FGPA-based Deep Neural Network MNIST for Digit Classification

Mingyang Mao
Department of Electrical and Computer Engineering
Johns Hopkins University

Abstract—

I. Introduction

A. Goal

This project is to provide practical experience in implementing deep neural network execution on FPGA for the classification of MNIST dataset classes. This involves designing an FPGA-based fully connected (FC) layer neural network, including ReLu activation and softmax functions. The project is divided into phases, including software coding, Verilog implementation, and hardware analysis.

B. Results

This project was fully completed, from software to nexys FPGA board.

II. Background

A. FPGA

FPGA (field-programmable gate array) are integrated circuits that can be configured manufacturing. It consists of a series of programmable logic block and interconnections, which achieves divers digital functions. FPGAs are frequently employed in scenarios demanding adaptability, rapid processing, and the ability to handle tasks in parallel. This makes them highly suitable for use in industries like telecommunications, automotive, aerospace, and various manufacturing sectors.

Configurable Logic Block (CLB) is the most basic piece of an FPGA. An individual CLB includes several discrete logic components itself, such as look up tables (LUTs) and flip-flops. Digital Signal Processing (DSP) Slice is one of the specialized components in an FPGA. It is more efficient than CLBs in specific works.

Block Random Access Memory (BRAM) is the dedicated memory on the chip. These units are capable of being partitioned or sequentially integrated, thereby enabling the availability of BRAM in assorted dimensions, ranging from reduced to augmented scales. Input/Output (IO) Blocks are components serve as the conduits for data to be transmitted into and out of the FPGA.

B. MNIST

MNIST database is a large database of hand handwritten digits that is commonly used for research in the field of image processing and machine learning. The images included in the MNIST are black and white, with size of 28×28 pixels. This dataset is divided into training dataset (60k images) and testing dataset (10k images), which is widely used for training and testing the performance of machine learning methods such as convolutional neuron network.

Fig. 1 MNIST Datasets.

C. Neural Network

Neural Network (NN), also known as artificial neural networks (ANN), forms the core of deep learning algorithms. The design of NN is derived from the human brain, emulating how biological neurons communicates with each other.

Neural Networks are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each neuron is linked to others and possesses a corresponding weight and threshold level. Should the output of a particular node exceed this threshold, the node becomes activated, transmitting information to the network's subsequent layer. If the output does not meet this threshold, the node remains inactive, and no data is forwarded to the following layer. As Figure 2 shows, this model includes two fully connected layers. In the first layer, there are 128 neurons and 784 weights for each neuron, followed by a ReLU activation function. ReLU introduces non-linearity to deep learning model, effectively addressing the issue of vanishing gradients. It processes only the positive portion of its input argument. ReLU stands as one of the most widely utilized activation functions in the field of deep learning. Second layer is an output layer with 10 classifications.

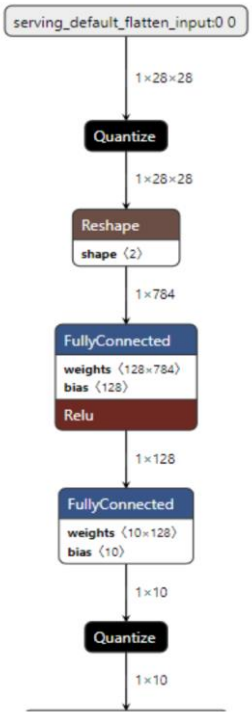

Fig. 2 FC-layer-based deep neural network for classifying MNIST data.

D. Existing Architecture

Figure 3 illustrates the architecture of the design blocks did by Cornell University students. They used two layers and 128 hidden layers as well. In their design, the control unit sends data and instructions to the initial forward propagation unit, which calculates the output for the first layer of neurons. This output is then conveyed sequentially to subsequent units, culminating at the classifier. The classifier determines the specific handwritten digit presented and subsequently relays this identification to the video stack.

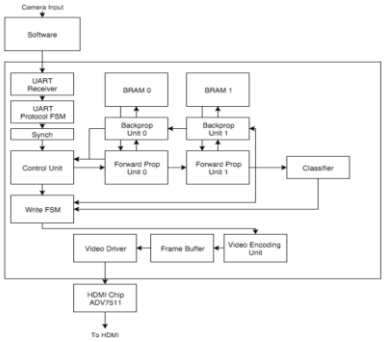

Fig. 3 System Block Diagram.

such as ‘sum’, ‘weight address’, ‘Relu’ and ‘write enable’ signal. Figure 5 shows the process of first neuron turning to the second neuron. Here, ‘weight address’ and ‘pixel address’ stop adding and wait for the final multiplication and sum, and writing to the feature map1. The ‘ReLu’ and ‘write enable’ are turned on when the ‘counter’ was between 783 and 785. Therefore, in each cycle, there are two another clock for the multiplication, sum and write. Then counter and sum are reset. ‘Weight address’ continues to add. Meanwhile, the counter outputs to the top, make the pixel address reset and the address of feature map1 to move to the next.

Similarly, the neuron 2 also has a counter that acts the same role as that in neuron 1. This help to refresh the input from feature map 1 and the output to feature map 2. In the top file, a step 2 enable signal is defined as if the last number of feature map 1[128] is assigned. If feature map 1[128] is given a number, it means step1 is finished and then step 2 could start working. Therefore, step 3 enable signal has the same define logic. If step 3 enable turn high, then the ‘argument max’ function can work. These are the overall design logic of this project.

