A Fixed-Point Architecture for Fully Connected Networks in a Convolutional Neural Network (CNN)

Joe Muhle: Michigan Technological University
Devon Schleyer: University of Kentucky
Mentor: Dr. Daniel Llamocca

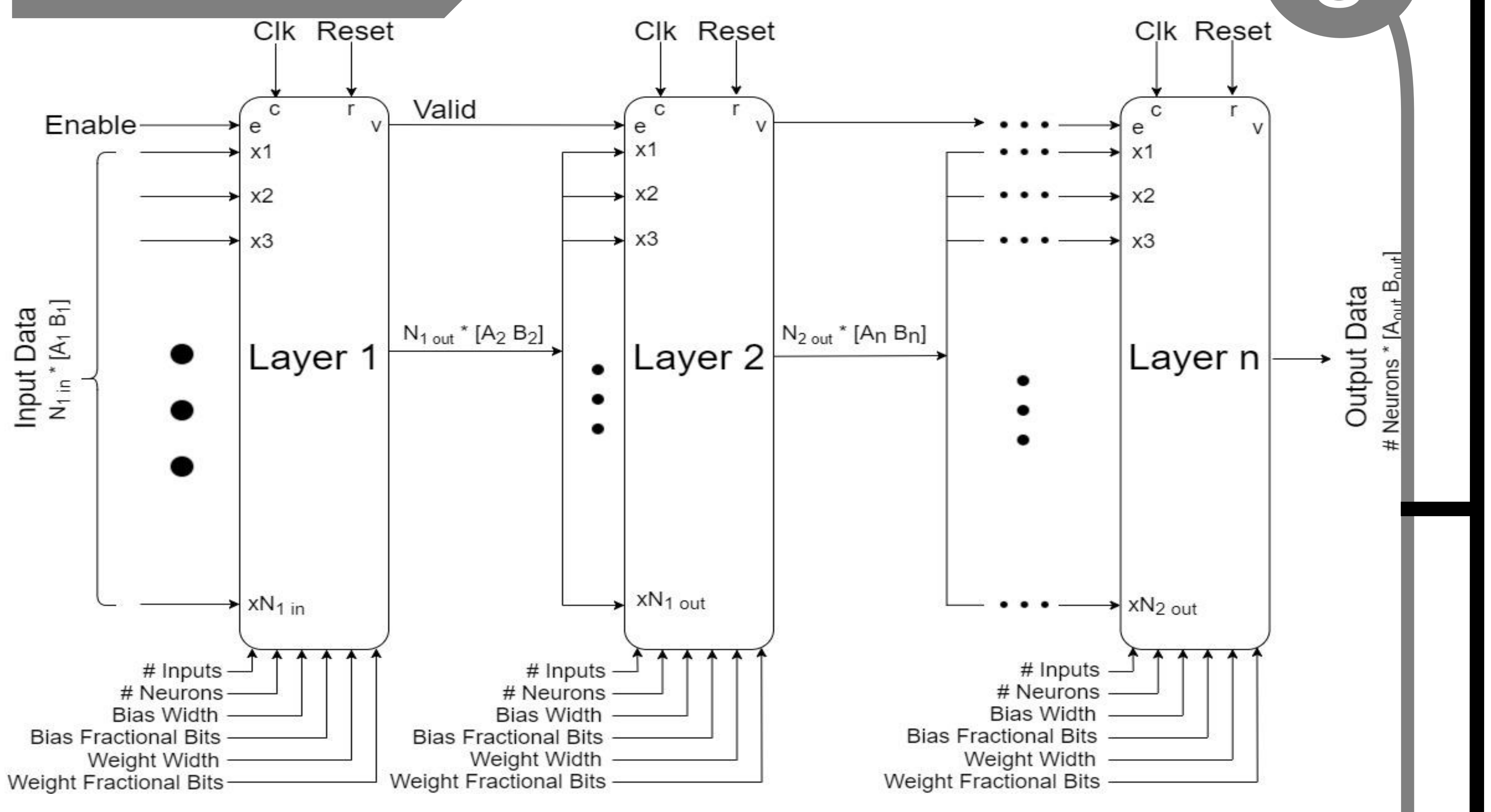
Motivation

1

To design a dedicated hardware architecture in parametric VHDL code that allows the user to specify: # of inputs, # of layers, # of neurons per layer, fixed point format of inputs and weights and biases, as well as the weights and biases themselves.

