



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ

Робототехника и комплексная автоматизация (РК)

КАФЕДРА

Системы автоматизированного проектирования (РК6)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

***«Обзор инструментов для программирования нейронных
сетей на FPGA-платах.»***

Студент РК6-72Б

(Подпись, дата)

Караф С.М.

И.О. Фамилия

Руководитель НИР

(Подпись, дата)

Витюков Ф.А.

И.О. Фамилия

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой РК6
А.П. Карпенко

«_____» _____ 2022 г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме: Обзор инструментов для программирования нейронных сетей на FPGA платах _____

Студент группы РК6-72Б

_____ Караф Сармат Майк _____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.) учебная
Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения НИР: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

Техническое задание: Провести обзор инструментов для создания моделей машинного обучения ,
нейронных сетей на FPGA платах.

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на 14 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

Дата выдачи задания «10» сентября 2022 г.

Руководитель НИР

(Подпись, дата)

Витюков Ф.А.
И.О. Фамилия

Студент

(Подпись, дата)

Караф С.М.
И.О. Фамилия

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1. Программируемые логические интегральные схемы	5
2. Что такое FPGA.....	8
3. Сфера применения FPGA.....	13
4. Реализация нейронных сетей на FPGA.....	15
ЗАКЛЮЧЕНИЕ	23
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	24

ВВЕДЕНИЕ

В зависимости от назначения встраиваемой системы, к ней могут предъявляться дополнительные требования по работоспособности в различных средах, высокому гарантированному сроку службы и т.д. В связи с тем, что встраиваемые системы обрабатывают сигналы в реальном времени при ограниченности вычислительных ресурсов, к алгоритмам предъявляются строгие требования по производительности.

Графические процессоры (GPU), которые чаще всего используются в качестве вычислителей для сверточных нейронных сетей, не соответствуют данным требованиям по нескольким причинам. Во-первых, GPU не предназначены к эксплуатации при постоянных вибрациях и ударных нагрузках. Во-вторых, графические процессоры отличаются значительным электропотреблением. В-третьих, требуется особая схема отвода тепла с GPU. Кроме того, практически все графические процессоры, присутствующие на рынке, являются импортными и не могут использоваться в приборах, имеющих требование использования только отечественных комплектующих.

Поэтому для запуска нейронных сетей на встраиваемых системах необходимо использовать другие вычислительные устройства. Чаще всего в качестве вычислительного устройства используются системы на кристалле (System on Chip, SoC) — электронные схемы, содержащие в себе один или несколько микропроцессоров, а также банк памяти, блоки стандартных интерфейсов для подключения внешних устройств, и выполняющие функции целого вычислительного устройства. Во многих системах на кристалле в качестве компонента присутствуют программируемые логические интегральные схемы (ПЛИС), а именно программируемая пользователем вентильная матрица (field-programmable gate array, FPGA). Особенностью SoC, содержащих FPGA, является то, что большая часть памяти устройства находится в банках памяти, тогда как собственная память ПЛИС мала и составляет порядка нескольких тысяч килобит. Доступ к внешним банкам памяти происходит дольше, чем

обращение к встроенному ОЗУ, поэтому уменьшение количества 5 памяти, потребляемой нейронной сетью, способно увеличить скорость её работы на ПЛИС.

FPGA помимо логических блоков и блоков коммутации содержит еще и блоки цифровой обработки сигналов (digital signal processor, DSP), которые позволяют быстро производить операции над вещественными числами. Несмотря на наличие данных блоков, максимальная пиковая производительность логических операций на ПЛИС больше, чем у операций с плавающей точкой. Поэтому лучше всего для реализации на FPGA подходят бинарные нейронные сети, поскольку их выполнение можно реализовать, используя преимущественно логические операции, при этом каждый вес такой сети требует всего один бит для хранения, что позволяет одновременно хранить все веса на устройстве.

1. Программируемые логические интегральные схемы

Проектирование новых интегральных схем — сложный и многогранный процесс. Разработчикам необходимо продумать расположение огромного числа элементов: сдвиговые регистры, дешифраторы, мультиплексоры и прочее. Большинство интегральных схем (ИС), в том числе процессоры и микроконтроллеры, имеют predetermined логику работы.

Проще говоря, в таких схемах после выпуска архитектура остается неизменной. Программисты в свою очередь получают набор определенных команд, с помощью которых и взаимодействуют с конкретной схемой.

Одна из главных проблем проектировщиков — это поиск компромисса между скоростью работы схемы и ее универсальностью.

Основным «кирпичиком» при построении любой интегральной схемы являются логические элементы (вентили) — «И», «ИЛИ», «НЕ». На их базе уже создаются триггеры, регистры и так далее.

