Разработка алгоритма настройки перестраиваемой вычислительной среды в составе аппаратного ускорителя искусственных нейронных сетей

В. Шатравин, Д.В. Шашев
Томский государственный университет
e-mail: shatravin@stud.tsu.ru ; dshashev@mail.tsu.ru

Аннотация
Возрастающая вычислительная сложность искусственных нейронных сетей поднимает важные вопросы производительности, гибкости и энергоэффективности используемого для их расчёта аппаратного обеспечения. Характеристики вычислительных устройств являются одним из основных факторов, ограничивающих применение сложных алгоритмов машинного обучения в мобильных и маломощных интеллектуальных системах. Проблеме разработки высокопроизводительных и энергоэффективных аппаратных ускорителей нейронных сетей на сегодняшний день уделяется большое внимание и предлагается множество различных решений. Одним из таких решений являются вычислители на основе перестраиваемых вычислительных сред, чьё устройство представляет собой сетку из однотипных настраиваемых вычислительных элементов. Важным аспектом их применения является необходимость настройки элементов для совместного решения сложных задач. В данной работе рассматривается алгоритм координатной настройки перестраиваемых вычислительных сред для применения в составе аппаратных ускорителей искусственных нейронных сетей. Описан процесс настройки сегментированной вычислительной среды, обеспечивающей высокое быстродействие ускорителя благодаря плавающему окну настройки и применению конвейеризации. Приведены результаты оценки длительности настройки сегментов разного размера на основе временных симуляций моделей среды на программируемых логических интегральных схемах (FPGA). Полученные результаты показывают высокое быстродействие предложенного алгоритма настройки. Полная настройка слоя из пятидесяти 15-входовых нейронов занимает 77.5 нс.

Ключевые слова
Искусственные нейронные сети, перестраиваемые вычислительные среды, реконфигурируемые аппаратные ускорители.

Нейронные сети являются одним из наиболее перспективных направлений развития сложных информационных систем. Их способность эффективно решать плохо формализованные задачи активно используется в самых разных областях человеческой деятельности: в поисковых системах, системах рекомендаций, анализа изображений, обработки естественного языка, предсказательной аналитики и многих других.

Однако решение сложных задач требует применения глубоких сетей с большим количеством слоёв и параметров, что приводит к необходимости разработки специализированных вычислительных устройств, учитывающих особенности алгоритмов машинного обучения. Такие устройства называют аппаратными ускорителями нейронных сетей. Аппаратные ускорители опираются на возможность распараллеливания вычислений, низкоуровневую аппаратную реализацию операций и оптимизацию работы с памятью. На сегодняшний день предлагается множество различный реализаций ускорителей на основе графических процессоров (GPU), программируемых логических интегральных схем (FPGA) и интегральных схем специального назначения (ASIC) [1.–3.]. Они показывают высокое быстродействие и широко применяются в системах разного назначения и масштаба.

Ключевым недостатком большинства предлагаемых ускорителей нейронных сетей является их узкая нацеленность на конкретную архитектуру сети и небольшой ограниченный набор поддерживаемых функций активации нейрона. В то же время информационные и технические системы усложняются, расширяется спектр решаемых ими задач, что приводит их к необходимости применения не одной, а сразу нескольких различных нейронных сетей с отличающимися архитектурой и параметрами. Особенно остро проблема стоит для маломощных мобильных и автономных устройств, предъявляющих дополнительные требования к характеристикам своих подсистем. Очевидно, что применение нескольких различных ускорителей в рамках одной системы может быть нежелательным с точки зрения надёжности, энергопотребления, массы и габаритов.

Одним из возможных решений этой проблемы является применение аппаратных ускорителей, поддерживающих произвольное количество моделей и архитектур сетей благодаря способности к реконфигурируемости. Такие устройства могут быть реализованы на основе перестраиваемых вычислительных сред (ПВС), поддерживающих тонкую динамическую настройку и обеспечивающих решение сложных задач благодаря совместному функционированию коллектива простых вычислителей.

В данной работе описывается алгоритм настройки ПВС для применения в основе динамически перестраиваемого аппаратного ускорителя нейронных сетей.

