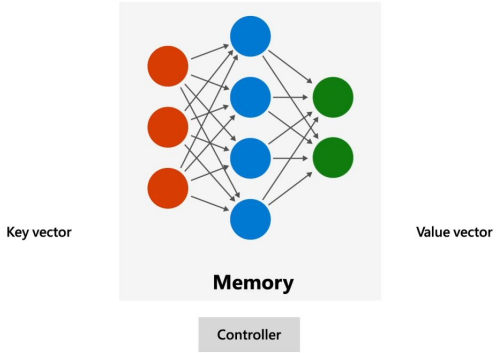
Hardware Approach of Forward Propagation in Neural Network on FPGA

Final Presentation

Team 9
Randy, Yu, Zhuojun & Wales

Project Overview

- Neural Network with forward propagation implemented on hardware and backward propagation on MicroBlaze
- This Neural Network will be able to recognize handwritten digits using the MNIST dataset to train it
- The Neural Network will be conducting forward propagation throughout 2 layers



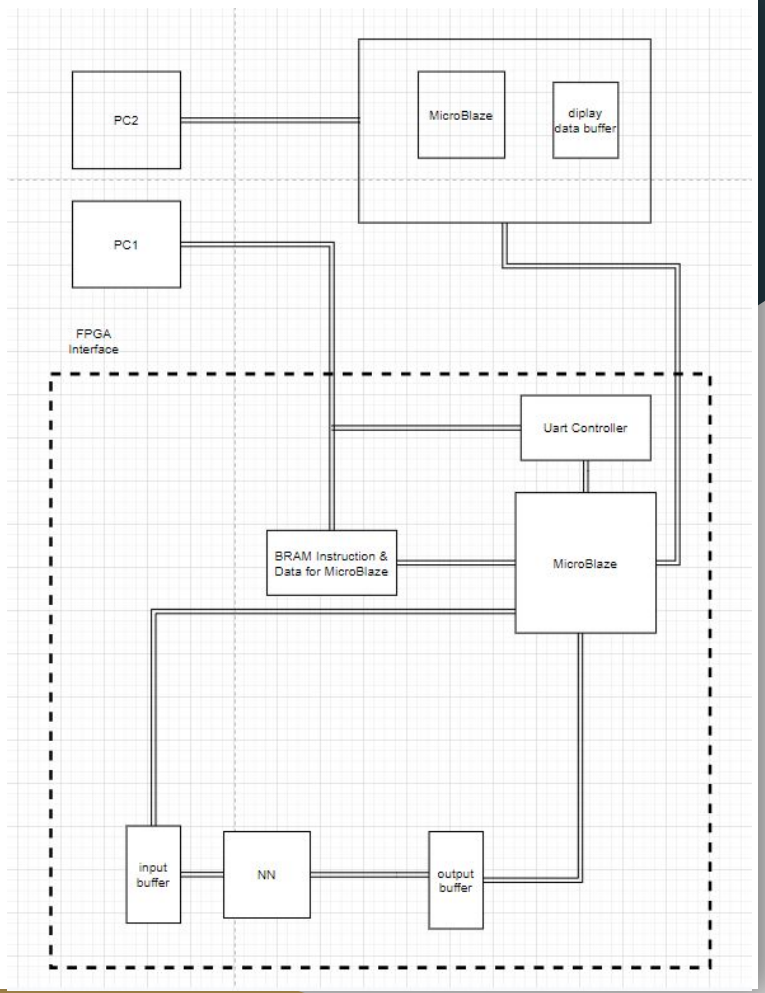
Initial Goals

Original Goal:

- A (784-800-10) Neural Network built on hardware able to recognize handwritten digits from the MNIST dataset
- The First FPGA will send performance and accuracy results to the second FPGA through the network to help with limited resources on the first FPGA
- Neural Network performance and accuracy will be visualized on a second PC connected to the second FPGA
- Backpropagation will be done in software on the MicroBlaze to train the Neural Network

