Федеральное государственное автономное образовательное учреждение
высшего образования «Северо-Кавказский федеральный университет»

На правах рукописи

ВАЛУЕВА МАРИЯ ВАСИЛЬЕВНА

РАЗРАБОТКА МЕТОДОВ И АЛГОРИТМОВ ПОСТРОЕНИЯ ЦИФРОВЫХ УСТРОЙСТВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ВИЗУАЛЬНЫХ ДАННЫХ

Специальность: 2.3.5 – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ

на соискание ученой степени кандидата технических наук

Научный руководитель:

Доктор физико-математических наук, доцент

Бабенко Михаил Григорьевич

Ставрополь – 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5

1 МЕТОДЫ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ВИЗУАЛЬНЫХ ДАННЫХ 15

1.1 Сверточные нейронные сети 15

1.2 Методы повышения производительности нейронных сетей 19

1.3 Применение системы остаточных классов для оптимизации вычислений в сверточной нейронной сети 27

1.4 Выводы по первой главе 29

2 ОПЕРАЦИЯ СВЕРТКИ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ 31

2.1 Цифровая фильтрация изображений 31

2.2 Архитектура цифровых фильтров 33

2.3 Метод проектирования устройства, реализующего операцию свертки на основе метода Винограда и СОК с модулями специального вида 36

2.4 Проектирование фильтра на основе метода Винограда $F(2 \times 2, 2 \times 2)$ с вычислениями в системе остаточных классов 38

2.5 Проектирование фильтра на основе метода Винограда $F(2 \times 2, 3 \times 3)$ с вычислениями в системе остаточных классов 53

2.6 Проектирование фильтра на основе метода Винограда $F(2 \times 2, 5 \times 5)$ с вычислениями в системе остаточных классов 65

2.7 Выводы по второй главе 83

3 ВЫЧИСЛИТЕЛЬНО СЛОЖНЫЕ ОПЕРАЦИИ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ 85

3.1 Метод преобразования чисел из позиционной системы счисления в систему остаточных классов с модулями специального вида 85

3.2 Операция определения знака числа в системе остаточных классов 88

3.3 Метод проектирования устройства, реализующего функцию активации ReLU на основе Китайской теоремы об остатках с дробными величинами 96

3.4 Операция сравнения чисел в системе остаточных классов 97

3.5 Метод проектирования устройства, реализующего операцию выбора максимального элемента из окрестности на основе Китайской теоремы об остатках с дробными величинами 102

3.6 Перевод чисел из системы остаточных классов в позиционную систему счисления 103

3.7 Выводы по третьей главе 106

4 АППАРАТНАЯ РЕАЛИЗАЦИЯ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ 108

4.1 Алгоритм проектирования аппаратной реализации сверточной нейронной сети в системе остаточных классов 108

4.2 Архитектура аппаратной реализации сверточной нейронной в системе остаточных классов 114

4.3 Комплекс программ на языке описания аппаратуры для аппаратной реализации сверточной нейронной сети с использованием модулярных вычислений на FPGA 115

4.3 Эксперимент и результаты 118

4.4 Выводы по четвертой главе 121

ЗАКЛЮЧЕНИЕ 123

СПИСОК ЛИТЕРАТУРЫ 126

Статьи автора в журналах, рекомендованных ВАК РФ, Scopus, Web of Science 126

Другие публикации автора по теме диссертации 127

Свидетельства о государственной регистрации программ для ЭВМ 131

Цитируемая литература 132

СПИСОК РИСУНКОВ 142

СПИСОК ТАБЛИЦ 146

ПРИЛОЖЕНИЕ А. Акт о внедрении результатов диссертационного исследования 148

ПРИЛОЖЕНИЕ Б. Свидетельства о государственной регистрации программ для ЭВМ 149

ВВЕДЕНИЕ

Актуальность работы. Современные алгоритмы интеллектуального анализа информации требуют больших вычислительных затрат и ресурсов для их реализации. Для решения данной проблемы на практике зачастую используются удаленные вычислительные ресурсы, такие как большие центры обработки данных [42], облачная инфраструктура [75] и другие. Но такие способы подходят не для всех практических задач, использующих решения на основе искусственного интеллекта на сегодняшний день. Например, для беспилотных транспортных средств [59] необходима автономность системы, так как потеря связи с удаленным сервером может привести к аварии. В связи с этим, существует необходимость в реализации автономной системы интеллектуального анализа визуальной информации [99]. В автономных системах остро стоит проблема потребления большого количества вычислительных ресурсов, которая требует разработки новых подходов к реализации алгоритмов интеллектуального анализа визуальных данных. Одним из основных методов решения задач в области искусственного интеллекта являются глубокие нейронные сети. Нейросетевые интеллектуальные системы используют значительное количество памяти и времени на этапе обучения искусственной нейронной сети, кроме того, на этапе эксплуатации важно обеспечивать работу системы в режиме реального времени. Это побуждает исследователей вести разработку специализированных аппаратных ускорителей нейросетевых вычислений.

Постоянно растущая практическая потребность в увеличении скорости и энергоэффективности работы устройств делает актуальной проблему качественного развития аппаратных ускорителей нейросетевых вычислений для обеспечения быстрой работы и экономически выгодной эксплуатации таких систем. Решение проблемы улучшения технических характеристик аппаратных реализаций глубоких нейронных сетей позволит существенно расширить применение интеллектуальных технических средств в практической деятельности человека. Успешное решение обозначенной

проблемы окажет большое влияние на развитие таких практических приложений как распознавание изображений, анализ речи и создание робототехнических систем.

Современные нейросетевые архитектуры обработки информации позволяют достичь высокого качества работы интеллектуальных систем, однако, происходит это за счет очень высокой вычислительной сложности, приводящей к медленной скорости работы. Аппаратная реализация компонентов нейронной сети позволяет значительно ускорить работу, а также обеспечить автономность системы. В настоящее время существуют аппаратные реализации простых нейронных сетей, в том числе осуществляющих и процесс обучения нейронной сети [94, 100]. Но архитектура глубокой нейронной сети имеет весьма сложную структуру, вследствие чего разработка микроэлектронных устройств на ее основе с аппаратной реализацией процесса обучения является малоисследованной областью [81, 92]. Одним из способов построения эффективной аппаратной реализации глубоких нейронных сетей является применение вычислений в конечных кольцах и полях [83, 84].

Среди различных альтернативных числовых систем особый интерес представляет система остаточных классов (СОК), способная обеспечить параллельную реализацию арифметических операций, в особенности сложения и умножения. СОК представляет собой прямую сумму конечных колец и оперирует с небольшими остатками вместо обработки чисел большой разрядности в позиционных системах счисления, что открывает возможности для параллельной обработки данных на арифметико-логическом уровне [79].

Значительный научный вклад в теорию модулярных вычислений и их приложений внесли отечественные и зарубежные исследователи: И.Я. Акушский, Д.И. Юдицкий, В.М. Амербаев, А.А. Коляда, Н.И. Червяков, О.А. Финько, N. Szabo, D. Svoboda, M. Valach, H.L. Garner, B. Parhami, A. Omondi, A. Premkumar, J. Ramires, L. Sousa и другие.

СОК имеет много приложений в различных областях, связанных с компьютерными системами и вычислительной техникой. Данная система счисления используется для реализации вычислительных алгоритмов цифровой обработки сигналов. В настоящее время активно разрабатываются системы цифровой обработки изображений, основанные на применении СОК [80]. Кроме того, СОК имеет большой потенциал для улучшения аппаратных решений, реализующих криптографические операции, например, обработку чисел большой разрядности в гомоморфной криптографии и системе RSA [93]. Также СОК используется в компьютерных сетях для повышения эффективности хранения и передачи информации и обеспечения ее целостности [51].

Таким образом, существует необходимость качественного улучшения аппаратных реализаций глубоких нейронных сетей и их технических характеристик.

Целью диссертационного исследования является увеличение производительности системы обработки и анализа изображений в условиях ограниченности ресурсов, требующихся для построения и эксплуатации системы.

Объект исследования – системы обработки и анализа изображений.

Предмет исследования – свойства и принципы функционирования систем анализа визуальных данных.

Научная задача – разработка методов проектирования аппаратных ускорителей системы интеллектуального анализа изображений для повышения их производительности.

Для решения поставленной общей научной задачи произведена ее декомпозиция на ряд частных задач:

1. Разработка математической модели сверточных нейронных сетей (СНС) с реализацией вычислений в СОК для создания высокопроизводительных систем интеллектуального анализа изображений.

2. Разработка методов проектирования аппаратных ускорителей компонентов СНС, выполняющих вычисления в СОК, и их реализация на языке описания аппаратуры.

3. Разработка алгоритма проектирования аппаратной реализации СНС с вычислениями в СОК.

4. Разработка программного комплекса на языке описания аппаратуры VHDL для аппаратной реализации СНС для распознавания образов с использованием модулярных вычислений на FPGA (Field-Programmable Gate Array).

Соответствие паспорту научной специальности. Область исследования соответствует паспорту специальности 2.3.5 – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей по следующим пунктам:

7. Модели, методы, архитектуры, алгоритмы, форматы, протоколы и программные средства человеко-машинных интерфейсов, компьютерной графики, визуализации, обработки изображений и видеоданных, систем виртуальной реальности, многомодального взаимодействия в социокиберфизических системах.

8. Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования.

Методы исследования. Для решения поставленных задач были использованы методы алгебры, линейной алгебры, теории чисел, теории модулярных вычислений в системе остаточных классов, теории алгоритмов, методы математического и имитационного моделирования.

Научная новизна:

1. Разработана математическая модель СНС с реализацией вычислений в СОК, которая отличается от известных тем, что использует набор модулей специального вида и не требует выполнения обратного преобразования в

позиционную систему счисления (ПСС) для выполнения вычислительно сложных операций в СОК в скрытых слоях СНС.

2. Разработан метод проектирования сверточных слоев СНС и его программная реализация на языке описания аппаратуры VHDL, отличающийся от известных тем, что основывается на методе Винограда и использует арифметику СОК с модулями специального вида.

3. Разработаны методы проектирования функции активации ReLU (Rectified Linear Unit) и слоя выборки максимального элемента из окрестности, требующие выполнения вычислительно сложных операций в СОК, и их программная реализация на языке описания аппаратуры VHDL.

4. Разработан алгоритм проектирования аппаратной реализации СНС, отличающийся от аналогов тем, что учитывает выбор модулей СОК, способ представления весовых коэффициентов в памяти устройства и оригинальные методы проектирования компонентов СНС.

5. Разработан комплекс программ на языке описания аппаратуры VHDL для аппаратной реализации СНС для распознавания образов с использованием модулярных вычислений на FPGA, которая разработана на основе оригинального алгоритма проектирования СНС в СОК.

Достоверность полученных в диссертации результатов подтверждается результатами математического и имитационного моделирования и анализом эффективности разработанных методов и алгоритмов.

Моделирование и вычислительный эксперимент проведены с использованием языка высокого уровня Python в среде PyCharm для обучения СНС, а также с использованием языка описания аппаратуры VHDL и среды автоматизированного проектирования Xilinx Vivado для аппаратной реализации СНС.

Практическая ценность. Разработанные в рамках диссертационного исследования алгоритмы для проектирования аппаратных ускорителей СНС и комплекс программ, использующих модулярные вычисления, внедрены в

организации ООО «Инфоком-С» в системе интеллектуального реагирования на инциденты и события «Darvis».

Положения, выносимые на защиту:

1. Метод преобразования чисел из ПСС в СОК с модулями специального вида и его программная реализация на языке описания аппаратуры.
2. Метод проектирования устройства, реализующего операцию свертки на основе метода Винограда и СОК с модулями специального вида, и его программная реализация на языке описания аппаратуры.
3. Метод проектирования устройства, реализующего функцию активации ReLU на основе Китайской теоремы об остатках с дробными величинами (КТОд), и его программная реализация на языке описания аппаратуры.
4. Метод проектирования устройства, реализующего операцию выбора максимального элемента из окрестности на основе КТОд, и его программная реализация на языке описания аппаратуры.
5. Алгоритм проектирования аппаратной реализации СНС, учитывающий выбор модулей СОК, способ представления весовых коэффициентов в памяти устройства и методы проектирования компонентов СНС в СОК.
6. Комплекс программ на языке описания аппаратуры для аппаратной реализации СНС с использованием модулярных вычислений на FPGA, который разработан на основе оригинального алгоритма проектирования СНС в СОК.

Внедрение. Результаты диссертационного исследования использованы при выполнении научно-исследовательских проектов Совета по грантам Президента РФ СП-126.2019.5 «Разработка микроэлектронных устройств интеллектуального анализа визуальных данных», МК-3918.2021.1.6 «Высокопроизводительные устройства цифровой обработки медицинских изображений на основе параллельной математики», РФФИ 18-37-20059-мол_а_вед «Разработка перспективной архитектуры на базе ядра векторного процессора для задач обработки и визуализации сигналов», 19-07-00130-А «Экономические средства интеллектуального анализа визуальной

информации на основе сверточных нейронных сетей», Минобрнауки № 13.2251.21.0064 «Фундаментальные алгоритмы, технологии глубокого обучения и безопасности для облачного хранения и обработки данных» и Министерства науки и высшего образования Российской Федерации № 075-15-2020-788 «Исследование и разработка передовых методов защиты информации, сохранения конфиденциальности и предотвращения утечки данных при обработке данных в распределенных средах»..

Апробация работы. Результаты диссертационного исследования прошли апробацию на научных мероприятиях всероссийского и международного уровней: International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON) 2017, 10th Mediterranean Conference on Embedded Computing (MECO) 2021, 9th Mediterranean Conference on Embedded Computing (MECO) 2020, IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus) 2020, International Conference on Engineering and Telecommunication (EnT) 2019, 42nd International Conference on Telecommunications and Signal Processing (TSP) 2019, 23rd Conference of Open Innovations Association (FRUCT) 2018, Third International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC) 2016, форум «Наука будущего – наука молодых» 2017.

Публикации по теме диссертации. Основные результаты исследования отражены в 33 работах, среди которых 12 статей в научных изданиях, входящих в перечень ВАК и системы индексирования научных работ Scopus и Web of Science. Получены 6 свидетельств о государственной регистрации программ для ЭВМ.

Личный вклад соискателя. Все результаты диссертационного исследования получены лично автором. Из результатов работ, выполненных коллективно, в диссертацию включены только полученные непосредственно автором. В работах [3, 12, 14, 20, 21, 24, 28, 34] автором предложены архитектуры аппаратных ускорителей слоев СНС, отвечающих за выделение признаков. В работах [1, 11] предложен метод для проектирования

эффективной аппаратной реализации компонентов СНС в СОК с модулями специального вида. В статьях [2, 6, 7, 10, 13, 27, 16-18, 35, 39] автором представлены методы и алгоритмы увеличения производительности цифровых фильтров, а также архитектуры устройств цифровой фильтрации. В работах [4, 9, 19, 31, 29] автором разработаны подходы к реализации вычислительно сложных операций в СОК, а также спроектированы устройства на основе предлагаемых методов. В статьях [22, 23] автором предложен метод преобразования чисел из ПСС в СОК и его аппаратная реализация на основе параллельно-префиксных сумматоров. В работах [8, 25, 26, 32, 37, 38] автором применен метод квантования коэффициентов масок фильтров для аппаратной реализации цифровых фильтров. В работах [15, 30, 33, 36] рассмотрено применение нейронных сетей и цифровых фильтров для решения практических задач, таких как медицинская диагностика и видеонаблюдение.

Структура диссертации. Диссертационная работа состоит из введения, 4-х глав, заключения, списка литературных источников и 2-х приложений. Содержит 48 рисунков и 19 таблиц. Список используемой литературы содержит 114 источников. В диссертации принята тройная нумерация формул, рисунков и таблиц: первая цифра соответствует номеру главы, вторая – номеру параграфа, а третья – порядковому номеру формулы, рисунка или таблицы внутри данного параграфа.

Краткое содержание работы.

Во введении обоснована актуальность темы диссертации, сформулированы цель и задачи работы, выбраны объект и предмет исследования, показаны научная новизна, практическая ценность полученных результатов, приведены основные положения, выносимые на защиту.

В первой главе представлена математическая модель СНС и проведен сравнительный анализ методов улучшения ее технических характеристик. Предложено использование СОК с модулями специального вида 2^{α} и $2^{\alpha}-1$ для улучшения характеристик аппаратной реализации СНС. Изложены

основные сведение о СОК и сформулирован алгоритм процесса проектирования аппаратной реализации СНС с вычислениями в СОК.

Во второй главе рассмотрена задача увеличения производительности цифровых фильтров, которые применяются для реализации операции свертки в СНС. Предложен метод для проектирования цифровых фильтров на основе метода Винограда и арифметики СОК с модулями специального вида. Рассмотрены случаи масок фильтров размерности 2×2, 3×3 и 5×5, для которых разработаны цифровые фильтры двумерной фильтрации на основе предложенного метода. Результаты теоретического анализа параметров устройств на основе «unit-gate»-модели и аппаратного моделирования на FPGA показали, что цифровые фильтры, разработанные на основе предлагаемого метода, имеют большую производительность по сравнению с рассмотренными известными аналогами.

В третей главе рассмотрены методы реализации вычислительно сложных операций в СОК и предложены устройства для их выполнения. Разработан метод вычисления остатка от деления по модулю $2^{\alpha} - 1$ и предложена архитектура устройства, использующего сумматоры CSA (Carry Save Adder) и KSA (Kogge-Stone Adder). Данное устройство может быть использовано как часть устройства для преобразования чисел из ПСС в СОК с модулями специального вида 2^{α} и $2^{\alpha} - 1$. Предложены архитектуры устройств определения знака числа, сравнения чисел в СОК, а также архитектура устройства обратного преобразования в СОК на основе КТОд. Результаты теоретического анализа и аппаратного моделирования показали, что разработанные архитектуры устройств реализации вычислительно сложных операций в СОК обладают меньшей задержкой и требуют меньше аппаратных ресурсов.

В четвертой главе представлен метод проектирования компонентов СНС с вычислениями в СОК и архитектура СНС, разработанная по предложенному методу. Использование модулей специального вида $\{2^{\alpha_1}, 2^{\alpha_2} - 1, \ldots, 2^{\alpha_n} - 1\}$ позволяет избежать операции деления по модулю. Весовые коэффициенты

сети преобразуются к целочисленному формату. Результаты эксперимента показали, что разрядность весовых коэффициентов может быть уменьшена без потери точности распознавания СНС. В сверточных слоях сети предлагается использовать цифровые фильтры на основе предлагаемого модифицированного метода Винограда с вычислениями в СОК. Устройства, реализующие функции активации ReLU, устройство, реализующее слой max pooling, и устройство обратного преобразования из СОК в ПСС основаны на КТОд. Результаты аппаратного моделирования показали, что использование метода Винограда позволяет увеличить производительность устройства в 4,3-5,56 раз, а применение предлагаемого подхода организации вычислений в СОК увеличивает производительность устройства на 2,84-12,13% по сравнению с рассмотренными аналогами.

В заключении подведены итоги и обобщены результаты проведенных исследований.

В приложениях представлены свидетельства о регистрации программ для ЭВМ по теме диссертации и акт о внедрении результатов диссертационного исследования.

1 МЕТОДЫ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ВИЗУАЛЬНЫХ ДАННЫХ

1.1 Сверточные нейронные сети

Сверточные нейронные сети (СНС) являются перспективным инструментом для решения задачи распознавания изображений. Алгоритмы, основанные на СНС, широко используются во встроенных системах машинного зрения, которые включают в себя решения по распознаванию рукописного текста [73], обнаружению лиц [95], местности [89], распознаванию объектов [57] и другие. Нейронные сети имеют ряд преимуществ, которые отличают их от других подходов в решении проблем машинного обучения. Главным из них является возможность распараллеливания обработки информации и способность к самообучению, т.е. созданию обобщений [57]. Наиболее известные реализации СНС требуют значительного объема памяти для хранения весовых коэффициентов при обучении и эксплуатации [56, 73, 83].

Идея применения искусственный нейронных сетей для обработки визуальной информации была предложена в работе [73], в которой решалась задача автоматизации распознавания рукописных цифр. Предложенная в указанной статье архитектура получила название СНС и главной ее особенностью является объединение сверточных слоев, реализующих извлечение визуальных признаков, и многослойного персептрона, реализующего операцию распознавания по данным результатов сверток. Эволюция данной научной идеи и развитие вычислительной техники привели к тому, что в настоящее время теория СНС и методов ее практического применения идет по пути экстенсивного увеличения количества слоев СНС, что приводит к высокой вычислительной сложности реализации таких систем. Например, архитектура сети [71], показавшая лучший результат распознавания изображений базы ImageNet в 2010 году, содержит около

650000 нейронов, 60 миллионов настраиваемых параметров и требует 27 гигабайт дискового пространства для обучения. В статье [98] представлена разработка компании Google, которая показала лучший результат распознавания изображений базы ImageNet в 2014 году. Для распознавания изображения данная СНС выполняет более полутора миллиардов вычислительных операций, что мотивировало компанию Google разработать специальный тензорный процессор для оптимизации работы данной СНС [68].

СНС состоит из входного и выходного слоев, а также из нескольких скрытых слоев. Скрытые слои СНС состоят из сверточных слоев, слоя подвыборки (pooling) и полносвязных слоев нейронов.

Предположим, что на вход сверточного слоя поступает набор из D карт признаков $I_{in}(x, y, z)$, состоящих из R строк, C столбцов, где $0 \le x < R$, $0 \le y < C$ и $0 \le z < D$ – пространственные координаты. Процедура получения одной карты признаков в сверточном слое может быть представлена в виде

$$I_f(x, y) = b + \sum_{i=-t}^{t} \sum_{j=-t}^{t} \sum_{z=0}^{D-1} W(i, j, z) \cdot I_{in}(x + i, y + j, z), \qquad (1.1.1)$$

где I_f – карта признаков после свертки, $W(i, j, z)$ – это коэффициенты 3D-фильтра размерности $d \times d$ для обработки D двухмерных массивов, $t = \left\lfloor \frac{d}{2} \right\rfloor$ и b – смещение [28]. На выходе сверточного слоя применяется функция активации. Наиболее распространенной функцией является линейный выпрямитель (ReLU), которая формирует карты признаков $F(x, y)$ [42]:

$$F(x, y) = \max\left(0, I_f(x, y)\right). \qquad (1.1.2)$$

Процедура получения одной карты признаков показана на рисунке 1.1.1. Количество карт признаков на выходе сверточного слоя соответствует количеству масок фильтров.

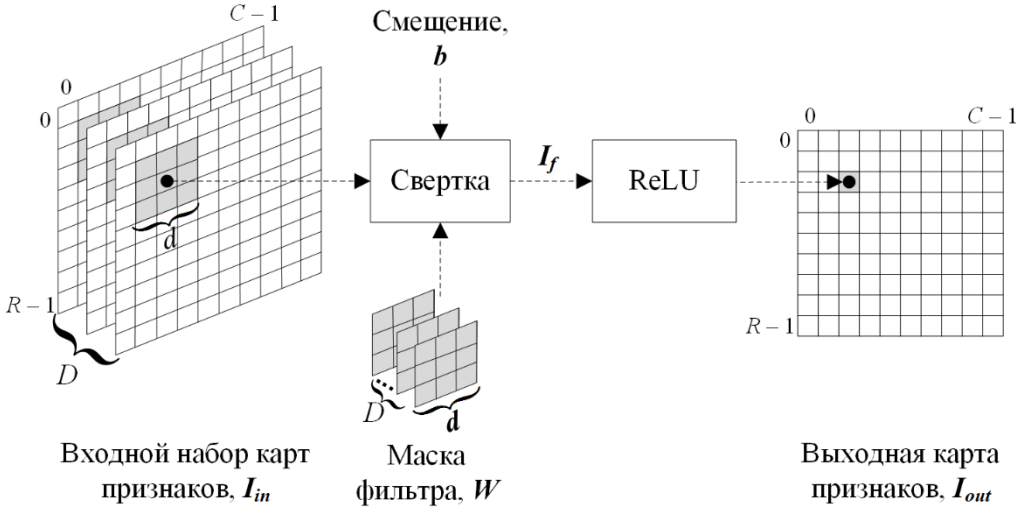


Рисунок 1.1.1. Процедура получения карт признаков

СНС обычно использует большое количество фильтров в сверточном слое. Это приводит к резкому увеличению объема обрабатываемых данных в сети. Для их уменьшения используется слой выборки.

Существуют разные подходы к реализации данного слоя: выбор медианного элемента, выбор максимального элемента, вычисление среднего арифметического всех элементов. Чаще всего используется выбор максимального элемента некоторой рассматриваемой окрестности. Следовательно, вход данного слоя можно описать как трехмерную функцию $P_{in}(x_{in}, y_{in}, z)$, где $0 \le x_{in} < R$, $0 \le y_{in} < C$ и $0 \le z < D$. Процедуру выбора максимального элемента из окрестности размером $s \times s$ с шагом s для z-ой карты признаков можно описать формулой

$$P_{out}(x_{out}, y_{out}, z) = \max_{\substack{s \cdot x_{out} \le x_{in} \le s \cdot (x_{out}+s)-1, \\ s \cdot y_{out} \le y_{in} \le s \cdot (y_{out}+s)-1}} \{P_{in}(x_{in}, y_{in}, z)\}, \quad (1.1.3)$$

где $P_{out}(x_{out}, y_{out}, z)$ – набор из D карт признаков на выходе сверточного слоя, $0 \le x_{out} \le \frac{R}{s} - 1$, $0 \le y_{out} \le \frac{C}{s} - 1$.

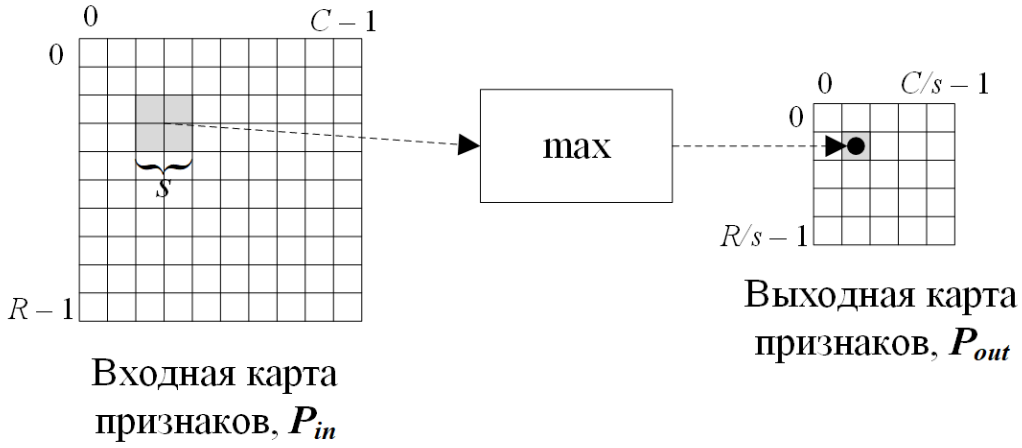


Рисунок 1.1.2. Процедура выбора максимальных элементов для карт признаков

Заключительными слоями сети являются полносвязные слои нейронов, выполняющие функцию классификатора [60]. Пусть $\{x_i\}$ – вектор, подаваемый на вход $(p-1)$-го слоя, где $0 \le i < m$ и m – общее число входов. Каждый элемент вектора умножается на соответствующий весовой коэффициент вектора $W_j = \{w_{ji}\}$, $0 \le i < m$, $0 \le j < n$, n – количество нейронов $(p-1)$-го слоя, и результат суммируется

$$y_j = \sum_{i=0}^{m-1} w_{ji} x_i. \tag{1.1.4}$$

Обозначим функцию активации для $(p-1)$-го слоя как $\phi(t)$, тогда результатом $(p-1)$-го слоя является вектор $\{h_j\}$, $0 \le j < n$, элементы которого вычисляются как

$$h_j = \phi(y_j). \tag{1.1.5}$$

Результат вычислений $(p-1)$-го слоя подается на вход p-го слоя, производится процедура умножения на соответствующие весовые коэффициенты, произведения складываются и подаются на функцию активации, результатом вычислений является вектор $\{z_k\}$, $0 \le k < l$, l – количество нейронов p-го слоя. Результат последнего слоя нормализуется с помощью функции softmax, таким образом, на выходе полносвязных слоев

формируется вектор $\{g_k\}$, $0 \leq k < l$, элементы которого вычисляются следующим образом:

$$g_k = \frac{e^{z_k}}{\sum_{i=0}^{l-1} e^{z_i}}. \quad (1.1.6)$$

На выходе полносвязного слоя формируется вектор, количество элементов которого соответствует количеству классов и отображает вероятность принадлежности образа, подаваемого на вход сети, к каждому классу.

Обобщая сказанное, можно сделать вывод о том, что современные архитектуры СНС являются весьма ресурсозатратными, что ограничивает возможности их широкого практического применения. В настоящее время нет единого подхода к снижению ресурсных затрат при реализации СНС на практике.

1.2 Методы повышения производительности нейронных сетей

Научная задача, рассматриваемая в данной работе, заключается в необходимости разработки методов и алгоритмов проектирования аппаратных ускорителей нейросетевых устройств. Потребность в разработке специализированных устройств нейросетевой обработки данных обусловлена постоянным усложнением архитектуры нейронных сетей и недостаточной производительностью процессоров общего назначения (CPU – Central Processing Unit) и графических ускорителей (GPU – Graphics Processing Unit) для решения задач такого рода [97].

Аппаратная реализация компонентов нейронной сети позволяет сократить временные и аппаратные затраты, а также обеспечить автономность системы. Аппаратные реализации СНС являются сложными и, как правило, процесс обучения производится до аппаратного проектирования. Аппаратное проектирование глубоких СНС сопряжено с проблемами, вызванными высокой вычислительной сложностью. В работах [106] и [107] предлагается

новый подход к снижению сложности операции свертки в СНС. Для аппаратной реализации свертки предлагается использовать энергоэффективные быстрые блоки свертки, состоящие из параллельных быстрых фильтров с конечной импульсной характеристикой (КИХ).

Для того, чтобы нейронная сеть могла быть адаптирована под разные задачи, она должна обладать гибкой архитектурой. В работе [90] представлена реализация многослойного персептрона на языке описания аппаратуры VHDL (Very-High-Speed-Integrated-Circuits-Program Hardware Description Language). В предлагаемом решении весовые коэффициенты представляются в формате с плавающей точкой, модель является гибкой и такие параметры как количество слоев и нейронов могут быть определены пользователем.

Для увеличения быстродействия систем широко используются методы параллельных вычислений. Статья [92] посвящена созданию быстрого и точного универсального нечеткого нейросетевого контроллера Neuro-Fuzzy на основе FPGA. Для достижения этой цели была создана библиотека VHDL. Предложенный контроллер сравнивается с его программной реализацией на основе микропроцессора. Результаты показали, что аппаратные контроллеры быстрее и точнее программных. На этапе обучения используются алгоритмы линейного обучения для оценки структуры контроллера и значений его параметров.

Кроме того, при аппаратной разработке нейронной сети часто требуется производить аппроксимацию функций активации. В работе [100] предлагается аппаратно-эффективная реализация сигмоидальной функции с регулируемой точностью для нейронных сетей и приложений глубокого обучения. Авторами разработан тестовый модуль для данного решения.

В статье [112] представлена аппаратная архитектура слоя softmax глубокой нейронной сети. Для обучения нейронной сети в работе [54] предлагается использовать упрощенную функцию активации. В статье представлена аппаратная реализация нелинейных функций активации нейрона для применения в различных искусственных нейронных сетях, в том числе

вейвлет-нейронных сетях. Подобные решения могут также использоваться в нечетких нейронных сетях.

Аппаратное проектирование СНС затруднительно в силу высокой вычислительной сложности таких структур. В работе [74] представлен ускоритель СНС, который оптимизирует использование оборудования и пропускную способность данных с помощью гибкой и эффективной архитектуры для достижения ускорения СНС в режиме реального времени. В последнее время становятся все более востребованными реализации СНС в режиме реального времени в системах с ограниченными ресурсами, например, на основе моделей FPGA низкой ценовой категории. Ограниченная пропускная способность и встроенная память являются узкими местами на пути ускорения СНС. В статье [107] предлагаются эффективные аппаратные архитектуры для ускорения глубоких моделей СНС. На основе КИХ-фильтров разработаны блоки быстрой свертки. Предложены новые схемы хранения и повторного использования данных, в которых все промежуточные пиксели хранятся на кристалле, а требования к полосе пропускания уменьшаются. В работе в качестве примера выбрана одна из самых крупных и точных сетей VGG16 и реализована на платах Xilinx Zynq ZC706 и Virtex VC707.

В статье [60] предлагается новая более быстрая архитектура СНС на основе FPGA, которая размещает все слои на кристалле и одновременно работает как конвейер. Авторами предложена комплексная методология картирования, позволяющая обеспечить оптимальное использование ресурсов и вычислительную эффективность. Кроме того, для облегчения процесса программирования предлагается среда проектирования, которая предоставляет разработчикам функцию для создания ускорителя с предлагаемой методологией оптимизации.

В работе [96] рассматриваются современные методы для ускорения сетей глубокого обучения на FPGA. Авторы выделили ключевые функции, используемые в различных методах повышения производительности и

ускорения, и предоставили рекомендации по расширению использования FPGA для ускорения СНС.

Еще одним подходом к улучшению производительности СНС является использование альтернативных непозиционных систем счисления. В работах [83, 84] предлагается использование вычислений в конечных кольцах и полях для улучшения технических характеристик устройства на основе СНС. Но предлагаемый авторами подход с использованием вычислений во вложенной СОК требует слишком большого объема памяти. В работе [91] авторы предлагают реализацию нейронной сети с использование СОК RNSnet. Предлагаемая архитектура сравнивалась с несколькими популярными реализациями нейронных сетей и показала хорошие результаты в скорости работы и энергопотреблении системы по сравнению с современными ускорителями нейронных сетей на графических процессорах NVIDIA GTX 1080.

Для автоматизации процессов проектирования и обучения нейронных сетей разработан ряд программных библиотек от различных производителей. TensorFlow является библиотекой для численных расчетов, но главной целью ее создания была помощь в проектировании и обучении моделей машинного обучения [40]. Она обладает интерфейсом для работы с Python и C++. Данная библиотека выражает математические вычисления в виде направленных графов Directed Acyclic Graph (DAG) и может компилировать DAG на CPU и GPU. Для ускорения нейросетевых вычислений, описываемых библиотекой TensorFlow, компания Google разработала специализированный тензорный процессор (TPU – Tensor Processing Unit), основным модулем которого является массивный матричный умножитель [68].

Известен набор инструментов с открытым исходным кодом под названием LeFlow, который позволяет разработчику программного обеспечения автоматически преобразовывать вычисления Tensorflow в код, написанный на языке описания аппаратуры Verilog, с помощью высокоуровневого инструмент синтеза с открытым исходным кодом LegUp

[86]. Этот процесс позволяет разработчикам Python создавать прототипы алгоритмов машинного обучения на FPGA, не беспокоясь о деталях аппаратного проектирования или кода C, оптимизированного с использованием аппаратных директив. Данная возможность позволяет автоматизировать процесс проектирования для разработчика, но сгенерированная аппаратная модель будет уступать по производительности модели, оптимизированной вручную [86].

Библиотека Caffe написана на языке C++, но имеет интерфейсы для работы с Matlab и Python [67]. В отличии от Tensorflow, она предназначена только для разработки глубоких нейронных сетей и объединила все современные методы для проектирования СНС. Caffe поддерживает выполнение вычислений на CPU и GPU, что позволяет выполнять обучение нейронной сети на компьютере с графическим процессором, а затем разворачивать ее на кластерах или мобильных устройствах. Существует версия Caffe с поддержкой FPGA для следующих операций, используемых в СНС: прямая и обратная свертка, ReLU, max pooling, скалярное произведение векторов [52, 53]. Эта версия взаимодействует только со средой Xilinx SDAccel OpenCL и используют арифметику с плавающей точкой заданной точности для уменьшения площади устройства и улучшения пропускной способности. Недостатком представленной библиотеки является поддержка лишь небольшого количества операций.

Компания Intel выпустила набор инструментов OpenVINO для развертывания нейросетевых приложений и решений в области компьютерного зрения [66]. Этот инструментарий основан на СНС и расширяет рабочие нагрузки компьютерного зрения на аппаратное обеспечение фирмы Intel. Набор инструментов последней версии программы Intel OpenVINO включает в себя инструменты для глубокого обучения Intel Deep Learning Deployment Toolkit и поддерживает лишь ограниченное количество моделей FPGA.

Высокоуровневый язык и интерактивная среда для программирования, численных расчетов и визуализации результатов MATLAB обладает мощным инструментом по работе с нейронными сетями Neural Network Toolbox, который позволяет автоматизировать процесс проектирования и обучения нейронных сетей [76]. В новых версиях продукта также осуществляется поддержка CHC. Кроме того, в Matlab существует возможность конвертирования кода, написанного на встроенном языке программирования, в код на языке описания аппаратуры VHDL, с помощью инструмента HDL Coder для автоматизации процесса проектирования FPGA [3].

Известен бесплатный пакет с открытым исходным кодом для использования Python в качестве языка описания и проверки аппаратуры MyHDL [82]. Python является популярным языком высокого уровня, и разработчики оборудования могут использовать его возможности для моделирования своих проектов. Проекты MyHDL могут быть преобразованы в Verilog или VHDL с учетом некоторых ограничений. Этот пакет обеспечивает традиционный процесс проектирования, включая синтез и реализацию.

Intel HLS Compiler – это инструмент высокоуровневого синтеза (HLS – High-Level Synthesis), который принимает в качестве входных данных код C ++ и генерирует код уровня передачи регистров (RTL – Register Transfer Level) производственного качества, оптимизированный для FPGA Intel [62]. Этот инструмент ускоряет время проверки по RTL, повышая уровень абстракции для аппаратного проектирования FPGA. Модели, разработанные в C ++, обычно проверяются на порядки быстрее, чем RTL.

Xilinx Vivado High-Level Synthesis, включенный в качестве бесплатного обновления во всех выпусках Vivado HLx, ускоряет создание IP (Intellectual Property), позволяя напрямую направлять спецификации C, C ++ и System C на программируемые устройства Xilinx без необходимости вручную создавать RTL [105]. Поддерживая среды проектирования ISE и Vivado, Vivado HLS

предоставляет системным и проектным разработчикам более быстрый путь для создания проектов.

Отдельно стоит отметить существующие нейрочипы, предназначенные для реализации глубоких нейронных сетей. Компания Xilinx разработала программируемый энергоэффективный процессор Xilinx Deep Neural Network (xDNN), работающий на специализированных платах Xilinx Alveo [110]. Механизм xDNN интегрируется в популярные платформы машинного обучения, такие как Caffe, MxNet и TensorFlow, через программный стек Xilinx xfDNN. Процессор xDNN, работающий на базе Alveo, способен обрабатывать 4000 или более изображений в секунду для архитектуры GoogLeNet v1, что в 4 раза больше по сравнению с самым быстрым GPU и в 20 больше по сравнению с лучшим CPU на момент публикации. Xilinx предлагает среду разработки Vitis AI (Artificial Intelligence) для реализации алгоритмов искусственного интеллекта на аппаратных платформах Xilinx, включая как периферийные устройства, так и карты Alveo [104]. Разработка поддерживает основные структуры и новейшие модели, способные решать разнообразные задачи глубокого обучения, предоставляет полный набор предварительно оптимизированных моделей, которые готовы к развертыванию на устройствах Xilinx и обеспечивает инструменты квантования, поддерживающие калибровку и точную настройку нейросетевой модели. Библиотека AI предлагает высокоуровневые API (Application Programming Interface) C++ и Python для переносимости проектов. Масштабируемые IP-ядра Xilinx могут быть настроены в соответствии с потребностями пользователя для различных приложений, с учетом требований к пропускной способности, задержке и энергопотреблению.

Компания Intel предлагает нейросетевой процессор Nervana NNP-T, созданный для быстрого обучения нейронной сети в условиях ограниченной мощности [65]. Другой разработкой Intel является Nervana NNP-I, разработанный для логического вывода и ускорения обучения [64]. Кроме того, указанная разработка обеспечивает высокую степень

программируемости без ущерба для производительности и энергоэффективности, имеет низкую задержку срабатывания и поддерживает все основные среды глубокого обучения. NNP-I обеспечивает увеличение пропускной способности в 1,6 раза по сравнению с GPU Nvidia T4. Intel Movidius Myriad X оснащен специально разработанным аппаратным ускорителем Neural Compute Engine, предназначенным для повышения производительности глубоких нейронных сетей без ущерба для энергопотребления [63]. Neural Compute Engine может быстро выполнять вычисления, необходимые для глубоких нейронных сетей при помощи массива блоков умножения с накоплением (MAC – Multiply-ACcumulate) и прямого взаимодействия с интеллектуальной матрицей памяти. Myriad X обеспечивает 10-кратную производительность по сравнению с предыдущими поколениями.

Энергоэффективный ускоритель нейронных сетей Eyeriss поддерживает современные архитектуры CHC, которые имеют много слоев, миллионы весовых коэффициентов фильтров и различные формы (размеры фильтров, количество фильтров и каналов) [55]. Тестовый чип имеет пространственный массив из 168 обрабатывающих элементов, на входы которых поступает реконфигурируемая многоадресная сеть на кристалле, которая обрабатывает множество форм и сводит к минимуму перемещение данных, используя их повторно. Сбор данных и сжатие данных используются для снижения потребления энергии. Чип полностью интегрирован со средой глубокого обучения Caffe и может выполнять задачу классификации изображений, принадлежащих 1000 классам в режиме реального времени с использованием предварительно обученной сети AlexNet, которая работает в системе Eyeriss Caffe. Eyeriss работает с AlexNet с частотой 35 кадров в секунду при энергопотреблении 278 мВт, что в 10 раз более эффективно по сравнению с мобильными GPU.

Все перечисленные коммерческие решения основаны на организации вычислений в двоичном дополнительном коде. В данном исследовании

предлагается использовать арифметику СОК для эффективной реализации наиболее проблемных и затратных участков матричного умножения в нейроускорителях. Реализация глубоких нейронных сетей с использованием СОК имеет ряд особенностей, требующих организации модульных вычислений в конечных кольцах, что не предусмотрено известными библиотеками. Поэтому в работе будет уделено внимание реализации архитектуры СНС с вычислениями в СОК на языке описания аппаратуры. Таким образом, задача исследования сводится к увеличению производительности СНС при сохранении высокого качества распознавания, в условиях ограниченных аппаратных ресурсов.

1.3 Применение системы остаточных классов для оптимизации вычислений в сверточной нейронной сети

В СОК целое число A представляется как набор остатков от деления $\{a_1, a_2, \ldots, a_n\}$ на соответствующие попарно взаимно простые модули $\{p_1, p_2, \ldots, p_n\}$, так что $a_i = |A|_{p_i}$ [87]. Произведение всех модулей $P = \prod_{i=1}^{n} p_i$ называется динамическим диапазоном СОК. Арифметические операции над числами, представленными в СОК, производятся параллельно по каждому модулю

$$A * B = \left(|a_1 * b_1|_{p_i}, |a_2 * b_2|_{p_2}, \ldots, |a_n * b_n|_{p_n}\right), \tag{1.3.1}$$

где $*$ означает операцию сложения, вычитания или умножения.

