Cellular Neural Network FPGA Implementation Report

Research - Dr. Hyesoon Kim

01 May, 2015

Enmao Diao

Ria Gupte

Sangeetha J.

Joshua Lee

Introduction

The goal of this research project is to do FPGA implementation of CNN algorithm for successfully pursuing image processing. The idea is to implement the algorithm in two ways- multiple multipliers and adder and shared multiplier and adder. The first approach takes lesser time but the second approach takes up lesser hardware resources. The team has successfully completed both implementations in Verilog.

Team work

Enmao Diao

I finished reading the paper during the winter break. During Spring 2015, I started to get familiar with CNN by doing schematic implementation using Quaruts. Initially, schematics is the most familiar approach for me. After struggling to link the nodes of each module for a week, the professor pointed out that it is not viable to implement and debug large dimension CNN using schematics. Then I started to learn Verilog, a digital circuit design tool. Although I have learned a little VHDL from ECE 2031, Verilog seems much easier to understand. I set up a gitHub account at that time and include my previous schematic implementations as release for record consideration. I finished the 4x4 CNN Verilog implementation after spring break. The major barriers I encountered at that time include the bit width and signed or unsigned decision of input and intermediate signals, simulation in Modelsim, and one clock delay of passing the output from the lower level module (this seems naive for me right now). Next I started to expand the 4x4 to 16x16 CNN. I finished that one week before the dead week. The major barrier is to write the module that links the center cell to all its neighbours. This took me more than two days to consistently debug and simulate.

Mostly, I built up a little every day during class recess.My main focus in the future will be Digital Signal Processing. This research experience introduces me to the world of CNN and the more important image processing. In the future, I will start to finish Sangeetha's implementation of sharing one multiplier and look up the a new paper which aims to find new templates for CNN.

Ria Gupte

Initially started out the project by reading up on the CNN research paper. Using the paper as a starting point, successfully wrote 1 by 1 CNN naive based code in Verilog. Started the 3 by 3 matrix but was not able to complete it. Commented onebyone.v, eq1.v, eq2.v, fourbyfour.v, CNN.v and sixteenbysixteen.v code on GitHub. Also made sure the equations on the code were correct by comparing it to the ones in the CNN research paper.

Sangeetha J.

I began with the familiarization of CNNs and the SP-CNN proposal implemented in the research paper using MATLAB. The MATLAB replication provided us the basic understanding and program flow in with a CNN worked.

I took up the implementation of a basic CNN array which shared the Multiplier and Adder units. Such a 4x4 CNN array was implemented in verilog using Quartus and simulated using ModelSim.The shared multiplier used a pipelined model with the multiplier and the Adder units as the two stages. This method took 15 times more time than the naive method of implementation of the CNN for a 4x4 array.

Joshua Lee

The familiarization of the the implementation of the SP-CNN took up the majority of the time of undergraduate research. After reading through and understanding the given CNN research paper, a Verilog naive CNN was attempted; only the equations could be initially completed. Because of the lack of knowledge of using Verilog, a MATLAB replicate was also attempted based on a provided reference. The MATLAB attempt became more of an understanding of the program flow of the provided reference. As a result, the code created by Diao on GitHub was utilized in order to familiarize with Verilog as well as the Verilog version of the naive 16-bit SP-CNN.

Summary

Currently the team has a working model of sixteen by sixteen using multiple multipliers and adders. The shared memory version of the implementation is also available which uses only one multiplier and one adder. Future work would be to implement the code on FPGA. (a DE5 board is anticipated for the use as the code is computationally expensive)