HW Design

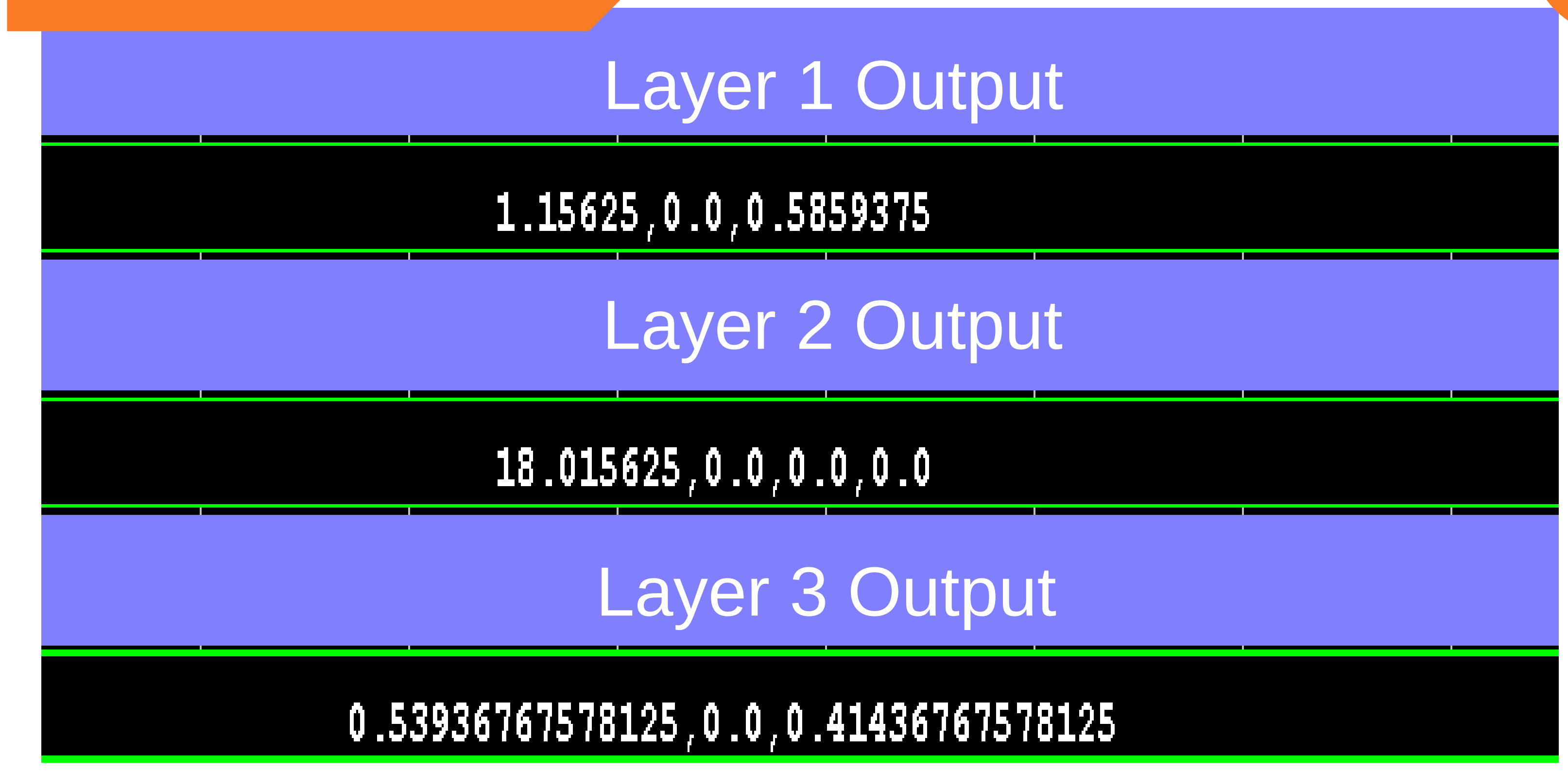
3



This is the structure for a generic neural network. Each layer has several parameters that help make a flexible system, such as the number of inputs, neurons, bias and weight widths, and the fractional bits of the bias and weights. The output of each layer is a large array that gets broken up into several pieces at the input of the next layer. An enable and valid bit help to indicate what values are part of the result for that layer.