Число $A = \{a_1, a_2, \ldots, a_n\}$ может быть преобразовано из СОК в позиционную систему счисления (ПСС) с использованием Китайской теоремы об остатках (КТО) [87]:

$$A = \left| \sum_{i=1}^{n} \left(\left| \left| P_i^{-1} \right|_{p_i} a_i \right|_P P_i \right) \right|_P, \tag{1.3.2}$$

где $P_i = \frac{P}{p_i}$, а $\left|P_i^{-1}\right|_{p_i}$ – мультипликативный обратный элемент для P_i.

Вид модулей СОК влияет на производительность вычислений, поэтому их выбор является важной задачей при проектировании прикладных систем, использующих арифметику СОК. Во-первых, набор модулей должен обеспечивать достаточный динамический диапазон системы для однозначного представления чисел в СОК. Во-вторых, модули должны быть сбалансированы таким образом, чтобы время выполнения операций по каждому каналу было примерно одинаково, и не было длительного простоя системы по какому-либо вычислительному каналу. Наконец, модули СОК специального вида 2^{α} и $2^{\alpha} \pm 1$, $\alpha \in N$, где N – множество натуральных чисел, позволяют избежать затратной по ресурсам операции деления по модулю. Проектирование арифметических устройств по модулю вида 2^{α} сводится к реализации α-битных устройств в двоичной системе счисления. Но требование попарной взаимной простоты модулей СОК позволяет использовать лишь один модуль вида 2^{α}, так как остальные модули должны быть нечетными. Использование модулей вида 2^{α} и $2^{\alpha} \pm 1$ позволяет использовать эффективные техники суммирования по модулю, аналогичные используемым в традиционной двоичной системе счисления [103]. Но практическая реализация вычислительного канала по модулю $2^{\alpha} + 1$, $\alpha \in N$, требует введения дополнительной логики по методу «diminished-1» для отслеживания нулевой кодовой комбинации, что является нежелательным явлением при разработке системы с минимальными аппаратными и временными затратами [78, 114]. Таким образом, модули вида $2^{\alpha} - 1$ предпочтительнее для использования в качестве нечетных модулей системы.

Применение арифметики СОК целесообразно в системах с преобладающей долей операций сложения и умножения. Такие операции как определение знака числа, сравнение чисел и деление являются вычислительно сложными в СОК, так как требуют вычисления позиционной характеристики числа, и могут снизить производительность всей системы. Поскольку в СНС основная вычислительная нагрузка содержится в сверточных слоях,

использующих операции сложения и умножения, то применение СОК увеличивает производительность сети, несмотря на наличие вычислительно сложных операций в других слоях СНС, таких как max pooling, и при реализации функций активации.

В общем виде алгоритм проектирования аппаратной реализации СНС в СОК состоит из следующих этапов:

1) выбор набора модулей СОК;
2) разработка блока преобразования из ПСС в СОК;
3) реализация блоков, выполняющих операцию свертки в СОК параллельно по каждому модулю системы;
4) квантование весовых коэффициентов, представленных действительными числами;
5) реализация блоков, выполняющих вычисление функций активации;
6) реализация блоков, выполняющих операцию выбора максимального элемента окрестности (блок max pooling);
7) разработка блока обратного преобразования из СОК в ПСС.

Далее в работе будет подробно рассмотрен каждый этап представленного алгоритма проектирования аппаратной реализации СНС в СОК.

1.4 Выводы по первой главе

В первой главе представлена математическая модель СНС и обзор методов улучшения ее технических характеристик. Анализ современного состояния исследований по теме диссертации показал, что современные архитектуры СНС являются весьма ресурсозатратными, и нет единого подхода к снижению ресурсных затрат при реализации СНС на практике. Одним из способов увеличения производительности устройств является использование альтернативных непозиционных систем счисления.

В работе предложено использование СОК с модулями специального вида 2^{α} и $2^{\alpha}-1$ для улучшения характеристик СНС при ее аппаратной реализации. В первой главе изложены основные сведения о СОК и сформулирован алгоритм проектирования аппаратной реализации СНС с вычислениями в СОК. Реализация глубоких нейронных сетей с использованием СОК имеет ряд особенностей, таких как, выбор модулей системы, преобразование чисел в СОК и обратно и реализация вычислительно сложных операций в СОК.

В следующих главах представлены методы для проектирования компонентов СНС в СОК с модулями специального вида 2^{α} и $2^{\alpha}-1$ и их аппаратные реализации.

2 ОПЕРАЦИЯ СВЕРТКИ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ

2.1 Цифровая фильтрация изображений

Операцию свертки в сверточном слое СНС (1.1.1) можно рассматривать как фильтрацию D изображений в оттенках серого I_ρ размера $R \times C$. Изображение обрабатывается фильтром W размером $k \times k$. После фильтрации изображения I_ρ фильтром размером $k \times k$ на выходе формируется изображение I_f размером $R \times C$ следующим образом

$$I_f(x,y) = \sum_{i=-t}^{t} \sum_{j=-t}^{t} W(i,j) \cdot I_\rho(x+i+t, y+j+t), \qquad (2.1.1)$$

где $0 \le x < R,\ 0 \le y < C,\ t = \left\lfloor \frac{k}{2} \right\rfloor$. Отметим, что для фильтрации границ изображения левее, правее, ниже и выше границ добавляются по t нулей, и в формуле (2.1.1) подразумевается, что пиксели входного изображения I_ρ перенумерованы с учетом добавленных нулей. Схема процесса фильтрации изображения представлена на рисунке 2.1.1.

Как видно из формулы (2.1.1), операции умножения и сложения лежат в основе цифровой фильтрации. Основная вычислительная нагрузка при линейной фильтрации заключается в многократном выполнении операции умножения. Аппаратная реализация процесса фильтрации позволяет увеличить скорость обработки изображений [85]. В связи с этим, исследователи ведут разработки в области проектирования эффективных, с точки зрения скорости, энергопотребления и площади, сумматоров и умножителей. В работе [101] авторы предлагают новое устройство умножения с накоплением. Авторы статьи [6] представили новое усеченное устройство умножения с накоплением для реализации цифровых фильтров с конечной импульсной характеристикой.

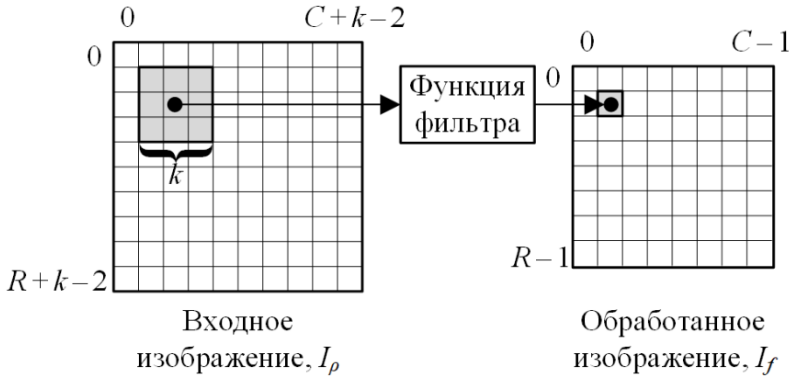


Рисунок 2.1.1. Фильтрация изображения

Одним из подходов к увеличению быстродействия цифрового фильтра является уменьшение количества операций умножения. В работе [108] предложен метод фильтрации Винограда, который позволяет сократить количество умножений в процессе фильтрации, за счет увеличения количества сложений. В статье [72] авторы предлагают использовать алгоритм Винограда для реализации операции свертки в сверточной нейронной сети при выполнении вычислений на графическом процессоре. Авторы статьи [111] предлагают аппаратный ускоритель сверточной нейронной сети, реализованный на FPGA с применением алгоритма Винограда.

Еще одним подходом к увеличению производительности, то есть количества обработанных кадров в единицу времени, цифровой фильтрации является использование непозиционной системы счисления – СОК [80]. Применение такого подхода позволяет увеличить скорость выполнения модульных операций сложения и умножения, являющихся основой для выполнения цифровой фильтрации. В работе [7] авторы предлагают архитектуру усеченного блока умножения с накоплением, использующего арифметику СОК. В статье [50] представлен метод сглаживающей фильтрации изображений с использованием СОК. Авторами предлагается метод квантования коэффициентов маски фильтра. В работе [102] предлагается архитектура устройства для сглаживания изображений и выделения контуров, которая использует трехмодульную СОК. Авторы статьи [48] предлагают

методологию построения цифровых фильтров в СОК для автоматизации процесса проектирования и обеспечения сбалансированности устройств по скорости и энергоэффективности. В работе [43] представлена аппаратная реализация фильтра с конечной импульсной характеристикой на FPGA в СОК с набором модулей специального вида.

Фильтрация изображений по формуле (2.1.1) в СОК выполняется в несколько этапов (рис. 2.3.1). Сначала необходимо перевести данные из ПСС в СОК. Затем, производится фильтрация параллельно по нескольким вычислительным каналам, которые соответствуют модулям СОК. Далее производится обратное преобразование из СОК в ПСС.

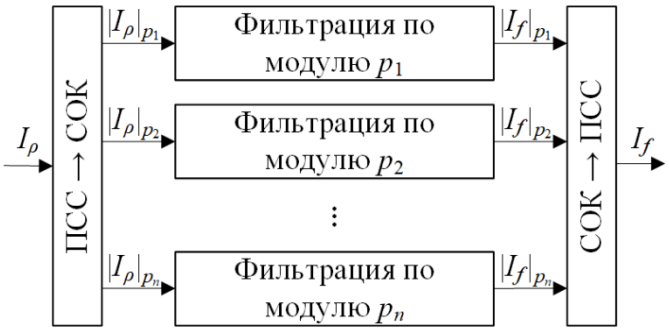


Рисунок 2.1.2. Фильтрация изображения в СОК

Как говорилось ранее, вид модулей СОК влияет на производительность вычислений, поэтому их выбор является важной задачей при проектировании прикладных систем, использующих арифметику СОК. Для реализации цифровых устройств в СОК предлагается использовать модули специального вида 2^α и $2^\alpha - 1$.

2.2 Архитектура цифровых фильтров

Инструментом реализации цифровой фильтрации являются цифровые фильтры, которые принято делить на фильтры с КИХ и фильтры с бесконечной импульсной характеристикой (БИХ). На вход КИХ цифрового

фильтра подается последовательность отсчетов сигнала $X(n)$, формируемая аналого-цифровым преобразователем из аналогового сигнала либо поступающая по вычислительной шине из цифрового источника. На выходе КИХ фильтра формируется сигнал $Y(n)$, определяемый формулой

$$Y(n) = \sum_{i=0}^{P} f_i X(n-i), \qquad (2.2.1)$$

где f_i – коэффициенты фильтра, P – порядок фильтра.

На рисунке 2.2.1 представлена архитектура КИХ фильтра. Введем для данного устройства обозначение FIR (Finite Impulse Response). На вход устройства поступает последовательность отсчетов сигнала $X(n)$ и коэффициенты фильтра f_i, а на выходе формируется сигнал $Y(n)$. Операция умножения с накоплением по формуле (2.2.1) производится с помощью MAC блоков, изображенных на рисунке 2.2.2. Устройство MAC состоит из блока генерации частичных произведений PPG (Partial Product Generator), который сформирован из массива вентилей AND [88] и сумматора со множественным входом MOA (Multi-Operand Adder).

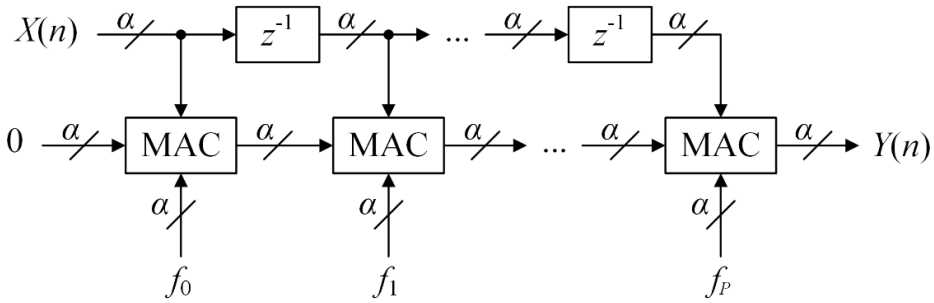


Рисунок 2.2.1. Архитектура α-разрядного устройства FIR порядка P

Блоки MOA могут быть реализованы с помощью дерева сумматоров. В данной работе используются сумматоры с сохранением переноса CSA (Carry-Save Adder) [88], которые преобразуют операцию сложения трех чисел, к сложению двух чисел. Результат дерева сумматоров CSA-tree складывается с помощью параллельно-префиксного сумматора Когге-Стоуна KSA [69].

Архитектура устройства MOA представлена на рисунке 2.2.3. На вход устройства подается последовательность слагаемых $\{P_i\}$, где $0 \leq i \leq n$, а на выход поступает сумма S.

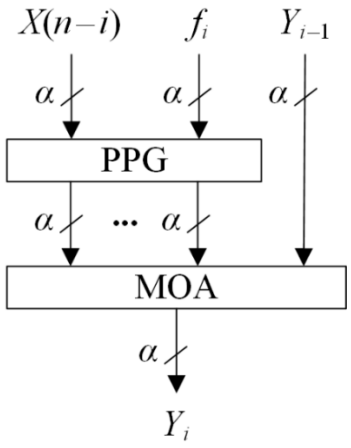


Рисунок 2.2.2. Архитектура α-разрядного устройства умножения с накоплением MAC

Частным случаем цифровой обработки сигналов является цифровая фильтрация изображений, которая широко используется во многих практических приложениях. Далее будут представлены методы повышения производительности цифровых фильтров для фильтрации изображений.

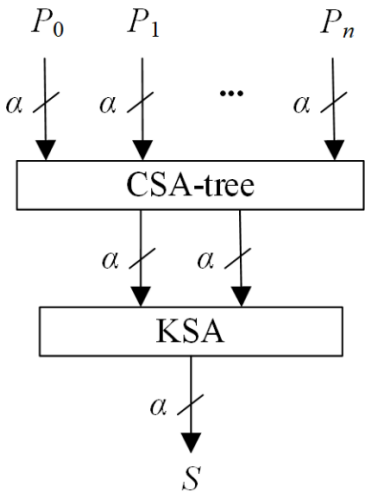


Рисунок 2.2.3. Архитектура α-разрядного устройства MOA

2.3 Метод проектирования устройства, реализующего операцию свертки на основе метода Винограда и СОК с модулями специального вида

Учитывая вышеописанные преимущества метода Винограда и СОК для увеличения быстродействия цифровой фильтрации, предлагается новый метод проектирования устройства, реализующего операцию свертки на основе метода Винограда и СОК с модулями специального вида 2^{α} и $2^{\alpha}-1$.

Одномерная фильтрация по методу Винограда в матричной форме имеет вид

$$z = A^{\mathrm{T}}\left((Gw) \odot (B^{\mathrm{T}}d)\right), \tag{2.3.1}$$

где оператор $\odot$ обозначает поэлементное умножение матриц, A, G и B – матрицы преобразования, w – маска одномерного фильтра, d – вектор данных, z – результат фильтрации [72]. Алгоритм одномерной фильтрации по методу Винограда принято обозначать $F(n,k)$, где n – размер вектора z, а k – размер маски фильтра w.

Двумерная фильтрация по методу Винограда в матричной форме имеет вид [72]:

$$Z = A^{\mathrm{T}}\left((GWG^{\mathrm{T}}) \odot (B^{\mathrm{T}}DB)\right)A, \tag{2.3.2}$$

где W – двумерная маска фильтра, D – двумерный массив данных и Z – двумерный массив результата фильтрации. Алгоритм двумерной фильтрации по методу Винограда принято обозначать $F(n \times n, k \times k)$.

Разделим изображение на фрагменты D размера $m \times m$, $m > k$. Каждый фрагмент обрабатывается фильтром w размерности $k \times k$ по методу Винограда $F(n \times n, k \times k)$ с шагом n по каждому измерению.

Процедура двумерной фильтрации по методу Винограда, описанная формулой (2.3.2), реализует обработку сигнала в несколько этапов (рис. 2.3.1). Результат преобразования маски фильтра обозначим как $U = GWG^{\mathrm{T}}$. Так как коэффициенты фильтра являются константами, то данное преобразование

может быть выполнено один раз предварительно, а значит не несет в себе вычислительной нагрузки. Результат преобразования входных данных D обозначим как $V = B^{\mathrm{T}}DB$, а результат поэлементного умножения матриц как $M = U \odot V$. Тогда, учитывая введенные обозначения, формула (2.3.2) принимает вид $Z = A^{\mathrm{T}}MA$.

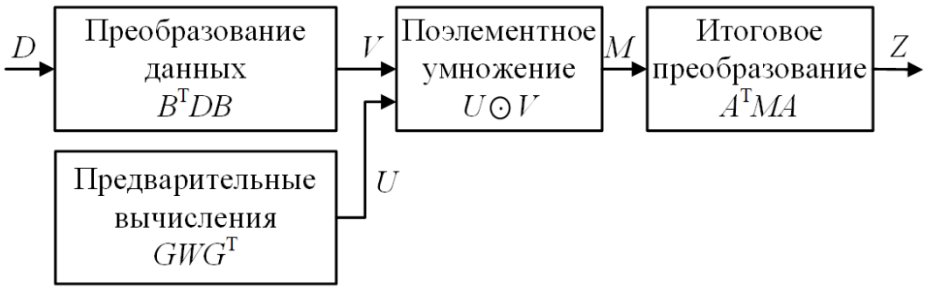


Рисунок 2.3.1. Этапы фильтрации фрагмента изображения по методу Винограда

На рисунке 2.3.2. представлена архитектура предлагаемого устройства $\mathrm{F}(n \times n, k \times k)_{\mathrm{RNS}}$ для фильтрации по методу Винограда в СОК с набором модулей $\{2^{\alpha_1}, 2^{\alpha_2} - 1, \dots, 2^{\alpha_\eta} - 1\}$. На вход устройства подается массив данных $D_{RNS} = \{|D|_{2^{\alpha_1}}, |D|_{2^{\alpha_2}-1}, \dots, |D|_{2^{\alpha_\eta}-1}\}$, представленный в СОК. Затем, входные данные обрабатываются параллельно устройствами $\mathrm{F}(n \times n, k \times k)_{2^{\alpha_1}}$, $\mathrm{F}(n \times n, k \times k)_{2^{\alpha_2}-1}$, …, $\mathrm{F}(n \times n, k \times k)_{2^{\alpha_\eta}-1}$, которые формируют на выходе устройства $\mathrm{F}(n \times n, k \times k)_{\mathrm{RNS}}$ массив обработанных данных $Z_{RNS} = \{|Z|_{2^{\alpha_1}}, |Z|_{2^{\alpha_2}-1}, \dots, |Z|_{2^{\alpha_\eta}-1}\}$.

В следующих параграфах рассмотрены случаи применения предлагаемого метода для проектирования фильтров с масками размерности 2×2, 3×3 и 5×5, реализующих операцию свертки на основе метода Винограда и СОК с модулями специального вида.

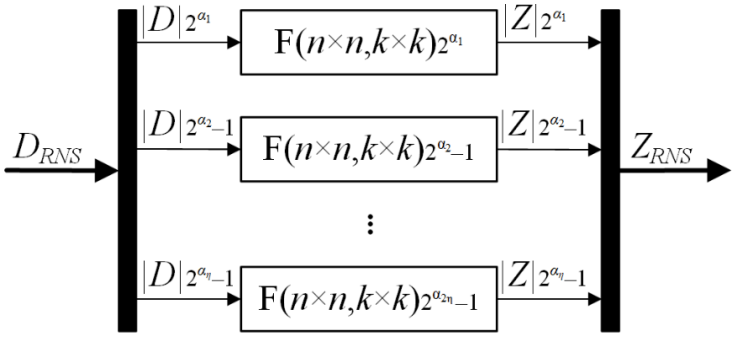


Рисунок 2.3.2. Архитектура устройства $\mathrm{F}(n\times n, k\times k)_{\mathrm{RNS}}$ для двумерной фильтрации по методу Винограда в СОК с модулями вида 2^{α} и $2^{\alpha}-1$

2.4 Проектирование фильтра на основе метода Винограда $\boldsymbol{F(2\times 2, 2\times 2)}$ с вычислениями в системе остаточных классов

Рассмотрим одномерную фильтрацию по методу Винограда [108] на примере случая $F(2,2)$. Представим векторы w, d и z в виде полиномов:

$$w(x) = w_1 x + w_0,$$
$$z(x) = z_1 x + z_0, \qquad (2.4.3)$$
$$d(x) = d_2 x^2 + d_1 x + d_0.$$

Тогда фильтрация может быть представлена в виде произведения полиномов

$$d(x) = w(x)z(x). \qquad (2.4.4)$$

Введем многочлен $m(x)$ степени 4, тогда $d(x)$ может быть представлен в виде остатка от деления по модулю $m(x)$

$$d(x) = w(x)z(x) \bmod m(x). \qquad (2.4.5)$$

Если заменить $m(x)$ степени 4 на полином степени 3, то

$$d(x) = w(x)z(x) \bmod m(x) + R_{m(x)}[d(x)]. \qquad (2.4.6)$$

где $R_{m(x)}[d(x)]$ – остаток от деления $d(x)$ на $m(x)$.

Выберем многочлен $m(x) = m^{(0)}(x)m^{(1)}(x)m^{(2)}(x) = x(x-1)(x-\infty)$, где $(x-\infty)$ соответствует $R_{m(x)}[d(x)]$. Тогда остатки от деления $w(x)$ на $m^{(i)}(x)$ равны

$$w^{(0)}(x) = w(x) \bmod m^{(0)}(x) = w_0,$$
$$w^{(1)}(x) = w(x) \bmod m^{(1)}(x) = w_0 + w_1, \quad (2.4.7)$$
$$w^{(2)}(x) = w(x) \bmod m^{(2)}(x) = w_1.$$

А остатки от деления $z(x)$ на $m^{(i)}(x)$ равны

$$z^{(0)}(x) = z(x) \bmod m^{(0)}(x) = z_0,$$
$$z^{(1)}(x) = z(x) \bmod m^{(1)}(x) = z_0 + z_1, \quad (2.4.8)$$
$$z^{(2)}(x) = z(x) \bmod m^{(2)}(x) = z_1.$$

Матрица преобразования A составляется из коэффициентов при остатках от деления $z(x)$ на $m^{(i)}(x)$ и имеет следующий вид

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}. \quad (2.4.9)$$

Пусть $M^{(i)}(x) = \frac{m(x)}{m^{(i)}(x)}$, тогда

$$M^{(0)}(x) = x - 1,$$
$$M^{(1)}(x) = x, \quad (2.4.10)$$
$$m(x) = x^2 - x.$$

Матрица преобразования B составляется из коэффициентов полиномов $M^{(i)}(x)$ и $m(x)$, так что коэффициенты $M^{(i)}(x)$ соответствуют i-му столбцу матрицы B.

$$B = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.4.11)$$

С помощью расширенного алгоритма Евклида нужно вычислить такие $h^{(i)}(x)$ и $H^{(i)}(x)$, что $h^{(i)}(x)m^{(i)}(x) + H^{(i)}(x)M^{(i)}(x) = 1$:

$$h^{(0)}(x) = 1, H^{(0)}(x) = -1;$$
$$h^{(1)}(x) = -1, H^{(1)}(x) = 1. \quad (2.4.12)$$

Матрица преобразования G составляется из коэффициентов остатков от деления $w^{(i)}(x)$, умноженных на $H^{(i)}(x)$:

$$G = \begin{bmatrix} -1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}. \quad (2.4.13)$$

Для двумерной фильтрации $F(2 \times 2, 2 \times 2)$ вычисления производятся по формуле (2.3.2). Далее представлены архитектуры устройств двумерной фильтрации по методу Винограда $F(2 \times 2, 2 \times 2)$ с вычислениями в СОК.

В случае $F(2 \times 2, 2 \times 2)$ двумерный сигнал разделяется на фрагменты размера 3×3 и обработка производится с шагом 2. На рисунке 2.4.1 представлен процесс фильтрации изображения размерности 256×256 по методу Винограда $F(2 \times 2, 2 \times 2)$. Этапы фильтрации фрагмента изображения размерности 3×3 по методу Винограда $F(2 \times 2, 2 \times 2)$ представлены на рисунке 2.4.2.

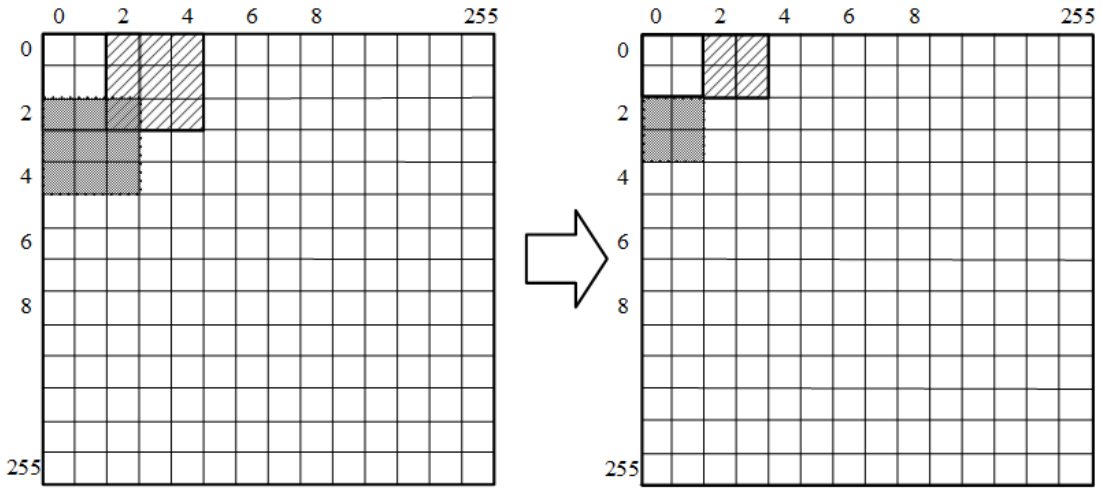


Рисунок 2.4.1. Процесс фильтрации изображения по методу Винограда $F(2 \times 2, 2 \times 2)$

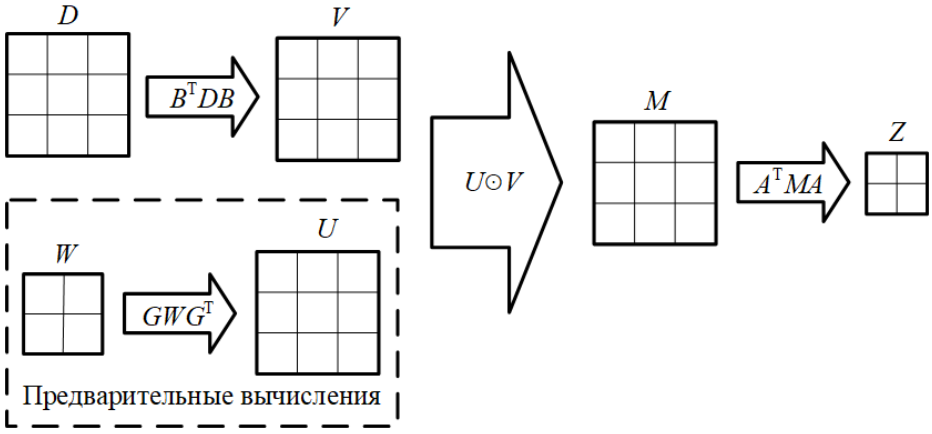


Рисунок 2.4.2. Этапы фильтрации фрагмента изображения по методу Винограда $F(2 \times 2, 2 \times 2)$

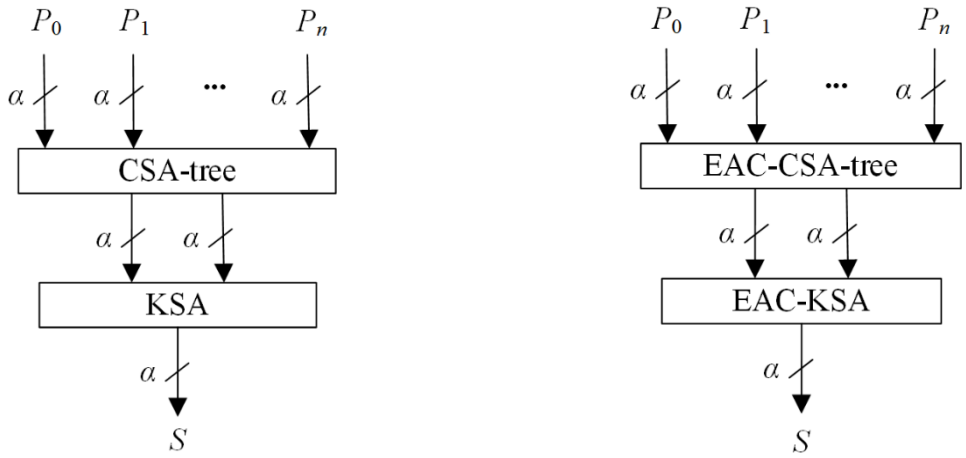
При фильтрации методом Винограда $F(2 \times 2, 2 \times 2)$ матрица V формируется следующим образом

$$V = \begin{bmatrix} d_{0,0} - d_{1,0} - d_{0,1} + d_{1,1} & -d_{0,1} + d_{1,1} & d_{0,1} - d_{1,1} - d_{0,2} + d_{1,2} \\ -d_{1,0} + d_{1,1} & d_{1,1} & -d_{1,1} + d_{1,2} \\ d_{1,0} - d_{2,0} - d_{1,1} + d_{2,1} & -d_{1,1} + d_{2,1} & d_{1,1} - d_{2,1} - d_{1,2} + d_{2,2} \end{bmatrix}, \quad (2.4.14)$$

а матрица итогового преобразования Z:

$$Z = \begin{bmatrix} M_{0,0} + M_{1,0} + M_{0,1} + M_{1,1} & M_{0,1} + M_{1,1} + M_{0,2} + M_{1,2} \\ M_{1,0} + M_{2,0} + M_{1,1} + M_{2,1} & M_{1,1} + M_{2,1} + M_{1,2} + M_{2,2} \end{bmatrix}. \quad (2.4.15)$$

Сложение нескольких чисел по модулю 2^α и $2^\alpha - 1$ предлагается производить с помощью сумматора по модулю с множественным входом, введем для них обозначения MOMA_{2^α} и $\mathrm{MOMA}_{2^\alpha-1}$ (Multi-Operand Modulo Adder) соответственно (рис. 2.4.3). Данные устройства состоят из дерева сумматоров с сохранением переноса CSA (Carry-Save Adder) и сумматора Когге-Стоуна KSA (Kogge-Stone Adder). На вход устройств поступает вектор $P = \{P_0, P_1, \dots, P_n\}$, а на выходе формируется сумма S. Для вычислений по модулю $2^\alpha - 1$ используется техника циклического переноса старших бит EAC (End-Around-Carry) [103].



а) б)

Рисунок 2.4.3. Архитектура сумматора по модулю с множественным входом MOMA: а) по модулю 2^α (MOMA_{2^α}); б) по модулю $2^\alpha - 1$ ($\mathrm{MOMA}_{2^\alpha-1}$)

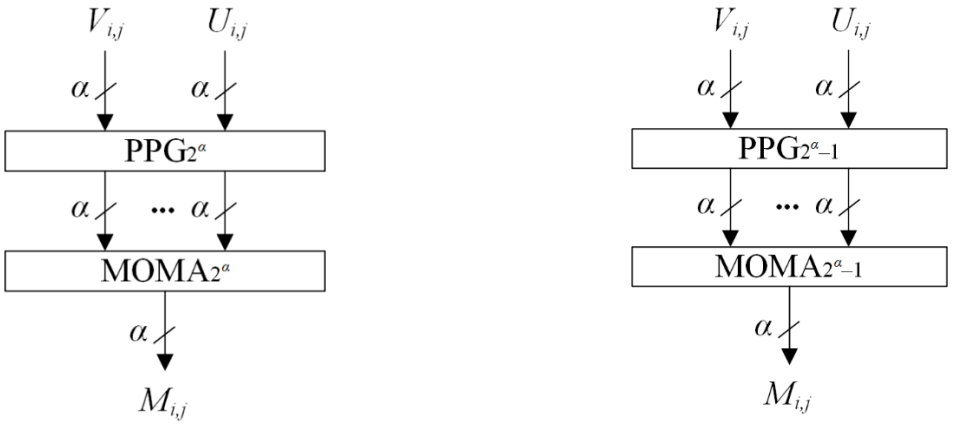
Элементы матрицы V по модулю 2^α вычисляются с помощью устройства MOMA_{2^α} (рис. 2.4.3а). Представление отрицательных чисел по модулю 2^α требует их преобразования в дополнительный код, то есть инверсии числа и добавления единицы. Следовательно, для вычисления элемента $V_{i,j}$ по модулю 2^α, на вход MOMA_{2^α} подается вектор данных $D^{i,j}$ и корректирующий коэффициент $C^{i,j}$, равный количеству отрицательных чисел, где $0 \le i \le 2$ и $0 \le j \le 2$. Исключением является элемент $V_{1,1} = D^{1,1} = d_{1,1}$, который не требует выполнения вычислений. Таким образом, устройство преобразования данных по модулю 2^α (обозначим его DT_{2^α}) состоит из восьми устройств MOMA_{2^α}, на входы которых поступают следующие данные:

$$
\begin{aligned}
D^{0,0} &= \{d_{0,0}, \overline{d_{1,0}}, \overline{d_{0,1}}, d_{1,1}\},\ C^{0,0} = 2;\\
D^{0,1} &= \{\overline{d_{0,1}}, d_{1,1}\},\ C^{0,1} = 1;\\
D^{0,2} &= \{d_{0,1}, \overline{d_{1,1}}, \overline{d_{0,2}}, d_{1,2}\},\ C^{0,2} = 2;\\
D^{1,0} &= \{\overline{d_{1,0}}, d_{1,1}\},\ C^{1,0} = 1;\\
D^{1,2} &= \{\overline{d_{1,1}}, d_{1,2}\},\ C^{1,2} = 1;\\
D^{2,0} &= \{d_{1,0}, \overline{d_{2,0}}, \overline{d_{1,1}}, d_{2,1}\},\ C^{2,0} = 2;\\
D^{2,1} &= \{\overline{d_{1,1}}, d_{2,1}\},\ C^{2,1} = 1;
\end{aligned}
\qquad (2.4.16)
$$

$$D^{2,2} = \{d_{1,1}, \overline{d_{2,1}}, \overline{d_{1,2}}, d_{2,2}\}, C^{2,2} = 2.$$

Вычисление элементов матрицы V по модулю $2^\alpha - 1$ требует представления отрицательных чисел в обратном коде, то есть выполнения инверсии числа, следовательно, корректирующие константы не участвуют в вычислениях. Устройство преобразования данных по модулю $2^\alpha - 1$ (обозначим его $\mathrm{DT}_{2^\alpha-1}$) состоит из четырех устройств $\mathrm{MOMA}_{2^\alpha-1}$ (рис. 2.4.3б), на входы которых поступают векторы $D^{0,0}$, $D^{0,2}$, $D^{2,0}$ и $D^{2,2}$, и из четырех EAC-KSA, на входы которых подаются векторы из двух элементов $D^{0,1}$, $D^{1,0}$, $D^{1,2}$ и $D^{2,1}$.

Устройства поэлементного умножения матриц U и V по модулям 2^α и $2^\alpha - 1$ состоят из девяти умножителей MUL_{2^α} и $\mathrm{MUL}_{2^\alpha-1}$ соответственно, представленных на рисунке 2.4.4. Умножитель MUL_{2^α} состоит из генератора частичных произведений по модулю 2^α PPG_{2^α}, который сформирован из массива вентилей AND [88] и MOMA_{2^α}. Устройство $\mathrm{MUL}_{2^\alpha-1}$ состоит из генератора частичных произведений по модулю $2^\alpha - 1$ $\mathrm{PPG}_{2^\alpha-1}$, использующего технику EAC, и $\mathrm{MOMA}_{2^\alpha-1}$. Обозначим устройство поэлементного умножения по модулю 2^α как EWM_{2^α}, а по модулю $2^\alpha - 1$ как $\mathrm{EWM}_{2^\alpha-1}$. На выходах данных устройств формируется матрица M.



а) б)

Рисунок 2.4.4. Архитектура умножителя MUL: а) по модулю 2^α (MUL_{2^α}); б) по модулю $2^\alpha - 1$ ($\mathrm{MUL}_{2^\alpha-1}$)

Матрица Z формируется с помощью устройств итогового преобразования по модулям 2^α и $2^\alpha - 1$, введем для них обозначения FT_{2^α} и $\mathrm{FT}_{2^\alpha-1}$. Устройство FT_{2^α} состоит из четырех MOMA_{2^α}, на входы которых поступают векторы $R^{i,j}$:

$$\begin{aligned} R^{0,0} &= \{M_{0,0}, M_{1,0}, M_{0,1}, M_{1,1}\}, \\ R^{0,1} &= \{M_{0,1}, M_{1,1}, M_{0,2}, M_{1,2}\}, \\ R^{1,0} &= \{M_{1,0}, M_{2,0}, M_{1,1}, M_{2,1}\}, \\ R^{1,1} &= \{M_{1,1}, M_{2,1}, M_{1,2}, M_{2,2}\}. \end{aligned} \tag{2.4.17}$$

а на выходе формируются $z_{i,j}$, где $0 \le i \le 2$ и $0 \le j \le 2$. Устройство $\mathrm{FT}_{2^\alpha-1}$ состоит из четырех $\mathrm{MOMA}_{2^\alpha-1}$, на входы которых также поступают векторы $R^{i,j}$.

Таким образом, устройство $\mathrm{F}(2\times 2, 2\times 2)_{2^\alpha}$ для фильтрации по методу Винограда по модулю 2^α состоит из устройства преобразования данных DT_{2^α}, устройства для поэлементного умножения матриц MUL_{2^α} и устройства итогового преобразования FT_{2^α} (рис. 2.4.5). На вход данного устройства подается фрагмент данных $|D|_{2^\alpha}$, а на выходе формируется результат фильтрации данного фрагмента $|Z|_{2^\alpha}$. Устройство $\mathrm{F}(2\times 2, 2\times 2)_{2^\alpha-1}$ для фильтрации по методу Винограда по модулю $2^\alpha - 1$ имеет схожую структуру и состоит из блоков $\mathrm{DT}_{2^\alpha-1}$, $\mathrm{MUL}_{2^\alpha-1}$ и $\mathrm{FT}_{2^\alpha-1}$ (рис. 2.4.6). На вход устройства поступает фрагмент данных $|D|_{2^\alpha-1}$, а на выходе формируется обработанный фрагмент $|Z|_{2^\alpha-1}$.

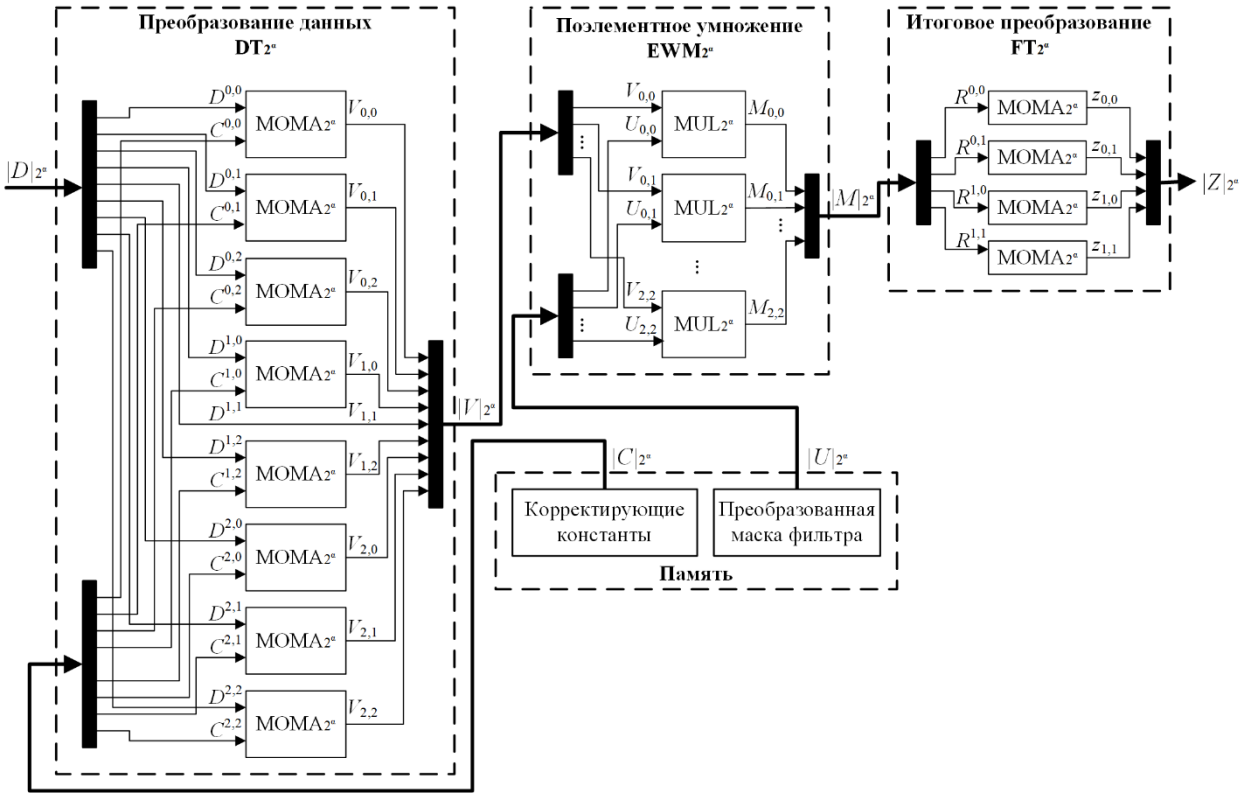


Рисунок 2.4.5. Архитектура устройства $\mathrm{F}(2\times 2, 2\times 2)_{2^\alpha}$ для двумерной фильтрации по методу Винограда по модулю 2^α

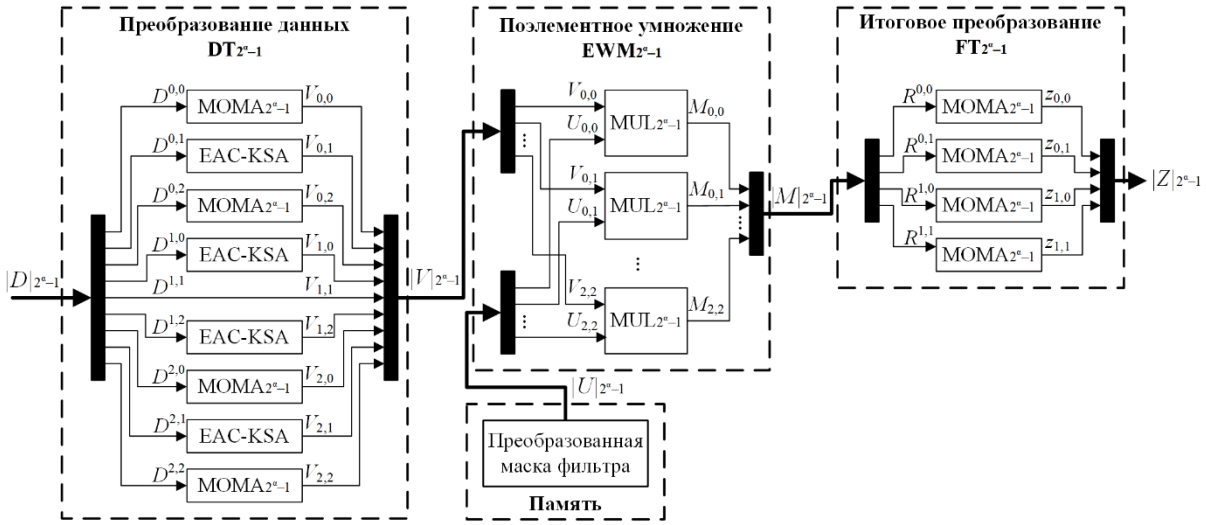


Рисунок 2.4.6. Архитектура устройства $\mathrm{F}(2\times 2, 2\times 2)_{2^\alpha-1}$ для двумерной фильтрации по методу Винограда по модулю $2^\alpha - 1$

На рисунке 2.4.7. представлена архитектура предлагаемого устройства $\mathrm{F}(2\times 2, 2\times 2)_{\mathrm{RNS}}$ для фильтрации по методу Винограда в СОК с набором модулей $\{2^{\alpha_1}, 2^{\alpha_2}-1, \dots, 2^{\alpha_\eta}-1\}$. На вход устройства подается массив данных $D_{RNS} = \{|D|_{2^{\alpha_1}}, |D|_{2^{\alpha_2}-1}, \dots, |D|_{2^{\alpha_\eta}-1}\}$, представленный в СОК. Затем,

входные данные обрабатываются параллельно устройствами $\mathrm{F}(2\times 2,2\times 2)_{2^{\alpha_1}}$, $\mathrm{F}(2\times 2,2\times 2)_{2^{\alpha_2-1}}$, …, $\mathrm{F}(2\times 2,2\times 2)_{2^{\alpha_\eta-1}}$, которые формируют на выходе устройства $\mathrm{F}(2\times 2,2\times 2)_{\mathrm{RNS}}$ массив обработанных данных $Z_{RNS}=\{|Z|_{2^{\alpha_1}},|Z|_{2^{\alpha_2-1}},\dots,|Z|_{2^{\alpha_\eta-1}}\}$.

