УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
июль–август 2020 Том 20 № 4 ISSN 2226-1494 http://ntv.itmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
July–August 2020 Vol. 20 No 4 ISSN 2226-1494 http://ntv.itmo.ru/en/
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК УНИВЕРСИТЕТ ИТМО ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

УДК 004.383 doi: 10.17586/2226-1494-2020-20-4-560-567

МОДЕЛЬ ИНСТРУМЕНТАЛЬНОГО СРЕДСТВА АВТОМАТИЗИРОВАННОГО СИНТЕЗА АППАРАТНЫХ УСКОРИТЕЛЕЙ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ

В.А. Егиазарян[a], С.В. Быковский[b]

[a] ООО «Сименс», Санкт-Петербург, 191186, Российская Федерация
[b] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
Адрес для переписки: aes3plex@ya.ru

Информация о статье
Поступила в редакцию 27.05.20, принята к печати 30.06.20
Язык статьи — русский

Ссылка для цитирования: Егиазарян В.А., Быковский С.В. Модель инструментального средства автоматизированного синтеза аппаратных ускорителей сверточных нейронных сетей для программируемых логических интегральных схем // Научно-технический вестник информационных технологий, механики и оптики. 2020. Т. 20. № 4. С. 560–567. doi: 10.17586/2226-1494-2020-20-4-560-567

Аннотация

В настоящее время все больше задач по обработке и анализу изображений решаются с использованием сверточных нейронных сетей. Сети, реализованные с использованием высокоуровневых языков программирования, библиотек и фреймворков невозможно использовать в системах реального времени, например, для обработки потокового видео в вычислительной системе автомобиля, из-за низкой скорости и энергоэффективности таких реализаций. Для подобных задач необходимо использовать специализированные аппаратные ускорители нейронных сетей. Проектирование таких ускорителей является сложным итеративным процессом, требующим узкоспециализированных знаний и квалификации. Это делает актуальным вопрос создания средств автоматизации высокоуровневого синтеза подобных вычислителей. Целью исследования стала разработка средства автоматизированного синтеза нейросетевых ускорителей из высокоуровневой спецификации, позволяющего снизить время разработки таких ускорителей для устройств программируемой логики. В качестве высокоуровневой спецификации используется описание сетей, которое можно получить с помощью фреймворка TensorFlow. Проведено исследование нескольких стратегий оптимизации структуры сверточных сетей, способов организации вычислительного процесса и форматов представления данных в нейронных сетях и их влияния на характеристики получаемого вычислителя. Показано, что оптимизация структуры полносвязных слоев нейронной сети на примере решения задачи распознавания рукописных цифр из набора MNIST сокращает количество параметров сети на 95 % с потерей точности около 0,43 %, конвейеризация вычислений ускоряет расчет в 1,7 раз, а благодаря распараллеливанию отдельных частей вычислительного процесса достигается ускорение почти в 20 раз, хотя на это и требуется в 4–6 раз больше ресурсов программируемых логических интегральных схем. Переход в вычислениях от чисел с плавающей точкой к числам с фиксированной точкой позволяет сократить используемые ресурсы в 1,7–2,8 раз. Проведен анализ полученных результатов, и предложена модель инструментального средства автоматизированного синтеза, которая позволяет выполнить обозначенные оптимизации с целью удовлетворения предъявляемых требований по быстродействию и используемым ресурсам при реализации нейросетевых ускорителей на микросхемах программируемых логических интегральных схем.

Ключевые слова

сверточные нейронные сети, нейросетевые ускорители, аппаратные ускорители, ПЛИС, САПР, высокоуровневый синтез

doi: 10.17586/2226-1494-2020-20-4-560-567

MODEL OF AUTOMATED SYNTHESIS TOOL FOR HARDWARE ACCELERATORS OF CONVOLUTIONAL NEURAL NETWORKS FOR PROGRAMMABLE LOGIC DEVICES

V.A. Egiazarian[a], S.V. Bykovskii[b]

[a] OOO Siemens, Saint Petersburg, 191186, Russian Federation
[b] ITMO University, Saint Petersburg, 197101, Russian Federation
Corresponding author: aes3plex@ya.ru

Article info
Received 27.05.20, accepted 30.06.20
Article in Russian

For citation: Egiazarian V.A., Bykovskii S.V. Model of automated synthesis tool for hardware accelerators of convolutional neural networks for programmable logic devices. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2020, vol. 20, no. 4, pp. 560–567 (in Russian). doi: 10.17586/2226-1494-2020-20-4-560-567