Potential Future Implementation Goal (From Proposal):

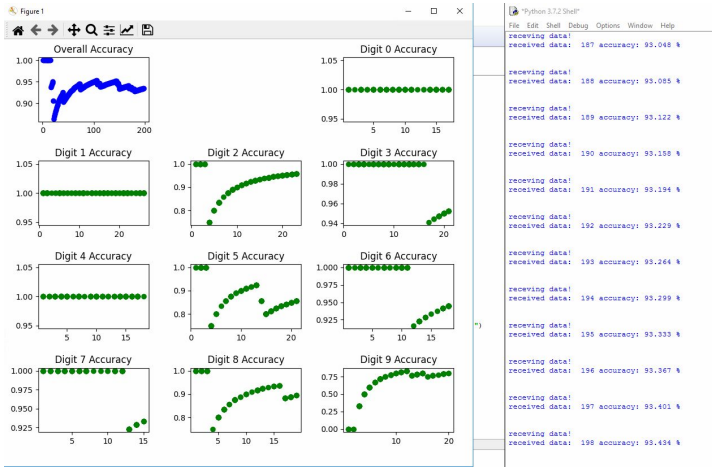
- Have one server FPGA to control flow from Client FPGAs and send their request to processing FPGAs with the Neural Network
- Another FPGA on the network with receive analytical data from the server FPGA and visually present it through a Python program on a PC
- Back Propagation implemented on hardware instead of MicroBlaze



