Assignment 5

Due Oct. 16th 2018

1 Readings

1-
NanoNets API
URL: https://medium.com/nanonets/how-to-easily-detect-objects-with-deep-learning-on-raspberrypi-225f29635c74
Github: https://github.com/NanoNets
Expected to discuss Next week (Oct. 16th) 2-
YOLO
paper title:
You Only Look Once: Unified, Real-Time Object Detection
expected to discuss in two weeks (Oct. 23rd)
Link: https://arxiv.org/abs/1506.02640

2 Presentations

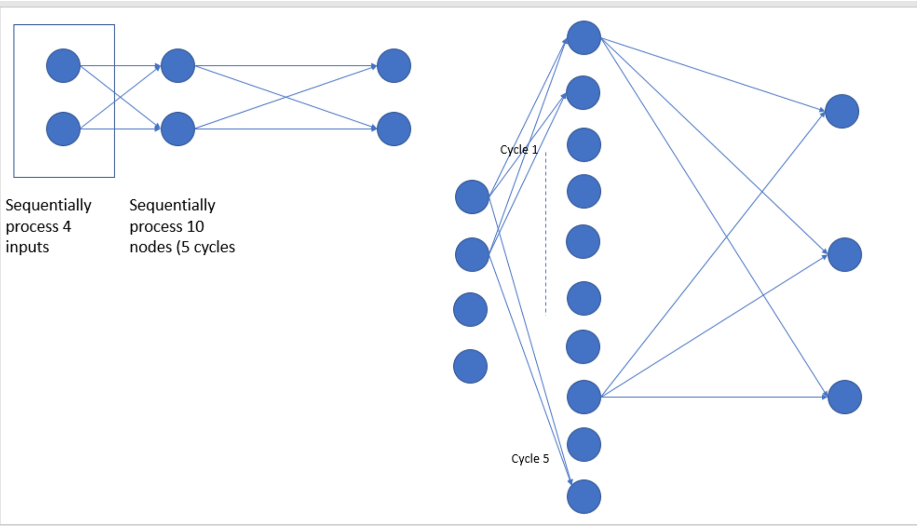
PIPECNN and openCL: Bobbie (Oct.23rd) Machine Learning Hardware Challenges: (Oct. 30th)
URL: https://arxiv.org/abs/1611.02450
Github: https://github.com/doonny/PipeCNN

Drivers: (Nov. 6th)

3 Problems

1-
use MNIST dataset to recognize digits. Try 4 different values for hidden nodes. Show the accuracy with each when you run them for multiple times.
MATLAB Reference: https://www.mathworks.com/help/deeplearning/examples/create-simple-deep-learning-network-for-classification.html
MNIST page (Full of resources and different Algorithm papers) : http://yann.lecun.com/exdb/mnist/
Github Example: https://github.com/floydhub/mnist
2-
You have a Neural Network with 4 Neurons for Input Layer, 10 Neurons for Hidden Layer, and 3 Neurons for Output Layer. The network is fully connected and each weight/bias/input is 8-bit value. Show how much memory we need.

a. If each layers has only 2 physical Nuerons implemented. Design a finite state machine showing how to utilize the 2 Neurons per layer. (Note that each Neuron will take the same processing time. Take timing into consideration and allow each layer to finish and handshake with the next layer) For example, we have the option of using two input nodes and go over 5 cycle delivering the input with their weights to next layer and then load the remaining two inputs. As the figure shows


Sequentially process 4 inputs
Sequentially process 10 nodes (5 cycles
Cycle 1
Cycle 5

Decide whether there is a waste in resources in such case.

b. If you have 4 Multiply Accumulate blocks and only one activation function, how would this impact your FSM and the whole design. (In such case assume you have 10 MACs with 3 activation functions in hidden layer, 3 MACs and one Activation function in output layer)

3-
(Due Oct. 23rd) Develop Modular 1-D Matrix Multiplier [(1xN)*(Nx1)] using Verilog and build its testbench. (You can use simple multiplication and loops to do it)