Перестраиваемые вычислительный среды

Перестраиваемая вычислительная среда (ПВС) – математическая модель вычислительных систем, имеющих структуру геометрически правильной решётки, узлами которой являются простые однотипные вычислительные элементы (рис. 1). В некоторых источниках ПВС также упоминаются как однородные среды [4.]. Вычислительные элементы (ВЭ) соединены друг с другом двусторонними связями. Каждый ВЭ может быть индивидуально настроен на выполнение простой операции из заранее заданного базиса, благодаря чему вся среда может реализовывать сложные алгоритмы потоковой обработки данных и допускает динамическое изменение этих алгоритмов. Таким образом, на среде достаточного размера с соответствующим базисом операций можно реализовать любой произвольный алгоритм [4., 5.]. ПВС относится к классу вычислительных устройств параллельно-конвейерного типа.

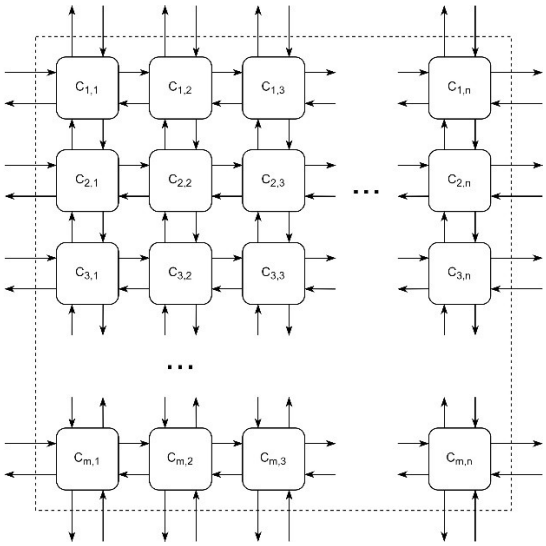

Рис. 1. Перестраиваемая вычислительная среда

Перестраиваемые вычислительные устройства занимают промежуточное положение между специализированными и универсальными [6.]. Перестраиваемость позволяет расширить класс решаемых системой задач по сравнению с узкоспециализированными системами. В то же время, каждая отдельная задача может быть решена перестраиваемой системой эффективнее, чем универсальной, благодаря специальной настройке. Иными словами, перестраиваемый аппаратный ускоритель может выполнять значительно различающиеся алгоритмы машинного обучения с большей (по сравнению с универсальными вычислительными устройствами) эффективностью с точки зрения быстродействия и энергопотребления. Помимо этого, способность ПВС изменять реализуемые алгоритмы даёт и другие преимущества:

1. Поздняя удалённая инициализация устройства. Устройство можно настроить на реализацию требуемых алгоритмов уже после развёртывания в рабочем окружении и определения требуемых параметров. Это необходимо в случаях, когда условия функционирования не могут быть определены на этапе разработки системы.

2. Модификация реализуемых алгоритмов играет важную роль для автономных и мобильных устройств, долгое время функционирующих в изменчивой среде, а также в случаях, когда список поставленных перед системой задач может быть со временем изменён.

3. Поддержка нескольких нейронных сетей для разных режимов функционирования. К примеру, применение менее точной, но более энергоэффективной сети в режиме ожидания с переключением на более сложную сеть в основном рабочем режиме.

4. Изотропность структуры и способность к перестраиваемости позволяют ПВС до определённой степени восстановить работоспособность посредством перераспределения вычислений по неповреждённым участкам среды.

Аппаратный ускоритель на основе перестраиваемых сред

Так как ПВС описывает лишь общие принципы построения и функционирования вычислителя, существует множество способов реализации конечных вычислительных устройств. В данной работе рассматривается реализация ПВС основанная на сильной декомпозиции нейросетевых алгоритмов, описанная в работах [7.–8.]. Эта реализация может использоваться для построения сетей прямого распространения, свёрточных и рекуррентных. Кратко рассмотрим основные принципы представленной архитектуры вычислительного устройства.

Каждый ВЭ реализует одну из нескольких простых переиспользуемых операций, среди которых умножение с накоплением, расчёт функции активации ReLU, передача сигнала, однотактная задержка и некоторые другие. Объединяя несколько элементов в цепочку определённой длины, можно получить искусственный нейрон с требуемым количеством входов. Группируя несколько таких цепочек вместе, можно получить полносвязный или свёрточный слой сети (рис. 2).

