Neural Networks FPGA Controller for Reconfigurable Antennas

E. Al Zuraiqi (1), M. Joler (2), C. G. Christodoulou (1)

(1) ECE Dept., University of New Mexico, Albuquerque, NM 87131
(2) CE Dept., Faculty of Engineering, University of Rijeka, 51000 Rijeka, Croatia

Abstract
In this paper, a neural network (NN) field programmable gate array (FPGA) controller is designed. NN-FPGA controller implies building of a cost efficient neural network by using customizable blocks designed in the Simulink and Xilinx System Generator. Network parameters are mapped into a hardware structure that improves the performance and the efficiency.

Introduction

In order to avoid the computational complexities involved in analyzing reconfigurable antennas, neural networks are used with reconfigurable antennas. NNs drastically reduce the computational complexities involved in the numerical modeling of reconfigurable antennas [1]. The mapping between the received signal and the antenna's behavior is a continuous function, and therefore it is possible to model it with a NN trained at discrete samples along the function. The inherent nonlinearities associated with antenna radiation patterns make antennas very suitable candidates for NNs. Also, a fully adaptive antenna array implementation requires a considerable increase in processing requirements. So, by using FPGAs, we have powerful DSP devices for handling these high performance requirements at sampled data rates. Furthermore, we can take advantage of the FPGA flexibility for directly handling acquisition control and other DSP functions, such as digital down-conversion, demodulation, and matched filtering.
Neural Networks rely on massive parallel computation. Thus, the high speed operation in real time applications can be achieved only if the NNs are implemented using parallel hardware architecture. NN will design the FPGA circuit using a standard back-propagation technique, as the back-propagation approach provides meaningful cues that steer the design to convergence [2]. FPGAs are used for NN implementation due to accessibility, ease of fast reprogramming, and low cost, permitting the fast and non-expensive implementation of the whole system. NN-FPGA architectures can be easily reconfigured to adapt the weights and topologies of a NN.

NN-FPGA Controller

Implementing a neural network in a reconfigurable architecture like FPGA is the best way to calculate weights and network topologies. Network parameters are mapped into a hardware structure that improves the performance and the efficiency. The size of the implemented NN is limited by the block RAM capacity of the FPGA board.

The size of a NN is the amount of neurons synaptic weights available. The number of synaptic weight connections (N_{SW}) available for FPGA board is [3]:

$$N_{SW} = \frac{\text{(FPGA Block RAM Size)(\# of Block RAMs)}}{\text{Size of Synaptic Weights}}$$

To ensure effective utilization of FPGA board resources, a highly efficient NN should be built. Network efficiency is the percentage of processing elements operating in parallel.

$$\text{Eff} = \frac{\text{\# of neurons in the smallest layer}}{\text{\# of neurons in the largest layer}} \text{x}100\%$$

Maximum network efficiency is achieved when the network layers have the same number of neurons. Thus, to increase the efficiency, neurons in the largest layer are multiplexed in groups.

NN modeling via an FPGA procedure

- Setting up of a complete NN that is capable of completely modeling a section of the related FPGA.
- Proper choice of number of bits, synaptic weights, and hidden neurons.
- Training the NN with the desired behavioral specifications.
- The fully trained weights coefficients and network topology are then configured on the FPGA.

NN Hardware Architecture

The system in Figure 1 below represents a four-input neuron with bias and sigmoid excitation function [4]. The neuron is built using adders and multipliers from MatLab Simulink blocks. The Xilinx "Gateway In" block converts MatLab Simulink blocks data types (integer, double, and fixed-point) into the System Generator fixed-point type. The excitation function is built entirely of Xilinx System Generator blocks.

The System Generator block provides control of system and simulation parameters, and is used to invoke the code generator.

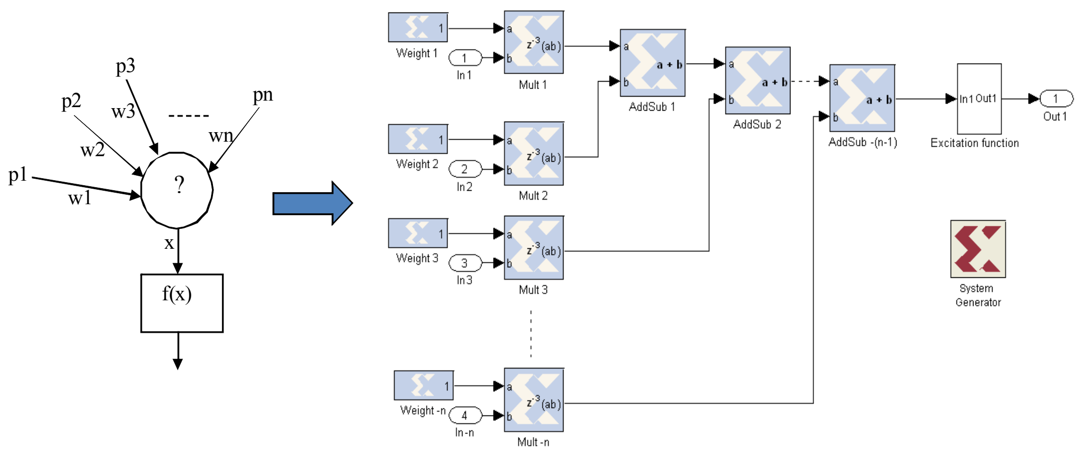

Fig. 1: Single Neuron Modeling Using Xilinx FPGA System Generator

The network blocks are built in Simulink environment, and use basic Xilinx blocks. The control block manages the control signals of the neurons; it provides neurons inputs, clock, and enable signals for other blocks. The winner neuron is the neuron with the weight vector that closely matches the input. Figure 2 below shows a block diagram for the proposed NN-FPGA controller.

