Rochester Institute of Technology
Department of Computer Engineering
CMPE 530/630 Digital IC Design

Final Project: Neural Network Datapath Design

Team Project (Two people per team)

Part 1 (RTL Design and Verification) Due: 11/21/2018
Part 2 (Synthesis, Layout, and Analysis) Due: 12/19/2018 @ 10:45 am

Table of Contents

1. Introduction

1.1 Objectives

- To learn about basic design and hardware implementation of artificial neural networks (ANNs)
- To learn how to integrate custom and standard cells in a top-down design flow

1.2 Neural Network Overview

Artificial neural networks (ANNs) are widely used in a variety of machine learning applications, which can be broadly classified as either (i) *classification problems* (*e.g.* identifying objects in a picture) or (ii) *regression problems* (*e.g.* predicting stock market prices). An ANN is essentially a network of nodes with weighted connections. The ANN's output $\mathbf{y}$ is computed by evaluating a hypothesis function $h_{\mathbf{W}}$ of its input $\mathbf{u}$.

$$\mathbf{y} = h_{\mathbf{W}}(\mathbf{u}) \tag{1}$$

where, $\mathbf{W}$ is a parameter matrix called the *weight matrix*. Each entry $w_{ij}^{l} \in [-w_{max}, w_{max}]$ of the weight matrix specifies the weight connecting the j^{th} node in layer $l-1$ to the i^{th} node in layer l. If two nodes are not physically connected, then their corresponding weight entry will always be 0. The type of ANN being implemented determines how the network's nodes are connected and, therefore, governs the form of $h_{\mathbf{W}}$. In this project students will design a *multilayer Perceptron* (MLP) network, shown in Figure 1.

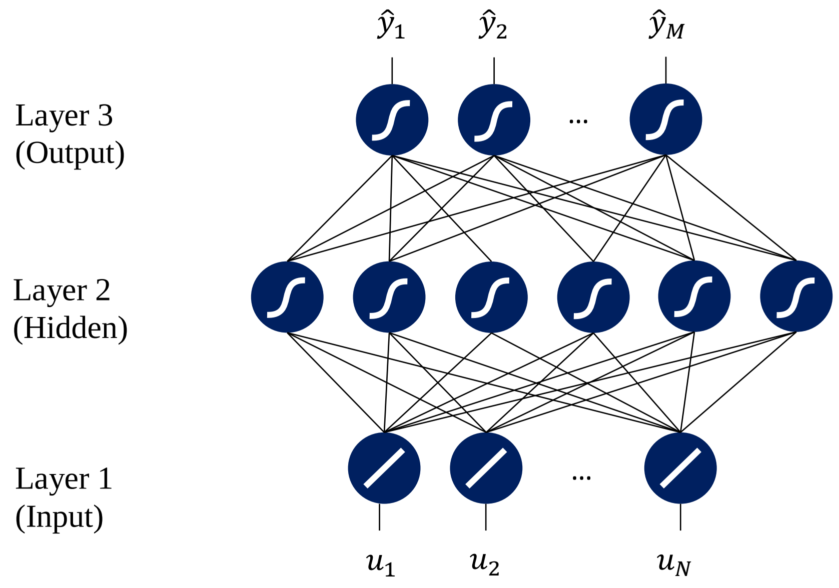


Figure 1. Multilayer Perceptron network.

The network has 3 layers of nodes: N nodes in the input layer (bottom row), H nodes in the middle layer (middle row, also called the *hidden layer*), and M nodes in the output layer (top row). In addition, each node, except for those in layer 1, has a $bias$ input that determines its constant offset. Each of the lines connecting nodes within the $h_{\mathbf{W}}$ portion of the network (*i.e.* excluding the lines connected to the inputs and outputs) is a weighted connection.

Each node computes a weighted summation of its inputs and applies an *activation function* to compute the node's output. The input and bias nodes use the identity function as their activation function:

$$\mathbf{x}^1 = \mathbf{u} \tag{2}$$

The rest of the nodes use a non-linear sigmoid function f as the activation function:

$$\mathbf{x}^l := f(\mathbf{W}^l \mathbf{x}^{l-1}) \tag{3}$$

where

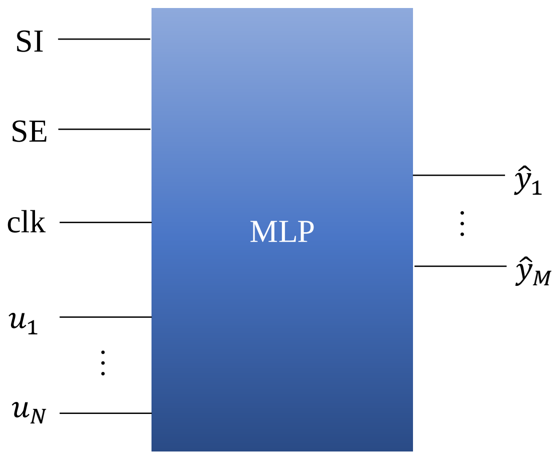
$$f(s) = \frac{1}{1 + \mathrm{e}^{-\mathrm{s}}} \tag{4}$$

The sigmoid function can be approximated using a piecewise linear function.

2. Procedure

2.1 Part 1

- Create a VHDL model of an MLP, with generics for N, H, and M:



```
entity mlp is begin
  generic(
    N : integer; -- # Inputs
    H : integer; -- # Hidden
    M : integer; -- # Outputs
    m   : integer; -- For Qm.n
    n   : integer  -- For Qm.n
 );
  port(
    SI : in std_logic;
    SE : in std_logic;
    clk ; in std_logic;
    u ; in std_logic_vector(N*(m+n+1) downto 0);
    yhat : out std_logic_vector(M*(m+n+1) downto 0
  );
end
```

Figure 2. MLP entity.

- In addition, you will need generics for the fixed point format used for all numerical signals, *i.e. m* and *n* (see Hints). SI is the scan chain input, which will be a serial input used to set all of the weights and bias values in the network for testing.
- There are several design approaches for each part of the MLP. For example, the activation function implementation could use piecewise linear approximations of the sigmoid function, or a lookup table to implement it more precisely. The network could be fully parallel, which will be fast but large, or serial, which will be slow but small. In a serial approach, you might consider a multiply accumulate (MAC) architecture at the input of each neuron.
- **Carefully map out your design approach using detailed block diagrams before you start writing code.**

- Use the provided MATLAB script to find the weight matrix **W**.
  - The script trains the network using an optimization algorithm called resilient backpropagation (RPROP), using the training vectors shown in Figure 3(b). It is not required that you understand the training algorithm implemented by the MATLAB script.
  - The only parameters that need to be specified (in top.m) are:
    - f – The activation function you are using
    - fp – The derivative of the activation function you are using
    - wmax – The maximum weight value you have chosen
    - wmin – The minimum weight value you have chosen
- Perform functional testing of the MLP using the parameters in Table 1, where the values in the weight matrix are found from the previous step. Figure 3(a) shows

how the MLP should be setup. A threshold component needs to be added to the output (in your VHDL code), so that the output is a 1-bit standard logic value, which will be '1' when the MLP input corresponds to an edge and '0' otherwise. **You should get 100% classification accuracy, meaning your output is '1' for every edge pattern and '0' for every non-edge pattern.**

