**ECE 298A Deliverable**

**Vivi Huang, Evan Zhang**

**Design Specifications**

Our design specifications are shown below.

Design Objective:

Implement a 8-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 U-turn.

Input/Output:

Use 8-bit parallel input and 4 binary outputs + 4 debug outputs, out of the 8 total I/Os. 8-bit parallel inputs are assumed to be “sensor-gathered data”. 4-bit output provides 4 different classifications for the input. 4-bit debug bits are used to observe intermediate results from the first layer of our 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 based on the cycle number. In two clock cycles, the 4 bidirectional I/O pins will finish writing into one neuron, and the complete process can be completed within 16 cycles. We use an additional I/O pin to use as the *load\_enable* signal to enable loading. An I/O table 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 turn around, 4 identifies stop. |
| uo[3:0] | 4-bit debug pins, shows the dataout\_1 bits in the intermediate results |
|
| Bi-directional I/Os | uio[7:4] | 4-bit weight data, loaded serially |
| uio[3] | *Load\_enable* signal to enable loading |
| uio[2:0] | This bidirectional pin is unused, and could be assigned to debugging purposes. |

Table 1. I/O table diagram

A block diagram is shown below:

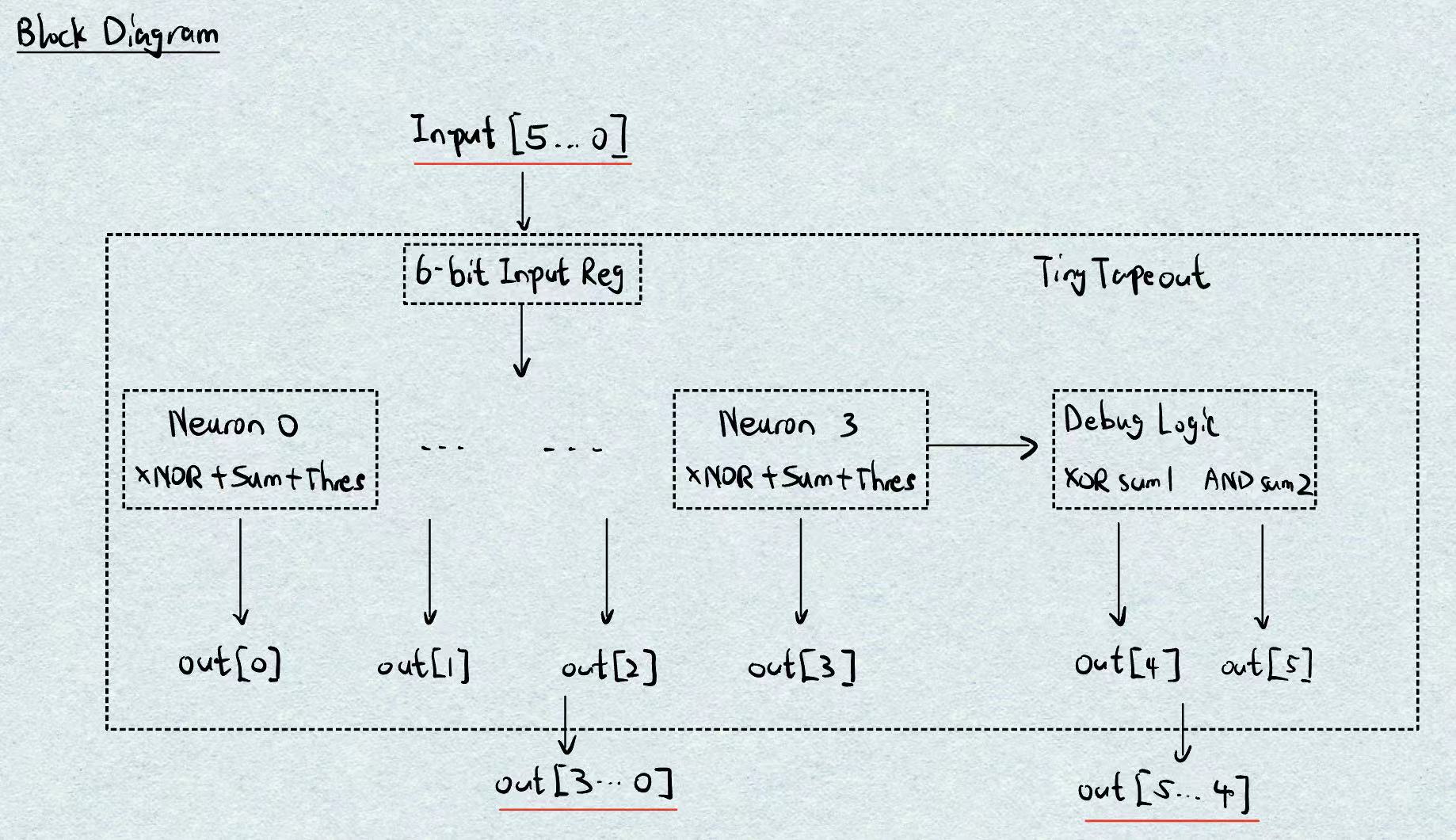
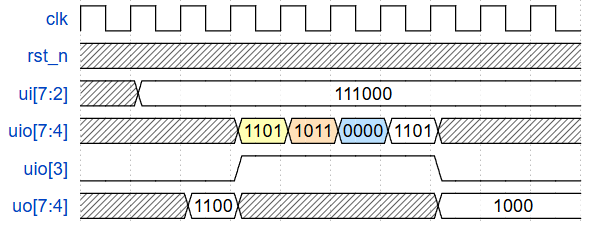


