Methods for embedded systems design with on-chip learning neural networks

A. Tisan, C. Gavrincea, S. Oniga, A. Buchman

Electrotechnical Department
North University of Baia Mare
Baia Mare, 430083, Romania
0362-401265, 0262-276153, atisan@ubm.ro

Abstract

In this paper, we propose a method to implement in FPGA circuit (Field Programmable Gate Array) an embedded system with on-chip learning neural network. The architecture proposed herein takes advantage of distinct modules for controlling the peripherals of the development board and for data processing as forward and backward stages of the propagation and learning phase. The architecture is easily scalable and able to cope with arbitrary network sizes with the same hardware.

1. INTRODUCTION

Signal processing systems for pattern recognition will have to operate in rapidly changing environments. To suitably adapt to the varying requirements, control strategies targeted at selecting and tuning the signal processing algorithms need to be developed.

For modeling dynamical systems, as neural networks with on-chip learning, we have designed a hardware – software platform powerful and flexible enough to allow NNs of different sizes to be efficiently computed.

We have developed an extendable digital architecture for the implementation of a neural network (NN) with on-chip learning using field programmable gate arrays (FPGAs) and a design methodology that allows the system designer to concentrate on a high level functional specification.

For this, we have created a new library Simulink block set constituted by Simulink Xilinx blocks, MCode blocks, VHDL blocks and an EDK processor block. With these new blocks, the designer will be able to develop the entire neural network by parameterize the ANN topologies as number of neurons and layers.

The implementation goal is achieved by using the Mathworks' Simulink environment for functional specification and System Generation to generate the VHDL code according to the characteristics of the chosen FPGA device.

The Xilinx System Generator is used in order to provide the capability to model and implement high performance digital processing systems in field-programmable gate arrays (FPGAs) using Simulink.

The Xilinx Blockset contains bit and cycle-true models of arithmetic and logic functions, memories, and DSP functions for parallel processing such as artificial neural networks.

Another import role of the System Generator is to converts a Simulink model of Xilinx blocks into an efficient hardware implementation that combines synthesizable VHDL and intellectual property blocks that have been developed to run efficiently in FPGAs.

2. EMBEDDED SYSTEM DESIGN

The entire proposed concept relies on the idea that a FPGAs implementable neural network can be reached only by choosing the predesigned generic blocks and to set the parameters of the network into a pop-up window.

2.1. Peripherals control block design

In order to control the peripherals of the development board, the MicroBlaze processor is used.

The MicroBlaze is a standard 32-bit RISC Harvard-style Soft Processor, which is especially developed for the Spartan-3-based FPGA architecture.

The MicroBlaze embedded processor soft core is a reduced instruction set computer (RISC) optimized for implementation in Xilinx Field Programmable Gate Arrays (FPGAs). The functional block diagram of the MicroBlaze core is shown in Figure 1

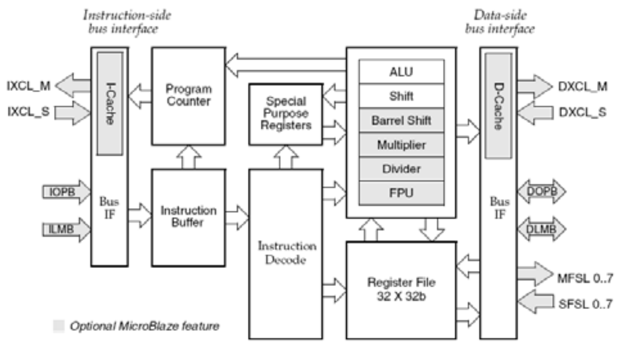


Fig. 1. MicroBlaze Core Block Diagram.

The MicroBlaze soft core processor is highly configurable, allowing designers to select a specific set of features required by the design and is parameterized to allow selective enabling of additional functionality.

The MicroBlaze core has been developed to support a high degree of user configurability. This allows tailoring of the processor to meet specific cost/performance requirements. Configuration is done via parameters that typically enable, size, or select certain processor features.

