**ECE 298A Deliverable**

**Vivi Huang, Evan Zhang**

**Design Specifications**

Our design specifications are shown below.

Design Objective:

Implement a 16-neuron Binary Neural Network (BNN), on a Tiny TapeOut ASIC to classify a 8-bit binary input (line inputs) to perform classification into 4 different classes: turn left, turn right, stop, and forward.

Input/Output:

We use all 8-bit parallel input and 8-bit outputs. The inputs are assumed to be “sensor-gathered data”, indicating where the obstacles are. 4-bits output will provide 4 different classifications based on the input, and other 4-bits indicate intermediate results of the BNN.

4 of the 8 bidirectional I/O pins will be used to implement dynamic weight loading. This is loaded during runtime, and weights are shifted serially into each neuron. In two clock cycles, the 4 bidirectional I/O pins will finish writing into one neuron, which contains 8 total weights. With a total of 16 neurons, the weight-loading process would be completed in 32 clock cycles. We use an additional I/O pin to use as the *load\_enable* signal to enable loading. To load successfully, *load\_enable* should be held at high for 32 clock cycles, and weights will be loaded serially into neurons 0-15, in that order. A pinout table for I/O is shown below:

| Description | Pinout | Description |
| --- | --- | --- |
| Input | ui[7:0] | 8-bit parallel line data input |
|
| Output | uo[7:4] | 4-bit classification output, uo[7] identifies left turn, 6 identifies right turn, 5 identifies forward, 4 identifies stop. |
| uo[3:0] | 4-bit debug pins. This shows intermediate output from neurons 9-12 in layer 2. |
|
| Bi-directional I/Os | uio[7:4] | 4-bit weight data, loaded serially |
| uio[3] | *Load\_enable* signal to enable loading |
| uio[2:0] | These bidirectional pins are unused, and could be used for debug later on. |

Table 1. I/O table diagram

A block diagram is shown below:

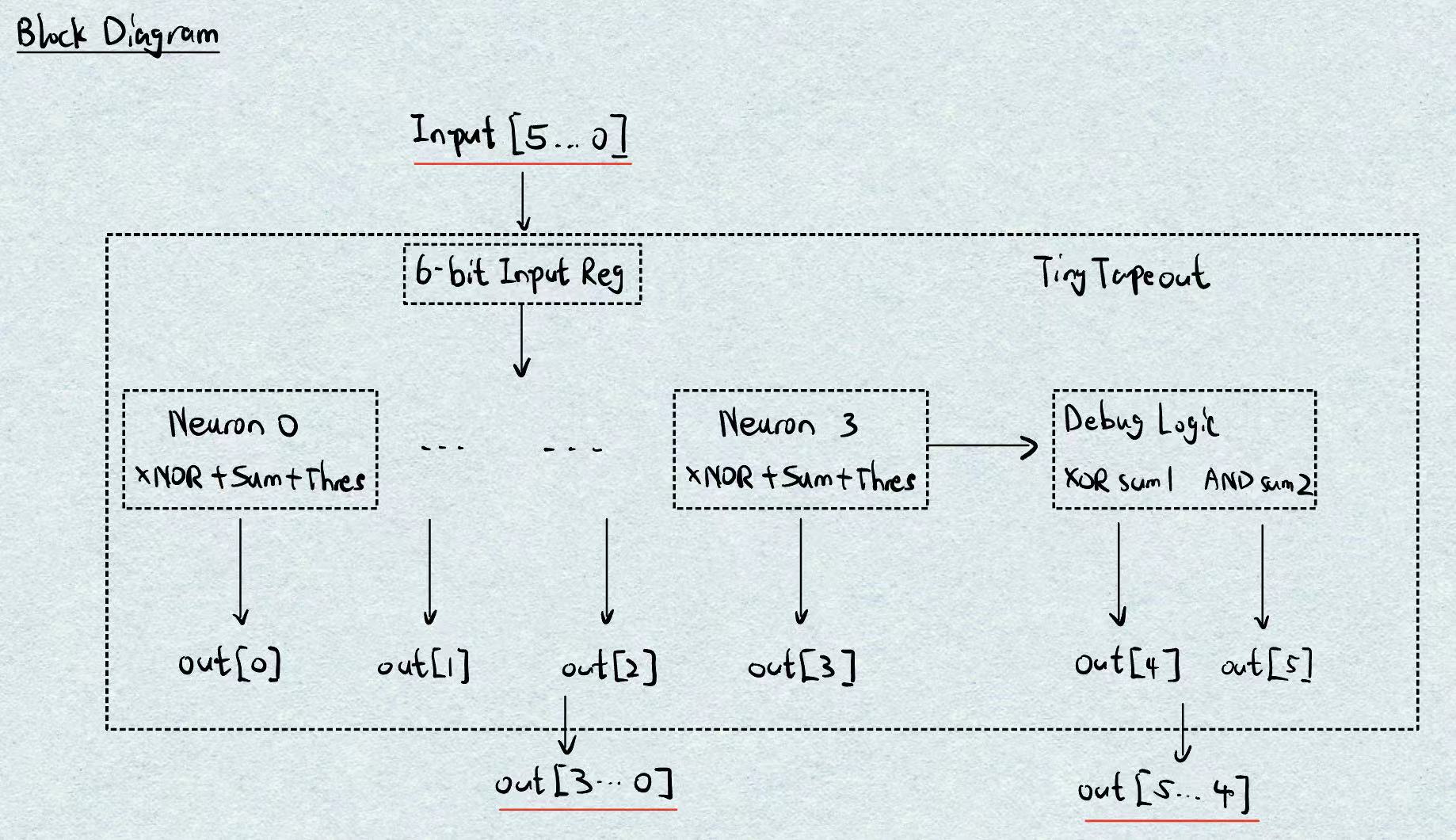
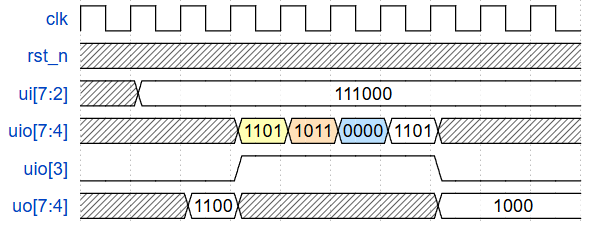


