CMPE-530/630 Digital IC Design
Final Project:
Neural Network Datapath Design

Alden Davidson
John Judge
Submitted: December 19th, 2018

Lab Section: L1
Instructor: Cory Merkel
Lecture Section: 01
TAs: Michael Shullick
Marisa Langlois
Gaurav Mohan Shende

Lecture Section: 01

By submitting this report, you attest that you neither have given nor have received any assistance (including writing, collecting data, plotting figures, tables or graphs, or using previous student reports as a reference), and you further acknowledge that giving or receiving such assistance will result in a failing grade for this course.

Your Signature: *Alden Davidson & John Judge*

Table of Contents

Abstract

In this exercise a hardware implementation of an Artificial Neural Network for edge detection in images was implemented in VHDL and tested. The VHDL design of a MLP was created and tested using Vivado. During the simulations of the MLP, it was determined that the layout of the circuit would take approximately 35% of the total LUT's of a Nexys 4 DDR FPGA. Furthermore, it was determined that the FPGA implementation of the board would consume approximately .477 watts of power (this was a low level of confidence). It was also recorded that the circuit would take approximately 86.5µs tp read from an image file and take about 732µs to complete the entire edge detection algorithm. Overall the exercise was a success and the edge detection of the photo was found to be accurate. Although, had there been more time to complete the project the requirement of a generated layout in Mentor Graphics could have been completed.

Design Methodology

In this project, a hardware implementation of an artificial neural network (ANN) was designed. To be more specific, a Multiplayer Perception (MLP) ANN was created to preform edge detection on images greyscale images. In order to implement the MLP, hardware designs for the different nodes composing the neural network as well as several smaller components.

The MLP

The top-level wrapper of the design, the MLP, was created to be a synchronous hardware implementation of an ANN for edge detection. The entity was created to have inputs for Serial Input (SI), Serial Enable (SE), Clock (clk), Scan (u), and a signal which tells the MLP that the data being fed serially into u is the image file (i_valid). The output of the MLP was a std_logic vector called yhat which was serial data of an image containing detected images. A signal o_valid was also was added to the design. This was a single bit output that was high when the MLP was outputting the image. Using the SE signal to enable and disable the circuit, an image could be read in serial using the Scan port to read nine pixels (organized in a 3x3 cube) at a time. For the purposes of this exercise, the data was fed into the MLP using the Testbench. Had there been more time to finish the assignment, there was plans to add a UART protocol so the design could be implemented on a Nexys 4 DDR FPGA.

As the image was being scanned in, the weights and biases of the ANN were fed into the SI port. These weights and biases, which describe the weighted connections between all the nodes in the network, were previously calculated using given Matlab script that trained the network using given examples of edges. In this exercise, values of -1 and 1 were used for W_{min} and W_{max} respectively. Meaning, all weight values being fed into the MLP were floating point numbers between -1 and 1. These values were not changed because there did not appear to be any added benefit in increasing the values since the activation function would be used to flatten the final values to a 0 or 1. To create the serial stream of weights the MLP's testbench was used to open and read the w.dat Matlab file that was then shifted into the SI port one bit at a time. Similarly, the testbench was used to open and read the image file and create the std_logic_vectors for the MLP's U input.

In order to create the actual ANN that would be determining if an edge existed or not, three different node entities needed to be created in the MLP; the Input Node (N), Hidden Node (H), and Output Node (M). The ANN can be seen in Figure 1. This node layer design is typical of ANNs as it is akin to neuron networks in the human brain.

