Chapter 5
A Reconfigurable Hardware for Artificial Neural Networks

Abstract. Artificial Neural Networks (ANNs) is a well known bio-inspired model that simulates human brain capabilities such as learning and generalization. ANNs consist of a number of interconnected processing units, wherein each unit performs a weighted sum followed by the evaluation of a given activation function. The involved computation has a tremendous impact on the implementation efficiency. Existing hardware implementations of ANNs attempt to speed up the computational process. However these implementations require a huge silicon area that makes it almost impossible to fit within the resources available on a state-of-the-art FPGAs. In this chapter, we devise a hardware architecture for ANNs that takes advantage of the dedicated adder blocks, commonly called MACs to compute both the weighted sum and the activation function. The proposed architecture requires a reduced silicon area considering the fact that the MACs come for free as these are FPGA’s built-in cores. The hardware is as fast as existing ones as it is massively parallel. Besides, the proposed hardware can adjust itself on-the-fly to the user-defined topology of the neural network, with no extra configuration, which is a very nice characteristic in robot-like systems considering the possibility of the same hardware may be exploited in different tasks.

5.1 Introduction

Artificial Neural Networks (ANNs) are useful for learning, generalization, classification and forecasting problems [3]. They consists of a pool of relatively simple processing units, usually called artificial neurons, which communicates with one another through a large set of weighted connections. There are two main network topologies, which are feed-forward topology [3], [4] where the data flows from input to output units is strictly forward and recurrent topology, where feedback connections are allowed. Artificial neural networks offer an attractive model that allows one to solve hard problems from examples or patterns. However, the computational process behind this model is complex. It consists of massively parallel non-linear

* This chapter was developed in collaboration with Rodrigo Martins da Silva.
calculations. Software implementations of artificial neural networks are useful but hardware implementations take advantage of the inherent parallelism of ANNs and so should answer faster.

Field Programmable Gate Arrays (FPGAs) [7] provide a re-programmable hardware that allows one to implement ANNs very rapidly and at very low-cost. However, FPGAs lack the necessary circuit density as each artificial neuron of the network needs to perform a large number of multiplications and additions, which consume a lot of silicon area if implemented using standard digital techniques.

The proposed hardware architecture described throughout this chapter is designed to process any fully connected feed-forward multi-layer perceptron neural network (MLP). It is now a common knowledge that the computation performed by the net is complex and consequently has a huge impact on the implementation efficiency and practicality. Existing hardware implementations of ANNs have attempted to speed up the computational process. However these designs require a considerable silicon area that makes them almost impossible to fit within the resources available on a state-of-the-art FPGAs [1], [2], [6]. In this chapter, we devise an original hardware architecture for ANNs that takes advantage of the dedicated adder blocks, commonly called MACs (short for Multiply, Add and Accumulate blocks) to compute both the weighted sum and the activation function. The latter is approximated by a quadratic polynomial using the least-square method. The proposed architecture requires a reduced silicon area considering the fact that the MACs come for free as these are FPGA’s built-in cores. The hardware is as fast as existing ones as it is massively parallel. Besides, the proposed hardware can adjust itself on-the-fly to the user-defined topology of the neural network, with no extra configuration, which is a very nice characteristic in robot-like systems considering the possibility of the same piece of hardware may be exploited in different tasks.

The remaining of this chapter is organized as follows: In Section 5.2 we give a brief introduction to the computational model behind artificial neural networks; In Section 5.3 we show how we approximate the sigmoid output function so we can implement the inherent computation using digital hardware; In Section 5.4 we provide some hardware implementation issues about the proposed design, that makes it original, efficient and compact; In Section 5.5 we present the detailed design of the proposed ANN Hardware; Last but no least, In Section 5.6 we draw some useful conclusions and announce some orientations for future work.

5.2 ANNs Computational Model

We now give a brief introduction to the computational model used in neural networks. Generally, is constituted of few layers, each of which includes several neurons. The number of neurons in distinct layers may be different and consequently the number of inputs and that of outputs may be different [3].

The model of an artificial neuron requires n inputs, say \( I_1, I_2, \ldots, I_n \) and the synaptic weights associated with these inputs, say \( w_1, w_2, \ldots, w_n \). The weighted sum \( a \), which, also called activation of the neuron, is defined in (5.1). The model usually