2010 International Symposium on Nonlinear Theory and its Applications
NOLTA2010, Krakow, Poland, September 5-8, 2010

Hardware-Software Co-design of Nonlinear Active Wave Generator with Microblaze Soft Core Processor

Selman Ergünay, Ramazan Yeniçeri and Müştak E. Yalçın

Department of Electronics and Communication Engineering,
Istanbul Technical University,
Istanbul, Maslak, TR-34469, Turkey.
Email: {selman.ergunay, ramazan.yeniceri, mustak.yalcin}@itu.edu.tr

Abstract—This paper introduces a hardware - software co-design of a Cellular Nonlinear Network (CNN) emulator. The Cellular Nonlinear Network is an two-dimensional array of locally coupled Nonlinear Processing Elements (NPEs). The network is used to observe evolution of spatio-temporal waves such as autowaves, travelling waves and spiral waves. The software part of the design controls the custom NPEs which are the hardware part of the design. In order to reach high performance, the software part provides control flexibility on hardware part which is vital for developing and executing active wave based computing algorithms. The system has been implemented on Xilinx Spartan 3E1600E FPGA with Microblaze soft core processor using Embedded Development Kit (EDK) design tools.

1. Introduction

Active wave based computing algorithms have received considerable attentions in the recent years because of new findings on biological systems. Biological systems can be considered as continuous distributed information processing system and process biological information. Retina is a good example for such a system which uses spatio-temporal waves for processing the visual information [2, 1].

In order to mimic the way biological systems process biological signals a practical network model is needed. Most recently a two dimensional reaction-diffusion Cellular Neural Network (CNN) [4] which consists of relaxation oscillators has been presented for wave computing applications. A fast FPGA implementation of real time autowave generation has been introduced in [5] and its application in robotic using an active wave computing algorithm has been presented in [7].

This paper is organized as follow: Section 2 the NPE which is the cell implementation of the network and its mathematical model is described. In Section 3, the hardware architecture of the network is explained and the software design is given in Section 4. The paper is concluded in Section 5.

2. Nonlinear Processing Element

A nonlinear processing element of the CNN model is described by the differential equation in (1).

$$\begin{aligned} \dot{x}_{i,j} &= \alpha x_{i,j} + \beta y_{i,j} + g(x_{i,j}) + I_{i,j} + u_{i,j}, \\ \dot{y}_{i,j} &= \epsilon x_{i,j} + \sigma y_{i,j} \end{aligned} \tag{1}$$

Where x and y are the state variables, $g(.)$ is the nonlinear function:

$$g\big(x_{i,j}\big) = \begin{cases} \mu \cdot (x_{i,j} - \lambda) & \text{if } x_{i,j} > \lambda; \\ 0 & \text{if } |x_{i,j}| \leq \lambda; \\ \mu \cdot (x_{i,j} + \lambda) & \text{if } x_{i,j} < -\lambda; \end{cases} \tag{2}$$

and the synaptic law $I_{i,j}$ of the model is given by

$$\begin{aligned} I_{i,j} \quad = \quad & a_{i,j+1}x_{i,j+1} + a_{i-1,j}x_{i-1,j} \\ & + a_{i,j-1}x_{i,j-1} + a_{i+1,j}x_{i+1,j}. \end{aligned} \tag{3}$$

The given network model (1) which has been introduced by Yalcin in 2008 [4] is a locally coupled relaxation oscillators. Any cell on this network can be fixed to constant value $x_{i,j} = x_{fixed}$. This is one of the essential features (called fixed-state) which has been already used in the active wave computing algorithms. Figure 1a and 1b show two snapshots depicting the obtained autowaves during the time evolution for two different parameter sets [4].

(a) (b)

Figure 1: Propagation of the autwaves on the network.

The discrete-time model of the model (1) is given by

$$\begin{aligned} x_{i,j}(k+1) = \quad & x_{i,j}(k) + \tau\Big[\alpha x_{i,j}(k) + \beta y_{i,j}(k) \\ & + g\big(x_{i,j}(k)\big) + I_{i,j}(k) + u_{i,j}\Big], \\ y_{i,j}(k+1) = \quad & y_{i,j}(k) + \tau\Big[\epsilon x_{i,j}(k) + \sigma y_{i,j}(k)\Big], \end{aligned} \tag{4}$$

where

$$g(x_{i,j}(k)) = \begin{cases} m \cdot (x_{i,j}(k) - \lambda) & \text{if } x_{i,j}(k) > \lambda; \\ 0 & \text{if } |x_{i,j}(k)| \leq \lambda; \\ m \cdot (x_{i,j}(k) + \lambda) & \text{if } x_{i,j}(k) < -\lambda; \end{cases} \tag{5}$$

and

$$I_{i,j}(k) = \begin{array}{l} a_{i,j+1}x_{i,j+1}(k) + a_{i-1,j}x_{i-1,j}(k) \\ +a_{i,j-1}x_{i,j-1}(k) + a_{i+1,j}x_{i+1,j}(k), \end{array} \tag{6}$$

The simple Forward Euler Method has been used to discretize the model. The dicrete-time model has been already implemented on an Field Programmable Gate Array in [5]. In order to use the network as an active wave computer, an improved design has been presented in [6].

