Jordan Dickson
65582692

FPGA Accelerated System for Template Recognition

# Problem:

Certain machine learning algorithms often rely on making numerous matrix multiplications to determine the output vector for the system given an input vector. Matrix multiplications are CPU intense operations, but they can be easily parallelized.

# Proposal:

The FASTR project will attempt to leverage the hardware speeds of the FPGA on the PYNQ board to accelerate and parallelize the matrix multiplications. We will create (or find) an overlay for the PYNQ board that allows the processor to offload and parallelize most of the computation for matrix multiplication.

For comparison, a software-based library will also be used to perform matrix calculations. Each solution will be given similar problems to measure the performance difference between the two approaches.

This project will use either image processing or machine learning example applications for this solution.

# Supplies:

This project will require minimal supplies:

- Xilinx PYNQ development board
- 8 GB microSD card loaded with the Linux image for PYNQ
- Ethernet cord

# Current Project Status:

All the necessary supplies for this project have been purchased and have arrived. The PYNQ board has been tested with a few example programs, and everything seems to run as expected. We have been able to compile and run C code on the board using the gcc compiler built into its Linux image.

Some prototyping has been done for the matrix library, but we're also looking for existing libraries so we can get a more accurate representation of a "standard" software solution for matrix calculations.

Some research has been done to determine how to make custom programs for the FPGA onboard the PYNQ.

# Remaining Steps:

The exact case applications that the matrix calculators will be tested on need to be selected.

A software solution to matrix multiplications needs to be selected. One can be created from scratch or sourced from an existing library.

We still need to find a way of programming the FPGA. It looks like we'll just need to create a custom ".bit" file and then interface with it using buffers, but we still don't know how to get started there.