III. Method

A . Architecture & control logic

The hierarchy of the design is shown in figure 4. The top file includes the memory of the pixel values, with size of 784, and the pixel address that controls pixel input to the neuron 1. Module ‘Neuron1’ and ‘Neuron 2’ were included in the top module.

In the ‘Neuron 1’, a counter works as a state machine. It enables the working state of operations

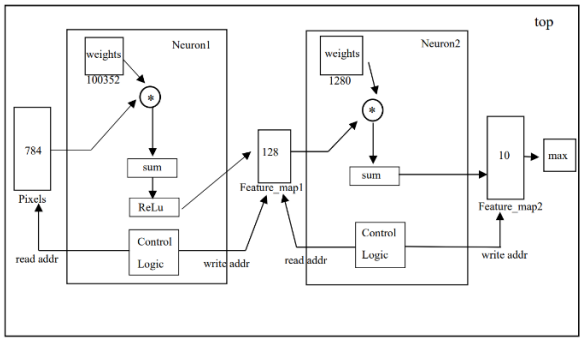

Fig. 4 Architecture of design.

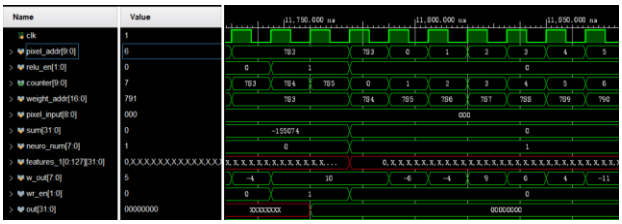

Fig. 5 Simulation waveform of Neruon1

B . Implementation

In the phase 1, software verification was implemented on Colab. The mnist inputs are imported from the keras, which is a deep learning API with built-in datasets. The weights in two fully connected layer were pre-trained and imported in. ReLU activation function was implemented after first FC layer. The function similar to argument max was implemented in the last part. Another part in phase 1 is to convert the weights and image data to the type that be directly read.

Phases 2 and 3 focus on the Verilog coding, about the first and second fully connected layer separately. In the ‘top’ file, the depth and width of

parameters are defined. The required memory could refer to the table in results. In the top, pre-processed image file was read and saved to the memory. In two neurons, processed weights files were also imported in the initial begin. The detailed logic has explained in the part A.

Phase 4 is mainly about the implementation on the board. In and out ports are configured in the constraints file or in the elaborated design. As shown in figure 5, AN0 to 7 enable the 8 7-segments display. CA to G control the state of each segment. A display module was designed that the input is recognized digit and output is the corresponding control signal to the segment display. The implementation result is shown in the next part.

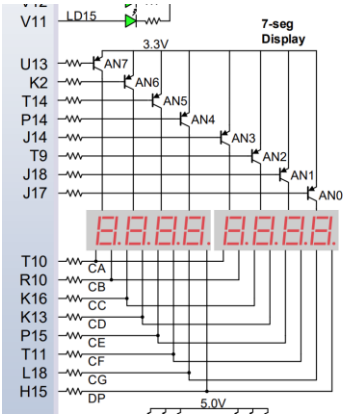

Fig. 6 General Purpose I/O devices on the Nexys4 DDR

IV. Results

The deep neural network on python shows high reliability. The final accuracy is 99.05% by testing 10000 samples. One drawback is that when the width of the input and weight truncated to 8 bits, the accuracy drops to 10%. Therefore, this working program required more memory.

The simulation implemented on phase 2 and phase 3 works correctly. The table below shows the source utilization, power and clock timing information on the board. The clock frequency was turned downed to 66.67 MHz to avoid time slack.

Implementation on the board successfully finished shown as figure 7. The recognition is correct. ILA (integrated logic analyzer) helps a lot in the debugging on board.

V. Conclusion

Table 1. Utilization, power and timing reports.

Parameter	Value
Implementation style	Serial
Weight memory 1	$784 \times 128 \times 9bit$
Weight memory 2	$128 \times 10 \times 9bit$
Feature map 1	$128 \times 32\ bit$
Feature map 2	$10 \times 32bit$
Slice LUTs	15706
LUTRAM	0
Slice Register	2990
DSP	3
Block RAM	0
Latency (ms)	1.53
Dynamic Power (mW)	234
Cycles	101916 @66.67MHz
Dynamic Energy(mJ)	0.351

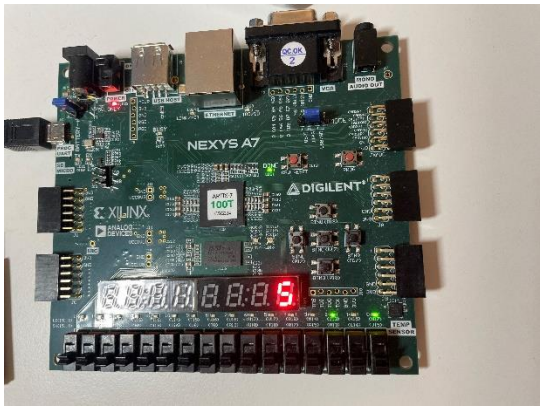

Fig. 7. demonstration of the board.