КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ И АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ

COMPUTER SIMULATION AND DESIGN AUTOMATION

УДК 004.436.2
DOI: 10.17586/0021-3454-2025-68-3-228-238

МЕТОД ВЫСОКОУРОВНЕВОГО ПРОЕКТИРОВАНИЯ МИКРОАРХИТЕКТУРЫ НЕЙРОМОРФНЫХ ПРОЦЕССОРОВ НА ОСНОВЕ ЯВНОГО ОТДЕЛЕНИЯ ВЫЧИСЛЕНИЙ ОТ ПОТОКА ТРАНЗАКЦИЙ

И. В. Лукашов*, А. А. Антонов, П. В. Кустарев

Университет ИТМО, Санкт-Петербург, Россия
** lukashov@itmo.ru*

Аннотация. Представлены оригинальный метод и прототип инструментального программного обеспечения для проектирования нейроморфных процессоров. Метод основан на высокоуровневом описании аппаратуры с явным (на уровне исходного кода) выделением потоков конвейерных транзакций, циркулирующих внутри аппаратной структуры, и отделением производимых при этом вычислений от логики динамического планирования и контроля потоков. Такой подход позволяет гибко комбинировать алгоритмы обработки данных с актуальными механизмами улучшения производительности и энергопотребления в аппаратной микроархитектуре, эффективно разделять ответственность при разработке сложной аппаратуры, повторно использовать автоконфигурируемые микроархитектурные структуры. Предлагается формализация концепции транзакции (в заданном контексте), маршрут проектирования аппаратуры на основе транзакций и алгоритм синтеза RTL-дизайна нейроморфных процессоров на основе „транзакционных“ описаний. Описан разработанный прототип инструментального обеспечения для синтеза процессоров, построенный на базе фреймворка для программно-управляемой генерации аппаратуры. Применение предлагаемого метода и компонентов САПР демонстрируется на примере разработки оригинального нейроморфного процессора, исполняющего модели полносвязных импульсных нейронных сетей. Данная разработка подтверждает достижимость конкурентных характеристик аппаратуры при существенном улучшении управляемости проекта, повторном использовании кода, снижении количества ошибок и общей трудоемкости проектирования.

Ключевые слова: *нейроморфные процессоры, микроархитектура, проектирование на уровне транзакций, САПР, RTL, HGF, HLS*

Благодарности: финансирование исследования выполнено за счет НИРСИИ Университета ИТМО (проект № 640097 „Разработка гибридного масштабируемого нейропроцессора на основе тензорных и нейроморфных ядер“).

Ссылка для цитирования: *Лукашов И. В., Антонов А. А., Кустарев П. В.* Метод высокоуровневого проектирования микроархитектуры нейроморфных процессоров на основе явного отделения вычислений от потока транзакций // Изв. вузов. Приборостроение. 2025. Т. 68, № 3. С. 228–238. DOI: 10.17586/0021-3454-2025-68-3-228-238.

METHOD OF HIGH-LEVEL MICROARCHITECTURE DESIGN OF NEUROMORPHIC PROCESSORS BASED ON EXPLICIT SEPARATION OF COMPUTATIONS FROM TRANSACTION FLOW

I. V. Lukashov*, A. A. Antonov, P. V. Kustarev

ITMO University, St. Petersburg, Russia
** lukashov@itmo.ru*

Abstract. An original method and a prototype of the software toolbox for designing neuromorphic processors are presented. The method is based on a high-level description of the hardware with explicit (at the source code level) allocation of pipeline transaction flows circulating inside the hardware structure and separation of the computations performed in this case from the logic of dynamic scheduling and flow control. This approach allows flexible combination

of data processing algorithms with up-to-date mechanisms for improving performance and energy consumption in the hardware microarchitecture, effective sharing of responsibilities in the development of complex hardware, and reuse of auto-configurable microarchitectural structures. A formalization of the transaction concept (in a given context), a hardware design route based on transactions, and an algorithm for synthesizing the RTL design of neuromorphic processors based on "transactional" descriptions are proposed. A prototype of the software toolbox for synthesizing processors built on a framework for software-controlled hardware generation is described. The application of the proposed method and CAD components is demonstrated using the example of the development of an original neuromorphic processor executing models of fully connected pulse neural networks. This development confirms the achievability of competitive hardware characteristics with a significant improvement in project manageability, code reuse, a reduction in the number of errors and the overall labor intensity of design.

Keywords: neuromorphic processors, microarchitecture, transaction-level design, CAD, RTL, HGF, HLS

Acknowledgements: The research was funded by Research Projects in the Field of Artificial Intelligence at ITMO University (project No. 640097 "Development of a hybrid scalable neuroprocessor based on tensor and neuromorphic cores").

For citation: Lukashov I. V., Antonov A. A., Kustarev P. V. Method of high-level microarchitecture design of neuromorphic processors based on explicit separation of computations from transaction flow. *Journal of Instrument Engineering*. 2025. Vol. 68, N 3. P. 228–238 (in Russian). DOI: 10.17586/0021-3454-2025-68-3-228-238.

