excitation in nerve // The Journal of Physiology. — 1952. — Т. 117, № 4. — С. 500—544.
28. *Hodgkin A. L.*, *Huxley A. F.* A quantitative description of membrane current and its application to conduction and excitation in nerve // The Journal of Physiology. — 1952. — Авг. — Т. 117, № 4. — С. 500—544.
29. *Hoefler T.*, *Schneider T.*, *Lumsdaine A.* Accurately measuring overhead, communication time and progression of blocking and nonblocking collective operations at massive scale // IJPEDS. — 2010. — Авг. — Т. 25. — С. 241—258.
30. *Hoshino T.*, *Maruyama N.*, *Matsuoka S.* [и др.]. CUDA vs OpenACC: Performance Case Studies with Kernel Benchmarks and a Memory-Bound CFD Application // Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. — Delft, Netherlands : IEEE Press, 2013. — С. 136—143. — (CCGRID '13).

31. *Islam R.*, *Cuellar C.*, *Felmlee B.* [и др.]. Multifactorial motor behavior assessment for real-time evaluation of emerging therapeutics to treat neurologic impairments // Scientific Reports. — 2019. — Дек. — Т. 9. — С. 1—16.
32. *Izhikevich E. M.* Which model to use for cortical spiking neurons? // IEEE Transactions on Neural Networks. — 2004. — Т. 15, № 5. — С. 1063—1070.
33. *Izhikevich E.* Simple model of spiking neurons // IEEE Transactions on Neural Networks. — 2003. — Нояб. — Т. 14, № 6. — С. 1569—1572.
34. *Jang H.*, *Park A.*, *Jung K.* Neural network implementation using CUDA and OpenMP // Proceedings - Digital Image Computing: Techniques and Applications, DICTA 2008. — 2008. — Янв. — С. 155—161.
35. *Khaydarova R.*, *Fishchenko V.*, *Mouromtsev D.* [и др.]. ROCK-CNN: a Distributed RockPro64-based Convolutional Neural Network Cluster for IoT. Verification and Performance Analysis. — 2020. — Апр.
36. *Kim Y.*, *Chortos A.*, *Xu W.* [и др.]. A bioinspired flexible organic artificial afferent nerve // Science. — 2018. — Т. 360, № 6392. — С. 998—1003.
37. *Lavrov I.*, *Dy C. J.*, *Fong A. J.* [и др.]. Epidural Stimulation Induced Modulation of Spinal Locomotor Networks in Adult Spinal Rats // Journal of Neuroscience. — 2008. — Т. 28, № 23. — С. 6022—6029.
38. *Lavrov I.*, *Gerasimenko Y. P.*, *Ichiyama R. M.* [и др.]. Plasticity of Spinal Cord Reflexes After a Complete Transection in Adult Rats: Relationship to Stepping Ability // Journal of Neurophysiology. — 2006. — Т. 96, № 4. — С. 1699—1710.
39. *Lavrov I.*, *Musienko P. E.*, *Selionov V. A.* [и др.]. Activation of spinal locomotor circuits in the decerebrated cat by spinal epidural and/or intraspinal electrical stimulation // Brain research. — 2015. — Т. 1600. — С. 84—92.
40. *Leukhin A.*, *Talanov M.*, *Suleimanova A.* [и др.]. Even Simpler Real-Time Model of Neuron // BioNanoScience. — 2020. — Т. 10, вып. 2. — С. 416—419.
41. *Leukhin A.*, *Talanov M.*, *Suleimanova A.* [и др.]. GPU-computing for Simulation of Neural Circuitry // Springer Lecture Notes in Computer Science (LNCS). — Springer International Publishing, 2020. — С. 1—10.