Abstract
Currently, more and more tasks on image processing and analysis are being solved using convolutional neural networks. Neural networks implemented using high-level programming languages, libraries and frameworks cannot be used in real-time systems, for example, for processing streaming video in cars, due to the low speed and energy efficiency of such implementations. The application of specialized hardware accelerators of neural networks is necessary for these tasks. The design of such accelerators is a complex iterative process requiring highly specialized knowledge and qualification. This consideration makes the creation of automation tools for high-level synthesis of such computers a relevant issue. The purpose of this research is a tool development for the automated synthesis of neural network accelerators from a high-level specification for programmable logic devices (FPGAs), which reduces the development time. A description of networks is used as a high-level specification, which can be obtained using the TensorFlow framework. The several strategies have been researched for optimizing the structure of convolutional networks, methods for organizing the computational process and formats for representing data in neural networks and their effect on the characteristics of the resulting computer. It was shown that structure optimization of neural network fully connected layers on the example of solving the handwritten digit recognition problem from the MNIST set reduces the number of network parameters by 95 % with a loss of accuracy equal to 0.43 %, pipelining of calculations speeds up the calculation by 1.7 times, and parallelization of the computing process individual parts provides the acceleration by almost 20 times, although it requires 4-6 times more FPGA resources. Applying of fixed-point numbers instead of floating-point numbers in calculations reduces the used FPGA resources by 1.7–2.8 times. The analysis of the obtained results is carried out and a model of an automated synthesis tool is proposed, which performs the indicated optimizations in automatic mode in order to meet the requirements for speed and resources used in the implementation of neural network accelerators on FPGA.

Keywords
convolutional neural networks, neural network accelerators, hardware accelerators, FPGA, CAD, high-level synthesis

Введение

Сегодня нейронные сети применяются во множестве отраслей, таких как транспорт, медицина, наука, энергетика и др. Они относительно просты в реализации, используя высокоуровневые языки программирования и десятки существующих на сегодняшний день библиотек и фреймворков, и могут быть организованы по-разному в зависимости от решаемой ими задачи. От выбора правильной архитектуры напрямую зависит скорость работы сети, ее точность и количество используемых ресурсов.

В работе исследован вопрос эффективной реализации сверточных нейронных сетей (СНС) в виде специализированных вычислителей, реализованных на базе программируемых логических интегральных схем (ПЛИС). СНС отличаются хорошей способностью к распознаванию; возможностью классификации объектов на изображениях; обеспечивают инвариантность к изменениям масштаба, поворотам, сдвигам и пространственным искажениям.

Использование специализированных аппаратных ускорителей (нейросетевых ускорителей) позволяет достичь скорости расчета выхода сети на порядки большей, чем с использованием процессоров общего назначения и графических процессоров, обеспечивая при этом низкое энергопотребление. Таким образом, становится возможным использование нейронных сетей для обработки потокового видео, аудио и другой информации, поступающей в реальном времени.Реализация нейронной сети на аппаратной платформе в виде специализированного вычислителя является сложным итеративным процессом, требующим узкоспециализированных знаний, что увеличивает стоимость разработки и препятствует повсеместному их использованию. В связи с этим становится актуальным создание средств автоматизации проектирования специализированных вычислителей и средств синтеза аппаратных ускорителей нейронных сетей на базе ПЛИС из высокоуровневого описания.

На данный момент существует ряд инструментов, способных автоматизировать процесс синтеза нейровычислителей на ПЛИС [1–3]. Входными данными для существующих инструментов являются высокоуровневая спецификация нейронной сети и значения ее весовых коэффициентов. Выходными данными является спецификация аппаратного ускорителя на языке описания аппаратуры (чаще всего VHDL или Verilog), учитывающая ограниченные ресурсы конкретной микросхемы ПЛИС.Существующие инструментальные средства синтеза можно условно разделить на две группы в зави-

симости от формата входных данных и, как следствие, алгоритма синтеза.

К первой группе следует отнести такие инструменты как LeFlow [4, 5] и OpenVINO[1]. Их главная особенность заключается в том, что они используют модель СНС, полностью описанную на языке программирования высокого уровня, и выполняют автоматическую трансляцию синтаксических конструкций в более низкоуровневое представление и их дальнейшую оптимизацию.

Ко второй группе следует отнести такие продукты как AccDNN [6], Caffeine [7], HLS4ML [8]. Продукты этой группы используют в качестве входных данных модель сети, описанную в конфигурационных файлах специального формата.

Основное преимущество подхода, которого придерживаются продукты первой группы, заключается в том, что можно использовать различные языковые конструкции для создания более гибкой модели и тонко настроить ее. Это достаточно трудоемкий процесс, и при этом не учитываются особенности целевой аппаратной платформы. Также часть реализации вычислительного процесса уже закреплена в спецификации сети, что приводит к неэффективному использованию ресурсов, и наблюдаются проблемы с производительностью.

Подход продуктов второй группы изначально предполагает определение ограничений синтеза с помощью директив и позволяет синтезировать лучшую по характеристикам модель за счет того, что в конфигурации сети не закреплена изначально ее реализация. Реализация же будет определяться в процессе синтеза в соответствии с установленными требованиями.