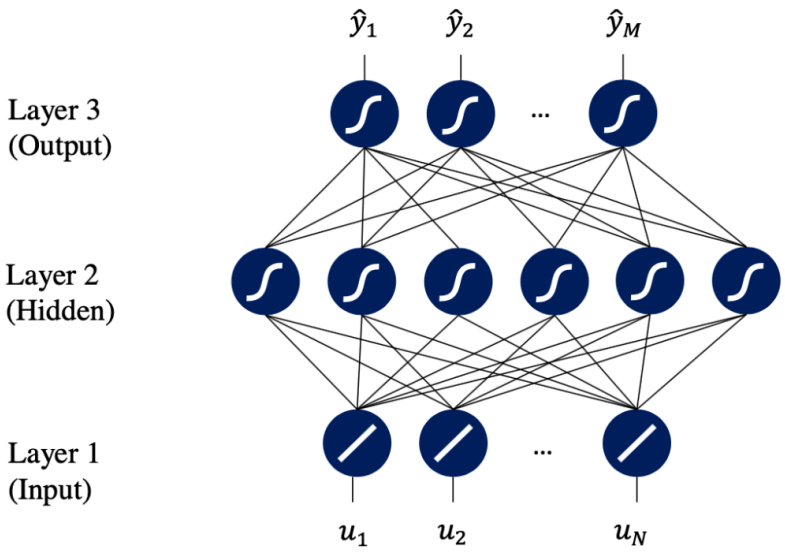

Figure 1: The ANN

As seen in Figure 1, the network can be created up to an undetermined number of nodes. In order to mimic this is in the MLP design, all the nodes needed to be created using a generic size. Since files were needed to create each node type, a package needed to be created with the generic definitions for how many nodes would be in each layer, as well as some definitions of array types that would be used in each of the files. Following this node components that make up the MLP could be created. A block diagram schematic of the components of the MLP can be seen in Figure 2.

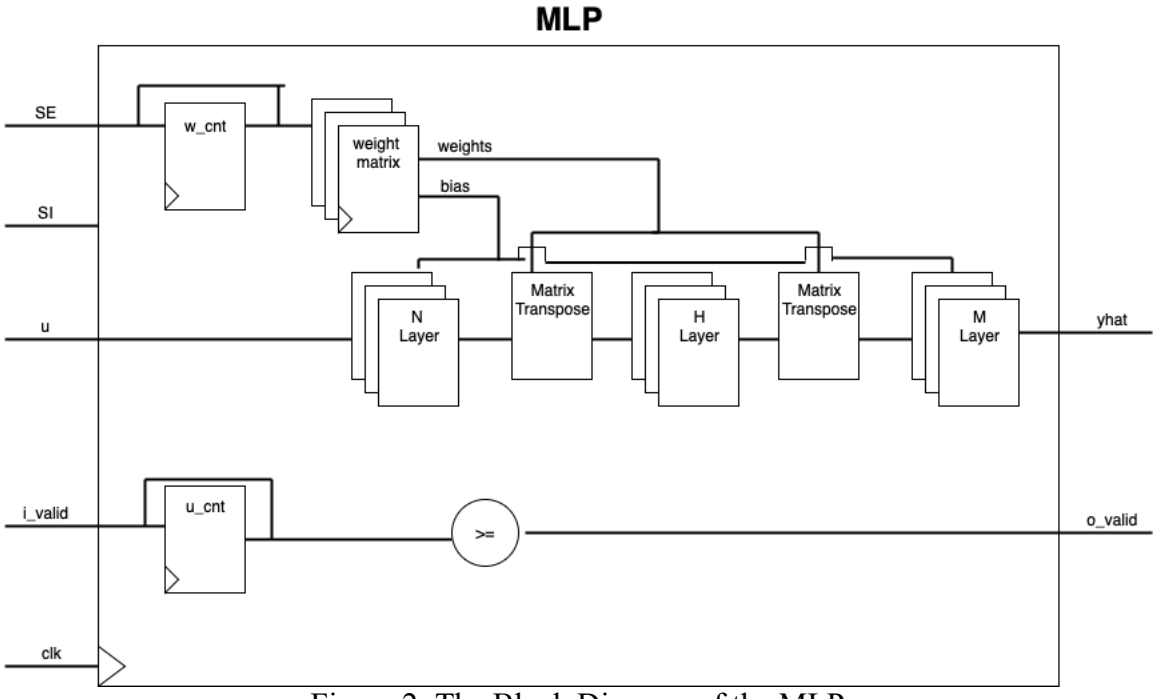

Figure 2: The Block Diagram of the MLP

As seen in Figure 2, the MLP was composed of the three node types and some arrays to hold data before being passed to the next sequential node type. These two-dimensional arrays were generated using registers and multiplexers. In addition, the values being held in the registers were converted to Qformat numbers using the Fixed_Package from IEEE_Proposed. Following the design of the MLP, the design of each of the node types could be created.

