HT2022: IL2230 HADL Lab 2
Hardware Design, Implementation and Evaluation of Artificial Neuron

Zhonghai Lu

November 11, 2022

1 Introduction

An artificial neuron is perceived as a conceptual model of a biological neuron but in its very primitive form. Figure 1 shows the original neuron model called perceptron, which uses the step function to achieve non-linear functionality.

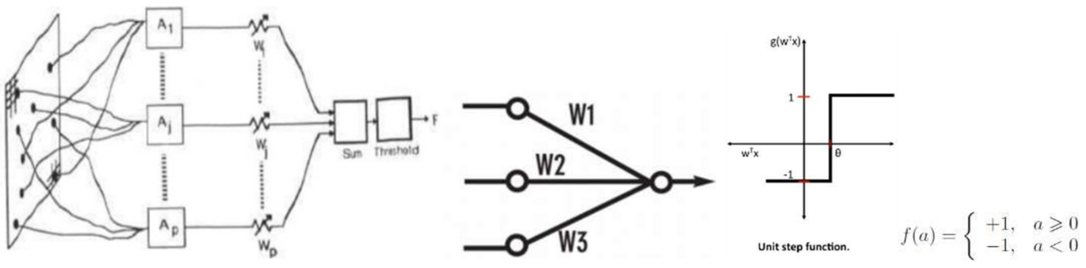

Figure 1: Original perceptron.

The artificial neuron model is mathematically a weighted sum of inputs followed by 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. Despite its simplicity, neuron is the building block of neural networks. It is the “brick” of the deep learning “house”. Therefore, to design an efficient hardware accelerator for neural networks, it is essential to understand various hardware design organizations and their tradeoffs of the basic unit, neuron.

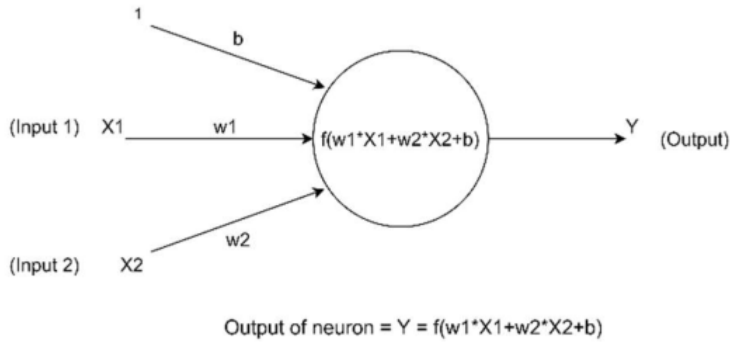

Figure 2: A two-input artificial neuron.

2 Purpose and Tools

The lab intends to design, implement, and evaluate a generic N-input neuron in hardware in terms of efficiency and cost. Therefore, a hardware model of the artificial neuron is to be built up in VHDL or Verilog and synthesized to obtain its area, frequency, and power profiles.

The lab is a team work. A typical team size is 3 to 4 students.
We use both hardware simulation and synthesis tools for this lab.
For students taking IL2230 and IL2225 Hardware Design in ASIC and FPGA:

1. Use Modelsim for simulation
2. Use Synopsys Design Vision for synthesis (software on KTH server, need license).
3. Refer to the Lab1 manual of IL2225.

For students taking only IL2230:

1. Use Xilinx Vivado for simulation and synthesis (standard version is free, you can use the latest one).
   `https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html`
2. Simulation tutorial:
   `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug937-vivado-design-suite-simulation-tutorial.pdf` (Chapter 1 and 2)
3. Synthesis tutorial:
   `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug901-vivado-synthesis.pdf` (Chapter 1)

3 Tasks

Looking into the operations of the neuron, we can find that the core computation is a repetitive application of Multiplier–Accumulator (MAC) operation, shown in Figure 3. One MAC operation is one multiplication plus one addition. Therefore, one can explore the digital hardware organization principle with respect to serialization and parallelization when exploring the trade-offs between performance and cost (hardware area, power etc.).

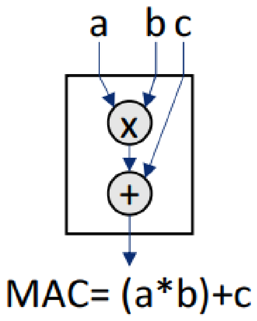

Figure 3: An MAC unit.

In fact, the lab shall be carried out with three alternative designs/implementations:

1. Generic N-input neuron modeling with a single MAC unit (N is generic). This means that the N-input neuron shall be modeled with only one MAC unit, and the control path takes care of the repetitive MAC operations for N times. The control path is typically designed as an FSM (Finite State Machine). This is one extreme case of *fully serial* architecture resulting in lowest area cost.
2. Generic N-input neuron modeling with N-MAC units (N is generic). This is the other extreme that uses a *fully parallel* architecture (as many MAC units as needed) leading to a fastest solution.
3. Generic N-input neuron modeling with K-MAC units (N and K is generic, and K<N). This is a semi-parallel solution that the control path takes care of the repetitive MAC operations over N/K times and sums the intermediate results in the end. In this lab, we set K=2. By varying K, we can play the trade-off between the two design extremes in terms of performance and cost.