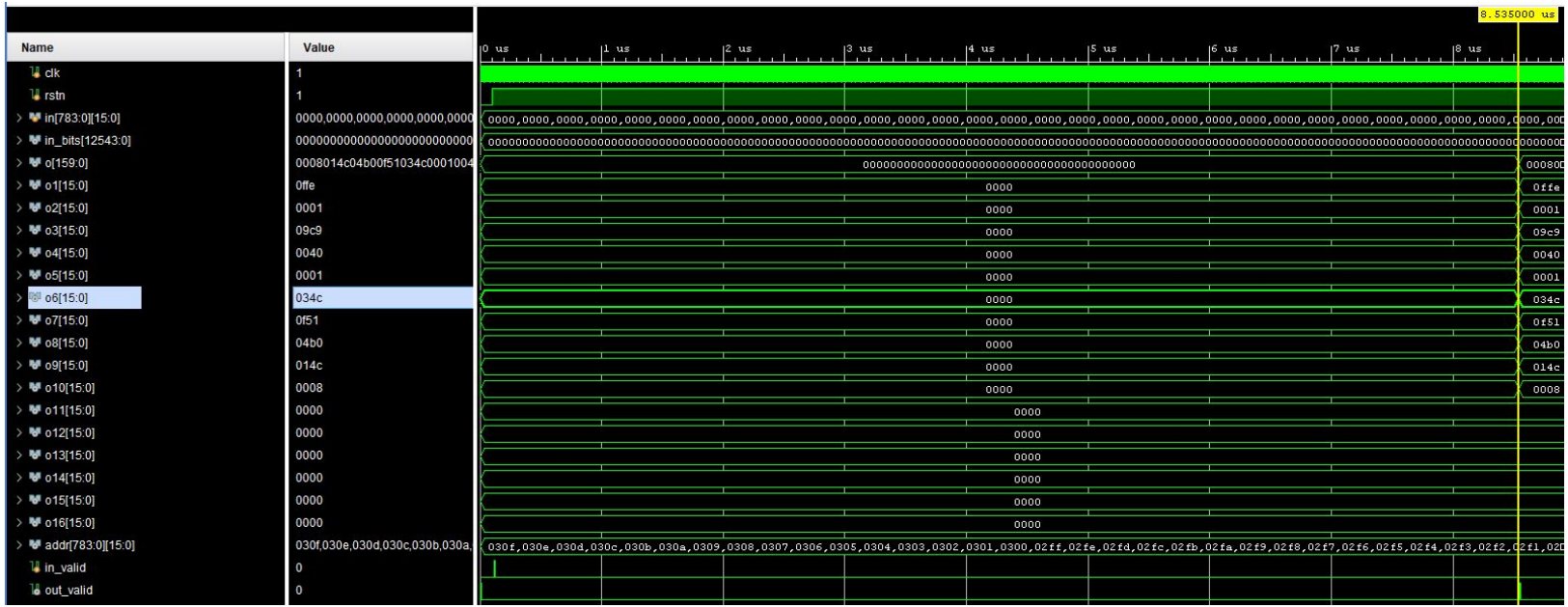
Final Accomplishment

- Successfully implemented a 784-16-10 Neural Network on Nexys Video Board, with pre-trained weights and biases
- Around 8 us for the Neural Network to process one image (100 MHz Input Clock Frequency)
- Sending the test images to the Neural Network through DMA communication
- Sending Neural Network labels & output results to the 2nd FPGA in real-time through TCP
- Visualizing overall accuracy and digit accuracy in real-time using PySerial

```
DMA Transmit Successful, Test Image 10000 Sent
Neural Network Digit Weights: 278  0
Neural Network Digit Weights: 1  1
Neural Network Digit Weights: 113  2
Neural Network Digit Weights: 4  3
Neural Network Digit Weights: 1533  4
Neural Network Digit Weights: 1720  5
Neural Network Digit Weights: 4095  6
Neural Network Digit Weights: 1  7
Neural Network Digit Weights: 59  8
Neural Network Digit Weights: 1  9
Accuracy Count: 9273    Accuracy: 92
Neural Network Output: 6        Label: 6
```



Neural Network Simulation



Challenges Endured

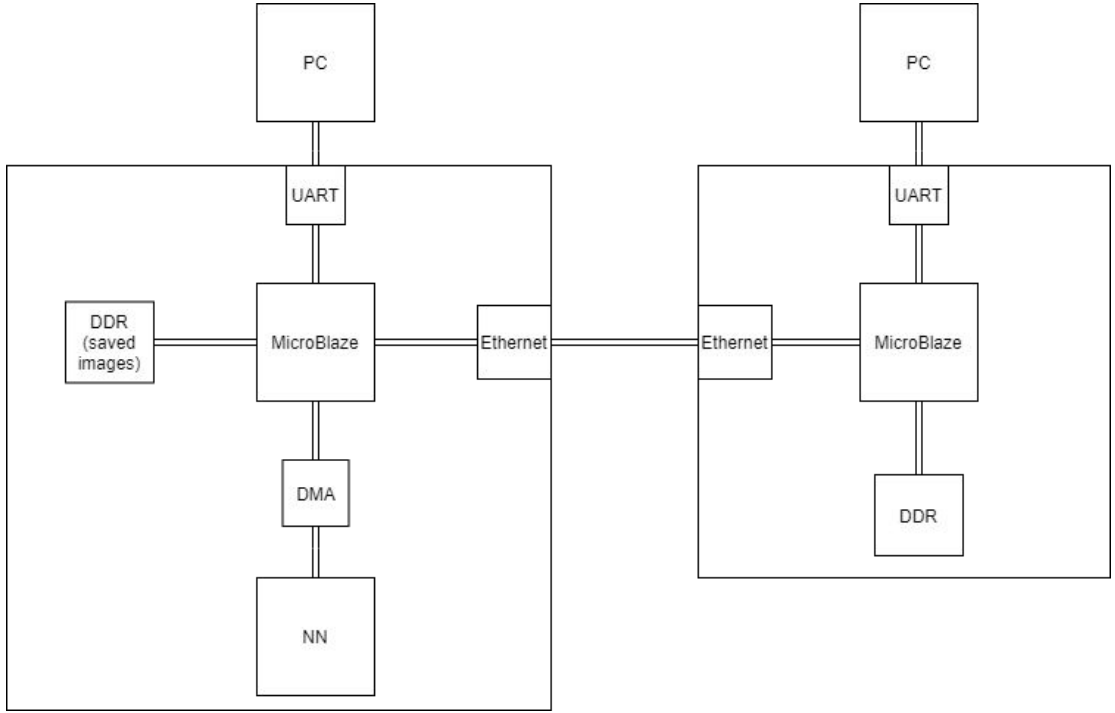
- MicroBlaze limiting performance to handle backpropagation and training
  - Decided to memory map the trained weights and biases into hardware block
- Increasing MicroBlaze performance through Vivado or increasing Input Clock Frequency
- Synthesizing a 784-32-10 Neural Network would use more than 128GB of memory (ECE1373 Dev Machine)
  - More than likely a module that is causing a lot of memory usage when synthesizing
- Transferring data through TCP too often will cause failure on DMA and UART
  - More than likely the shared AXI Interconnect and SmartConnect to the MIG/DDR can cause this