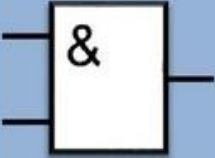
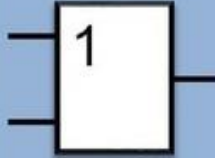
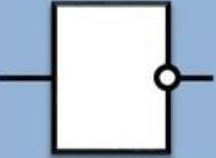
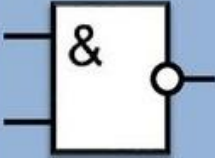
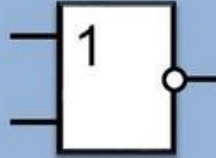
И	ИЛИ	НЕ	И-НЕ	ИЛИ-НЕ
				
$F = A \& B$	$F = A \vee B$	$F = \bar{A}$	$F = \overline{A \& B}$	$F = \overline{A \vee B}$

Рис. 1. Логические элементы.

Как именно связываются между собой логические элементы — определяют разработчики микросхемы еще на этапе проектирования. И главная проблема в том, что эти связи в последствие уже изменить нельзя.

Однако в начале 70-х годов на рынке начали появляться первые программируемые логические устройства. Ключевое отличие от процессоров и микроконтроллеров — можно самостоятельно задавать архитектуру. Связи между логическими элементами не были предопределены и поддавались редактированию.

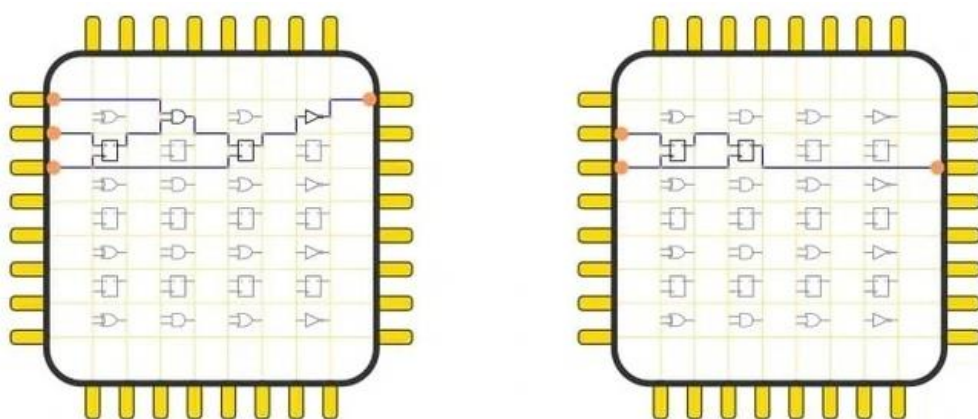


Рис. 2. Иллюстрация редактирования микроконтроллеров.

Изначально такие связи выполнялись в виде пережигаемых тонких проводников. По мере совершенствования технологий стали использоваться МОП-транзисторы с плавающим затвором. Появилась возможность реконфигурировать внутреннюю структуру микросхемы. Так появились различные подвиды ПЛИС.

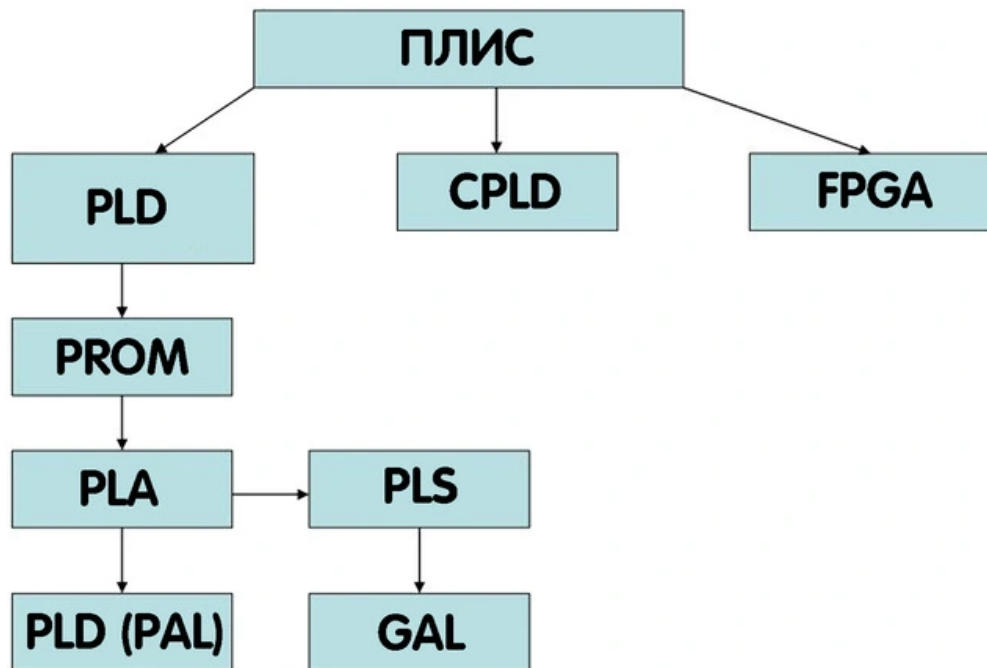


Рис. 3. Классификация ПЛИС.

Схемах PAL элементы «И» — программируемые, а элементы «ИЛИ» — фиксированные. На базе таких блоков можно было создавать достаточно сложные заказные схемы с минимальными затратами. Однако PAL имели плавкие титановольфрамовые перемычки, поэтому не могли использоваться повторно.

