# КВАТЕРНИОННАЯ НЕЙРОННАЯ СЕТЬ НА БАЗЕ FPGA

*Осипов А. С, студент гр.150701*

*Белорусский государственный университет информатики и радиоэлектроники*
*г. Минск, Республика Беларусь*

*Петровский Н.А. – канд. техн. наук*

В работе представлена реализация нейронной сети на базе FPGA, использующую алгебру кватернионов для классификации трёхкомпонентных цветных изображений. Структура нейронной сети состоит из полносвязных слоев. В работе представлены результаты симуляции и синтеза модели, составленной на языке System Verilog, а также варианты развития проекта.

**Введение**

Использование алгебры кватернионов для реализации нейронных сетей приносит существенное преимущество при работе с цветными изображениями. Преимущество достигается за счет более тесной связи между каналами одного пикселя. Так, устройство автокодировщика [1], использующую алгебру кватернионов, показывает лучшие результаты восстановления изображений при большем сжатии по сравнению со стандартной реализации.

В данной статье представлен результат реализации полносвязной нейронной сети на базе *FPGA*. Целевой задачей нейронной сети была выбрана классификация изображений с перспективой модернизации *IP*-блока для решения разнообразных задач.

**Структура нейронной сети и процесс обучения**

В качестве обучающей выборки для нейронной сети использовался датасет *CIFAR-10* [2], содержащий 60000 цветных изображений размером 32x32 пикселя, распределенных по десяти классам.

Перед подачей данных на вход кватернионной нейронной сети, значения пикселей, представленные в диапазоне $[0, 255]$, нормализуются к диапазону $[0, 1)$ путем деления на значение 256. Опционально диапазон значений может быть сдвинут в область $[-0.5, 0.5)$, что также влияет на результаты обучения нейронной сети и в последствии используется в качестве гиперпараметра. При представлении данных в виде кватернионов каждому пикселю соответствует свое значение при мнимых единицах кватерниона. Значение при действительной части приравнивается нулю.

В данной работе представлена реализация кватернионной нейронной сети, основанной на полносвязных слоях. Выходом нейронной сети является номер класса, представленный в унитарном коде в составе кватернионов. Так, для представления десяти классов требуется три кватерниона, два из которых не будут иметь полезных данных при мнимой единице *k*.

На рисунке 1 представлен график обучения нейронной сети. Максимальное значение предсказания на тренировочном наборе составляет 56.91%. Используя текущую структуру сети, состоящую исключительно из полносвязных слоев, достичь лучшего значения предсказания не представляется возможным. В связи с этим далее в статье описаны варианты по ее модернизации, влияющие, как только на процесс обучения (использование *dropout* слоев), так и на функционал итогового *IP*-ядра (использование сверточных слоев).

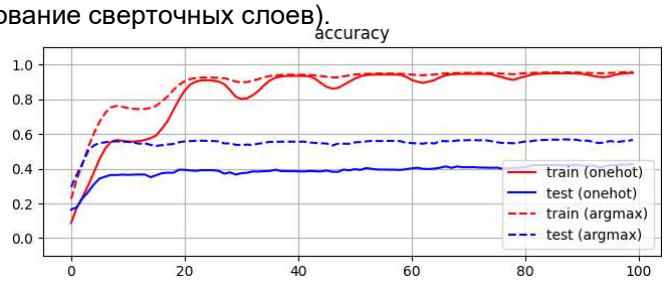



Рисунок 1 – График метрик тренировочного и тестового наборов при обучении нерейронной сети

**Реализация IP-ядра**

Разработанное *IP*-ядро реализовано на языке *SystemVerilog* и имеет два интерфейса *AXI-Stream*: *slave*-интерфейс для приема входных изображений и *master*-интерфейс для выдачи результатов работы нейронной сети.

*IP*-ядро состоит из следующих модулей: модуль управления, отвечающего за регулирование транзакций обращения к памяти ядра (чтения и записи), управление сигналами внешних интерфейсов, выбор текущего слоя и функции активации; *ROM* и *RAM* память для хранения весов модели и промежуточных результатов; вычислительный модуль, отвечающего за перемножение кватернионов, накопление суммы и применение функции активации.

Память *ROM* и *RAM* была реализована на основе блоков *BRAM*, поскольку синтез на *LUT*-блоках оказался невозможным из-за ограничений по их количеству. В результате экспериментов квантования весов нейронной сети оптимальным вариантом формата данных был принят *Q16.13* (16-ти разрядное знаковое число с 13-ю разрядами после запятой).

*BRAM*-память имеет ряд ограничений, таких как невозможность одновременной записи и чтения (критично для буферизации промежуточных данных) и требование синхронного чтения.

Для решения этих проблем была применена архитектура, включающая разделение *RAM*-памяти на два буфера, операции чтения и записи которых выполняются поочередно, и внедрение задержки (*pipeline*) управляющих сигналов для коррекции задержанного прихода данных с *ROM*.