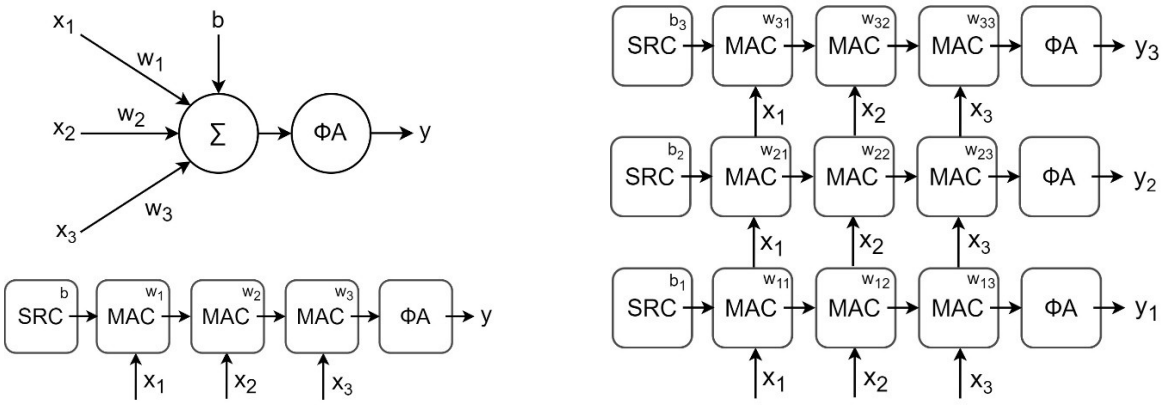

Рис. 2. Нейрон из элементов среды (слева) и полносвязный слой из трёх нейронов (справа)

В предложенной архитектуре каждый ВЭ настраивается не только на операцию, но и на одно из четырёх направлений. Это связано с тем, что каждый ВЭ соединён с четырьмя соседями. Направленность не меняет внутреннюю структуру ВЭ, она лишь определяет какие входы и выходы он будет использовать при выполнении заданной операции. Каждой направленности соответствует один основной выход (на который будет поступать результат текущей операции) и три неосновных (выводят результаты с противоположных им входов элемента).

Некоторым операциям также требуется дополнительный аргумент. К примеру, элементу, настроенному на операцию умножения с накоплением, на входы поступают накопленное значение (аккумулятор) и только один из множителей. Второй множитель хранится во внутренней памяти самого элемента и устанавливается в процессе настройки. При построении нейрона этот аргумент соответствует весу связи.

Таким образом, длина настроечного сигнала ВЭ, оперирующего k-разрядными двоичными числами, может быть выражена:

$$c = k + op + dir = k + 6, \tag{1}$$

где c – разрядность настроечного сигнала, бит,
op – длина кода операции (4 бита),
dir – длина кода направления (2 бита).

Режимы функционирования среды

Применяемые на практике нейронные сети сильно различаются как количеством скрытых слоёв, так и их размерами. В связи с этим, для рассматриваемой ПВС предлагаются два режима функционирования: интегральный и сегментированный [7., 8.].

В интегральном режиме вся среда реализует один слой сети, что позволяет вычислять особо крупные слои, но приводит к низкой утилизации ресурсов, большому потоку внешних данных и долгому ожиданию полной перенастройки после каждого такта (рис. 3).

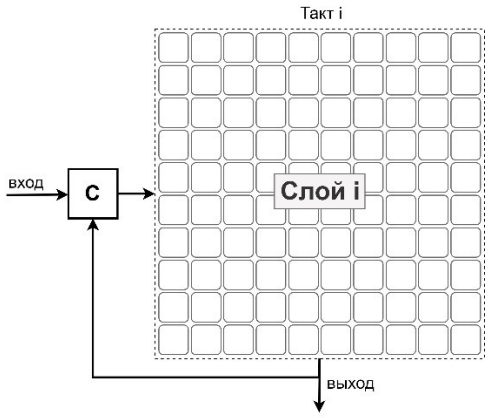

Рис. 3. Интегральный режим функционирования среды

Для устранения этих недостатков предлагается второй режим – сегментированный. В этом режиме среда разбивается на несколько сегментов произвольного размера, каждый из которых реализует один слой сети. Входной сигнал последовательно перемещается между этими сегментами, проходя соответствующую обработку. По мере перемещения сигнала, сегменты перестраиваются на реализацию последующих слоёв сети, что позволяет выполнить полный расчёт всех слоёв без вывода из среды промежуточных результатов. При помощи плавающего окна настройки возможно реализовать на среде конвейерную обработку сигнала, при которой среда из n сегментов сможет обрабатывать $n-1$ сигнал одновременно, пока n-ый сегмент проходит перенастройку (рис. 4). Эти особенности сегментированного режима позволяют улучшить утилизацию ресурса среды, а также повысить быстродействие благодаря применению конвейеризации и плавающего окна настройки.