В 1985 году появилось семейство микросхем GAL — Generic Array Logic. В отличие от предыдущих эти схемы можно было программировать многократно, повторно задавая связи между элементами. Группу из PROM, PAL, PLA, GAL принято обобщать термином SPLD — Simple Programmable Logic Device. Однако большую часть рынка сегодня занимают FPGA

2. Что такое FPGA

FPGA (Field-Programmable Gate Array) — это один из подвидов программируемых интегральных схем. Строятся такие микросхемы на логических блоках с гибкой коммутацией — причем число блоков может достигать до сотен тысяч штук. Прошивка с «картой» необходимых связей между логическими ячейками сохраняется в энергонезависимой памяти.

Условно FPGA состоят из трех основных элементов — конфигурируемые логические блоки (CLB), блок ввода-вывода (IOB) и межсоединения. Каждый CLB включает в себя таблицы поиска, триггеры, регистры, мультиплексоры и не только. Благодаря этому CLB могут выполнять логические и арифметические операции, а также использоваться для хранения данных.

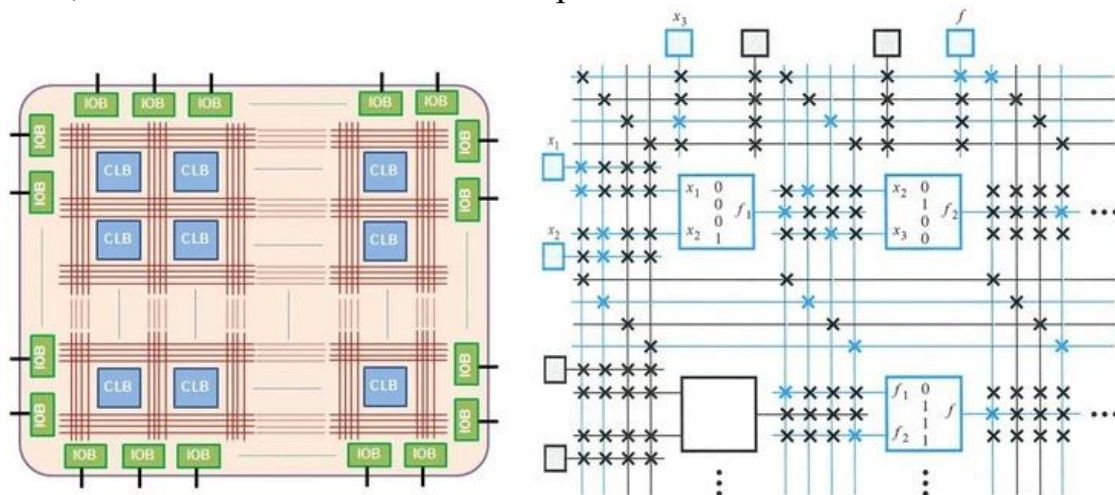


Рис. 4. Общая схема и программируемая секция FPGA.

Если смотреть структуру CLB подробнее, то здесь все зависит от каждой конкретной микросхемы. Чем дороже плата — тем обычно большие возможности предлагает каждый конкретный конфигурируемый логический блок.

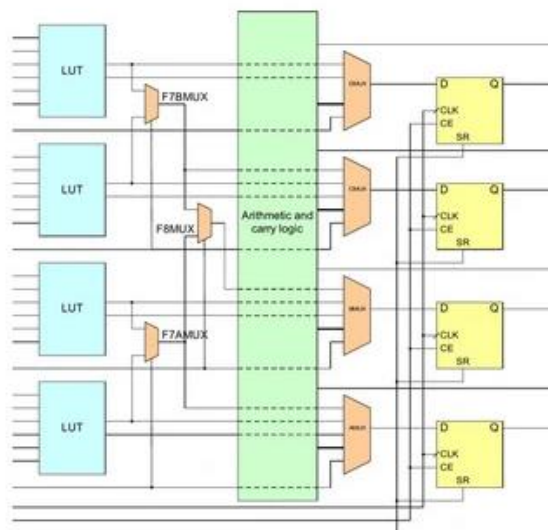


Рис. 5. Работа конфигурируемого логического блока (CLB).

- LUT — Look-Up Tables. Каждый такой блок способен реализовать любую логическую функцию используя в качестве данных операнды с входов.
- Специализированные управляемые пользователем мультиплексоры (MUX) для комбинационной логики.
- Блок арифметической логики, который позволяет делать суммирование и умножение операндов (зеленый блок).
- Несколько однобитных регистров для хранения информации (желтые блоки).

Таким образом, CLB — это своеобразный универсальный кирпичик, который способен выполнить практически любую операцию над данными. На FPGA плате таких — сотни и тысячи.

