Computer Aided Design (CAD) -Assignment Description
University of Tehran
Electrical and Computer Engineering Department
Fall 99

Final Project: Hardware implementation of a Multi-Layer Perceptron (MLP) Neural Network

In final project, you have to design and implement a simple MLP artificial neural network to classify the MNIST dataset using Verilog/VHDL language.

- ***INTRODUCTION***

In this project, the goal is to implement a simple MLP (Multi-Layer Perceptron) model of neural networks to apply image classification based on the MNIST dataset.

Multi-Layer Perceptron (MLP) *- The most common form of neural networks is the Feed-Forward Multi-Layer Perceptron (MLP). A feed-forward neural network is an ANN wherein connections between the neurons do not form a cycle. An n-layer MLP consists of one input layer, n-2 intermediate (hidden layers), and one output layer. Each layer consists of a set of basic processing elements or neurons. An individual neuron is connected to several neurons in the previous layer, from which it receives data, and also it is connected to several neurons in the next layer, to which it sends data. Except for the input nodes, each neuron uses a nonlinear activation function.*

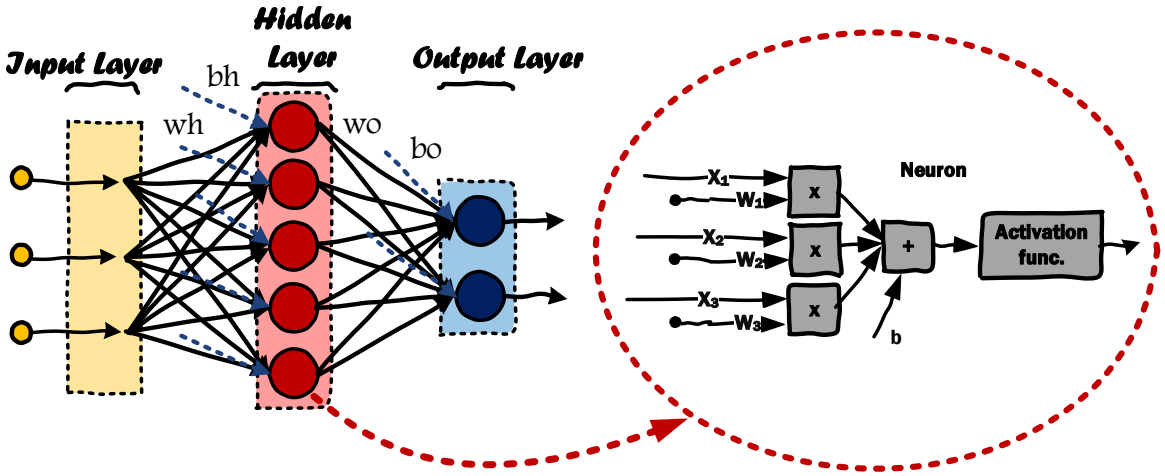

Figure 1. An MLP neural network with one hidden layer

Figure 1 illustrated the structure of such a network with one hidden layer. Consequently, all neurons in any given layer i receive the same set of inputs from layer i-1. Associated with each input, each neuron keeps a weight that specifies the impact of the input in the final output.

The output of each neuron in hidden layers is the weighted sum of the neuron inputs, and it is presented as input to the next layer after passing through a nonlinear function. The data is finally propagated to the output layer after passing through one or more hidden layers. So, in hidden and output layers, each neuron has the computation formula as Relation.1.

$$y_j = f(\sum_{i=1}^{n} W_{ij} \times x_i + b_j) \quad \textbf{Relation. 1}$$

x_i *and* y_j *are the input and output values of the neuron.* W_{ij} *and* b_j *shows the value of neuron weights and biases, respectively. Function* $f(.)$ *represents the nonlinear activation function.*

Image Classification Application- *Today, artificial neural networks are used in various critical applications such as classification, pattern recognition, clustering, optimization, etc.* ***Image classification*** *is the process of analyzing the image to identify the 'class' of it (or the probability of the image being part of a 'class'). As shown in figure 2, a class is essentially a label, for instance, 'cat', 'dog', and so on.*

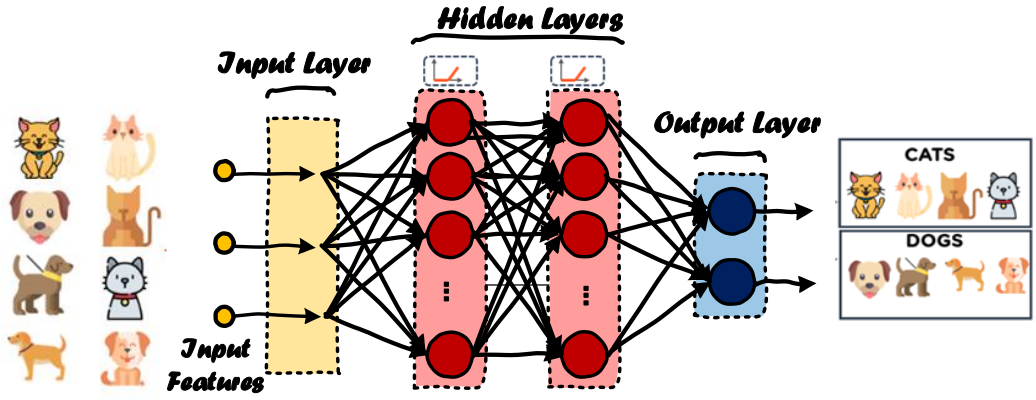

Figure 2. An Example of an image classification application

Input Dataset (MNIST) - *In this project, you should implement an MLP network to* ***classify*** *a set of images related to the "****MNIST****" dataset that is widely used for training and testing in the field of machine learning. MNIST dataset includes some black and white images of handwritten digits that are between 0 and 9. Some examples of these images are shown in Figure 3.*