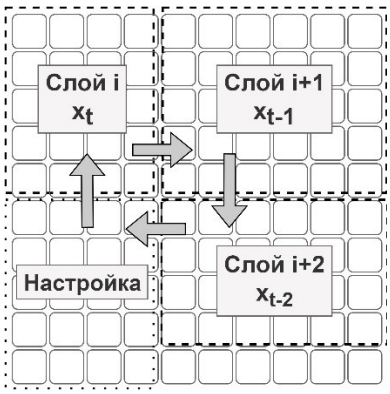

Рис. 4. Сегментированный режим функционирования среды

Основным недостатком сегментированного режима является меньший, по сравнению с интегральным режимом, допустимый размер слоя. Реализация небольшой полносвязной сети на сегментированной среде приведена на рисунке 5.

Настройка среды

Выделяют три основных способа настройки элементов ПВС: непосредственный, адресный и координатный [5.]. При непосредственной настройке каждый вычислительный элемент имеет собственные внешние выводы, через которые на него подаётся настроечный сигнал. Адресный способ основан на применении общей для всех ВЭ настроечной шины, в которую отсылаются коды настройки вместе с адресом ВЭ, для которого эта настройка предназначена. Для координатной настройки используется настроечная сетка, в которой каждый ВЭ имеет уникальные координаты,

посредством которых осуществляется его индивидуальная настройка. В данной работе рассматривается координатный способ настройки при помощи двумерной сетки настроечных каналов (рис. 6).