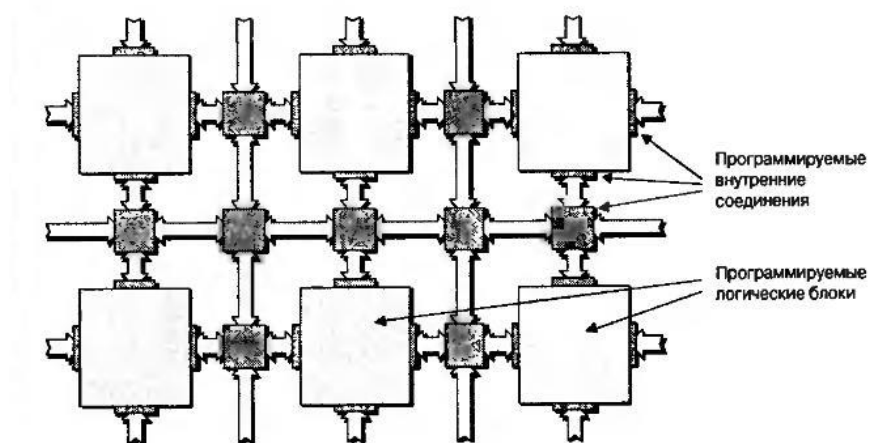
Можно собрать сперва одну схему, например, условный игрушечный автомобиль, а затем разобрать ее и собрать совсем другую. Благодаря этому на FPGA можно создавать самые разнообразные устройства, при этом не изменяя аппаратную начинку. Возможности пользователей ограничены лишь числом блоков CLB и интерфейсами, которые имеются на плате.

3. Принцип работы

Микросхема FPGA — это заказная микросхема, состоящая из транзисторов, из которых собираются триггеры, регистры, мультиплексоры и другие логические элементы для обычных схем. Изменить порядок соединения этих транзисторов, конечно, нельзя. Но архитектурно микросхема построена таким хитрым образом, что можно изменять коммутацию сигналов между более крупными блоками: их называют CLB — программируемые логические блоки.

Также можно изменять логическую функцию, которую выполняет CLB. Достигается это за счет того, что вся микросхема пронизана ячейками конфигурационной памяти Static RAM. Каждый бит этой памяти либо управляет каким-то ключом коммутации сигналов, либо является частью таблицы истинности логической функции, которую реализует CLB.

Так как конфигурационная память построена по технологии Static RAM, то, во-первых, при включении питания FPGA микросхему обязательно надо сконфигурировать, а во-вторых, микросхему можно реконфигурировать



практически бесконечное количество раз.

Рис. 6. Упрощенная 2D-структура микросхемы без конфигурационной памяти

Блоки CLB находятся в коммутационной матрице, которая задает соединения входов и выходов блоков CLB.

На каждом пересечении проводников находится шесть переключающих ключей, управляемых своими ячейками конфигурационной памяти. Открывая одни и закрывая другие, можно обеспечить разную коммутацию сигналов между CLB.

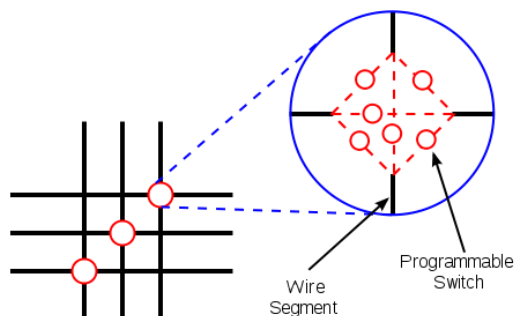


Рис. 7. Схема коммутационной матрицы

LB очень упрощенно состоит из блока, задающего булеву функцию от нескольких аргументов (она называется таблицей соответствия — Look Up Table, LUT) и триггера (flip-flop, FF). В современных FPGA LUT имеет шесть входов, но на рисунке для простоты показаны три. Выход LUT подается на выход CLB либо асинхронно (напрямую), либо синхронно (через триггер FF, работающий на системной тактовой частоте).

Значение каждой из ячеек подается на свой вход выходного мультиплексора LUT, а входные аргументы булевой функции используются для выбора того или иного значения функции. CLB — важнейший аппаратный ресурс FPGA. Количество CLB в современных кристаллах FPGA может быть разным и зависит от типа и емкости кристалла. У Xilinx есть кристаллы с количеством CLB в пределах примерно от четырех тысяч до трех миллионов.

Помимо CLB, внутри FPGA есть еще ряд важных аппаратных ресурсов. Например, аппаратные блоки умножения с накоплением или блоки DSP. Каждый из них может делать операции умножения и сложения 18-битных чисел каждый такт. В топовых кристаллах количество блоков DSP может превышать 6000.

Другой ресурс — это блоки внутренней памяти (Block RAM, BRAM). Каждый блок может хранить 2 Кбайт. Полная емкость такой памяти в

зависимости от кристалла может достигать от 20 Кбайт до 20 Мбайт. Как и CLB, BRAM и DSP-блоки связаны коммутационной матрицей и пронизывают весь кристалл. Связывая блоки CLB, DSP и BRAM, можно получать весьма эффективные схемы обработки данных.

4. Сфера применения FPGA.

По своей универсальности FPGA проигрывают типичным процессорам и микроконтроллерам. Однако возможность буквально программировать архитектуру дает преимущества в специфических задачах, в том числе перед ASIC.