Compromises Made

- Implemented 784-16-10, instead of 784-800-10 Neural Network because of limited resources
- Designed the vector multiplication unit with accumulators instead of an add tree, which increases the processing time
- Due to limited time, we chose to map the pre-trained weights and biases to the hardware block instead of training in SW or HW
- Chose to do more processing on the First FPGA than the Second due to TCP/DMA issues
- Remove Instruction and Data Cache enablement for TCP

Final Block Diagram

- 1st PC sends the image file and label file to the 1st FPGA
- 1st FPGA running Neural Network sends test labels and Neural Network output to the 2nd FPGA
- 2nd FPGA receives data, compares with the test labels, then sends the result to PC
- 2nd PC runs visualization program through Python



Design Process

Hardware (Yu & Zhuojun):

- Beginning with the designs of Sigmoid, ReLU and VecMult, and Neural Net module in Verilog
- Completed the post-implementation simulation to check their functionality from the waveform
- Created an AXI-stream wrapper that wrapped the whole Neural Network block

Software (Randy, Wales & Yu):

- Implemented the Neural Network (Logic Function & MNIST Handwritten Digits) on MicroBlaze in C
- Tested TCP protocol communication between two FPGAs

Integration (Randy, Wales & Yu):

- Send data from PC to MicroBlaze through UART
- Tested the Neural Network output with Microblaze and DMA, examined the output data with ILA and terminal output
- Add the TCP transfer program to the system, examine the received data on the 2nd FPGA
- Add the visualization program to the 2nd PC, examine the received data and its accuracy

Referenced Code/Blocks

Everything was made/modified ourselves with help from the following links:

- Help from YouTuber Vipin Kizheppatt's videos to design and test UART communication between PC & MicroBlaze and DMA communication between MicroBlaze & custom IP block (https://www.youtube.com/user/TheVipinkmenon)
- Neural Network & Deep Learning (https://medium.com/analytics-vidhya/building-neural-network-framework-in-c-using-backpropagation-8ad589a0752d)
- Fixed-Point Arithmetic (https://projectf.io/posts/fixed-point-numbers-in-verilog/)
- Neural Network on FPGA (https://arxiv.org/ftp/arxiv/papers/1711/1711.05860.pdf)
- Matrix Multiplication (http://www.seas.ucla.edu/~baek/FPGA.pdf)
- Sigmoid Function in Verilog https://www.researchgate.net/publication/224843989_IMPLEMENTATION_OF_A_SIGMOID_ACTIVATION_FUNCTION_FOR_NEURAL_NETWORK_USING_FPGA

What We've Learnt

- Digital system design in Verilog has been improved
- AXI-Stream protocol understanding
- TCP & DMA data transfer on MicroBlaze
- Vivado workflow
- Debugging the hardware & system design through unit testing & system testing
- Communicating together being 12 hrs (Canada & China) apart

DEMO TIME!