The N Node

The Input Node entity was created first. A total of nine of these nodes would be used in the design, one for each pixel being read in by the image. This meant that the node would only need two inputs; a clock signal and U (the pixels). In order to process the image, the pixel value would need to be divided by 255 in order to get a value between 0 and 1. Following this, that value could be multiplied by the weight associated with the node. This would result in a fractional number, in order to pass this value between nodes, it would need to be stored in a Qformat array using the Fixed_Package. This array was the node's one output, n_out and would be passed to its connected nodes in the H layer.

The H Node

In the hidden layer, the node designs accounted for one extra input, the bias which was also represented by a fixed-point number. All of the inputs to each node, which were represented as array types, were summed. After accounting for the bias, an activation function was used to push the output (h_out), which was designed as a single bit, to either a 0 or a 1.

The M Node

The Output Node entity was then created. In this layer, one single node would be created. This node was created with input for the clock signal, an array of all the connections from the previous layer (m_in), and a bias represented in Q-format. This node worked by first summing all of the elements of the m_in array and accounting for bias. Following this, the activation function was used to push the signal to either a 0 or 1. In the case of this node, since it was the final node in the network, a 0 was used to denote no edge and a 1 was used to denote an edge in the 9-pixel space being observed. A collection of these edge or no-edge bit were then stored into an array and driven to the yhat output of the MLP.

Activation Function

After contemplating several activation functions such as Sigmoid, ReLU, and Leaky ReLU, it was decided that a Heaviside step function would be used. This equation can be seen below in Equation 1.

$$H(x) := \frac{d}{dx}\max\{x, 0\} \quad \text{for } x \neq 0 \qquad \text{Eq. 1}$$

This equation essentially drives any negative value to 0 and any value greater than 0 to 1. This equation was used in the H and M layers to convert the Qformat numbers to a binary value. This function was chosen over Sigmoid, ReLU or other such activation functions due to its simplicity. After testing it was determined that this function did not cause and loss of resolution in the final image generated.

Results

Following the initial design of the MLP and its sub components, a testbench needed to be developed. This testbench was implemented in such a way that it would open and read the

grayscale images and then shift the data into the MLP serially. The testbench would also drive the SE, SI, and i_valid signals accordingly, as well as provide the MLP with a 100 MHz clock. The testbench was written to plot the appropriate output waveforms as well as several internal signals and counters.

Following the creation of the testbench it was determined the initial design of the MLP was incorrectly indexing several of the internal 2D arrays. After fixing the error the MLP provided meaningful results. However, there was one major issue encountered in the design; the testbench incorrectly wrote to the output image file. The testbench wrote an incorrect size in the PGM header, resulting in black bars at the bottom of the design. After fixing this issue the design provided some meaningful results. The simulation output waveform results can be seen in Figure 3.