Введение. Развитие методов искусственного интеллекта и возможностей аппаратных систем, обеспечивающих выполнение ими вычислений, позволяет решать все более сложные задачи в различных областях. Однако применение классических платформ для ускорения вычислений, таких как графические ускорители (Graphics Processing Unit — GPU) и различные тензорные ускорители (Tensor Processing Unit — TPU), приводит к большому энергопотреблению, особенно в процессе обучения. Это становится критичным в граничных (краевых) приложениях, где устройства работают на автономном питании [1]. В связи с этим возрастает актуальность энергоэффективных решений, таких как импульсные нейронные сети (ИмНС), реализованные на нейроморфных процессорах. Такие процессоры способны обеспечивать значительное снижение энергопотребления за счет использования событийно-управляемой обработки данных и большей разреженности вычислений [2]. Также нейроморфные процессоры, благодаря использованию нейронных сетей на импульсном базисе, открывают дополнительные возможности для изучения высокопроизводительных методов обучения и создания новых архитектур моделей нейронных сетей [3, 4]. При этом производительность и энергоэффективность системы максимизируется в случае, если микроархитектура нейропроцессора, включая его вычислительные и коммуникационные механизмы, адаптируется к параметрам конкретных моделей [5]. Цель настоящей статьи — ускорение разработки микроархитектуры специализированных нейроморфных процессоров за счет применения оригинальных подходов к высокоуровневому проектированию аппаратуры.

Предмет исследования. Нейроморфные процессоры обладают рядом особенностей микроархитектуры, которые в той или иной мере отличают их от других типов аппаратных вычислительных блоков, таких как универсальные процессоры, ускорители с фиксированной функцией, шинные матрицы и пр. Эти особенности обусловлены характером обработки данных и вычислительной нагрузкой, специфичными для моделей ИмНС и требующими соответствующей адаптации аппаратных механизмов для достижения высокой производительности и энергоэффективности (табл. 1).

В свою очередь, характерные особенности обработки потоков спайков в нейроморфных процессорах определяют эффективность применения тех или иных языков, методов и технологий проектирования. Обработка потоков спайков, представленных на микроархитектурном уровне в виде потоков транзакций различных типов, характеризуется множественными операциями обращения к данным и операциями над этими данными, как показано на рис. 1. Транзакции подразделяются на отдельные фазы вычислительного процесса.

На микроархитектурном уровне обработка спайков реализуется множественными обращениями к локальным банкам памяти, управлением выборкой нейронов для обработки, синхронизацией фаз и параллелизацией вычислений. Все вместе это образует типовой тракт обработки спайков в нейроморфных процессорах, который представлен на рис. 2.

Таблица 1

Особенности нагрузки	Особенности микроархитектуры
Динамически изменяющийся объем нагрузки. Вычислительный процесс в основном представляет собой обмен спайками* между нейронами, которые тоже могут испускать спайки при превышении порогового значения	Множественные потоки спайков реализуются в виде связанных между собой конвейерных транзакций, которые динамически зарождаются и циркулируют внутри вычислителя
Высокая интенсивность обмена спайками между нейронами	Локализация блоков обработки (нейроядер) с блоками памяти
Асинхронность обработки. В ИмНС спайки генерируются и передаются между нейронами асинхронно	Отсутствие жесткой привязки к расписанию выполнения транзакций друг относительно друга, динамичность планирования и применение глобально асинхронных, локально синхронных (GALS) и самосинхронных схем
Массовая параллельность. В течение каждого временного отрезка могут обрабатываться группы операций параллельно	Параллелизация вычислений. Массовая параллельность ИмНС позволяет реализовывать разные уровни пространственного и временного параллелизма, включая: 1) пространственный параллелизм; 2) временной параллелизм (посредством временного мультиплексирования операций нейронов и вычисления слоев ИмНС; 3) пространственный параллелизм вычислительных ядер (в многоядерных реализациях нейроморфных процессоров)

