FPGA Implementation of Quaternionic Fully Connected Neural Network for Image Classification

Aleksey Osipov
Department of Computer Engineering
Belarusian State University
of Informatics and Radioelectronics
Minsk, Belarus
axelosip3345@gmail.com
ORCID:0009-0007-0041-4229

Nick Petrovsky
Department of Computer Engineering
Belarusian State University
of Informatics and Radioelectronics
Minsk, Belarus
nick.petrovsky@bsuir.by
ORCID:0000-0001-5807-8685

***Abstract*. In this paper, we propose the design, implementation, and FPGA-based evaluation of a fully connected quaternion neural network for color image classification, benchmarked on the CIFAR-10 dataset. The IP core was successfully synthesized for UltraScale series FPGAs, demonstrating minor resource utilization. While the current implementation achieves its resource efficiency goals using an FC-only structure, classification performance is constrained by the lack of softmax implementation for quaternion outputs.**

***Keywords:* quaternion algebra, Neural Network, fully connected layer, FPGA**

I. Introduction

In recent decade, neural networks (NN) in the real domain have been extensively studied and have shown promising results in various vision tasks. According to theoretical analysis, representations such as hyper-complex numbers can acquire richer representational capacities than real numbers, and capture intrinsic interchannel relationships [1]. Quaternion algebra offers an approach that allows for the mapping of each pixel to a single entity. Furthermore, this could lead to a more efficient internal representation in colors space, resulting in improved results in color image processing tasks [2].

The purpose of this research is to describe the design, training, and evaluation of a quaternion-valued fully connected neural network that can classify color images using the *CIFAR-10* dataset. Efficient hardware implementation based on FPGA requires model quantization to avoid floating point logic design complexity and resource utilization on MAC operations.

The paper is organized as follows. Section two outlines the quaternion algebra and quaternionic pixel representation. Section three explains the quaternionic neural network classifier. Section four illustrates the hardware implementation; simulation experiments with real image is presented. Section five is devoted to conclusions and future work.

II. Application of quaternion algebra

A. Basics

The quaternion algebra $\mathbb{H}$ is an associative non-commutative four-dimensional algebra $\mathbb{H} = \{Q = q_1 + +q_2 i + q_3 j + q_4 k | \, q_1, q_2, q_3, q_4 \in \mathbb{R}\}$ where the orthogonal imaginary numbers obey the following multiplicative rules $i^2 = j^2 = k^2 = ijk = -1$, $ij = = -ji = k$, $ki = -ik = j$. This becomes evident after identifying quaternions with vectors and writing the operation in matrix notation [3].

B. Color space representation

To represent a picture in quaternion space, the color components of every pixel, which are in the RGB color space, are translated into the imaginary parts of a quaternion:

$$q = 0 + R \cdot i + G \cdot j + B \cdot k,$$

where $(R, B, G) \in \mathbb{R}$ is RGB pixel and q is pure quaternion.

For the sake of simplicity RGB channels is expected to be shifted to range $[-0.5, 0.5]$ before converting to the quaternion space.

The key operation in quaternion algebra is non-commutative multiplication (or Hamilton product), the geometric meaning of which is the rotation of a vector around axis in three-dimensional space. The multiplication operation involves two operands: the original vector *x*, representing the pixel, and the quaternion *w*, encoding information about the axis and angle of rotation. The quaternion *w* can be represented as follows:

$$w = a \cdot (\cos\theta + \sin\theta \cdot u),$$

where a – scale value; θ – rotate angle; u – rotation axis in 3D space, represented as normalized pure quaternion.

For example, the quaternion $q_{gray} = 0 + 0 \cdot i + +0 \cdot j + 0 \cdot k$ corresponds to a middle grey color. A white pixel is encoded as $q_{white} = 0 + 0.5 \cdot i + 0.5 \cdot j +$

Pattern Recognition and Information Processing (PRIP2025) :
Proceedings of the 17th International Conference, 16–18 Sept. 2025, Minsk, Belarus. – Minsk : UIIP NASB, 2025. – 496 p. – ISBN 978-985-7198-20-7.

$+0.5 \cdot k$, and black as $q_{black} = 0 - 0.5 \cdot i - 0.5 \cdot j - -0.5 \cdot k$.

