University of Toronto

ECE532H1 Digital Systems Design

Project Proposal

Hardware Approach of Forward Propagation in Neural Network on FPGA

Group 9:

Yu Gao	gaoyu14
Zhuojun Yu	yuzhuoj1
Wales Zhou	walesz
Randy Ewing Chow	ewingcho

January 30, 2021

1. Introduction

Neural networks and deep learning currently provide probably one of the best solutions for many applications like image recognition, speech recognition and natural language processing. In traditional programming, the software is designed with little or no hyperparameters, and the whole program runs in a decided fashion. The advantage of a neural network is it does not need us to tell the computer how to deal with the problem, but by learning from observing data, it can generate its solution. Learning from data automatically looks promising. Although this is a powerful tool, it requires significant computing powers and is time-consuming. Under these circumstances, people are interested in implementing the system on a hardware platform like GPU, TPU, or FPGA.

2. Existed Similar Work & Discussion

2.1 OpenCL Method [1]

Some successful experiments of implementing machine learning on FPGA have made promising results. For example, using OpenCL to design convolutional neural networks on Intel FPGA. It can classify 50k images at a speed of 500 images per second [2]. This gave us inspiration for implementing neural network forward propagation on Nexys DDR board.

Pros: Efficient design with high throughput, easy to use and a powerful machine learning tool.

Cons: Only for CNNs and not flexible.

2.2 General Neural Network Hardware Architecture [3]

There was also a group of students from the University of Birmingham doing a project on general neural network hardware architecture on FPGA.

Pros: Pipelined architecture, each stage has a buffer and which is easy for reuse.

Cons: The efficiency of backpropagation in the design is unclear, it is doubtful whether it is faster than PC or not.

2.3 Pipelined OnLine Backpropagation [4]

“Artificial Neural Network Implementation on a single FPGA of a Pipelined OnLine Backpropagation”.

Pros: Maximum capability of parallelism (parallel execution of forward and back propagation and pipelined parallelism of data inputs).

Cons: Not suitable for a prototype design, maybe could later be implemented when the whole system is finished.

2.4 Discussion

Our goal is to implement a neural network, forward propagation, and backpropagation algorithm in digital hardware, particularly in this case, an FPGA platform. This hardware approach utilizes the advantage of the inherent parallelism of neural networks.

3. Biography of Project Team Members

3.1 Group member: Yu Gao.

Strengths:

a. Former experience in programming microcontrollers (Arm Cortex, Atmega, etc).
b. Basic knowledge in digital hardware design (VHDL, FPGA).
c. Networking knowledge (TCP/IP, MAC).

Weaknesses:

a. Not enough knowledge in Verilog.
b. No former experience in digital communication protocols like AXI.
c. No former experience in designing system-level digital hardware.

3.2 Group member: Zhuojun Yu

Strengths:

a. Relatively more experience in Verilog.
b. Good understanding of basic digital IC design techniques and methodology.
c. Have experience of AMBA protocol to some extent .

Weaknesses:

a. Almost no experience with high-level language like C.
b. Don't know much about Internet protocols and cutting-edge technology such as ML.
c. Not enough understanding of computer architecture, e.g., the data flow inside chips.

3.3 Group member: Wales Zhou

Strengths:

a. Has some experience in Verilog development with STM32F4 board.
b. Has fundamental knowledge in digital circuit design.
c. Work experience with some industrial communication protocols.

Weaknesses:

a. Has no experience with communication between FPGAs.
b. Limited understanding of machine learning and neural networks.

3.4 Group member: Randy Ewing Chow

Strengths:

a. Experienced in automating tasks and conducting system-level testing
b. Can pinpoint and debug issues that occur in the design and breakdown the issues at hand
c. Skilled in coding which includes C & Python coding languages and can integrate various libraries or pieces of code to meet the required design goal
d. Creative and can find workarounds and find ways to improve a design to improve the xexperience of the end-user

Weaknesses:

a. Limited knowledge on using FPGAs for NNs
b. Little experience with Verilog coding and digital design

Some of the members in this group lack experience in digital system design. We are expecting to build up skills in Verilog during this project. Another weakness some of us have is we don't have experience in neural network development. Our group hopes by the end of this project, we will have a solid understanding of neural networks.

4. Project Description

Implement a hardware architecture on FPGA with forward propagation. This neural network can perform simple pattern recognition. The input data we have chosen to use are hand-written digits, which will be easier to test, and the NN itself doesn't need to be very large.

