A2 - Brain-Inspired Neuromorphic Computing project

Software tools required:

1. Verilog simulation: Questa*-Intel® FPGAs Standard Edition Software Version 24.1 (5.70 GB on Linux, 3.40GB on Windows)
2. Verilog RTL & synthesis:
    a. Visual Studio Code with plugin EDAcation (preferred option: <1GB) *(supports windows and Linux)*
    b. Other tools for this are: Vivado and Quartus (quite large: 10GB+) *(supports windows only)*

Project files:

The files can be cloned or downloaded from: https://github.com/whtech-nl/tinyODIN

There a multiple file types in use:

- .v Verilog (IEEE 1364): files describing neuromorphic architecture
- .sv SystemVerilog (IEEE 1800, superset of Verilog): file containing the testbench
- .wlf ModelSim/Questa waveform dataset: file containing the simulation result (waveform)
- .do "do file": containing instructions like adding fields to the wave

1 PROJECT DESCRIPTION:

The goal of the project is to get some experience with Verilog and utilize that to get experience with working with neuromorphic architecture. In this project we use the tinyODIN architecture for this purpose.

For the project there are two options to choose from, the first one is expanding the architecture to support larger inputs, because it is currently limited to 256 neurons, which allows only for a 16x16 image input for example.

The other option is to implement the control mechanism to classify an input image. The classification will be an MNIST digit classification using a 16x16 input image.

For those with already some Verilog experience it is possible to combine both projects, start with making the 16x16 image classification work, then expand the architecture to work with 28x28 images and do classification for those larger images as well.

1.1 EXPANDING/SHRINKING THE ARCHITECTURE

The goal of this project is to expand or shrink the architecture to allow for a different sized input. Keep in mind that changing the architecture not only requires changing the number of instantiated elements but also changing the data width of the memory addressing.

1.2 DOING IMAGE CLASSIFICATION

This consists of three parts: first initiate the architecture with the supplied weights, for this the provided weights file should be used.

The next step is providing image input to the system. For this input spikes, corresponding to image pixels will be provided.

The final stage is reading out the neuron spikes and doing classification based on spike frequency.

1.3 COMBINING BOTH PROJECTS

When combining the projects, use the provided weights and image for the combined project. First make the two other projects work separately before combining them.