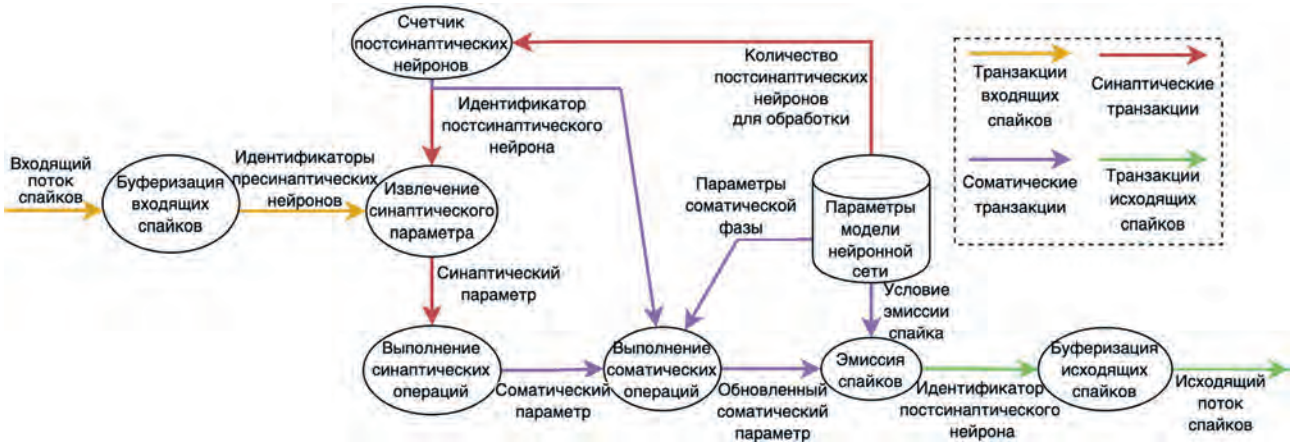

Рис. 1

Рис. 2

* Спайк (от англ. spike) — бинарное событие, возникающее в результате срабатывания функции активации импульсного нейрона, которая, как правило, представляет собой сравнение накопленного мембранного потенциала со значением порогового напряжения.

Существующие технологии проектирования аппаратуры предлагают различные способы ее абстрактного описания, эффективные, как правило, для синтеза вычислителей с конкретными особенностями микроархитектуры. Однако ни одна их технологий не предлагает адекватные методы описания и синтеза потоков транзакций нейроморфных процессоров. Причины этого обозначены в табл. 2, где перечислены преимущества и ограничения существующих технологий проектирования нейроморфных микроархитектур.

Таблица 2

Технология	Преимущества	Ограничения
RTL, синтезируемый SystemVerilog HDL	Универсальность	Акцент на разделении описаний комбинационной и последовательностной логики, отсутствие средств для абстрактного описания элементов микроархитектуры
Высокоуровневый синтез из C/C++ [6]	Отделение алгоритма от микроархитектуры (включая количество одновременно исполняемых транзакций)	Описание одиночного потока транзакций в рамках одного дизайна. Статическое планирование транзакций (временные слоты фиксируются на этапе синтеза на основе наихудшей задержки вычисления последних состояний переменных)
Моделирование на уровне транзакций SystemC/TLM	Отделение вычислительных процессов от коммуникационных транзакций между ними	Использование для моделирования в основном операций чтения/записи системной шины в системах на кристалле. Не подходит для детального описания микроархитектуры отдельных блоков
„Атомарные транзакции“ Bluespec [7]	Предоставление структурированного и детального контроля над переходами состояний в регистрах в рамках синхросигнала	Отсутствие автоматизации создания и оптимизации конвейеров и других высокоуровневых механизмов управления вычислительным процессом
„Конвейерные транзакции“ TL-Verilog [8]	Обеспечение возможности отделить описание данных в конвейерных транзакциях от механизмов контроля потока („сшивания“ стадий)	Фокус на автоматизации интеграции простых, “стандартных” механизмов на основе препроцессинга расширенного кода на SystemVerilog. Не поддерживается определение собственных типов транзакций, иерархий транзакций и т. д.

Из рассмотренных примеров наиболее близкой к рассматриваемой задаче представляется технология TL-Verilog (в первую очередь, благодаря базовой поддержке отделения конвейерных транзакций от контроля потока), однако в контексте синтеза нейроморфных процессоров требуется развитие данного подхода в следующих направлениях:

— формализовать понятие транзакции в виде, абстрагированном от конкретных механизмов динамического планирования и контроля потока в нейроморфных процессорах, описать ее свойства и атрибуты;

— предоставить возможность задавать собственные типы транзакций, определяющие поведение, планирование и контроль потоков транзакций в рамках высокоуровневой модели управления вычислительным процессом;

— разработать методы ручной и/или (полу)автоматической оптимизации микроархитектуры на основе транзакционного описания для ускорения исследования пространства проектных решений и адаптации нейропроцессоров к специфическим и изменяющимся требованиям проектов.

Транзакционный метод проектирования. Предлагаемый метод проектирования основан на описании микроархитектуры в терминах потоков динамических конвейерных транзакций, ассоциированных со спайками импульсных нейронных сетей и другими операциями внутри нейроморфного процессора. В отличие от описания на уровне RTL, в „транзакционном“ описании в явном виде выделяется контекст транзакции (набор состояний), который динамически зарождается, циркулирует и завершается в рамках описываемой аппаратной структуры. Циркуляция происходит через последовательность аппаратных подсистем, в ходе чего производится обработка данных, и при этом в системе может одновременно находиться множество транзакций, в том числе на разных стадиях обработки. Таким образом, порядок обработки

транзакций отделяется от описания аппаратной инфраструктуры динамического планирования и контроля потоков транзакций, что открывает возможность автоматической генерации (и конфигурирования под проектные требования) данной инфраструктуры на основе задействованных в пользовательском описании (и поддерживаемых генератором) типов состояний, типов транзакций и самой пользовательской логики обработки. В отличие от большинства развитых технологий высокоуровневого синтеза, данный метод не направлен на полную (и, как правило, недостаточно эффективную) автоматизацию синтеза микроархитектуры. Вместо этого фокус направлен на улучшение конфигурируемости, контролируемости проектов, гибкой композируемости произвольной логики обработки и переиспользуемых (повторно используемых) аппаратных механизмов.