4.1 Requirements & Features

- FPGA-FPGA and PC-FPGA communication using MAC/IP packets
- UART transmission (over RS232) between PC and FPGA for data transmission
- NN core with a multilayer neural network (input layer, hidden layer, softmax layer, output layer)
- Back Propagation algorithm implemented on the MicroBlaze soft core
- Monitoring system for displaying training results

The overall system block diagram is shown below. As we can see each of the subsystems mentioned above is shown here with a few extra blocks. A BRAM buffer is used here to store the temporary results from the neural network (NN). A data pre-processing unit might be required for the input data (typically images) to become usable for NN. The BRAM instruction & data unit does not need to be implemented by us, since this is just an explanation that we will program the MicroBlaze from a PC.

4.2 Project Details

We decided to build the whole system in a pipelined fashion, with buffers in between different stages.

The overall pipeline of this digital system:

- Software and data (for NN) will be downloaded to the Nexys DDR board, which a soft core (MicroBlaze) will be running on
- The data will be stored in a BRAM buffer
- NN block will retrieve the input data from the buffer
- Forward Propagation will be performed from the input layer of the NN to the output layer
- 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
- NN 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 could be on another FPGA board)

4.3 Block Diagrams

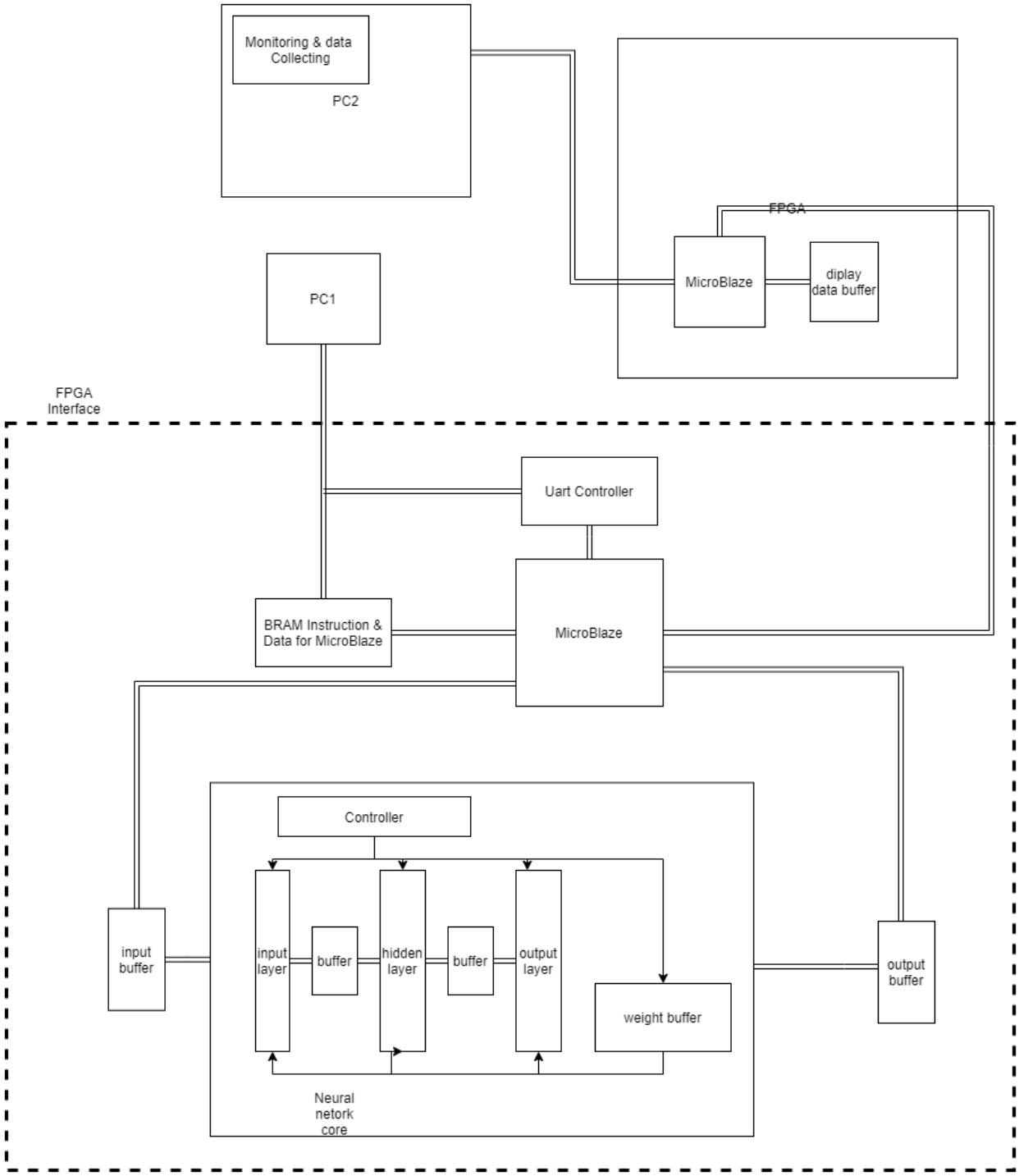

Figure 1. Block diagram for hardware