For a complete hardware designed of the embedded processor system, the Xilinx Embedded Development Kit (EDK) was used.

The simplified flow for an embedded design is presented in figure 2, and includes hardware and software development, a device configuration with needed IPs and a verification of those.

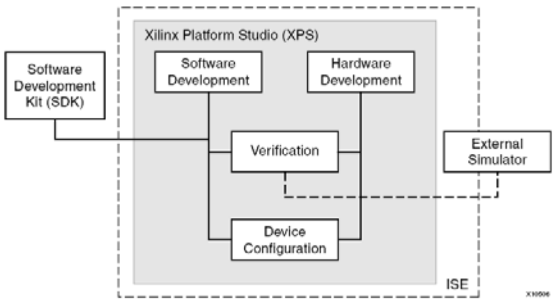


Fig. 2. Embedded Design Process Flow.

In this paper, the used IPs for peripherals of the development board controlling and includes: a RS-232 port, slide and push-button switches and LCD module.

For RS-232 port control, the UART Lite module - provided by Xilinx - was used with the following features: 8-bit bus interfaces, one transmits and one receives channel (full duplex) and a configurable baud rate. But, the main advantage of using of the UART module is the low resource utilization for hardware implementation, about 50 flip-flops and 100 LUTs.

For slide and push-button switches control, the IP used is a Xilinx-provided IP and get access to the four slide switches and five push button switches. These switches are use to control the development board and the phases of the neural network.

For controlling the LCD module, it was necessary to add in a custom peripheral. For that, it was required to create a user peripheral from an HDL module, add an instance of the imported peripheral, and modify the system's user constrain file to provide an interface to the on-board LCD module.

The embedded hardware platform that includes the processors, along with peripherals and memory blocks is presented in figure 3. These blocks of IP, use an interconnect network to communicate and additional ports to connect with the peripherals.

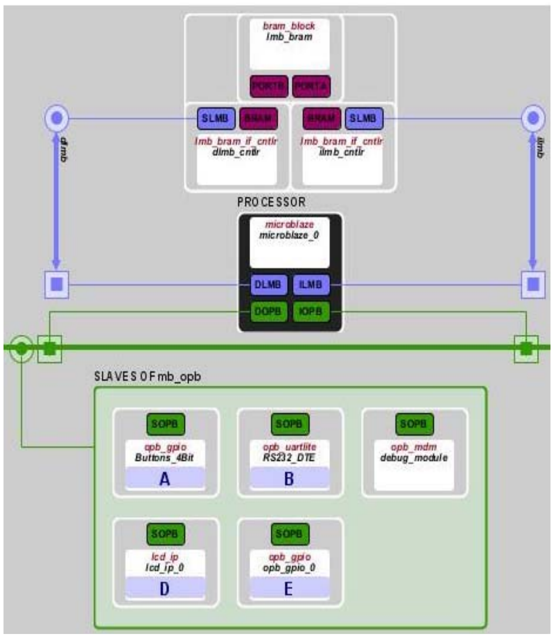


Fig. 3. MicroBlaze Block Diagram.

After the embedded system was configured, the embedded hardware platform is imported into System generator project as a Black Box.

The imported MicroBlaze processor interface is exposed through the EDK processor block provided by System Generator. In the figure 4 is shown the communication between user-defined logic and the MicroBlaze processor.

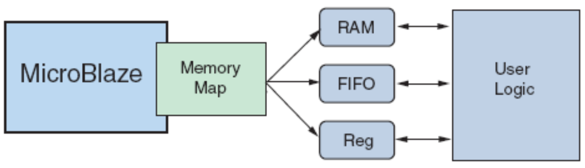


Fig. 4. Memory-Mapped User Logic

The memory-map needed for communications is automatically created by the System generator and is presented in figure 5.

