Computer Aided Design (CAD) -Assignment Description

University of Tehran
Electrical and Computer Engineering Department
Fall 99

PRJ#1: Hardware implementation of a basic "Neuron" in Artificial Neural Networks (ANNs)

In this assignment, you have to implement a pre-designed model of a basic artificial neuron which is used in ANNs by using Verilog/VHDL language.

• INTRODUCTION

An Artificial Neural Network (ANN) is a computational model based on the structure and functions of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, in a sense - based on that input and output. ANNs are considered non-linear statistical data modeling tools where the complex relationships between inputs and outputs are modeled or patterns are found. An ANN is based on a collection of connected Basic units called "Artificial Neurons".

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 several neurons in the next layer, to which it sends data. Figure 1 shows an example of a 3-layer neural network with 3 inputs, 5 neurons in the hidden layer and 2 neurons at the output 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.

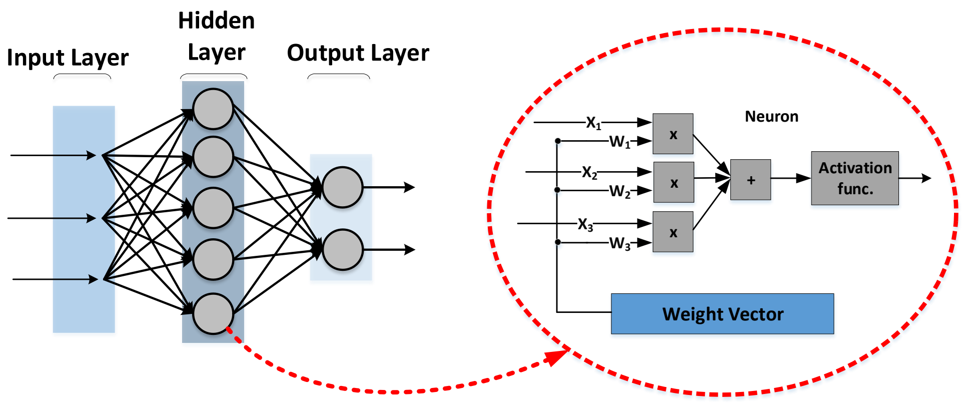

Fig1. A three-layer feed-forward MLP

- **PROJECT DESCRIPTION**

In this assignment, you should implement a single neuron, depicted in Figure 2. Each neuron in MLP computes the dot product of the inputs and weight vectors and passes the result to an activation function to produce the final output. The weight vector of each neuron is determined during an offline or online training phase. Figure 2 illustrates the required arithmetic operations of a basic neuron, but realistic implementations usually use a **MAC** (multiply-and-accumulate) unit for each neuron to carry out the multiplication/addition operations in serial. Figure 3 shows the implementation of a neuron by applying a MAC module.

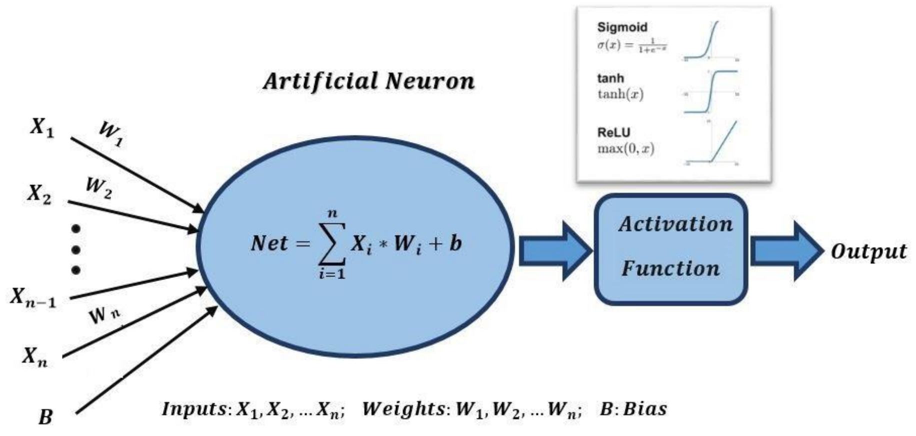
Artificial Neuron
X_1
W_1
X_2
W_2
X_{n-1}
W_n
X_n
B
$Net = \sum_{i=1}^{n} X_i * W_i + b$
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$
tanh
$\tanh(x)$
ReLU
$\max(0, x)$
Activation Function
Output
Inputs: $X_1, X_2, \dots X_n$; Weights: $W_1, W_2, \dots W_n$; B: Bias