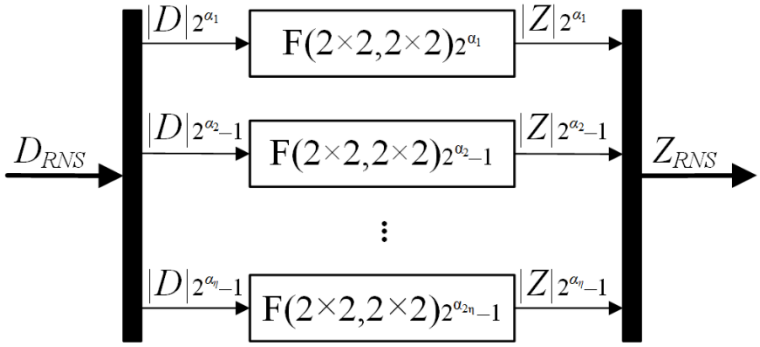


Рисунок 2.4.7. Архитектура устройства $\mathrm{F}(2\times 2,2\times 2)_{\mathrm{RNS}}$ для двумерной фильтрации по методу Винограда в СОК с модулями вида 2^{α} и $2^{\alpha}-1$

Далее представлены результаты теоретического анализа технических параметров предлагаемых устройств фильтрации по методу Винограда с использованием арифметики СОК.

Для оценки параметров предлагаемых цифровых устройств будем использовать абстрактную модель подсчета задержки и площади, известную как «unit-gate»-модель [113]. Обозначим задержку логического устройства U_{delay}, а площадь логического устройства U_{area}, тогда логические вентили имеют следующее описание

$$\begin{gathered}U_{\mathrm{delay}}(\mathrm{NOT})=0,U_{\mathrm{area}}(\mathrm{NOT})=0;\\ U_{\mathrm{delay}}(\mathrm{AND})=1,U_{\mathrm{area}}(\mathrm{AND})=1;\\ U_{\mathrm{delay}}(\mathrm{OR})=1,U_{\mathrm{area}}(\mathrm{OR})=1;\\ U_{\mathrm{delay}}(\mathrm{XOR})=2,U_{\mathrm{area}}(\mathrm{XOR})=2;\\ U_{\mathrm{delay}}(\mathrm{XNOR})=2,U_{\mathrm{area}}(\mathrm{XNOR})=2.\end{gathered}\qquad(2.4.18)$$

В соответствии с (2.4.18) параметры сумматора $\mathrm{MOMA}_{2^{\alpha}}$ имеют вид

$$U_{\mathrm{delay}}(\mathrm{MOMA}_{2^{\alpha}})=6{,}8\log_2 N+2\log_2\alpha+4,\qquad(2.4.19)$$

$$U_{\text{area}}(\text{MOMA}_{2^\alpha}) = 3\alpha\log_2\alpha + 7\alpha N - 11\alpha + 1.$$

где N – количество слагаемых [7]. Аналогично, параметры сумматора $\text{MOMA}_{2^\alpha-1}$ могут быть представлены в виде [7]

$$\begin{gathered} U_{\text{delay}}(\text{MOMA}_{2^\alpha-1}) = 6{,}8\log_2 N + 2\log_2\alpha + 4, \\ U_{\text{area}}(\text{MOMA}_{2^\alpha-1}) = 3\alpha\log_2\alpha + 7\alpha N - 8\alpha. \end{gathered} \tag{2.4.20}$$

Устройство преобразования данных DT_{2^α} содержит четыре сумматора MOMA_{2^α}, на вход которых поступает пять слагаемых, и четыре MOMA_{2^α}, на вход которых поступает три слагаемых. Тогда, принимая во внимание (2.4.19), параметры устройства DT_{2^α} рассчитываются следующим образом

$$\begin{gathered} U_{\text{delay}}(\text{DT}_{2^\alpha}) = 6{,}8\log_2 5 + 2\log_2\alpha + 4 \approx 2\log_2\alpha + 24{,}4, \\ U_{\text{area}}(\text{DT}_{2^\alpha}) = 24\alpha\log_2\alpha + 136\alpha + 8. \end{gathered} \tag{2.4.21}$$

Устройство преобразования данных $\text{DT}_{2^\alpha-1}$ состоит из четырех сумматоров $\text{MOMA}_{2^\alpha-1}$, на вход которых подается четыре слагаемых, и четырех сумматоров EAC-KSA. Параметры задержки и площади сумматора EAC-KSA вычисляются следующим образом [7]

$$\begin{gathered} U_{\text{delay}}(\text{EAC-KSA}) = 2\log_2\alpha + 4, \\ U_{\text{area}}(\text{EAC-KSA}) = 3\alpha\log_2\alpha + 6\alpha. \end{gathered} \tag{2.4.22}$$

Тогда, на основании (2.4.18) и (2.4.20), параметры устройства $\text{DT}_{2^\alpha-1}$ равны:

$$\begin{gathered} U_{\text{delay}}(\text{DT}_{2^\alpha-1}) = 6{,}8\log_2 4 + 2\log_2\alpha + 4 \approx 2\log_2\alpha + 17{,}6, \\ U_{\text{area}}(\text{DT}_{2^\alpha-1}) = 24\alpha\log_2\alpha + 104\alpha. \end{gathered} \tag{2.4.23}$$

Умножитель MUL состоит из генератора частичных произведений PPG и блока MOMA на α входов. Генератор частичных произведений PPG_{2^α} имеет следующие параметры задержки и площади

$$\begin{gathered} U_{\text{delay}}(\text{PPG}_{2^\alpha}) = 0{,}5\alpha^2 + 0{,}5\alpha, \\ U_{\text{area}}(\text{PPG}_{2^\alpha}) = 0{,}5\alpha^2 + 0{,}5\alpha. \end{gathered} \tag{2.4.24}$$

Генератор частичных произведений $\text{PPG}_{2^\alpha-1}$ имеет следующие параметры

$$\begin{gathered} U_{\text{delay}}(\text{PPG}_{2^\alpha-1}) = \alpha^2, \\ U_{\text{area}}(\text{PPG}_{2^\alpha-1}) = \alpha^2. \end{gathered} \tag{2.4.25}$$

Тогда, учитывая (2.4.19) и (2.4.24), умножитель MUL_{2^α} имеет параметры

$$U_{\mathrm{delay}}(\mathrm{MUL}_{2^\alpha}) = 8{,}8\log_2\alpha + 0{,}5\alpha^2 + 0{,}5\alpha + 4,$$
$$U_{\mathrm{area}}(\mathrm{MUL}_{2^\alpha}) = 3\alpha\log_2\alpha + 7{,}5\alpha^2 - 10{,}5\alpha + 1. \quad (2.4.26)$$

А умножитель $\mathrm{MUL}_{2^\alpha-1}$, на основании (2.4.20) и (2.4.25) имеет параметры

$$U_{\mathrm{delay}}(\mathrm{MUL}_{2^\alpha-1}) = 8{,}8\log_2\alpha + \alpha^2 + 4,$$
$$U_{\mathrm{area}}(\mathrm{MUL}_{2^\alpha-1}) = 3\alpha\log_2\alpha + 8\alpha^2 - 8\alpha. \quad (2.4.27)$$

Устройство поэлементного умножения матриц EWM_{2^α} состоит из девяти умножителей MUL_{2^α}, имеющих параметры (2.4.26), следовательно,

$$U_{\mathrm{delay}}(\mathrm{EWM}_{2^\alpha}) = 8{,}8\log_2\alpha + 0{,}5\alpha^2 + 0{,}5\alpha + 4,$$
$$U_{\mathrm{area}}(\mathrm{EWM}_{2^\alpha}) = 27\alpha\log_2\alpha + 67{,}5\alpha^2 - 94{,}5\alpha + 9. \quad (2.4.28)$$

Устройство $\mathrm{EWM}_{2^\alpha-1}$ состоит из девяти умножителей $\mathrm{MUL}_{2^\alpha-1}$, имеющих параметры (2.4.27), следовательно, его параметры можно представить как

$$U_{\mathrm{delay}}(\mathrm{EWM}_{2^\alpha-1}) = 8{,}8\log_2\alpha + \alpha^2 + 4,$$
$$U_{\mathrm{area}}(\mathrm{EWM}_{2^\alpha-1}) = 27\alpha\log_2\alpha + 72\alpha^2 - 72\alpha. \quad (2.4.29)$$

Устройство итогового преобразования FT_{2^α} состоит из четырех устройств MOMA_{2^α}, на вход которых поступают четыре слагаемых. Тогда, учитывая параметры MOMA_{2^α} (2.4.19), устройство FT_{2^α} имеет следующие параметры

$$U_{\mathrm{delay}}(\mathrm{FT}_{2^\alpha}) = 6{,}8\log_2 4 + 2\log_2\alpha + 4 = 2\log_2\alpha + 17{,}6,$$
$$U_{\mathrm{area}}(\mathrm{FT}_{2^\alpha}) = 12\alpha\log_2\alpha + 68\alpha + 4. \quad (2.4.30)$$

Аналогично, устройство $\mathrm{FT}_{2^\alpha-1}$ состоит из четырех устройств $\mathrm{MOMA}_{2^\alpha-1}$, на вход которых поступают четыре слагаемых. Учитывая параметры $\mathrm{MOMA}_{2^\alpha-1}$ (2.4.20), устройство $\mathrm{FT}_{2^\alpha-1}$ имеет следующие параметры

$$U_{\mathrm{delay}}(\mathrm{FT}_{2^\alpha-1}) = 6{,}8\log_2 4 + 2\log_2\alpha + 4 = 2\log_2\alpha + 17{,}6,$$
$$U_{\mathrm{area}}(\mathrm{FT}_{2^\alpha-1}) = 12\alpha\log_2\alpha + 80\alpha. \quad (2.4.31)$$

Принимая во внимание параметры устройств DT_{2^α}, EWM_{2^α} и FT_{2^α} – (2.4.21), (2.4.28), (2.4.30), устройство фильтрации по методу Винограда $\mathrm{F}(2\times 2,2\times 2)_{2^\alpha}$ с вычислениями по модулю 2^α имеет следующие параметры задержки и площади

$$\begin{gathered} U_{\mathrm{delay}}(\mathrm{F}(2\times 2,2\times 2)_{2^\alpha}) = U_{\mathrm{delay}}(\mathrm{DT}_{2^\alpha}) + U_{\mathrm{delay}}(\mathrm{EWM}_{2^\alpha}) + \\ + U_{\mathrm{delay}}(\mathrm{FT}_{2^\alpha}) = 12{,}8\log_2\alpha + 0{,}5\alpha^2 + 0{,}5\alpha + 46, \\ U_{\mathrm{area}}(\mathrm{F}(2\times 2,2\times 2)_{2^\alpha}) = U_{\mathrm{area}}(\mathrm{DT}_{2^\alpha}) + U_{\mathrm{area}}(\mathrm{EWM}_{2^\alpha}) + \\ + U_{\mathrm{area}}(\mathrm{FT}_{2^\alpha}) = 63\alpha\log_2\alpha + 67{,}5\alpha^2 + 109{,}5\alpha + 21. \end{gathered} \tag{2.4.32}$$

Принимая во внимание параметры блоков $\mathrm{DT}_{2^\alpha-1}$, $\mathrm{EWM}_{2^\alpha-1}$ и $\mathrm{FT}_{2^\alpha-1}$ – (2.4.23), (2.4.29), (2.4.31), устройство фильтрации по методу Винограда $\mathrm{F}(2\times 2,2\times 2)_{2^\alpha-1}$ с вычислениями по модулю $2^\alpha-1$ имеет следующие параметры задержки и площади

$$\begin{gathered} U_{\mathrm{delay}}(\mathrm{F}(2\times 2,2\times 2)_{2^\alpha-1}) = U_{\mathrm{delay}}(\mathrm{DT}_{2^\alpha-1}) + \\ + U_{\mathrm{delay}}(\mathrm{EWM}_{2^\alpha-1}) + U_{\mathrm{delay}}(\mathrm{FT}_{2^\alpha-1}) = 12{,}8\log_2\alpha + \alpha^2 + 39{,}2, \\ U_{\mathrm{area}}(\mathrm{F}(2\times 2,2\times 2)_{2^\alpha-1}) = U_{\mathrm{delay}}(\mathrm{DT}_{2^\alpha-1}) + \\ + U_{\mathrm{delay}}(\mathrm{EWM}_{2^\alpha-1}) + U_{\mathrm{delay}}(\mathrm{FT}_{2^\alpha-1}) = 63\alpha\log_2\alpha + 72\alpha^2 + 112\alpha. \end{gathered} \tag{2.4.33}$$

Параметры предлагаемого устройства фильтрации $\mathrm{F}(2\times 2,2\times 2)_{\mathrm{RNS}}$ с вычислениями в СОК с набором модулей $\{2^{\alpha_1}, 2^{\alpha_2}-1, \dots, 2^{\alpha_\eta}-1\}$, изображенного на рисунке 2.4.7, принимая во внимание (2.4.32) и (2.4.33), вычисляются как

$$\begin{gathered} U_{\mathrm{delay}}(\mathrm{F}(2\times 2,2\times 2)_{RNS}) = \max\big(U_{\mathrm{delay}}(\mathrm{F}(2\times 2,2\times 2)_{2^{\alpha_1}}), \\ \big\{U_{\mathrm{delay}}(\mathrm{F}(2\times 2,2\times 2)_{2^{\alpha_i}-1})\big\} | 2\le i\le\eta\big), \\ U_{\mathrm{area}}(\mathrm{F}(2\times 2,2\times 2)_{RNS}) = U_{\mathrm{area}}(\mathrm{F}(2\times 2,2\times 2)_{2^{\alpha_1}}) + \\ + \sum_{i=2}^{\eta} U_{\mathrm{area}}(\mathrm{F}(2\times 2,2\times 2)_{2^{\alpha_i}-1}). \end{gathered} \tag{2.4.34}$$

На основании (2.4.34) были рассчитаны параметры задержки и площади предлагаемого устройства фильтрации по методу Винограда $\mathrm{F}(2\times 2,2\times 2)_{\mathrm{RNS}}$ с различными наборами модулей СОК, представленными в

таблице 2.4.1, которые соответствуют различным разрядностям входных данных.

Таблица 2.4.1. Модули СОК

Разрядность фильтра, бит	Набор модулей СОК	Динамический диапазон СОК
8	$\{2^5, 2^3 - 1, 2^2 - 1\}$	672
16	$\{2^8, 2^5 - 1, 2^4 - 1\}$	119040
32	$\{2^{12}, 2^{11} - 1, 2^{10} - 1\}$	8577355776

Проведено сравнение предлагаемой архитектуры фильтра с вычислениями в СОК и известной архитектуры фильтра с вычислениями в ПСС [111]. Также были рассчитаны параметры фильтров с конечной импульсной характеристикой, основанные на MAC блоках [101], введем для них обозначение FIR(MAC), тогда для фильтра порядка P параметры площади и задержки устройства вычисляются следующим образом

$$\begin{gathered} U_{\text{delay}}\big(\text{FIR(MAC)}\big) = 8{,}8P\log_2\alpha + 8{,}8\log_2\alpha + 5P + 5, \\ U_{\text{area}}\big(\text{FIR(MAC)}\big) = 3\alpha P\log_2\alpha + 3\alpha\log_2\alpha + 8\alpha^2 P + 8\alpha^2 - \\ -4\alpha P - 4\alpha + P + 1. \end{gathered} \quad (2.4.35)$$

В работе [6] представлены усеченные MAC блоки – TMAC (truncated MAC). Параметры фильтра порядка P, состоящего из этих блоков вычисляются следующим образом

$$\begin{gathered} U_{\text{delay}}\big(\text{FIR(TMAC)}\big) = 6{,}8P\log_2\alpha + 8{,}8\log_2\alpha + P + 5, \\ U_{\text{area}}\big(\text{FIR(TMAC)}\big) = 3\alpha\log_2\alpha + 8\alpha^2 P + 8\alpha^2 + 3\alpha + 1. \end{gathered} \quad (2.4.36)$$

Кроме того, были рассчитаны параметры фильтров, состоящих из блоков TMAC с вычислениями в СОК с модулями специального вида [7]. Параметры устройства, выполняющего вычисления по модулю 2^α, вычисляются по формулам (2.4.36), а для устройств по модулю $2^\alpha - 1$ следующим образом

$$\begin{gathered} U_{\text{delay}}(\text{FIR(TMAC)}_{2^\alpha-1}) = 6{,}8P\log_2\alpha + 8{,}8\log_2\alpha + P + 5, \\ U_{\text{area}}(\text{FIR(TMAC)}_{2^\alpha-1}) = 3\alpha\log_2\alpha + 8\alpha^2 P + 8\alpha^2 + 6\alpha. \end{gathered} \quad (2.4.37)$$

Результаты расчета параметров задержки и площади предлагаемого фильтра и известных аналогов представлены в таблице 2.4.2. Так же представлено время обработки изображения размерности 256×256, рассчитанное на основе задержки устройства.

Таблица 2.4.2. Теоретические параметры двумерных фильтров с маской 2×2

Параметр	Разрядность фильтра, бит	Известные методы				Предлагаемый метод
		[101]	[6]	[7]	[111]	
Задержка	8	157	117	93	121	**91**
	16	201	153	**117**	234	121
	32	245	189	**139**	638	205
Площадь	8	2765	2657	**1622**	6729	4910
	16	10885	10481	**4410**	23085	11925
	32	42725	41537	**15108**	82725	36521
Время обработки	8	10289152	7667712	6094848	1982464	**1490944**
	16	13172736	10027008	7667712	3833856	**1982464**
	32	16056320	12386304	9109504	10452992	**3358720**

Теоретический анализ на основе «unit-gate»-модели показал, что:

– по сравнению с известной реализацией на основе метода Винограда в ПСС [111] предлагаемое устройство, использующее СОК, позволяет сократить задержку на 24,79-66,87%, а площадь на 27,03-55,85%;

– по сравнению с известной архитектурой фильтра, основанной на MAC блоках [101] предлагаемая архитектура устройства имеет на 16,33-42,04% меньшую задержку, но на 9,55-77,58% большую площадь; исключением здесь является 32-битное устройство, которое имеет на 14,52% меньшую площадь;

– по сравнению с известной архитектурой устройства на основе TMAC блоков с вычислениями в ПСС [6] задержка предлагаемого устройства на 20,92-22,22% меньше, но площадь на 13,78-84,79% больше для 8- и 16-битных устройств; для 32-битных устройств задержка на 8,47% больше, но площадь на 12,08% меньше;

– по сравнению с известной архитектурой на основе TMAC блоков с вычислениями в СОК [7] предлагаемая архитектура 8-битного фильтра имеет на 2,15% меньшую задержку, но 16- и 32- битные устройства имеют на 3,42-

47,48% большую задержку; площадь предлагаемого устройства на 141,73-202,71% больше.

Главным преимуществом предлагаемой архитектуры фильтра на основе метода Винограда и арифметики СОК с модулями специального вида является сокращение времени обработки двумерного сигнала. Использование предлагаемого устройства позволяет сократить время обработки сигнала размерности 256×256 в 1,33-6,9 раза по сравнению с другими известными архитектурами.

Произведено аппаратное моделирование предлагаемой архитектуры устройства двумерной фильтрации $\mathrm{F}(2 \times 2, 2 \times 2)_{\mathrm{RNS}}$ по методу Винограда с вычислениями в СОК с модулями специального вида 2^{α} и $2^{\alpha} - 1$. Было выполнено сравнение технических характеристик предлагаемой архитектуры фильтра с известными аналогами. Результаты аппаратного моделирования представлены в таблице 2.4.3. Моделирование производилось в среде автоматизированного проектирования Xilinx Vivado 2018.3 на целевой плате Artix-7 xc7a200tffg1156-3 со стратегией оптимизации Flow_Perfoptimized_high. При оценке устройств учитывались следующие параметры: тактовая частота (измеряется в МГц), количество занятых просмотровых таблиц (Look-Up-Table, LUT), энергопотребление (измеряется в Вт) и производительность, равная количеству обработанных кадров 256×256 за секунду (кадр/с).

Результаты аппаратного моделирования показали, что производительность предлагаемой архитектуры фильтра в 1,31-4,12 раза выше по сравнению с известными архитектурами.

Максимальная тактовая частота предлагаемого устройства на 31,03-38,46% выше по сравнению с устройством на основе метода Винограда [111]. По сравнению с фильтром, основанным на MAC блоках [101] тактовая частота предлагаемого устройства на 1,89-2,94% выше для 16- и 32-битных устройств, но на 26,21% меньше для 8-битных устройств. По сравнению с архитектурами

на основе ТМАС блоков в ПСС [6] и СОК [7] тактовая частота предлагаемого устройства на 7,89-35,59% ниже.

Таблица 2.4.3. Результаты аппаратного моделирования двумерных фильтров с маской 2×2

Параметр	Разрядность фильтра, бит	Известные методы				Предлагаемый метод
		[101]	[6]	[7]	[111]	
Тактовая частота, МГц	8	103	103	**118**	58	76
	16	53	59	**76**	39	54
	32	34	38	**45**	26	35
Количество LUT	8	65	82	**60**	660	414
	16	402	388	**199**	2236	1541
	32	1582	1654	**1023**	8148	6675
Энергопотребление, Вт	8	**0,257**	**0,257**	0,263	0,277	0,285
	16	**0,256**	0,264	0,264	0,290	0,303
	32	**0,252**	0,259	0,258	0,350	0,359
Производительность, кадр/с	8	1571	1571	1800	3540	**4638**
	16	808	900	1159	2380	**3295**
	32	518	579	686	1586	**2136**

Количество занятых LUT в предлагаемом устройстве на 18,0837,27% меньше по сравнению с известной реализацией фильтра на основе метода Винограда [111], но в 3,83-7,74 раза больше по сравнению с другими рассмотренными архитектурами.

Энергопотребление предлагаемого устройства фильтрации на 2,5742,46% выше по сравнению с известными устройствами.

Преимущество во времени обработки изображения фильтром на основе метода Винограда, а также в производительности фильтра объясняется тем, что на выходе устройства формируется четыре пикселя обработанного изображения.

2.5 Проектирование фильтра на основе метода Винограда $\boldsymbol{F(2 \times 2, 3 \times 3)}$ с вычислениями в системе остаточных классов

Рассмотрим пример фильтрации по методу Винограда $F(2 \times 2, 3 \times 3)$. На рисунке 2.5.1 представлен процесс фильтрации изображения размерности

256×256 по методу Винограда $F(2 \times 2, 3 \times 3)$. Этапы фильтрации фрагмента изображения размерности 4×4 методом Винограда $F(2 \times 2, 3 \times 3)$ представлены на рисунке 2.5.2.

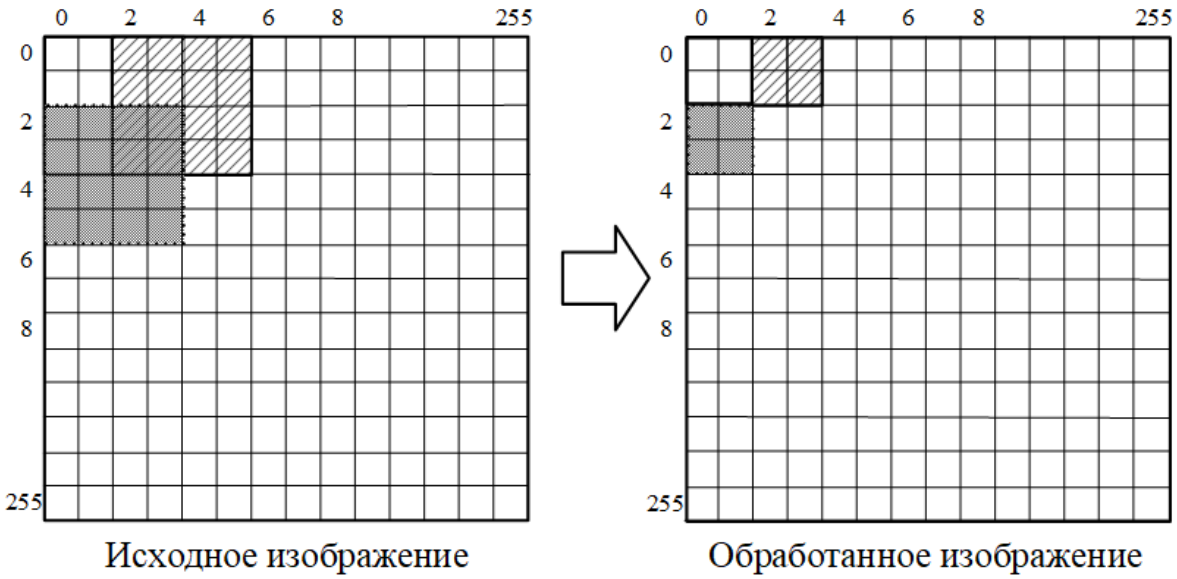


Рисунок 2.5.1. Процесс фильтрации изображения по методу Винограда $F(2 \times 2, 3 \times 3)$

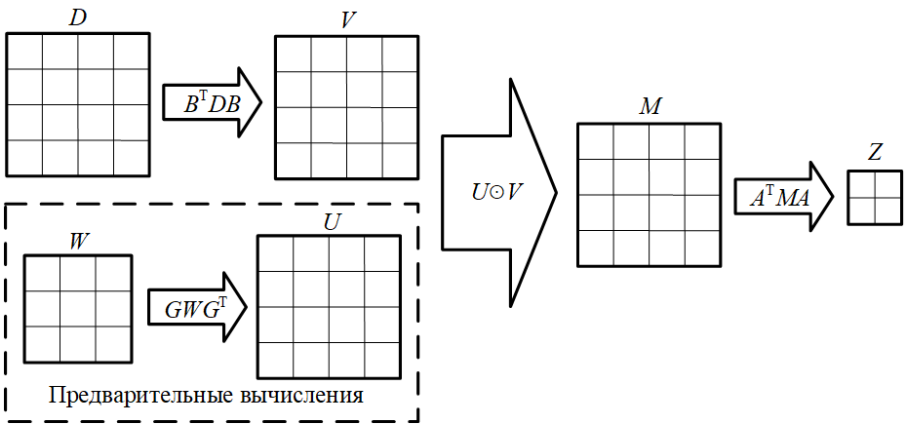


Рисунок 2.5.2. Этапы фильтрации фрагмента изображения по методу Винограда $F(2 \times 2, 3 \times 3)$

Матрицы преобразования для $F(2 \times 2, 3 \times 3)$ имеют вид

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 \\ -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & -1 \\ 0 & -1 \end{bmatrix}. \tag{2.5.1}$$

В случае двумерной фильтрации $F(2 \times 2, 3 \times 3)$ вычисления выполняются по формуле (2.3.2). Далее представлено описание предлагаемых архитектур устройств фильтрации изображений на основе метода Винограда $F(2 \times 2, 3 \times 3)$ с вычислениями в СОК с модулями специального вида 2^α и $2^\alpha - 1$. Предлагаемую архитектуру устройства цифровой фильтрации по модулю 2^α в СОК можно также использовать для выполнения вычислений в ПСС с числами разрядности α бит.

Для случая $F(2 \times 2, 3 \times 3)$, изображенного на рисунке 2.5.1, выбираются фрагменты изображения D размером 4×4 с шагом 2, результатом обработки фрагмента фильтром W с маской размером 3×3 по формуле (2.3.2) является фрагмент обработанного изображения Z размера 2×2. То есть при фильтрации методом Винограда $F(2 \times 2, 3 \times 3)$ формируются сразу четыре пикселя обработанного изображения.

В выражении (2.3.2) преобразование маски фильтра GWG^{T} может быть выполнено предварительно один раз и, следовательно, не несет вычислительной нагрузки. Для случая $F(2 \times 2, 3 \times 3)$ размер матриц U, V и M составляет 4×4. Вычисление матриц V и Z может быть представлено в виде

$$V_{i,0} = \begin{bmatrix} d_{0,0} - d_{2,0} - d_{0,2} + d_{2,2} \\ d_{1,0} + d_{2,0} - d_{1,2} - d_{2,2} \\ d_{2,0} - d_{1,0} - d_{2,2} + d_{1,2} \\ d_{1,0} - d_{3,0} - d_{1,2} + d_{3,2} \end{bmatrix}, \tag{2.5.2}$$

$$V_{i,1} = \begin{bmatrix} d_{0,1} - d_{2,1} + d_{0,2} - d_{2,2} \\ d_{1,1} + d_{2,1} + d_{1,2} + d_{2,2} \\ d_{2,1} - d_{1,1} - d_{1,2} + d_{2,2} \\ d_{1,1} - d_{3,1} + d_{1,2} - d_{3,2} \end{bmatrix},$$

$$V_{i,2} = \begin{bmatrix} -d_{0,1} + d_{2,1} + d_{0,2} - d_{2,2} \\ -d_{1,1} - d_{2,1} + d_{1,2} + d_{2,2} \\ -d_{2,1} + d_{1,1} - d_{1,2} + d_{2,2} \\ -d_{1,1} + d_{3,1} + d_{1,2} - d_{3,2} \end{bmatrix},$$

$$V_{i,3} = \begin{bmatrix} d_{0,1} - d_{2,1} - d_{0,3} + d_{2,3} \\ d_{1,1} + d_{2,1} - d_{1,3} - d_{2,3} \\ d_{2,1} - d_{1,1} + d_{1,3} - d_{2,3} \\ d_{1,1} - d_{3,1} - d_{1,3} + d_{3,3} \end{bmatrix},$$

$$Z_{0,j} =$$

$$= \begin{bmatrix} M_{0,0} + M_{1,0} + M_{2,0} + M_{0,1} + M_{1,1} + M_{2,1} + M_{0,2} + M_{1,2} + M_{2,2} \\ M_{0,1} + M_{1,1} + M_{2,1} - (M_{0,2} + M_{1,2} + M_{2,2}) - M_{0,3} - M_{1,3} - M_{2,3} \end{bmatrix}^{\mathrm{T}},$$

$$Z_{1,j} = \tag{2.5.3}$$

$$= \begin{bmatrix} M_{1,0} - M_{2,0} - M_{3,0} + M_{1,1} - M_{2,1} - M_{3,1} + M_{1,2} - M_{2,2} - M_{3,2} \\ M_{1,1} - M_{2,1} - M_{3,1} - (M_{1,2} - M_{2,2} - M_{3,2}) - M_{1,3} + M_{2,3} + M_{3,3} \end{bmatrix}^{\mathrm{T}},$$

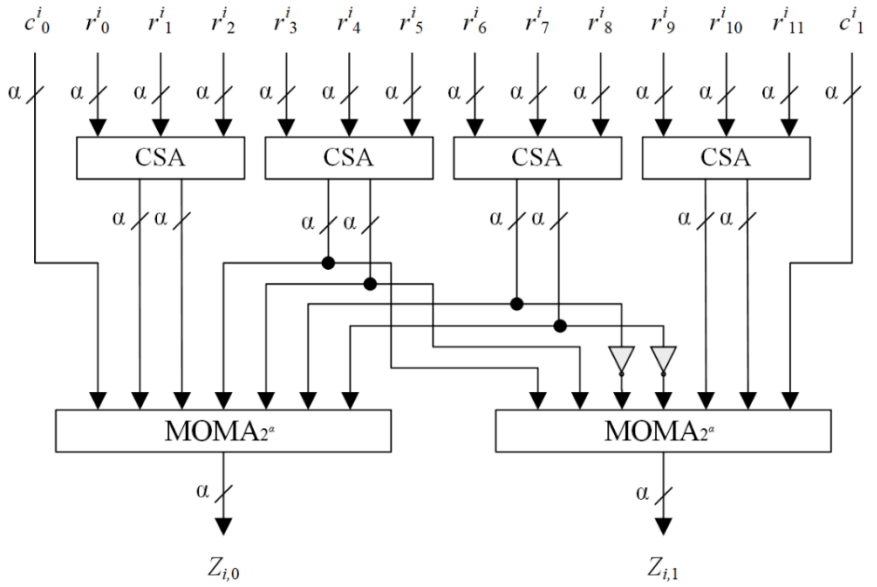
где $0 \le i \le 3$ и $0 \le j \le 1$.

Вычисление одного элемента матрицы V по модулю 2^{α} производится с помощью устройства $\mathrm{MOMA}_{2^{\alpha}}$, представленного на рисунке 2.4.3а. Данное устройство производит преобразование данных из матрицы D и генерирует элемент $V_{i,j}$, $0 \le i \le 3, 0 \le j \le 3$. Для вычисления элемента $V_{i,j}$ по модулю 2^{α}, на вход $\mathrm{MOMA}_{2^{\alpha}}$ подается вектор данных $D^{i,j}$ и корректирующий коэффициент $C_{\mathrm{DT}}^{i,j}$, равный количеству отрицательных чисел, где $0 \le i \le 3$ и $0 \le j \le 3$. Например, для вычисления элемента $V_{0,0}$ на вход устройства $\mathrm{MOMA}_{2^{\alpha}}$ поступает вектор данных $D^{0,0} = \{d_{0,0}, \overline{d_{2,0}}, \overline{d_{0,2}}, d_{2,2}\}$ и корректирующий коэффициент $C_{\mathrm{DT}}^{0,0} = 2$. Таким образом, устройство преобразования данных по модулю 2^{α} (обозначим его $\mathrm{DT}_{2^{\alpha}}$) состоит из 16 устройств $\mathrm{MOMA}_{2^{\alpha}}$.

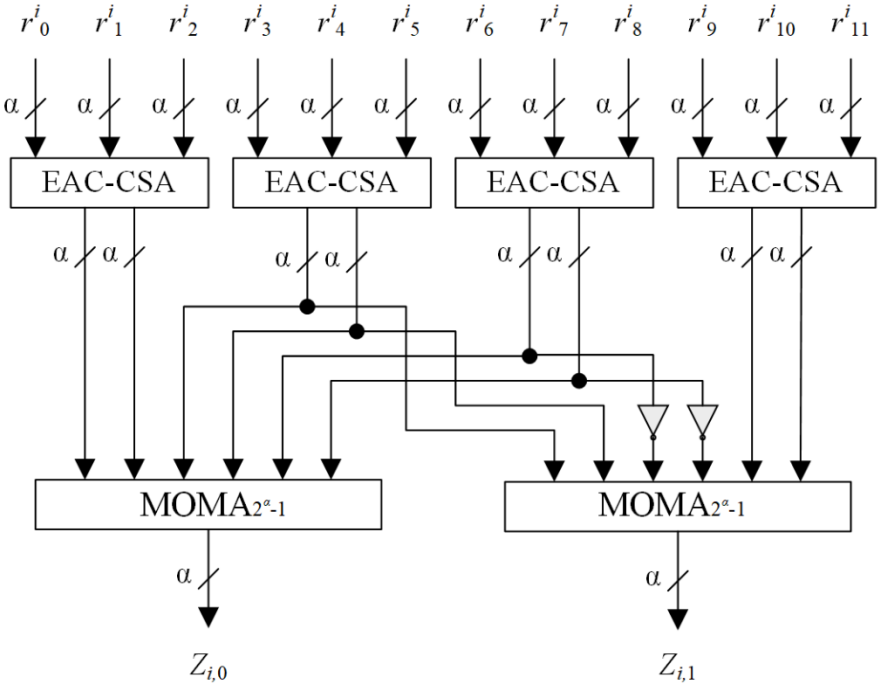
Вычисление элементов матрицы V по модулю $2^\alpha - 1$ требует представления отрицательных чисел в обратном коде, то есть выполнения инверсии числа, следовательно, корректирующие константы не участвуют в вычислениях. Следовательно, устройство преобразования данных по модулю $2^\alpha - 1$ (обозначим его $\mathrm{DT}_{2^\alpha-1}$) состоит из 16 устройств $\mathrm{MOMA}_{2^\alpha-1}$, представленных на рисунке 2.4.3б, на входы которых поступают векторы $D^{i,j}$.

Поэлементное умножение матриц U и V производится с помощью устройств EWM_{2^α} и $\mathrm{EWM}_{2^\alpha-1}$, состоящих из 16 параллельных умножителей двух чисел MUL_{2^α} и $\mathrm{MUL}_{2^\alpha-1}$ соответственно, и изображенных на рисунке 2.4.4. На вход устройства поступают элементы матриц U и V. Таким образом формируется матрица M размером 4×4.

Схема устройства для вычисления i-ой строки матрицы Z представлена на рисунке 2.5.3. Введем для данного устройства обозначение FTR (Final Transform for Row). Для вычисления матрицы Z используются два устройства FTR, изображенных на рисунке 2.5.3, работающих параллельно. На вход поступает вектор $R^i = \{r_0^i, r_1^i, \dots, r_{11}^i\}$, сформированный из элементов матрицы M. Для выполнения расчетов по модулю 2^α на вход устройства также подаются корректирующие константы $C_{\mathrm{FT}}^i = \{c_0^i, c_1^i\}$ (рис. 2.5.3а). Для расчета элементов первой строки $Z_{0,j}$, $j = 0, 1$, $R^0 = \{M_{0,0}, M_{1,0}, M_{2,0}, M_{0,1}, M_{1,1}, M_{2,1}, M_{0,2}, M_{1,2}, M_{2,2}, \overline{M_{0,3}}, \overline{M_{1,3}}, \overline{M_{2,3}}\}$ и $C_{\mathrm{FT}}^0 = \{0, 5\}$. Для расчета элементов второй строки $Z_{1,j}$, $j = 0,1$, $R^1 = \{M_{1,0}, \overline{M_{2,0}}, \overline{M_{3,0}}, M_{1,1}, \overline{M_{2,1}}, \overline{M_{3,1}}, M_{1,2}, \overline{M_{2,2}}, \overline{M_{3,2}}, \overline{M_{1,3}}, M_{2,3}, M_{3,3}\}$ и $C_{\mathrm{FT}}^1 = \{6, 7\}$. Сложение чисел по модулю 2^α производится с помощью сумматоров CSA и сумматора KSA, а также MOMA. Для выполнения вычислений по модулю $2^\alpha - 1$ используется техника EAC, а отрицательные числа представляются в обратном коде (рис. 2.5.3б).



а)



б)

Рисунок 2.5.3. Архитектура устройства FTR для вычисления i-ой строки матрицы Z: а) по модулю 2^{α}; б) по модулю $2^{\alpha}-1$

Предлагаемые схемы устройств для двумерной фильтрации с использованием модифицированного метода Винограда на основе СОК представлены на рисунках 2.5.4, 2.5.5. Вычисления производятся по формуле (2.3.2). Преобразование маски фильтра $U = GWG^{\mathrm{T}}$ производится предварительно, и результат хранится в памяти устройства. На рисунке 2.5.4

представлена схема предлагаемого устройства фильтрации по модулю 2^α $\mathrm{F}(2\times 2,3\times 3)_{2^\alpha}$. Так как для выполнения операций с отрицательными числами требуется их представление в дополнительном коде, то в памяти устройства также хранятся корректирующие константы. Блок преобразования данных состоит из 16 устройств MOMA_{2^α}, представленных на рисунке 2.4.3а. Блок поэлементного умножения матриц состоит из 16 устройств MUL_{2^α}, изображенных на рисунке 2.4.4а. Блок итогового преобразования состоит из двух устройств FTR_{2^α}, представленных на рисунке 2.5.3а.

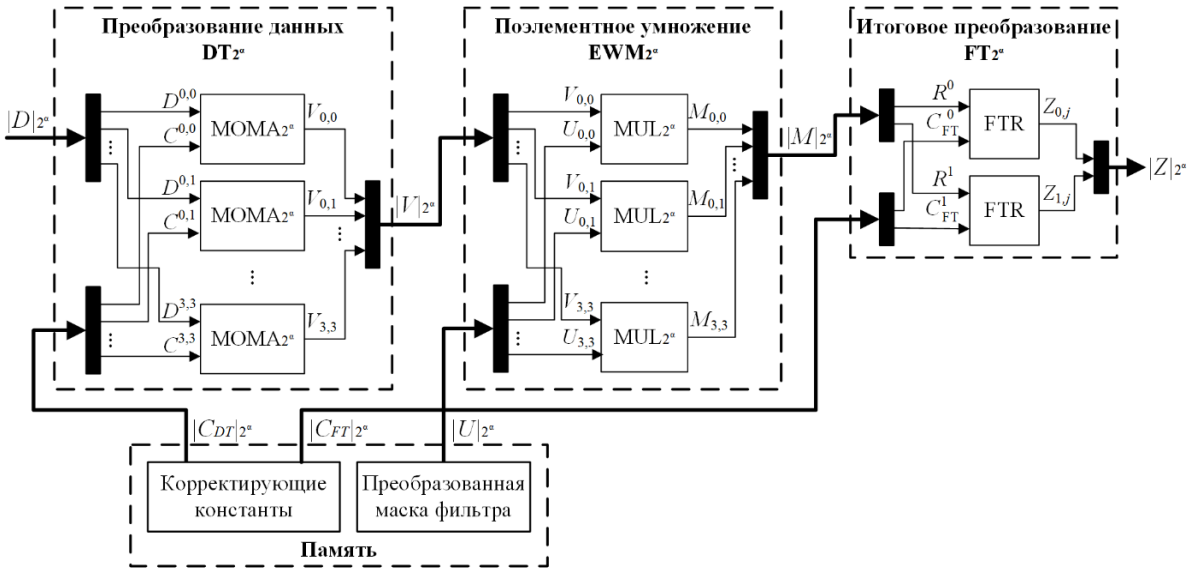


Рисунок 2.5.4. Архитектура устройства для двумерной фильтрации с использованием модифицированного метода Винограда $F(2\times 2,3\times 3)$ по модулю 2^α

На рисунке 2.5.5 изображена схема предлагаемого устройства фильтрации по модулю $2^\alpha-1$ $\mathrm{F}(2\times 2,3\times 3)_{2^\alpha-1}$. В памяти данного устройства хранится только преобразованная маска фильтра. Блоки преобразования данных, поэлементного умножения матриц и итогового преобразования состоят из соответствующих устройств, представленных на рисунках 2.4.3б, 2.4.4б и 2.5.3б.