В настоящее время существующие инструменты из второй группы недостаточно гибко реализуют преимущества вышеописанного подхода и ограничиваются прямым переносом сети на базе зафиксированных шаблонов синтеза типовых аппаратных блоков. Прямой перенос не позволяет эффективно синтезировать сети с десятками и сотнями нейронов и требует большего количество аппаратных ресурсов ПЛИС. Также в инструментах, реализующих второй подход, недостаточное внимание уделяется предварительной оптимизации структуры сетей с целью уменьшения числа используемых параметров и сохранением точности работы в допустимых для задачи пределах [9–12].

В данной работе приведено описание модели инструментального средства автоматизированного синтеза аппаратных ускорителей СНС, которое позволит устранить описанные недостатки подхода второй группы. В процессе синтеза проведена оптимизация структуры сети, и выбраны способы организации вычислительного процесса, удовлетворяющие требованиям по производительности и используемым ресурсам ПЛИС. Оптимизация позволит сохранить точность работы сети в допустимых пределах и одновременно увеличить производительность синтезируемого вычислителя, а также снизить количество требуемых ресурсов ПЛИС.

Описание предлагаемого средства автоматизированного синтеза

Предлагаемый инструмент синтеза призван объединить преимущества форматов входных данных двух описанных групп инструментов. Он получает на вход конфигурацию сети из фреймворка TensorFlow [13, 14] без определения алгоритма расчета выхода сети и синтезирует ее с учетом заданных ограничений в реализацию сети на высокоуровневом языке программирования C++. Таким образом, можно использовать гибкость первой группы инструментов, не закрепляя на уровне спецификации реализацию сети, а определяя ее с учетом заданных требований к производительности и используемым аппаратным ресурсам.

Общий маршрут синтеза аппаратных ускорителей нейронных сетей с использованием разрабатываемого средства приведен на рис. 1. Маршрут синтеза включает выполнение следующих шагов:

1) создание и тестирование нейронной сети с использованием языка Python и фреймворка TensorFlow, выполнение предварительной оптимизации сети и экспорт ее модели;
2) использование разрабатываемого средства синтеза для генерации спецификации нейросетевого ускорителя на языках Verilog HDL или VHDL;
3) создание файла конфигурации для выбранной ПЛИС с использованием САПР Vivado от фирмы Xilinx.

В процессе своей работы разрабатываемый инструмент синтеза выполняет ряд оптимизаций, с целью удовлетворения установленным требованиям к производительности и используемым аппаратным ресурсам ПЛИС:

1) оптимизация структуры полносвязных слоев нейронной сети с целью уменьшения количества используемых параметров (весов) сети с контролем снижения точности работы сети в установленных пределах;
2) оптимизация структуры сверточных слоев с целью уменьшения количества используемых параметров сети с контролем точности работы сети;
3) выбор варианта реализации вычислительного процесса с использованием параллельных блоков обработки данных и конвейеризации вычислений. Предварительный синтез и оценка результатов синтеза выполняется с помощью системы автоматизированного проектирования (САПР) Vivado HLS от фирмы Xilinx.

Далее представлен качественный и количественный анализ эффекта от проводимых оптимизаций структуры сети и организации вычислительного процесса.

Анализ используемых стратегий оптимизации нейронных сетей

Разрабатываемый инструмент синтеза в процессе своей работы реструктурирует сеть и выбирает вариант организации вычислительного процесса с целью удо-

[1] Release Notes for Intel Distribution of OpenVINO toolkit 2018 [Электронный ресурс]. URL: https://software.intel.com/en-us/articles/OpenVINO-RelNotes-2018 (дата обращения: 01.02.2020).