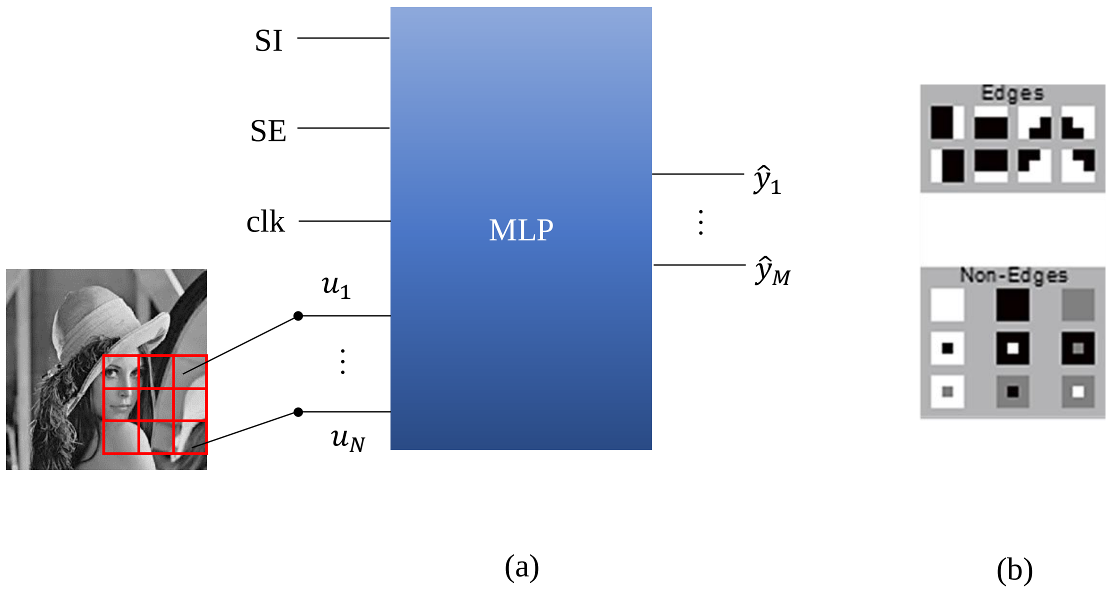


Figure 3. (a) Network setup and (b) training vectors.

Table 1. MLP functional verification parameters.

Parameter	**Test Value**
N	9
H	20
M	1
$Qm.n$	Q4.4
$\mathbf{W}^2$, $\mathbf{W}^3$, $\mathbf{b}^2$, b^3	See previous step.
$\mathbf{u}$	Test using all of the training vectors shown in Figure 3(a).

- In addition, detect the edges of the images provided on MyCourses
- Turn in your status report for Part 1

2.2 Part 2

- Synthesize the MLP using the parameters listed above and find the worst-case delay, dynamic power, and static power using the synthesized netlist.

- Generate the layout of your design using autolayout tools and perform post-layout simulations to find the maximum delay, dynamic power, and static power.

3. Graduate Students

Graduate students will be required to add one additional design feature to their neural network, which must be approved by the professor. Options include, but are not limited to:

- Built-in self-test based on a built-in logic block observer and random test pattern generation
- On-chip training
- Stochastic design

4. Hints

- Q format numbers should be used in this project. They are integers which can be interpreted at rational numbers. A Q format number is specified as Q$m.n$, where m is the number of integer bits and n is the number of fractional bits. An additional bit is used as the sign bit. Therefore, the total number of bits is $m+n+1$. Note that this is still two's complement, though, not sign magnitude. The resulting number is interpreted as $b = b_m b_{m-1} b_{m-2} \dots b_0 . b_{-1} b_{-2} \dots b_{-n}$, where b_m is the sign bit. For example, with m=1 and $n=4$:

    - 11.0101 = -0.6875
    - 10.1100 = -1.25
    - 01.0010 = 1.125
    - 00.0001 = 0.0625

  Addition and subtraction of fixed point numbers is identical to integer addition and subtraction:

    - $$\begin{array}{rr} 11.0101_2 & -0.6875_{10} \\ +\ \underline{01.0010_2} & +\underline{1.125_{10}} \\ 00.0111_2 & 0.4375_{10} \end{array}$$
    - $$\begin{array}{rr} 10.1100_2 & -1.25_{10} \\ +\ \underline{00.0001_2} & +\underline{0.0625_{10}} \\ 10.1101_2 & -1.1875_{10} \end{array}$$

Multiplication is also straightforward. The idea is to perform "normal" multiplication, and then shift to the right by n and truncate:

- $\begin{array}{r} 11.0101_2 \\ \times \underline{01.0010_2} \\ 111011.1010_2 \gg n = 11.10111010_2 \rightarrow 11.1011_2 \end{array}$ $\quad$ $\begin{array}{r} -0.6875_{10} \\ \times \underline{1.125_{10}} \\ -0.7734375 \end{array}$

5. Final Project Report

Your project status report should include the following:

1. Cover Page
   - Name and number of the lab
   - Your name and the names of your instructor and T.A.
   - Submission Date
2. Table of Contents
3. Abstract
   - Describe what was done, how you did it, and the significant results
4. Design Methodology
   - Discuss your design approach. In particular:
     i. Did you have a fully parallel or serial design, or some hybrid? Was your design driven by optimal area or optimal performance, or both?
     ii. How did you implement fixed-point arithmetic operations?
     iii. What approach did you take to implement the activation functions (*e.g.* piecewise linear approximation, lookup table, etc.)
     iv. What values did you use for w_{min} and w_{max} and why?
     v. Graduate students: What optional design feature did you include?
   - Include block diagrams of your design.
5. Results/Data Analysis
   - Present and analyze the functional, timing, and power results.
6. Conclusions
   - Restate important results, how they were obtained, and any uncertainties, possible improvements, or implications
7. Appendices
   - VHDL code for your design
   - Functional, timing, and power simulation waveforms.

Highlighted items to be included in the Part 1 status report.