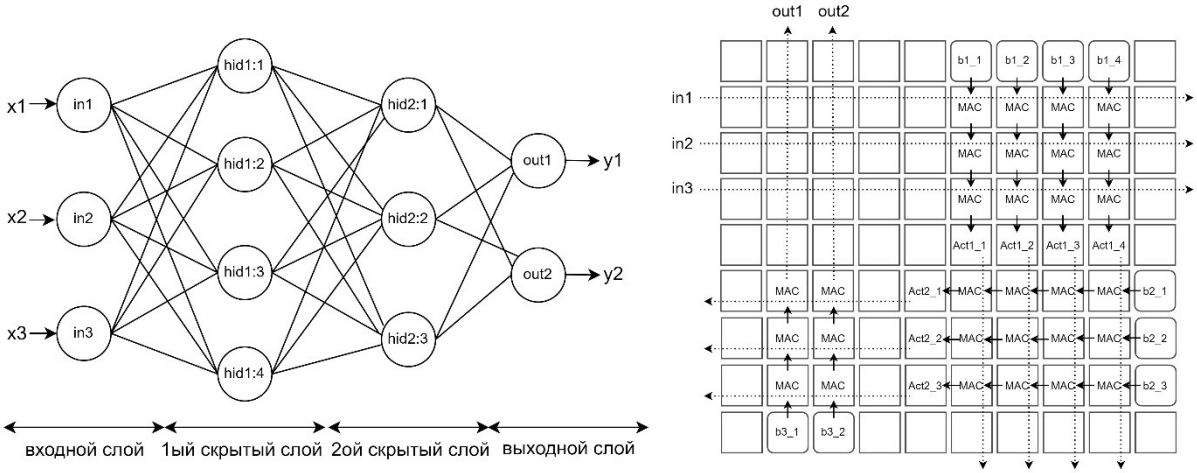

Рис. 5. Реализация полносвязной нейронной сети на сегментированной среде

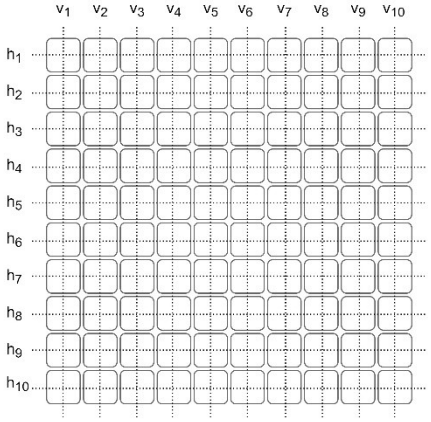

Рис. 6. Настроечная сетка среды

Настроечная сетка состоит из непересекающихся вертикальных (v_i) и горизонтальных (h_i) настроечных каналов. Горизонтальные каналы имеют разрядность один бит, а вертикальные – два. В точках, показанных на рисунке 6 как пересечения прямых, располагаются ВЭ. Каждый ВЭ соединён с одним горизонтальным и одним вертикальным каналом, то есть координаты элемента можно записать как (h_m, v_n).

Настроечная сетка служит для перевода элементов в режим настройки. Для перехода ВЭ в этот режим требуется, чтобы у обоих его настроечных каналов были установлены в единицу первые биты. При переходе в режим настройки ВЭ очищает внутреннюю память, хранящую его текущую настройку, что равносильно переключению элемента на операцию «передача сигнала», имеющей двоичный код «0000». Настроенные на эту операцию элементы выполняют канальную функцию – передают входной сигнал сквозь себя без изменения. Это позволяет осуществлять передачу кода настройки по основным сигнальным связям между ВЭ, что значительно упрощает структуру соединений в ПВС. Когда хотя бы один из настроечных каналов вернётся в состояние «0», ВЭ выйдет из режима настройки, сохранив при этом значения со своих сигнальных входов во внутреннюю память в виде настройки.

Однако, как видно из (1), длина кода настройки c превышает разрядность связей k между элементами среды. Поэтому код настройки разбивается на две составляющие – код аргумента (k бит) и объединённый код настройки и направления (6 бит). Эти составляющие передаются раздельно по вертикальным и горизонтальным связям ВЭ.

Рассмотрим процесс настройки сегмента среды. Пусть требуется настроить нижний правый участок среды. Обозначим индекс левого края сегмента как l, правого r, верхнего t, нижнего b (рис. 7).

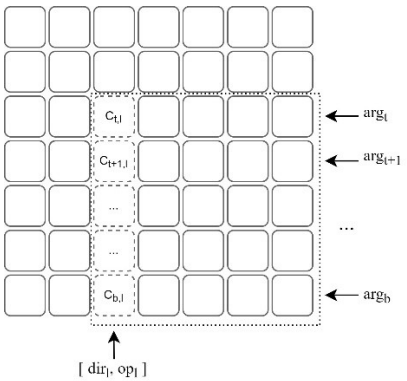

Рис. 7. Передача кодов настройки на столбец настраиваемого сегмента

Тогда алгоритм настройки можно записать в виде последовательности шагов:

1. Установить в единицу первый бит вертикальных каналов от v_l до v_r включительно, а также горизонтальные каналы от h_t до h_b. После этого, как было сказано ранее, лежащие в этой области вычислительные элементы очищают свою текущую настройку и переходят в канальный режим. Если коды операций и направлений будут передаваться по горизонтальным связям, то дополнительно требуется установить в «1» второй бит соответствующих вертикальных каналов. Далее рассмотрим сценарий, когда операция и направление передаются по вертикальным связям, и соответствующий бит настроечного канала равен нулю.

2. Выбирается текущий настраиваемый столбец ВЭ из области настройки как наиболее удалённый от края среды. Согласно текущему примеру это столбец, соответствующий каналу v_l.

3. По строкам вычислительных элементов с индексами от t до b начинают передаваться соответствующие настраиваемому столбцу коды аргументов. В это же время по элементам столбца l снизу вверх передаются коды команды и направления. Обозначим время, за которое коды аргументов достигнут настраиваемого столбца как th^l, а время передачи операции и направления до верхнего настраиваемого элемента как tv_l. Тогда полное время передачи сигнала tf_l составляет:

$$tf_l = max\left(tv_l, th^l\right), \tag{2}$$

$$th^l = max\left(th_t^l, th_{t+1}^l, \ldots, th_b^l\right). \tag{3}$$

4. Настроечный канал-столбец v_l устанавливается в «0», что приводит к выходу элементов соответствующего столбца среды из режима настройки. При этом каждый ВЭ объединяет сигналы на своих горизонтальных и вертикальных входах в единую настройку, которую сохраняет во внутренней памяти. После этого столбец l считается настроенным. Так как память настройки невелика и расположена в самом ВЭ, время сохранения настроечного кода пренебрежимо мало.