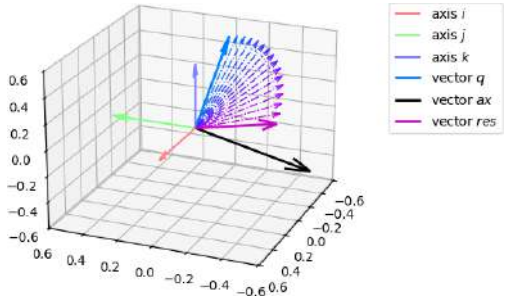

Fig. 1. Quaternion rotation in color space

On the Fig. 1, the operation of rotating vector q (represents color blue):

$$q = 0 - 0.5 \cdot i + 0 \cdot j + 0.5 \cdot k,$$

around non normalized axis *ax* (black):

$$ax = 0 - 0.5 \cdot i - 0.5 \cdot j - 0.5 \cdot k,$$

to an angle of $\theta = 90°$ without scaling ($a = 1$). Result of rotation is vector *res* (magenta).

The quaternion *w* in this case will be equal to:

$$w = 0 + 1 \cdot \frac{ax}{\|ax\|} = \frac{1}{\sqrt{3}} \cdot (i + j + k).$$

The order of quaternion multiplication determines the direction of vector rotation. In this work, will use "right"-order multiplication $w \cdot x$, that corresponds to counterclockwise rotation [3].

III. Model Architecture

A. Training of classifier

The input data to the neural network is a color image of size 32×32 pixels, which is transformed into a vector consisting of 1024 quaternions. As shown in Figure 2, the neural network architecture comprises three fully connected layers. The hidden layers have dimensions of 24×24, 16×16 and 8×8, respectively. The output of the neural network consists of three quaternions, each of whose 10 components represents a distinct class from the dataset.

The Leaky ReLU activation function is chosen for its ease of hardware implementation [4]. In the negative value region, the reduction in value is achieved by an arithmetic right shift, and the sign of the number is determined by the sign bit. The slope value in this case is 2^{-n}. Since the output layer of the quaternion neural network does not contain the *softmax* function [5, 6], which is typically used in classification tasks, the Cross-Entropy loss function is not efficient for training. Instead, the Mean Squared Error was chosen. Adam optimizer is used with adaptive learning rate, varying according to a cosine law.

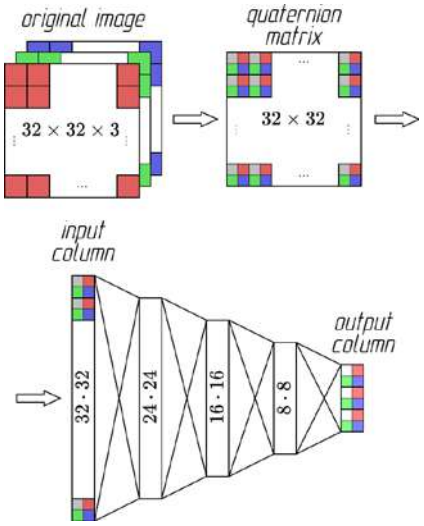

Fig. 2. Quaternion Neural Network architecture based on fully connected layers

B. Experimental results

Fig. 3 presents the result of training a neural network. Analysis of the graph indicates a rapid plateau in the training process. Further research will be aimed, among other things, at modernizing the architecture of the neural network in order to increase the efficiency of the NN. Confusion matrix of trained Neural Network is shown in Fig. 4.

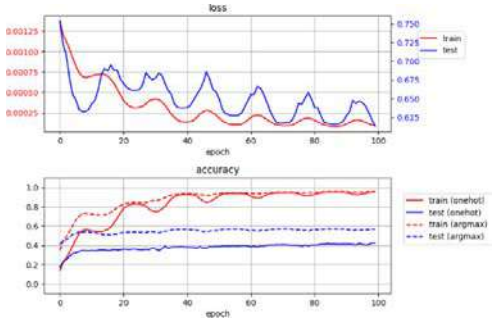

Fig. 3. Training curves

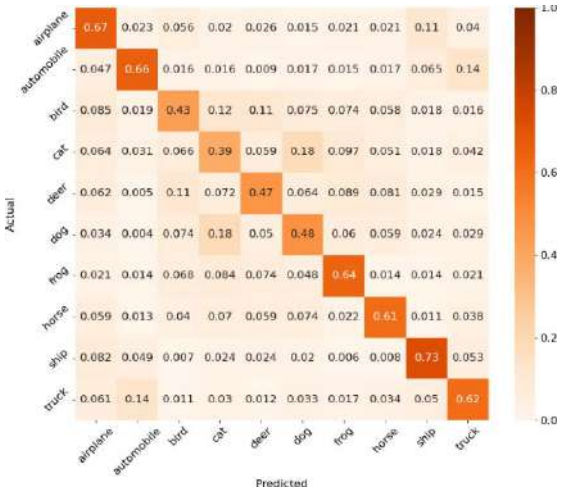

Fig. 4. Confusion matrix

Pattern Recognition and Information Processing (PRIP2025) :
Proceedings of the 17th International Conference, 16–18 Sept. 2025, Minsk, Belarus. – Minsk : UIIP NASB, 2025. – 496 p. – ISBN 978-985-7198-20-7.