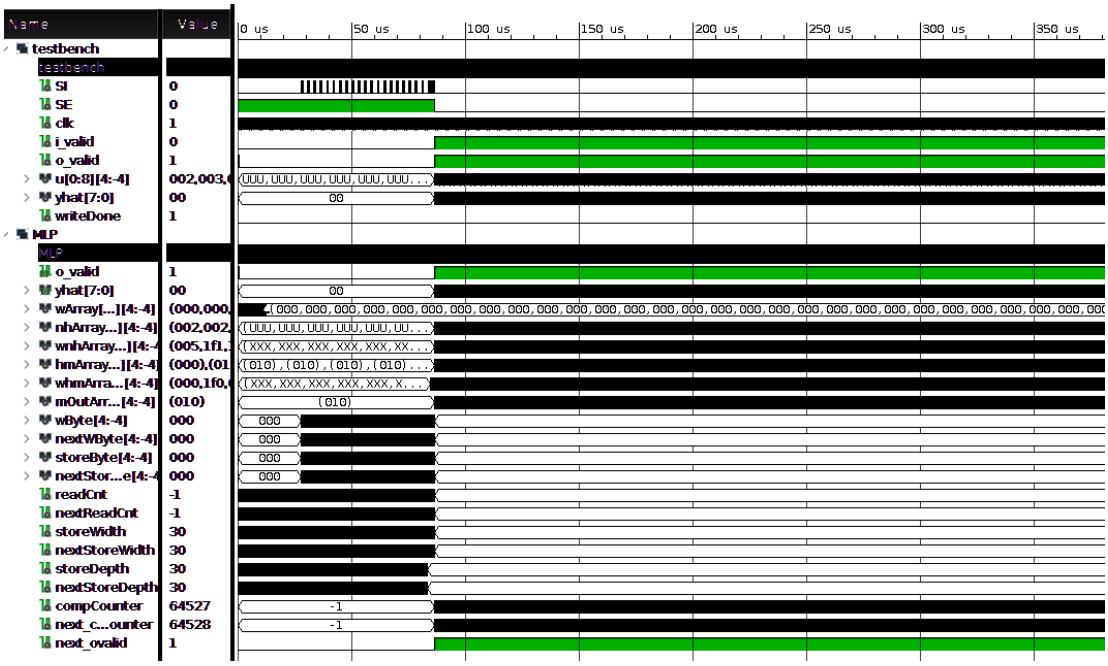

Figure 3: Simulation Results

As seen in Figure 3, serial data is read into the MLP until the entire weight file is read. Then the i_valid signal is driven to '1' indicating that the image file is being serially fed into the MLP. It took a number of clock cycles for the data to propagate through the design, but the serial data eventually makes its it way to the y_hat port. It's was at that point that the o_valid signal was driven to a '1' and then new image file was written by the testbench.

The output data was then written to a PGM file using the testbench. Using the Linux program ImageMagick, the PGM file was then converted to a PNG file so that the edge detection results could be verified. The original image used for testing the MLP as well as the output can be seen in Figure 4.

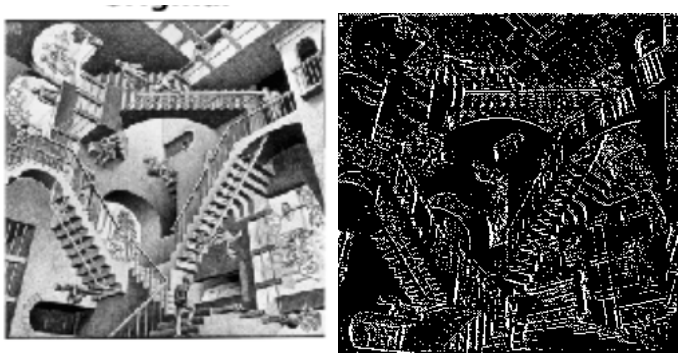

Figure 4: The Input and Output Images, Respectively

As seen in the above photos, the ANN was able to detect edges in the provided test images. Although the Heaviside step function was determined to be accurate enough for the purposes of this exercise, it was inferred that the noise in the output image was most likely due to the activation function. Specifically, the detected edges which appear as dots on the ceiling are most likely due to the approximation made in the activation function.

Following the ANN was validated and verified, several other result metrics were studied. In particular, the time that the MLP took to read in the file as well as the total time the MLP took from reading in the image file to wrting the output image was of interest. The internal counters of the MLP recorded that it took a total of 8,649 clock cycles to read in the file. Using a 100MHz clock, it was calculated that the reading of the image file took approximately 86.5μs. The total operational time of the MLP was found to be 73,178 clock cycles or approximately 732μs.

Next, using a simulated environment of a Nexys 4 DDR FPGA, some performance metrics were recorded. Vivado estimated the power consumption of the MLP to be approximately .477 watts, with a low confidence rating. In addition, it was found that the design would use approximately 35% of the total slice LUTs of the board.

Had there been more time to work on the project, there was intentions to implement a UART protocol to the design to read in the serial data to the MLP. In addition, it was originally planned to generate the layout of the circuit in Mentor Graphics. As the deadline of the project grew near, these additions were trimmed from the deliverables. Proposed future work for this project include both of these deliverables.

Conclusion

In this exercise a hardware implementation of an artificial neural network was implemented in VHDL and tested. The design was created and tested in Vivado. During the simulations on the circuits it was determined that the layout of the circuit would take approximately 35% of the total LUT's of the board. Furthermore, it was determined that the FPGA implementation of the board would consume approximately .477 watts. It was also recorded that the circuit would take approximately 86.5μs tp read from an image file and take about 732μs to complete the entire edge detection algorithm. Overall the exercise was a success and the edge detection of the photo was found to be accurate. Although, had there been more time to complete the project the requirement of a generated layout in Mentor Graphics could have been completed. However, the implementation of this feature is planned to be completed sometime soon. Overall the project was a success and may techniques were learned through implementation.

Appendix

*Please see the attached zip file of code