42. *Li C.*, *Belkin D.*, *Li Y.* [и др.]. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks // Nature Communications. — 2018. — Дек. — Т. 9. — С. 1—8.
43. *Limare N.* Integer and Floating-point Arithmetic Speed vs Precision [Электронный ресурс] / Nicolas LIMARE: Software Engineering. — 2014. — URL: nicolas.limare.net/pro/notes/2014/12/12_arit_speed ; (дата обращения: 23.02.2020).
44. *Lobov S.*, *Kazantsev V.*, *Makarov V. A.* Spiking Neurons as Universal Building Blocks for Hybrid Systems // Advanced Science Letters. — 2016. — Т. 22, № 10. — С. 2633—2637.
45. *Lobov S.*, *Mironov V.*, *Kastalskiy I.* [и др.]. A spiking neural network in sEMG feature extraction // Sensors. — 2015. — Т. 15, № 11. — С. 27894—27904.
46. *Matsufuru H.*, *Aoki S.*, *Aoyama T.* [и др.]. OpenCL vs OpenACC: Lessons from development of lattice QCD simulation code // Procedia Computer Science. — 2015. — Дек. — Т. 51. — С. 1313—1322.
47. *Mayo Clinic*. Spinal Cord Injury Rehabilitation Program [Электронный ресурс] / Patient Care & Health Information. — 2020. — URL: www.mayoclinic.org/tests-procedures/spinal-cord-injury-rehabilitation/about/pac-20395044 ; (дата обращения: 20.05.2020).
48. *McCulloch W. S.*, *Pitts W.* A logical calculus of the ideas immanent in nervous activity // The bulletin of mathematical biophysics. — 1943. — Т. 5, № 4. — С. 115—133.
49. *McDougal R. A.*, *Morse T. M.*, *Carnevale T.* [и др.]. Twenty years of ModelDB and beyond: building essential modeling tools for the future of neuroscience // Journal of Computational Neuroscience. — 2017. — Февр. — Т. 42, № 1. — С. 1—10.
50. *McIntosh-Smith S.*, *Boulton M.*, *Curran D.* [и др.]. On the Performance Portability of Structured Grid Codes on Many-Core Computer Architectures // Supercomputing / под ред. J. M. Kunkel, T. Ludwig, H. W. Meuer. — Cham : Springer International Publishing, 2014. — С. 53—75.
51. *McIntyre C. C.*, *Grill W. M.* Extracellular Stimulation of Central Neurons: Influence of Stimulus Waveform and Frequency on Neuronal Output // Journal of Neurophysiology. — 2002. — Т. 88, № 4. — С. 1592—1604.

52. *Mei G.*, *Tian H.* Impact of Data Layouts on the Efficiency of GPU-accelerated IDW Interpolation // SpringerPlus. — 2016. — Февр. — Т. 5. — С. 1—18.
53. *Mudalige G. R.*, *Reguly I. Z.*, *Giles M. B.* [и др.]. Performance Analysis of a High-Level Abstractions-Based Hydrocode on Future Computing Systems // High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation / под ред. S. A. Jarvis, S. A. Wright, S. D. Hammond. — Cham : Springer International Publishing, 2015. — С. 85—104.
54. *Nabavi S.*, *Fox R.*, *Proulx C. D.* [и др.]. Engineering a memory with LTD and LTP // Nature. — 2014. — Т. 511. — С. 348—352 ; — Number: 7509 Publisher: Nature Publishing Group.
55. *National Spinal Cord Injury Statistical Center*. Facts and Figures at a Glance [Электронный ресурс] / University of Alabama at Birmingham. — 2020. — URL: www.nscisc.uab.edu/Public/Facts%20and%20Figures%202020.pdf ; (дата обращения: 12.05.2020).
56. *Navarro C.*, *Hitschfeld N.*, *Mateu L.* A Survey on Parallel Computing and its Applications in Data-Parallel Problems Using GPU Architectures // Communications in Computational Physics. — 2013. — Сент. — Т. 15. — С. 285—329.
57. *Nickolls J.*, *Buck I.*, *Garland M.* [и др.]. Scalable Parallel Programming with CUDA // Queue. — 2008. — Март. — Т. 6. — С. 40—53.
58. *Niwa M.*, *Nakayama K.*, *Sasaki S.-I.* Morphological study of external oblique motor nerves and nuclei in cats // Anatomical Science International. — 2008. — Т. 83, № 1. — С. 17—25.
59. *NVIDIA Corporation*. NVIDIA CUDA C Programming Guide v10.2 [Электронный ресурс] / Developer Zone. — 2019. — URL: docs.nvidia.com/cuda/cuda-c-programming-guide/index.html ; (дата обращения: 29.03.2020).
60. *NVIDIA Corporation*. Compare 10 series graphics cards [Электронный ресурс] / GeForce products. — 2020. — URL: www.nvidia.com/en-us/geforce/products/10series/compare ; (дата обращения: 15.03.2020).
61. *OpenACC.org*. OpenACC Programming and Best Practices Guide [Электронный ресурс] / Programming Guide. — 2015. — URL: www.openacc.org/sites/default/files/inline-files/OpenACC_Programming_Guide_0.pdf ; (дата обращения: 19.02.2020).