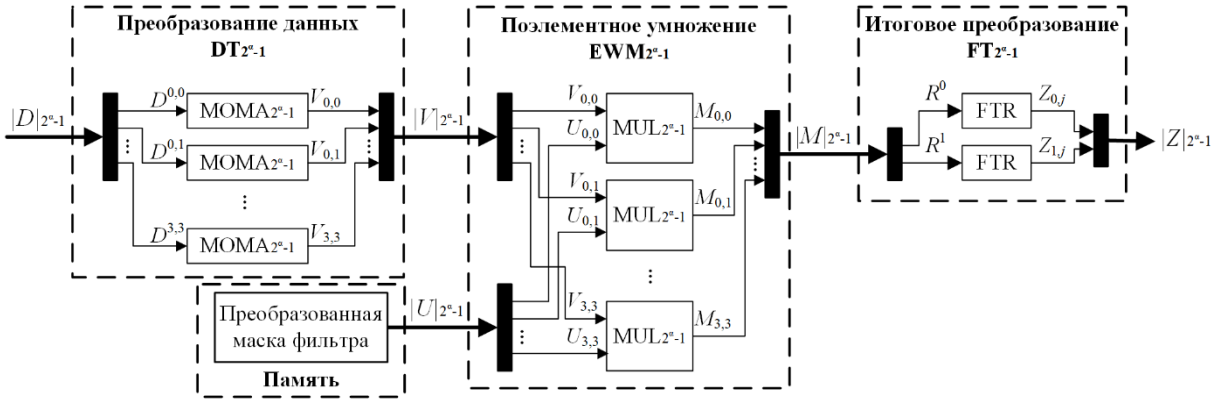


Рисунок 2.5.5. Архитектура устройства для двумерной фильтрации с использованием модифицированного метода Винограда $F(2 \times 2, 3 \times 3)$ по модулю $2^{\alpha} - 1$

На рисунке 2.5.6 представлена архитектура предлагаемого устройства $\mathrm{F}(2 \times 2{,}3 \times 3)_{\mathrm{RNS}}$ для фильтрации по методу Винограда в СОК с набором модулей $\{2^{\alpha_1}, 2^{\alpha_2} - 1, \ldots, 2^{\alpha_\eta} - 1\}$. На вход устройства подается массив данных $D_{RNS} = \{|D|_{2^{\alpha_1}}, |D|_{2^{\alpha_2}-1}, \ldots, |D|_{2^{\alpha_\eta}-1}\}$. Затем, входные данные обрабатываются параллельно устройствами $\mathrm{F}(2 \times 2, 3 \times 3)_{2^{\alpha}}$ и $\mathrm{F}(2 \times 2, 3 \times 3)_{2^{\alpha}-1}$. На выходе устройства $\mathrm{F}(2 \times 2, 3 \times 3)_{\mathrm{RNS}}$ формируется массив обработанных данных $Z_{RNS} = \{|Z|_{2^{\alpha_1}}, |Z|_{2^{\alpha_2}-1}, \ldots, |Z|_{2^{\alpha_\eta}-1}\}$.

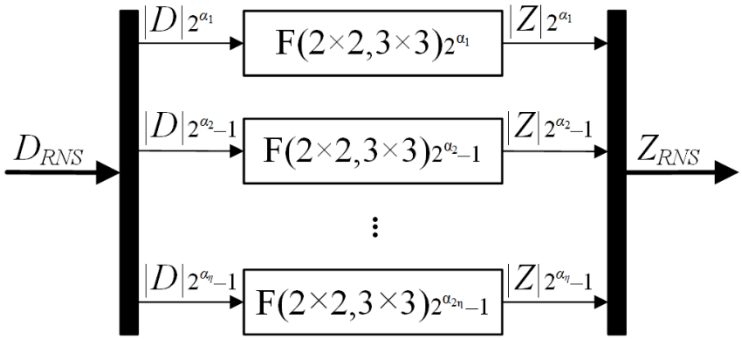


Рисунок 2.5.6. Архитектура устройства $\mathrm{F}(2 \times 2, 3 \times 3)_{\mathrm{RNS}}$ для двумерной фильтрации по методу Винограда в СОК с модулями вида 2^{α} и $2^{\alpha} - 1$

Для проверки эффективности предлагаемого устройства фильтрации изображений по методу Винограда было проведено аппаратное моделирование. Для эксперимента использован фильтр Гаусса размером 3×3:

$$W = \frac{1}{15}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \quad (2.5.4)$$

Как видно из равенства (2.5.4), коэффициенты маски фильтра являются вещественными числами. Для представления в цифровом устройстве их необходимо привести к целочисленному виду. В работе [26] представлен метод масштабирования коэффициентов для фильтров Гаусса. В соответствии с данным методом исходные коэффициенты фильтра умножаются на 2^n, где $n = 10 + t$ – параметр масштабирования. В данном случае $t = 1$ и $n = 11$. Далее коэффициенты округляются до ближайшего большего целого. Для фильтра Гаусса размером 3×3 маска фильтра с масштабированными коэффициентами имеет вид

$$\widetilde{W} = \begin{bmatrix} 137 & 274 & 137 \\ 274 & 410 & 274 \\ 137 & 274 & 137 \end{bmatrix}. \quad (2.5.5)$$

После наложения цифрового фильтра на изображение выполняется масштабирование полученных результатов на 2^{-n} и округление до ближайшего меньшего целого.

Динамический диапазон СОК для устройства фильтрации изображения с 8-битным представлением пикселей должен удовлетворять неравенству [26]

$$P \geq 255 \cdot \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} W(i,j). \quad (2.5.6)$$

Таким образом, для обработки изображения фильтром (2.5.5) динамический диапазон должен удовлетворять условию $P \geq 523770$. Тогда, 20 бит достаточно для разрядности устройства, выполняющего вычисления в ПСС. Для реализации вычислений в СОК был выбран набор модулей специального вида $\{2^7, 2^7 - 1, 2^6 - 1\}$, динамический диапазон которого равен $P = 1024128$ и удовлетворяет условию (2.5.6).

Для теоретической оценки задержки и площади цифрового устройства использовалась «unit-gate» модель [113]. Предлагаемая архитектура фильтра на основе метода Винограда $F(2 \times 2, 3 \times 3)$ и СОК с модулями специального

вида состоят из устройств MOMA, MUL и FTR, описанных выше. Параметры устройств MOMA и MUL вычисляются по формулам (2.4.19), (2.4.20), (2.4.26), (2.4.27). Устройство FTR_{2^α} состоит из сумматоров CSA и MOMA_{2^α}. Параметры устройства CSA и EAC-CSA

$$\begin{aligned} U_{\mathrm{delay}}(\mathrm{CSA}) &= U_{\mathrm{delay}}(\text{EAC-CSA}) = 4, \\ U_{\mathrm{area}}(\mathrm{CSA}) &= U_{\mathrm{area}}(\text{EAC-CSA}) = 7\alpha. \end{aligned} \tag{2.5.7}$$

Тогда, параметры устройства FTR по модулю 2^α

$$\begin{aligned} U_{\mathrm{delay}}(\mathrm{FTR}_{2^\alpha}) &= 2\log_2\alpha + 28{,}4, \\ U_{\mathrm{area}}(\mathrm{FTR}_{2^\alpha}) &= 6\alpha\log_2\alpha + 104\alpha + 2, \end{aligned} \tag{2.5.8}$$

а по модулю $2^\alpha - 1$

$$\begin{aligned} U_{\mathrm{delay}}(\mathrm{FTR}_{2^\alpha-1}) &= 2\log_2\alpha + 28{,}4, \\ U_{\mathrm{area}}(\mathrm{FTR}_{2^\alpha-1}) &= 6\alpha\log_2\alpha + 104\alpha. \end{aligned} \tag{2.5.9}$$

Предлагаемые устройства фильтрации на основе метода Винограда $F(2\times 2, 3\times 3)$ состоят из 16 устройств MUL, MOMA и из 2 устройств FTR по модулю 2^α и $2^\alpha - 1$. Таким образом, параметры предлагаемого устройства по модулю 2^α, основываясь на (2.4.19), (2.4.26) и (2.5.8), равны

$$\begin{aligned} &U_{\mathrm{delay}}(\mathrm{F}(2\times 2, 3\times 3)_{2^\alpha}) = \\ &= U_{\mathrm{delay}}(\mathrm{MOMA}_{2^\alpha}) + U_{\mathrm{delay}}(\mathrm{MUL}_{2^\alpha}) \\ &+ U_{\mathrm{delay}}(\mathrm{FTR}_{2^\alpha}) = \\ &= 12{,}8\log_2\alpha + 0{,}5\alpha^2 + 0{,}5\alpha + 56{,}8, \\ &U_{\mathrm{area}}(\mathrm{F}(2\times 2, 3\times 3)_{2^\alpha}) = \\ &= 16U_{\mathrm{area}}(\mathrm{MOMA}_{2^\alpha}) + 16U_{\mathrm{area}}(\mathrm{MUL}_{2^\alpha}) + 2U_{\mathrm{area}}(\mathrm{FTR}_{2^\alpha}) \\ &= 114\alpha\log_2\alpha + 120\alpha^2 + 528\alpha + 38, \end{aligned} \tag{2.5.10}$$

а по модулю $2^\alpha - 1$, основываясь на (2.4.20), (2.4.27) и (2.5.9), равны

$$\begin{aligned} &U_{\mathrm{delay}}(\mathrm{F}(2\times 2, 3\times 3)_{2^\alpha-1}) = \\ &= U_{\mathrm{delay}}(\mathrm{MOMA}_{2^\alpha-1}) + U_{\mathrm{delay}}(\mathrm{MUL}_{2^\alpha-1}) + U_{\mathrm{delay}}(\mathrm{FTR}_{2^\alpha-1}) = \\ &= 12{,}8\log_2\alpha + \alpha^2 + 50, \\ &U_{\mathrm{area}}(\mathrm{F}(2\times 2, 3\times 3)_{2^\alpha-1}) = 16U_{\mathrm{area}}(\mathrm{MOMA}_{2^\alpha-1}) + \end{aligned} \tag{2.5.11}$$

$$+16U_{\text{area}}(\text{MUL}_{2^{\alpha}-1}) + 2U_{\text{area}}(\text{FTR}_{2^{\alpha}-1}) =$$
$$= 108\alpha\log_2\alpha + 128\alpha^2 + 512\alpha.$$

Параметры предлагаемого устройства фильтрации $\text{F}(2\times 2, 3\times 3)_{\text{RNS}}$ с вычислениями в СОК с набором модулей $\{2^{\alpha_1}, 2^{\alpha_2}-1, \dots, 2^{\alpha_\eta}-1\}$, изображенного на рисунке 2.5.6, принимая во внимание (2.5.10) и (2.5.11), вычисляются как

$$U_{\text{delay}}(\text{F}(2\times 2, 3\times 3)_{RNS}) = \max\big(U_{\text{delay}}(\text{F}(2\times 2, 3\times 3)_{2^{\alpha_1}}),$$
$$\big\{U_{\text{delay}}(\text{F}(2\times 2, 3\times 3)_{2^{\alpha_i-1}})\big\}|2\le i\le\eta\big),$$
$$U_{\text{area}}(\text{F}(2\times 2, 3\times 3)_{RNS}) = U_{\text{area}}(\text{F}(2\times 2, 3\times 3)_{2^{\alpha_1}}) + \qquad (2.5.12)$$
$$+\sum_{i=2}^{\eta} U_{\text{area}}(\text{F}(2\times 2, 3\times 3)_{2^{\alpha_i-1}}).$$

Было проведено сравнение предлагаемого устройства фильтрации изображений с устройством для фильтрации по методу Винограда без использования СОК [111]. Кроме того, проведено сравнение с устройством фильтрации, состоящим из блоков MAC [101], и устройством, состоящим из усеченных блоков TMAC без арифметики СОК [6] и с использованием СОК [7].

Параметры устройства на основе метода Винограда в ПСС [111] рассчитываются по формулам (2.5.10), при условии, что разрядность $\alpha = 20$ бит.

Параметры фильтра, состоящего из MAC блоков, FIR(MAC) [101] рассчитываются по формулам (2.4.35). Параметры фильтра, состоящего из TMAC блоков, FIR(TMAC) [6] вычисляются по формулам (2.4.36), для рассматриваемого случая разрядность $\alpha = 20$ бит.

Параметры фильтра FIR(TMAC) с вычислениями в СОК [7] по модулю 2^{α} вычисляются по формулам (2.4.36), а по модулю $2^{\alpha}-1$ по формулам (2.4.37).

В таблице 2.5.1 представлены результаты теоретической оценки площади и задержки предложенного и известных фильтров на основе «unit-

gate» модели. Так же была произведена оценка времени обработки изображения размером 256×256 на основе задержки фильтра.

Теоретический анализ на основе «unit-gate» модели (табл. 2.5.1) параметров предлагаемого устройства и известных аналогов показал, что задержка предлагаемого фильтра на основе метода Винограда и арифметики СОК в 1,56-3,19 раза меньше, а время обработки одного изображения размером 256×256 в 2,39-12,77 раза меньше, по сравнению с другими известными устройствами. Отметим, что фильтр на основе TMAC блоков с вычислениями в СОК [7] обладает в 2,94-6,23 раза меньше площадью по сравнению с другими рассмотренными устройствами.

Таблица 2.5.1. Теоретические параметры двумерных фильтров с маской 3×3

Параметр	Известные методы				Предлагаемый метод
	[101]	[6]	[7]	[111]	
Задержка	431	317	211	323	**135**
Площадь	33804	32321	**10985**	68451	33188
Время обработки	28246016	20774912	13828096	5292032	**2211840**

Аппаратное моделирование на FPGA проводилось в среде автоматизированного проектирования Xilinx Vivado 2018.3 на языке описания аппаратуры VHDL для целевой платы Artix-7 xc7a200tffg1156-3 со стратегией оптимизации Flow_PerfOptimized_high. Результаты аппаратного моделирования устройств для фильтрации изображений представлены в таблице 2.5.2. Для оценки устройств использовались такие параметры как тактовая частота, количество LUT, энергопотребление, и производительность, которые были получены в результате симуляции в среде проектирования. Под производительностью устройства подразумевается количество обработанных кадров размером 256×256 пикселей в секунду.

Результаты аппаратного моделирования предлагаемых устройств фильтрации и известных аналогов (табл. 2.5.2) показали, что тактовая частота устройства, разработанного с использованием метода [7], на 33,33-80,65% выше по сравнению с другими рассмотренными методами. Также данный

метод требует на 53,41-79,79% меньше LUT и на 1,57-21,32% меньше энергозатрат, по сравнению с другими рассмотренными методами.

Таблица 2.5.2. Результаты аппаратного моделирования двумерных фильтров с маской 3×3

Параметр	Известные методы				Предлагаемый метод
	[101]	[6]	[7]	[111]	
Тактовая частота, МГц	31	36	**56**	37	42
Количество LUT	1872	1584	**738**	3633	3652
Энергопотребление, Вт	0,275	0,255	**0,251**	0,318	0,319
Производительность, кадр/с	473	549	854	2258	**2563**

Предлагаемый в данной работе метод характеризуется в 1,14-5,42 раза большей производительностью по сравнению с другими рассмотренными известными методами.

Предложенные архитектуры устройств могут быть использованы для фильтров с масками размером 3×3 и коэффициентами произвольного вида. Поэтому при проведении эксперимента не учитывалась симметрия фильтра Гаусса.

2.6 Проектирование фильтра на основе метода Винограда $\boldsymbol{F(2 \times 2, 5 \times 5)}$ с вычислениями в системе остаточных классов

Рассмотрим пример фильтрации по методу Винограда $F(2 \times 2, 5 \times 5)$. На рисунке 2.6.1 представлен процесс фильтрации изображения размерности 256×256 методом Винограда $F(2 \times 2, 5 \times 5)$. Этапы фильтрации фрагмента изображения размера 6×6 методом Винограда $F(2 \times 2, 5 \times 5)$ представлены на рисунке 2.6.2.

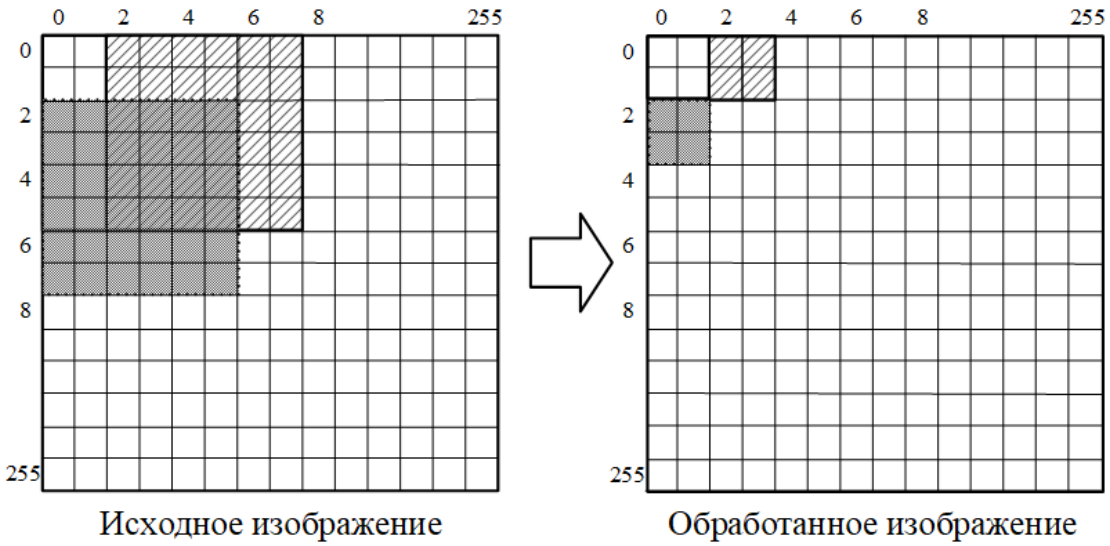


Рисунок 2.6.1. Процесс фильтрации изображения по методу Винограда $F(2 \times 2, 5 \times 5)$

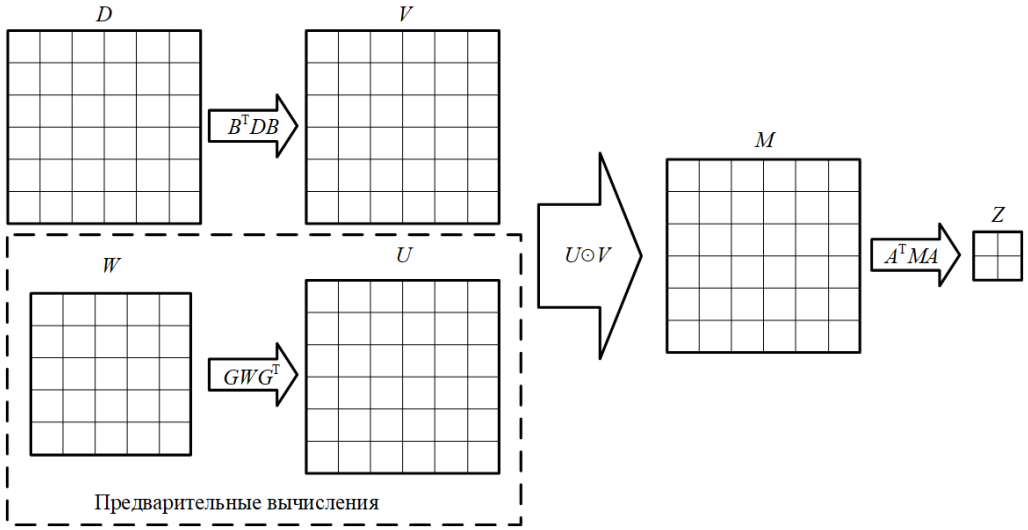


Рисунок 2.6.2. Этапы фильтрации фрагмента изображения по методу Винограда $F(2 \times 2, 5 \times 5)$

Матрицы преобразования для $F(2 \times 2, 5 \times 5)$ имеют вид

$$B = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & 4 & -2 & 2 & 4 \\ -5 & -4 & -4 & -1 & -1 & 0 \\ 0 & 1 & -1 & 2 & -2 & -5 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$G = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 & 0 \\ -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} \\ -\frac{1}{6} & \frac{1}{6} & -\frac{1}{6} & \frac{1}{6} & -\frac{1}{6} \\ \frac{1}{24} & \frac{1}{12} & \frac{1}{6} & \frac{1}{3} & \frac{2}{3} \\ \frac{1}{24} & -\frac{1}{12} & \frac{1}{6} & -\frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & -1 \\ 1 & 2 \\ 1 & -2 \\ 0 & 1 \end{bmatrix}. \tag{2.6.1}$$

В случае двумерной фильтрации $F(2 \times 2, 5 \times 5)$ вычисления выполняются по формуле (2.3.2). Элементы нулевой строки матрицы $V = B^{\mathrm{T}}DB$ формируются следующим образом:

$$\begin{gathered} V_{0,0} = 4\left(4d_{0,0} - 5d_{2,0} + d_{4,0}\right) - 5\left(4d_{0,2} - 5d_{2,2} + d_{4,2}\right) + \\ + \left(4d_{0,4} - 5d_{2,4} + d_{4,4}\right), \\ V_{0,1} = -4\left(4d_{0,1} - 5d_{2,1} + d_{4,1}\right) - 4\left(4d_{0,2} - 5d_{2,2} + d_{4,2}\right) + \\ + \left(4d_{0,3} - 5d_{2,3} + d_{4,3}\right) + \left(4d_{0,4} - 5d_{2,4} + d_{4,4}\right), \\ V_{0,2} = 4\left(4d_{0,1} - 5d_{2,1} + d_{4,1}\right) - 4\left(4d_{0,2} - 5d_{2,2} + d_{4,2}\right) - \\ - \left(4d_{0,3} - 5d_{2,3} + d_{4,3}\right) + \left(4d_{0,4} - 5d_{2,4} + d_{4,4}\right), \\ V_{0,3} = -2\left(4d_{0,1} - 5d_{2,1} + d_{4,1}\right) - \left(4d_{0,2} - 5d_{2,2} + d_{4,2}\right) + \\ + 2\left(4d_{0,3} - 5d_{2,3} + d_{4,3}\right) + \left(4d_{0,4} - 5d_{2,4} + d_{4,4}\right), \\ V_{0,4} = 2\left(4d_{0,1} - 5d_{2,1} + d_{4,1}\right) - \left(4d_{0,2} - 5d_{2,2} + d_{4,2}\right) - \\ - 2\left(4d_{0,3} - 5d_{2,3} + d_{4,3}\right) + \left(4d_{0,4} - 5d_{2,4} + d_{4,4}\right), \\ V_{0,5} = 4\left(4d_{0,1} - 5d_{2,1} + d_{4,1}\right) - 5\left(4d_{0,3} - 5d_{2,3} + d_{4,3}\right) + \\ + \left(4d_{0,5} - 5d_{2,5} + d_{4,5}\right); \end{gathered} \tag{2.6.2}$$

первой строки

$$V_{1,0} = 4\left(-4d_{1,0} - 4d_{2,0} + d_{3,0} + d_{4,0}\right) -$$
$$-5\left(-4d_{1,2} - 4d_{2,2} + d_{3,2} + d_{4,2}\right) + \left(-4d_{1,4} - 4d_{2,4} + d_{3,4} + d_{4,4}\right),$$
$$V_{1,1} = -4\left(-4d_{1,1} - 4d_{2,1} + d_{3,1} + d_{4,1}\right) -$$
$$-4\left(-4d_{1,2} - 4d_{2,2} + d_{3,2} + d_{4,2}\right) +$$
$$+\left(-4d_{1,3} - 4d_{2,3} + d_{3,3} + d_{4,3}\right) + \left(-4d_{1,4} - 4d_{2,4} + d_{3,4} + d_{4,4}\right),$$
$$V_{1,2} = 4\left(-4d_{1,1} - 4d_{2,1} + d_{3,1} + d_{4,1}\right) -$$
$$-4\left(-4d_{1,2} - 4d_{2,2} + d_{3,2} + d_{4,2}\right) -$$
$$-\left(-4d_{1,3} - 4d_{2,3} + d_{3,3} + d_{4,3}\right) + \left(-4d_{1,4} - 4d_{2,4} + d_{3,4} + d_{4,4}\right),$$
$$V_{1,3} = -2\left(-4d_{1,1} - 4d_{2,1} + d_{3,1} + d_{4,1}\right) - \quad (2.6.3)$$
$$-\left(-4d_{1,2} - 4d_{2,2} + d_{3,2} + d_{4,2}\right) +$$
$$+2\left(-4d_{1,3} - 4d_{2,3} + d_{3,3} + d_{4,3}\right) + \left(-4d_{1,4} - 4d_{2,4} + d_{3,4} + d_{4,4}\right),$$
$$V_{1,4} = 2\left(-4d_{1,1} - 4d_{2,1} + d_{3,1} + d_{4,1}\right) -$$
$$-\left(-4d_{1,2} - 4d_{2,2} + d_{3,2} + d_{4,2}\right) -$$
$$-2\left(-4d_{1,3} - 4d_{2,3} + d_{3,3} + d_{4,3}\right) + \left(-4d_{1,4} - 4d_{2,4} + d_{3,4} + d_{4,4}\right),$$
$$V_{1,5} = 4\left(-4d_{1,1} - 4d_{2,1} + d_{3,1} + d_{4,1}\right) -$$
$$-5\left(-4d_{1,3} - 4d_{2,3} + d_{3,3} + d_{4,3}\right) + \left(-4d_{1,5} - 4d_{2,5} + d_{3,5} + d_{4,5}\right);$$

второй строки

$$V_{2,0} = 4\left(4d_{1,0} - 4d_{2,0} - d_{3,0} + d_{4,0}\right) -$$
$$-5\left(4d_{1,2} - 4d_{2,2} - d_{3,2} + d_{4,2}\right) + \left(4d_{1,4} - 4d_{2,4} - d_{3,4} + d_{4,4}\right),$$
$$V_{2,1} = -4\left(4d_{1,1} - 4d_{2,1} - d_{3,1} + d_{4,1}\right) -$$
$$-4\left(4d_{1,2} - 4d_{2,2} - d_{3,2} + d_{4,2}\right) + \left(4d_{1,3} - 4d_{2,3} - d_{3,3} + d_{4,3}\right) +$$
$$+\left(4d_{1,4} - 4d_{2,4} - d_{3,4} + d_{4,4}\right), \quad (2.6.4)$$
$$V_{2,2} = 4\left(4d_{1,1} - 4d_{2,1} - d_{3,1} + d_{4,1}\right) -$$
$$-4\left(4d_{1,2} - 4d_{2,2} - d_{3,2} + d_{4,2}\right) - \left(4d_{1,3} - 4d_{2,3} - d_{3,3} + d_{4,3}\right) +$$
$$+\left(4d_{1,4} - 4d_{2,4} - d_{3,4} + d_{4,4}\right),$$
$$V_{2,3} = -2\left(4d_{1,1} - 4d_{2,1} - d_{3,1} + d_{4,1}\right) -$$

$$-\left(4d_{1,2}-4d_{2,2}-d_{3,2}+d_{4,2}\right)+2\left(4d_{1,3}-4d_{2,3}-d_{3,3}+d_{4,3}\right)+$$
$$+\left(4d_{1,4}-4d_{2,4}-d_{3,4}+d_{4,4}\right),$$
$$V_{2,4}=2\left(4d_{1,1}-4d_{2,1}-d_{3,1}+d_{4,1}\right)-$$
$$-\left(4d_{1,2}-4d_{2,2}-d_{3,2}+d_{4,2}\right)-2\left(4d_{1,3}-4d_{2,3}-d_{3,3}+d_{4,3}\right)+$$
$$+\left(4d_{1,4}-4d_{2,4}-d_{3,4}+d_{4,4}\right),$$
$$V_{2,5}=4\left(4d_{1,1}-4d_{2,1}-d_{3,1}+d_{4,1}\right)-$$
$$-5\left(4d_{1,3}-4d_{2,3}-d_{3,3}+d_{4,3}\right)+\left(4d_{1,5}-4d_{2,5}-d_{3,5}+d_{4,5}\right);$$

третей строки

$$V_{3,0}=4\left(-2d_{1,0}-d_{2,0}+2d_{3,0}+d_{4,0}\right)-$$
$$-5\left(-2d_{1,2}-d_{2,2}+2d_{3,2}+d_{4,2}\right)+\left(-2d_{1,4}-d_{2,4}+2d_{3,4}+d_{4,4}\right),$$
$$V_{3,1}=-4\left(-2d_{1,1}-d_{2,1}+2d_{3,1}+d_{4,1}\right)-$$
$$-4\left(-2d_{1,2}-d_{2,2}+2d_{3,2}+d_{4,2}\right)+$$
$$+\left(-2d_{1,3}-d_{2,3}+2d_{3,3}+d_{4,3}\right)+\left(-2d_{1,4}-d_{2,4}+2d_{3,4}+d_{4,4}\right),$$
$$V_{3,2}=4\left(-2d_{1,1}-d_{2,1}+2d_{3,1}+d_{4,1}\right)-$$
$$-4\left(-2d_{1,2}-d_{2,2}+2d_{3,2}+d_{4,2}\right)-$$
$$-\left(-2d_{1,3}-d_{2,3}+2d_{3,3}+d_{4,3}\right)+\left(-2d_{1,4}-d_{2,4}+2d_{3,4}+d_{4,4}\right), \quad (2.6.5)$$
$$V_{3,3}=-2\left(-2d_{1,1}-d_{2,1}+2d_{3,1}+d_{4,1}\right)-$$
$$-\left(-2d_{1,2}-d_{2,2}+2d_{3,2}+d_{4,2}\right)+$$
$$+2\left(-2d_{1,3}-d_{2,3}+2d_{3,3}+d_{4,3}\right)+\left(-2d_{1,4}-d_{2,4}+2d_{3,4}+d_{4,4}\right),$$
$$V_{3,4}=2\left(-2d_{1,1}-d_{2,1}+2d_{3,1}+d_{4,1}\right)-$$
$$-\left(-2d_{1,2}-d_{2,2}+2d_{3,2}+d_{4,2}\right)-$$
$$-2\left(-2d_{1,3}-d_{2,3}+2d_{3,3}+d_{4,3}\right)+\left(-2d_{1,4}-d_{2,4}+2d_{3,4}+d_{4,4}\right),$$
$$V_{3,5}=4\left(-2d_{1,1}-d_{2,1}+2d_{3,1}+d_{4,1}\right)-$$
$$-5\left(-2d_{1,3}-d_{2,3}+2d_{3,3}+d_{4,3}\right)+\left(-2d_{1,5}-d_{2,5}+2d_{3,5}+d_{4,5}\right);$$

четвертой строки

$$V_{4,0}=4\left(2d_{1,0}-d_{2,0}-2d_{3,0}+d_{4,0}\right)- \quad (2.6.6)$$

$$-5\left(2d_{1,2}-d_{2,2}-2d_{3,2}+d_{4,2}\right)+\left(2d_{1,4}-d_{2,4}-2d_{3,4}+d_{4,4}\right),$$

$$V_{4,1}=-4\left(2d_{1,1}-d_{2,1}-2d_{3,1}+d_{4,1}\right)-$$

$$-4\left(2d_{1,2}-d_{2,2}-2d_{3,2}+d_{4,2}\right)+\left(2d_{1,3}-d_{2,3}-2d_{3,3}+d_{4,3}\right)+$$

$$+\left(2d_{1,4}-d_{2,4}-2d_{3,4}+d_{4,4}\right),$$

$$V_{4,2}=4\left(2d_{1,1}-d_{2,1}-2d_{3,1}+d_{4,1}\right)-$$

$$-4\left(2d_{1,2}-d_{2,2}-2d_{3,2}+d_{4,2}\right)-\left(2d_{1,3}-d_{2,3}-2d_{3,3}+d_{4,3}\right)+$$

$$+\left(2d_{1,4}-d_{2,4}-2d_{3,4}+d_{4,4}\right),$$

$$V_{4,3}=-2\left(2d_{1,1}-d_{2,1}-2d_{3,1}+d_{4,1}\right)-$$

$$-\left(2d_{1,2}-d_{2,2}-2d_{3,2}+d_{4,2}\right)+2\left(2d_{1,3}-d_{2,3}-2d_{3,3}+d_{4,3}\right)+$$

$$+\left(2d_{1,4}-d_{2,4}-2d_{3,4}+d_{4,4}\right),$$

$$V_{4,4}=2\left(2d_{1,1}-d_{2,1}-2d_{3,1}+d_{4,1}\right)-$$

$$-\left(2d_{1,2}-d_{2,2}-2d_{3,2}+d_{4,2}\right)-2\left(2d_{1,3}-d_{2,3}-2d_{3,3}+d_{4,3}\right)+$$

$$+\left(2d_{1,4}-d_{2,4}-2d_{3,4}+d_{4,4}\right),$$

$$V_{4,5}=4\left(2d_{1,1}-d_{2,1}-2d_{3,1}+d_{4,1}\right)-$$

$$-5\left(2d_{1,3}-d_{2,3}-2d_{3,3}+d_{4,3}\right)+\left(2d_{1,5}-d_{2,5}-2d_{3,5}+d_{4,5}\right);$$

пятой строки

$$V_{5,0}=4\left(4d_{1,0}-5d_{3,0}+d_{5,0}\right)-5\left(4d_{1,2}-5d_{3,2}+d_{5,2}\right)+$$

$$+\left(4d_{1,4}-5d_{3,4}+d_{5,4}\right),$$

$$V_{5,1}=-4\left(4d_{1,1}-5d_{3,1}+d_{5,1}\right)-4\left(4d_{1,2}-5d_{3,2}+d_{5,2}\right)+$$

$$+\left(4d_{1,3}-5d_{3,3}+d_{5,3}\right)+\left(4d_{1,4}-5d_{3,4}+d_{5,4}\right),$$

$$V_{5,2}=4\left(4d_{1,1}-5d_{3,1}+d_{5,1}\right)-4\left(4d_{1,2}-5d_{3,2}+d_{5,2}\right)-$$

$$-\left(4d_{1,3}-5d_{3,3}+d_{5,3}\right)+\left(4d_{1,4}-5d_{3,4}+d_{5,4}\right), \qquad (2.6.7)$$

$$V_{5,3}=-2\left(4d_{1,1}-5d_{3,1}+d_{5,1}\right)-\left(4d_{1,2}-5d_{3,2}+d_{5,2}\right)+$$

$$+2\left(4d_{1,3}-5d_{3,3}+d_{5,3}\right)+\left(4d_{1,4}-5d_{3,4}+d_{5,4}\right),$$

$$V_{5,4}=2\left(4d_{1,1}-5d_{3,1}+d_{5,1}\right)-\left(4d_{1,2}-5d_{3,2}+d_{5,2}\right)-$$

$$-2\left(4d_{1,3}-5d_{3,3}+d_{5,3}\right)+\left(4d_{1,4}-5d_{3,4}+d_{5,4}\right),$$

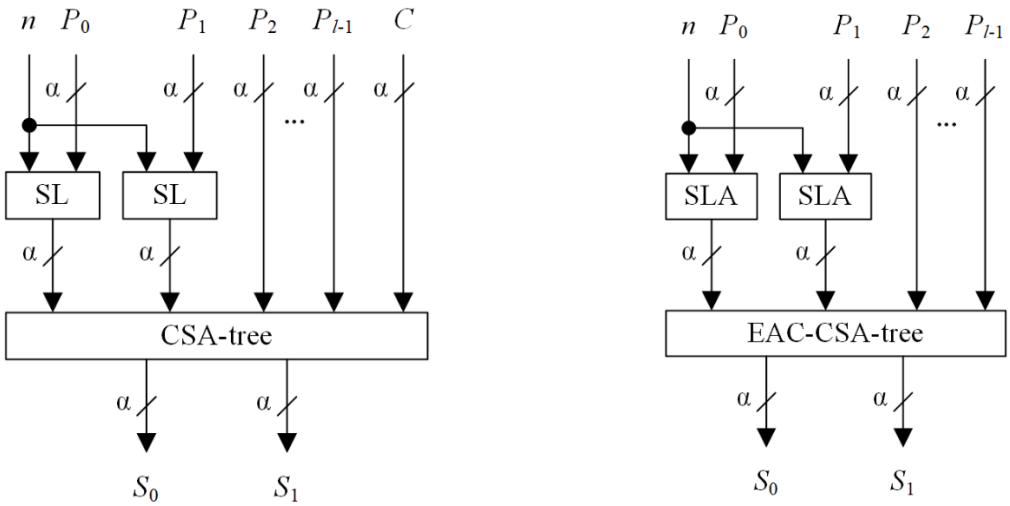
$$V_{5,5} = 4\left(4d_{1,1} - 5d_{3,1} + d_{5,1}\right) - 5\left(4d_{1,3} - 5d_{3,3} + d_{5,3}\right) + \\ + \left(4d_{1,5} - 5d_{3,5} + d_{5,5}\right).$$

Элементы матрицы $Z = A^{\mathrm{T}}MA$ вычисляются следующим образом

$$\begin{gathered}
Z_{0,0} = \left(M_{0,0} + M_{1,0} + M_{2,0} + M_{3,0} + M_{4,0}\right) + \\
+\left(M_{0,1} + M_{1,1} + M_{2,1} + M_{3,1} + M_{4,1}\right) + \\
+\left(M_{0,2} + M_{1,2} + M_{2,2} + M_{3,2} + M_{4,2}\right) + \\
+\left(M_{0,3} + M_{1,3} + M_{2,3} + M_{3,3} + M_{4,3}\right) + \\
+\left(M_{0,4} + M_{1,4} + M_{2,4} + M_{3,4} + M_{4,4}\right), \\
Z_{0,1} = \left(M_{0,1} + M_{1,1} + M_{2,1} + M_{3,1} + M_{4,1}\right) - \\
-\left(M_{0,2} + M_{1,2} + M_{2,2} + M_{3,2} + M_{4,2}\right) + \\
+2\left(M_{0,3} + M_{1,3} + M_{2,3} + M_{3,3} + M_{4,3}\right) - \\
-2\left(M_{0,4} + M_{1,4} + M_{2,4} + M_{3,4} + M_{4,4}\right) + \\
+\left(M_{0,5} + M_{1,5} + M_{2,5} + M_{3,5} + M_{4,5}\right), \\
Z_{1,0} = \left(M_{1,0} - M_{2,0} + 2M_{3,0} - 2M_{4,0} + M_{5,0}\right) + \\
+\left(M_{1,1} - M_{2,1} + 2M_{3,1} - 2M_{4,1} + M_{5,1}\right) + \\
+\left(M_{1,2} - M_{2,2} + 2M_{3,2} - 2M_{4,2} + M_{5,2}\right) + \\
+\left(M_{1,3} - M_{2,3} + 2M_{3,3} - 2M_{4,3} + M_{5,3}\right) + \\
+\left(M_{1,4} - M_{2,4} + 2M_{3,4} - 2M_{4,4} + M_{5,4}\right), \\
Z_{1,1} = \left(M_{1,1} - M_{2,1} + 2M_{3,1} - 2M_{4,1} + M_{5,1}\right) - \\
-\left(M_{1,2} - M_{2,2} + 2M_{3,2} - 2M_{4,2} + M_{5,2}\right) + \\
+2\left(M_{1,3} - M_{2,3} + 2M_{3,3} - 2M_{4,3} + M_{5,3}\right) - \\
-2\left(M_{1,4} - M_{2,4} + 2M_{3,4} - 2M_{4,4} + M_{5,4}\right) + \\
+\left(M_{1,5} - M_{2,5} + 2M_{3,5} - 2M_{4,5} + M_{5,5}\right).
\end{gathered} \tag{2.6.8}$$

Преобразование данных по формулам (2.6.2)-(2.6.8) по модулю 2^{α} производится с помощью устройства $\mathrm{DTE}_{2^{\alpha}}$ (Data Transform Element), представленного на рисунке 2.6.3а. На вход устройства подается вектор $\{P_i\}$, где $0 \le i < l$. Поскольку отрицательные числа по модулю 2^{α} представляются

в дополнительном коде, то вводится корректирующая константа C, равная количеству отрицательных элементов вектора $\{P_i\}$. Блоки SL (Shift Left) производят сдвиг влево на n бит, что соответствует умножению на 2^n. Далее производится сложение с помощью дерева сумматоров CSA-tree. Преобразование данных по модулю $2^\alpha - 1$ производится с помощью устройства $\mathrm{DTE}_{2^\alpha-1}$ (рис. 2.6.3б) и отличается тем, что используется техника циклического переноса старших бит EAC, а устройства SLA (Shift Left Around) выполняют циклический сдвиг на n бит. Так как отрицательные числа по модулю $2^\alpha - 1$ представляются в обратном коде, то добавление корректирующей константы не требуется.



а) б)

Рисунок 2.6.3. Архитектура устройства преобразования данных DTE: а) по модулю 2^α; б) по модулю $2^\alpha - 1$

Вычисление элементов одной строки матрицы V по модулю 2^α производится устройством DTR_{2^α} (рис. 2.6.4). Данное устройство выполняет преобразование данных из матрицы D и генерирует элементы $V_{i,j}$, $0 \le i \le 5$, $0 \le j \le 5$. Для вычисления элементов i-ой строки матрицы V по модулю 2^α, на вход DTR_{2^α} подается вектор данных $D^i = \{D_0^i, D_1^i, \dots, D_{23}^i\}$, корректирующий коэффициент $C = 2$ и вектор смещения $N^i = \{N_0^i, N_1^i, N_2^i\}$. Например, для вычисления элементов строки $V_{0,j}$, на вход поступает вектор

данных $D^0 = \{d_{0,0}, \overline{d_{2,0}}, \overline{d_{2,0}}, d_{4,0}, d_{0,1}, \overline{d_{2,1}}, \overline{d_{2,1}}, d_{4,1}, d_{0,2}, \overline{d_{2,2}}, \overline{d_{2,2}}, d_{4,2}, d_{0,3}, \overline{d_{2,3}}, \overline{d_{2,3}}, d_{4,3}, d_{0,4}, \overline{d_{2,4}}, \overline{d_{2,4}}, d_{4,4}, d_{0,5}, \overline{d_{2,5}}, \overline{d_{2,5}}, d_{4,5}\}$, корректирующий коэффициент $C = 2$ вектор сдвига $N^0 = \{1, 2, 2\}$. Входные данные поступают на устройства преобразования данных DTE_{2^α}, результат складывается с помощью сумматоров MOMA_{2^α}. Таким образом, устройство преобразования данных по модулю 2^α (обозначим его DT_{2^α}) состоит из 6 устройств DTR_{2^α}.

Вычисление элементов матрицы V по модулю $2^\alpha - 1$ требует представления отрицательных чисел в обратном коде, то есть выполнения инверсии числа, следовательно, корректирующие константы не участвуют в вычислениях. Следовательно, устройство преобразования данных по модулю $2^\alpha - 1$ (обозначим его $\mathrm{DT}_{2^\alpha - 1}$) состоит из 6 устройств $\mathrm{DTR}_{2^\alpha - 1}$, представленных на рисунке 2.6.5, на входы которых поступают векторы D^i и N^i.