В результате симуляции и синтеза *IP*-ядра были получены временная диаграмма работы (рисунок 2), потребление ресурсов *FPGA* и временные задержки схемы (рисунок 3).

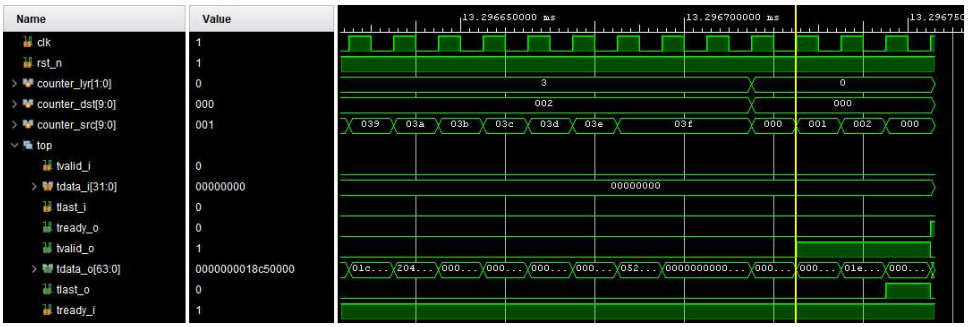
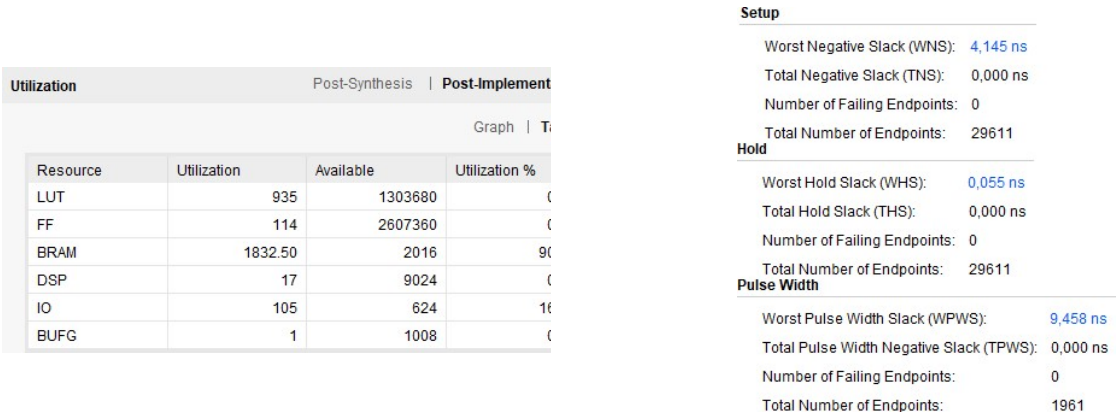



Рисунок 2 – Часть временной диаграммы с выводом результата работы *IP*-ядра




а б

Рисунок 3 – Результаты синтеза и имплементации *IP*-ядра: а – затраты ресурсов *FPGA*; б – запас на задержку схемы при тактовой частоте 50 МГц

**Заключение и дальнейшее развитие**

Низкая эффективность и высокие требования к памяти существующей модели могут быть нивелированы путем модификации архитектуры сети.

Во-первых, предлагается использовать сверточные слои [3] на начальных уровнях, что позволит снизить размерность данных, поступающих в последующие полносвязные слои и, соответственно, уменьшить количество весов. Данный тип слоя использует ядро свертки k×k для преобразования входного изображения N×N в матрицу размером (N+1-k)×(N+1-k).

Во-вторых, для предотвращения переобучения модели целесообразно встраивать слои регуляризации dropout [4] между слоями различного типа. Активация dropout-слоев, осуществляемая исключительно на этапе обучения, приводит к частичному отключению нейронов, стимулируя тем самым извлечение сущностных признаков и закономерностей из данных вместо простого запоминания входного набора.

***Список использованных источников:***


1. *Автокодировщик цветных изображений на основе кватернионной полносвязной нейронной сети // Цифровая обработка сигналов и её применение (DSPA 2024) (Москва, РФ)*
2. *The CIFAR-10 dataset [Электронный ресурс]. – Режим доступа : https://www.cs.toronto.edu/~kriz/cifar.html*
3. *Багаев И.И. Анализ понятий нейронная сеть и сверточная нейронная сеть, обучение сверточной нейросети при помощи модуля Tensorflow Математическое и программное обеспечение систем в промышленной и социальной сферах. 2020. Т. 8. № 1. С. 15-22.*
4. *Analysis of Dropout [Электронный ресурс]. – Режим доступа : https://pgaleone.eu/deep-learning/regularization/2017/01/10/anaysis-of-dropout*