The operations to compute the next-time value of the state variables x and y are done by a special digital hardware called NPE. In Figure 2, a detailed block diagram of the NPE circuit is given.

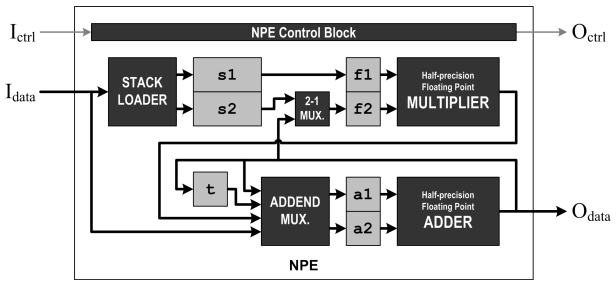

Figure 2: NPE block diagram.

Half-precision floating-point number arithmetic is used for the NPE implementation. The parameters of the network is first loaded into the NPE and then NPE calculates the state values of the next-time at a short interval with respect to its arithmetic workload.

In the following section how NPEs are organized to emulate the CNN is described and the observed dynamical behavior on the network is presented.

3. System Architecture

The NPE are organized as 5×5 processor array in [5] and 4×4 processor array in [6]. These designs are introduced a full hardware solution to emulate the CNN. The operation of the NPE arrays in these works are controlled by specially designed hardwares which only improves performance but do not have flexibility. While designing pure hardware it is required to change digital circuitries, connections between them and consider time scheduling to change the dynamic behavior of the network emulator.

In this work, there NPEs are operated by a Microblaze processor. The nonlinear network is emulated by NPE hardwares and the software runs on the Microblaze that loads values to NPEs, runs and collects computed values from them. If it is asked to add supplement expressions to the network model such as a new nonlinearity, new synaptic law or a new state equation, this requirement can be met by just changing the software. Hence a tradeoff between flexibility and speed exists. Block diagram of the design is given in Figure 3.

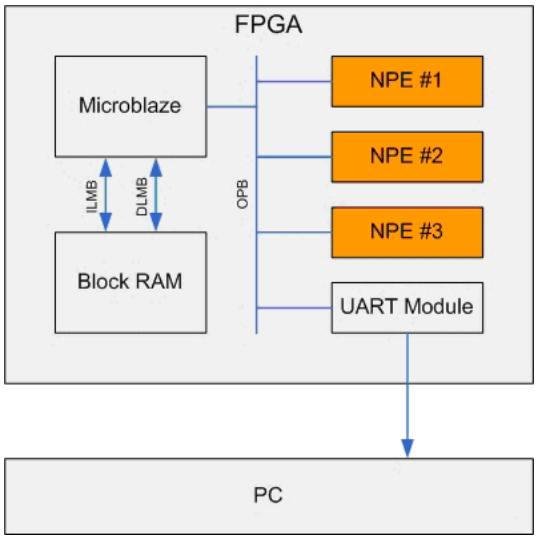

Figure 3: System architecture.

In this paper, Xilinx Embedded Development Kit (EDK) [3] which is a set of tools that enables to design and implement processor based digital system in Xilinx FPGAs is used to build up the system. One of these tools is Xilinx Platform Studio (XPS) to make hardware project of system and the other one is Platform Studio Software Development Kit (SDK) for preparing software part of design. While configuring microprocessor and interconnection with peripherals, XPS is used. SDK provides creation and verification C software application.

A 32-bit MicroBlaze soft core processor which is a reduced instruction set computer (RISC) and optimized for Xilinx FPGAs is added to the design. Instruction and data accesses are done in separate address spaces because MicroBlaze is implemented with Harvard memory architecture. These address spaces are located in 64 Kbyte Block RAM. Owing to BRAM is chosen as boot memory, reset vector of MicroBlaze points to the start address of Block RAM. When bitstream of design is sent to FPGA, Microblaze starts to execute the software.

In this design, Microblaze has two different types of data buses, Processor Local Bus (PLB) to communicate with serial port and NPEs, LMB (Local Memory Bus) to access BRAM for execution of the software.

Instruction side and data side LMB buses enable to connect dual ported BRAM via separate LMB BRAM interface controllers.

Processor Local Bus (PLB) provides a high-speed interface between the processor and high-performance peripherals. Designed custom Intellectual Properties (here NPEs) are connected to MicroBlaze via PLB. 14 software accessible registers are created in top module of NPE, so inputs can be loaded and outputs can be read via these registers. In this work, parameters of NPE and initial values of x and y state variables are loaded to these registers and control signals of NPE are managed by changing corresponding bits of related register.

After connecting modules to buses, bitstream file of hardware project is generated by using ISE tools. Obtained address map of system is used to generate library and Board Support Package (BSP) files, and these files enable to control software accessible registers of NPEs.