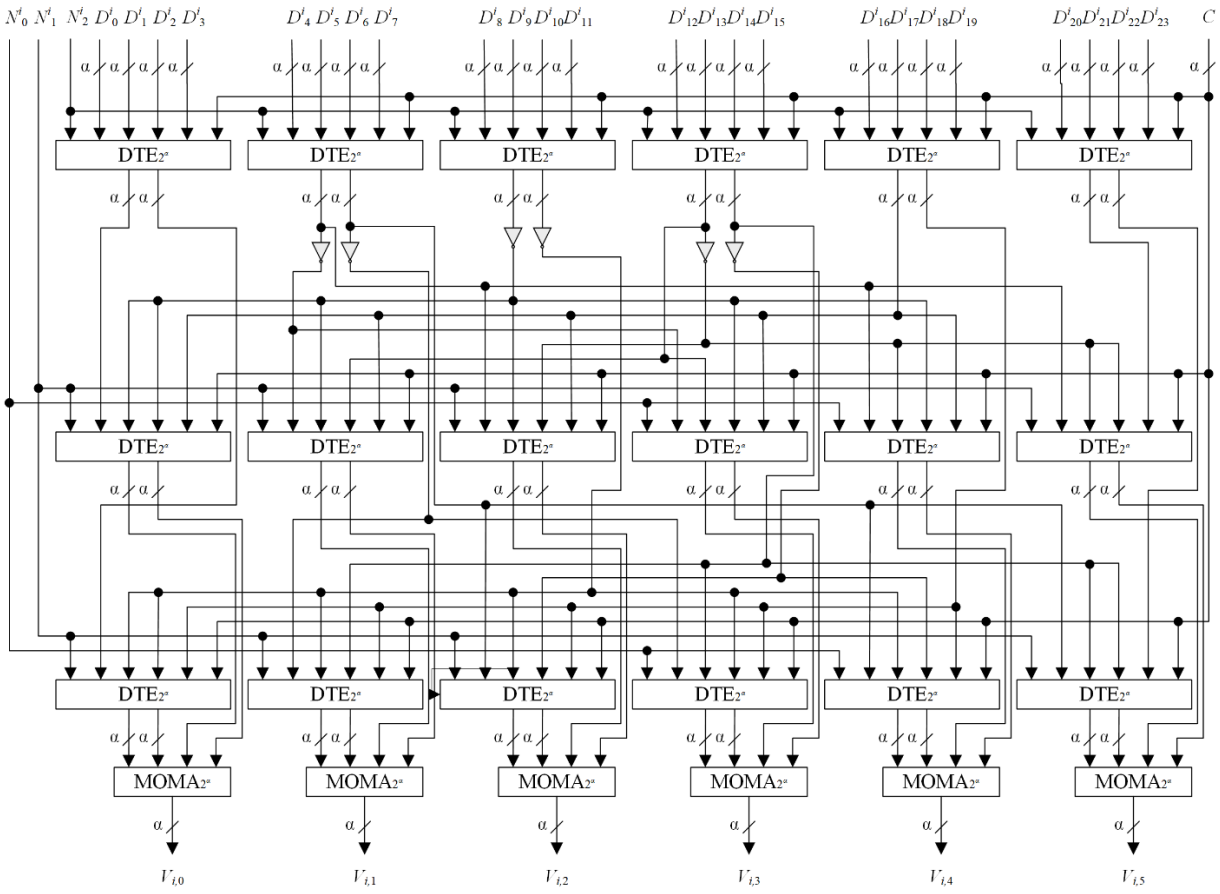


Рисунок 2.6.4. Архитектура устройства DTR_{2^α} для вычисления элементов i-ой строки матрицы V по модулю 2^α

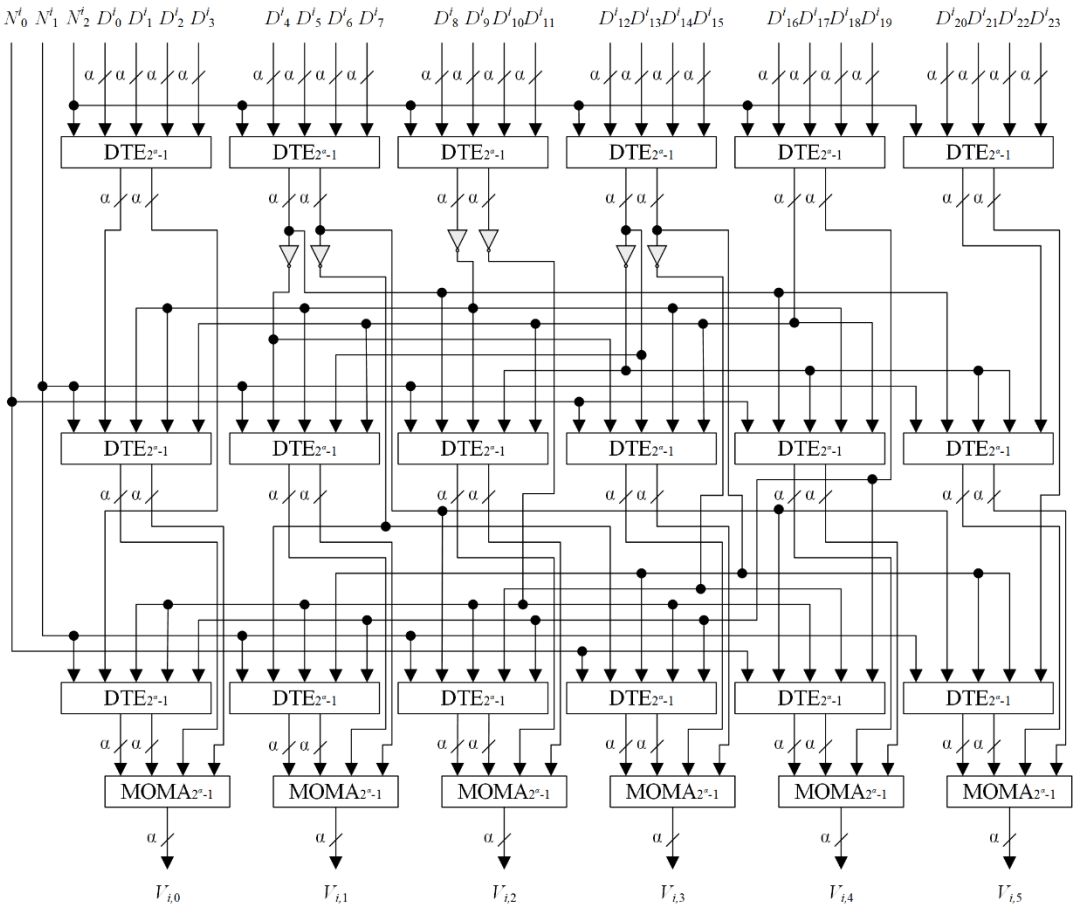


Рисунок 2.6.5. Архитектура устройства $\mathrm{DTR}_{2^\alpha-1}$ для вычисления элементов i-ой строки матрицы V по модулю $2^\alpha - 1$

Поэлементное умножение матриц U и V производится с помощью устройств EWM_{2^α} и $\mathrm{EWM}_{2^\alpha-1}$, состоящих из 36 параллельных умножителей двух чисел MUL_{2^α} и $\mathrm{MUL}_{2^\alpha-1}$ соответственно, которые изображены на рисунке 2.4.4. На вход устройства поступают элементы матриц U и V. Таким образом формируется матрица M размером 6×6.

Вычисление элементов одной строки матрицы Z по модулю 2^α производится устройством FTR_{2^α} (рис. 2.6.6). Данное устройство выполняет итоговое преобразование данных из матрицы M и генерирует элементы $Z_{i,j}$, $0 \le i \le 1,\ 0 \le j \le 1$. Для вычисления элементов i-ой строки матрицы Z по модулю 2^α на вход FTR_{2^α} подается вектор данных $R^i = \{R^i_0, R^i_1, \dots, R^i_{29}\}$, корректирующие коэффициенты $C^i = \{C^i_0, C^i_1, C^i_2\}$ и вектор смещения $N^i =$

$\{N_0^i, N_1^i, N_2^i\}$. Например, для вычисления элементов строки $Z_{0,j}$, на вход поступает вектор данных $R^0 = \{M_{0,0}, M_{1,0}, M_{2,0}, M_{3,0}, M_{4,0}, M_{0,1}, M_{1,1}, M_{2,1}, M_{3,1}, M_{4,1}, M_{0,2}, M_{1,2}, M_{2,2}, M_{3,2}, M_{4,2}, M_{0,3}, M_{1,3}, M_{2,3}, M_{3,3}, M_{4,3}, M_{0,4}, M_{1,4}, M_{2,4}, M_{3,4}, M_{4,4}, M_{0,5}, M_{1,5}, M_{2,5}, M_{3,5}, M_{4,5}\}$, корректирующие коэффициенты $C^0 = \{0, 0, 2\}$ и вектор сдвига $N^0 = \{1, 0, 0\}$. Входные данные поступают на устройства преобразования данных DTE_{2^α}, результат складывается с помощью сумматоров MOMA_{2^α}. Таким образом, устройство преобразования данных по модулю 2^α (обозначим его FT_{2^α}) состоит из 2 устройств FTR_{2^α}.

Вычисление элементов матрицы Z по модулю $2^\alpha - 1$ требует представления отрицательных чисел в обратном коде, следовательно, корректирующие константы не участвуют в вычислениях. Следовательно, устройство преобразования данных по модулю $2^\alpha - 1$ (обозначим его $\mathrm{FT}_{2^\alpha-1}$) состоит из 2 устройств $\mathrm{FTR}_{2^\alpha-1}$, представленных на рисунке 2.6.7, на входы которых поступают векторы R^i и N^i.

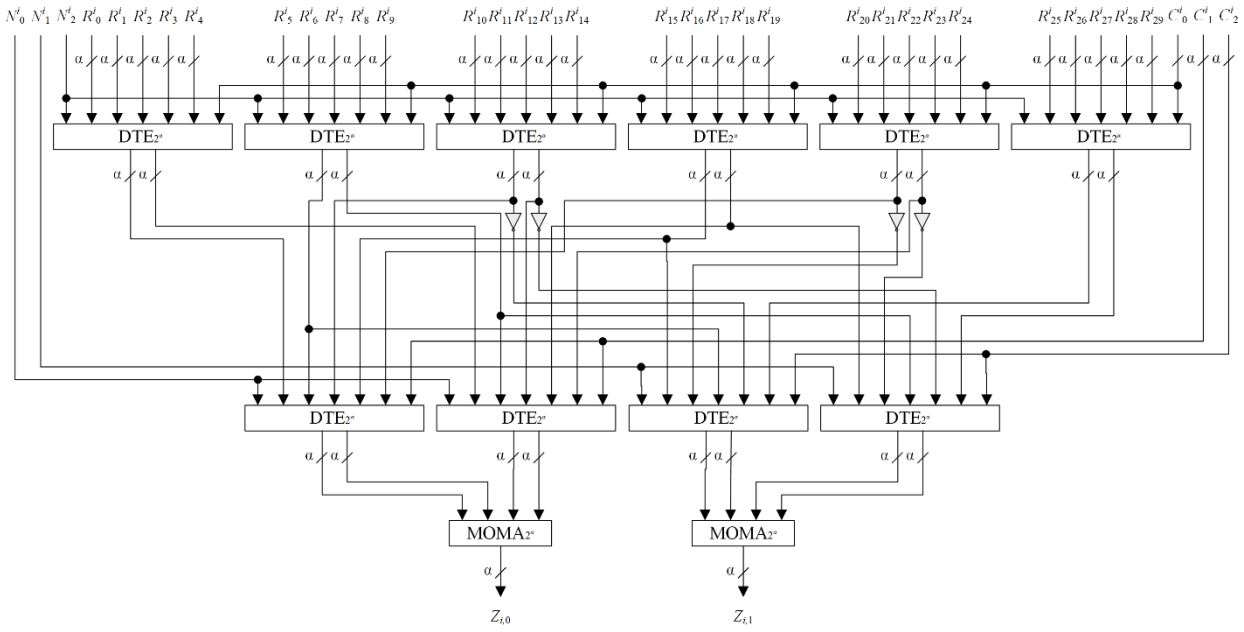


Рисунок 2.6.6. Архитектура устройства FTR_{2^α} для вычисления элементов i-ой строки матрицы Z по модулю 2^α

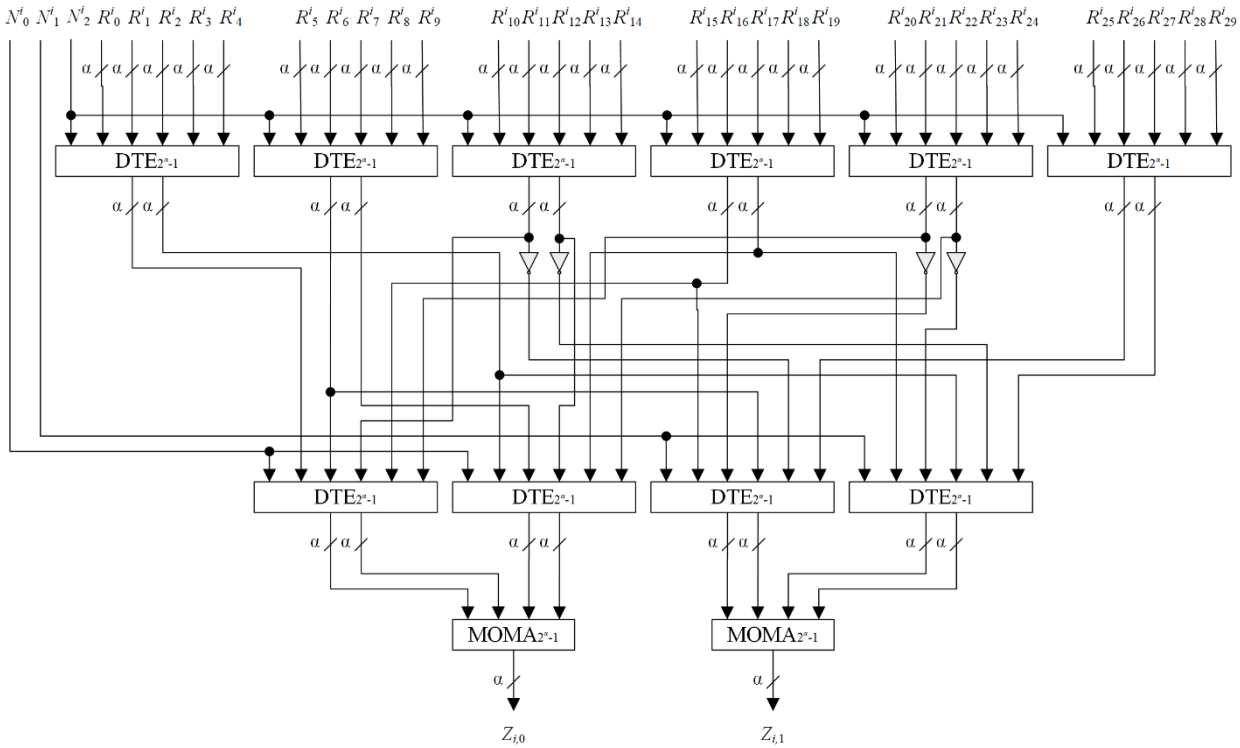


Рисунок 2.6.7. Архитектура устройства $\mathrm{FTR}_{2^\alpha-1}$ для вычисления элементов i-ой строки матрицы Z по модулю $2^\alpha - 1$

На рисунке 2.6.8 представлена схема предлагаемого устройства фильтрации $\mathrm{F}(2\times 2, 5\times 5)_{2^\alpha}$ по модулю 2^α. Преобразование маски фильтра $U = GWG^{\mathrm{T}}$ производится предварительно, и результат хранится в памяти устройства. Так как для выполнения операций с отрицательными числами требуется их представление в дополнительном коде, то в памяти устройства также хранятся корректирующие константы. На рисунке 2.6.9 изображена схема предлагаемого устройства фильтрации $\mathrm{F}(2\times 2, 5\times 5)_{2^\alpha-1}$ по модулю $2^\alpha - 1$. В памяти данного устройства хранится только преобразованная маска фильтра.

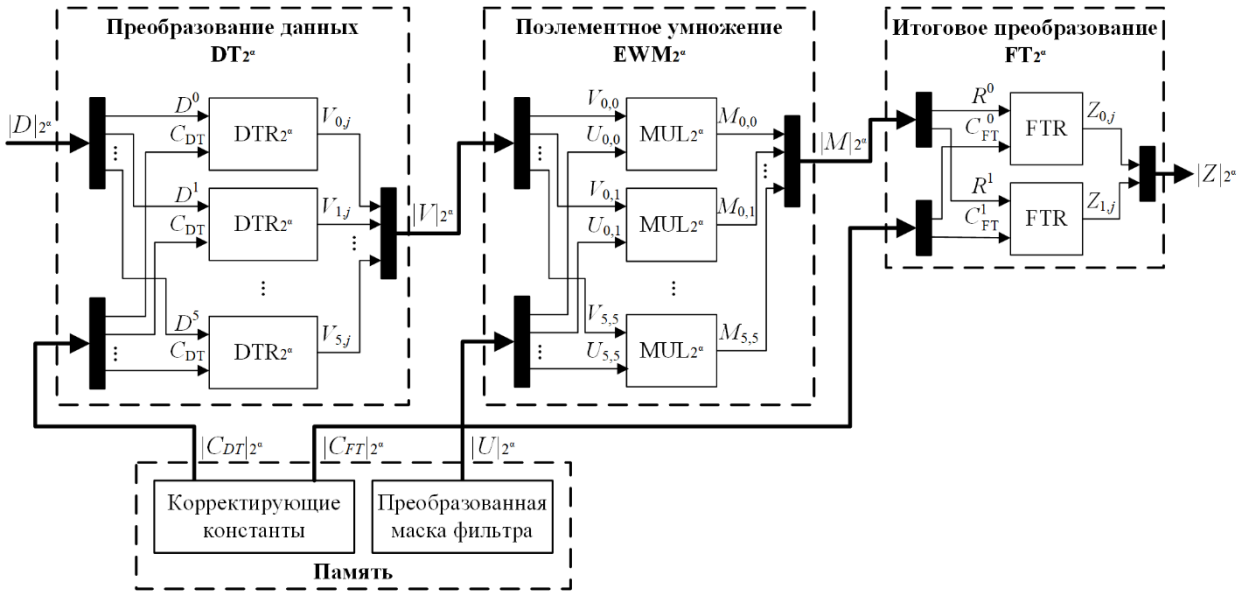


Рисунок 2.6.8. Архитектура устройства двумерной фильтрации $\mathrm{F}(2\times 2,5\times 5)_{2^\alpha}$ по методу Винограда с вычислениями по модулю 2^α

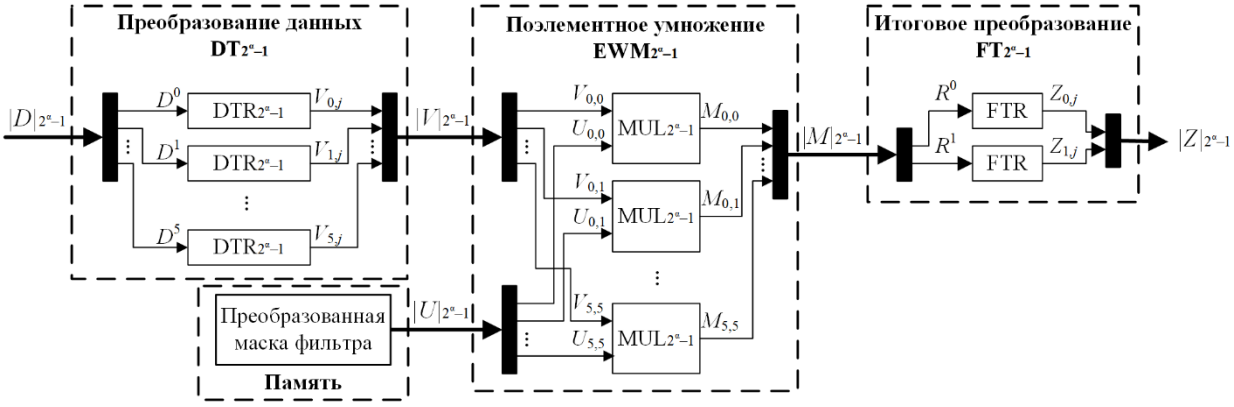


Рисунок 2.6.9. Архитектура устройства двумерной фильтрации $\mathrm{F}(2\times 2,5\times 5)_{2^\alpha-1}$ по методу Винограда с вычислениями по модулю $2^\alpha-1$

На рисунке 2.6.10 представлена архитектура предлагаемого устройства $\mathrm{F}(2\times 2,5\times 5)_{\mathrm{RNS}}$ для фильтрации по методу Винограда в СОК с набором модулей $\{2^{\alpha_1},2^{\alpha_2}-1,\dots,2^{\alpha_\eta}-1\}$. На вход устройства подается массив данных $D_{RNS}=\{|D|_{2^{\alpha_1}},|D|_{2^{\alpha_2}-1},\dots,|D|_{2^{\alpha_\eta}-1}\}$. Затем, входные данные обрабатываются параллельно устройствами $\mathrm{F}(2\times 2,5\times 5)_{2^\alpha}$ и $\mathrm{F}(2\times 2,5\times 5)_{2^\alpha-1}$. На выходе устройства $\mathrm{F}(2\times 2,5\times 5)_{\mathrm{RNS}}$ формируется массив обработанных данных $Z_{RNS}=\{|Z|_{2^{\alpha_1}},|Z|_{2^{\alpha_2}-1},\dots,|Z|_{2^{\alpha_\eta}-1}\}$.

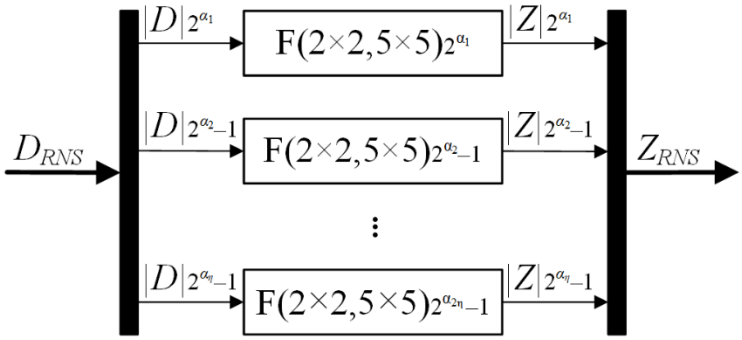


Рисунок 2.6.10. Архитектура устройства $\mathrm{F}(2\times 2,5\times 5)_{\mathrm{RNS}}$ для двумерной фильтрации по методу Винограда в СОК с модулями вида 2^{α} и $2^{\alpha}-1$

Для теоретической оценки задержки и площади цифрового устройства использовалась «unit-gate» модель [113]. Предлагаемые фильтры на основе метода Винограда $F(2\times 2,5\times 5)$ и СОК с модулями специального вида состоят из устройств DTR, MUL и FTR, описанных выше.

Устройство DTR состоит из блоков DTE и MOMA. Параметры устройства $\mathrm{DTE}_{2^{\alpha}}$ (рис. 2.6.3а) вычисляются по формуле

$$\begin{gathered} U_{\mathrm{delay}}(\mathrm{DTE}_{2^{\alpha}}) = 6{,}8\log_2(l+1), \\ U_{\mathrm{area}}(\mathrm{DTE}_{2^{\alpha}}) = 7\alpha l - 7\alpha, \end{gathered} \tag{2.6.9}$$

а устройства $\mathrm{DTE}_{2^{\alpha}-1}$ (рис. 2.6.3б) по формуле

$$\begin{gathered} U_{\mathrm{delay}}(\mathrm{DTE}_{2^{\alpha}-1}) = 6{,}8\log_2 l, \\ U_{\mathrm{area}}(\mathrm{DTE}_{2^{\alpha}-1}) = 7\alpha l - 14\alpha. \end{gathered} \tag{2.6.10}$$

Параметры устройства $\mathrm{DTR}_{2^{\alpha}}$ (рис. 2.6.4) рассчитываются по формулам (при $N=4$, $l=5$)

$$\begin{gathered} U_{\mathrm{delay}}(\mathrm{DTR}_{2^{\alpha}}) = 2U_{\mathrm{delay}}(\mathrm{DTE}_{2^{\alpha}}) + U_{\mathrm{delay}}(\mathrm{MOMA}_{2^{\alpha}}) = \\ = 2\log_2\alpha + 58{,}4, \\ U_{\mathrm{area}}(\mathrm{DTR}_{2^{\alpha}}) = 18U_{\mathrm{area}}(\mathrm{DTE}_{2^{\alpha}}) + 6U_{\mathrm{area}}(\mathrm{MOMA}_{2^{\alpha}}) = \\ = 18\alpha\log_2\alpha + 606\alpha + 6, \end{gathered} \tag{2.6.11}$$

а устройства $\mathrm{DTR}_{2^{\alpha}-1}$ по формулам (при $N=4$, $l=4$)

$$\begin{gathered} U_{\mathrm{delay}}(\mathrm{DTR}_{2^{\alpha}-1}) = 2U_{\mathrm{delay}}(\mathrm{DTE}_{2^{\alpha}-1}) + U_{\mathrm{delay}}(\mathrm{MOMA}_{2^{\alpha}-1}) = \\ = 2\log_2\alpha + 44{,}8, \\ U_{\mathrm{area}}(\mathrm{DTR}_{2^{\alpha}-1}) = 18U_{\mathrm{area}}(\mathrm{DTE}_{2^{\alpha}-1}) + 6U_{\mathrm{area}}(\mathrm{MOMA}_{2^{\alpha}-1}) = \end{gathered} \tag{2.6.12}$$

$$= 18\alpha \log_2 \alpha + 372\alpha.$$

Параметры устройства MUL для умножения чисел вычисляются по формулам (2.4.26) и (2.4.27). Параметры устройства FTR по модулю 2^α

$$\begin{gathered} U_{\text{delay}}(\text{FTR}_{2^\alpha}) = 2U_{\text{delay}}(\text{DTE}_{2^\alpha}) + U_{\text{delay}}(\text{MOMA}_{2^\alpha}) = \\ = 2\log_2\alpha + 58{,}4, \\ U_{\text{area}}(\text{FTR}_{2^\alpha}) = 10U_{\text{area}}(\text{DTE}_{2^\alpha}) + 2U_{\text{area}}(\text{MOMA}_{2^\alpha}) = \\ = 6\alpha \log_2 \alpha + 384\alpha + 2, \end{gathered} \tag{2.6.13}$$

а по модулю $2^\alpha - 1$

$$\begin{gathered} U_{\text{delay}}(\text{FTR}_{2^\alpha-1}) = 2U_{\text{delay}}(\text{DTE}_{2^\alpha-1}) + U_{\text{delay}}(\text{MOMA}_{2^\alpha-1}) = \\ = 2\log_2\alpha + 58{,}4, \\ U_{\text{area}}(\text{FTR}_{2^\alpha-1}) = 10U_{\text{area}}(\text{DTE}_{2^\alpha-1}) + 2U_{\text{area}}(\text{MOMA}_{2^\alpha-1}) \\ = 6\alpha \log_2 \alpha + 250\alpha. \end{gathered} \tag{2.6.14}$$

Предлагаемые устройства фильтрации на основе метода Винограда $\text{F}(2 \times 2, 5 \times 5)$ состоят из 6 устройств DTR, 36 устройств MUL и 2 устройств FTR. Таким образом, параметры предлагаемого устройства по модулю 2^α, основываясь на (2.6.11), (2.4.26) и (2.6.13), равны

$$\begin{gathered} U_{\text{delay}}(\text{F}(2 \times 2, 5 \times 5)_{2^\alpha}) = \\ = U_{\text{delay}}(\text{DTR}_{2^\alpha}) + U_{\text{delay}}(\text{MUL}_{2^\alpha}) + U_{\text{delay}}(\text{FTR}_{2^\alpha}) = \\ = 12{,}8\log_2\alpha + 0{,}5\alpha^2 + 0{,}5\alpha + 120{,}8, \\ U_{\text{area}}(\text{F}(2 \times 2, 5 \times 5)_{2^\alpha}) = \\ = 6U_{\text{area}}(\text{DTR}_{2^\alpha}) + 36U_{\text{area}}(\text{MUL}_{2^\alpha}) + 2U_{\text{area}}(\text{FTR}_{2^\alpha}) = \\ = 228\alpha \log_2 \alpha + 270\alpha^2 + 4026\alpha + 76, \end{gathered} \tag{2.6.15}$$

а по модулю $2^\alpha - 1$, основываясь на (2.6.12), (2.4.27) и (2.6.14), равны

$$\begin{gathered} U_{\text{delay}}(\text{F}(2 \times 2, 5 \times 5)_{2^\alpha-1}) = \\ = U_{\text{delay}}(\text{DTR}_{2^\alpha-1}) + U_{\text{delay}}(\text{MUL}_{2^\alpha-1}) + U_{\text{delay}}(\text{FTR}_{2^\alpha-1}) = \\ = 12{,}8\log_2\alpha + \alpha^2 + 107{,}2, \\ U_{\text{area}}(\text{F}(2 \times 2, 5 \times 5)_{2^\alpha-1}) = \\ = 6U_{\text{area}}(\text{DTR}_{2^\alpha-1}) + 36U_{\text{area}}(\text{MUL}_{2^\alpha-1}) + 2U_{\text{area}}(\text{FTR}_{2^\alpha-1}) = \\ = 228\alpha \log_2 \alpha + 288\alpha^2 + 2444\alpha. \end{gathered} \tag{2.6.16}$$

Параметры предлагаемого устройства двумерной фильтрации $\mathrm{F}(2\times 2,5\times 5)_{\mathrm{RNS}}$ с вычислениями в СОК с набором модулей $\{2^{\alpha_1}, 2^{\alpha_2}-1,\dots,2^{\alpha_\eta}-1\}$ (рис. 2.6.10), принимая во внимание (2.6.15) и (2.6.16), вычисляются как

$$U_{\mathrm{delay}}(\mathrm{F}(2\times 2,5\times 5)_{RNS}) = \max\big(U_{\mathrm{delay}}(\mathrm{F}(2\times 2,5\times 5)_{2^{\alpha_1}}),$$
$$\big\{U_{\mathrm{delay}}(\mathrm{F}(2\times 2,5\times 5)_{2^{\alpha_i-1}})\big\}|2\le i\le \eta\big),$$
$$U_{\mathrm{area}}(\mathrm{F}(2\times 2,5\times 5)_{RNS}) = U_{\mathrm{area}}(\mathrm{F}(2\times 2,5\times 5)_{2^{\alpha_1}}) + \qquad (2.6.17)$$
$$+\sum_{i=2}^{\eta} U_{\mathrm{area}}(\mathrm{F}(2\times 2,5\times 5)_{2^{\alpha_i-1}}).$$

Как и для случаев $F(2\times 2,2\times 2)$ и $F(2\times 2,3\times 3)$, было проведено сравнение предлагаемого устройства фильтрации изображений с устройством для фильтрации по методу Винограда в ПСС [111]. Кроме того, проведено сравнение с устройством фильтрации, состоящим из блоков MAC [101], и устройством, состоящим из блоков TMAC в ПСС [6] и с использованием СОК [7]. Для реализации вычислений в СОК использовались наборы модулей специального вида, представленные в таблице 2.4.1.

В таблице 2.6.1 представлены результаты теоретической оценки параметров площади и задержки предложенного и известных фильтров на основе «unit-gate» модели. Так же была произведена оценка времени обработки изображения размером 256×256.

Теоретический анализ параметров предлагаемого и известных двумерных фильтров с маской размерности 5×5 показал, что использование предлагаемого подхода на основе метода Винограда и СОК с модулями специального вида сокращает задержку устройства в 1,18-5,34 раза, время обработки изображения также уменьшается в 1,18-21,35 раза по сравнению с известными подходами. Кроме того, устройство на основе предложенного метода имеет на 14,4-48,33% меньшую площадь по сравнению с устройством на основе метода Винограда [111]. Однако, следует отметить, что применение

предлагаемого подхода увеличивает площадь устройства в 1,03-5,37 раза по сравнению с другими рассмотренными известными методами.

Таблица 2.6.1. Теоретические параметры двумерных фильтров с маской размерности 5×5

Параметр	Разрядность фильтра, бит	Известные методы				Предлагаемый метод
		[101]	[6]	[7]	[111]	
Задержка	8	817	567	446	196	**166**
	16	1046	746	567	308	**196**
	32	1274	924	671	713	**273**
Площадь	8	14378	13409	**8006**	55036	47108
	16	56602	53489	**22050**	148204	93311
	32	222170	213569	**76428**	441868	228300
Время обработки	8	53542912	37158912	29229056	3211264	**2719744**
	16	68550656	48889856	38465280	5046272	**3211264**
	32	83492864	60555264	43974656	11681792	**4472832**

Аппаратное моделирование на FPGA проводилось в среде автоматизированного проектирования Xilinx Vivado 2018.3 yf zpsrt jgbcfybz fggfhfnehs VHDL для целевой платы Virtex UltraScale xcvu440-flgb2377-3-e со стратегией оптимизации Flow_PerfOptimized_high. Результаты аппаратного моделирования устройств для фильтрации изображений представлены в таблице 2.6.2. Для оценки устройств использовались такие параметры как тактовая частота, количество LUT, энергопотребление и производительность, которые были получены в результате симуляции в среде проектирования. Под производительностью устройства подразумевается количество обработанных кадров размером 256×256 пикселей в секунду.

Результаты аппаратного моделирования показали, что предлагаемый метод построения фильтров на основе метода Винограда и СОК позволяет увеличить тактовую частоту 16- и 32-разрядных устройств на 38,24% и 29,63% соответственно по сравнению с фильтром на основе метода Винограда [111] без использования арифметики СОК. Однако, у 8-разрядных устройств тактовая частота фильтра на основе предлагаемого метода на 3,23% меньше. Кроме того, предлагаемый метод увеличивает тактовую частоту устройства на 7,14-105,88% по сравнению с методами на основе КИХ фильтров с блоками

MAC и TMAC [101, 6, 7] за исключением 8-разрядного устройства из работы [7], имеющего такую же тактовую частоту как и устройство на основе предлагаемого метода.

Таблица 2.6.2. Результаты аппаратного моделирования двумерных фильтров с маской размерности 5×5

Параметр	Разрядность фильтра, бит	Известные методы				Предлагаемый метод
		[101]	[6]	[7]	[111]	
Тактовая частота, МГц	8	49	56	60	**62**	60
	16	29	37	38	34	**47**
	32	17	26	30	27	**35**
Количество LUT	8	510	589	**472**	5745	5199
	16	2916	2417	**1278**	18144	13456
	32	18059	13872	**6044**	63927	45920
Энергопотребление, Вт	8	2,936	2,930	**2,928**	3,056	3,041
	16	**2,948**	2,951	2,949	3,168	3,160
	32	2,995	3,012	**2,977**	3,744	3,589
Производительность, кадр/с	8	747	854	915	**3784**	3662
	16	442	564	579	2075	**2868**
	32	259	396	457	1647	**2136**

Совместное применение СОК и метода Винограда позволяет сократить количество занятых LUT на 9,5-28,17%, а энергопотребление на 0,4-94,14% по сравнению с методом Винограда в ПСС. При этом следует отметить, что устройства, спроектированные по предлагаемому методу, используют в 2,54-11,01 раза больше LUT и имеют на 3,58-20,56% выше энергопотребление, по сравнению с устройствами на основе методов [101, 6, 7].

Аппаратное моделирование показало, что использование предлагаемого метода на основе метода Винограда и СОК повышает производительность фильтра в 1,3-8,25 раза по сравнению с фильтрами на основе рассмотренных известных методов. Исключение составляет 8-разрядное устройство на основе метода Винограда и ПСС, производительность которого на 3,22% выше по сравнению с устройством на основе предлагаемого метода.

2.7 Выводы по второй главе

Во второй главе предложен метод проектирования устройства, реализующего операцию свертки на основе метода Винограда и СОК с модулями специального вида. Рассмотрены случаи масок фильтров размерности 2×2, 3×3 и 5×5, для которых разработаны цифровые фильтры двумерной фильтрации на основе предложенного метода. Произведено сравнение предложенного метода с известными аналогами. Результаты теоретического анализа на основе «unit-gate»-модели показали, что время обработки сигнала устройством, спроектированным по предлагаемому методу, меньше, чем у аналогов, но площадь устройства больше. Результаты аппаратного моделирования показали, что предлагаемый метод позволяет увеличить производительность устройств по сравнению с рассмотренными аналогами за счет увеличения аппаратных затрат. Незначительное отличие между результатами теоретического анализа и результатами аппаратного моделирования объясняется особенностью «unit-gate»-модели, которая на учитывает нагрузочную способность выхода устройств, а также задействованную память и время обращения к ней.

Высокая производительность фильтров с вычислениями по методу Винограда объясняется тем, что на выходе формируются сразу несколько обрабатываемых элементов (рис. 2.4.1, 2.5.1, 2.6.1). Прирост производительности достигается за счет увеличения аппаратных затрат, таких как количество занятых LUT и энергопотребление. Таким образом, предложенные архитектуры фильтров могут быть успешно применены в системах цифровой обработки сигналов, в которых производительность является ключевым критерием, но с точки зрения занимаемой площади архитектура на основе TMAC блоков с вычислениями в СОК [7] предпочтительнее, а фильтры на основе MAC блоков [101] целесообразно использовать в системах с малым энергопотреблением.

Предложенный метод проектирования устройства, реализующего операцию свертки на основе метода Винограда и СОК с модулями специального вида может быть успешно применен для реализации операции свертки в СНС.

3 ВЫЧИСЛИТЕЛЬНО СЛОЖНЫЕ ОПЕРАЦИИ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ

3.1 Метод преобразования чисел из позиционной системы счисления в систему остаточных классов с модулями специального вида

Первым блоком при выполнении вычислений в СОК является перевод чисел в СОК. Для преобразования числа из ПСС в СОК необходимо вычислить остатки от деления по каждому модулю. Данная операция является ресурсозатратной. Использование модулей вида $\{2^{\alpha_1}, 2^{\alpha_2}-1, \dots, 2^{\alpha_n}-1\}$ позволяет ее избежать. Операция вычисления остатка от деления по модулю вида 2^{α} осуществляется простым взятием α младших бит исходного числа. Введем для устройства, реализующего вычисление остатка от деления по модулю 2^{α} обозначение $\mathrm{MOD}_{2^{\alpha}}$. Для модулей вида $2^{\alpha}-1$ вычисление остатка от деления происходит следующим образом. Пусть $A=\sum_{i=0}^{g-1} A_i 2^i$ – исходное g-битное число, представленное в ПСС битами A_i, $0 \le i < g$. Разобьем его на $s=\left\lceil \frac{g}{\alpha} \right\rceil$ частей по α бит. Для этого, при необходимости, дополним A старшими значащими битами, равными 0, до размера $g' = s\alpha$ бит. Таким образом, число A' в двоичной ПСС представляется как $A'=\sum_{i=0}^{g'-1} A_i 2^i$, а его α-битные части имеют вид $Y_0=\sum_{i=0}^{\alpha-1} A_i 2^i$, $Y_1=\sum_{i=\alpha}^{2\alpha-1} A_i 2^i$, …, $Y_{s-1}=\sum_{i=(s-1)\alpha}^{g'-1} A_i 2^i$. С использованием введенных обозначений Y_0, Y_1, …, Y_{s-1}, число A' может быть записано как $A' = Y_0 + Y_1 \cdot 2^{\alpha} + Y_2 \cdot 2^{2\alpha} + \dots + Y_{s-1} \cdot 2^{(s-1)\alpha}$. Так как $\left|Y_i \cdot 2^{i\alpha}\right|_{2^{\alpha}-1} = |Y_i|_{2^{\alpha}-1}$, $0 \le i < s$, то

$$|A'|_{2^{\alpha}-1} = |Y_0 + Y_1 + Y_2 + \dots + Y_{s-1}|_{2^{\alpha}-1}. \qquad (3.1.1)$$

Формула (3.1.1) показывает, что вычисление остатка от деления по модулю $2^{\alpha}-1$ сводится к сложению α-битных чисел по модулю $2^{\alpha}-1$. Для сложения по модулю $2^{\alpha}-1$ больше двух слагаемых используют устройство $\mathrm{MOMA}_{2^{\alpha}-1}$ (рис. 2.4.3б), которое состоит из дерева CSA с циклическим

переносом старшего бита, а результат передается на KSA с циклическим переносом старшего бита. Архитектура g-битного устройства вычисления остатка от деления по модулю $2^{\alpha}-1$ представлена на рисунке 3.1.1. Введем для него обозначение $\mathrm{MOD}_{2^{\alpha}-1}$.

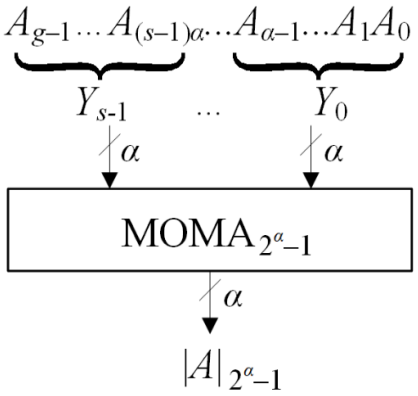


Рисунок 3.1.1. Архитектура устройства $\mathrm{MOD}_{2^{\alpha}-1}$ для вычисления остатка от деления по модулю $2^{\alpha}-1$

Проведем теоретический анализ технических параметров устройства $\mathrm{MOD}_{2^{\alpha}-1}$, используя «unit-gate»-модель. Тогда, основываясь на формуле (2.4.20), параметры задержки и площади устройства вычисляются по формулам

$$U_{\mathrm{delay}}(\mathrm{MOD}_{2^{\alpha}-1}) = 6{,}8\log_2 s + 2\log_2\alpha + 4,$$
$$U_{\mathrm{area}}(\mathrm{MOD}_{2^{\alpha}-1}) = 3\alpha\log_2\alpha + 7\alpha s - 8\alpha. \quad (3.1.2)$$

Параметры площади и задержки устройства MOD для вычисления остатка от деления g-битного числа на α-битный модуль, построенного с помощью функции mod из библиотеки IEEE.numeric_bit, вычисляются по формулам

$$U_{\mathrm{delay}}(\mathrm{MOD}) = (g-\alpha+1)(4\log_2\alpha + 18),$$
$$U_{\mathrm{area}}(\mathrm{MOD}) = 6\alpha\log_2\alpha + 23\alpha + 2. \quad (3.1.3)$$

Параметры площади и задержки устройств для вычисления остатка от деления 16-битного числа по модулю $2^{\alpha}-1$ представлены в таблице 3.1.1. Результаты теоретического анализа на основе «unit-gate»-модели показали,

что задержка предлагаемого устройства $\mathrm{MOD}_{2^\alpha-1}$ в 13,64-15,88 раза меньше по сравнению с устройством MOD. Кроме того, площадь предлагаемого устройства $\mathrm{MOD}_{2^\alpha-1}$ в 1,36-2,75 раза меньше по сравнению с известным устройством MOD. Тем не менее, при $\alpha = 3$ площадь предлагаемого устройства на 17% больше.