IV. HARDWARE IMPLEMENTATION

A. Quaternion MAC acceleration

The primary challenge in the implementation of an IP core for neural network resides in the substantial memory requirements for storing both weight coefficients and intermediate computational results. To resolve this, the following methodologies are employed:

- Weight coefficients are stored within *Block RAM* modules, which imposes a limitation of requiring synchronous reading from the weight memory.
- Intermediate results of NN computing are buffered using a dual-buffer memory architecture. The size of each buffer is determined by the maximum layer size of the NN. Simultaneously, writing is performed only in one of the buffers, while a read operation is performed from the other.

For the hardware implementation of the IP core, a balanced parallel-serial architecture is used, which allows one quaternion multiplication operation to be performed per clock cycle. Quaternion multiplier is implemented in vector-matrix form [3], such straightforward require 16 real multiplications and 12 additions, which can be implemented using DSP blocks on target FPGA.

The computational process for the fully connected quaternion layer:

$$h = W \cdot x + b,$$

where all variables are quaternion-valued, i.e. h denote output, x denote input, b denote bias and W is weight. The formula for calculating i -th element of h is following:

$$h_i = \sum_{k=0}^{M-1} \left(W_{k,\,i} \cdot x_k \right) + b_i,$$

where $W_{k,\,i}$ is k -th weight of layer size of M. Alternatively, the addition of the bias value b_i can be integrated into the matrix multiplication:

$$h_i = \sum_{k=0}^{M} W_{k,\,i} \cdot x_k,$$

where $W_{M,\,i} = b_i$, and $x_M = 1 + 0 \cdot i + 0 \cdot j + 0 \cdot k$.

Activation function is implemented as split activation [2]:

$$f(p) = g_1(p_1) + g_2(p_2)i + g_3(p_3)j + g_4(p_4)k,$$

where p – quaternion-valued argument of the function, $g_{1\ldots4}(\,)$ – real-valued activation function, such as leaky ReLU.

The block diagram of the IP core is shown in Fig. 5.