Маршрут проектирования на основе транзакций. Предлагаемый маршрут проектирования показан на рис. 3. Проектирование начинается с формирования функциональной спецификации проектируемой аппаратуры. Далее необходимо осуществить концептуальное проектирование микроархитектуры, определив общую структуру вычислителя, функциональность ключевых подсистем и набор микроархитектурных транзакций, включая типы транзакций, место и время их зарождения, маршрутизацию через аппаратную структуру, стадии обработки, а также место и время их завершения (см. рис. 1 и 2). Спецификация уровня микроархитектурных транзакций формализуется в виде транзакционной модели и приводится на специальном языке описания транзакций (см. далее). Данный язык реализован как библиотека программных классов (готовых либо кастомизированных), генерирующих описание аппаратуры на языке SystemVerilog, которые, в свою очередь, используются в рамках стандартных маршрутов проектирования для ПЛИС и ASIC.

Рис. 3

Формализация понятия транзакции. Транзакция — структура данных, представляющая собой набор связанных полей, имеющих общий порядок инициализации, циркуляции в рамках аппаратного блока и завершения. Транзакция имеет жизненный цикл:

1. Инициация транзакции по некоторому событию или расписанию.

2. Активный этап циркуляции транзакции с обработкой и изменением ее полей.

3. Завершение транзакции с выполнением или невыполнением завершающего действия.

Транзакция может быть представлена кортежем следующего вида:

$$\tau = (\text{Type}, \text{Payload}, \text{AST}, \text{Metadata}); \quad (1)$$

$$\text{Payload} = (\text{field}_1, \text{field}_2, \ldots, \text{field}_n), \quad (2)$$

где Payload — полезные данные транзакции, содержащие набор полей (field), для нейроморфных процессоров это могут быть, например, поля статических, динамических и других параметров нейрона; Metadata — метаданные, содержащие служебные атрибуты (например, идентификаторы очередей, статус транзакции и пр.); AST (Abstract Syntax Tree) — аппаратно-ориентированное абстрактное синтаксическое дерево, содержащее набор операций, реализующее логику обработки транзакции; Type — тип транзакции.

Синхронизация транзакций осуществляется путем разбиения на фазы обработки. Каждая транзакция проходит набор фаз обработки Phases:

$$\text{Phases} = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}, \tag{3}$$

где каждая фаза φ является кортежем вида

$$\varphi = \{\text{Event}, \text{Type}, f_{\text{logic}}\}, \tag{4}$$

где Event — событие, инициирующее фазу обработки (например, сигнал тика, запускающий соматическую фазу обработки); Type — тип фазы (например, синаптическая, соматическая); f_{logic} — функция обработки транзакции (например, логика обработки импульсного нейрона):

$$f_{\text{logic}}: \text{AST} \times \tau \rightarrow \tau'. \tag{5}$$

Язык транзакционного описания нейроморфных микроархитектур. Как было рассмотрено ранее, в типичной микроархитектуре нейроморфных процессоров существуют несколько основных потоков данных: потоки входящих и исходящих спайков, потоки синаптических и соматических параметров. Для описания их обработки предлагаются следующие типы транзакций:

— транзакция синаптического параметра, которая инкапсулирует микроархитектурные механизмы обращения к параметрам синапсов (весовым коэффициентам, синаптическим задержкам, типам синапса и др.);

— транзакция соматического параметра (например, мембранный потенциал), которая инкапсулирует микроархитектурный механизм обращения к параметрам нейрона (мембранному потенциалу, параметру адаптивного порога и др.);

— транзакция входящего спайка, которая инкапсулирует микроархитектурные механизмы буферизации и извлечения спайка из входящей очереди;

— транзакция исходящего спайка, которая инкапсулирует микроархитектурный механизм эмиссии выходного спайка.

Обработка транзакций осуществляется в ходе вычислительного процесса (см. рис. 1), фазы которого обеспечивают согласованную работу микроархитектурных компонентов системы, в частности механизмов квитирования, конвейерный и пространственный параллелизм вычислений. Язык описания позволяет реализовать следующие фазы:

— синаптическую фазу, выполняемую между событиями тика; во время этой фазы происходит обработка входящих спайков, извлечение синаптических параметров и выполнение синаптических операций;

— соматическую фазу, выполняемую единожды по событию тика; во время этой фазы выполняются соматические операции и эмиссия спайков в исходящую очередь.

Для программной реализации описанных транзакций и фаз используется язык промежуточного описания, основанный на классах. Диаграмма классов для моделирования нейроморфной микроархитектуры представлена на рис. 4.

На основе приведенных классов можно реализовать описание нейроморфного процессора с отделением вычислений от управления потоком транзакций. Описание процессора состоит из следующих основных этапов.

1. Определение входящих и исходящих очередей.
2. Определение синаптических и соматических транзакций.
3. Определение события синхронизационного тика.
4. Определение фаз обработки транзакций.
5. Разработка дизайна пользовательской микроархитектуры на основе транзакций в рамках определенных фаз.