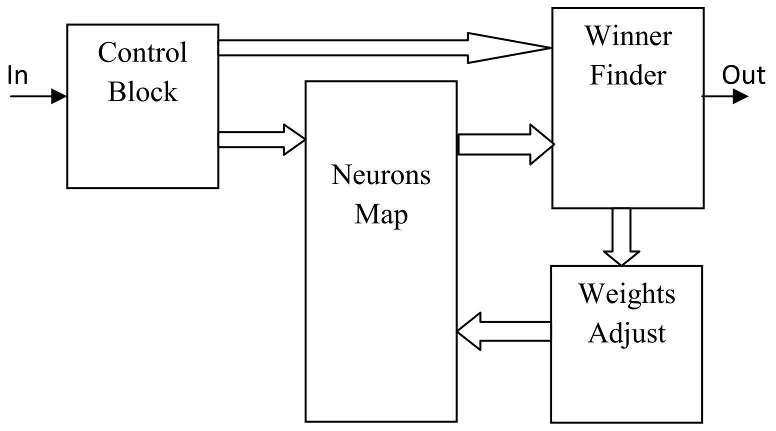

Fig. 2: NN-FPGA controller hardware architecture

The Neurons Map block (Figure 3) contains the neurons (in Figure 1) and their connections that cover both learning phase and propagation phase. In this block the weights and connections are adjusted and compared to the input pattern vector, the difference is sent to Winner Finder block.

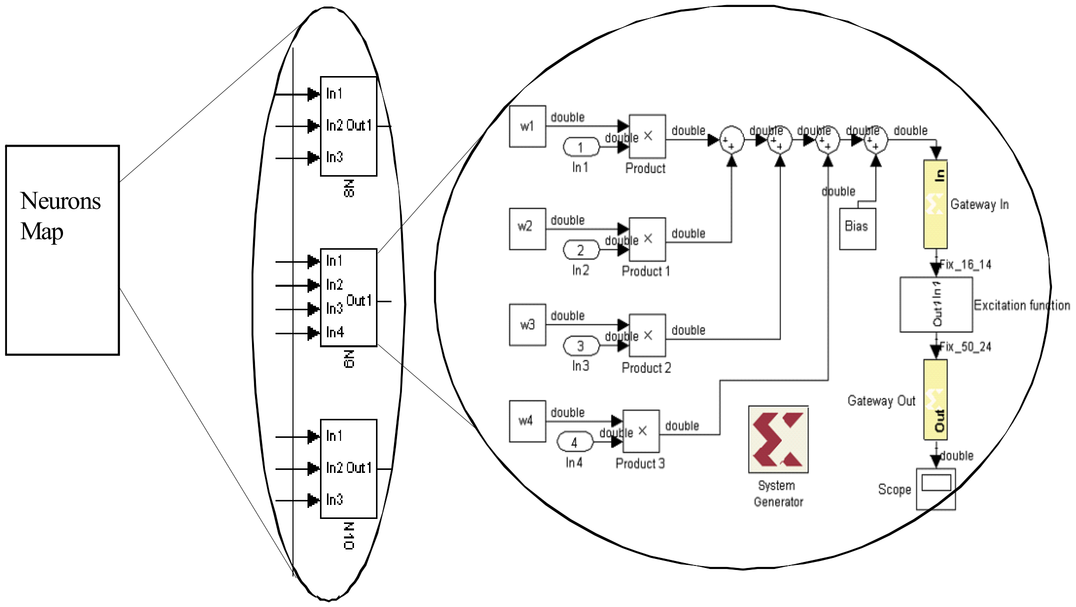

Fig. 3: Neurons Map block structure

Conclusion

In this paper, a neural network FPGA controller was described and designed. The network blocks were built in MatLab Simulink environment, and use basic Xilinx System Generator blocks. Network parameters were mapped into a hardware structure that improves both performance and efficiency of the network.

References:

[1]. A. Patnaik, D. Anagnostou, C. G. Christodoulou, J. C. Lyke, "Modeling frequency reconfigurable antenna array using neural networks", Microwave and Optical Technology Letters, Vol. 44, No. 4, pp 351-354, Feb 2005.
[2]. J. Lyke, "A Cellular Automata FPGA Architecture that can be Trained with Neural Networks", Aerospace Conference Proceedings, Vol. 5, pp 2347-2354, 2002.
[3]. M. Bonnici, E. J. Gatt, J. Micallef, I. Grech, "Artificial Neural Network Optimization for FPGA", 13th IEEE International Conference on Electronics, Circuits and Systems, pp 1340-1343, Dec 2006.
[4]. J. L. Bastos, H. P. Figueroa, A. Monti, "FPGA Implementation of Neural Network-Based Controllers for Power Electronics Applications", Applied Power Electronics Conference and Exposition, pp 1443-1484, March 2006.