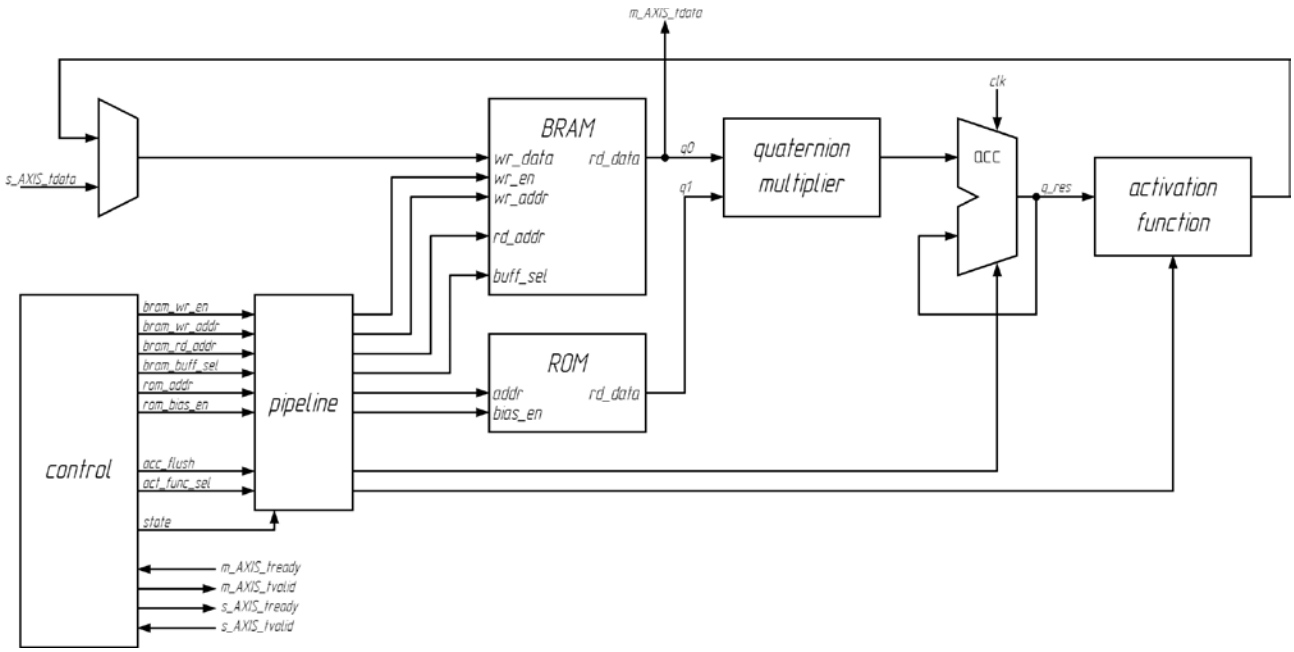

Fig. 5. Quaternion fully connected layer accelerator with AXI-stream interface

Pattern Recognition and Information Processing (PRIP2025) :
Proceedings of the 17th International Conference, 16–18 Sept. 2025, Minsk, Belarus. – Minsk : UIIP NASB, 2025. – 496 p. – ISBN 978-985-7198-20-7.

B. System design

The quaternion IP core is a standalone block and transactions with it are carried out via two AXI-Stream interfaces. A block design of processor system integrated with accelerator core connected to a DMA block is depicted in Figure 6. Weight coefficients and intermediate data of the neural network are stored in a quantized form in Q16.13 fixed point format per word or 64 bits per quaternion. This format is the most resource efficient, without noticeable impact to the accuracy of the network. FPGA resource utilization of this quantized NN is shown in Table for SoC Ultrascale+ XCVU37P. Synthes, implementation and Place & Route is performed using Xilinx Vivado 2023.2.

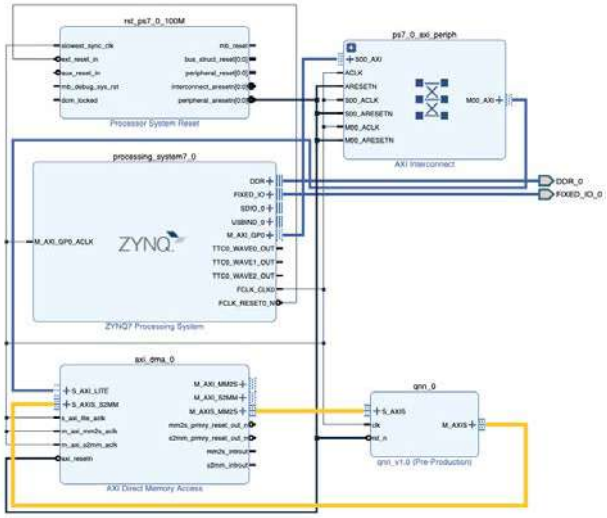

Fig. 6. Block design of developed system
(yellow color bus is datapath for input image)

FPGA RESOURCES UTILIZATION OF THE
VIRTEX ULTRASCALE+ XCVU37P-L2FSVH2892E

Resource	*Utilization*	*Available*	*Utilization [%]*
LUT	935	1303680	0.07
FF	114	2607360	0.01
BRAM	1833	2016	90.90
DSP	17	9024	0.19

C. IP core simulation results

During the simulation of the IP core operation, a timing diagram was obtained. Fig. 7 presents a fragment of this diagram showing the classifier output transmitted via the AXI-Stream interface. Payload is on the `tdata_o` bus asserted when the `tvalid_o` signal is high.

The output data consist of three quaternions, which collectively represent twelve classes, of which ten are utilized. Quaternion components marked with a $\langle - \rangle$ sign correspond to irrelevant data and will be ignored during representation. The simulation result can be presented in the form of the neural network's confidence vector as follows:

$$\begin{cases} q_0 = 0\text{x}0000\_0000\_18c5\_0000 \\ q_1 = 0\text{x}01e8\_0000\_0000\_0000 = \\ q_2 = 0\text{x}0000\_0000\_0000\_0000 \end{cases}$$

$$= \begin{cases} q_0 = 0 + 0 \cdot i + 0.7740 \cdot j + 0 \cdot k \\ q_1 = 0.0596 + 0 \cdot i + 0 \cdot j + 0 \cdot k = \\ q_2 = 0 + 0 \cdot i + 0 \cdot j + 0 \cdot k \end{cases}$$

$$= \begin{cases} 0\% & \langle airplane \rangle; & 0\% & \langle horse \rangle; \\ 0\% & \langle cat \rangle; & 0\% & \langle - \rangle; \\ 77.4\% & \langle frog \rangle; & 0\% & \langle bird \rangle; \\ 0\% & \langle truck \rangle; & 0\% & \langle dog \rangle; \\ 5.96\% & \langle automobile \rangle; & 0\% & \langle ship \rangle; \\ 0\% & \langle deer \rangle; & 0\% & \langle - \rangle; \end{cases}$$