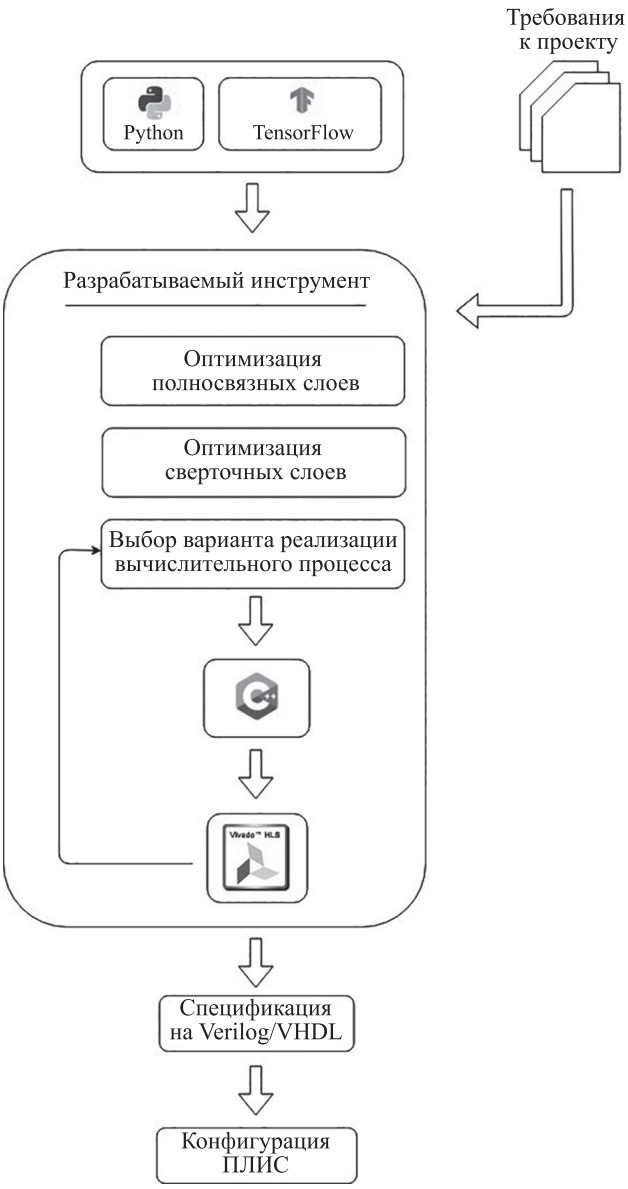

Рис. 1. Маршрут синтеза аппаратных ускорителей нейронных сетей на программируемых логических интегральных схемах с использованием разрабатываемого инструмента

влетворения требованиям проекта. Для иллюстрации эффекта от проводимых оптимизаций реализована СНС для распознавания рукописных цифр из набора MNIST. Модель состоит из следующих слов:

1) Con2d — сверточный слой, на вход которого подается изображение 28 × 28 пикселей, состоит из 32 нейронов, ядро свертки 3 × 3 пикселя, входной слой;
2) MaxPool — слой подвыборки с ядром 2 × 2, который осуществляет выбор максимального числа из результатов работы сверточного слоя;
3) Dense1 — полносвязный слой, состоящий из 100 нейронов;
4) Dense2 — полносвязный слой, состоящий из 10 нейронов, выходной слой.

Данная конфигурация сети содержит 644 120 параметров и способна распознавать число с точностью 99,85 %. Прямой синтез такой модели потребует от аппаратной платформы чрезвычайно большого количества ресурсов, что приведет к исключительно низкой производительности синтезируемого ускорителя. На первом шаге процесса синтеза разрабатываемый инструмент фиксирует допустимый порог в уменьшении точности расчета выхода сети в пределах 1 %, что практически не сказывается на способности сети к распознаванию. Далее в итеративном режиме проводится анализ зависимости точности и величины ошибки от количества параметров сети, и выбирается оптимальная структура синтезируемой нейронной сети. Оптимизация структуры начинается с полносвязного слоя Dense1, содержащего бо̀льшее количество нейронов, структура которого сильно влияет на время расчета выхода сети.

Обучение сети проводилось на наборе из 60 000 изображений в течение 10 эпох, точность распознавания оценивалась на наборе из 10 000 тестовых изображений.

В табл. 1 представлено изменение точности распознавания для обучающей и тестовой выборок в зависимости от количества нейронов в слое Dense1.

Таким образом, даже избавившись от целого слоя, сеть потеряла в точности всего 0,43 %, в то время как число параметров сократилось на 581 070 (90 %).

Следующим шагом происходит оптимизация сверточного слоя с постепенным уменьшением количества нейронов и контролем точности. Результаты оценки изменения точности работы сети представлены в табл. 2.

По результатам, представленным в табл. 2, видно, что, сократив в два раза количество нейронов сверточного слоя, точность составила 98,88 %, т. е. в итоге уменьшилась менее чем на 1 %, а число параметров сократилось до 31 530.

Таблица 1. Зависимость параметров сети от количества нейронов в слое Dense1

Количество нейронов	Количество параметров сети	Точность при обучении	Точность при проверке
100	644 120	0,9985	0,9835
80	502 970	0,9979	0,9846
60	377 310	0,9976	0,9792
40	251 650	0,9969	0,9823
20	125 990	0,9943	0,9759
0	63 050	0,9942	0,9812

Таблица 2. Зависимость параметров сети от количества нейронов в слое Con2d

Количество нейронов	Количество параметров сети	Точность при обучении	Точность при проверке
32	63 050	0,9942	0,9812
28	55 170	0,9927	0,9810
24	47 290	0,9911	0,9802
20	39 410	0,9908	0,9804
16	31 530	0,9888	0,9795

По результатам структурных оптимизаций сверточного и полносвязного слоев удалось сократить количество параметров сети на 95 %, потеряв в точности всего 1 %.

Эффект от использования различных вариантов организации вычислительного процесса показан на базе реализации сверточного слоя используемой сети на конкретной микросхеме ПЛИС. В качестве целевой платформы синтеза выбрана ПЛИС XC7VC485T-FFG1157-1 фирмы Xilinx.

