HT2023: IL2230 HADL Lab 3A
Design, Implementation and Evaluation of MLPs in Hardware

Zhonghai Lu

November 22, 2023

1 Introduction

Multi-Layer Perceptron (MLP) is a regular layered construction of neural network whereas each layer comprises multiple or a number of neurons. There are three types of layers, namely, input layer, hidden layer and output layer. While there is only one input layer and one output layer, there may exist one or multiple or many hidden layers. It is "hidden" because it looks invisible to input and output. The inter-layer connections are all fully connected. Figure 1 shows an example of a 3-layer MLP. MLP is a foundational form of construction for modern neural networks such as the fully connection layers in convolutional neural networks (CNNs) and the recurrent, LSTM layer of recurrent neural networks (RNNs).

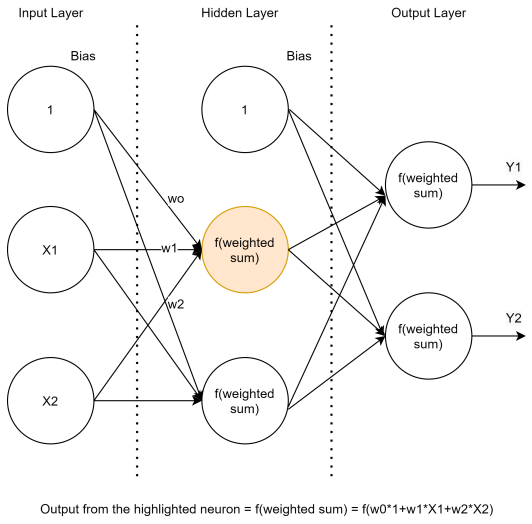

Figure 1: A 3-layer MLP.

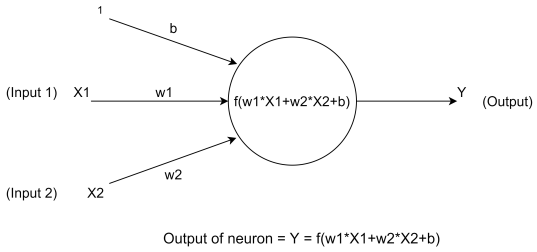

Figure 2: A neuron model.

In MLPs, all neurons are typically homogeneous (Strictly speaking, one should exclude the input layer as they only provide input data but not computation. Also, the output layer may use an identity function instead of sigmoid, tanh, or ReLU function etc. as the activation function). The basic neuron model is mathematically a weighted sum of inputs applied to a nonlinear transformation, as shown in Figure 2. Here the nonlinear function f can be in different forms such as the step function, sigmoid, hyperbolic tangent, ReLU etc. Neuron is the building block of neural networks. It is the "brick" of the deep learning "house".

Therefore it is essential to understand its impact on the hardware design and organizations of MLPs. In Lab 2, you have designed and implemented a hardware neuron in three forms: a serial neuron, a semi-parallel neuron, and a parallel neuron. After understanding the basic implementations of one neuron, we design and implement MLPs in this lab.

2 Objectives

The purpose of the lab is to understand the hardware design complexity of MLPs, especially when the scale of MLP is large, and comprehend the performance/cost trade-offs for different organizations of MLP implementation alternatives. In particular, we investigate how parallelism can be exploited when designing MLPs in alternative ways. We have a few specific objectives in this lab:

- One-neuron MLP design: Write an RTL model for a generic MLP by reusing a single neuron.
- N-neuron MLP design or Two-Neuron MLP design: Write an RTL model for a generic MLP by reusing N neurons (N-neuron MLP design) or two (equivalent) neurons (two-neuron MLP design)
- Synthesize the designs to evaluate the performance-cost trade-offs. The designs can reuse the software, code and tools you used in Lab 2.

The lab shall be carried out as a team work with 3 to 4 students in one group.

3 Tasks

The lab intends to design, implement, and evaluate a generic MLP in hardware. In particular, we are interested in how the hardware complexity scales when the number M of layers increases and the number N of neurons per layer grows. To this end, generic hardware models for MLP are to be designed, implemented at the Register Transfer Level (RTL) in VHDL or Verilog, and synthesized into logic circuits in order to obtain their area, frequency and power statistics.