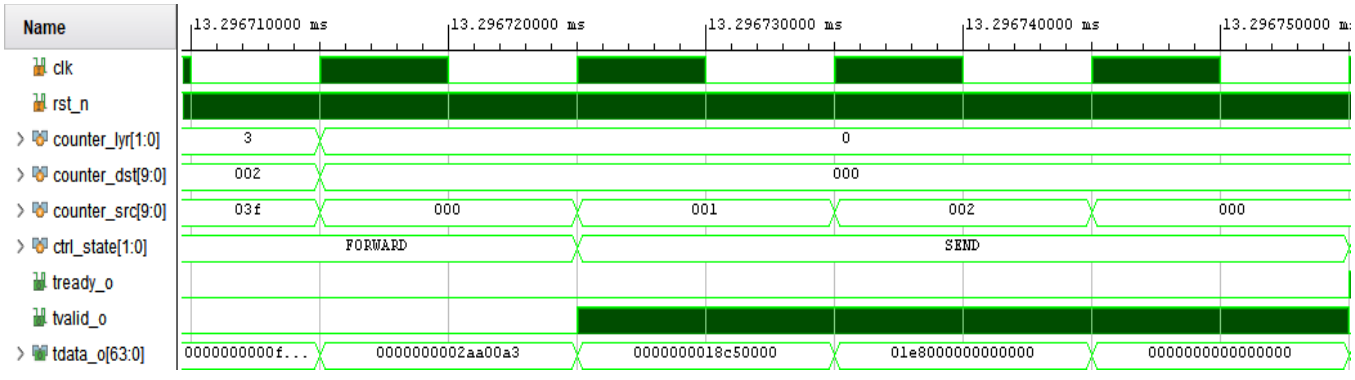

Fig. 7. Waveform of developed IP core with classification results

Pattern Recognition and Information Processing (PRIP2025) :
Proceedings of the 17th International Conference, 16–18 Sept. 2025, Minsk, Belarus. – Minsk : UIIP NASB, 2025. – 496 p. – ISBN 978-985-7198-20-7.

V. Conclusion

This work presents a neural network designed for the classification of color images from the *CIFAR-10* dataset. The choice of the classification task is due to the benchmarking of the neural network architecture on small weight number. As a result, an adaptive IP core has been developed that can be applied to solve various color image processing tasks. The proposed neural network uses an architecture based solely on fully connected layers. As part of further research, it is planned to expand the network architecture by integrating convolutional layers to improve data processing efficiency. Quaternion output layers have no softmax implementation, which limiting classification performance based split activation functions such as *ReLU* or *Leaky ReLU*. However, future research is planned to address the quaternion activations functions.

References

[1] T. Parcollet, M. Morchid, and G. Linar` es, “Quaternion convolutional neural networks for heterogeneous image processing,” in ICASSP 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 8514–8518.

[2] T. Parcollet, M. Morchid, and G. Linar` es, “A survey of quaternion neural networks,” Artificial Intelligence Review, vol. 53, no. 4, 2020, pp. 2957–2982.

[3] I. L. Kantor, A. S. Solodovnikov, “Hypercomplex Numbers: an Elementary Introduction to Algebras”, 1989.

[4] A. Zhang, Y. Tay, S. Zhang, A. Chan, A. T. Luu, S. C. Hui, and J. Fu,“Beyond fully-connected layers with quaternions: Parameterization ofhypercomplex multiplications with 1/n parameters,” arXiv preprint arXiv:2102.08597, 2021.

[5] Z. Xuanyu, X. Yi, X. Hongteng, and C. Changjian, “Quaternion Convolutional Neural Networks”, Shanghai Jiao Tong University.

[6] J. Pöppelbaum, Andreas Schwung, “Improving Quaternion Neural Networks with Quaternionic Activation Functions”.

Pattern Recognition and Information Processing (PRIP2025) :
Proceedings of the 17th International Conference, 16–18 Sept. 2025, Minsk, Belarus. – Minsk : UIIP NASB, 2025. – 496 p. – ISBN 978-985-7198-20-7.