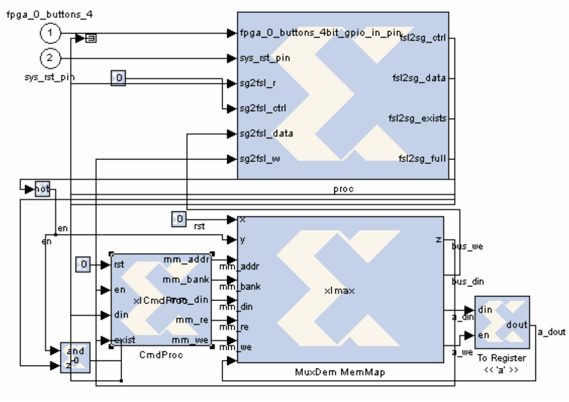


Fig 5: Generated Memory-Mapped

2.2. *Neural network design*

The neural network designed deals with an extendable digital architecture for the implementation of a multilayer feedforward networks (MLF) and Hebbian neural network using field programmable gate arrays (FPGAs).

In order to design generic blocks, used for neural network building, it was developed an algorithm for setting the customizable block parameters according to the chosen network topology.

From point of view of the role of the blocks into design, these are divided in control blocks and computing blocks, figure 6.

2.2.1. *Control blocks*

The control blocks are designed in MCode and VHDL code and incorporated into system by Black Box HDL and MCode Blocks. The role of these blocks is to manage the control signal of the processing bloc in order to initialize and command the processing components

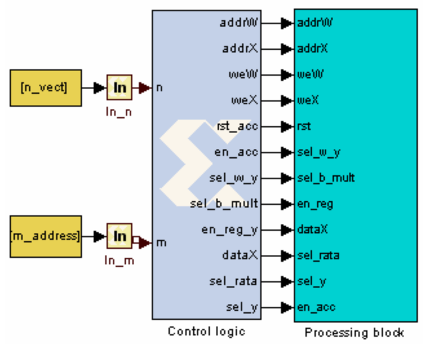


Fig. 6. Neural network Blocks

An important feature of the control block is to load from Mathlab workspace the following variables: the number of vectors used for training and the number of bits used for data representation.

Depending on these variables, the control logic block will configure the size of the RAMs used for data and weights storage and will manage the enable signals of the processing elements of the processing block in order to run the processing block in a propagation phase or in a training phase

The control block consists of one General Counter block for all the layers and one Signal Generator block for each layer in part.

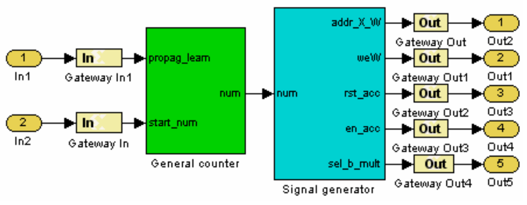
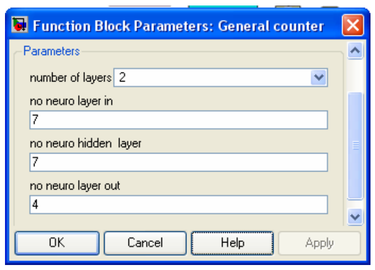


Fig.7. Architecture of the control block

The properties of the block are set according to the numbers of total layers, number of the neurons from the layers and the number of layer from whom and in different Function Block Parameters window, fig 8.


Function Block Parameters: General counter

Parameters

number of layers 2

no neuro layer in

7

no neuro hidden layer

7

no neuro layer out

4

Fig.8. Function Block Parameters window for parameters setting

2.2.2. PROCESSING BLOCK

The processing blocks are the main block of the design. It incorporates both the artificial neuron and the logic for on-chip learning algorithm.