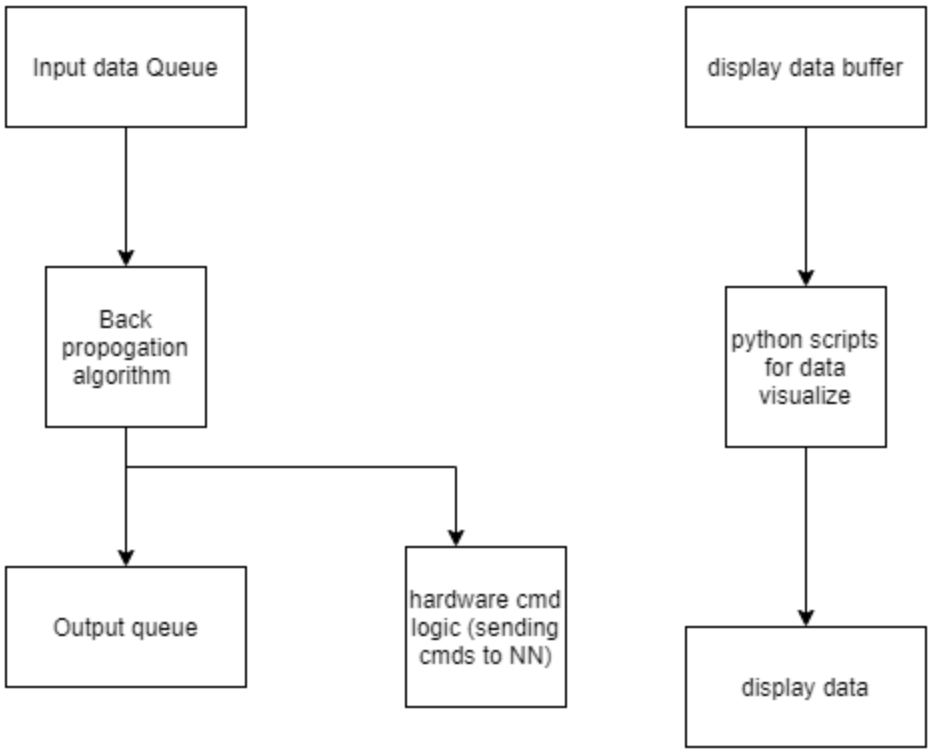

Figure2: Block diagram for software

5. Testing

The approach we decide to take is unit testing first, then integrate blocks and perform system testing.

Finished Testing:

- TCP/IP connection between FPGA and PC, FPGA and FPGA
- UART data transmission between host PC and FPGA

Further Testing:

- NN block testing: NN controller, 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 weight correctly, retrieve/output data, etc
- Monitoring unit testing: display meaningful results and collect statistics

6. Project Complexity

Data transfer over USB	Transfer data from Desktop to FPGA using UART + MicroBlaze + python script (0.25 points)
Desktop to FPGA network connection	Connect using TCP/UDP from python script (0 points)
FPGA network connectivity	Connect to network with raw IP/MAC packets using MicroBlaze (+ LWIP?) (0.25 points)
Other Peripherals	SD card access using MicroBlaze (0.75 points)
IP Core implemented in FPGA Hardware	Difficulty scales with complexity of core (1 points)
Performance monitoring	Implement performance monitoring in MicroBlaze (0.10 points)
Software algorithm implemented on MicroBlaze	Difficulty scales with complexity of algorithm (0.6 points)
Meaningful visualization of program run/statistics gathered/results obtained (i.e. Marketing)	Print on terminal something more meaningful (0.25 points)
TOTAL:	**3.2 points**

7. Risks

Compared to software simulators, FPGA has a much slower clock frequency. On the other hand, unfamiliarity with the structure and interfaces between different hardware blocks of our team may cause unexpected data transferring failure.

