Hardware Approach of Forward Propagation in Neural Network on FPGA

Team 9
Randy, Wales, Yu & Zhuojun

XILINX

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

What Makes it Multi-FPGA

Current Implementation Plan:

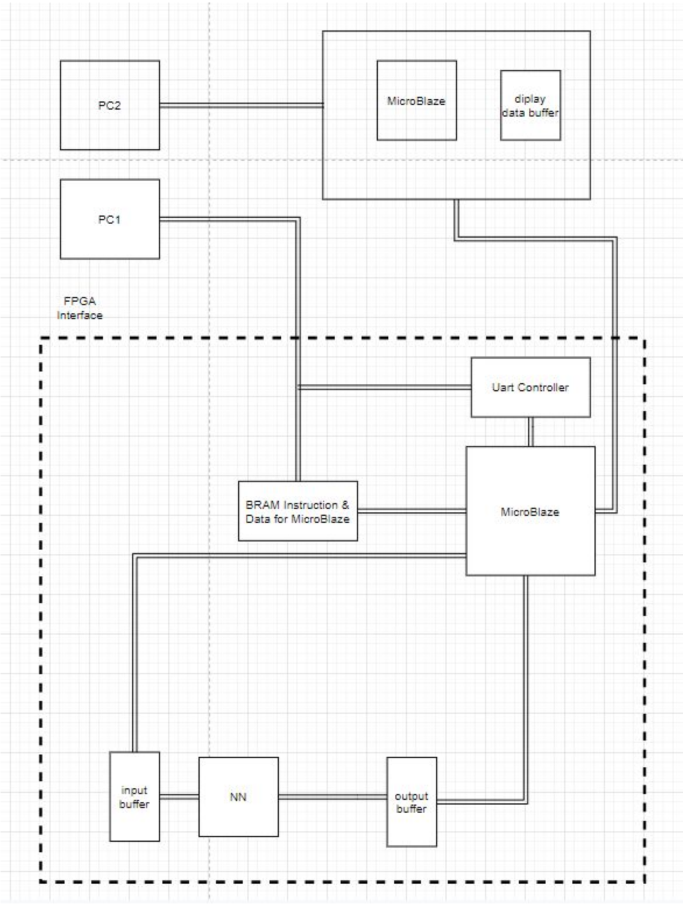
- First FPGA will send performance and accuracy results to the second FPGA through the network to help with limited resources on the first FPGA
- Neural Network performance and accuracy will be visualized on a second PC connected to the second FPGA

Potential Future Implementation Plan:

- Have one server FPGA to control flow from Client FPGAs and send their request to processing FPGAs with the Neural Network implemented on
- 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

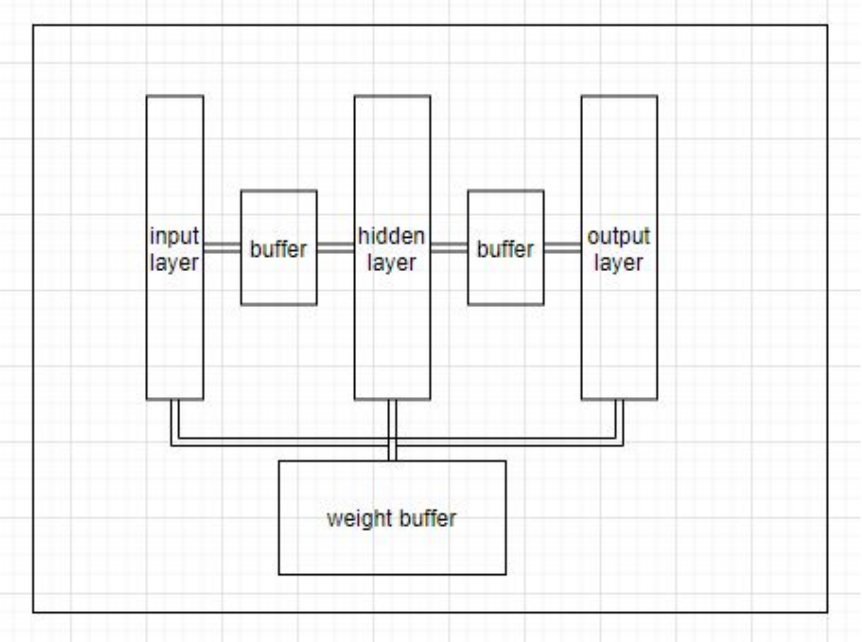
System Diagram (Hardware)

- The data will be stored in a BRAM buffer
- Neural Network block will retrieve the input data from the buffer
- Output data, as well as the target output (expected output), will be stored in a BRAM buffer
- The MicroBlaze will retrieve the data from the BRAM buffer and perform backpropagation, updated weights are stored in another BRAM buffer