При определении объектов классов производится их параметризация, которая будет отображена в целевом синтезируемом дизайне. Этап разработки пользовательского дизайна обеспечивает

гибкий подход к описанию микроархитектуры, при этом освобождая разработчика от детального описания множественных обращений к локальным банкам локальной памяти, управления выборкой нейронов для обработки, синхронизации фаз и параллелизации вычислений.

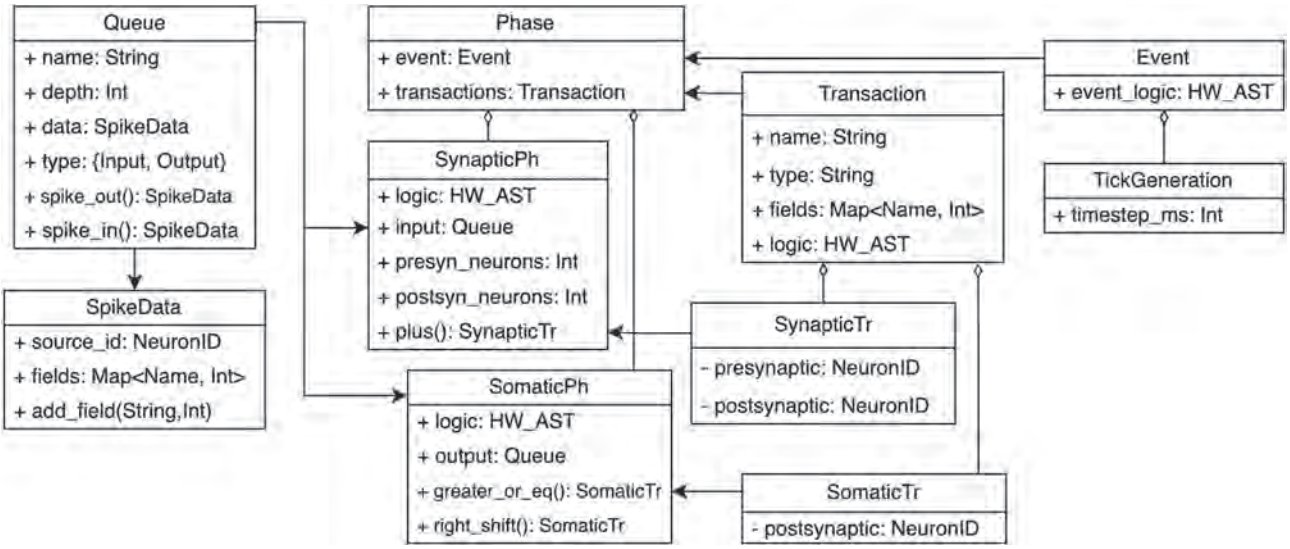

Рис. 4

Для разработки дизайна пользовательской микроархитектуры доступны арифметические и логические операции для синаптической и соматической фаз. Данные операции реализуют абстрактное синтаксическое дерево, которое преобразуется в конвейерные аппаратные структуры.

Автоматизированный синтез нейроморфных процессоров на основе транзакционного описания микроархитектуры. Для синтеза нейроморфных микроархитектур на основе транзакционного описания используется базовая микроархитектурная модель, в соответствии с которой пользовательские описания обработки транзакций интегрируются в инфраструктуру динамического планирования транзакций и контроля потоков. В рамках модели определен набор типичных микроархитектурных блоков и механизмов нейроморфных процессоров, что позволяет обеспечить их согласованность, повысить переиспользуемость кода и упростить разработку новых дизайнов. Базовая микроархитектурная модель нейроморфного процессора представлена на рис. 5.

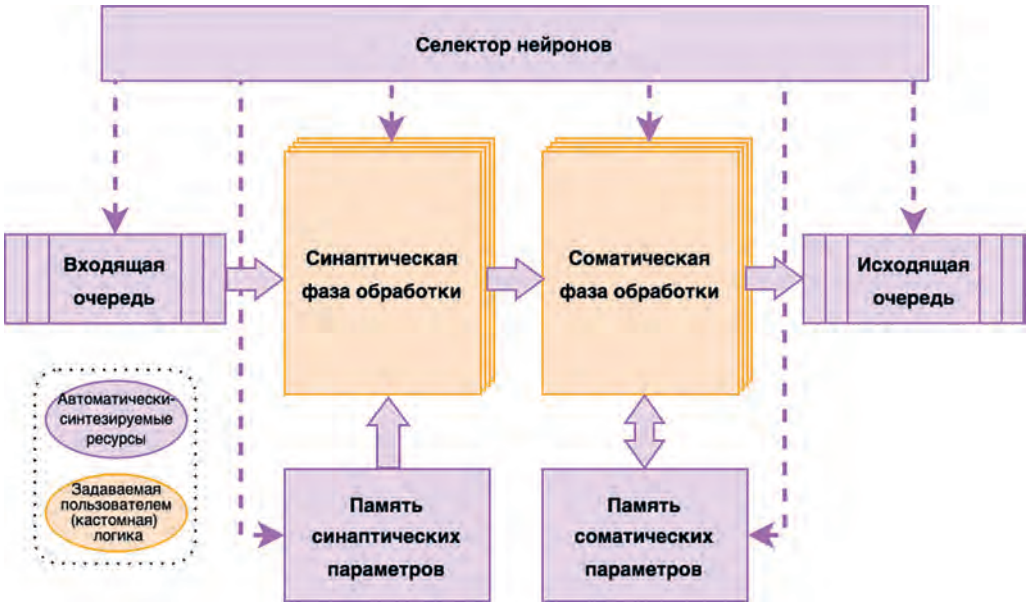

Рис. 5

Генерация HDL-кода — кода на языке описания аппаратуры (Hardware Discription Language — HDL) производится трансляцией транзакционного описания на уровень RTL. Она основывается на базовой микроархитектурной модели и генерирует аппаратные подсистемы, представленные в табл. 3.