Figure 1. Block Diagram

A timing diagram is shown below:

Figure 2. Timing Diagram

Hardware Specs:

Each neuron will contain 8 weights and registers accordingly to perform **XOR + popcount + threshold** operations to produce a single output bit. By estimation and per neuron, we will need ~400 transistors to support all the FF as well as the Popcount logics, which take up most of the chip space. Note that the 2nd layer of neurons are simpler as they use a 4-bit input and thus uses fewer adders/FFs. This comes out to a total of less than 2500 transistors, which is less than half of the limit supported by TinyTapeout.

A block diagram for the above circuit components is shown below.

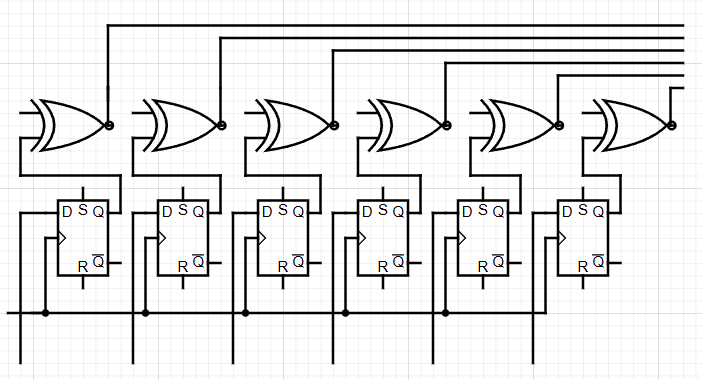


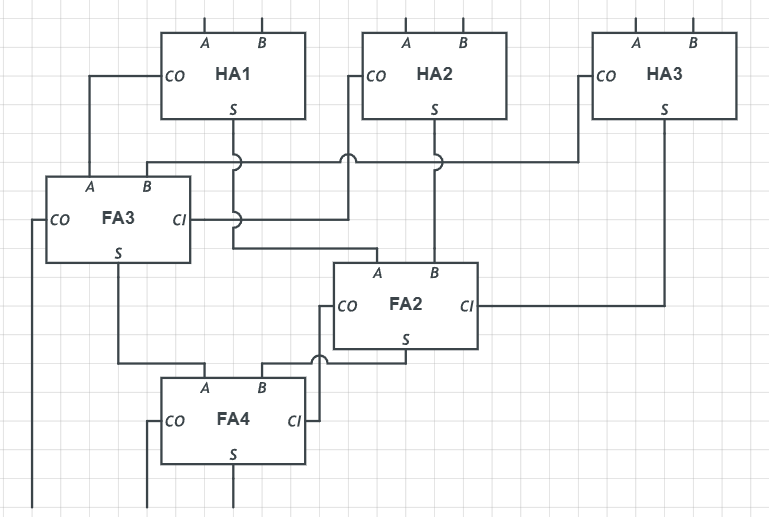
Figure 3. XNOR + weight loading circuit

Figure 4. Popcount logic circuit, 6-bits input on top, 3-bit sum output at bottom

Testing and Trainng:

Testing will be done based on the CocoTB functionalities built into TT’s repository. The 8-bit I/O, clock, reset, power and ground will be used like written above. The weight-loading can also be tested using 8 bi-directional I/Os. A python script [test.py](http://test.py) will be used to test BNN outputs as well as weight-loading cases.

Taking that into account, we will have 50 samples of training data sets and 20 sets of testing data in practice. These numbers are chosen for the following reasons. 1, BNN overfits the data quickly and a smaller dataset is beneficial to prevent overfitting. 2. 20 individual tests suffice to include all 4 feature detections, including some variations, to see if the classification process is satisfactory.

Lastly, weight-training will be done outside of the Tiny-Tapeout chip, and will be performed separately using the Larq framework in Python to train the weights for BNN. A library of training data is provided for the BNN, and the Tensorflow + Larq libraries repeats the training until the resultant weights are satisfactory.