В первую очередь FPGA отлично проявляют себя при разработке, в частности, прототипировании различных микросхем ASIC. создание ASIC — это крайне дорогостоящая задача, и перед выпуском платы в массы необходимо тщательно протестировать всю логику. Чтобы после нахождения недоработки каждый раз не выпускать новую плату, логику тестируют на нескольких кристаллах FPGA. При нахождении критической ошибки проектировщикам достаточно переработать архитектуру и выполнить реконфигурирование.

На структурах FPGA достаточно легко реализовать распараллеливание операций. В видеообработке это позволяет получить производительность на порядок выше, чем у программируемых DSP — цифровых процессоров обработки сигналов.

Например, возьмем задачу распознавания автомобильных номеров. Первый вариант — камера с возможностью передачи видео через Ethernet и обработкой потока на удаленном сервере. Однако при увеличении числа камер будет расти и нагрузка на сеть. А вот с энергоэффективной платой FPGA распознавание можно делать буквально в самой камере, а после — передавать на сервер лишь текстовые данные — полученные номера. При этом если формат автомобильных номеров изменится, вы быстро сможете реконфигурировать плату FPGA.

Параллелизм также отлично подходит для быстрой обработки больших объемов данных, что позволяет частично использовать FPGA в суперкомпьютерах или информационно-измерительных системах.

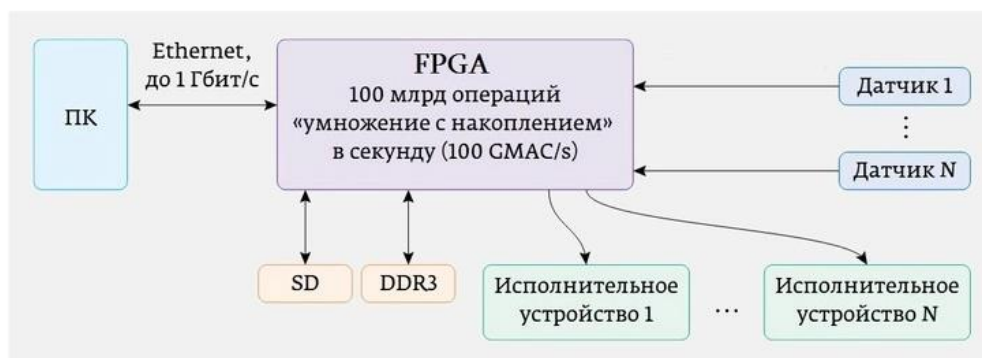


Рис. 8. Структурная схема цифровой части информационно-измерительной системы с применением FPGA.

Помимо этого, платы применяются в сфере коммуникаций — оборудование базовых станций GSM и так далее. Не менее интересен и тот факт, что львиная доля FPGA-ускорителей используется в военной сфере — схемы обеспечивают высокоскоростную обработку радиолокационных сигналов. Найти FPGA можно в беспилотных автомобилях, дронах, научных приборах, медицинской технике и не только.

5. Реализация нейронных сетей на ПЛИС

Нейронные сети и глубокие нейронные сети сейчас активно используются в разных областях, но реализация их на процессоре оказывается неэффективной существует много вычислений, которые можно распараллелить

Поэтому перенеся нейронную сеть на FPGA удастся на много порядков ускорить работу нейронной сети, остается обеспечить высокоскоростной интерфейс для загрузки исходных данных и получения результата. В качестве примера — реализация системы распознавания лиц на процессоре i7/9Gen распознает до 20 лиц за секунду с одной видеокамеры HD, реализация на FPGA — порядка 1000 лиц с нескольких камер.

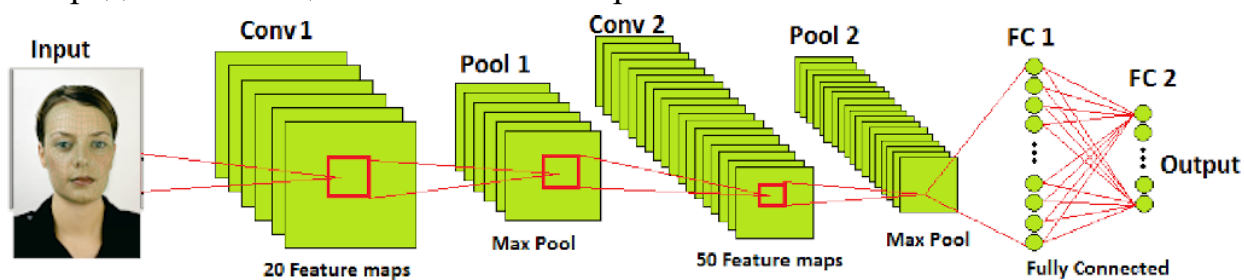


Рис. 9. Структура используемой глубокой нейронной сети.

Проектирование приложения нейронных сетей представляет собой трехэтапный процесс. Шаги заключаются в выборе правильной сети, обучении сети, а затем применении новых данных к обученной модели для прогнозирования (вывода). На рисунке 10 показаны шаги приложения для распознавания кошек.