Перебор различных вариантов организации вычислительного процесса осуществлялся с использованием САПР Vivado HLS. С помощью разрабатываемого средства синтеза Vivado HLS итеративно запускался с применением директив UNROLL и PIPELINE для блоков сети; по результатам синтеза оценивалась производительность и используемые ресурсы ПЛИС, осуществлялся поиск варианта, удовлетворяющего заданным требованиям.

На рис. 2 представлена зависимость времени расчета выхода сверточного слоя в тактах от способа расчета: без использования директив (последовательный), с директивой UNROLL (параллельный способ расчета), с использованием директивы PIPELINE (конвейерный способ расчета).

Директива UNROLL показывает наилучшее время в отсутствии ограничений, однако требует больших ресурсов как от инструментального компьютера, так и от ПЛИС. Имеется возможность более тонко настроить эту директиву с помощью коэффициента повторного использования (КПИ), который указывает число одновременно исполняемых итераций.

В процессе синтеза разрабатываемый инструмент производит оценку количества используемых ресурсов ПЛИС. На рис. 3 показано количество задействованных элементов регистровой памяти (D-триггеров) и комбинационных элементов в виде просмотровых таблиц (LUT-элементов) ПЛИС при расчете выхода сверточного слоя. Как можно увидеть, применение директивы UNROLL ведет к использованию большего количества ресурсов, хотя и значительно снижает время вычислений.

Еще одна оптимизация на уровне синтезируемого языка (в данном случае C++), которая применяется в разрабатываемом инструменте – это переход от чисел с плавающей точкой к числам с фиксированной точкой. При таком переходе незначительно теряется точность, однако используется меньшее количество ресурсов, меньшее количество выделяемой регистровой памяти под каждое значение, а также снижается сложность

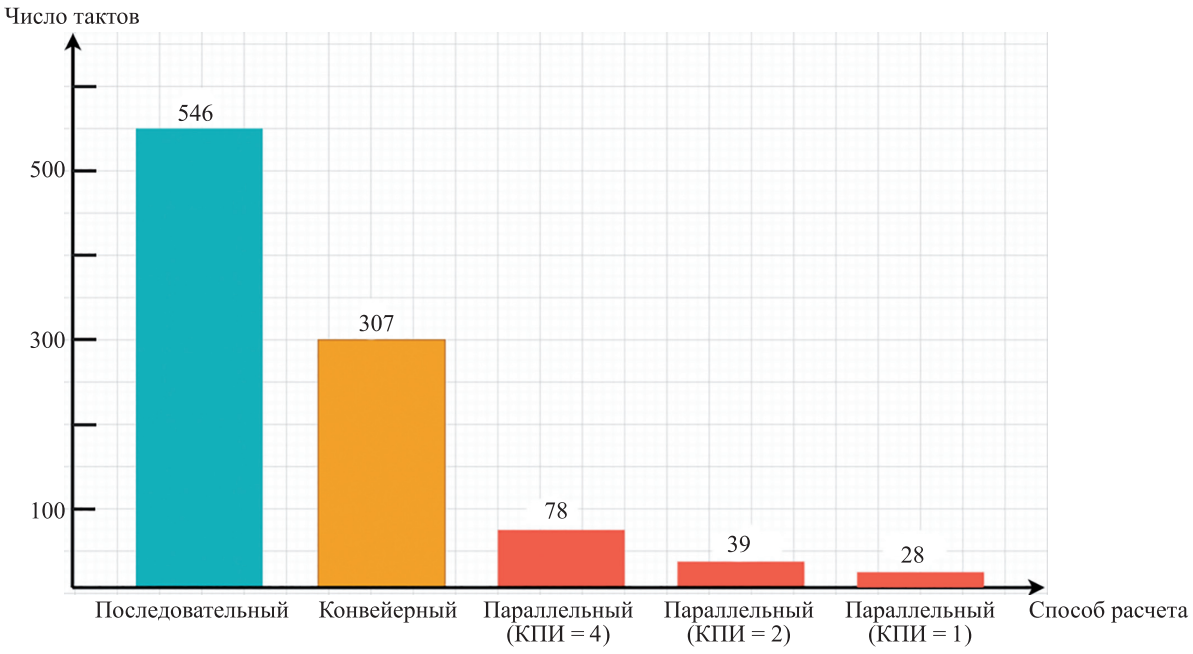

Рис. 2. Зависимость времени расчета выхода сверточного слоя от способа расчета (КПИ — коэффициент повторного использования)