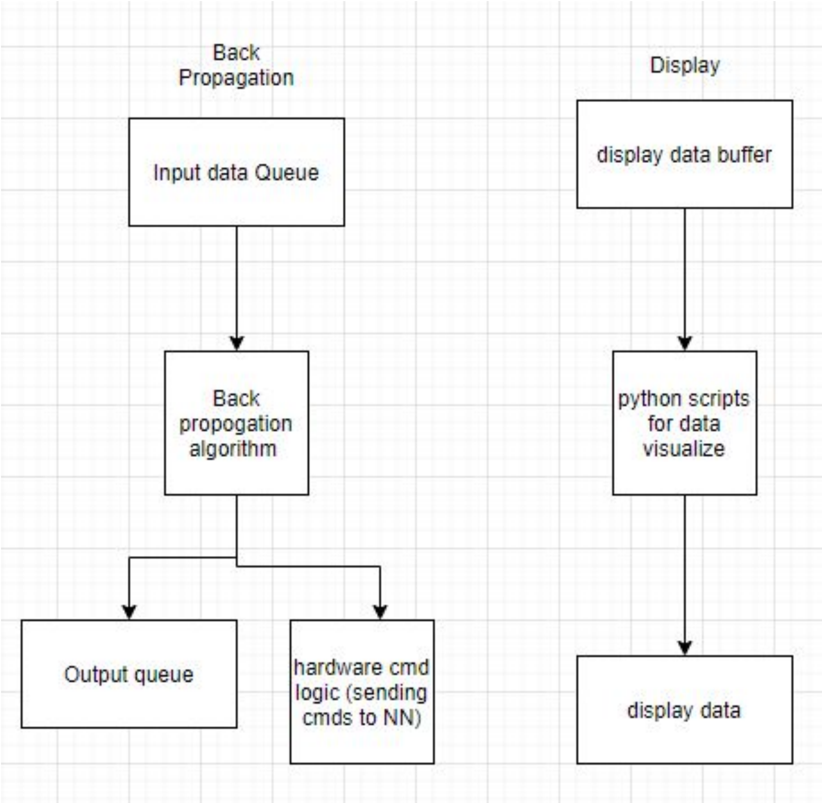
System Diagram (Neural Network)

- Multi-Stage 2 -Layer Pipelined Architecture
- Forward Propagation will be performed from the input layer of the Neural Network to the output layer, through fully connected layers
- Each layer output to a temporary buffer
- Each layer connected through AXI4-stream
- Weights stored as a matrix



System Diagram (Software)

- Back propagation is conducted using MicroBlaze through software
- Neural Network block receives a signal indicating weights are to be updated, retrieve the weights from the buffer
- Cycle through the previous process, then after certain numbers of runs, send the data to a BRAM buffer for monitoring display
- The results will be collected by the monitoring unit (in our case, it will be on another FPGA board)



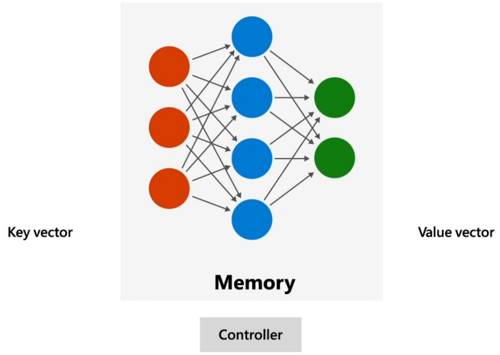
Designed IP & Leveraged IPs

Designed IP:

- 2-Layer Neural Network (784-800-10)
    - Design the input layer, hidden layer & output layer
        - Within each layer design the neurons
    - Design sigmoid & softmax stages between the 2-Layer network

Leveraged IPs:

- Softcore MicroBlaze
- AXI4 (mainly AXI-stream, e.g., FIFO converter IPs)
- UART
- BRAM
- EthernetLite



Implementation Plan

- MicroBlaze design handle communication and control flow between
  - PC & DDR Memory
  - DDR Memory & Neural Network IP
  - DDR Memory & Second FPGA

- Neural Network IP Core
  - Initially designed as 2-layer (2-3-1) using logic functions (AND, OR, XOR, etc)
  - Move to handwritten images (784-800-10)

- Python program on 2nd PC to handle data visualization analysis of Neural Network
  - Communicate using UART or TCP/IP to get data from second FPGA
  - GUI design and data interpretation

Unit Testing & Validation

- TCP/IP connection between PC and FPGA, FPGA and FPGA
- UART data transmission between host PC and FPGA
- Neural Network block testing: ability to read from buffer, ability to output to a buffer, the ability to update weights, etc
- Back Propagation unit testing: the ability to update weights correctly, retrieve/output data, etc
- Monitoring unit testing: display meaningful results and collect statistics

System Testing & Validation

- Training Neural Network
- Testing & verifying accuracy with handwritten images sent from PC using Python program

Pretty Much Make Sure It Works All Together :)

Uncertainties & Open Questions

- Resources required on the FPGA
  - Will a large number of AXI FIFO IPs to potentially cause Clock Domain Crossing (CDC) issues?
  - Will there be enough slices in the FPGA for the Neural Network?

- Will the MicroBlaze be able to handle our requirements implemented through software?

- Pipeline and non-pipeline Neural Network?

- How to interpret analytical results in Python program?

Risks & Fall-Back Plan

- Unfamiliarity with the structure and interfaces between different hardware blocks of our team may cause unexpected data transferring failure

- Those complex features can also be alternated by changing from difficult implementation to easier ones

    - Weights are pre-trained on PC

    - Single layer implemented on hardware only; the rest done on MicroBlaze

    - Rely on TCP/IP communication

Questions?