Verification

5



Layer 1 Output

```
1.15625,0.0,0.5859375
```

Layer 2 Output

```
18.015625,0.0,0.0,0.0
```

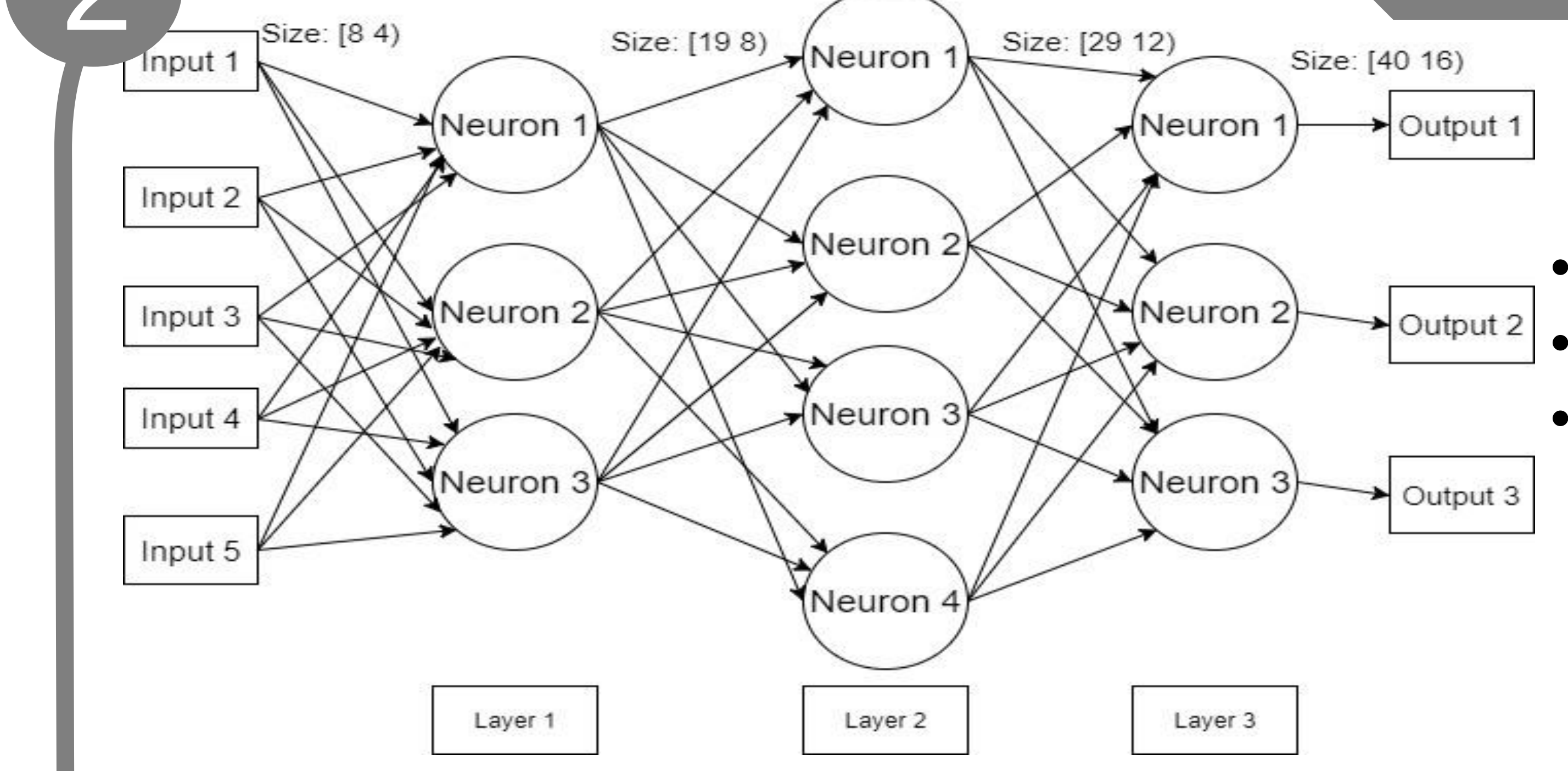
Layer 3 Output

```
0.53936767578125,0.0,0.41436767578125
```