Looking into the functions of an MLP, we can see that the core computation rests on the rudimentary neuron model, which is considered as uniform across the entire neural network (excluding the input layer) in this lab. Therefore, one can explore the digital hardware organization principles with respect to serialization and parallelization. In fact, the lab shall be carried out with two generic hardware designs with different levels of parallelization.

- Task 1. One-neuron MLP design.

  Generic M $\times$ N MLP modeling (M layers, N neurons per layer) with one single neuron unit (M, N are generic parameters). This means that the generic M $\times$ N MLP shall be modeled with a single neuron unit, and the control path takes care of the repetitive computation using the single neuron to achieve the desired functionality of the neural network. This is one extreme case of fully serial implementation at the neuron level.

  As a reference, you can find Figure 3, which draws a schematic for a generic one-neuron MLP. Think about if it is preferable to use a serial neuron or a parallel neuron for this design. Why?

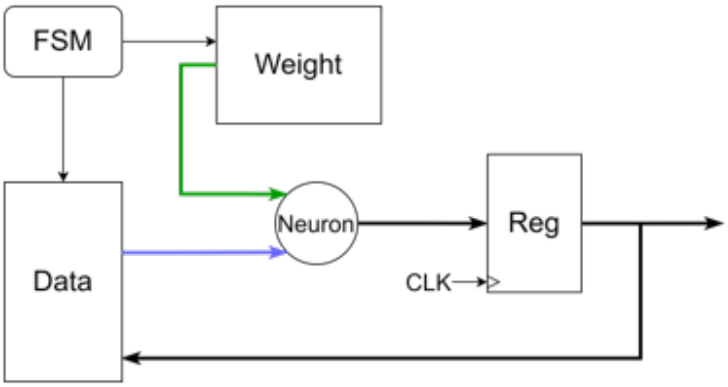

Figure 3: Schematic of a generic one-neuron MLP.

- Task 2. N-neuron MLP design.

  Generic M × N MLP modeling (M layers, N neurons per layer) with N neuron units (M, N are generic). This can be viewed as another extreme that uses a parallel architecture (as many per-layer neurons as needed). Still you need to consider how to re-use the one-layer neurons to implement the M layers of neuron computations.

  As a reference, you can find Figure 4, which draws a schematic for a generic N-neuron MLP reusing N neurons of one layer. Think about if it is preferable to use a serial neuron or a parallel neuron for this design. Why?

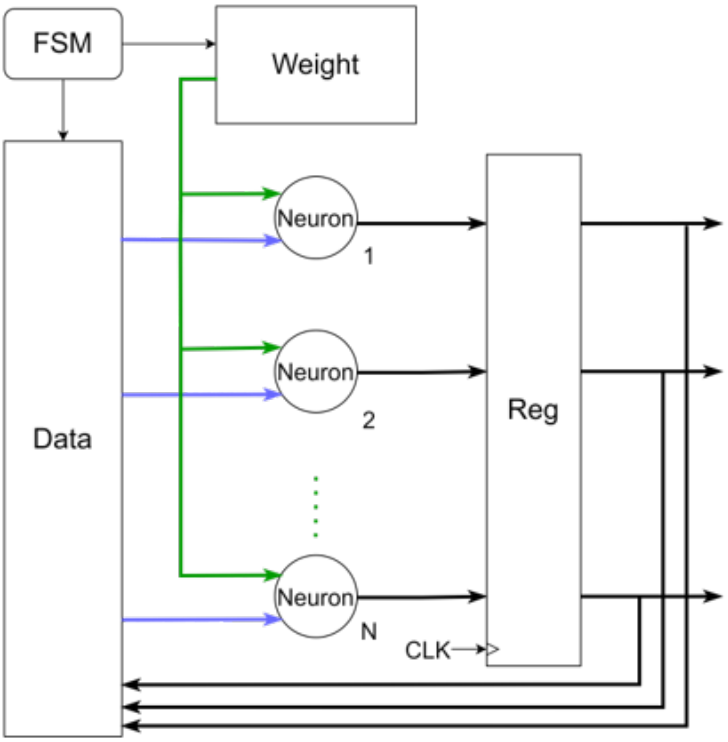

Figure 4: Schematic of a generic N-neuron MLP.