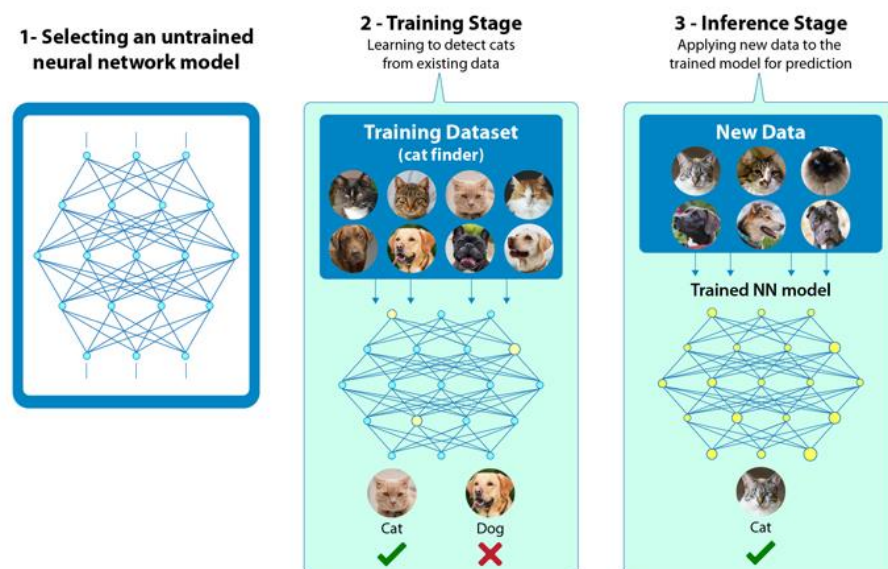


Рис. 10. Три шага распознавания кошки.

Как уже упоминалось, в модели нейронной сети есть несколько слоев, и каждый слой имеет определенную задачу. В глубоком обучении каждый слой предназначен для извлечения объектов на разных уровнях. Например, в нейронной сети обнаружения краев первый средний слой обнаруживает такие объекты, как ребра и кривые. Выход первого среднего слоя затем подается во второй слой, который отвечает за обнаружение объектов более высокого уровня, таких как полукруги или квадраты. Третий средний слой собирает выходные данные других слоев для создания знакомых объектов, а последний слой обнаруживает объект.

Алгоритмы сверточных нейросетей являются перспективными с точки зрения качества распознавания изображений. Наиболее надежным способом измерения качества систем машинного зрения является тестирование на больших базах. Тестирование на этих базах позволяет оценить способность алгоритма решать задачи классификации и детектирования объектов в реальных условиях. Алгоритмы на основе сверточных сетей уверенно лидируют уже несколько лет на этих базах и в решении подобных задач. Круг применения их расширяется с каждым годом. Однако реализация сверточных нейронных сетей обладает существенными требованиями по быстродействию. Для

«стационарных» решений наиболее распространенным способом обеспечения такой производительности является использование для вычисления универсальных графических процессоров. Однако в составе встраиваемых и мобильных решений следует обратить внимание на возможность использования программируемых логических интегральных схем.

ПЛИС прочно занимает нишу компонентов для штучных и мелкосерийных устройств, для которых с одной стороны требуется сложная логическая структура, а с другой стороны экономически не целесообразно разрабатывать и заказывать специализированную микросхему. В настоящий момент известны реализации сверточных сетей: устройства Tegra 4 произведены в 2013-2014 по технологии с проектными нормами 28 нм, процессоры Xilinx Zynq-7000 также выпущены в 2013м году по технологии с такими же проектными нормами. Таким образом, реализация сверточных сетей в виде микро интегральных устройства в настоящее время весьма актуальна.

5.1. Структура сверточной нейронной сети

Сверточные сети — это специализированный тип нейронных сетей, которые используют свертку вместо общего матричного умножения по крайней мере в одном из своих слоев. или, другими словами, Сверточная нейронная сеть (CNN) - это тип искусственной нейронной сети, используемой в распознавании и обработке изображений, которая специально предназначена для обработки пиксельных данных.

В обычном перцептроне, который представляет собой полносвязную нейронную сеть, каждый нейрон связан со всеми нейронами предыдущего слоя, причём каждая связь имеет свой персональный весовой коэффициент. В свёрточной нейронной сети в операции свёртки используется лишь ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою (в самом начале — непосредственно по входному изображению), формируя после каждого сдвига сигнал активации для нейрона

следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются одна и та же матрица весов, которую также называют ядром свёртки. Её интерпретируют как графическое кодирование какого-либо признака, например, наличие наклонной линии под определённым углом. Тогда следующий слой, получившийся в результате операции свёртки такой матрицей весов, показывает наличие данного признака в обрабатываемом слое и её координаты, формируя так называемую карту. Естественно, в свёрточной нейронной сети набор весов не один, а целая гамма, кодирующая элементы изображения (например линии и дуги под разными углами). При этом такие ядра свёртки не закладываются исследователем заранее, а формируются самостоятельно путём обучения сети классическим методом обратного распространения ошибки. Проход каждым набором весов формирует свой собственный экземпляр карты признаков, делая нейронную сеть многоканальной (много независимых карт признаков на одном слое). Также следует отметить, что при переборе слоя матрицей весов её передвигают обычно не на полный шаг (размер этой матрицы), а на небольшое расстояние. Так, например, при размерности матрицы весов 5×5 её сдвигают на один или два нейрона (пикселя) вместо пяти, чтобы не «перешагнуть» искомый признак.