5. Шаги 2–4 повторяются для оставшихся $r-l$ столбцов области. Тогда полное время настройки сегмента ts составляет:

$$ts = \sum_{i=l}^{r} tf_i. \tag{4}$$

Описанный сценарий является наиболее распространённым, так как основную долю вычислений в нейронных сетях классических архитектур занимают полносвязные и свёрточные слои, которые при реализации на среде будут представлены в виде блока элементов, имеющих одинаковую операцию и направленность, но с разными аргументами-весами (рис. 2).

Если ширина настраиваемой области меньше высоты, то целесообразным может быть передача кодов операции и направления сразу для всех столбцов области, что позволит переписать (2) для $i>l$ как:

$$tf_i = th^i. \tag{5}$$

Очевидно, что наиболее быстрой будет настройка сегмента, все элементы которого имеют полностью совпадающий код настройки. В этом случае можно подавать настройку на все столбцы одновременно, а общее время настройки сегмента будет составлять:

$$ts \approx max(t f_i). \tag{6}$$

В то же время возможны случаи, когда у элементов настраиваемого столбца различаются не только аргументы, но и операция с направлением. Настройка таких столбцов предложенным методом возможна только поэлементно. Это означает, что на шаге 3 будет передаваться код операции и направления для наиболее удалённого от нижнего края ВЭ, который в первой итерации будет иметь координаты (t, l). Затем будет обнуляться горизонтальный настроечный канал t, после чего по столбцу начнёт передаваться код для нижележащего элемента $(t+1, l)$, и процесс будет повторяться для всех $(b-t)$ строк столбца. При этом время передачи кода аргумента повлияет только на время настройки первого (верхнего) элемента столбца:

$$t f_l = max\left(t v_l^t, t h_t^l\right) + \sum_{i=t+1}^{b} t v_l^i. \tag{7}$$

Можно видеть, что на скорость настройки сегмента большое влияние оказывает выбор осей передачи двух составляющих кода настройки. К примеру, настройка пяти строк из 25 элементов будет осуществляться гораздо быстрее, чем 25 столбцов из пяти элементов. Поэтому целесообразно выбирать оси таким образом, чтобы коды операции и направления передавались вдоль большей оси сегмента. Для этого был введён упомянутый ранее второй бит вертикальных настроечных каналов. Его значение определяет какие входные значения будут интерпретироваться элементом как код аргумента, а какие – как код операции и направления. Если этот бит установлен в единицу, то передача кодов настройки и направления должна осуществляться по горизонтальным связям, а кода аргумента – по вертикальным.

Определение временных характеристик алгоритма

Для анализа характеристик предложенного алгоритма была осуществлена симуляция описанной перестраиваемой вычислительной среды в программном пакете Quartus Prime 20.1 при помощи языка описания аппаратуры Verilog. Измерение временных характеристик проводилось во встроенном в Quartus инструменте Timing Analyzer Tool, который позволяет оценивать задержку распространения сигнала по заданной цифровой схеме. При симуляции оценивалось время передачи сигнала по цепочке элементов, функционирующих в канальном режиме.

В качестве целевого устройства симуляции была выбрана программируемая логическая интегральная схема (FPGA) Altera Cyclone V (5CGXFC9E7F35C8). Симуляция проводилась в режиме «Fast corners 1100mV 0°C». При симуляции рассматривались цепочки ВЭ разной длины. Используемые модели ВЭ оперируют 16-разрядными двоичными числами с фиксированной точкой. По результатам симуляции было определено, что удельная задержка передачи сигнала через один ВЭ составляет 0.5 нс.

Исходя из найденной длительности передачи сигнала 0.5 нс и согласно формулам (2), (3) и (4) было рассчитано время настройки сегментов разного размера. Расчёт производился для наиболее распространённого случая, когда сегмент состоит из элементов с одинаковыми операцией и направлением, но разными аргументами. Полученные результаты приведены в таблице 1. Можно видеть, что скорость настройки среды предложенным алгоритмом достаточно высока. Сегмент размером 50×75 (3750 ВЭ) будет настроен за 650 нс (0.173 нс/ВЭ). В то же время, для квадратных сегментов алгоритм показывает меньшую эффективность из-за невозможности использовать поворот осей.