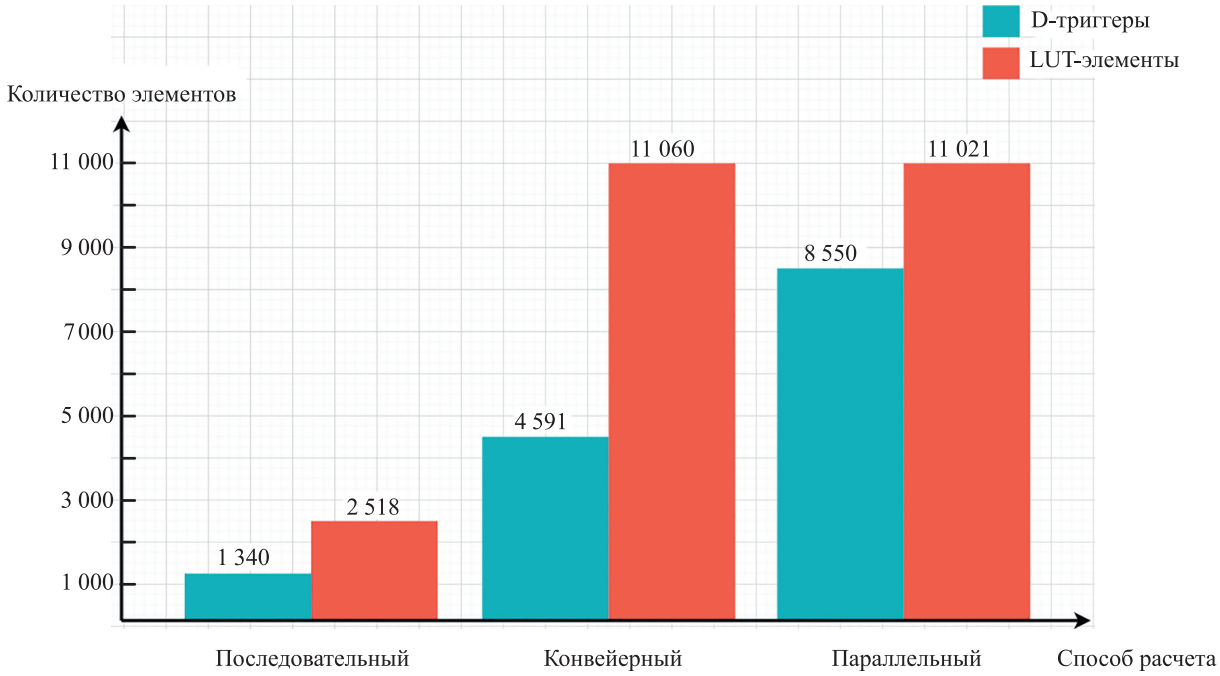

Рис. 3. Число задействованных D-триггеров и LUT-элементов при расчете выхода сверточного слоя для разных способов расчета

выполнения арифметических операций. Сравнение количества необходимых D-триггеров (регистровой памяти) при использовании чисел с фиксированной и плавающей точками показано на рис. 4.

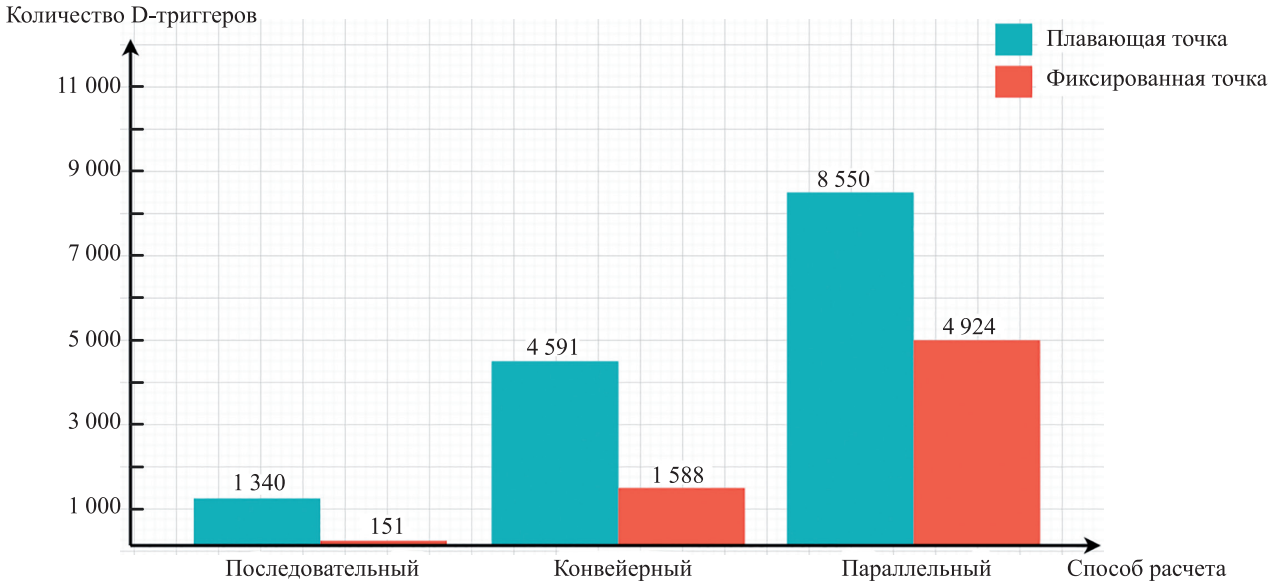

Рис. 4. Число использованных D-триггеров при расчете сверточного слояс применением чисел с фиксированной и плавающей точками

