University of Tehran
Electrical and Computer Engineering Department
Computer Aided Design (CAD) – Fall 96

CA3: **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 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 nonlinear 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.

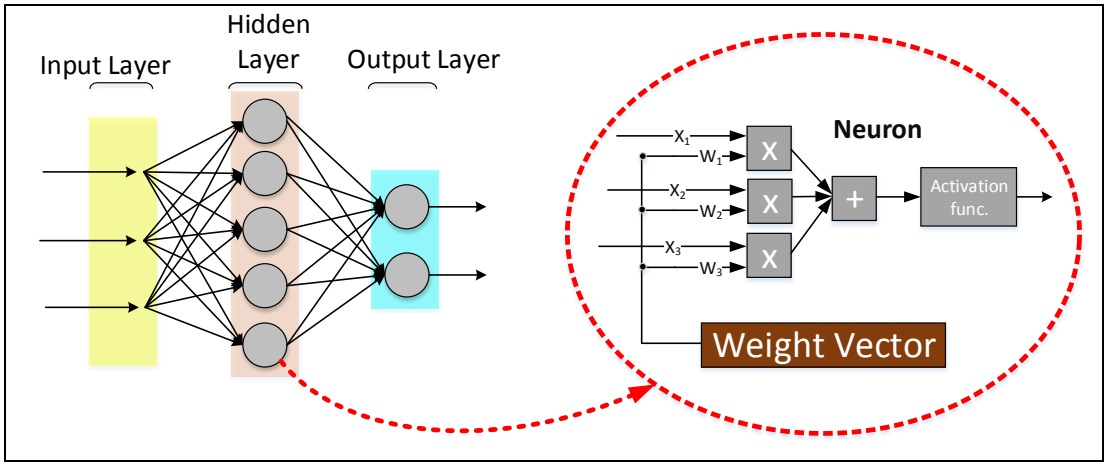

Fig1. A three-layer feed-forward MLP

- **Project Definition**

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.

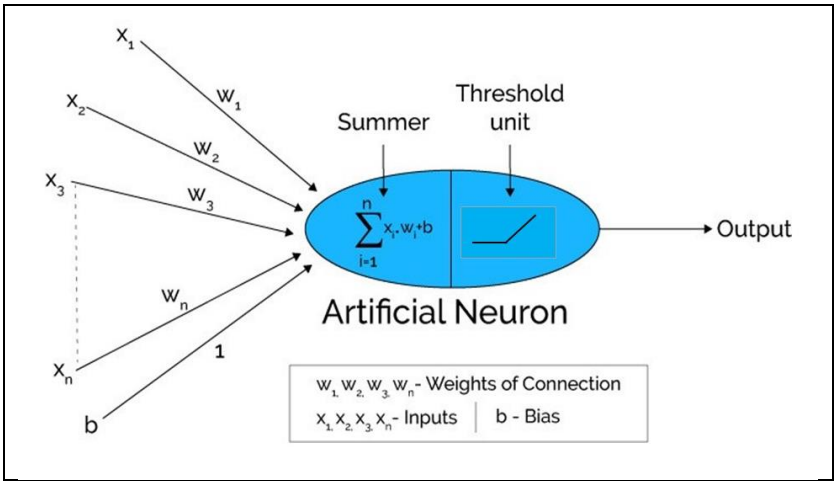

Fig2. A basic artificial neuron model

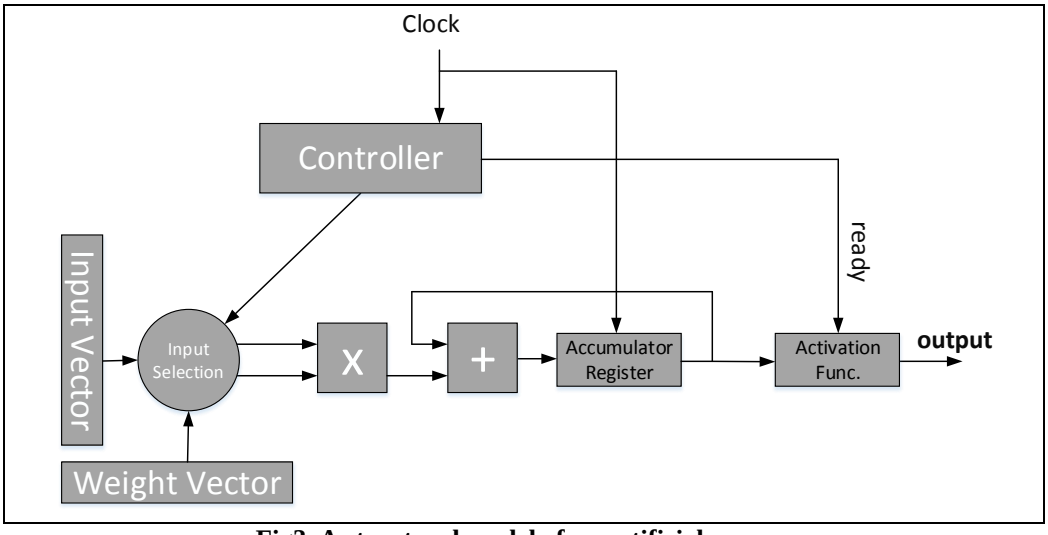

Fig3. A structural model of an artificial neuron

As illustrated in Figure 3, the controller 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: if the MAC output is greater than 0.5, the MAC result is directly passed to the output, otherwise the neuron output is zero. We will use more realistic activation functions later.

$$\textbf{\textit{Activation Function}}: \quad f(\textbf{\textit{Output}}) = \begin{cases} \textbf{\textit{Output}} & \textbf{\textit{Output}} > \mathbf{0.5} \\ \mathbf{0} & \textbf{\textit{Output}} \leq \mathbf{0.5} \end{cases}$$

Also please note that weight register, controller, adder, and multiplier must be constructed separately as different entities and then be integrated into a structural design.

- The number of neuron inputs (and so, the length of the two vectors in Figure 3) must be parameterized. Set it to some integer value when you are testing the design.
- For this step of the project, you can define all numbers as the real data type and simply use the dataflow + and * operations to build the adder and multiplier units.
- Assume one MAC operation (one multiply and one add) can be completed in a single cycle.
- Fill weight and input vectors with random real numbers that range between -1 to +1.

- **Deliverables:**
  - The complete code of the design in VHDL
  - A testbench that instantiates the design as a component and feeds it with a clock and monitors its output

Good luck