Операция субдискретизации, выполняет уменьшение размерности сформированных карт признаков. В данной архитектуре сети считается, что информация о факте наличия искомого признака важнее точного знания его координат, поэтому из нескольких соседних нейронов карты признаков выбирается максимальный и принимается за один нейрон уплотнённой карты признаков меньшей размерности. За счёт данной операции, помимо ускорения дальнейших вычислений, сеть становится более инвариантной к масштабу входного изображения.

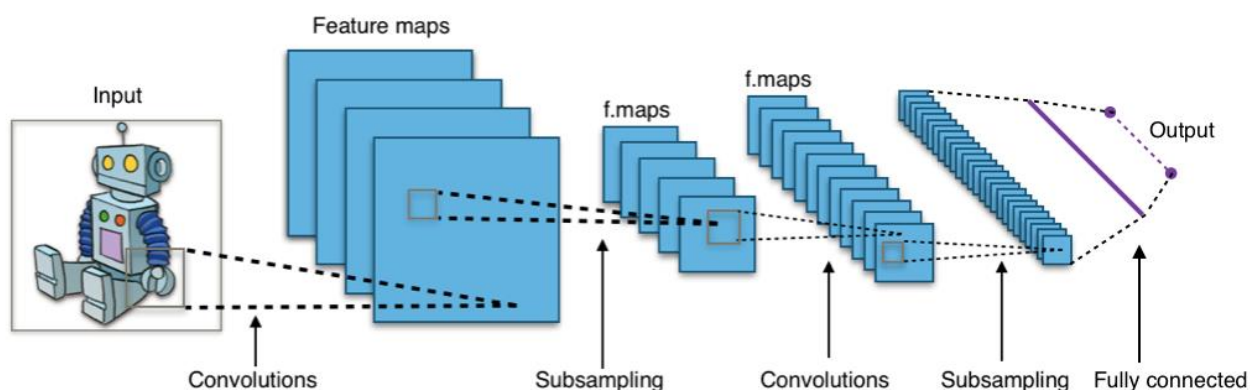


Рис. 11. Типовая архитектура свёрточной нейронной сети

5.2. Реализация свёрточной нейронной сети на FPGA

ПЛИС содержит элементы ядра — 16-битные числа с фиксированной точкой и 12 битами дробной части. Это обусловлено следующими соображениями: реализация операций с плавающей точкой на ПЛИС требует существенно больших ресурсов, а в задаче свёрточных сетей диапазон значений ядра является хорошо предсказуемым.

Одной из сложностей в реализации СНС на плис является недостаток памяти и умножителей на ПЛИС. То есть нельзя одновременно на одном кристалле осуществить свертку со всеми входными ядрами сети либо осуществить свертку всего кадра с входным окном.

Схема управления (см. Рис. 12) подключена с помощью шины

AXI4-Lite. Загрузка данных из основной памяти и отправка результата осуществляется с помощью DMA через шину AXI4.

Реализуемые функции:

- Загрузка ядер
- Обработка
- Замена ядра
- Сброс

Загрузка ядер представляет собою запись всех ядер (с использованием DMA) сразу в память ядер, размещенную на ПЛИС.



Рис. 12. Общая схема подключения модуля

Процесс обработки данных:

1. Получение данных от источника
2. Заполнение буфера FIFO для строк
3. Вычисление свертки в каждой позиции
4. Обновление и сохранение значения для операции подвыборки (maxpool)
5. Сохранение итогового значения (карты признаков) в локальной памяти
6. Сдвиг буфера строк

В качестве хранилища для строк изображения использовалось 11 (по размеру ядра) блоков памяти в режиме FIFO. Для хранения ядер используется еще один блок памяти (одно ядро занимает 244 байта). В качестве арифметического модуля используются блоки DSP48, они обеспечивают фиксированную задержку в 1 такт на операцию умножение и сложение (в режиме потоковой обработки).

Процесс свертки осуществляется непосредственно с помощью циклического списка очередей FIFO 1 — FIFO n из блоков памяти и набора регистров. Таким образом, значение интенсивности каждого следующего

пиксела изображения последовательно сдвигается через регистры последней строки, а затем записывается в очередь FIFO n одновременно с перемножением на каждом такте на коэффициент ядра. Это эквивалентно горизонтальному движению ядра свертки. Затем, по достижении конца строки, адреса счетчиков FIFO меняются, что эквивалентно сдвигу ядра свертки вниз на один пиксел.