Таблица 3

Подсистема	Описание
Входящая и исходящая очереди спайков	Очереди для буферизации спайков в рамках временного слота. Параметризуется форматом спайка и глубиной буфера. При генерации реализуется логика контроллера для операций чтения/записи спайков на основе механизма с кредитным счетчиком
Память синаптических параметров	Параметризируемый банк памяти, хранящий параметры, заданные полями синаптических транзакций. При генерации реализуется логика доступа к параметрам произвольных синапсов пре- и постсинаптических нейронов
Память мембранных потенциалов	Параметризируемый банк памяти, хранящий мембранные параметры нейронов, задаваемые полями соматических транзакций. При генерации реализуется логика доступа к мембранным параметрам постсинаптических нейронов
Селектор нейронов	Блок контроллера, определяющий идентификатор пре- и постсинаптических нейронов на различных фазах обработки. Обеспечивает синхронизацию конвейерных стадий обработки слоев сети и временное мультиплексирование по сигналу тика
Обработчик синаптической фазы	Набор блоков обработки синаптических операций заданного пользователем транзакционного описания. При генерации задается количество данных блоков с реализацией пространственного параллелизма вычислений. Синхронизация вычислений данной фазой обеспечивается блоком селектора нейронов
Обработчик соматической фазы	Набор блоков обработки соматических операций заданного пользователем транзакционного описания. При генерации определяется количество данных блоков с реализацией пространственного параллелизма вычислений. Синхронизация вычислений данной фазой обеспечивается блоком селектора нейронов

Для реализации микроархитектурной модели используется оригинальный фреймворк ActiveCore программно-управляемого конструирования аппаратуры (Hardware Construction/Generation Framework) [9] и ряд предложенных ранее классов для шаблонной генерации некоторых подсистем нейроморфных процессоров [10]. В частности, данный фреймворк включает модуль Hwast, реализующий аппаратно-ориентированное абстрактное синтаксическое дерево (AST) и упрощающий разработку различных типов данных, операций и логики управления. Это обеспечивает возможность отделения автоматически генерируемой инфраструктуры (коммуникация, управление потоком) от пользовательской логики. Пакет Hwast предоставляет базовые классы для создания контейнеров с процедурами, которые затем транслируются в синтезируемый код на уровне регистровых передач (RTL).

Результаты экспериментального проектирования. На основе представленного метода было разработано экспериментальное ядро нейроморфного процессора, реализующего обработку полносвязных импульсных нейронных сетей [11]. Разработанный прототип моделирует последовательную обработку импульсных Leaky Integrate-and-Fire (LIF) нейронов с временем таймслота, равным 1 мс. Размерность слова статической локальной памяти для хранения синаптических весов равна 4 бит на одну связь, размерность динамической памяти мембранных потенциалов равна 5 бит на нейрон. Для моделирования импульсной нейронной сети задаются параметры тока утечки, порогового мембранного потенциала и напряжения сброса. Ядро поддерживает конвейерную обработку двух слоев импульсной сети с 1024 нейронами в каждом слое. Для буферизации спайков используется очередь с кредитным счетчиком. Синаптические операции выполняются параллельно для групп постсинаптических нейронов с последовательной выборкой входящих спайков от пресинаптических нейронов. Соматические операции выполняются последовательно, после чего производится сравнение с пороговым потенциалом и запись исходящих спайков в выходную очередь. Пример исходного кода для описания экспериментального ядра на транзакционном уровне представлен на рис. 6.

```
class LIF(name: String) : Neuromorphic(name, SNN) {
    val synaptic = synaptic_phase(SNN.presyn_neurons, SNN.postsyn_neurons)
    val synaptic_transaction = SynapticTr("synapse")
    val weight = synaptic_transaction.add_field("weight", SNN.weightWidth)

    val soma = somatic_phase(SNN.postsyn_neurons)
    var somatic_transaction = SomaticTr("soma")
    val membrane_potential = somatic_transaction.add_field(
        "membrane_potential", SNN.potentialWidth
        )

    init {
        synaptic.plus(membrane_potential, weight, SNN.nnType)
        soma.srl(membrane_potential, SNN.leak)

        soma.begif(membrane_potential.geq(SNN.threshold))
        run {
            soma.single_spike()
            membrane_potential.assign(SNN.reset)
        }
        soma.endif()
    }
}
```

Рис. 6

Симуляция сгенерированного аппаратного дизайна проводилась в среде Xilinx Vivado с синтезом для отладочной платы ПЛИС Digilent Nexys A7 (Nexys 4 DDR) при частоте 100 МГц. Сравнение ресурсов, занимаемых экспериментальным ядром нейроморфного процессора, приведено в табл. 4.