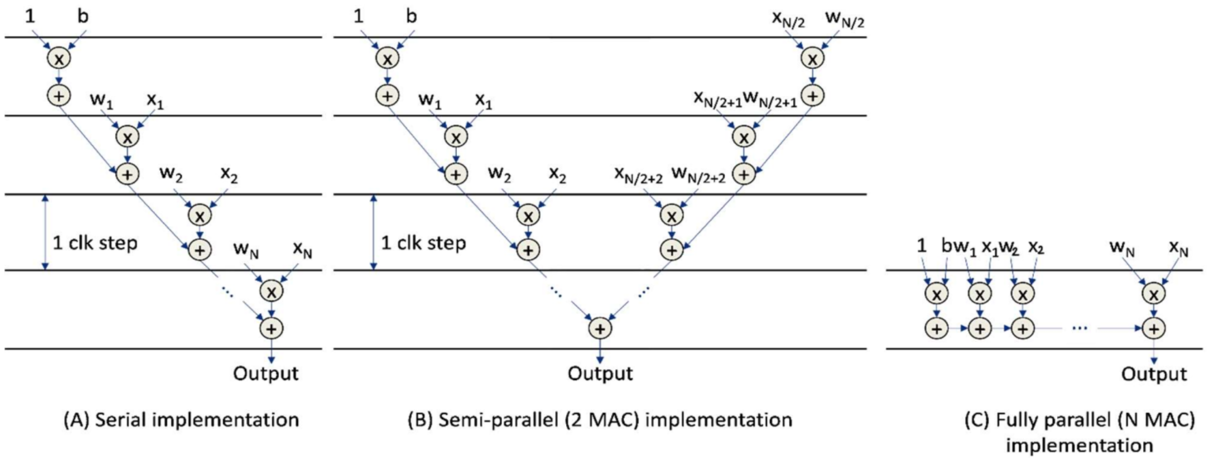
1
b
w1
x1
w2
x2
wN
xN
xN/2
wN/2
xN/2+1
wN/2+1
xN/2+2
wN/2+2
1 clk step
Output
(A) Serial implementation
(B) Semi-parallel (2 MAC) implementation
(C) Fully parallel (N MAC) implementation

Figure 4: Operation schedules of the three alternative N-input neuron implementations.

The operation schedule of each of these solutions is illustrated in Figure 4. Figure 4(A) is for the fully serial implementation with 1 MAC unit. Figure 4(B) assumes a semi-parallel implementation with 2 MAC units, and Figure 4(C) is a fully-parallel implementation with N MAC units.

There are additional constraints or considerations for the three hardware implementations:

1. Besides exploring the digital hardware organization principle, the lab shall also investigate the impact of data precision (weight, input, output) on hardware complexity.

   For hardware efficiency, the input data, weight, and output data use 32-bit, 16-bit and 8-bit fixed point numbers instead of floating point numbers. For 32-bit/16-bit/8-bit fixed point number, use 12/6/3 bits for the integer part and 20/10/5 bits for the fractional part, respectively. Note that the intermediate partial sum may use as many bits as needed.

2. The nonlinear function to be considered is the step function, the sigmoid function and the ReLU function. For the sigmoid function, one shall use an approximation method to realize, for example, piece-wise linear approximation or look-up table.

   In this lab, you need to implement the ReLu function for 32-bit/16-bit/8-bit fixed point number design and implement the sigmoid function for 8-bit fixed point number design.

You shall make a comparative evaluation of area, frequency and power for the following:

- The one-MAC design, K-MAC design, and N-MAC design for an N-input neuron under the same data precision (32-bit, 16-bit or 8-bit data precision) with the same nonlinear function.
- Consider N varying from 2 to 64 for the N-input neuron. Set N = 2, 4, 8, 16, 32, 64.

Answer the following questions when your lab results are checked by TAs and write them in your report:

1. Draw a state-transition diagram and a design schematic figure each for your one-MAC design and K-MAC design. How did you implement the control path (controller) in your MAC designs? How many states do they have?

2. Assume your design is partitioned into a data path and a control path in general, which path consumes more area? To clarify, the data path means the data computation path from input to output (e.g. The MAC and nonlinear function are on the data path), while the control path means the FSM controller which controls reading input, generating output, and performs computation at each clock cycle. What if the data precision is reduced or the number of inputs N increases? Draw a bar diagram to make the comparison.

3. How did you implement the sigmoid function? If you are using piece-wise linear approximation, how many segments did you use, why? If you are using a look-up table, how did you decide the number of entries in the table?

4 Documentation

1. Source code and test benches verifying the correct functionality of your designs.
2. Hardware model simulation results and synthesis results.
3. A technical report, which shall be submitted to the Canvas after your results are approved by a lab assistant.

The technical report should contain the design, implementation, and evaluation details. More importantly, proper conclusions shall be drawn from the comparative evaluations under different data precisions, and discussions for the design scalability (e.g. when N varies from 2 to 64) be given.

You need to answer the questions when your lab results are checked by TAs and write them in your report. In addition, you can write down the lessons learnt from the lab.