Заключение

Обоснована необходимость создания средств автоматизированного синтеза аппаратных ускорителей сверточных нейронных сетей и предложена модель такого инструмента. Разработан прототип библиотеки на синтезируемом подмножестве языка C++, содержащий описания классов слоев сверточной нейронной сети и используемый в процессе синтеза. Для автоматизации каждого этапа синтеза разработан набор скриптов для системы автоматизированного проектирования Vivado HLS. При этом оценка результатов проведенных оптимизаций и решение о повторении каждого этапа, но уже с другими параметрами, осуществлялись вручную.

Предлагаемый инструмент способен выполнять оптимизацию структуры полносвязного слоя, сверточного слоя и способов организации вычислительного

процесса. Произведена оценка эффективности данного набора оптимизаций на примере синтеза нейронной сети для распознавания рукописных цифр из набора MNIST. Показано, что оптимизация структуры полносвязных слоев нейронной сети является оправданной с точки зрения сокращения количества ее параметров с допустимой потерей точности (около 0,43 %), распараллеливание отдельных частей вычислительного процесса позволяет рассчитать выход нейронной сети в среднем в 12 раз быстрее, однако требует в 4–6 раз больше вычислительных ресурсов. Также для экономии ресурсов программируемых логических интегральных схем может использоваться переход от вычислений с плавающей точкой к вычислениям с фиксированной точкой. Такая оптимизация позволяет сократить используемые ресурсы еще в 1,7–2,8 раз.

Таким образом, устанавливая на начальном этапе синтеза требования к производительности и используемым ресурсам программируемых логических интегральных схем, с помощью предлагаемого инструмента возможно оптимизировать в широких пределах получаемую реализацию аппаратного ускорителя нейронной сети.

Следующей задачей в развитии предлагаемого инструмента планируется разработка метода многокритериальной оптимизации результатов синтеза на основе результатов каждого этапа и объединение разработанных составных частей в единую систему автоматизированного проектирования.

Литература

1. Guan Y., Liang H., Xu N., Wang W., Shi S., Chen X., Sun G., Zhang W., Cong J. FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates // Proc. 25th Annual IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM 2017). 2017. P. 152–159. doi: 10.1109/FCCM.2017.25
2. Venieris S.I., Bouganis C.-S. FpgaConvNet: A framework for mapping convolutional neural networks on FPGAs // Proc. 24th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM 2016). 2016. P. 40–47. doi: 10.1109/FCCM.2016.22
3. Venieris S.I., Bouganis C.-S. Latency-driven design for FPGA-based convolutional neural networks // Proc. 27th International Conference on Field Programmable Logic and Applications (FPL 2017). 2017. P. 8056828. doi: 10.23919/FPL.2017.8056828
4. Noronha D.H., Salehpour B., Wilton S.J.E. LeFlow: enabling flexible FPGA high-level synthesis of tensorflow deep neural networks // Proc. 5th International Workshop on FPGAs for Software Programmers (FSP 2018), co-located with International Conference on Field Programmable Logic and Applications (FPL 2018). 2018. P. 46–53.
5. Lattner C., Adve V. LLVM: A compilation framework for lifelong program analysis & transformation // Proc. of the International Symposium on Code Generation and Optimization (CGO 2004). 2004. P. 75–86. doi: 10.1109/CGO.2004.1281665
6. Zhang X., Wang J., Zhu C., Lin Y., Xiong J., Hwu W.-M., Chen D. DNNBuilder: an automated tool for building high-performance DNN hardware accelerators for FPGAs // Proc. 37th IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2018). 2018. P. a56. doi: 10.1145/3240765.3240801
7. Zhang C., Sun G., Fang Z., Zhou P., Pan P., Cong J. Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2019. V. 38. N 11. P. 2072–2085. doi: 10.1109/TCAD.2017.2785257
8. Duarte J., Han S., Harris P., Jindariani S., Kreinar E., Kreis B., Ngadiuba J., Pierini M., Rivera R., Tran N., Wu Z. Fast inference of deep neural networks in FPGAs for particle physics // Journal of Instrumentation. 2018. V. 13. N 7. P. P07027. doi: 10.1088/1748-0221/13/07/P07027
9. Cheng Y., Wang D., Zhou P., Zhang T. A survey of model compression and acceleration for deep neural networks // arXiv:1710.09282.
10. Rastegari M., Ordonez V., Redmon J., Farhadi A. XNOR-net: Imagenet classification using binary convolutional neural networks // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2016. V. 9908. P. 525–542. doi: 10.1007/978-3-319-46493-0_32
11. Vanhoucke V., Senior A., Mao M.Z. Improving the speed of neural networks on CPUs // Proc. of the Deep Learning and Unsupervised Feature Learning Workshop (NIPS 2011). 2011.
12. Jang H., Park A., Jung K. Network implementation using CUDA and OpenMP // Proc. of the Digital Image Computing: Techniques and Applications (DICTA 2008). 2008. P. 155–161. doi: 10.1109/DICTA.2008.82

