Final Report: Fully Connected Neural Network implementation on FPGA

-Anura Mantri-22110144
-Mohit Maurya-22110145
-Neerja Kasture-22110165

Abstract:

This project aims to implement a fully connected neural network (FCNN) on the Nexys 4 DDR FPGA platform for digit prediction tasks using the MNIST dataset. The FCNN architecture is optimized for FPGA constraints to achieve high accuracy and low latency in real-time inference. By leveraging FPGA hardware acceleration, the system demonstrates the feasibility of deploying neural networks in embedded environments for efficient digit recognition.

Neural Network Parameters:

1. Input Image Size: The input image has dimensions of 28x28 pixels. This means each image in the MNIST dataset is a grayscale image with a width of 28 pixels and a height of 28 pixels.
2. Number of Layers: The FCNN consists of two layers:
   - Hidden Layer: This layer contains 512 neurons. Each neuron in this layer receives input from all 784 (28x28) pixels of the input image.
   - Output Layer: This layer has 10 neurons, corresponding to the 10 possible digits (0 to 9). Each neuron in this layer takes input from all 512 neurons in the hidden layer.
3. Activation Function: Rectified Linear Unit (ReLU) activation function is applied to the neurons in the hidden layer. ReLU function introduces non-linearity to the model, enabling it to learn complex patterns in the data more effectively.
4. Output Encoding: The output layer consists of 10 neurons, each representing a digit from 0 to 9. The digit with the highest output value among the 10 neurons is considered as the predicted digit. This is achieved by applying a function similar to softmax activation function to the output layer.

In summary, the FCNN processes 28x28 pixel input images through a hidden layer of 512 neurons with ReLU activation, followed by an output layer of 10 neurons representing digits 0 to 9, with the predicted digit being the one with the highest probability output by the network.

Implementation:

We have modified the input image to be predicted, weights matrix of layer 1, and that of layer 2 to suit our convenience as follows:
The image matrix is flattened and a zero(0) added at the beginning and a one(1) added at the end of the flattened image. Now the input size is (28*28)+1+1=786.

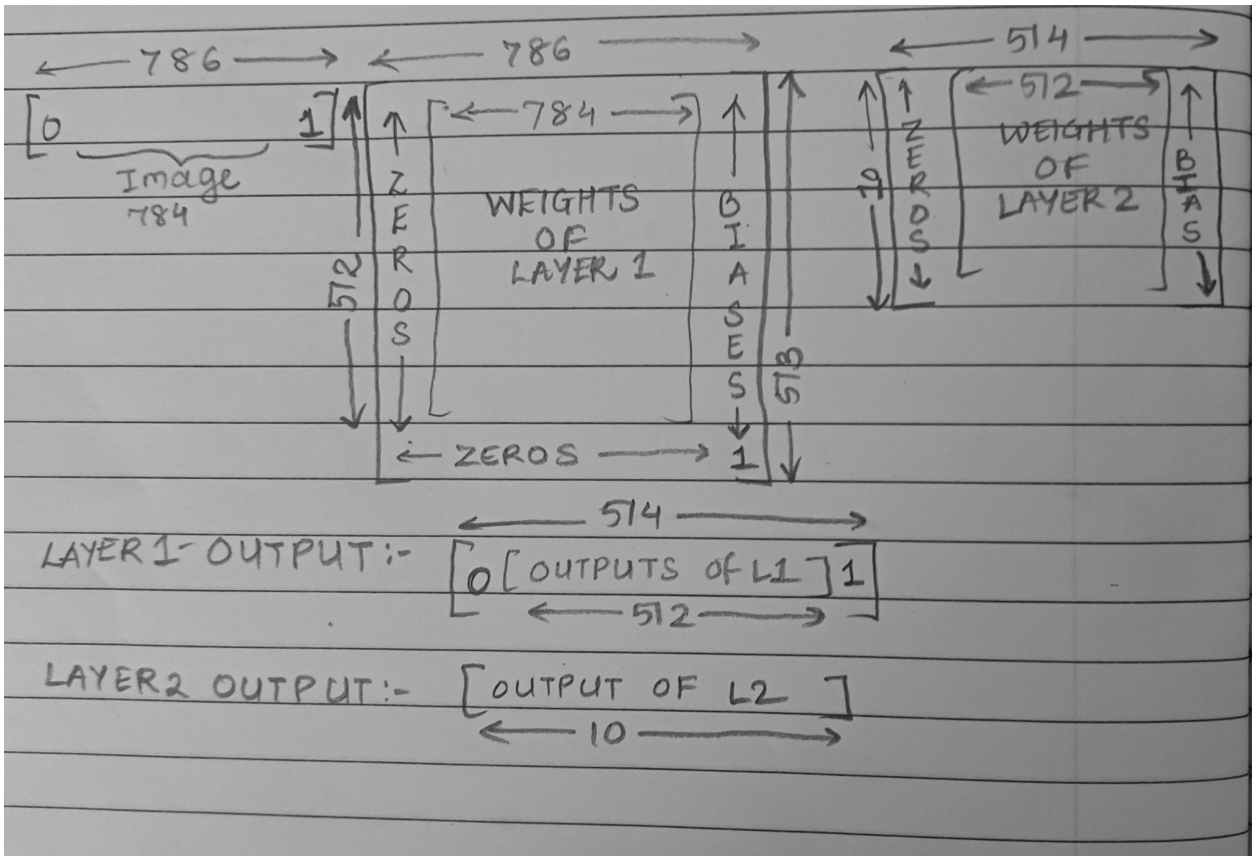
The weights of layer 1 have 512 rows for 512 neurons in the layer and each row having 786 columns for each entry in the image matrix to be multiplied by the weights. Each row has a padded zero(0), 784 weights and a bias for every neuron making each row of size 786. After 512 rows, one extra row is added to the weights layer 1 matrix to account for the biases of the next layer. This row consists of 785 zeros and a one(1) at the end so that the last output of layer 1 is a one(1). This matrix is flattened having a size of (512+1)*786 =403218.

The weights of layer 2 are stored in a similar way, having 10 rows (for 10 neurons in the output layer) and 514 columns (padded zero(0) + 512 weights for neurons of layer 1 + bias of respective neuron). This matrix is then flattened having a size of 10*514=5140.

All this data i.e. input image, weights of layer 1 and weights of layer 2 are sent through UART in a single BRAM(BRAM1). The total number of bytes being- 786+403218+5140=409144.

The output of layer 1 is a single array of size 514 (padded zero(0)+ output of 512 neurons+one(1) for the biases of the next layer) . This array is passed through the ReLu activation function and stored in a new BRAM. These values are then retrieved along with the weights of layer 2 in BRAM 1 to perform the respective matrix multiplication and get the desired output. The output of layer 2 is a single array of size 10 representing one neuron for each digit.

Below is the matrix representation of the modifications:



Conclusion:

This project has demonstrated the successful deployment of a fully connected neural network (FCNN) on the Nexys 4 DDR FPGA platform for digit prediction with the MNIST dataset. It highlights the feasibility and effectiveness of using FPGA-based hardware accelerators for real-time inference tasks, especially in resource-constrained environments. Moving forward, there's ample scope for further optimization and exploration in leveraging hardware accelerators for neural network inference, opening doors to innovative applications in embedded machine learning systems.