Таблица 3.1.1. Теоретические параметры устройств для вычисления остатка от деления 16-битного числа по модулю $2^\alpha - 1$

Параметр	α	$\mathrm{MOD}_{2^\alpha-1}$	MOD
Задержка	3	25	341
	4	22	338
	5	23	328
	6	20	312
	7	21	293
	8	17	270
Площадь	3	117	100
	4	104	142
	5	135	187
	6	125	234
	7	150	281
	8	120	330

Моделирование производилось в среде ISE Design Suite 14.7. Целевая плата – Kintex 7 XC7K70T. Целью моделирования было сравнение работы схемы вычисления остатка от деления 16-битного числа по модулю $2^\alpha - 1$ и встроенной в среду функции mod из библиотеки IEEE.numeric_bit. Результаты моделирования представлены в таблице 3.1.2.

Результаты аппаратного моделирования показали, что при использовании предлагаемого метода скорость в среднем в 3,11 раза выше, чем при использовании встроенной функции mod. Аппаратные затраты для $\alpha = \overline{3,5}$ сократились в среднем в 20,08 раза, для $\alpha = \overline{6,8}$ – в среднем в 9,9 раза.

Таблица 3.1.2. Результаты аппаратного моделирования устройств для вычисления остатка от деления 16-битного числа по модулю $2^{\alpha}-1$

Параметр	α	Предлагаемый метод	Встроенная функция mod
Задержка, нс	3	6,924	24,459
	4	6,586	22,879
	5	6,856	20,900
	6	6,854	20,657
	7	7,226	19,669
	8	6,770	19,597
Количество LUT	3	27	559
	4	31	607
	5	26	519
	6	40	434
	7	51	482
	8	44	414

3.2 Операция определения знака числа в системе остаточных классов

Любое целое число $0 \le A < P$ может быть однозначно представлено в СОК как $A = \{a_1, a_2, \dots, a_n\}$. Для представления отрицательных чисел в СОК динамический диапазон системы делится на две примерно равные части, при этом должно выполняться одно из следующих условий

$$-\frac{P-1}{2} \le A \le \frac{P-1}{2} \text{ для нечетных } P,$$
$$-\frac{P}{2} \le A \le \frac{P}{2} - 1 \text{ для четных } P. \qquad (3.2.1)$$

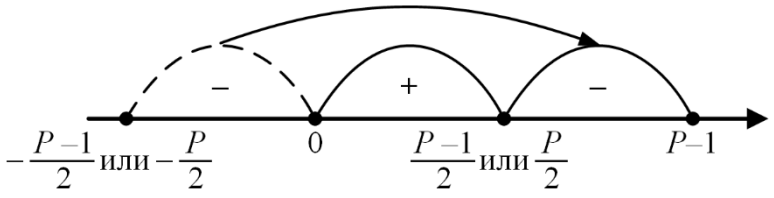


Рисунок 3.2.1. Расположение положительных и отрицательных чисел в СОК

Операция определения знака числа является ресурсозатратной в СОК, так как требует вычисления позиционной характеристики числа [9].

Вычисление позиционной характеристики числа по формуле (1.3.2) требует вычисления остатка от деления на число P разрядностью равной разрядности полного диапазона системы DR. На практике, одним из самых эффективных подходов является модификация КТО, называемая КТО с дробными величинами (КТОд) [49].

Согласно КТОд позиционную характеристику числа A' можно вычислить по формуле

$$A' = \left| \sum_{i=1}^{n} a_i \tilde{k}_i \right|_{2^N}, \tag{3.2.2}$$

где $\tilde{k}_i = \left\lfloor 2^N \frac{|P_i^{-1}|_{p_i}}{p_i} \right\rfloor$. Для гарантированного точного перевода чисел из СОК в ПСС достаточно выбрать N равным

$$N = \lceil \log_2(P\mu) \rceil, \tag{3.2.3}$$

где $\mu = -n + \sum_{i=1}^{n} p_i$.

Для определения знака числа в СОК по его позиционной характеристике A' необходимо выполнить проверку следующих условий:

1) если $0 \le A' < 2^{N-1}$, тогда число A положительное;
2) если $2^{N-1} \le A' < 2^N$, тогда число A отрицательное.

На рисунке 3.2.2 представлена схема устройства, которое вычисляет позиционную характеристику числа по формуле (3.2.2) с помощью КТОд. Введем для него обозначение $\mathrm{NPC}_{\mathrm{CRTf}}$. На вход устройства поступает число $\{a_1, a_2, \dots, a_n\}$, представленное в СОК с модулями $\{p_1, p_2, \dots, p_n\}$ и разрядностями $\{\beta_1, \beta_2, \dots, \beta_n\}$ по каждому модулю соответственно. Также на вход подаются коэффициенты $\tilde{k}_i$, имеющие разрядность N бит, они являются константами и могут быть вычислены предварительно. Генерация частичных произведений осуществляется с помощью устройств PPG_{2^α}. Далее N-битные частичные произведения складываются с помощью устройства MOMA_{2^α}. Устройства PPG_{2^α} и MOMA_{2^α} являются N-битными, то есть $\alpha = N$. Если

старший значащий бит (Most Significant Bit, MSB) числа A' равен 1, то число A отрицательное, если равен 0, то положительное.

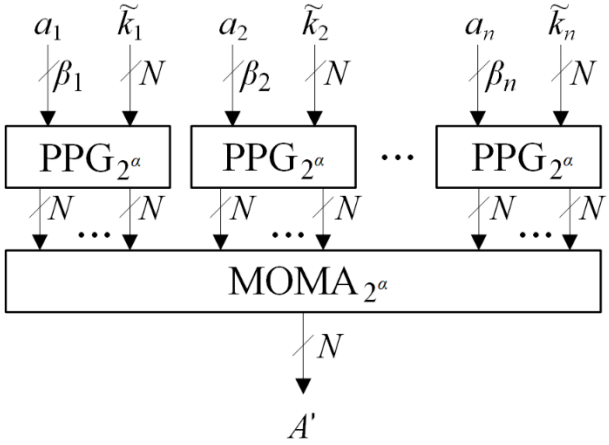


Рисунок 3.2.2. Архитектура устройства для вычисления позиционной характеристики числа, представленного в СОК, с помощью КТОд

Учитывая технические параметры устройств PPG_{2^α} (2.4.24) и MOMA_{2^α} (2.4.19), параметры площади и задержки устройства $\mathrm{NPC}_{\mathrm{CRTf}}$ могут быть вычислены по формулам

$$U_{\mathrm{delay}}(\mathrm{NPC}_{\mathrm{CRTf}}) = 6{,}8\log_2 nN + 2\log_2 N + 0{,}5N^2 + 0{,}5N + 4,$$
$$U_{\mathrm{area}}(\mathrm{NPC}_{\mathrm{CRTf}}) = 3N\log_2 N + 7{,}5nN^2 + 0{,}5nN - 11N + 1. \quad (3.2.4)$$

Таким образом, технические параметры устройства $\mathrm{SIGN}_{\mathrm{CRTf}}$ определения знака числа в СОК с помощью КТОд равны параметрам устройства $\mathrm{NPC}_{\mathrm{CRTf}}$, то есть $U_{\mathrm{delay}}(\mathrm{SIGN}_{\mathrm{CRTf}}) = U_{\mathrm{delay}}(\mathrm{NPC}_{\mathrm{CRTf}})$ и $U_{\mathrm{area}}(\mathrm{SIGN}_{\mathrm{CRTf}}) = U_{\mathrm{area}}(\mathrm{NPC}_{\mathrm{CRTf}})$.

Проведем сравнение технических параметров устройств определения знака числа в СОК на основе известных методов КТО [88] и ОПСС [44] с параметрами предлагаемого устройства на основе КТОд. Метод определения знака числа в СОК на основе КТО требует перевода числа из СОК в ПСС по формуле (1.3.2), обозначим данное устройство как $\mathrm{RC}_{\mathrm{CRT}}$. Умножение на константы производится с помощью n $(DR + \beta)$-разрядных генераторов частичных произведений PPG ($\beta = \max\limits_{1 \le i \le n}\{\beta_i\}$) и сумматора со множественных

входом MOA разрядности $\big((n+1)\cdot(DR+\beta)-1\big)$ бит, на вход которого подаются $n\cdot(DR+\beta)$ частичных произведений. Далее вычисляется остаток от деления по модулю динамического диапазона СОК P с помощью DR-разрядного устройства MOD. Параметры устройства $\mathrm{RC_{CRT}}$ [88] могут быть рассчитаны по формулам

$$\begin{gathered}U_{\mathrm{delay}}(\mathrm{RC_{CRT}})=\\ =6{,}8\log_2 n(DR+\beta)+2\log_2\big((n+1)(DR+\beta)-1\big)+\\ +(n\cdot DR+\beta+n\beta)(4\log_2 DR+18)+0{,}5(DR+\beta)^2+\\ +0{,}5(DR+\beta)+4,\\ U_{\mathrm{area}}(\mathrm{RC_{CRT}})=\\ =3\big((n+1)(DR+\beta)-1\big)\log_2\big((n+1)(DR+\beta)-1\big)+\\ +6DR\log_2 DR+7n\big((n+1)(DR+\beta)-1\big)(DR+\beta)-\\ -11\big((n+1)(DR+\beta)-1\big)+0{,}5n(DR+\beta)^2+0{,}5n(DR+\beta)+\\ +23DR+3.\end{gathered}\tag{3.2.5}$$

Далее необходимо определить к какому интервалу относится число в ПСС (рис. 3.2.1), то есть выполнить операцию сравнения. Для данной цели необходимо использовать устройство сравнения чисел (компаратор). На рисунке 3.2.3 представлено α-разрядное устройство сравнения двух чисел A и B в ПСС. Введем для него обозначение COMP. Процесс сравнения сводится к вычислению разности между числами A и B. Для этого число B представляется в дополнительном коде, затем производится сложение с помощью α-разрядных сумматоров CSA и KSA. Если старший значащий бит на выходе устройства равен 0, то $A\geq B$, если равен 1, то $A<B$. Параметры площади и задержки данного устройства вычисляются следующим образом

$$\begin{gathered}U_{\mathrm{delay}}(\mathrm{COMP})=2\log_2\alpha+8,\\ U_{\mathrm{area}}(\mathrm{COMP})=3\alpha\log_2\alpha+10\alpha+1.\end{gathered}\tag{3.2.6}$$

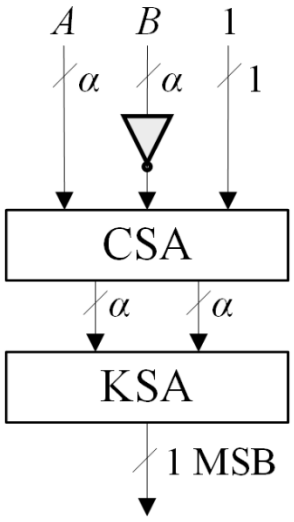


Рисунок 3.2.3. Архитектура компаратора COMP

Для определения к какому интервалу относится число A (рис. 3.2.1) на вход DR-разрядного компаратора COMP подается число A и $\frac{P-1}{2}$ или $\frac{P}{2}$. Если на выход компаратора поступает 1, то число A положительное, если 0, то число A – отрицательное.

Таким образом, технические параметры устройства $\mathrm{SIGN_{CRT}}$ определения знака числа в СОК с помощью КТО складываются из параметров устройства $\mathrm{RC_{CRT}}$ и DR-разрядного устройства COMP, то есть $U_{\mathrm{delay}}(\mathrm{SIGN_{CRT}}) = U_{\mathrm{delay}}(\mathrm{RC_{CRT}}) + U_{\mathrm{delay}}(\mathrm{COMP})$ и $U_{\mathrm{area}}(\mathrm{SIGN_{CRT}}) = = U_{\mathrm{area}}(\mathrm{RC_{CRT}}) + U_{\mathrm{area}}(\mathrm{COMP})$. Следовательно, основываясь на (3.2.5) и (3.2.6), параметры задержки и площади устройства $\mathrm{SIGN_{CRT}}$ могут быть вычислены следующим образом

$$U_{\mathrm{delay}}(\mathrm{SIGN_{CRT}}) =$$
$$= 6{,}8\log_2 n(DR+\beta) + 2\log_2\big((n+1)(DR+\beta)-1\big) +$$
$$+(n\cdot DR+\beta+n\beta)(4\log_2 DR+18) + 2\log_2 DR +$$
$$+0{,}5(DR+\beta)^2 + 0{,}5(DR+\beta) + 12, \qquad (3.2.7)$$
$$U_{\mathrm{area}}(\mathrm{SIGN_{CRT}}) =$$
$$= 3\big((n+1)(DR+\beta)-1\big)\log_2\big((n+1)(DR+\beta)-1\big) +$$
$$+9DR\log_2 DR + 7n\big((n+1)(DR+\beta)-1\big)(DR+\beta) -$$

$$-11\big((n+1)(DR+\beta)-1\big)+0{,}5n(DR+\beta)^2+0{,}5n(DR+\beta)+$$
$$+33DR+4.$$

Рассмотрим метод определения знака числа, основанный на его представлении в ОПСС [44]. Умножение по каждому вычислительному каналу p_i, $1 \le i \le n$, производится с помощью генератора частичных произведений PPG и сумматора со множественным входом MOA с разрядностью $\max\limits_{j=1\ldots i}\{\log_2 p_j\}+\log_2 p_i+i$. Допустим, что вычисления по каждому модулю производятся с разрядностью $2\beta+n$, а операция деления по модулю p_i сводится к делению на β-битное число с помощью устройства MOD. Тогда устройство вычисления позиционной характеристики числа с помощью перевода в ОПСС $\mathrm{NPC_{MRC}}$ имеет следующие параметры задержки и площади

$$\begin{gathered}U_{\mathrm{delay}}(\mathrm{NPC_{MRC}})=6{,}8n\log_2(n\beta)+2n\log_2(2\beta+n)+\\+n(\beta+n+1)(4\log_2\beta+18)+2n\beta^2+n\beta+10{,}8n,\\U_{\mathrm{area}}(\mathrm{NPC_{MRC}})=3n(2\beta+n)\log_2(2\beta+n)+6n\beta\log_2\beta+\\+30n^2\beta^2+14n^3\beta+n^2\beta+n\beta-11n^2+3n.\end{gathered}\qquad(3.2.8)$$

Определение к какому интервалу относится число, представленное в ОПСС, производится аналогично методу на основе КТО с помощью компаратора COMP. Для выполнения операции сравнения в ОПСС производится конкатенация чисел по каждому модулю, поэтому на компаратор поступает $\sum_{i=1}^{n}\beta_i$-битное число. Будем считать, что $\sum_{i=1}^{n}\beta_i \approx DR$. Следовательно, основываясь на (3.2.6) и (3.2.8), параметры задержки и площади устройства $\mathrm{SIGN_{MRC}}$ определения знака на основе ОПСС могут быть вычислены следующим образом

$$\begin{gathered}U_{\mathrm{delay}}(\mathrm{SIGN_{MRC}})=6{,}8n\log_2(n\beta)+2n\log_2(2\beta+n)+\\+n(\beta+n+1)(4\log_2\beta+18)+2\log_2 DR+2n\beta^2+\\+n\beta+10{,}8n+8,\\U_{\mathrm{area}}(\mathrm{SIGN_{MRC}})=3n(2\beta+n)\log_2(2\beta+n)+6n\beta\log_2\beta+\\+30n^2\beta^2+14n^3\beta+n^2\beta+n\beta-11n^2+3n+\end{gathered}\qquad(3.2.9)$$

$$+3DR\log_2 DR + 10DR + 1.$$

Таблица 3.2.1. Теоретические параметры площади и задержки устройств определения знака числа в СОК

Параметр	Набор модулей	КТОд	КТО	ОПСС
Задержка	{5, 7, 11}	**125**	1476	873
	{41, 43, 47}	**331**	2927	1242
	{1621, 1627, 1637}	**1053**	6535	2373
	{2642239, 2642257, 2642287}	**3813**	15835	5975
	{6981463658333, 6981463658341, 6981463658353}	**14959**	40724	17058
	{3, 5, 7, 11}	**157**	2230	1276
	{13, 11, 19, 23}	**288**	3334	1519
	{251, 257, 263, 269}	**969**	7749	2640
	{65537, 65539, 65543, 65551}	**3477**	18023	5674
	{4294967279, 4294967291, 4294967311, 4294967357}	**13286**	44669	14953
	{2, 3, 5, 7, 11}	**175**	2928	1734
	{5, 7, 11, 13, 17}	**290**	4284	2043
	{79, 83, 89, 97, 101}	**928**	8813	2714
	{7121, 7127, 7129, 7151, 7159}	**3236**	20144	5200
	{50858993, 50858999, 50859013, 50859037, 50859041}	**12174**	49083	13152
Площадь	{5, 7, 11}	**3257**	15061	6454
	{41, 43, 47}	**11998**	46748	13157
	{1621, 1627, 1637}	43864	169249	**39431**
	{2642239, 2642257, 2642287}	167252	656093	**145111**
	{6981463658333, 6981463658341, 6981463658353}	667839	2550346	**529302**
	{3, 5, 7, 11}	**5915**	32834	12114
	{13, 11, 19, 23}	**13319**	64131	17688
	{251, 257, 263, 269}	53223	254521	**49653**
	{65537, 65539, 65543, 65551}	202547	964495	**160015**
	{4294967279, 4294967291, 4294967311, 4294967357}	789431	3749383	**565933**
	{2, 3, 5, 7, 11}	**8487**	55495	20072
	{5, 7, 11, 13, 17}	**16637**	104649	28997
	{79, 83, 89, 97, 101}	63349	344328	**51581**
	{7121, 7127, 7129, 7151, 7159}	234861	1303274	**155203**
	{50858993, 50858999, 50859013, 50859037, 50859041}	903005	5129936	**565678**

В таблице 3.2.1 представлены параметры площади и задержки устройств определения знака числа в СОК разными методами. Результаты теоретического анализа параметров устройств показали, что задержка устройства на основе КТОд в 3,34-16,73 раза меньше по сравнению с

устройством на основе КТО и в 1,08-8,13 раза меньше по сравнению с устройством на основе ОПСС. Более того, метод на основе КТОд позволяет сократить площадь устройства в 3,82-6,54 раза. При использовании СОК с небольшим динамическим диапазоном, площадь устройств на основе КТОд на 8,81-57,72% меньше по сравнению с площадью устройств на основе ОПСС. Тем не менее, при увеличении динамического диапазона площадь устройств на основе КТОд становится на 7,19-59,63% больше по сравнению с площадью устройств на основе ОПСС

Аппаратное моделирование операции определения знака числа в СОК было произведено в среде Xilinx ISE 14.7 с использованием целевой платы xc7k70t-2fbg676 семейства Kintex-7. Результаты аппаратного моделирования представлены в таблице 3.2.2.

Моделирование показало, что метод определения знака числа на основе КТОд, работает в 2,06-14,98 раза быстрее и использует в 1,05-2,08 раза меньше Slice LUT чем метод на основе КТО. Исключение по площади здесь составляет набор {5, 7, 11, 13, 17}, для которого метод на основе КТО эффективнее метода на основе КТОд в 2,38 раза.

Метод на основе КТОд работает в 3,57-27,14 раза быстрее и использует в 1,21-1,65 раза меньше Slice LUT чем метод на основе ОПСС. Исключение по площади здесь составляют наборы {7121, 7127, 7129, 7151, 7159} и {50858993, 50858999, 50859013, 50859037, 50859041}, для которых метод на основе КТОд затратнее метода на основе ОПСС на 2,89% и 5,46% соответственно.

Одним из примеров применения операции определения знака является функция активации ReLU (1.1.2), которая широко применяется в СНС.

Таблица 3.2.2. Результаты аппаратного моделирования операции определения знака числа в СОК

Параметр	Набор модулей	КТОд	КТО	ОПСС
Задержка, нс	{5, 7, 11}	**4,183**	11,856	14,927
	{41, 43, 47}	**5,322**	20,109	32,922
	{1621, 1627, 1637}	**5,120**	32,993	70,084
	{2642239, 2642257, 2642287}	**7,231**	61,433	114,922
	{6981463658333, 6981463658341, 6981463658353}	**8,582**	128,582	219,329
	{3, 5, 7, 11}	**4,260**	8,758	24,870
	{13, 11, 19, 23}	**5,297**	18,615	35,119
	{251, 257, 263, 269}	**5,961**	31,141	69,371
	{65537, 65539, 65543, 65551}	**7,418**	50,195	146,219
	{4294967279, 4294967291, 4294967311, 4294967357}	**9,319**	100,490	252,883
	{2, 3, 5, 7, 11}	**5,191**	11,693	30,642
	{5, 7, 11, 13, 17}	**6,194**	21,121	38,348
	{79, 83, 89, 97, 101}	**6,839**	26,090	73,671
	{7121, 7127, 7129, 7151, 7159}	**7,799**	39,978	140,262
	{50858993, 50858999, 50859013, 50859037, 50859041}	**10,186**	81,283	231,630
Количество LUT	{5, 7, 11}	**70**	125	115
	{41, 43, 47}	**219**	397	295
	{1621, 1627, 1637}	**685**	1419	945
	{2642239, 2642257, 2642287}	**2530**	5253	3944
	{6981463658333, 6981463658341, 6981463658353}	**9346**	19251	15423
	{3, 5, 7, 11}	**102**	107	144
	{13, 11, 19, 23}	**202**	365	293
	{251, 257, 263, 269}	**694**	1413	1072
	{65537, 65539, 65543, 65551}	**2675**	4946	3226
	{4294967279, 4294967291, 4294967311, 4294967357}	**9782**	17307	13075
	{2, 3, 5, 7, 11}	**125**	139	188
	{5, 7, 11, 13, 17}	250	**105**	329
	{79, 83, 89, 97, 101}	**661**	1058	848
	{7121, 7127, 7129, 7151, 7159}	2887	4305	**2806**
	{50858993, 50858999, 50859013, 50859037, 50859041}	11496	17461	**10901**

3.3 Метод проектирования устройства, реализующего функцию активации ReLU на основе Китайской теоремы об остатках с дробными величинами

Функция активации ReLU требует выполнения операции определения знака числа. Формула (1.1.2) может быть представлена в виде

$$f(x) = \begin{cases} x, & \text{если } x \geq 0; \\ 0, & \text{если } x < 0. \end{cases} \quad (3.3.1)$$

Для определения знака числа, представленного в СОК, предлагается вычислять его позиционную характеристику с использованием КТОд.

Для реализации функции активации ReLU в СОК предлагается использовать архитектуру устройства, представленную на рисунке 3.3.1. На вход устройства поступает число $\{a_1, a_2, \dots, a_n\}$, представленное в СОК с модулями $\{p_1, p_2, \dots, p_n\}$ и разрядностями $\{\beta_1, \beta_2, \dots, \beta_n\}$ по каждому модулю соответственно. Знак числа определяется с помощью устройства $\mathrm{SIGN}_{\mathrm{CRTf}}$ и передается в качестве управляющего сигнала на вход мультиплексора MUX, который в соответствии с ним принимает решение какое значение подавать на выход устройства. Если число положительное, то на выход устройства подается число $\{a_1, a_2, \dots, a_n\}$, если отрицательное, то на выход подается 0.

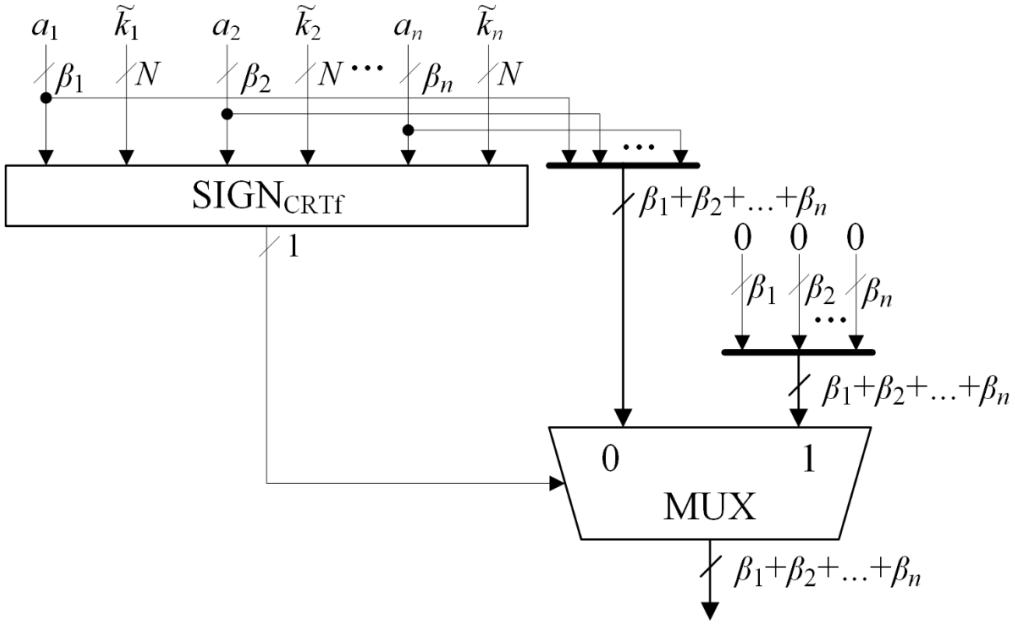


Рисунок 3.3.1. Архитектура устройства ReLU вычисления функции активации ReLU в СОК

3.4 Операция сравнения чисел в системе остаточных классов

Операция сравнения двух чисел в СОК сводится к сравнению их позиционных характеристик. Устройство COMP для сравнения двух чисел A и

B в ПСС представлено на рисунке 3.4.1. Параметры площади и задержки данного устройства вычисляются по формулам

$$U_{\text{delay}}(\text{COMP}) = 2\log_2\alpha + 0{,}5\alpha + 9,$$
$$U_{\text{area}}(\text{COMP}) = 3\alpha\log_2\alpha + 11\alpha + 1. \quad (3.4.1)$$

Таким образом, компаратор для сравнения чисел в СОК с использованием КТОд $\text{COMP}_{\text{CRTf}}$ (рис. 3.4.2) состоит из двух устройств NPC_{CRTf}, вычисляющих позиционную характеристику сравниваемых чисел, и устройства COMP (рис. 3.4.1), выполняющего сравнение двух чисел в ПСС.

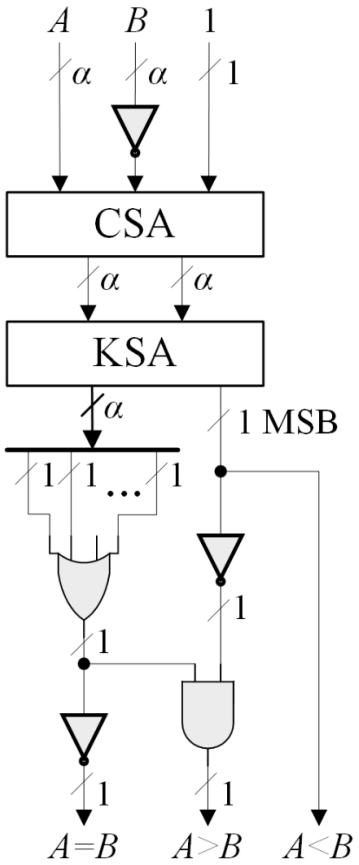


Рисунок 3.4.1. Архитектура компаратора с тремя выходами

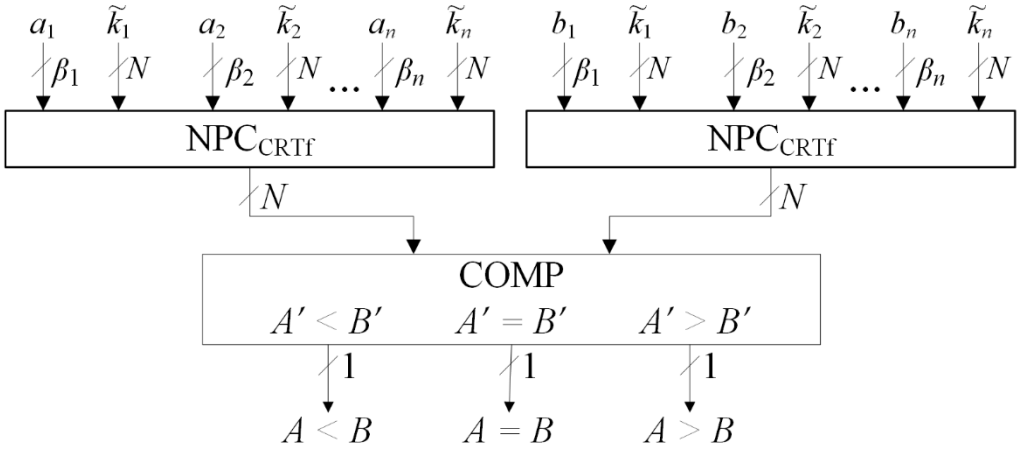


Рисунок 3.4.2. Архитектура устройства $\mathrm{COMP}_{\mathrm{CRTf}}$ для сравнения двух чисел в СОК

Поскольку при вычислении позиционной характеристики по формуле (3.2.2) производится умножение на константы, то частичные произведения могут быть вычислены заранее [5, 77], что уменьшит вычислительную сложность устройства $\mathrm{NPC}_{\mathrm{CRTf}}$. Тогда параметры площади и задержки устройства $\mathrm{NPC}_{\mathrm{CRTf}}$ будут вычисляться следующим образом

$$\begin{gathered} U_{\mathrm{delay}}(\mathrm{NPC}_{\mathrm{CRTf}}) = 6{,}8\log_2 nN + 2\log_2 N + 4, \\ U_{\mathrm{area}}(\mathrm{NPC}_{\mathrm{CRTf}}) = 3N\log_2 N + 7nN^2 - 11N + 1. \end{gathered} \tag{3.4.2}$$

Учитывая параметры устройств COMP (3.4.1) и $\mathrm{NPC}_{\mathrm{CRTf}}$ (3.4.2), параметры площади и задержки устройства $\mathrm{COMP}_{\mathrm{CRTf}}$ вычисляются следующим образом

$$\begin{gathered} U_{\mathrm{delay}}(\mathrm{COMP}_{\mathrm{CRTf}}) = 6{,}8\log_2 nN + 4\log_2 N + 0{,}5N + 13, \\ U_{\mathrm{area}}(\mathrm{COMP}_{\mathrm{CRTf}}) = 9N\log_2 N + 14nN^2 - 11N + 3. \end{gathered} \tag{3.4.3}$$

Проведем сравнение технических характеристик предлагаемого устройства сравнения чисел в СОК на основе КТОд и устройств на основе КТО [79] и ОПСС [58]. Так как при переводе числа из СОК в ПСС с помощью КТО (1.3.2) производится умножение на константы, то как и в случае КТОд частичные произведения могут быть вычислены заранее [5, 77]. Тогда параметры площади и задержки устройства $U_{\mathrm{delay}}(\mathrm{RC}_{\mathrm{CRT}})$ вычисляются по формулам

$$U_{\text{delay}}(\text{RC}_{\text{CRT}}) = 6{,}8\log_2 n(DR+\beta) +$$
$$+2\log_2\big((n+1)(DR+\beta)-1\big) +$$
$$+(n\cdot DR+\beta+n\beta)(4\log_2 DR+18)+4,$$
$$U_{\text{area}}(\text{RC}_{\text{CRT}}) = \qquad (3.4.4)$$
$$= 3\big((n+1)(DR+\beta)-1\big)\log_2\big((n+1)(DR+\beta)-1\big) +$$
$$+6DR\log_2 DR + 7n\big((n+1)(DR+\beta)-1\big)(DR+\beta) -$$
$$-11\big((n+1)(DR+\beta)-1\big)+23DR+3.$$

Тогда, учитывая параметры COMP (3.4.1) и RC_{CRT} (3.4.4), параметры площади и задержки устройства COMP_{CRT} вычисляются следующим образом

$$U_{\text{delay}}(\text{COMP}_{\text{CRT}}) = 6{,}8\log_2 n(DR+\beta) +$$
$$+2\log_2\big((n+1)(DR+\beta)-1\big) +$$
$$+(n\cdot DR+\beta+n\beta)(4\log_2 DR+18)+2\log_2 DR +$$
$$+0{,}5DR+13+2\log_2\alpha+0{,}5\alpha+9,$$
$$U_{\text{area}}(\text{COMP}_{\text{CRT}}) = \qquad (3.4.5)$$
$$= 3\big((n+1)(DR+\beta)-1\big)\log_2\big((n+1)(DR+\beta)-1\big) +$$
$$+9DR\log_2 DR + 7n\big((n+1)(DR+\beta)-1\big)(DR+\beta) -$$
$$-11\big((n+1)(DR+\beta)-1\big)+34DR+4.$$

Аналогично, учитывая параметры COMP (3.4.1) и NPC_{MRC} (3.2.8), параметры площади и задержки устройства COMP_{MRC} для сравнения чисел вычисляются по следующим формулам

$$U_{\text{delay}}(\text{COMP}_{\text{MRC}}) = 6{,}8n\log_2(n\beta) + 2n\log_2(2\beta+n) +$$
$$+n(\beta+n+1)(4\log_2\beta+18)+2\log_2 DR+2n\beta^2+n\beta +$$
$$+10{,}8n+0{,}5DR+9,$$
$$U_{\text{area}}(\text{COMP}_{\text{MRC}}) = 6n(2\beta+n)\log_2(2\beta+n)+12n\beta\log_2\beta + \qquad (3.4.6)$$
$$+60n^2\beta^2+28n^3\beta+2n^2\beta+2n\beta-22n^2+6n +$$
$$+3DR\log_2 DR+11DR+1.$$

В таблице 3.4.1 представлены теоретические параметры площади и задержки устройств сравнения чисел в СОК на основе КТОд, КТО и ОПСС,

рассчитанные с помощью «unit-gate»-модели. Теоретический анализ параметров устройств показал, что задержка устройства на основе КТОд в 101,9-111,84 раза меньше, по сравнению с задержкой устройства на основе КТО, и в 35,95-49,48 раза меньше по сравнению с задержкой устройства на основе ОПСС. Кроме того, метод на основе КТОд позволяет уменьшить площадь устройства в 3,5-3,64 раза по сравнению с методом на основе КТО и на 13,9-52,34% по сравнению с методом на основе ОПСС.

Таблица 3.4.1. Теоретические параметры площади и задержки устройств сравнения чисел в СОК

Параметр	Набор модулей	Метод		
		КТОд	КТО	ОПСС
Задержка	{23, 1433, 13, 29, 681, 821}	**120**	13421	5162
	{5, 2399, 7, 11, 23, 1691}	**115**	11718	5690
	{47, 293, 25, 193, 41, 257}	**116**	12021	4170
Площадь	{23, 1433, 13, 29, 681, 821}	**247150**	864135	336816
	{5, 2399, 7, 11, 23, 1691}	**187392**	683009	393221
	{47, 293, 25, 193, 41, 257}	**203625**	711991	236488

Аппаратное моделирование рассмотренных методов было выполнено на Artix 7 xc7a200tfbg484-2 в среде Xilinx Vivado 2016.3 со стратегией оптимизации по площади устройства. Результаты моделирования устройств сравнения чисел на основе КТОд, КТО и ОПСС представлены в таблице 3.4.2. Аппаратное моделирование показало, что задержка устройства на основе КТОд на 40,85-74,78% меньше, а площадь на 9,99-39,2% меньше по сравнению с устройствами на основе КТО и ОПСС.

Таблица 3.4.2. Результаты аппаратного моделирования устройств сравнения чисел в СОК

Параметр	Набор модулей	КТОд	КТО	ОПСС
Задержка, нс	{23, 1433, 13, 29, 681, 821}	**25,56**	50,663	96,201
	{5, 2399, 7, 11, 23, 1691}	**27,292**	47,803	102,618
	{47, 293, 25, 193, 41, 257}	**28,552**	48,268	113,216
Количество LUT	{23, 1433, 13, 29, 681, 821}	**2572**	4230	3293
	{5, 2399, 7, 11, 23, 1691}	**3630**	4430	4033
	{47, 293, 25, 193, 41, 257}	**3748**	4999	4822

3.5 Метод проектирования устройства, реализующего операцию выбора максимального элемента из окрестности на основе Китайской теоремы об остатках с дробными величинами

На рисунке 3.5.1 изображена архитектура устройства выбора максимального из двух чисел $\{a_1, a_2, \ldots, a_n\}$ и $\{b_1, b_2, \ldots, b_n\}$, представленных в СОК с модулями $\{p_1, p_2, \ldots, p_n\}$ и разрядностями $\{\beta_1, \beta_2, \ldots, \beta_n\}$ по каждому модулю соответственно. Сначала с помощью устройств $\mathrm{NPC_{CRTf}}$ вычисляются позиционные характеристики A' и B'. Компаратор сравнивает позиционные характеристики и передает управляющий сигнал на мультиплексор, в соответствии с которым выбирается число, которое больше.

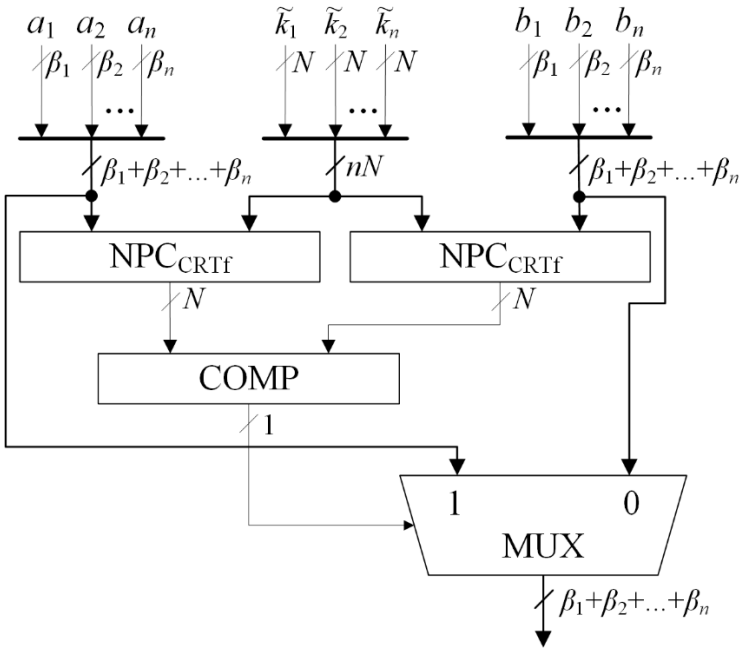


Рисунок 3.5.1. Архитектура устройства вычисления большего из чисел в СОК

Так как в СНС слой max pooling идет после сверточного слоя, то позиционная характеристика числа получена на этапе вычисления функции активации ReLU. Следовательно, архитектура устройства вычисления большего из чисел в СОК (Рисунок 3.5.1) преобразуется к виду, представленному на рисунке 3.5.2. На вход устройства поступают числа двух

чисел $\{a_1, a_2, \ldots, a_n\}$ и $\{b_1, b_2, \ldots, b_n\}$, представленные в СОК, и их позиционные характеристики A' и B'.

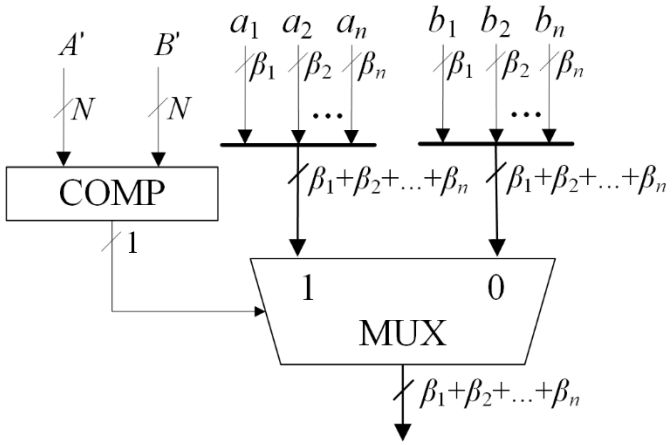


Рисунок 3.5.2. Архитектура устройства MAX вычисления большего из чисел в СОК

Для вычисления максимального элемента из набора чисел необходимо составить дерево из нескольких устройств MAX выбора большего из двух чисел в СОК, представленного на рисунке 3.5.2. Таким образом, представленное устройство предлагается использовать при реализации слоя max pooling в СНС.

3.6 Перевод чисел из системы остаточных классов в позиционную систему счисления

Для перевода числа из СОК в ПСС согласно КТОд необходимо умножить позиционную характеристику $A^{'}$ на динамический диапазон P. При этом результатом алгоритма являются старшие биты, начиная с бита под номером N. Таким образом

$$A = \left\lfloor \frac{A^{'}P}{2^N} \right\rfloor. \qquad (3.6.1)$$

В (3.6.1) операция деления при аппаратной реализации игнорируется, так как на выход подаются старшие значащие биты начиная с N-го. В

программной реализации эта операция эквивалентна сдвигу на N разрядов вправо.

На рисунке 3.6.1 изображена архитектура устройства, аппаратно реализующего алгоритм обратного преобразования из СОК в ПСС на основе КТОд. Сначала с помощью устройства $\mathrm{NPC_{CRTf}}$ вычисляется позиционная характеристика числа. Затем, с помощью устройств PPG_{2^α} и MOMA_{2^α} (где α равно N и $N+DR$ соответственно) производится умножение на динамический диапазон P.

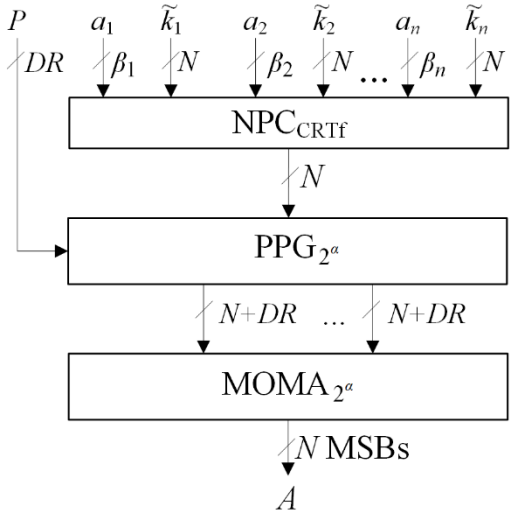


Рисунок 3.6.1. Архитектура устройства $\mathrm{RC_{CRTf}}$ обратного преобразования из СОК в ПСС на основе КТОд

Проведем теоретический анализ параметров площади и задержки устройства $\mathrm{RC_{CRTf}}$ обратного преобразования из СОК в ПСС на основе КТОд и сравним их с параметрами устройств на основе КТО ($\mathrm{RC_{CRT}}$) и ОПСС ($\mathrm{RC_{MRC}}$).