Таблица 4

Источник	Аппаратные ресурсы ПЛИС, количество	
	LUT	FF
[12]	11 489	4705
[13]	5381	7309
Настоящая статья	12 000	7248

Для оценки объема исходных кодов (Source Lines of Code — SLOC) были выбраны нейроморфные процессоры с аналогичными функциональными требованиями и описанные на языке Verilog/SystemVerilog. Подсчет строк исходных кодов был произведен на основе исходных файлов функциональных ядер данных процессоров, оценка приведена в табл. 5. Оценка количества исходных кодов для настоящей работы производилась с учетом используемых классов и методов для описанного экспериментального ядра нейроморфного процессора.

Таблица 5

Источник	SLOK, количество строк кода	
	Переиспользуемый код	Общее количество
[14]	—	~1100
[15]	—	~1500
Настоящая статья	~1350	~1500

Заключение. Предложен оригинальный транзакционный метод проектирования нейроморфных процессоров. Метод основан на явном отделении порядка обработки моделей импульсных нейронов от механизмов динамического планирования и контроля потока конвейерных транзакций. Предложено формальное определение транзакции, приведены конструкции языка транзакционного описания нейроморфного процессора, а также представлен метод автоматизированного синтеза нейроморфных процессоров на основе транзакционных описаний с применением базовой микроархитектурной модели. Экспериментальное проектирование, выполненное с использованием предложенных решений, продемонстрировало возможность

гибкого проектирования ядра нейроморфного процессора в рамках базовой микроархитектурной модели. Оценка задействованных ресурсов ПЛИС показала соизмеримое расходование блоков LUT и FF, что, в свою очередь, занимает 35 % от всех ресурсов ПЛИС Nexys 4 DDR при высокой переиспользуемости кода, равной ~90 %.

СПИСОК ЛИТЕРАТУРЫ

1. *Schuman C. D., Kulkarni S. R., Parsa M., Mitchell J. P., Date P., Kay B.* Opportunities for neuromorphic computing algorithms and applications // Nature Computational Science. 2022. Vol. 2(1). P. 10–19. DOI: 10.1038/s43588-021-00184-y.
2. *Kudithipudi D., Schuman C., Vineyard C. M., Pandit T., Merkel C., Kubendran R., Aimone J. B., Orchard G., Mayr C., Benosman R., Hays J.* Neuromorphic computing at scale // Nature. 2025. Vol. 637(8047). P. 801–812. DOI: 10.1038/s41586-024-08253-8.
3. *Kiselev M., Ivanitsky A., Larionov D.* A purely spiking approach to reinforcement learning // Cognitive Systems Research. 2025. Vol. 89. P. 101317. DOI: 10.1016/j.cogsys.2024.101317.
4. *Su Q., Mei S., Xing X., Yao M., Zhang J., Xu B., Li G.* SNN-BERT: Training-efficient Spiking Neural Networks for energy-efficient BERT // Neural Networks. 2024. Vol. 180. P. 106630. DOI: 10.1016/j.neunet.2024.106630.
5. *Frenkel C., Bol D., Indiveri G.* Bottom-up and top-down approaches for the design of neuromorphic processing systems: Tradeoffs and synergies between natural and artificial intelligence // Proc. of the IEEE. 2023. Vol. 111(6). P. 623–652. DOI: 10.1109/JPROC.2023.3273520.
6. *Coussy P., Gajski D., Meredith M., Takach A.* An Introduction to High-Level Synthesis // IEEE Design & Test of Computers. 2009. Vol. 26(4). P. 8–17. DOI: 10.1109/MDT.2009.69.
7. *Bourgeat T., Pit-Claudel C., Chlipala A., Arvind*. The essence of Bluespec: a core language for rule-based hardware design // Proc. of the 41st ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI 2020). 2020. P. 243–257. DOI: 10.1145/3385412.3385965.
8. *Hoover S., Salman A.* Top-Down Transaction-Level Design with TL-Verilog. 2018 [Электронный ресурс]: https://doi.org/10.48550/arXiv.1811.01780, 02.03.2025.
9. *Antonov A.* Structured Design of Complex Hardware Microarchitectures Based on Explicit Generic Implementations of Custom Microarchitectural Mechanisms // Electronics. 2022. Vol. 11(7). P. 1055. DOI: 10.3390/electronics11071055.
10. *Lukashov I., Antonov A., Tabunschik S.* High-level design of neuromorphic processors based on explicit decoupling of computations and transaction flow control // Proc. of SPIE. 2024. Vol. 13239. P. 132391K. DOI: 10.1117/12.3036460.
11. Neuromorphix (GitHub) [Электронный ресурс]: https://github.com/ivlukashov/activecore/tree/master/kernelip/neuromorphix/src, 02.03.2025.
12. *Ma D., Shen J., Gu Z., Zhang M., Zhu X., Xu X., Xu Q., Shen Y., Pan G*. Darwin: A neuromorphic hardware co-processor based on spiking neural networks // J. of Systems Architecture. 2017. Vol. 77. P. 43–51.
13. *Han J., Li Z., Zheng W., Zhang Y*. Hardware implementation of spiking neural networks on FPGA // Tsinghua Science and Technology. 2020. Vol. 25(4). P. 479–486. DOI: 10.26599/TST.2019.9010019.
14. OpenSpike (GitHub) [Электронный ресурс]: https://github.com/sfmth/OpenSpike, 02.03.2025.
15. *Frenkel C., Lefebvre M., Legat J.-D., Bol D.* A 0.086-mm 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-nm CMOS // IEEE Trans. on Biomedical Circuits and Systems. 2019. Vol. 13(1). P. 145–158.