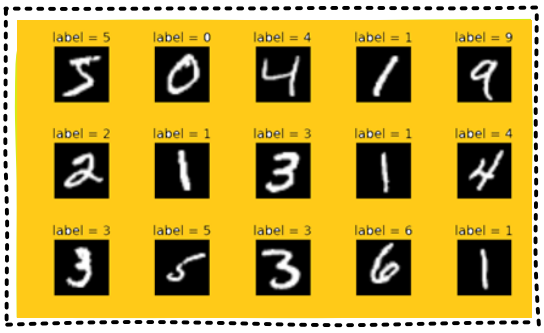

Figure 3. Some example images from the MNIST dataset

- ***PROJECT DESCRIPTION***

In this project, you have to design and implement a simple MLP neural network to classify the MNIST dataset by using Verilog language, based on the following issues:

- *The original MNIST dataset's images have dimensions of 28 x 28 pixels, totally included 784 input features. In order to simplify the classifier neural network, the size of images is reduced to 62 features using some feature reduction algorithms. So, in this case, your neural network will have **62 input** features.*
- *MNIST dataset has **10 output classes** corresponding to the digits from 0 to 9, in which by recognizing the digit i, the i' th output becomes one. To identify the label of an input image, the **maximum** value between the outputs of 10 neurons must be selected. If neuron i has the highest output value, the input image tag will be i.*
- *In both layers of your MLP model for MNIST classification, the **ReLU function** is considered as an activation function.* $ReLU(x) = max(0, x)$

(As shown in Figure 4, the MLP in this project has 62 inputs in the input layer, 10 output neurons in the output layer, and also one hidden layer with 30 neurons is considered for this issue. The activation function of neurons is ReLU function)

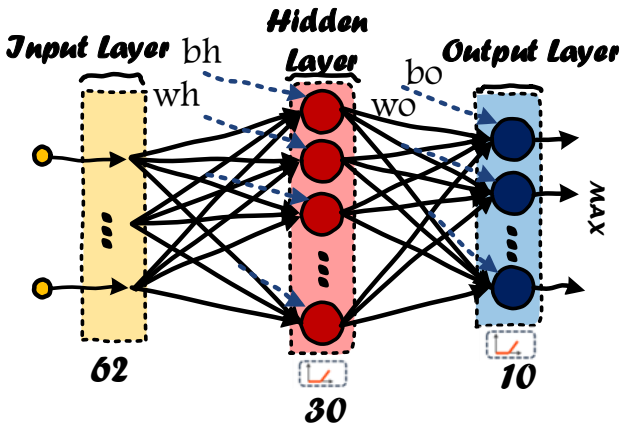

Figure 4. The required MLP network in this project

- *In order to implement this network, the hidden and output layers' weights and biases are required (*$w_{h(hidden)}, w_{o(output)}, b_{h(hidden)}, b_{o(output)}$*). Based on the described network,* w_h *and* w_o *are two matrices with the dimensions of 30 x 62 and 10 x 30, respectively, including 2160 weights totally. Also,* b_h *and* b_o *are matrices (vectors) with dimensions of 30 x 1 and 10 x 1, respectively, including 40 bias values.*

The pseudo code of this neural network is described below:

Pseudo Code of Neural network:

```
//Initial Parameters
Init Wh[30][62];   //Weights of hidden layer, 8-bit sign-mag
Init Wo[10][30];   //Weights of output layer, 8-bit sign-mag
Init Bh[30];       //Biases of hidden layer, 8-bit sign-mag
Init Bo[10];       //Biases of output layer,8-bit sign-mag


// Get Input feature
Get In[62];      //Input test data, 8-bit sign-mag

// Hidden layer:
H_out [30];
For i=1:30
        H_out[i]=0;
        For j=1:62
                H_out[i] += Wh[i][j]* x In[j];  //Sum of weight-input products
        H_out[i] = (H_out[i] + Bh[i]*127);      //Bias addition
                                                //multiplication is for bias alignment , 127=(01111111)8-bit sign-mag
        H_out[i] = H_out[i) >> 9-bit;           //Shift to right due to scaling
        H_out[i] = ReLU(H_out[i]);              // Activation Function , ReLU=max(H_out[i], 0)
       H_out[i] = Sat(H_out[i]);                //Saturation to 8-bit


// Output Layer;
O_out [10];
For i=1:10
        O_out[i]=0;
        For j=1:30
                O_out[i] += Wo[i][j] x* H_out[j]; //Sum of weight-input products
        O_out[i] = (O_out[i] + Bo[i]*127);        //Bias addition
        H_out[i] = H_out[i) >> 9-bit;             //Shift to right due to scaling
        O_out[i] = ReLU(O_out[i]);                // Activation Function , ReLU=max(O_out[i], 0)
        O_out[i] = Sat(O_out[i]);                 //Saturation to 8-bit


// Softmax
Label = arg max(O_out);                   // calculate the maximum between outputs
```