62. *Purves D.*, *Augustine G.*, *Fitzpatrick D.* [и др.]. Neuroscience. — 2-е изд. — Sunderland (MA): Sinauer Associates, 2001. — Гл. Excitatory and Inhibitory Postsynaptic Potentials. С. 58—80.
63. *Quintero D.*, *Huertas L.*, *Kamenoue T.* [и др.]. Implementing an IBM High-Performance Computing Solution on IBM Power System S822LC. — IBM Redbooks, 2016. — Гл. CUDA C programs with the NVIDIA CUDA Toolkit. С. 186—189.
64. *Rosenblatt F.* The perceptron: a probabilistic model for information storage and organization in the brain // Psychological review. — 1958. — Т. 65, № 6. — С. 386.
65. *Shimobaba T.*, *Ito T.*, *Masuda N.* [и др.]. Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL // Optics Express. — 2010. — Май. — Т. 18, № 10. — С. 9955—9960.
66. *Sulyok A.*, *Balogh G.*, *Reguly I.* [и др.]. Locality optimized unstructured mesh algorithms on GPUs // Journal of Parallel and Distributed Computing. — 2019. — Авг. — Т. 134. — С. 50—64.
67. *Talanov M.* Spiking neural networks [Электронный ресурс] / Neuromorphic computing, neurosimulations. — 2019. — URL: github.com/max-talanov/1/blob/master/affective_computing_course/realistic_nns.md ; (дата обращения: 02.06.2020).
68. *Talanov M.*, *Leukhin A.*, *Suleimanova A.* [и др.]. Oscillator Motif as Design Pattern for the Spinal Cord Circuitry Reconstruction // BioNanoScience. — 2020. — С. 1—5.
69. *Teka W.*, *Marinov T. M.*, *Santamaria F.* Neuronal Spike Timing Adaptation Described with a Fractional Leaky Integrate-and-Fire Model // PLoS Computational Biology. — 2014. — Март. — Т. 10, № 3.
70. *Tian X.*, *Xu R.*, *Yan Y.* [и др.]. Compiler transformation of nested loops for general purpose GPUs // Concurrency and Computation: Practice and Experience. — 2016. — Т. 28, № 2. — С. 537—556.
71. *Ukidave Y.*, *Nina Paravecino F.*, *Yu L.* [и др.]. NUPAR: A benchmark suite for modern GPU architectures // ICPE 2015 - Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering. — 2015. — Февр. — С. 253—264.

72. *Wagner F. B.*, *Mignardot J.-B.*, *Goff-Mignardot C. G. L.* [и др.]. Targeted neurotechnology restores walking in humans with spinal cord injury // Nature. — 2018. — Нояб. — Т. 563, № 7729. — С. 65.
73. *World Health Organization*. Spinal cord injury. Key facts [Электронный ресурс] / WHO. — 2013. — URL: www.who.int/news-room/fact-sheets/detail/spinal-cord-injury ; (дата обращения: 11.04.2020).
74. *Zhang S.*, *Li W.*, *Jing Z.* [и др.]. Comparison of Three Different Parallel Computation Methods for a Two-Dimensional Dam-Break Model // Mathematical Problems in Engineering. — 2017. — Сент. — Т. 2017. — С. 1—12.
75. *Морозов И.*, *Млявых С.* Эпидемиология позвоночно-спинномозговой травмы (обзор) // Медицинский альманах. — 2011. — Т. 4. — С. 157—159.
76. *Сулейманова А.* — Моделирование базисных компонентов генераторов шагательных движений спинного мозга млекопитающего в нейросимуляторе NEURON: диплом. работа. Казанский (Приволжский) Федеральный Университет, Казань, 2019.