Figure 1. Block Diagram

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.

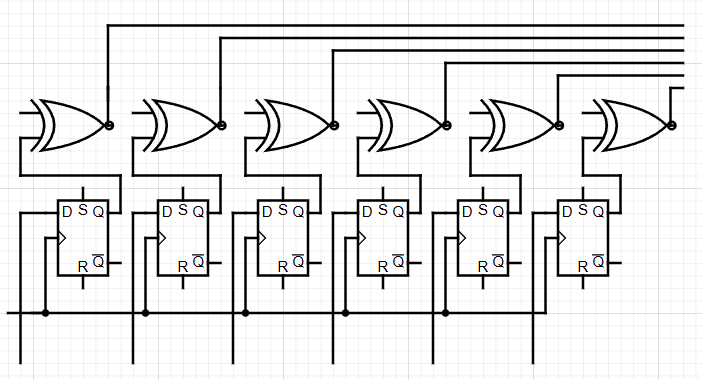


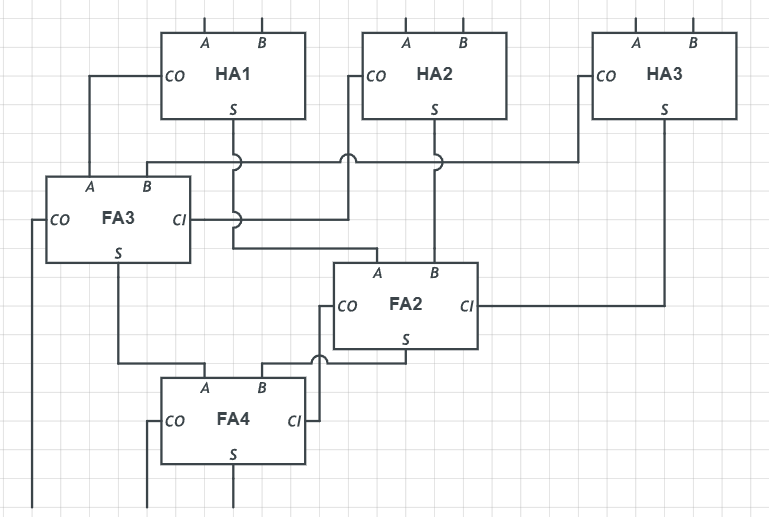
Figure 3. XNOR + weight loading circuit

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

Testing:

Testing will be done just using CocoTB. We’ll be able to use the 8-bit I/O, clock, reset, power and ground like written above. The weight-loading can also be tested using 8 bi-directional I/Os.

Note that 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 will be given to the BNN; this is what underlies the fundamental weights for our neural network.

Taking that into account, we will have 20 samples of training data sets and 10 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. 15 individual tests suffice to include all 4 feature detections, including some variations, to see if the classification process is satisfactory.