- An alternative to Task 2: The two-neuron MLP design exploiting inter-layer optimization.

You can choose to implement the two-neuron MLP presented in the paper below.

Shenggang Chen and Zhonghai Lu. "Hardware Acceleration of Multi-layer Perceptron Based on Inter-layer Optimization". International Conference on Computer Design (ICCD), November 2019.

If you choose this option as Task 2, you only implement a 3-layer MLP with N neurons per layer, i.e. M is fixed to 3, M = 3.

Certainly, one can play the area-speed trade-off game between the two extremes by using more than 1 neuron but less than N neurons to achieve the same functionality of the neural network. But for the lab, your interest may lie in the two extremes that define the boundary performance and cost metrics.

Additional constraints or considerations for both designs (models):

- To focus on the layer issue, the project simplifies the neuron design itself. You bring your parallel neuron design from your second lab, which uses a fully parallel structure to construct the basic neuron model, realizing a straightforward implementation of weighted sum of the inputs.

- Besides the digital organization, the project shall also investigate the impact of data precision (weight, input, output) on hardware implementation complexity. For hardware efficiency, the input data, weight and output use 32-bit and 8-bit fixed point numbers rather than floating point numbers. The intermediate partial sum may use as many bits as needed.

  You can choose to implement either 32-bit or 8-bit fixed point data type. If you have time, you can choose 32-bit and 8-bit fixed point data types.

- The nonlinear activation functions which can be considered are the sigmoid function or the ReLU function. For the sigmoid function, one can use an approximation method to realize, for example, piece-wise linear approximation or look-up table.

  You can choose to implement one of the two non-linear functions for your neuron.

Your designs should be synthesizable. Based on synthesis results, you will make a comparative evaluation of area, frequency and power for the following:

- The one-neuron design for M × N MLP under 32-bit or 8-bit data precision.

- The N-neuron design for M × N MLP under 32-bit or 8-bit data precision.

- If you choose to do the N-neuron design as Task 2, conduct scalability evaluation for your one-neuron design and your N-neuron design by giving different numbers for M and N (e.g. when M varies from 3 to 16, N from 3 to 16). You may even try M, N numbers larger than 16.

  You are supposed to test at least the following six combinations of M and N, namely, (M, N) = (3, 3), (3, 8), (3, 16) and (8, 3), (8, 8), (16, 16).

- If you choose to implement the two-neuron MLP as Task 2, your comparative evaluation is to be done for the single-neuron design and the two-neuron design. You can compare your synthesised designs for 3-layer MLPs with different N values, N = 3, 8, 16.

4 Documentation

1. The lab shall result in a technical report (names of team members on the first page), which describes the design, implementation and evaluation details. You should include a schematic block diagram for each of your designs.

   The report should contain hardware model simulation results and synthesis results. It shall also discuss the following in the technical report.

   - Discussion the scalability of your designs for the given choices of non-linear function and data precision. Report your synthesis results of area, power, and frequency.

     What conclusions can be drawn from the comparative and scalability evaluations? For example, the performance of design for small/large values of N and M, one-neuron/N-neuron, etc.
   - Discussion on how your memory subsystem (for storing weights, activations) impacts your designs? For example, what if only half or 25% of the weights can be loaded in one cycle for each layer computation?
   - In your N-neuron design, you have considered to parallelize the computations within each layer. This is a natural layer-after-layer processing, meaning that the next layer computation starts only after all the previous layer computation is complete. Is it possible to realize overlapped layer processing? This means that computation for the next layer m is performed simultaneously with computation for the previous layer $m-1$. If this is possible, discuss how? If not, why?

2. Upload your source RTL code, test bench for simulation and the technical report to the Canvas course page. The source RTL code with test bench should accompany a simple Readme.txt for how to run the code and obtain the simulation results in your report.

   After your lab results have been approved by the lab assistant, submit your deliverable package via the Canvas course website:

   https://kth.instructure.com/courses/42938