СВЕДЕНИЯ ОБ АВТОРАХ

Иван Викторович Лукашов — аспирант; Университет ИТМО, факультет программной инженерии и компьютерной техники; мл. научный сотрудник; E-mail: lukashov@itmo.ru

Александр Александрович Антонов — канд. техн. наук; Университет ИТМО, факультет программной инженерии и компьютерной техники; доцент; E-mail: antonov@itmo.ru

Павел Валерьевич Кустарев — канд. техн. наук; Университет ИТМО, факультет программной инженерии и компьютерной техники; доцент; E-mail: kustarev@itmo.ru

Поступила в редакцию 25.07.24; одобрена после рецензирования 30.07.24; принята к публикации 28.01.25.

REFERENCES

1. Schuman C.D., Kulkarni S.R., Parsa M., Mitchell J.P., Date P., and Kay B. *Nature Computational Science*, 2022, no. 1(2), pp. 10–19, DOI: 10.1038/s43588-021-00184-y.
2. Kudithipudi D., Schuman C., Vineyard C.M., Pandit T., Merkel C., Kubendran R., Aimone J.B., Orchard G., Mayr C., Benosman R., and Hays J. *Nature*, 2025, no. 637(8047), pp. 801–812, DOI: 10.1038/s41586-024-08253-8.
3. Kiselev M., Ivanitsky A., and Larionov D. *Cognitive Systems Research*, 2025, vol. 89, p. 101317, DOI: 10.1016/j.cogsys.2024.101317.
4. Su Q., Mei S., Xing X., Yao M., Zhang J., Xu B., and Li G. *Neural Networks*, 2024, vol. 180, p. 106630, DOI: 10.1016/j.neunet.2024.106630.
5. Frenkel C., Bol D., and Indiveri G. *Proceedings of the IEEE*, 2023, no. 6(111), pp. 623–652, DOI: 10.1109/JPROC.2023.3273520.
6. Coussy P., Gajski D., Meredith M., Takach A. *IEEE Design & Test of Computers*, 2009, no. 4(26), pp. 8–17, DOI: 10.1109/MDT.2009.69.
7. Bourgeat T., Pit-Claudel C., Chlipala A.A. *Proc. of the 41st ACM SIGPLAN Conf. on Programming Language Design and Implementation* (PLDI 2020), 2020, pp. 243–257, DOI: 10.1145/3385412.3385965.
8. Hoover S., Salman A. *Top-Down Transaction-Level Design with TL-Verilog*, 2018, https://doi.org/10.48550/arXiv.1811.01780.
9. Antonov A. *Electronics 2022*, 2022, no. 7(11), pp. 1055, DOI: 10.3390/electronics11071055.
10. Lukashov I., Antonov A., Tabunschik S. *Proceedings of SPIE*, 2024, vol. 13239, pp. 132391K, DOI: 10.1117/12.3036460.
11. Neuromorphix (GitHub), https://github.com/ivlukashov/activecore/tree/master/kernelip/neuromorphix/src.
12. Ma D., Shen J., Gu Z., Zhang M., Zhu X., Xu X., Xu Q., Shen Y., Pan G. *Journal of Systems Architecture*, 2017, vol. 77, pp. 43–51.
13. Han J., Li Z., Zheng W. and Zhang Y. *Tsinghua Science and Technology*, 2020, no. 4(25), pp. 479–486. DOI: 10.26599/TST.2019.9010019.
14. OpenSpike (GitHub), https://github.com/sfmth/OpenSpike.
15. Frenkel C., Lefebvre M., Legat J.-D., Bol D. *IEEE Transactions on Biomedical Circuits and Systems*, 2019, no. 1(13), pp. 145–158.

DATA ON AUTHORS

Ivan V. Lukashov — Post-Graduate Student; ITMO University, Faculty of Software Engineering and Computer Science; Junior Researcher; E-mail: lukashov@itmo.ru

Alexander A. Antonov — PhD; ITMO University, Faculty of Software Engineering and Computer Science; Associate Professor; E-mail: antonov@itmo.ru

Pavel V. Kustarev — PhD; ITMO University, Faculty of Software Engineering and Computer Science; Associate Professor; E-mail: kustarev@itmo.ru

Received 25.07.24; approved after reviewing 30.07.24; accepted for publication 28.01.25.