MILESTONE REPORT

Milestone Progress:

1. Milestone 0 - A report on comparison of SNN models with reasoning for the chosen model ✔
   Due date: 04/23/19
   Progress: Done
   Deliverable: Milestone 0 Report

2. Milestone 1 - Demonstrating working simulation of a simple block on vivado & implementation on FPGA board! ✔
   Due date: 04/30/19
   Progress: Done
   Deliverable: PYNQ Board Setup Report

3. Milestone 2 - Building software implementation of SNN for binary image classification ✔ (Revision 1)
   Due date: 05/07/19
   Progress: Done
   Deliverable: Code, trained weights and output spikes - Software Implementation Report

4. ~~Milestone 3 - A report on profiling results and block division~~ (Revision 2)

5. Milestone 3 - Building software implementation of SNN for multiclass image classification ✔ (Revision 2)
   Due date: 05/14/19
   Progress: Done
   Deliverable: Software Implementation Report

6. Milestone 4- A C/C++ implementation of the SNN network architecture suitable for HLS (Revision 3)
   Due Date: 05/24/19
   Progress: On going
   Deliverable: Final C/C++ Code and Classified Image Results

7. Milestone 5 - HLS and testing to attain working baseline implementation on FPGA board for binary classification
   Due Date: 05/26/19 (Revision 4)
   Progress: Yet to be completed
   Deliverable: Demonstration of successful binary classification on PYNQ

8. Milestone 6 - HLS and testing to attain working implementation on FPGA board for multiclass classification (at least 6 different digits)
   Due Date: 05/30/19 (Revision 4)
   Progress: Yet to be completed
   Deliverable: Demonstration of successful multiclass classification on PYNQ

9. Milestone 7 - Performance improvement (10x speedup) of hardware implementation
   Due date: 06/09/19
   Progress: Yet to be completed
   Deliverable: Report and Graphs showing performance improvement

10. Milestone 8 - Final report and video
    Due date: 06/14/19
    Progress: Yet to be completed
    Deliverable: A report on all implemented network architectures

All of the deliverables have been updated on our github repo:
https://github.com/snagiri/CSE237D-PYNQ-SNN-Accelerator

Revision of Milestones:

1. Revision 1:
   Milestone 2 - Earlier it was just software implementation of SNN. But as we started with it, we faced difficulties understanding how the input is converted to spike train and how spikes are generated through the network and how the SNN really work. Also, we tried using SRM neuron model for the SNN. SRM is a rarely used model. We tried modeling it ourselves and there were no proper reference materials or previous implementations available for the same. The SRM model we implemented was not functioning properly. We then shifted to using LIF neuron model which is relatively easy to understand and model. The milestone got delayed because of this reason. We first did software implementation of SNN for binary classification and we got good results.

2. Revision 2:
   Milestone 3 - We didn't find any requirement for doing profiling. The plan was to train the network locally, obtain the trained weights and implement just inference on FPGA. So, there was no need for profiling the code.

   For implementing multiclass classification we tried 3 different network architectures. Firstly, we modified the network used for binary classification such that it's suitable for multiclass classification and also modified few parameters used for training. Running time for training was really long. We tried modeling parameters to give better results but the neurons were not able to detect all the 10 classes correctly. But the same implementation works well for classifying 6 classes. This architecture has no hidden layers. Secondly, we tried implementing SNN for multiclass classification using a library exclusively designed for neural networks named Nengo DL. This architecture involved few hidden layers which performed convolution and pooling. An accuracy of around 99% was achieved for this network. As implementing convolution and pooling layers in hardware might get too complicated we tried modifying the same network so as to make it fully connected without convolution and pooling operations. This was our third implementation and the accuracy was around 97%. Nengo Dl is a huge library and it took us good amount of time to understand how things work as it was too high level.

3. Revision 3:
   Milestone 4 - This milestone was not present initially but was added later. Building the C/C++ implementation of the corresponding python one seemed to be a trivial job in the beginning but we realised that it's not. We are trying two implementations. One of them is implementing a deep network and it involves converting the code which uses Nengo Dl library. For this we had to figure out all the class dependencies as the library is highly interlinked. The other one is a shallow network and it doesn't involve any libraries.

4. Revision 4:
   Milestone 5 - Earlier HLS and baseline implementation were two different milestones but we realised that conversion to C/C++ consumed more time and so, these two milestones can be combined into a single milestone. It is further divided into two milestones one for implementation of binary classification and the other for multiclass classification.

Important things to do!

The most important task right now is to get our baseline implementation working on FPGA. Currently we are trying out two implementations parallely. Initially we designed our schedule allocating more time for performance improvement but realised that getting the baseline implementation work on FPGA is a tedious and the most important task. We updated our milestones reflecting the same.

Division of milestones with respect to grades:

Grade **B+** :
Completion of **Milestones 2,4,5**

Grade **A-** :
Completion of **Milestones 2,3,4,5,6**

Grade **A** :
Completion of **Milestones 2,3,4,5,6,7**

Note: We are trying different implementation approaches for each of the milestones which is why all three members work on each of the milestones.