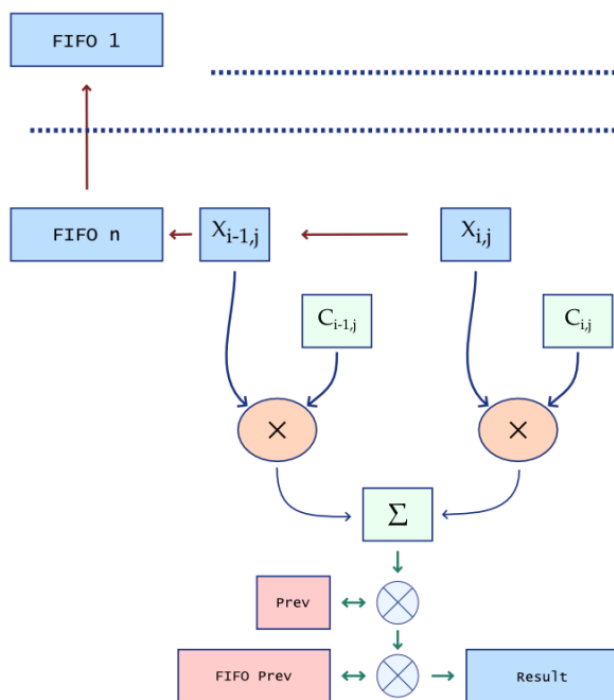


Рис. 13. Процесс обработки сигнала

Процесс свертки осуществляется непосредственно с помощью циклического списка очередей FIFO 1 — FIFO n из блоков памяти и набора регистров. Таким образом, значение интенсивности каждого следующего пиксела изображения последовательно сдвигается через регистры последней строки, а затем записывается в очередь FIFO n одновременно с перемножением на каждом такте на коэффициент ядра. Это эквивалентно горизонтальному движению ядра свертки. Затем, по достижении конца строки, адреса счетчиков FIFO меняются, что эквивалентно сдвигу ядра свертки вниз на один пиксел.

Для операции подвыборки результат свертки в нечетных столбцах нечетной строки записывается прямо в ячейку Prev предыдущего значения. В четных столбцах нечетной строки результат свертки сравнивается с ячейкой Prev и max записывается в блок памяти FIFO Prev. В следующей (четной) строке —

последовательность действий обратная, для нечетного столбца значение из памяти FIFO Prev читается и сравнивается со значением свертки в данной точке, затем записывается в Prev. Для четного столбца четной строки после сравнения с предыдущим значением максимальное значение (таким образом, являющееся подвыборкой по четырем ячейкам) записывается в память результата.

ЗАКЛЮЧЕНИЕ

Программируемые интегральные схемы в лице FPGA стремительно набирают популярность. Такие платы способны эффективно выполнять узкоспециализированные задачи, с которыми не способны быстро справиться процессоры или видеокарты. В то же время благодаря возможности реконфигурирования применение FPGA во многих сценариях намного выгоднее дорогостоящих ASIC.

FPGA предлагает максимальную гибкость, а со временем благодаря совершенствованию техпроцесса вырастет и производительность плат.

Алгоритмы на основе сверточных нейронных сетей получают все большую популярность в практических задачах. Эти алгоритмы могут сыграть большую роль во встраиваемых системах видео аналитики, поэтому адаптация СНС на ПЛИС является перспективным направлением. В работе показано как можно реализовать прямое распространение сигнала первого слоя СНС на ПЛИС. При этом выработана эффективная (в плане использования ресурсов) стратегия, рассчитанная на специфику сверток СНС.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Е.И. Литвинов. Лабораторный практикум, проектирование цифровых устройств на базе FPGA фирмы Xilinx/ Е.И. Литвинов, И.И. Шагурин, 2012. -173с.
2. Шагурин И., Шалтырев В., Волов А. «Большие» FPGA как элементная база для реализации систем на кристалле//Электронные компоненты, 2006, №5, с.83—88.
3. ПЛИС// «Википедия» — универсальная энциклопедия
// <https://ru.wikipedia.org/wiki/ПЛИС#FPGA>
4. Гонтаренко Б.В. Проблемы реализации искусственных нейронных сетей на FPGA // Информатика и компьютерные технологии-2011. – 2011. – С.15–18.
5. Girshick R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation //arXiv preprint arXiv:1311.2524. – 2013.
6. Gokhale V. et al. A 240 G-ops/s Mobile Coprocessor for Deep Neural Networks //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. – 2014. – С. 682-687.
7. Schmidhuber J. Deep Learning in Neural Networks: An Overview //arXiv preprint arXiv:1404.7828. – 2014.
8. Pham P. H. et al. NeuFlow: dataflow vision processing system-on-a-chip //Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on. – IEEE, 2012. – С. 1044-1047
9. Шауэрман А.А. Архитектура ПЛИС. Часть 1. Логический элемент [Электронный ресурс]. – Режим доступа:
// http://www.labfor.ru/articles/fpga_arch_le.

10. LeCun Y. et al. Gradient-based learning applied to document recognition
//Proceedings of the IEEE. – 1998. – T. 86. – №. 11. – C. 2278-2324.
11. Everingham M. et al. The pascal visual object classes (voc) challenge
//International journal of computer vision. – 2010. – T. 88. – №. 2. – C. 303-338.