References

1. Guan Y., Liang H., Xu N., Wang W., Shi S., Chen X., Sun G., Zhang W., Cong J. FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates. *Proc. 25th Annual IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM 2017)*, 2017, pp. 152–159. doi: 10.1109/FCCM.2017.25
2. Venieris S.I., Bouganis C.S. FpgaConvNet: A Framework for mapping convolutional neural networks on FPGAs. *Proc. 24th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM 2016)*, 2016, pp. 40–47. doi: 10.1109/FCCM.2016.22
3. Venieris S.I., Bouganis C.-S. Latency-driven design for FPGA-based convolutional neural networks. *Proc. 27th International Conference on Field Programmable Logic and Applications (FPL 2017)*, 2017, pp. 8056828. doi: 10.23919/FPL.2017.8056828
4. Noronha D.H., Salehpour B., Wilton S.J.E. LeFlow: enabling flexible FPGA high-level synthesis of tensorflow deep neural networks. *Proc. 5th International Workshop on FPGAs for Software Programmers (FSP 2018), co-located with International Conference on Field Programmable Logic and Applications (FPL 2018)*, 2018, pp. 46–53.
5. Lattner C., Adve V. LLVM: a compilation framework for lifelong program analysis & transformation. *Proc. of the International Symposium on Code Generation and Optimization (CGO 2004)*, 2004, pp. 75–86. doi: 10.1109/CGO.2004.1281665
6. Zhang X., Wang J., Zhu C., Lin Y., Xiong J., Hwu W.-M., Chen D. DNNBuilder: an automated tool for building high-performance DNN hardware accelerators for FPGAs. *Proc. 37th IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2018)*, 2018, pp. a56. doi: 10.1145/3240765.3240801
7. Zhang C., Sun G., Fang Z., Zhou P., Pan P., Cong J. Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019, vol. 38, no. 11, pp. 2072–2085. doi: 10.1109/TCAD.2017.2785257
8. Duarte J., Han S., Harris P., Jindariani S., Kreinar E., Kreis B., Ngadiuba J., Pierini M., Rivera R., Tran N., Wu Z. Fast inference of deep neural networks in FPGAs for particle physics. *Journal of Instrumentation*, 2018, vol. 13, no. 7, pp. P07027. doi: 10.1088/1748-0221/13/07/P07027
9. Cheng Y., Wang D., Zhou P., Zhang T. A survey of model compression and acceleration for deep neural networks. *arXiv:1710.09282*.
10. Rastegari M., Ordonez V., Redmon J., Farhadi A. XNOR-net: Imagenet classification using binary convolutional neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9908, pp. 525–542. doi: 10.1007/978-3-319-46493-0_32
11. Vanhoucke V., Senior A., Mao M.Z. Improving the speed of neural networks on CPUs. *Proc. of the Deep Learning and Unsupervised Feature Learning Workshop (NIPS 2011)*, 2011.
12. Jang H., Park A., Jung K. Network implementation using CUDA and OpenMP. *Proc. of the Digital Image Computing: Techniques and Applications (DICTA 2008)*, 2008, pp. 155–161. doi: 10.1109/DICTA.2008.82

13. Abadi M., Agarwal A., Barham P. et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems // arXiv:1603.04467.
14. Chollet F. et. al. Keras. 2015 [Электронный ресурс]. URL: https://github.com/keras-team/keras (дата обращения: 01.02.2020).

Авторы

Егиазарян Виктор Александрович — младший инженер, ООО «Сименс», Санкт-Петербург, 191186, Российская Федерация, ORCID ID: 0000-0003-1642-2307, aes3plex@ya.ru
Быковский Сергей Вячеславович — кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, Scopus ID: 57211618416, ORCID ID: 0000-0003-4163-9743, sergei_bykovskii@itmo.ru

13. Abadi M., Agarwal A., Barham P. et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*.
14. *Chollet F. et. al. Keras*. 2015. Available at: https://github.com/keras-team/keras (accessed: 01.02.2020).

Authors

Victor A. Egiazarian — Junior Engineer, OOO Siemens, Saint Petersburg, 191186, Russian Federation, ORCID ID: 0000-0003-1642-2307, aes3plex@ya.ru
Sergei V. Bykovskii — PhD, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, Scopus ID: 57211618416, ORCID ID: 0000-0003-4163-9743, sergei_bykovskii@itmo.ru