The structure of the artificial neuron consist in two memory blocks, one for data samples and one for weight coefficients, and one MAC unit, figure 9

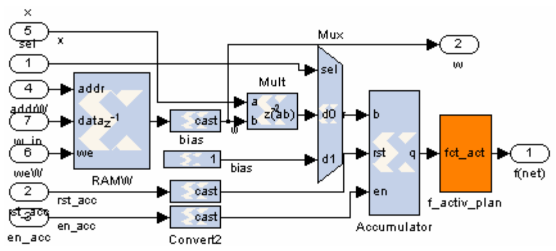


Fig 9. Architecture of the neuron

In order to complete the neural network design, beside the main processing block, we have created block that are able to calculate the new weights that minimize the total error. That means a series of blocks that calculate the proper delay of the signals and some blocks that computes the new weights of the output layer, the error signal for each neuron and the values of the corrected weights.

The overview architecture of the designed neural network is presented in figure 10 and includes as an inputs one input layer with seven neurons and one target layer, one hidden layer with seven neurons and an output layer with four neurons

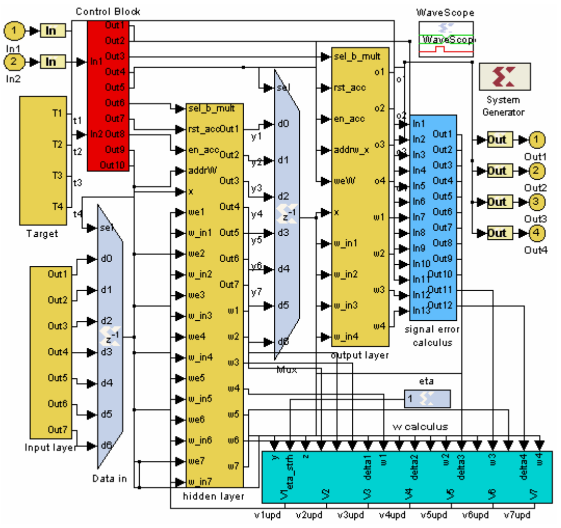


Fig. 10. The architecture of the neural network

4. CONCLUSIONS

We have presented hardware architecture of embedded system with on-chip learning controlled by a generic control unit described in VHDL code, MCode and Xilinx blocks. This method uses minimal hardware resources for implementation of this kind of data processing systems. The main advantage of this solution is high modularity and versatility in neural network designing.

The advantage of developing an embedded system in EDK environment is that the computing core and I/O resources are linked under the control of the bitstream by a programmable interconnect architecture that allows them to be wired together into systems. FPGAs are high performance data processing devices and its performance is derived from the ability they provide to construct highly parallel architectures for processing data.

A combination of increasingly high system clock rates and a highly distributed memory architecture gives the system designer an ability to exploit parallelism in neural network applications that operate on data streams.

The neural network architecture is build with generic modules and used to design neural networks that have the following features: on-line training and on-chip learning, all weights have to be initialized prior to the start of the learning process, the initialization of the neurons number per layer, number of layers, weights RAMs must be done by setting the variables from the Function Block Parameters from Matlab environment.

REFERENCES

[1] A. Tisan, S. Oniga, A. Buchman, C. Gavrincea, “Architecture and Algorithms for Syntetizable Neural Networks with On-Chip Learning”, *International Symposium on Signals, Circuits and Systems*, ISSCS 2007, July 12-13, 2007, Iasi, Romania, vol.1, p. 265 - 268, ISBN 1-4244-0968-3, IEEE Catalog Number: 07EX1678, Library of Congress: 2007920356.

[2] A. Tisan, S. Oniga, C. Gavrincea, Hardware Implementation of Various Neural Network with On-Chip Learning, *WSEAS TRANSACTIONS on SIGNAL PROCESSING*, Issue 10, Volume 2, October 2006, ISSN 1790-5022.

[3] EDK Concepts, Tools, and Techniques, www.xilinx.com/ise/embedded/edk91i_docs/edk_ctt.pdf.

[4] Xilinx System Generator for DSP, User’s guide, www.xilinx.com/Support/sw_manuals/sysgen_gs.pdf.