Along with the VHDL model, we also created a neural network model in MATLAB. In order to verify our VHDL results pictured above, we ran this hardware simulation against the software simulation in MATLAB with the same network and determined the results to be fully precise.

Neural Network Design

2

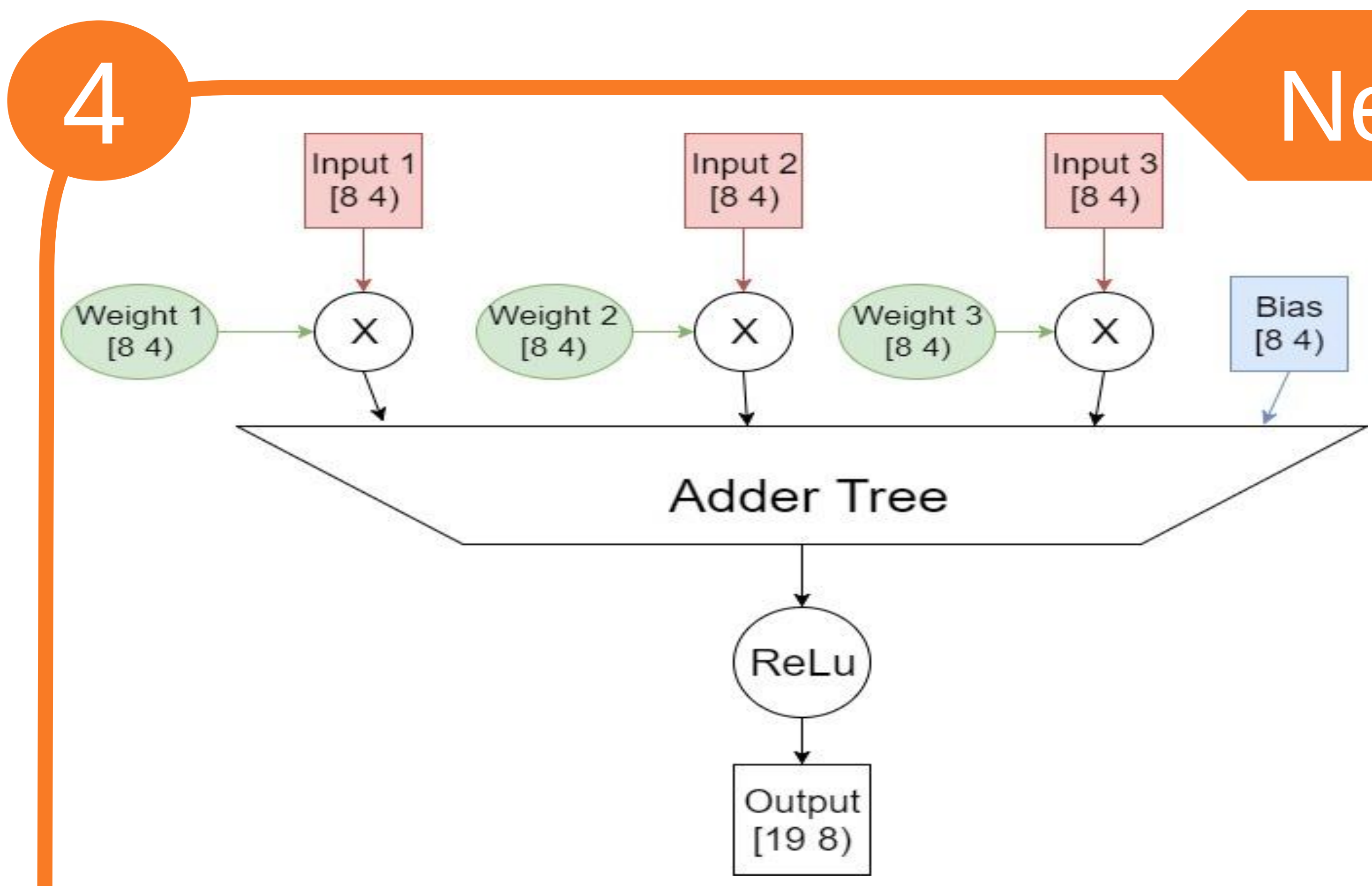


- 5 Inputs
- 3 Layers
- 3-4-3 Neuron Layout

Size: [A B) refers to the size of the real number being "A" total bits long with "B" amount of fractional bits. Example - [8 4) is a number that is 8 bits long with 4 fractional bits and therefore 4 integer bits. Since there is multiplication and addition in each layer, the size grows as each layer progresses.

Neuron

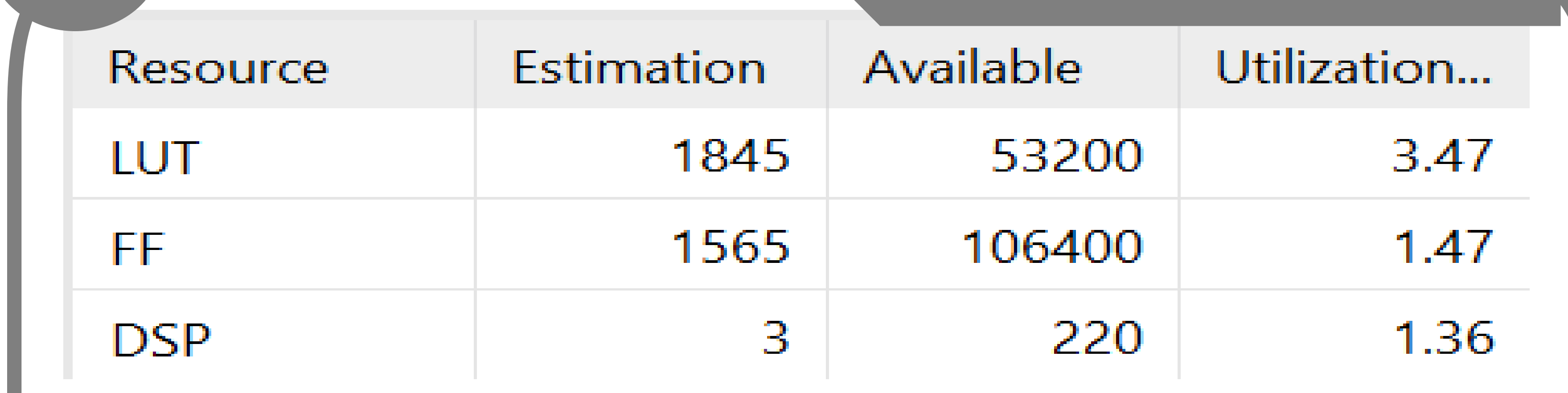
4



This is a neuron in layer 1. It shows the Inputs being multiplied by the weights and then being added to the bias. This result is then put through an activation function (ReLu: if y < 0 then set y equal to 0, else set y equal to y). This result is then stored into the output. The number of clock cycles this operation takes depends on the number of registers the adder tree uses. In this case the number is 4 + ceil_log2(# of inputs into adder tree): 4 + 2 = 6 cycles.

Results/Conclusion

6



Resource	Estimation	Available	Utilization...
LUT	1845	53200	3.47
FF	1565	106400	1.47
DSP	3	220	1.36

Using the ZYNQ-7010 Zedboard FPGA, our neural network implementation used a considerable amount of LUT and FF resources due to the arithmetic operations used in the adder tree portion of the algorithm. The DSP resources stem from the multiplicative operations used. While the utilization percentages (3.47% etc) are high, this pipelined, hardware network shows great performance in terms of speed due to the completion of one neural network output per clock cycle.

This program was supported by the National Science Foundation Grant No EEC-1659650. Additional support was provided by the School of Engineering and Computer Science at Oakland University.