CS152B Final Project: Integer Identification

Group 3: Michael Hale, Matthew Nuesca, Shilin Patel

Project Introduction

Proposed Project: Convolutional Neural Network

- Train on MNIST dataset
- Identify integer digit from drawing

Interesting?

- Neural networks are used everywhere
- High industry demand for experts, understanding how FPGAs can be used to improve CNNs is useful

Purpose? Training performance improvements with custom hardware

Features Developed?

- Serial Communication for Image Transfer
- Multiple Neural Network Layers

Real-World Constraints

Constraints:

- High-Speed Clock Generation
- Modules for Fixed Point Computation and BRAM
- Serial Communication Hardware, Seven Segment Display

Costs: Economic: Board, SevenSeg, Serial Device (Laptop)

Manufacturability: Nothing unique to manufacture

Socio-Political Fit: Not breaking the status quo

Sustainability: General purpose CPU architecture commonly used for training neural networks is limited in performance. Custom designed hardware systems are expensive to produce at scale. Our solution is viable for cost-aware, performance heavy users

Industry Standards

Convolutional Neural Network: Convolutional, Fully-Connected, Softmax, ReLU layers

FPGA Block RAM: Storing weights on device

Serial Communication: RS232 Communication

Seven Segment Display: PMOD 12-Pin Connector

Training Data: MNIST image dataset

Test Data: Python program to draw image and convert to byte array

Costs

Development:

- Human costs in time to design and build the product
- Cost of FPGA, Serial Hardware, Data Transmission Device

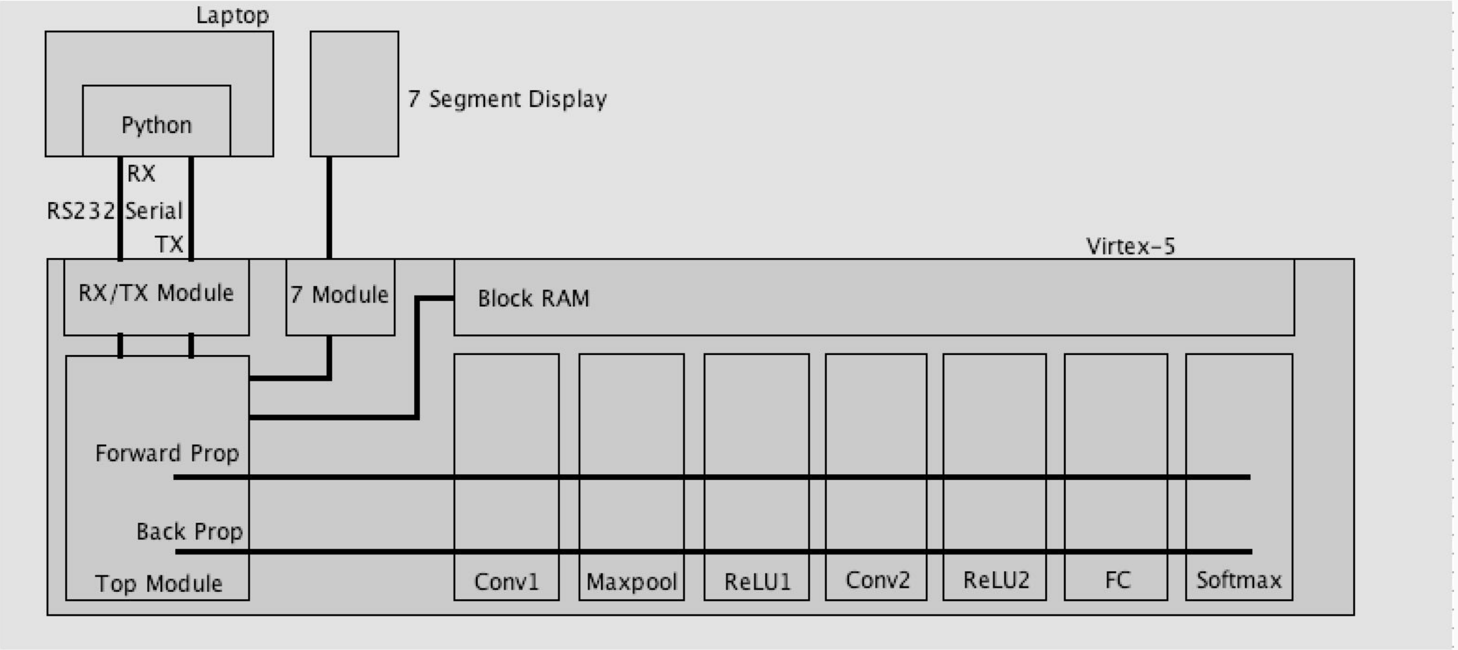
Marketing:

- Strong competition from large companies such as Amazon, w/ built in cloud services
- Would not work as standalone product
  - Need to sell technology or personal skills in area
- Marketing costs would be high

Project Description

Constraint Based Implementation: We can continue as planned, no constraints apply

Block Diagram:



Design Architecture

- Each layer has a series of 4 BRAM cores:
    - Last input -- used to update weights
    - Weights -- forward & backpropagation
    - Error -- backpropagation
    - Output value -- updated continuously upon receiving inputs
- In/Out Valid and Ready signals between layers
    - Controls flow between layers
    - Layer A will hold the output value until Layer B acknowledges it with a ready signal
- Each layer modeled with a state diagram:
    1. Receive forward input
    2. Perform forward computation
    3. Output forward values
    4. Receive output error
    5. Perform backprop updates
    6. Output error to input layers

Algorithms

Serial Communication:

- Communication between PC and FPGA
- Two directional RX/TX for streams of 784 bytes per transfer

Convolutional Neural Network Layers:

- Convolutional
- Maxpool
- ReLU
- Fully Connected
- Softmax

Computation Tasks

- Fixed-size gradient descent
- Fixed Point multiplication and addition

FPGA

Base System:

- Xilinx Virtex-5 FPGA

Hardware Peripherals:

- Block RAM: Weight Storage
- Seven Segment Display (PMOD): Display Test Data Result
- RS232 Module: Send ready signal and receive image byte arrays

Algorithm Implementation:

- Fixed Point Arithmetic Module
- Neural Network Layer Modules
    - Each layer is a module which interfaces with the top module

Software

Algorithm Implementation:

- Python
- PC end of serial communication
  - PySerial to receive ready signal and send byte arrays
- Downloaded MNIST data and labels loaded using mnist
- Image drawing for test data
  - Tkinter, Pillow
    - User draws image
    - Resize to 28x28 and convert to byte array for easy serial transfer

Challenges

Device Constraints

- Limited number of connections to memory
- Core Generator reset pin does not work consistently
- Low number of Arithmetic modules (multipliers)
- Broken DDR2 SDRAM

Lack of Implementation and Debugging Tools

- Only able to synthesize in this lab
- Core generators only able to simulated in Xilinx (in the lab)
- ISim provides buggy waveform generation in computationally heavy test benches