Таблица 1. Длительность настройки сегментов среды разного размера

		Ширина сегмента, ВЭ				
		5	15	25	50	75
	5	7.5 нс	12.5 нс	17.5 нс	30 нс	42.5 нс

Высота сегмента, ВЭ	15	12.5 нс	60 нс	65 нс	77.5 нс	90 нс
	25	17.5 нс	65 нс	162.5 нс	175 нс	187.5 нс
	50	30 нс	77.5 нс	175 нс	637.5 нс	650 нс
	75	42.5 нс	90 нс	187.5 нс	650 нс	1425 нс

Заключение

Применение искусственных нейронных сетей позволяет значительно расширить функциональные возможности современных информационных и технических систем. Однако присущая им вычислительная сложность приводит к необходимости разработки специализированных вычислительных устройств – аппаратных ускорителей нейронных сетей. Применение динамически перестраиваемых аппаратных ускорителей на основе перестраиваемых вычислительных сред позволяет не только добиться высокого быстродействия и энергоэффективности, но и обеспечивает поддержку широкого многообразия алгоритмов в рамках одного вычислительного устройства. При этом важную роль играет процесс настройки среды на реализацию заданного алгоритма.

В данной работе предлагается алгоритм координатной настройки ПВС для применения в аппаратных ускорителях. В основе алгоритма лежит групповая настройка строк/столбцов среды и независимая передача составляющих кода настройки по разным осям. Переиспользование сигнальных соединений между вычислительными элементами для передачи кода настройки позволяет значительно упростить внутреннюю структуру среды. Механизм смены осей передачи составляющих настройки повышает эффективность настройки сегментов среды со значительно различающимися высотой и шириной.

При помощи временной симуляции рассматриваемой модели ПВС на FPGA и последующих расчётов была определена длительность настройки сегментов разных размеров. Полученные результаты демонстрируют относительно высокое быстродействие. К примеру, настройка на реализацию полносвязного слоя из пятидесяти 15-входовых нейронов требует 77.5 нс. В то же время, видна необходимость в дальнейших оптимизациях алгоритма, так как на данный момент длительность настройки сегмента превышает длительность вычисления результата на нём. Эта проблема отчасти решается применением сегментированного режима функционирования среды и конвейеризацией вычислительного процесса. Таким образом, предложенный в данной работе алгоритм настройки может быть успешно использован для динамически перестраиваемых аппаратных ускорителей на основе ПВС.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-37-90034.

СПИСОК ЛИТЕРАТУРЫ

1. Ghimire D., Kil, D., Kim, S.-h. A Survey on Efficient Convolutional Neural Networks and Hardware Acceleration // J. Electronics, MDPI. – 2022, 11. – vol. 945. – pp. 1-23.
2. Nabavinejad S. M., Reda S., Ebrahimi M. Coordinated Batching and DVFS for DNN Inference on GPU Accelerators // IEEE Transactions on Parallel and Distributed Systems. – 2022. – pp. 1-12.
3. Jouppi N. P. [et al.] In-Datacenter Performance Analysis of a Tensor Processing Unit // 44th Annual International Symposium on Computer Architecture (ISCA '17). – 2017. – pp. 1-12.
4. Евреинов Э.В. Однородные вычислительные системы, структуры и среды, М.: Радио и связь, 1981. – 208 с.
5. Шидловский С. В. Автоматическое управление. Перестраиваемые структуры / С. В. Шидловский. – Томск: ТГУ, 2006. – 288 с.
6. Каляев И. А. [и др.] Реконфигурируемые мультиконвейерные вычислительные структуры. Ростов н/Д : ЮНЦ РАН, 2008. – 393 с.
7. Shatravin V., Shashev D., Shidlovskiy S. Sigmoid Activation Implementation for Neural Networks Hardware Accelerators Based on Reconfigurable Computing Environments for Low-Power Intelligent Systems // MDPI: Applied Sciences. – 2022. – 12(10).
8. Shatravin V., Shashev D. V., Shidlovskiy S.V. Applying the Reconfigurable Computing Environment Concept to the Deep Neural Network Accelerators Development // International Conference on Information Technology (ICIT) – 2021. – pp. 842-845.