Garret Huibregtse, Sean Cohen, Austin Wirtz
ECE 551
May 9, 2018

Final Project Report: Simple Neural Network

Our ECE 551 final project was to design and implement a simple artificial neural network (SNN) to read and correctly detect handwritten digits via a 28x28 bitmap input into the network. The result of the digit detection is output onto the LED bar of our FPGA in binary and transmitted to the PC via UART to indicate which digit was detected.

The SNN is comprised of several modules. First, it has a UART receiver that is used to read the input bitmap. It also contains several RAM and ROM modules used to input, store, and output memory. The RAM input module sends the data into the SNN_CORE module, which runs the activation function on the input data to calculate the hidden layer and output layers of the SNN. It does this by using an internal MAC (multiplier-accumulator) module to make the calculation. Once the SNN's output layer is calculated, the index of the bit with the largest contents in the output layer is output as the digit it detects. This digit is then sent through a UART transmitter module to be output to the PC and displayed on an FPGA led bar. All the modules are encapsulated and controlled by a top-level SNN module. Test bench modules were also made for the UART receiver/transmitter modules, the mac module, and the SNN/SNN_CORE modules.

The distribution of work among our group went as follows: Garret completed the SNN_CORE module, some of the RAM/ROM modules, and wrote the synthesis script for the project. Sean completed the SNN module tying the rest of the modules together, wrote the test bench for it, and wrote some of the RAM/ROM modules. Austin wrote the UART receiver and transmitter modules, the MAC, some of the RAM/ROM modules, and the remaining test

benches. All three group members worked on the debugging of the SNN_CORE and SNN modules together.

To increase efficiency by reducing area, power, and time, we employed a few different techniques. In one instance, we were able to replace an adder with a concatenation which is a much simpler circuit and has a smaller area. We also discovered that we could remove the RAM_OUTPUT_UNIT module further decreasing our area and memory usage. Also, by removing that module, we were able to remove some of the overhead logic that we needed to calculate the correct digit. This also slightly decreased our time due to not having to wait for the output of the ram to be read. Another way we reduced our logic area was trying to reuse signals that did the same thing. This includes clear signals for our registers, our UART_TX start signal being asserted by the SNN_CORE done signal, and others.

Due to the limited time we had to optimize our design effectively we did not pursue some of the more difficult and better ways to optimize our design. One of the ways we would have liked to improve our design include: splitting our ram initialization files, adding a second MAC, and calculating the multiplications concurrently. This would have greatly decreased our time yet only slightly increase the area of our design. Since our design only used ~2% of the FPGA's space, this is a very good tradeoff.

Overall this project provided us with great experience designing a project with a loose specification and with lots of room for customization. Working in groups allowed us to piggyback off each other's ideas and help us be more successful. The finished product was even more satisfying after facing the adversity of getting our design to function correctly and we recommend this class for future students.