Учитывая параметры устройств $\mathrm{NPC_{CRTf}}$ (3.2.4), PPG_{2^α} (2.4.24) и MOMA_{2^α} (2.4.19), параметры устройства $\mathrm{RC_{CRTf}}$ могут быть вычислены по формулам

$$U_{\mathrm{delay}}(\mathrm{RC_{CRTf}}) = 15{,}6\log_2 N + 6{,}8\log_2 DR + 2\log_2(N+DR) + \\ +6{,}8\log_2 n + N^2 + N + 8, \tag{3.6.2}$$

$$U_{\text{area}}(\text{RC}_{\text{CRTf}}) = 3(N+DR)\log_2(N+DR) + 7{,}5nN^2 +$$
$$+0{,}5(N+DR)^2 + 3N\log_2 N + 0{,}5nN +$$
$$+7N\cdot DR(N+DR) - 21{,}5N - 10{,}5DR + 2.$$

Параметры устройства RC_{CRT} на основе КТО вычисляются по формулам (3.2.5). А параметры устройства RC_{MRC} на основе ОПСС могут быть рассчитаны следующим образом

$$U_{\text{delay}}(\text{RC}_{\text{MRC}}) = 6{,}8n\log_2 n\beta + 2n\log_2(2\beta+n) +$$
$$+n(\beta+n+1)(4\log_2\beta + 18) + 6{,}8\log_2 n + 6{,}8\log_2(DR+\beta) +$$
$$+2\log_2(DR+\beta+n-1) + 2n\beta^2 + n\beta + 10{,}8n +$$
$$+0{,}5(DR+\beta)^2 + 0{,}5(DR+\beta) + 4,$$
$$U_{\text{area}}(\text{RC}_{\text{MRC}}) = 3n(2\beta+n)\log_2(2\beta+n) + 6n\beta\log_2\beta + \qquad (3.6.8)$$
$$+30n^2\beta^2 + 3(DR+\beta+n-1)\log_2(DR+\beta+n-1) +$$
$$+14n^3\beta + n^2\beta + n\beta - 11n^2 + 3n + 0{,}5n(DR+\beta)^2 +$$
$$+0{,}5n(DR+\beta) + 7n(DR+\beta+n-1)(DR+\beta) -$$
$$-11(DR+\beta+n-1) + 1.$$

В таблице 3.6.1 представлены параметры устройств обратного преобразования чисел из СОК в ПСС, вычисленные с помощью «unit-gate»-модели. Результаты теоретического анализа показали, что устройство на основе КТОд имеет в 2,2-4,83 раза меньшую задержку по сравнению с устройствами на основе других рассмотренных методов. Однако, использование КТОд увеличивает площадь устройства в 1,88-5,53 раза по сравнению с методами на основе КТО и ОПСС.

Таблица 3.6.1. Теоретические параметры площади и задержки устройств обратного преобразования чисел из СОК в ПСС

Параметр	Набор модулей	Метод		
		КТОд	КТО	ОПСС
Задержка	{23, 1433, 13, 29, 681, 821}	**3136**	13380	6884
	{5, 2399, 7, 11, 23, 1691}	**2417**	11681	7110
	{47, 293, 25, 193, 41, 257}	**2613**	11981	5623
Площадь	{23, 1433, 13, 29, 681, 821}	1713967	862961	**309963**
	{5, 2399, 7, 11, 23, 1691}	1087027	682053	**309249**
	{47, 293, 25, 193, 41, 257}	1334316	710911	**235822**

Для сравнения характеристик рассматриваемых устройств было проведено аппаратное моделирование в среде Vivado 2016.3 с использованием целевой платы Xilinx Artix 7 xc7a200tfbg484-2 на языке описания аппаратуры VHDL. Результаты моделирования схем обратного преобразования на основе КТОд, КТО и ОПСС представлены в таблице 3.6.2.

Таблица 3.6.2. Результаты моделирования устройств обратного преобразования из СОК в ПСС

Параметр	Набор модулей	КТОд	КТО	ОПСС
Задержка, нс	{23, 1433, 13, 29, 681, 821}	**33,370**	46,789	101,835
	{5, 2399, 7, 11, 23, 1691}	**36,820**	44,870	113,386
	{47, 293, 25, 193, 41, 257}	**38,929**	45,524	125,090
Количество LUT	{23, 1433, 13, 29, 681, 821}	2877	**2059**	2228
	{5, 2399, 7, 11, 23, 1691}	3657	**2159**	2804
	{47, 293, 25, 193, 41, 257}	4667	**2452**	3367

Моделирование обратного преобразования из СОК в ПСС показало, что устройство на основе КТОд работает на 14,49-68,88% быстрее по сравнению с устройствами на основе КТО и ОПСС. Но уменьшение задержки устройств обратного преобразования на основе КТОд достигается за счет увеличения площади устройства (количества занятых LUT), на 29,13-90,33% по сравнению с устройствами на основе других рассмотренных методов.

3.7 Выводы по третьей главе

В третьей главе рассмотрены подходы к выполнению вычислительно сложных операций в СОК и предложены реализующие их устройства.

Предложен метод преобразования чисел из ПСС в СОК с модулями специального вида 2^{α} и $2^{\alpha}-1$ и его реализация на языке описания аппаратуры VHDL. Предложены архитектуры устройств определения знака числа, сравнения чисел в СОК, а также архитектура устройства обратного преобразования в СОК на основе КТОд.

Проведен теоретический анализ технических параметров предлагаемых архитектур устройств на основе «unit-gate»-модели, и выполнено аппаратное моделирование на FPGA. Результаты теоретического анализа и моделирования показали, что разработанные архитектуры устройств, реализующих вычислительно сложные операции в СОК, обладают меньшей задержкой.

Предложен метод проектирования устройства, реализующего функцию активации ReLU в СОК на основе КТОд. Разработан метод проектирования устройства, реализующего операцию выбора максимального элемента из окрестности на основе КТОд, который может быть применен для реализации слоя max pooling СНС.

Предложенные в данной главе методы могут быть использованы для реализации компонентов СНС.

4 АППАРАТНАЯ РЕАЛИЗАЦИЯ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

4.1 Алгоритм проектирования аппаратной реализации сверточной нейронной сети в системе остаточных классов

Рассмотрим каждый этап предлагаемого алгоритма проектирования аппаратной реализации СНС.

Входные данные: архитектура СНС.

Этап 1. Выбор набора модулей СОК.

Как было отмечено ранее, выбор набора модулей СОК влияет на производительность вычислений. Набор модулей должен обеспечивать достаточный динамический диапазон системы и модули должны быть сбалансированы таким образом, чтобы время выполнения операций по каждому каналу было примерно одинаково, и не было длительного простоя системы по какому-либо вычислительному каналу. В работе предлагается использовать модули СОК специального вида 2^{α} и $2^{\alpha}-1$, которые позволяют избежать затратной по ресурсам операции деления по модулю.

Этап 2. Разработка блока преобразования из ПСС в СОК.

Для выполнения вычислений в СОК, сначала необходимо произвести преобразование из ПСС в СОК. На рисунке 4.1.1 изображено устройство PNS→RNS для перевода DR-битного числа A в СОК с модулями вида $\{2^{\alpha_1}, 2^{\alpha_2}-1, \ldots, 2^{\alpha_n}-1\}$. Вычисление остатков от деления производится с помощью устройств $\mathrm{MOD}_{2^{\alpha}}$ и $\mathrm{MOD}_{2^{\alpha}-1}$ (рис. 3.1.1), рассмотренных ранее. На выходе устройства PNS→RNS формируется число $\{a_1, a_2, \ldots, a_n\}$, представленное в СОК и имеющее разрядности $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ соответственно. Далее вычисления в СОК выполняются параллельно по каждому вычислительному каналу, соответствующему модулю системы.

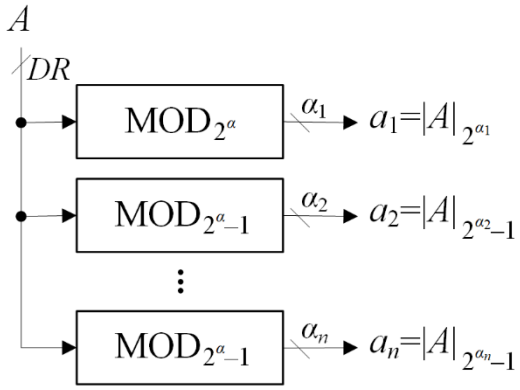


Рисунок 4.1.1. Архитектура устройства PNS→RNS преобразования чисел из ПСС в СОК с модулями вида $\{2^{\alpha_1}, 2^{\alpha_2}-1, \ldots, 2^{\alpha_n}-1\}$

Таким образом, в архитектуре СНС с вычислениями в СОК слои сети состоят из одного вычислительного канала по модулю 2^α и $n-1$ вычислительных каналов по модулю вида $2^\alpha - 1$.

Этап 3. Реализация блоков, выполняющих операцию свертки в СОК параллельно по каждому модулю системы.

Рассмотрим реализацию сверточного слоя СНС с вычислениями в СОК. Процедура получения одной карты признаков в сверточном слое (1.1.1) по модулю m_l может быть представлена в виде

$$\left|I_f(x,y)\right|_{m_l} = \\ = \left| |b|_{m_l} + \left| \sum_{i=-t}^{t} \sum_{j=-t}^{t} \sum_{z=0}^{D-1} \left|W_{i,j,z}\right|_{m_l} \cdot |I(x+i, y+j, z)|_{m_l} \right|_{m_l} \right|_{m_l}, \quad (4.1.1)$$

где I_f – карта признаков после свертки, $W_{i,j,z}$ – это коэффициенты 3D-фильтра размерности $d \times d$ для обработки D двухмерных массивов, $t = \left\lfloor \frac{d}{2} \right\rfloor$ и b – смещение [28].

Пусть $F = \{F_0, F_1, \ldots, F_{D-1}\}$ набор из D векторов размерности d^2, которые соответствуют фрагментам размерности $d \times d$ карт признаков I, поступающих на вход сверточного слоя. Аналогично, представим маску фильтра как набор из D векторов размерности d^2 и обозначим как $W = \{W_0, W_1, \ldots, W_{D-1}\}$. Тогда операция свертки по модулю m_l для вычисления

одного значения R карты признаков I_f может быть представлена в виде суммы одномерных сверток и смещения

$$|R|_{m_l} = \left| |b|_{m_l} + \left| \sum_{i=0}^{D-1} \sum_{j=0}^{d^2-1} |W_{i,j}|_{m_l} \cdot |F_{i,j}|_{m_l} \right|_{m_l} \right|_{m_l}. \qquad (4.1.2)$$

Одномерная свертка может быть реализована с помощью FIR фильтра [101]. Результаты одномерной свертки суммируются с помощью сумматора со множественным входом. Если расчеты производятся по модулю 2^{α}, то биты старше α не участвуют в вычислениях. Для выполнения вычислений по модулю $2^{\alpha}-1$ используется техника циклического переноса старших бит EAC. На рисунке 4.1.2 представлена схема устройства $\mathrm{CONV}_{2^{\alpha}}$ для свертки фрагмента изображения по модулю 2^{α}. Устройство свертки $\mathrm{CONV}_{2^{\alpha}-1}$ по модулю $2^{\alpha}-1$ имеет аналогичную структуру.

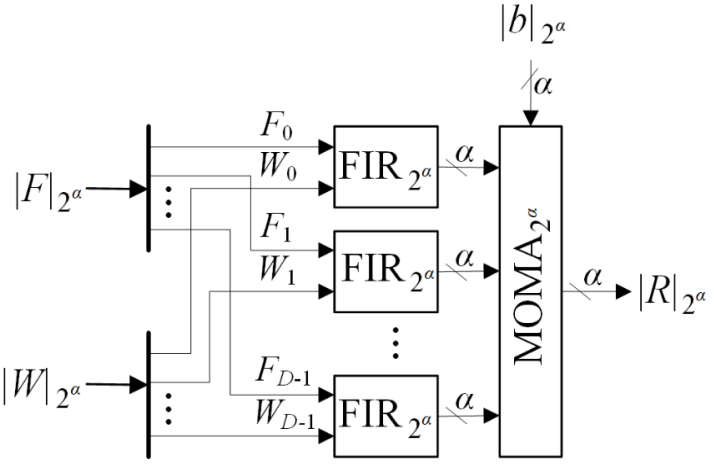


Рисунок 4.1.2. Архитектура устройства $\mathrm{CONV}_{2^{\alpha}}$ для свертки по модулю 2^{α} фрагмента карты признаков

Во второй главе предложен новый метод проектирования устройства, реализующего операцию свертки на основе метода Винограда и СОК с модулями специального вида, для реализации цифровых фильтров и проведено сравнение с известными архитектурами цифровых фильтров. Поскольку предлагаемый метод показал преимущество в производительности

по сравнению с известными аналогами, он может быть успешно применен для реализации операции свертки в СНС (4.1.2).

Операция свертки так же применяется в полносвязных слоях нейронов, которые обычно являются заключительными слоями сети. На вход данного слоя поступает вектор входных данных $X = \{x_i\}$, где $0 \leq i < m$, m – количество элементов вектора. Для каждого нейрона производится свертка входного вектора X с соответствующим вектором весовых коэффициентов $W_j = \{w_{ji}\}$, и результат суммируется со смещением b_j, где $0 \leq j < n$, n – количество нейронов (1.1.4). Далее применяется функция активации (1.1.5).

На рисунке 4.1.3 представлена архитектура устройства FC_{2^α}, выполняющего вычисления по формуле (1.1.4) для j-го нейрона по модулю 2^α. Устройство $\mathrm{FC}_{2^\alpha-1}$, выполняющее вычисления для j-го нейрона по модулю $2^\alpha - 1$ имеет аналогичную архитектуру, но использует технику EAC. Результаты работы устройств FC_{2^α} и $\mathrm{FC}_{2^\alpha-1}$ подаются на вход устройства ReLU.

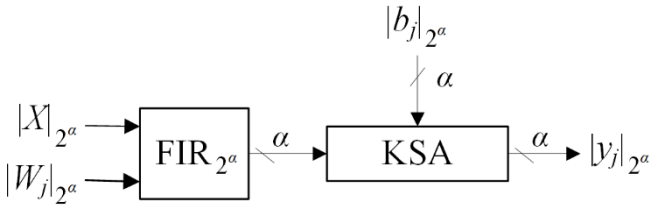


Рисунок 4.1.3. Архитектура устройства FC_{2^α} полносвязного слоя с вычислениями по модулю 2^α

Этап 4. Квантование весовых коэффициентов, представленных действительными числами.

Весовые коэффициенты СНС являются вещественными числами. Для аппаратной реализации СНС требуется представить их в целочисленном виде. Для этого весовые коэффициенты необходимо умножить на 2^p и округлить результат к большему целому. После выполнения вычислений, полученный

результат масштабируется на 2^{-p} и округляется к меньшему целому. Коэффициент масштабирования p показывает сколько бит необходимо для представления весовых коэффициентов в устройстве. От выбора p зависит точность работы СНС и объем памяти, необходимый для хранения весовых коэффициентов.

Этап 5. Реализация блоков, выполняющих вычисление функций активации.

На выходе сверточного слоя применяется функция активации. Как говорилось ранее, наиболее распространенной функцией является ReLU (1.1.2) [41], которая сводится к определению знака числа. Данная операция считается вычислительно сложной в СОК. Анализ методов реализации устройств определения знака числа в СОК (параграф 3.2) показал, что по сравнению с другими рассмотренными методами наиболее эффективным по скорости и аппаратным затратам является метод на основе КТОд. Для реализации функции активации ReLU в СОК предлагается использовать устройство, представленное на рисунке 3.3.1.

Этап 6. Реализация блоков, выполняющих операцию выбора максимального элемента окрестности (блок max pooling).

Слой max pooling (1.1.3) так же требует выполнения вычислительно сложной операции в СОК – сравнения чисел. Сравнительный анализ методов реализации операции сравнения двух чисел в СОК (параграф 3.4) показал, что метод на основе КТОд имеет меньшую задержку и площадь устройства по сравнению с другими рассмотренными методами. Устройство MAX для определения большего из двух чисел в СОК по методу на основе КТОд представлено на рисунке 3.5.2. Наиболее часто на слое max pooling используется значение $s = 2$. То есть рассматривается окрестность размерности 2×2, и вычисления производятся с шагом 2. На рисунке 4.1.4 представлена схема устройства для выбора максимального из четырех чисел в СОК. На вход устройства подаются числа $A = \{a_1, a_2, \dots, a_n\}$, $B = \{b_1, b_2, \dots, b_n\}$, $C = \{c_1, c_2, \dots, c_n\}$ и $D = \{d_1, d_2, \dots, d_n\}$, представленные в СОК.

Так как слой max pooling всегда идет после сверточного слоя, то позиционные характеристики чисел были рассчитаны при вычислении функции активации ReLU и поступают на вход устройства. Выбор наибольшего числа производится с помощью дерева устройств MAX.

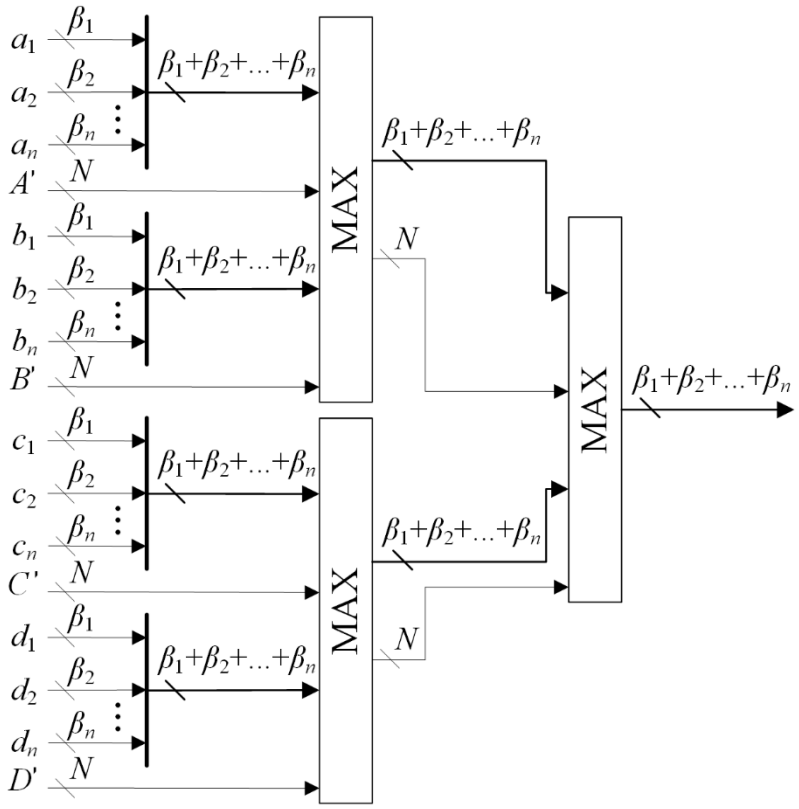


Рисунок 4.1.4. Архитектура устройства max pooling для выбора максимального из четырех чисел в СОК

Этап 7. Разработка блока обратного преобразования из СОК в ПСС.

Последним блоком системы, выполняющей вычисления в СОК, является преобразование результата обратно в ПСС. Для перевода числа из СОК в ПСС предлагается использовать метод на основе КТОд, который показал преимущество в скорости работы устройства по сравнению с рассмотренными аналогами (параграф 3.6). Таким образом, архитектура устройства RNS→PNS для обратного преобразования чисел из СОК в ПСС такая же, как и архитектура устройства RC_{CRTf}, изображенного на рисунке 3.6.1.

Выходные данные: архитектура аппаратной реализации СНС.

4.2 Архитектура аппаратной реализации сверточной нейронной в системе остаточных классов

Основываясь на алгоритме проектирования аппаратной реализации СНС, представленном в предыдущем параграфе, разработана архитектура СНС с вычислениями в СОК с модулями специального вида $\{2^{\alpha_1}, 2^{\alpha_2} - 1, \ldots, 2^{\alpha_n} - 1\}$ в общем виде изображенная на рисунке 4.2.1. На вход поступает изображение в виде последовательности пикселей. Сначала в блоке PNS→RNS производится преобразование данных в СОК. Затем они поступают в оперативное запоминающее устройство (ОЗУ) и передаются в другие блоки СНС (сверточные слои, слои max pooling и полносвязные слои). Весовые коэффициенты СНС хранятся в постоянном запоминающем устройстве (ПЗУ). Перевод весовых коэффициентов в СОК производится с помощью блоков PNS→RNS, затем данные в формате СОК поступают на сверточные и полносвязные слои. Промежуточные результаты работы слоев хранятся в буферной памяти в ОЗУ. Результат работы СНС переводится в ПСС с помощью блока RNS→PNS и поступает на выход устройства.

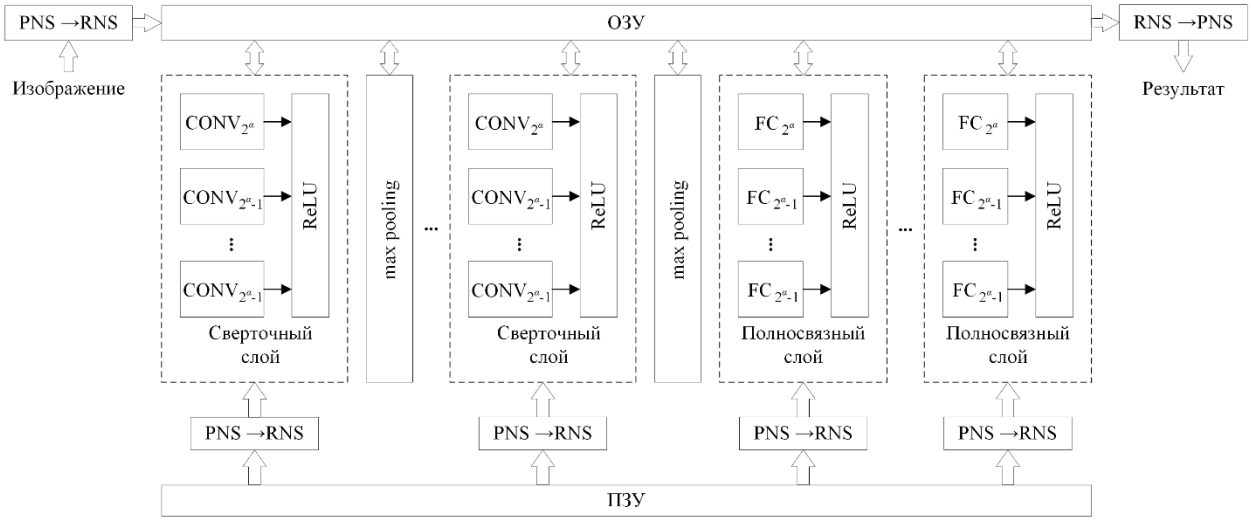


Рисунок 4.2.1. Предлагаемая архитектура аппаратной реализации СНС с вычислениями в СОК

Представленный алгоритм для аппаратной реализации СНС с вычислениями в СОК позволяет выполнять вычисления в сверточных и полносвязных слоях параллельно с числами меньшей размерности чем в ПСС, что увеличивает быстродействие системы. В следующем параграфе представлен пример применения на практике предложенного метода.

4.3 Комплекс программ на языке описания аппаратуры для аппаратной реализации сверточной нейронной сети с использованием модулярных вычислений на FPGA

Разработан комплекс программ на языке описания аппаратуры VHDL для аппаратной реализации СНС с использованием модулярных вычислений на FPGA. Предлагаемый комплекс программ состоит из следующих модулей:

1) модуль описания архитектуры СНС в СОК;
2) библиотека компонентов СНС в СОК;
3) библиотека компонентов, выполняющих арифметические операции в СОК.
4) модуль преобразования данных из ПСС в СОК;
5) модуль преобразования данных из СОК в ПСС;
6) модуль ОЗУ;
7) модуль ПЗУ;
8) модуль управления памятью.

На рисунке 4.3.1 представлена иерархическая структура взаимодействия модулей и библиотек предлагаемого комплекса программ для аппаратной реализации СНС с использованием модулярных вычислений на FPGA.

Модуль описания архитектуры СНС в СОК задает параметры каждого блока архитектуры СНС (рис. 4.2.1) и описывает связи между ними. Данный модуль взаимодействует с библиотекой компонентов СНС в СОК, модулем преобразования данных из ПСС в СОК, модулем преобразования данных из СОК в ПСС и модулем управления памятью.

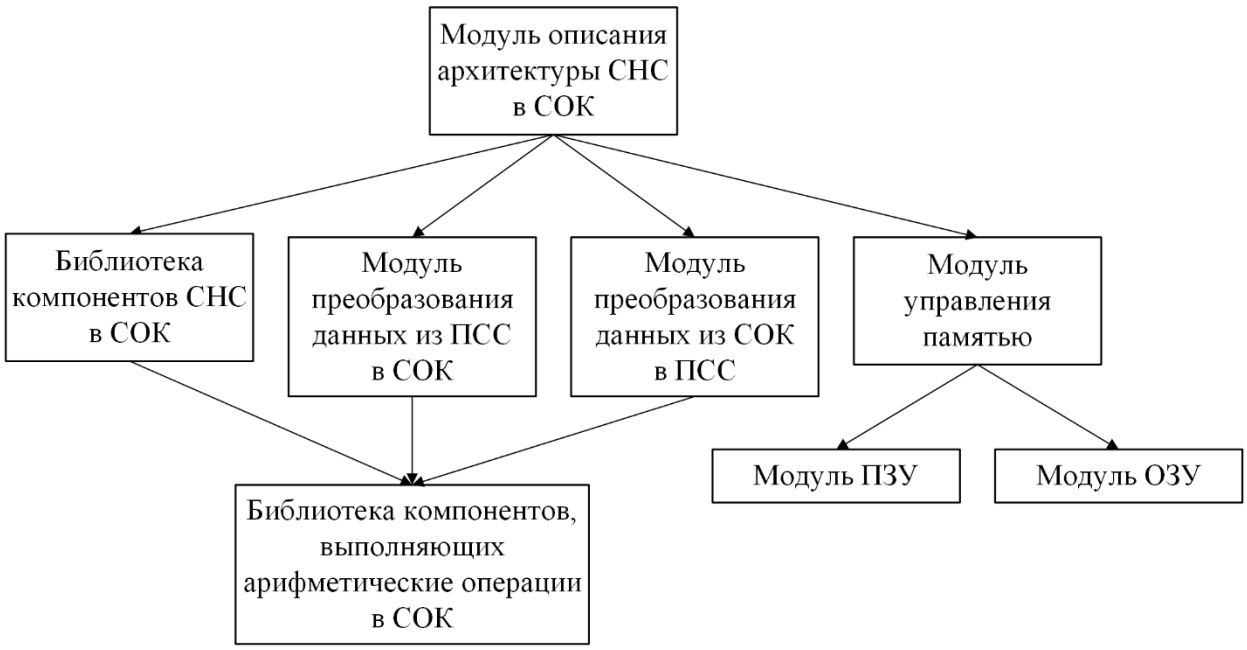


Рисунок 4.3.1. Структура комплекса программ для аппаратной реализации СНС

Библиотека компонентов СНС в СОК взаимодействует с библиотекой компонентов, выполняющих арифметические операции в СОК, и описывает следующие устройства

– устройство CONV_{2^α} для свертки по модулю 2^α;

– устройство $\mathrm{CONV}_{2^\alpha-1}$ для свертки по модулю $2^\alpha - 1$;

– устройство ReLU, реализующее функцию активации;

– устройство max pooling для выбора максимального элемента окрестности;

– устройство FC_{2^α} с вычислениями по модулю 2^α

– устройство $\mathrm{FC}_{2^\alpha-1}$ с вычислениями по модулю $2^\alpha - 1$.

Библиотека компонентов СНС в СОК содержит программные реализации на языке описания аппаратуры следующих методов

– метод проектирования устройства, реализующего операцию свертки на основе метода Винограда и СОК с модулями специального вида, представленный в параграфе 2.3;

– метод проектирования устройства, реализующего функцию активации ReLU на основе КТОд, представленный в параграфе 3.3;

– метод проектирования устройства, реализующего операцию выбора максимального элемента из окрестности на основе КТОд, представленный в параграфе 3.5.

Библиотека компонентов, выполняющих арифметические операции в СОК с модулями вида 2^{α} и $2^{\alpha}-1$ описывает модульные сумматоры, умножители по модулю и генераторы частичных произведений по модулю.

Модуль преобразования данных из ПСС в СОК описывает устройство PNS→RNS. На вход которого подается число в ПСС и разрядности модулей СОК. На выход поступает число, представленное в СОК. Данный модуль является программной реализацией на языке описания аппаратуры метода преобразования чисел из ПСС в СОК с модулями специального вида, представленного в параграфе 3.1. Модуль преобразования данных из ПСС в СОК взаимодействует с библиотекой компонентов, выполняющих арифметические операции в СОК.

Модуль преобразования данных из СОК в ПСС описывает устройство RNS→PNS. На вход подается число, представленное в СОК, коэффициенты для вычисления позиционной характеристики числа с помощью КТОд и динамический диапазон СОК. На выход поступает число в ПСС. Модуль взаимодействует с библиотекой компонентов, выполняющих арифметические операции в СОК.

Модуль ПЗУ описывает внешнюю память, в которой хранятся весовые коэффициенты СНС. Весовые коэффициенты записывались в файлы с расширением сое.

Модуль ОЗУ описывает встроенную память, используемую в качестве буфера при передаче данных между блоками СНС.

Модуль управления памятью обеспечивает доступ к внутренней и внешней памяти устройства. Модуль взаимодействует с модулем ПЗУ, модулем ОЗУ и модулем описания архитектуры СНС в СОК.

4.3 Эксперимент и результаты

Рассмотрим применение предлагаемого алгоритма для аппаратной реализации СНС с использованием СОК с набором модулей вида $\{2^{\alpha_1}, 2^{\alpha_2} - 1, \ldots, 2^{\alpha_n} - 1\}$ на примере архитектуры LeNet-5 [73]. Обучение производилось с помощью библиотек машинного обучения TensorFlow [40] и Keras, использовался язык программирования Python. Функция активации гиперболический тангенс (tanh) заменена на ReLU. Параметры архитектуры LeNet-5 представлены в таблице 4.3.1.

Таблица 4.3.1. Параметры архитектур СНС LeNet-5

Слой	Размер маски фильтра	Количество фильтров	Функция активации
свертка	5×5×3	6	ReLU
max pooling	2×2	1	
свертка	5×5×6	16	ReLU
max pooling	2×2	1	
свертка	5×5×16	120	ReLU
полносвязный	84 нейрона		ReLU
полносвязный	10 нейронов		softmax

Для обучения СНС использовались базы изображений MNIST [73], FMNIST [109] и CIFAR-10 [70]. База MNIST содержит изображения рукописных цифр от 0 до 9 размера 28×28 в оттенках серого и состоит из 60000 изображений для обучения и 10000 изображений для тестирования. База FMNIST содержит 10 классов изображений одежды и обуви (футболка, брюки, свитер, платье, пальто, сандалии, рубашка, кроссовки, сумка, ботинки) размера 28×28 в оттенках серого, состоит из 60000 изображений для обучения и 10000 изображений для тестирования. База CIFAR-10 содержит 10 классов изображений (самолет, автомобиль, птица, кот, олень, собака, лягушка, лошадь, корабль, грузовик) размера 32×32 формата RGB, состоит из 50000 изображений для обучения и 10000 изображений для тестирования.

Был проведен анализ влияния разрядности весовых коэффициентов СНС на точность распознавания. Основываясь на результатах эксперимента (табл. 4.3.2) можно сделать вывод, что разрядность весовых коэффициентов может быть уменьшена без потери точности распознавания. Для архитектуры СНС LeNet-5, обученной на базах MNIST и FMNIST, достаточно разрядности весовых коэффициентов 8 бит, а архитектуре, обученной на базе CIFAR-10, необходимо 12 бит. В таблице 4.3.2 представлено какой объем памяти требуется для хранения весовых коэффициентов СНС в зависимости от их разрядности. Таким образом, для архитектуры с 8- и 12-битным представлением весовых коэффициентов требуется в 4 и 2,67 раза меньший объем памяти соответственно по сравнению с 32-битным представлением.

Таблица 4.3.2. Результаты работы СНС LeNet-5 с различной разрядностью весовых коэффициентов

Разрядность, бит	Точность распознавания, %			Объем памяти, Кбайт
	MNIST	FMNIST	CIFAR-10	
4	25,17	10,13	11,15	31,003
6	97,92	81,77	14,68	46,505
8	**98,87**	**89,06**	56,84	62,006
12	98,82	89,16	**61,28**	93,009
16	98,83	89,16	61,20	124,012
20	98,83	89,15	61,22	155,015
24	98,83	89,15	61,23	186,018
28	98,83	89,15	61,23	217,021
32	98,83	89,15	61,23	248,024

Было проведено сравнение нескольких конфигураций архитектур СНС, представленных в таблице 4.3.3. Устройство с вычислениями в ПСС является 32-разрядным. Для организации вычислений в СОК был выбран набор модулей $\{2^{12}, 2^{11}-1, 2^{10}-1\}$. Аппаратное моделирование было проведено в среде Xilinx Vivado 2018.3 на целевой плате Virtex-7 xc7v2000tfhg1761-2L со стратегией оптимизации AreaOpimized_high. Результаты аппаратного моделирования представлены в таблице 4.3.4. Для оценки эффективности были рассмотрены временные и аппаратные затраты устройств. К временным затратам относится тактовая частота устройства, измеряющаяся в МГц, и

производительность, измеряющаяся количеством обработанных кадров в секунду (кадр/с). Для наборов данных MNIST и FMNIST размер кадра составляет 28×28 пикселей, а для набора данных CIFAR-10 размер кадра 32×32 пикселя. Под аппаратными затратами подразумевается количество занятых просмотровых таблиц (Look-Up-Table, LUT), памяти с произвольным доступом (Random Access Memory, RAM) LUTRAM и Block RAM (BRAM), а также энергопотребление устройства, которое измеряется в Вт.

Таблица 4.3.3. Конфигурации архитектур СНС

Номер архитектуры	Вид цифровых фильтров		Система счисления	
	FIR(MAC)	$F(2 \times 2, 5 \times 5)$	ПСС	СОК
1		+		+
2		+	+	
3	+		+	
4	+			+

Таблица 4.3.4. Результаты аппаратного моделирования СНС LeNet-5

Параметр	Набор данных	Архитектура			
		1	2	3	4
Тактовая частота, МГц	MNIST	**72**	70	50	56
	FMNIST				
	CIFAR-10	**64**	61	53	59
Количество LUT	MNIST	976576	917247	**593291**	647821
	FMNIST				
	CIFAR-10	793891	911182	612196	**557297**
Количество LUTRAM	MNIST	519	6308	**483**	2212
	FMNIST				
	CIFAR-10	518	6369	**483**	2212
Количество BRAM	MNIST	57	**19**	63	181
	FMNIST				
	CIFAR-10	76,5	**25,5**	69,5	200,5
Энергопотребление, Вт	MNIST	20,3	18,865	**9,326**	12,833
	FMNIST				
	CIFAR-10	18,678	20,685	**11,718**	13,518
Производительность, кадр/с	MNIST	**1556**	1513	272	305
	FMNIST				
	CIFAR-10	**1059**	1009	221	246

Таким образом, было проведено аппаратное моделирование СНС, использующей MAC блоки в сверточном слое [101], и сети, использующей свертку по методу Винограда. Данные архитектуры были спроектированы в

ПСС и с использованием предлагаемого метода для аппаратной реализации СНС в СОК. Результаты моделирования показали, что использование метода Винограда позволяет увеличить производительность устройства в 4,3-5,56 раза, а применение предлагаемого подхода организации вычислений в СОК увеличивает производительность устройства на 2,84-12,13%. Следует отметить, что использование метода Винограда привело к увеличению аппаратных затрат, таких как площадь устройства и энергопотребление.

4.4 Выводы по четвертой главе

В четвертой главе представлен алгоритм проектирования аппаратной реализации СНС и архитектура СНС, разработанная по предложенному алгоритму. Разработан комплекс программ на языке описания аппаратуры VHDL для аппаратной реализации СНС с вычислениями в СОК на FPGA, который разработана на основе предложенного алгоритма проектирования СНС в СОК

В качестве модулей СОК выбраны модули специального вида $\{2^{\alpha_1}, 2^{\alpha_2}-1, \ldots, 2^{\alpha_n}-1\}$. Использование модулей специального вида позволяет реализовать вычислительно сложную операцию деления по модулю комбинацией операции сложения и сдвига. С учетом необходимости сведения операций в СОК к целочисленным, весовые коэффициенты сети они преобразуются к целочисленному формату. Результаты эксперимента показали, что разрядность весовых коэффициентов может быть уменьшена без потери точности распознавания СНС. В сверточных слоях сети предлагается использовать цифровые фильтры на основе предлагаемого модифицированного метода Винограда с вычислениями в СОК. Устройство, вычисления функции активации ReLU предлагается проектировать с использованием метода определения знака числа на основе КТОд. Устройства, реализующие слой max pooling и обратное преобразования из СОК в ПСС, так же основаны на КТОд.

Было проведено сравнение архитектуры СНС, использующей свертку по методу Винограда в сверточном слое, и архитектуры, использующей МАС блоки. Данные архитектуры были спроектированы в ПСС и с использованием предлагаемого метода аппаратной реализации СНС в СОК. Результаты моделирования показали, что использование метода Винограда позволяет увеличить производительность устройства в 4,3-5,56 раза, а применение предлагаемого подхода организации вычислений в СОК увеличивает производительность устройства на 2,84-12,13%.

ЗАКЛЮЧЕНИЕ

В рамках диссертационного исследования разработаны методы для проектирования системы интеллектуального анализа изображений, направленные на повышение их производительности. Проведен анализ современного состояния исследований по теме диссертации. Установлено, что существует необходимость улучшения технических характеристик аппаратных реализаций глубоких нейронных сетей, поскольку они являются ресурсозатратными. Для решения данной проблемы в диссертации предлагается использовать модулярную арифметику. В работе предложены методы и алгоритмы для проектирования аппаратных ускорителей СНС. Основные результаты исследования могут быть сформулированы следующим образом:

1. Разработана математическая модель СНС с реализацией вычислений в СОК с модулями специального вида. Предложенная математическая модель описывает каждый слой СНС и блоки перевода чисел из ПСС в СОК и обратно. Вычислительно сложные операции в СОК реализованы с использованием позиционной характеристики числа, получаемой на основе КТОд.

2. Разработан метод преобразования чисел из ПСС в СОК с модулями специального вида 2^{α} и $2^{\alpha}-1$. Вычисление остатка от деления по модулю 2^{α} производится с помощью простой операции битового сдвига. Показано, что вычисления остатка от деления по модулю $2^{\alpha}-1$ сводятся к сложению α-битных чисел по модулю $2^{\alpha}-1$. Разработана программа на языке описания аппаратуры VHDL для проектирования устройства вычисления остатка от деления по модулю $2^{\alpha}-1$, которое состоит из дерева сумматоров EAC-CSA и сумматора EAC-KSA, использующих технику циклического переноса старших бит. Для оценки эффективности предложенной архитектуры был проведен теоретический анализ параметров площади и задержки устройства на основе «unit-gate»-модели и выполнено аппаратное моделирование на FPGA. Результаты теоретического анализа и аппаратного моделирования показали,

что предложенная архитектура устройства вычисления остатка от деления по модулю $2^{\alpha}-1$ обладает меньшей задержкой и площадью по сравнению с рассмотренными аналогами.

3. Разработан метод проектирования сверточных слоев СНС на основе метода Винограда и СОК с модулями специального вида. Разработаны архитектуры двумерных цифровых фильтров с масками размера 2×2, 3×3 и 5×5, а также программы на языке VHDL для проектирования предлагаемых устройств. Для оценки эффективности предложенных архитектур был проведен теоретический анализ параметров площади и задержки устройств на основе «unit-gate»-модели. Кроме того, проведено аппаратное моделирование на FPGA. Результаты теоретической оценки параметров и моделирования показали, что предложенные архитектуры цифровых фильтров обладают большей производительностью по сравнению с рассмотренными аналогами, но за счет увеличения аппаратных затрат.

4. Разработан метод проектирования функции активации ReLU, вычисление которой сводится к операции определения знака числа, являющейся вычислительно сложной в СОК. Предложена архитектура устройства определения знака числа в СОК на основе КТОд. Кроме того, разработана программа на языке VHDL для проектирования устройства, основанного на предложенной архитектуре. Результаты теоретической оценки технических параметров устройства и результаты аппаратного моделирования на FPGA показали, что разработанное устройство определения знака числа имеет меньшую задержку и площадь устройства по сравнению с рассмотренными аналогами.

5. Разработан метод проектирования слоя выборки максимального элемента из окрестности, требующего выполнения вычислительно сложной операции сравнения чисел в СОК. Разработана архитектура устройства сравнения чисел в СОК, использующая позиционную характеристику числа, вычисляемую на основе КТОд, а также программа на языке VHDL для проектирования предлагаемого устройства. Результаты теоретической оценки

технических параметров устройства и аппаратного моделирования на FPGA показали, что предлагаемое устройство обладает меньшей задержкой и площадью по сравнению с рассмотренными аналогами. Для сравнения более чем двух чисел в слое max pooling предлагается использовать дерево предлагаемых устройств.

6. Разработана архитектура устройства обратного преобразования чисел из СОК в ПСС на основе КТОд и ее программная реализация на языке VHDL. Проведен теоретический анализ временных и аппаратных затрат, а также аппаратное моделирование на FPGA. Разработанное устройство показало преимущество в скорости работы по сравнению с рассмотренными аналогами, но за счет увеличения площади.

7. Разработан алгоритм для аппаратной реализации СНС, который учитывает выбор модулей СОК, способ представления весовых коэффициентов в памяти устройства и использует разработанные оригинальные методы для проектирования компонентов СНС. Установлено, что разрядность весовых коэффициентов может быть снижена без потери точности распознавания, следовательно, требуется меньше памяти для их хранения в устройстве.

8. Разработан комплекс программ на языке описания аппаратуры VHDL для аппаратной реализации СНС с использованием модулярных вычислений на FPGA, включающей все разработанные методы для проектирования СНС в СОК. Рассмотрено несколько конфигураций СНС. Проведено сравнение архитектуры СНС, использующей свертку по методу Винограда в сверточном слое, и архитектуры, использующей MAC блоки. Данные архитектуры были спроектированы в ПСС и с использованием предлагаемого метода аппаратной реализации СНС в СОК. Результаты моделирования показали, что использование метода Винограда позволяет увеличить производительность устройства в 4,3-5,56 раза, а применение предлагаемого подхода организации вычислений в СОК увеличивает производительность устройства на 2,84-12,13%.

СПИСОК ЛИТЕРАТУРЫ

Статьи автора в журналах, рекомендованных ВАК РФ, Scopus, Web of Science

1. Валуева, М.В. Метод аппаратной реализации сверточной нейронной сети на основе системы остаточных классов / М.В. Валуева, Г.В. Валуев, М.Г. Бабенко, А.Н. Черных, Х. Кортес-Мендоса // Труды Института системного программирования РАН. – 2022. –№3 (34). – С. 61-74. – DOI: 10.15514/ISPRAS-2022-34(3)-5.
2. Валуева, М.В. Высокопроизводительные архитектуры цифровой фильтрации изображений в системе остаточных классов на основе метода Винограда / М.В. Валуева, П.А. Ляхов, Н.Н. Нагорнов, Г.В. Валуев // Компьютерная оптика. – 2022. – Т. 46, № 5. – С. 752-762. – DOI: 10.18287/2412-6179-CO-933.
3. Червяков, Н.И. Аппаратная реализация свёрточной нейронной сети с использованием вычислений в системе остаточных классов / Н.И. Червяков, П.А. Ляхов, Н.Н. Нагорнов, М.В. Валуева, Г.В. Валуев // Компьютерная оптика. – 2019. – Т. 43, № 5. – С. 857-868. – DOI: 10.18287/2412-6179-2019-43-5-857-868.
4. Boyvalenkov, P. Classification of Moduli Sets for Residue Number System With Spe-cial Diagonal Functions / P. Boyvalenkov, N. Chervyakov, M. Valueva [et al.] // IEEE Access. – 2020. – vol. 8. – P. 156104-156116. – DOI: 10.1109/ACCESS.2020.3019452.
5. Chervyakov, N.I. Residue Number System-Based Solution for Reducing the Hardware Cost of a Convolutional Neural Network / N.I. Chervyakov, P.A. Lyakhov, M.V. Val-ueva [et al.] // Neurocomputing. – 2020. – Vol. 407, P. 439-453. – DOI: 10.1016/j.neucom.2020.04.018.