The number of used NPEs in this design can be increased while considering the area consumption of the Field Programmable Gate Array and service time of the Microblaze processor to each NPE. It is efficient to increase the number of NPEs if the execution time of NPE is much more than the service time of the Microblaze to each NPE. Service time is defined as the time required while the processor is loading data into the NPE before it starts to run.

4. Active Wave Generation Software

Many applications can be implemented by changing only the software. When (4) is analyzed, only a sum operation or a multiplication can be calculated by loading proper parameters and initial values to the NPE by just changing the software. Coupling defined by the synaptic law (6) can be changed by adding new neighbor signals on to the state value calculated by NPE using the software flexibility. Also, pipeline stages, CNN dimensions, initial values and parameters can be easily adjusted by the software . In this work, the 9×9 cellular nonlinear network (1) is emulated and spatio-temporal waves on the network are observed in real-time.

In the sequential CNN emulator architecture, after loading initial values of the current cell on the network, the next cell value can only be calculated after the response of the NPE for the current cell. Instead of having this sequential hardware architecture, a pipeline is implemented. After initialization of one NPE with the data of a cell, without waiting the NPE's response, Microblaze passes to load next cell data to the other NPE. Number of NPE determines pipeline stage number, so in this work 3 stage pipeline structure is used. Scheduling of three NPE in the pipeline is shown in Figure 4.

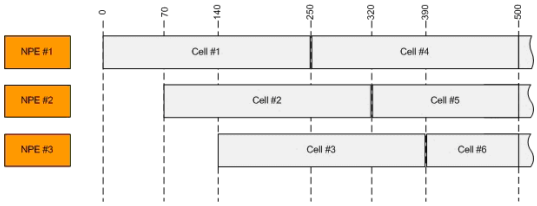

Figure 4: Scheduling of three NPE by the Microblaze controller.

There are 4 matrices that have 9×9 sizes in 2-dimension to save both current and calculated next time values of x and y state variables.

After the computing and saving obtained values to next-value-matrix for each state variable of each cell on the CNN, pointers of current and next-time matrices interchanges and so next value matrix becomes current value matrix. After the coded numbers of iteration is executed by the emulator, the calculated values are sent to a computer via serial port.

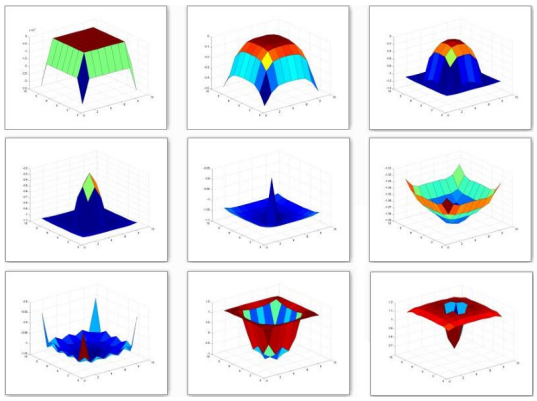

Figure 5: 9×9 network emulation.

5. Results and Conclusion

In this paper, a hardware - software co-design of the CNN is introduced and an implementation of this design is presented. When a comparison is done between this emulator design and previous pure hardware designs, it is obvious that the performance of the emulator worsens but the flexibility improves. In order to use the active wave computing based algorithms in the applications, hardware implementations of the wave computer are vital to deal with computational cost of the wave generating part of the algorithm. However full hardware implementation restricts the application specific part of the active wave computing algorithm. Currently, the application specific part of the

algorithms are implemented on the computer. Communication between computer and the wave computer hardware is another aggravating effect on performance. The proposed hardware-software co-design might help to designer to have a better performance and flexibility for their active wave computing algorithms.

Acknowledgments

This work was supported by The Scientific and Technical Research Council of Turkey, under project 105E103.

References

[1] B. Roska and F. Werblin. Vertical interactions across ten parallel, stacked representations in the mammalian retina. *Nature*, 410:583–587, 2001.

[2] F. Werblin. Synoptic connections, receptive fields, and patterns of activity in the tiger salamander retina. *Investigative Ophthalmology& Visual Science*, 32(3):459–483, 1991.

[3] Xilinx. *EDK Concepts, Tools, and Techniques*. `http://www.xilinx.com/support/documentation/dt_edk_edk9-2.htm`, 2010.

[4] M.E. Yalcin. A simple programmable autowave generator network for wave computing applications. *IEEE Transactions on Circuits and Systems II-Express Briefs*, 55(11):1173–1177, Nov 2008.

[5] R. Yeniceri and M.E. Yalcin. An implementation of 2D locally coupled relaxation oscillators on an FPGA for real-time autowave generation. *11th International Workshop on Cellular Neural Networks and their Applications, CNNA 2008*, pages 29–33, Jul 2008.

[6] R. Yeniceri and M.E. Yalcin. An emulated digital wave computer core implementation. *Proceeding of the 19th European Conference on Circuit Theory and Design, (ECCTD'09)*, Aug 2009.

[7] R. Yeniceri and M.E. Yalcin. Path planning on cellular nonlinear network using active wave computing technique. *accepted to publish in Proceedings of SPIE Europe Microtechnologies for the New Millenium Symposium*, 7365, May 2009.