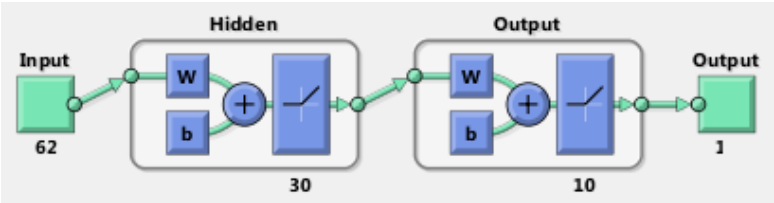

Figure 5. The required MLP network (MATLAB View)

- *In this project, you should implement the design based on access to limited hardware resources. Consider the case that only **10 hardware neurons** are provided as resources to perform all network computations (Figure 5). In this scheme, each neuron has a single **MAC** unit. If a layer has more than 10 neurons, the computations of this layer must be executed **serially** on the hardware (running the computations of 10 neurons in parallel each time).*
  Also, the computations of two consecutive layers (hidden and output) should be executed serially. The hidden layer should be processed before the output layer on the same set of MACs. The controller is responsible for arranging the data assignment to the hardware neurons in this design.

  - *For **MAC** unit, you can simply use the high level multiplication(a*b) in the Verilog code in this project (**Note that** to multiply two 8-bit sign-magnitude numbers, you should use the 7 * 7 unsigned multiplication and handle the sign-bit separately.)*

- *As mentioned before, to make better use of resources, all network computations should be performed using the limited number of Neurons (multiply-accumulate units (MACs)). Therefore, it is required to design a system **controller** and specify a **memory access pattern** (read and write).*
  *Use separate **single port memory buffers** for input data and network parameters such as weights and biases. Also to save the middle data (the output values of the hidden layer that are inputs of the output layer) you can use some **registers** (Figure 5).*

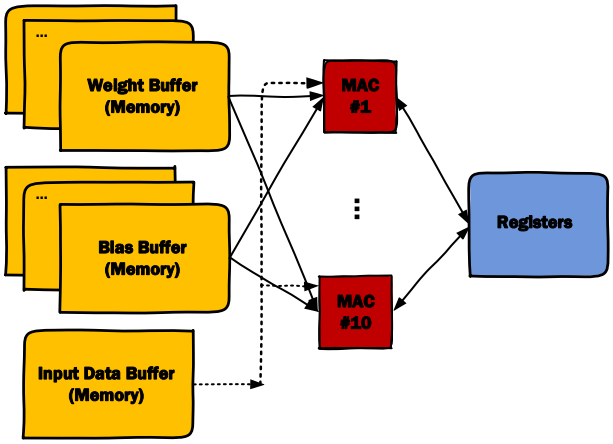

Figure 5. Resources for hardware implementation of neural network in this project

Note that *

- *Weights and biases are 8-bit with sign-magnitude representation.*
- *In the MAC structure each 8-bit weight must be multiplied by the corresponding 8-bit input, and each 8-bit bias value must be multiplied by +127(8-bit sign-magnitude) and then added to the sum of the weight-input products.*
- *The values of weights and biases and the network input test data are uploaded in separate files on the course page, which need to be stored in memory according to your design.(Please read the Guideline File in the project files folder)*
- *You should Consider 15 bits for multiplier output, 21 bits for MAC accumulator and 8 bits for the output of Relu Function.*

- *The output of the hidden layer must shift 9 bits to the right before entering the activation function (Due to the scaling of the output layer inputs).*
- *For passing the accumulator output through the activation function if the result value exceeds from 8 bits, the saturation operation should be performed. If a register has a **saturation** mode of operation, then an **overflow** and **underflow** condition is set to the maximum positive or negative value that are allowed (for example +127 and -127 in 8-bit sign-magnitude representation). It is also necessary to handle the sign bit correctly.*

Deliverables:

- *The **complete Verilog code** of the design and the Xilinx ISE/ Vivado synthesis and implementation results*
  - *Choose your target FPGA in the synthesis tool as Artix7-series family if you use Vivado and Virtex6-Series family when using ISE*
  - *Your design must be Synthesizable.*
- *A **testbench** that instantiates the design as a component and feeds it with a clock, and monitors its output*
- *You should assess the following and mention them in the report*
  - *The functionality and **accuracy** of the network based on the test data*
  - *The area and the device utilization consist of the number of LUTs, DSP blocks, flip flops, etc.*
  - *The performance (=1/Max Delay) of the network*
  - *The Power Consumption of different parts of the network*
- **A **comprehensive report** of the project's steps, a detailed block diagram of your design (Controller and Datapath structures), and the results.*

Good luck :)