6. Lyakhov, P. A Method of Increasing Digital Filter Performance Based on Truncated Multiply-Accumulate Units / P. Lyakhov, M. Valueva, G. Valuev, N. Nagornov // Appl. Sci. – 2020. – Vol. 10, 9052. –DOI: /10.3390/app10249052.
7. Lyakhov, P. High-Performance Digital Filtering on Truncated Multiply-Accumulate Units in the Residue Number System / P. Lyakhov, M. Valueva, G. Valuev, N. Nagornov // IEEE Access. – 2020. – vol. 8. – P. 209181-209190. – DOI: 10.1109/ACCESS.2020.3038496.
8. Nagornov N.N. RNS-Based FPGA Accelerators for High-Quality 3D Medical Image Wavelet Processing Using Scaled Filter Coefficients / N.N. Nagornov, P.A. Lyakhov, M.V. Valueva [et al.] // IEEE Access. – 2022 – Vol. 10 – P. 19215-19231. – DOI: 10.1109/ACCESS.2022.3151361.
9. Valueva, M. Construction of Residue Number System Using Hardware Efficient Diagonal Function / M. Valueva, G. Valuev, N. Semyonova [et al.] // Electronics. – 2019. – Vol. 8. № 6. – 694. – DOI: 10.3390/electronics8060694.
10. Valueva, M. Digital Filter Architecture With Calculations in the Residue Number Sys-tem by Winograd Method F (2×2, 2×2) / M. Valueva, P. Lyakhov, G. Valuev [et al.] // IEEE Access. – 2021. – vol. 9. – P. 143331-143340. – DOI: 10.1109/ACCESS.2021.3121520.
11. Valueva, M. Method for Convolutional Neural Network Hardware Implementation Based on a Residue Number System / M. Valueva, G. Valuev, M. Babenko [et al.] // Programming and Computer Software. – 2022. – Vol. 48, No. 8. – P. 735–744. – DOI: 10.1134/S0361768822080217.
12. Valueva, M.V. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation / M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, N.I. Chervyakov // Mathematics and Computers in Simulation. – 2020. – Vol. 177. – P. 232-243. – DOI: 10.1016/j.matcom.2020.04.031.

Другие публикации автора по теме диссертации

13. Валуева М.В. Архитектуры устройств двумерной фильтрация по методу винограда F(2x2,2x2) с вычислениями в системе остаточных классов // Современная наука и инновации. 2021. № 3 (35). С. 41-52.
14. Валуева М.В. Разработка аппаратной реализации нейросетевого классификатора визуальных образов с использованием вычислений в системе остаточных классов // Сборник тезисов участников форума "Наука будущего – наука молодых". – Нижний Новгород, 2017. –296 С.
15. Ляхов, П.А. Аппаратная реализация устройства обработки видео с использованием системы остаточных классов / П.А. Ляхов, А.С. Ионисян, В.В. Масаева, М.В. Валуева // Современная наука и инновации. – 2021. – №1 (33). – С. 15-21. – DOI: 10.37493/2307-910X.2021.1.2.
16. Ляхов, П.А. Высокопроизводительная цифровая фильтрация на модифицированных умножителях с накоплением в системе остаточных классов с модулями специального вида / П.А. Ляхов, А.С. Ионисян, М.В. Валуева, А.С. Ларикова // Информационные технологии. – 2021. – Т. 27. №4. – С. 171–179. – DOI: 10.17587/it.27.171-179.
17. Ляхов, П.А. Модифицированные умножители с накоплением для повышения производительности цифровых фильтров / П.А. Ляхов, А.С. Ионисян, М.В. Валуева, А.С. Ларикова // Инфокоммуникационные технологии. – 2020. – Т. 18. № 4. – С. 403-410.
18. Ляхов, П.А. Применение сглаживающих фильтров для очистки от шума изображений в оттенках серого / П.А. Ляхов, М.В. Валуева // Наука. Инновации. Технологии. – 2015. – №3. – С. 37-50.
19. Семенова, Н.Ф. Построение системы остаточных классов с диагональной функцией специального вида / Н.Ф. Семенова, Н.И. Червяков, П.А. Ляхов, М.В. Валуева, Г.В. Валуев // Современная наука и инновации. – 2019. – №4 (28). – С. 10-21.
20. Червяков, Н.И. Архитектура сверточной нейронной сети с вычислениями в систе-ме остаточных классов с модулями специального вида / Н.И.

Червяков, П.А. Ля-хов, Д.И. Калита, М.В. Валуева // Нейрокомпьютеры: разработка, применение. – 2017. – №1. – С. 3-15.

21. Червяков, Н.И. Архитектура сверточной нейронной сети с извлечением признаков в системе остаточных классов / Н.И. Червяков, П.А. Ляхов, М.В. Валуева, Н.Н. Нагорнов, Г.В. Валуев // Мягкие измерения и вычисления. – 2019. – №5. – С. 25-36.
22. Червяков, Н.И. Применение сумматоров с параллельно-префиксной архитектурой для перевода чисел из двоичной системы счисления в систему остаточных классов / Н.И. Червяков, П.А. Ляхов, Н.Ф. Семенова, М.В. Валуева // Нейрокомпьютеры: разработка, применение. – 2016. – №10. – С. 31-39.
23. Червяков Н.И., Ляхов П.А., Валуева М.В., Криволапова О.В. Сравнительный анализ аппаратной реализации сумматоров на FPGA // Наука. Инновации. Технологии. – 2016. – №4. – С. 99-108.
24. Chervyakov, N.I. Area-Efficient FPGA Implementation of Minimalistic Convolutional Neural Network Using Residue Number System / N.I. Chervyakov, P.A. Lyakhov, M.V. Valueva [et al.] // 2018 23rd Conference of Open Innovations Association (FRUCT). – 2018. – P. 112-118. – DOI: 10.23919/FRUCT.2018.8588106.
25. Chervyakov, N.I. High-Performance Hardware 3D Medical Imaging using Wavelets in the Residue Number System / N.I. Chervyakov, P.A. Lyakhov, M.V. Valueva [et al.] // 2020 9th Mediterranean Conference on Embedded Computing (MECO). – 2020. – P. 1-4. – DOI: 10.1109/MECO49872.2020.9134123.
26. Chervyakov, N.I. High-Quality 3D Medical Imaging by Wavelet Filters with Reduced Coefficients Bit-Width / N.I. Chervyakov, P.A. Lyakhov, M.V. Valueva [et al.] // 2019 International Conference on Engineering and Telecommunication (EnT). – 2019. – P. 1-5. – DOI: 10.1109/EnT47717.2019.9030532.

27. Chervyakov, N.I. High-speed smoothing filter in the Residue Number System / N. I. Chervyakov, P. A. Lyakhov, M.V. Valueva, [et. al.] // 2016 Third International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC). – 2016. – P. 121-126. – DOI: 10.1109/DIPDMWC.2016.7529375.
28. Chervyakov, N.I. Increasing of Convolutional Neural Network Performance Using Res-idue Number System / N.I. Chervyakov, P.A. Lyakhov, M.V. Valueva. // International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). – 2017. – P. 135–140. – DOI: 10.1109/SIBIRCON.2017.8109855.
29. Chervyakov, N.I. On RNS with VLSI-friendly diagonal function / N.I. Chervyakov, P.A. Lyakhov, M.V. Valueva [et. al.] // 2017 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). – 2017. – P. 131-134. – DOI: 10.1109/SIBIRCON.2017.8109854.
30. Kaplun, D.I. Hardware Implementation of Video Processing Device using Residue Number System / D.I. Kaplun, N.I. Chervyakov, M.V. Valueva [et al.] // 2019 42nd International Conference on Telecommunications and Signal Processing (TSP). – 2019. – P. 701-704. – DOI: 10.1109/TSP.2019.8768827.
31. Lyakhov, P.A. A New Method of Sign Detection in RNS Based on Modified Chinese Remainder Theorem / P.A. Lyakhov, M.V. Valueva, D.I. Kaplun [et al.] // 2021 10th Mediterranean Conference on Embedded Computing (MECO). – 2021. – P. 1-4. – DOI: 10.1109/MECO52532.2021.9460255.
32. Lyakhov, P.A. Low-Bit Hardware Implementation of DWT for 3D Medical Images Processing / P.A. Lyakhov, M.V. Valueva, N.N. Nagornov // 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – 2020. – P. 1396-1399. – DOI: 10.1109/EIConRus49466.2020.9038946.
33. Lyakhov, P.A. Single Image Super-Resolution Method Based on Bilinear Interpolation and U-Net Combination / P.A. Lyakhov, G. V. Valuev, M. V. Valueva [et al.] // 2021 10th Mediterranean Conference on Embedded

Computing (MECO). – 2021. – P. 1-4. – DOI: 10.1109/MECO52532.2021.9460201.

Свидетельства о государственной регистрации программ для ЭВМ

34. Свидетельство о государственной регистрации программы для ЭВМ № 2017617504 Российская Федерация. Среда моделирования распознавания изображений с использованием сверточных нейронных сетей / Червяков Н.И., Ляхов П.А., Валуева М.В., Калита Д.И.; заявитель и правообладатель ФГАОУ ВО «Северо-Кавказский федеральный университет». № 2017614396; заявл. 12.05.2017; опубл. 05.07.2017 – 1 с.
35. Свидетельство о государственной регистрации программы для ЭВМ № 2018613585 Российская Федерация. Программа реализации метода сглаживающей фильтрации изображений в СОК / Червяков Н.И., Ляхов П.А., Нагорнов Н.Н., Валуева М.В.; заявитель и правообладатель ФГАОУ ВО «Северо-Кавказский федеральный университет». № 2018610959; заявл. 02.02.2018; опубл. 19.03.2018 – 1 с.
36. Свидетельство о государственной регистрации программы для ЭВМ № 2019663305 Российская Федерация. Среда поиска лиц на фотографии / Червяков Н.И., Ляхов П.А., Валуева М.В., Валуев Г.В.; заявитель и правообладатель ФГАОУ ВО «Северо-Кавказский федеральный университет». № 2019662300; заявл. 09.10.2019; опубл. 15.10.2019 – 1 с.
37. Свидетельство о государственной регистрации программы для ЭВМ № 2020612962 Российская Федерация. Среда аппаратного моделирования дискретного вейвлет-преобразования в системе остаточных классов для медицинской визуализации / Червяков Н.И., Ляхов П.А., Валуева М.В., Нагорнов Н.Н.; заявитель и правообладатель ФГАОУ ВО «Северо-Кавказский федеральный университет». № 2020611770; заявл. 25.02.2020; опубл. 06.03.2020 – 1 с.

38. Свидетельство о государственной регистрации программы для ЭВМ № 2020612961 Российская Федерация. Среда аппаратного моделирования дискретного вейвлет-преобразования с квантованными коэффициентами для обработки медицинских изображений / Червяков Н.И., Ляхов П.А., Валуева М.В., Нагорнов Н.Н., Бергерман М.В., Валуев Г.В.; заявитель и правообладатель ФГАОУ ВО «Северо-Кавказский федеральный университет». № 2020611741; заявл. 25.02.2020; опубл. 06.03.2020 – 1 с.
39. Свидетельство о государственной регистрации программы для ЭВМ № 2021619007 Российская Федерация. Среда аппаратного моделирования фильтрации изображений по методу винограда / Валуева М.В., Ляхов П.А., Нагорнов Н.Н., Валуев Г.В.; заявитель и правообладатель ФГАОУ ВО «Северо-Кавказский федеральный университет». № 2021618215; заявл. 31.05.2021; опубл. 03.06.2021 – 1 с.

Цитируемая литература

40. Abadi, M. {TensorFlow}: a system for {Large-Scale} machine learning / M. Abadi, P. Barham, J. Chen, [et al.] // In 12th USENIX symposium on operating systems design and implementation (OSDI 16). – 2016. – P. 265-283.
41. Aghdam, H.H. Guide to Convolutional Neural Networks / H.H. Aghdam, E.J. Heravi. – Springer International Publishing, Cham. – 2017. – 282 P. – DOI:10.1007/978-3-319-57550-6.
42. Ahn, S. Soft Memory Box: A Virtual Shared Memory Framework for Fast Deep Neural Network Training in Distributed High Performance Computing / S. Ahn, J. Kim, E. Lim, S. Kang // IEEE Access. – 2018. – Vol. 6. – P. 26493-26504. – DOI: 10.1109/ACCESS.2018.2834146.
43. Belghadr, A. FIR Filter Realization via Deferred End-Around Carry Modular Addition / A. Belghadr, G. Jaberipur // IEEE Transactions on Circuits and Systems I: Regular Papers. – 2018. – Vol. 65, no. 9. – P. 2878-2888. – DOI: 10.1109/TCSI.2018.2798595.

44. Bi, S. The Mixed-Radix Chinese Remainder Theorem and Its Applications to Residue Comparison / S. Bi, W.J. Gross // IEEE Transactions on Computers. – 2008. – Vol. 57, no. 12. – P. 1624-1632. – DOI: 10.1109/TC.2008.126.
45. Simonyan, K. Very deep convolutional networks for large-scale image recognition / K. Simonyan, A. Zisserman // 3rd International Conference on Learning Representations (ICLR 2015). – 2015. – P. 1-14.
46. Bovik, A.C. Handbook of image and video processing, 2nd ed. / A.C. Bovik. – San Diego, Elsevier Academic Press. – 2005. – 1384 p.
47. Cardarilli, G.C. Residue number system for low-power DSP applications / G.C. Cardarilli, A. Nannarelli, M. Re // Proc. 41st Asilomar Conf. Signals, Syst., Comput. – 2007. – P. 1412 - 1416.
48. Cardarilli, G.C. Design Space Exploration based Methodology for Residue Number System Digital Filters Im-plementation / G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, [et. al.] // IEEE Transactions on Emerging Topics in Computing. – P. 186 – 198. – DOI: 10.1109/TETC.2020.2997067.
49. Chervyakov, N.I. Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem / N.I. Chervyakov, A.S. Molahosseini, P.A. Lyakhov, [et al.] // Int. J. Comput. Math. – 2017. – Vol. 94, № 9. – P. 1833-1849. – DOI: 10.1080/00207160.2016.1247439.
50. Chervyakov, N.I. Implementation of Smoothing Image Filtering in the Residue Number System / N.I. Chervyakov, P.A. Lyakhov, N.N. Nagornov, [et al.] // 2019 8th Mediterranean Conference on Embedded Computing (MECO). – 2019. – P. 1-4. – DOI: 10.1109/MECO.2019.8760190.
51. Chervyakov, N. AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security / N. Chervyakov, M. Babenko, A. Tchernykh, [et al.] // Future Generation Computer Systems. – 2019. – Vol. 92. – P. 1080-1092. – DOI: 10.1016/j.future.2017.09.061.
52. DiCecco, R. Caffeinated FPGAs: FPGA framework for convolutional neural networks / R. DiCecco, G. Lacey, R. DiCecco, [et al.] // 2016 International

Conference on Field-Programmable Technology (FPT). – 2016. – P. 265-268. – DOI: 10.1109/FPT.2016.7929549.

53. DiCecco, R. FPGA-based training of convolutional neural networks with a reduced precision floating-point library / R. DiCecco, L. Sun, P. Chow // 2017 International Conference on Field Programmable Technology (ICFPT). – Melbourne, 2017. – P. 239-242. – DOI: 10.1109/FPT.2017.8280150.
54. Dlugosz, Z. Nonlinear Activation Functions for Artificial Neural Networks Realized in Hardware / Z. Dlugosz, R. Dlugosz //25th International Conference "Mixed Design of Integrated Circuits and System" (MIXDES). – 2018. – P. 381-384. – DOI: 10.23919/MIXDES.2018.8436869.
55. Eyeriss [электронный ресурс]: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. URL: http://eyeriss.mit.edu/ (дата обращения: 07.08.2022).
56. Fan, J. Human Tracking Using Convolutional Neural Networks / J. Fan, W. Xu, Y. Wu // IEEE Transactions on Neural Networks. – 2010. – Vol. 21, no. 10. – P.1610 -1623. – DOI: 10.1109/TNN.2010.2066286.
57. Farabet, C. Hardware accelerated convolutional neural networks for synthetic vision systems / C. Farabet, B. Martini, P. Akselrod, [et al.] // Int'l Symp. on Circuits and Systems (ISCAS2010). – 2010. – P.257-260. – DOI: 10.1109/ISCAS.2010.5537908.
58. Gbolagade, K.A. An O(n) Residue Number System to Mixed Radix Conversion technique / K.A. Gbolagade, S.D. Cotofana // 2009 IEEE International Symposium on Circuits and Systems. – 2009. – P. 521-524. – DOI: 10.1109/ISCAS.2009.5117800.
59. Geng, H. Connected and autonomous vehicles / H. Geng // Internet of Things and Data Analytics Handbook. – Wiley, 2017. – P. 581-595. – DOI: 10.1002/9781119173601.ch35.
60. Gong, L. MALOC: A Fully Pipelined FPGA Accelerator for Convolutional Neural Networks with All Layers Mapped on Chip / Gong L., Wang C., Li X., [et al.] // IEEE Transactions on Computer-Aided Design of Integrated Circuits

and Systems. – 2018. –Vol. 37, no. 11. – P. 2601-2612. – DOI: 10.1109/TCAD.2018.2857078.

61. Haykin, S.S. Neural networks: a comprehensive foundation / S.S. Haykin. – Prentice Hall, 1999.
62. Intel HLS Compiler [электронный ресурс]: URL: https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/hls-compiler.html (дата обращения: 08.03.2021).
63. Intel Movidius Myriad X VPU [электронный ресурс]: URL: https://www.movidius.com/myriadx (дата обращения: 08.03.2021).
64. Intel Nervana NNP-I [электронный ресурс]: URL: https://www.intel.ai/benchmarks/#gs.w90vlg (дата обращения: 08.03.2021).
65. Intel Nervana NNP-T [электронный ресурс]: URL: https://www.intel.ai/nervana-nnp/nnpt/#gs.w8jhu3 (дата обращения: 08.03.2021).
66. Intel OpenVINO [электронный ресурс]: Install Intel Distribution of OpenVINO toolkit for Linux with FPGA Support. URL: https://docs.openvinotoolkit.org/2019_R1/_docs_install_guides_installing_openvino_linux_fpga.html (дата обращения: 07.08.2022).
67. Jia, Y. Caffe: Convolutional architecture for fast feature embedding / Jia, Y., Shelhamer, E., Donahue, [et al.] // Proceedings of the 22nd ACM international conference on Multimedia. – 2014. – P. 675-678. – DOI: 10.1145/2647868.2654889.
68. Jouppi, N. Motivation for and Evaluation of the First Tensor Processing Unit / N. Jouppi, C. Young, N. Patil, [et al.] // IEEE Micro. – 2018. – Vol. 38, no. 3. – P. 10-19. – DOI: 10.1109/MM.2018.032271057.
69. Kogge, P.M. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations / P.M. Kogge, H.S. Stone // IEEE Trans. Comput. – 1973. – Vol. C–22, no. 8. – P. 786-793. – DOI: 10.1109/TC.1973.5009159.
70. Krizhevsky, A. Learning multiple layers of features from tiny images / A. Krizhevsky, G. Hinton. – Toronto: University of Toronto. – 2009.

71. Krizhevsky, A. ImageNet Classification with Deep Convolutional Neural Networks / A. Krizhevsky, I. Sutskever, G.E. Hinton. // Advances in neural information processing systems. – 2012. – Vol.25, no. 2.
72. Lavin, A. Fast Algorithms for Convolutional Neural Net-works / A. Lavin, S. Gray // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016. – P. 4013-4021. – DOI: 10.1109/CVPR.2016.435.
73. LeCun, Y. Gradient-based learning applied to document recognition / Y. LeCun, L. Bottou, Y. Bengio, [et al.] // Proc. of the IEEE. – 1998. – Vol.86, no. 11. – P.2278-2324.
74. Lin, Y. Data and Hardware Efficient Design for Convolutional Neural Network / Y. Lin, T.S. Chang // IEEE Transactions on Circuits and Systems I: Regular Papers. –2018. – Vol. 65, no. 5. – P. 1642-1651. – DOI: 10.1109/TCSI.2017.2759803.
75. Luo, X. A novel hidden danger prediction method in cloud-based intelligent industrial production management using timeliness managing extreme learning machine / X. Luo, X. Yang, W. Wang, [et. al.] // China Communications. – 2016. – Vol. 13, no. 7. – P. 74-82. – DOI: 10.1109/CC.2016.7559078.
76. MATLAB [электронный ресурс]: URL: https://matlab.ru/products/matlab (дата обращения: 07.08.2022).
77. Matos R. Efficient implementation of modular multiplication by constants applied to RNS reverse converters / R. de Matos, R. Paludo, N. Chervyakov, [et al.] // 2017 IEEE International Symposium on Circuits and Systems (ISCAS). – 2017. – P. 1-4. – DOI: 10.1109/ISCAS.2017.8050779.
78. Mohan, P.V.A. RNS-to-Binary Converters for Two Four-Moduli Sets $\{2^{n}-1,2^{n},2^{n}+1,2^{{n}+1}-1\}$ and $\{2^{n}-1,2^{n},2^{n}+1,2^{{n}+1}+1\}$ / P.V.A. Mohan, A.B. Premkumar // IEEE Transactions on Circuits and Systems I: Regular Papers. – 2007. – Vol. 54, no. 6. – P. 1245-1254. – DOI: 10.1109/TCSI.2007.895515.
79. Mohan, P.V.A. Residue Number Systems: Theory and Applications / P.V.A. Mohan. – Basel: Birkhauser. – 2016.

80. Molahosseini, A.S. Embedded systems design with special arithmetic and number systems / A.S. Molahosseini, L.S. De Sousa, C.H. Chang (eds.). – Springer International Publishing, 2017. – 390 p.
81. Moreno, F. Reconfigurable Hardware Architecture of a Shape Recognition System Based on Specialized Tiny Neural Networks with Online Training / F. Moreno, J. Alarcon, R. Salvador, [et al.] // IEEE Transactions on Industrial Electronics. – 2009. –vol. 56, no. 8. – P. 3253-3263. DOI: 10.1109/TIE.2009.2022076
82. MyHDL [электронный ресурс]: From Python to Silicon! URL: http://www.myhdl.org/ (дата обращения: 07.08.2022).
83. Nakahara, H. A deep convolutional neural network based on nested residue number system / H. Nakahara, T. Sasao // 2015 25th International Conference on Field Programmable Logic and Applications (FPL). – 2015. – P.1-6. – DOI: 10.1109/FPL.2015.7293933.
84. Nakahara, H. A High-speed Low-power Deep Neural Network on an FPGA based on the Nested RNS: Applied to an Object Detector / H. Nakahara, T. Sasao // 2018 IEEE Int. Symp. Circuits Syst. – 2018. – P. 1–5. – DOI:10.1109/ISCAS.2018.8351850.
85. Nakasone, S. An OpenCL Implementation of an Image Filter on FPGA / S. Nakasone, L. Öfverstedt, G. Wilken, U. Skoglund // 2019 IEEE 5th International Conference on Computer and Communications (ICCC). – 2019. – P. 272-276. – DOI: 10.1109/ICCC47050.2019.9064160.
86. Noronha, D.H. LeFlow: Enabling Flexible FPGA High-Level Synthesis of Tensorflow Deep Neural Networks / D.H. Noronha, B. Salehpour, S.J.E. Wilton // FSP Workshop 2018; Fifth International Workshop on FPGAs for Software Programmers. – 2018. – P. 1-8.
87. Omondi, A. Residue Number Systems: Theory and Implementation / A. Omondi, B. Premkumar. – London: Imperial College Press. – 2007. – 296 p.
88. Parhami, B. Computer arithmetic: algorithms and hardware designs / Parhami, B. – Oxford University Press. – 2010. – 492 p.

89. Peemen, M. Memory centric accelerator design for convolutional neural networks / M. Peemen, A.A.A. Setio, B. Mesman, [et al.] // 31st International Conference on Computer Design (ICCD2013). – 2013. – P. 13-19. – DOI: 10.1109/ICCD.2013.6657019.

90. Rezvani, R. A new method for hardware design of Multi-Layer Perceptron neural networks with online training / Rezvani R., Katiraee M., Jamalian A.H., [et al.] // 2012 IEEE 11th International Conference on Cognitive Informatics and Cognitive Computing. – 2012. – P. 527-534. – DOI: 10.1109/ICCI-CC.2012.6311205.

91. Salamat, S. RNSnet: In-Memory Neural Network Acceleration Using Residue Number System / Salamat S., Imani M., Gupta S., [et al.] // 2018 IEEE International Conference on Rebooting Computing (ICRC). – 2018. – P. 1-12. – DOI: 10.1109/ICRC.2018.8638592.

92. Sameh, A. Generic floating point library for neuro-fuzzy controllers based on FPGA technology / A. Sameh, M.S. A.A. El Kader // Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology. – 2004. – P. 369-372. – DOI: 10.1109/ISSPIT.2004.1433796.

93. Samuel, A. RNS-based elliptic curve point multiplication for massive parallel architectures / A. Samuel, J.-C. Bajard, L. Sousa // The Computer Journal. – 2012. – Vol. 55, no. 5. – P. 629-647. – DOI: 10.1093/comjnl/bxr119.

94. Sang, R. FPGA-based acceleration of neural network training / R. Sang, Q. Liu, Q. Zhang // 2016 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO). – 2016. – P. 1-2. – DOI: 10.1109/NEMO.2016.7561676.

95. Sankaradas, M. A massively parallel coprocessor for convolutional neural networks / M. Sankaradas, V. Jakkula, S. Gadami, [et al.] // 20th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP2009). – 2009. – P. 53-60. – DOI: 10.1109/ASAP.2009.25.

96. Shawahna, A. FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review / A. Shawahna, S.M. Sait, A. El-Maleh

// IEEE Access. – 2019. – Vol. 7. – P. 7823-7859. – DOI: 10.1109/ACCESS.2018.2890150.

97. Sze, V. Efficient Processing of Deep Neural Networks: A Tutorial and Survey / V. Sze, Y. Chen, T. Yang, [et al.] // Proceedings of the IEEE. – 2017. – Vol. 105, no. 12. – P. 2295-2329. – DOI: 10.1109/JPROC.2017.2761740.

98. Szegedy, C. Going deeper with convolutions / C. Szegedy, W. Liu, Y. Jia, [et al.] // 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2015. – P. 1–9. – DOI: 10.1109/CVPR.2015.7298594.

99. Tay, A.L.P. The Hierarchical Fast Learning Artificial Neural Network (HieFLANN) – An Autonomous Platform for Hierarchical Neural Network Construction / Tay A.L.P., Zurada J.M., Wong L., [et al.] // IEEE Transactions on Neural Networks. – 2007. – Vol. 18, no. 6. – P. 1645-1657. – DOI: 10.1109/TNN.2007.900231.

100. Tsai, C. A Hardware-Efficient Sigmoid Function With Adjustable Precision for a Neural Network System / Tsai C., Chih Y., Wong W.H., Lee C. // IEEE Transactions on Circuits and Systems II: Express Briefs. – 2015. – Vol. 62, no. 11. – P. 1073-1077. – DOI: 10.1109/TCSII.2015.2456531.

101. Tung, C. A High-Performance Multiply-Accumulate Unit by Integrating Additions and Accumulations Into Partial Product Reduction Process / C. Tung, S. Huang // IEEE Access. – 2020. – Vol. 8. – P. 87367-87377. – DOI: 10.1109/ACCESS.2020.2992286.

102. Vassalos, E. RNS assisted image filtering and edge detection / E. Vassalos, D. Bakalis, H.T. Vergos // 2013 18th International Conference on Digital Signal Processing (DSP). – 2013. – P. 1-6. – DOI: 10.1109/ICDSP.2013.6622821.

103. Vergos, H.T. On Modulo 2^n+1 Adder Design / H.T. Vergos, G. Dimitrakopoulos // IEEE Transactions on Computers. – 2012. – Vol. 61, no. 2. – P. 173-186. – DOI: 10.1109/TC.2010.261.

104. Vitis AI [электронный ресурс]: Adaptable and Real-Time AI Inference Acceleration. URL: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html (дата обращения: 07.08.2022).

105. Vivado HLS [электронный ресурс]: Vitis Model Composer URL: https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html (дата обращения: 07.08.2022).

106. Wang, J. Efficient convolution architectures for convolutional neural network / J. Wang, J. Lin, Z. Wang // 2016 8th Int. Conf. Wirel. Commun. Signal Process. – 2016. – P. 1–5. – DOI:10.1109/WCSP.2016.7752726.

107. Wang, J. Efficient Hardware Architectures for Deep Convolutional Neural Network / J. Wang, J. Lin, Z. Wang // IEEE Trans. Circuits Syst. I Regul. Pap. – 2018. – Vol. 65. – P. 1941-1953. – DOI:10.1109/TCSI.2017.2767204.

108. Winograd, S. Arithmetic complexity of computations / S. Winograd. – Siam, 1980. – Vol. 33. – 93. P. – DOI: 10.1137/1.9781611970364.

109. Xiao, H. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms / H. Xiao, R. Kashif, R. Vollgraf // 2017. – arXiv preprint arXiv:1708.07747.

110. Xilinx Deep Neural Network [электронный ресурс]: Accelerating DNNs with Xilinx Alveo Accelerator Cards. URL: https://www.xilinx.com/support/documentation/white_papers/wp504-accel-dnns.pdf (дата обращения: 07.08.2022).

111. Yepez, J. Stride 2 1-D, 2-D, and 3-D Winograd for Convolutional Neural Networks / J. Yepez, S. Ko // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – 2020. – Vol. 28, no. 4. – P. 853-863. – DOI: 10.1109/TVLSI.2019.2961602.

112. Yuan, B. Efficient hardware architecture of softmax layer in deep neural network / B. Yuan // 2016 29th IEEE International System-on-Chip Conference (SOCC). – 2016. P. 323-326. – DOI: 10.1109/SOCC.2016.7905501.

113. Zimmermann, R. Binary adder architectures for cell-based VLSI and their synthesis. / R. Zimmermann. – Hartung-Gorre. – 1998. – 205 P.

114. Živaljević, D. Digital filter implementation based on the RNS with diminished-1 encoded channel / D. Živaljević, N. Stamenković, V. Stojanović // 2012 35th International Conference on Telecommunications and Signal

Processing (TSP). – Prague, 2012. – P. 662-666. – DOI: 10.1109/TSP.2012.6256380.

СПИСОК РИСУНКОВ

1.1.1. Процедура получения карт признаков

1.1.2. Процедура выбора максимальных элементов для карт признаков

2.1.1. Фильтрация изображения

2.1.2. Фильтрация изображения в СОК

2.2.1. Архитектура α-разрядного устройства FIR порядка P

2.2.2. Архитектура α-разрядного устройства умножения с накоплением MAC

2.2.3. Архитектура α-разрядного устройства MOA

2.3.1. Этапы фильтрации фрагмента изображения по методу Винограда

2.3.2. Архитектура устройства $\mathrm{F}(n \times n, k \times k)_{\mathrm{RNS}}$ для двумерной фильтрации по методу Винограда в СОК с модулями вида 2^{α} и $2^{\alpha} - 1$

2.4.1. Процесс фильтрации изображения по методу Винограда $F(2 \times 2, 2 \times 2)$

2.4.2. Этапы фильтрации фрагмента изображения по методу Винограда $F(2 \times 2, 2 \times 2)$

2.4.3. Архитектура сумматора по модулю с множественным входом MOMA: а) по модулю 2^{α} ($\mathrm{MOMA}_{2^{\alpha}}$); б) по модулю $2^{\alpha} - 1$ ($\mathrm{MOMA}_{2^{\alpha}-1}$)

2.4.4. Архитектура умножителя MUL: а) по модулю 2^{α} ($\mathrm{MUL}_{2^{\alpha}}$); б) по модулю $2^{\alpha} - 1$ ($\mathrm{MUL}_{2^{\alpha}-1}$)

2.4.5. Архитектура устройства $\mathrm{F}(2 \times 2, 2 \times 2)_{2^{\alpha}}$ для двумерной фильтрации по методу Винограда по модулю 2^{α}

2.4.6. Архитектура устройства $\mathrm{F}(2 \times 2, 2 \times 2)_{2^{\alpha}-1}$ для двумерной фильтрации по методу Винограда по модулю $2^{\alpha} - 1$

2.4.7. Архитектура устройства $\mathrm{F}(2 \times 2, 2 \times 2)_{\mathrm{RNS}}$ для двумерной фильтрации по методу Винограда в СОК с модулями вида 2^{α} и $2^{\alpha} - 1$

2.5.1. Процесс фильтрации изображения по методу Винограда $F(2 \times 2, 3 \times 3)$

2.5.2. Этапы фильтрации фрагмента изображения по методу Винограда $F(2 \times 2, 3 \times 3)$

2.5.3. Архитектура устройства FTR для вычисления i-ой строки матрицы Z: а) по модулю 2^{α}; б) по модулю $2^{\alpha} - 1$

2.5.4. Архитектура устройства для двумерной фильтрации с использованием модифицированного метода Винограда $F(2 \times 2, 3 \times 3)$ по модулю 2^{α}

2.5.5. Архитектура устройства для двумерной фильтрации с использованием модифицированного метода Винограда $F(2 \times 2, 3 \times 3)$ по модулю $2^{\alpha} - 1$

2.5.6. Архитектура устройства $\mathrm{F}(2 \times 2, 3 \times 3)_{\mathrm{RNS}}$ для двумерной фильтрации по методу Винограда в СОК с модулями вида 2^{α} и $2^{\alpha} - 1$

2.6.1. Процесс фильтрации изображения по методу Винограда $F(2 \times 2, 5 \times 5)$

2.6.2. Этапы фильтрации фрагмента изображения по методу Винограда $F(2 \times 2, 5 \times 5)$

2.6.3. Архитектура устройства преобразования данных DTE: а) по модулю 2^{α}; б) по модулю $2^{\alpha} - 1$

2.6.4. Архитектура устройства $\mathrm{DTR}_{2^{\alpha}}$ для вычисления элементов i-ой строки матрицы V по модулю 2^{α}

2.6.5. Архитектура устройства $\mathrm{DTR}_{2^{\alpha}-1}$ для вычисления элементов i-ой строки матрицы V по модулю $2^{\alpha} - 1$

2.6.6. Архитектура устройства $\mathrm{FTR}_{2^{\alpha}}$ для вычисления элементов i-ой строки матрицы Z по модулю 2^{α}

2.6.7. Архитектура устройства $\mathrm{FTR}_{2^{\alpha}-1}$ для вычисления элементов i-ой строки матрицы Z по модулю $2^{\alpha} - 1$

2.6.8. Архитектура устройства двумерной фильтрации $\mathrm{F}(2 \times 2, 5 \times 5)_{2^{\alpha}}$ по методу Винограда с вычислениями по модулю 2^{α}

2.6.9. Архитектура устройство двумерной фильтрации $\mathrm{F}(2\times 2,5\times 5)_{2^\alpha-1}$ по методу Винограда с вычислениями по модулю $2^\alpha-1$

2.6.10. Архитектура устройства $\mathrm{F}(2\times 2,5\times 5)_{\mathrm{RNS}}$ для двумерной фильтрации по методу Винограда в СОК с модулями вида 2^α и $2^\alpha-1$

3.1.1. Архитектура устройства $\mathrm{MOD}_{2^\alpha-1}$ для вычисления остатка от деления по модулю $2^\alpha-1$

3.2.1. Расположение положительных и отрицательных чисел в СОК

3.2.2. Архитектура устройства для вычисления позиционной характеристики числа, представленного в СОК, с помощью КТОд

3.2.3. Архитектура компаратора COMP

3.3.1. Архитектура устройства ReLU вычисления функции активации ReLU в СОК

3.4.1. Архитектура компаратора с тремя выходами

3.4.2. Архитектура устройства $\mathrm{COMP}_{\mathrm{CRTf}}$ для сравнения двух чисел в СОК

3.5.1. Архитектура устройства вычисления большего из чисел в СОК

3.5.2. Архитектура устройства MAX вычисления большего из чисел в СОК

3.6.1. Архитектура устройства $\mathrm{RC}_{\mathrm{CRTf}}$ обратного преобразования из СОК в ПСС на основе КТОд

4.1.1. Архитектура устройства $\mathrm{PNS}\rightarrow\mathrm{RNS}$ преобразования чисел из ПСС в СОК с модулями вида $\{2^{\alpha_1},2^{\alpha_2}-1,\ldots,2^{\alpha_n}-1\}$

4.1.2. Архитектура устройства CONV_{2^α} для свертки по модулю 2^α фрагмента карты признаков

4.1.3. Архитектура устройства FC_{2^α} полносвязного слоя с вычислениями по модулю 2^α

4.1.4. Архитектура устройства для выбора максимального из четырех чисел в СОК

4.2.1. Предлагаемая архитектура аппаратной реализации СНС с вычислениями в СОК

4.3.1. Структура комплекса программ для аппаратной реализации СНС

СПИСОК ТАБЛИЦ

2.4.1. Модули СОК

2.4.2. Теоретические параметры двумерных фильтров с маской 2×2

2.4.3. Результаты аппаратного моделирования двумерных фильтров с маской 2×2

2.5.1. Теоретические параметры двумерных фильтров с маской 3×3

2.5.2. Результаты аппаратного моделирования двумерных фильтров с маской 3×3

2.6.1. Теоретические параметры двумерных фильтров с маской размерности 5×5

2.6.2. Результаты аппаратного моделирования двумерных фильтров с маской размерности 5×5

3.1.1. Теоретические параметры устройств для вычисления остатка от деления 16-битного числа по модулю $2^{\alpha} - 1$

3.1.2. Результаты аппаратного моделирования устройств для вычисления остатка от деления 16-битного числа по модулю $2^{\alpha} - 1$

3.2.1. Теоретические параметры площади и задержки устройств определения знака числа в СОК

3.2.2. Результаты аппаратного моделирования операции определения знака числа в СОК

3.3.1. Теоретические параметры площади и задержки устройств сравнения чисел в СОК

3.3.2. Результаты аппаратного моделирования устройств сравнения чисел в СОК

3.4.1. Теоретические параметры площади и задержки устройств обратного преобразования чисел из СОК в ПСС

3.4.2. Результаты моделирования устройств обратного преобразования из СОК в ПСС

4.3.1. Параметры архитектур СНС LeNet-5

4.3.2. Результаты работы СНС LeNet-5 с различной разрядностью весовых коэффициентов

4.3.3. Конфигурации архитектур СНС

4.3.4. Результаты аппаратного моделирования СНС LeNet-5

ПРИЛОЖЕНИЕ А. Акт о внедрении результатов диссертационного исследования

Infocom
25.05.23 № 117
На №_____от_____

Общество с ограниченной ответственностью
«Инфоком-С»
355035, г. Ставрополь, ул. Суворова, 7
info@infocom-s.ru www.infocom-s.ru
+7 (8652) 20-58-20
ИНН: 2635811319 ОКПО: 38852042
КПП: 263501001 ОГРН: 1122651011559
+7 (495) 700-00-65

Акт

о внедрении результатов диссертационного исследования Валуевой М.В. на тему «Разработка методов и алгоритмов построения цифровых устройств интеллектуального анализа визуальных данных».

Комиссия в составе:

председатель – генеральный директор Д.Т.Н., профессор Копытов В.В.

члены комиссии – директор по развитию к.т.н. доцент Шульгин А.О., директор по проектам к.т.н. доцент Демурчев Н.Г., руководитель департамента интеллектуальных платформ Касимов Р.И..

Составили настоящий акт о том, что результаты диссертационного исследования Валуевой М.В. на тему: «Разработка методов и алгоритмов построения цифровых устройств интеллектуального анализа визуальных данных», включающей такие результаты как:

1. Алгоритм проектирования аппаратной реализации сверточных нейронных сетей с вычислениями в системе остаточных классов, учитывающий выбор модулей системы, способ представления весовых коэффициентов в памяти устройства и особенности проектирования компонентов сверточной нейронной сети в системе остаточных классов;
2. Комплекс программ на языке описания аппаратуры VHDL для аппаратной реализации сверточной нейронной сети с вычислениями в системе остаточных классов на программируемых логических интегральных схемах;

использованы при выполнении НИОКР «Доработка кроссплатформенной PSIM-системы Darvis для обеспечения безопасности промышленных предприятий» по гранту Российского фонда развития информационных технологий (РФРИТ) на разработку отечественных ИТ-решений в рамках реализации федерального проекта «Цифровые технологии» национальной программы «Цифровая экономика Российской Федерации».

Председатель комиссии: _____ Копытов В.В.

Члены комиссии: _____ Демурчев Н.Г.

_____ Касимов Р.И.

ПРИЛОЖЕНИЕ Б. Свидетельства о государственной регистрации программ для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2017617504

Среда моделирования распознавания изображений с использованием сверточных нейронных сетей

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Червяков Николай Иванович (RU), Ляхов Павел Алексеевич (RU), Валуева Мария Васильевна (RU), Калита Диана Ивановна (RU)***

Заявка № **2017614396**
Дата поступления **12 мая 2017 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***05 июля 2017 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2018613585

Программа реализации метода сглаживающей фильтрации изображений в СОК

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Червяков Николай Иванович (RU), Ляхов Павел Алексеевич (RU), Нагорнов Николай Николаевич (RU), Валуева Мария Васильевна (RU)***

Заявка № **2018610959**
Дата поступления **02 февраля 2018 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***19 марта 2018 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2019663305

Среда поиска лиц на фотографии

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Червяков Николай Иванович (RU), Ляхов Павел Алексеевич (RU), Валуева Мария Васильевна (RU), Валуев Георгий Вячеславович (RU)***

Заявка № **2019662300**
Дата поступления **09 октября 2019 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***15 октября 2019 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2020612961

Среда аппаратного моделирования дискретного вейвлет-преобразования с квантованными коэффициентами для обработки медицинских изображений

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Червяков Николай Иванович (RU), Ляхов Павел Алексеевич (RU), Валуева Мария Васильевна (RU), Нагорнов Николай Николаевич (RU), Бергерман Максим Валерьевич (RU), Валуев Георгий Вячеславович (RU)***

Заявка № **2020611741**

Дата поступления **25 февраля 2020 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***06 марта 2020 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2020612962

Среда аппаратного моделирования дискретного вейвлет-преобразования в системе остаточных классов для медицинской визуализации

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Червяков Николай Иванович (RU), Ляхов Павел Алексеевич (RU), Валуева Мария Васильевна (RU), Нагорнов Николай Николаевич (RU)***

Заявка № **2020611770**
Дата поступления **25 февраля 2020 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***06 марта 2020 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2021619007

Среда аппаратного моделирования фильтрации изображений по методу Винограда

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Валуева Мария Васильевна (RU), Ляхов Павел Алексеевич (RU), Нагорнов Николай Николаевич (RU), Валуев Георгий Вячеславович (RU)***

Заявка № **2021618215**
Дата поступления **31 мая 2021 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***03 июня 2021 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0x02A5CFBC00B1ACF59A40A2F08092E9A118
Владелец **Ивлиев Григорий Петрович**
Действителен с 15.01.2021 по 15.01.2035

Г.П. Ивлиев