Fig2. A basic artificial neuron model

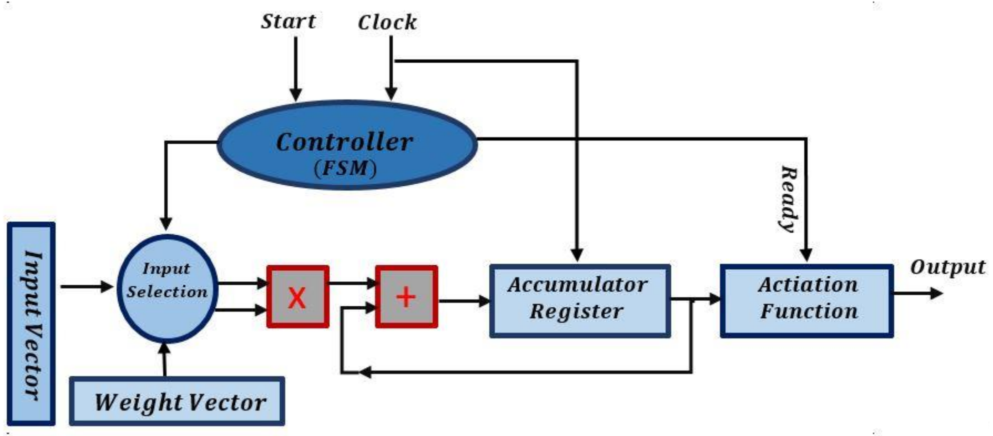

Fig3. A structural model of an artificial neuron

As illustrated in Figure 3, computation is initiated by receiving a start signal. The controller then reads each input data and its corresponding weight and passes them to the MAC unit in several iterations. Once the MAC calculation is completed, a ready signal is passed to the activation function unit to generate the neuron output.

Activation function for this step of the project is a simple threshold-based function ReLU (Rectified Linear Unit): if the MAC output is greater than 0.5, the MAC result is directly passed to the output, otherwise the neuron output is zero

$$\boldsymbol{Activation\ Function}: \quad \boldsymbol{f(Output)} = \begin{cases} \boldsymbol{Output} & \boldsymbol{Output > 0} \\ \boldsymbol{0} & \boldsymbol{Output \leq 0} \end{cases}$$

Note that *

- ✓ Design the controller by an **FSM**.
- ✓ The weight register, controller, adder, and multiplier must be constructed separately as different entities/modules and then be integrated into a structural design.
- ✓ For the Multiplier unit in MAC module you use:
  1) **A 8*8 multiplier based on the 2*2-bit Mult in CA1 and synthesized on FPGA LUTs**
     - o In this case all logics are synthesized and implemented on CLBs.
  2) **A 8*8 multiplier based on the Xilinx DSP IP-Cores.**
     - o In this case the synthesizer use the hardwired DSP cores in FPGAs.
     - o There are several ways for using DSP hardware resources (instead of LUTs): Using IP Core generator, instantiating the resources in the code or using Coding Style. **You should use the **IP-Core Generator** in this assignment. Find "Multiplier" in IP Catalog and select "DSP" or "Mult" for generating IP Core.*
- ✓ The MAC module receives one neuron input and one weight in each iteration. If the neuron has *n* inputs, they are multiplied and accumulated in *n* consecutive cycles. It is controlled by an FSM.
- ✓ The number of neuron inputs in Figure 2 (and the number of iterations that the inputs are applied to the MAC) should be changeable.
- ✓ Assume one MAC operation (one multiply and one add) can be completed in a single clock cycle.
- ✓ Fill weight and input vectors with random numbers that range between -127 to +127with sign-magnitude signed numbers.
  - o Multiplying signed numbers. To support both positive and negative numbers by the multiplier, you should consider the sign bit as the leftmost of the operands to have sign-magnitude representation of them. "0" indicates a positive integer, and "1" indicates a negative integer. You should handle the sign of the result value in the multiplier by using logical gates.

Deliverables:

- *The **complete HDL code** of the design and the Xilinx ISE or Vivado synthesis and implementation results*
  - o *Choose your target FPGA in the synthesis tool as Artix7-series family if you use Vivado and Virtex6-Series family when using ISE*
  - o *Your design must be **Synthesizable.***
- *A **testbench** that instantiates the design as a component and feeds it with a clock and monitors its output*
- *A script for compiling and simulating the whole design.*
- *A **table** comparing the design specifications based on the two specified multipliers*
  - o *The area and the device utilization consist of the number of LUTs, DSP blocks, flip flops,etc.*
  - o *The performance(=1/Max Delay) of the neuron*
  - o *The Power Consumption of different parts of the neuron*
- **<u>A **comprehensive report** of the steps involved in the project, detailed block diagram of your design and the results</u>. The report also must include the FSM of the controller unit*

The format of your files in trunk folder:

Doc: *Please place your report files here*
Prj: *Please place your Vivado or ISE project here*
Sim:

- ✓ ***File:*** *Please place your input or output files(.dat, .txt, .hex) here*
- ✓ ***Model:*** *Please place your high level model codes(c, c++, mat,py,…) here*
- ✓ ***Tb:*** *Please place your testbench modules here*
- ✓ ***sim_top.tcl:*** *To simulate testbench run this script file with do command in modelsim: "do sim_top.tcl"*

Src:

- ✓ ***Constraint:*** *Please place *.ucf files for ise and *.xdc files for Vivado here*
- ✓ ***Hdl:*** *Please place all RTL codes here*
- ✓ ***IP:*** *Please place generated IPs here*
- ✓ ***Inc:*** *Please place include files here*

The name of your final Zip file:: "CAD99_Your Student Number_PRJ1.zip"

Good luck :)