The processing element of the neural network is limited by FPGA's logic density. In this case, a complex multilayer neural is very hard to achieve on FPGA. Since we have several FPGA on the network, we can have this neural network implemented on two FPGAs to build a small layer of neural.

For buffer week, the reading week would be a good choice for us to mitigate the pressure if the beginning doesn't go on well. Those complex features can also be alternated by changing from difficult implementation to easier ones, e.g., Internet protocols.

8. Resource Requirements

Some requirements and features can be FPGA-FPGA and PC-FPGA communication using MAC/IP packets; UART transmission between PC and FPGA for data transmission; ML core with a multilayer neural network (input layer, hidden layer, softmax layer, etc); Monitoring system for updating training results; SD cards accessing.

9. Milestone

9.1 Milestone #1: Research

- Yu Gao: Research on the hardware implementation of NN, how to write a MAC/IP software on Microblaze
- Zhuojun Yu: Research on the concept of NN, AXI protocol
- Wales Zhou: Research on python scripting and visualization tools, Verilog programming
- Randy Ewing Chow: Research on data sets in training the NN for handwritten digits and look into how to send an image from PC to FPGA

9.2 Milestone #2:

- Yu Gao & Randy Ewing Chow: Build the NN block top-level (mainly the input, the output of the whole block, as well as the sub-blocks) in Verilog, including the datapath and controller
- Zhuojun Yu: Testing the AXI protocol, implementing the buffers and the communication between MicroBlaze and several buffers, meanwhile get familiar with MAC/IP
- Wales Zhou: Designing simple visualization programs in python, including producing statistics, and generating graphs
- Randy Ewing Chow: Find and implement a small, memory tolerant data set with a Python-based NN implementation to be used to train our NN

9.3 Milestone #3:

- Yu Gao & Randy Ewing Chow: Continue building the NN block, focusing on writing Verilog for sub-blocks, as well as building the controller part
- Zhuojun Yu: Building up the whole pipeline-architecture for the system
- Wales Zhou: Modify, and improve the visualization program, meanwhile learning Verilog and help Zhuojun build up the system
- Randy Ewing Chow: Be able to send handwritten digit image over to MicroBlaze and verify it is stored on the memory

9.4 Milestone #4:

- Yu Gao & Randy Ewing Chow: Continue building the NN block, as well as implementing the MAC/IP communication
- Zhuojun Yu: Connecting the system and testing it, as well as help Yu finishing the MAC/IP protocol
- Wales Zhou: Modify, and improve the visualization program, meanwhile learning Verilog and help Zhuojun build up the system

9.5 Milestone #5:

- Yu Gao & Zhuojun Yu & Randy Ewing Chow: Simulating the whole system, and try implementing it on Nexys boards
- Wales Zhou: Receiving data output from one Nexys buffer, displaying the results and error checking

9.6 Milestone #6:

Yu Gao, Zhuojun Yu, Wales Zhou & Randy Ewing Chow: Finishing the project, preparing for the final demo

---

[1] Intel.com, "CNN Implementation Using an FPGA and OpenCL™ Device"

https://www.intel.com/content/www/us/en/products/docs/storage/programmable/applications/machine-learning.html#:~:text=Neural%20networks%20are%20inspired%20by,in%20particular%20the%20human%20brain.&text=FPGAs%20are%20a%20natural%20choice,resources%20in%20the%20same%20device.

[2] intel.com, A video link for demo: https://www.intel.com/content/dam/www/programmable/us/en/video/convolution-neural-network-implementation-on-altera-fpga-using-opencl.mp4)

[3] Yufeng Hao. "A General Neural Network Hardware Architecture on FPGA"

https://arxiv.org/ftp/arxiv/papers/1711/1711.05860.pdf

[4] Rafael Gadea , Joaquín Cerdá , Franciso Ballester , Antonio Mocholí. "Artificial Neural Network Implementation on a single FPGA of a Pipelined OnLine Backpropagation "

http://www.cecs.uci.edu/~papers/compendium94-03/papers/2000/isss00/pdffiles/11_5.pdf