A Novel Intelligent Environment Dedicated to ANN Fast Prototyping

Eduardo do Valle Simões, George Fabris Justo & Dante Augusto Barone
Universidade Federal do Rio Grande do Sul - Porto Alegre - Brazil
E-mail: EDSIM@inf.ufrgs.br

1. Introduction

The utilization of Field-Programmable Gate Arrays (FPGA) to implement Artificial Neural Networks (ANN) becomes very attractive since it allows fast hardware design and modification at low costs. This work presents a fast prototyping system for Boolean neural network designs within a proper FPGA matrix, named FLECHA. In such context, considering just the GSN Boolean model, the previous referred system enables the designer to define the structure of the neural network and the pattern to be learned. It also performs the simulation of the ANN, helping to choose a particular architecture and deliver a VHDL description, which is taken as input to a FPGA prototyping tool. This object-oriented software tool was developed according neural networks [1][2] and genetic algorithms [3] techniques. It allows the user to implement the desired ANN with the FLECHA matrix, to verify and to edit it, and thus generates the matrix configuration pattern.

This article contains a brief description of how the proposed ANN design system works, as well as a brief presentation of the employed methodology in the implementation of neural networks, and finally, the results obtained from the prototyping of some GSN neural network circuits within the FLECHA matrix.

2. Presentation of the FLECHA Matrix

The general topology of the FLECHA matrix has 40 3-input programmable logic cells with 40 available configurable I/O Pads [4-5]. The FLECHA logic cells are able to accomplish any Boolean function of up to three inputs, representing the function true table in the internal static memory cells. A multiplexer can connect an internal register to the output of the functional bloc if it is necessary to implement sequential logic.

The function of the logic cells, the configuration of the I/O Pads, and the routing of the interconnection network are defined by a configuration program stored in internal static memory cells automatically loaded at power-up. Configuration data is stored in an external EPROM memory, specified by the prototyping software tool.

2.1. Performance Considerations

The 48 pin DIL-type allowed the design of a matrix with 40 logic cells and 40 I/O Pads, performing 600 equivalent gates. Any Boolean function can be fulfilled in each logic cell, where a large number of registers are provided, and any necessary routing paths can be defined between the logic cells and the I/O Pads. The matrix operational frequency, due to its simplified internal structures, is about 145 MHz, once a single logic cell delay is less than 7.0 ns.

3. Brief Description of the GSN Model

The GSN model was proposed by Edson Filho *et al*. [6-7]. It is derived from Kan and Aleksander's PLN model [8]. The GSN differs from the PLN model because it can present on its output an indefinite state **u**. The network consists of pyramids that can be connected with different organizations, making integrated circuit (IC) design easier.

This type of network has many advantages over other neural network implementations [9][10] as follows : ***i*)** it presents a lower number of interconnections between neurons; ***ii*)** its simple structure permits asynchronous neurons design, allowing massive parallel processing, and ***iii*)** its pyramidal regular structure gives good flexibility to the designer.

The GSN neural network has three different states of operation: the *Validation*, the *Learning Modes*, performed by software, and the *Recall Mode*, executed in hardware. After the one-shot learning provided by the GSN supervised learning algorithm, consisted of the validation and learning modes, the network is ready to classify patterns previously learned. Each neuron of the network is then implemented in the FLECHA matrix, and the circuit becomes ready to work in the recall mode.

4. Tools for ANN CAD

The design environment is divided into two different steps: ANN design; and circuit implementation. The first step has five phases: ***i*)** sizing of the neural net by the user; ***ii*)**net training with the patterns delivered by the user; ***iii*)** logic mapping of each neuron of the net, which is treated as a *black-box*; ***iv*)** generation of the VHDL description of the complete GSN neural net, with all pyramids and neuron circuits already to the FPGA design system; and ***v*)** simulation of the neural net behavior. The second step consists of the following four phases: ***i)*** logic gates mapping; ***ii)*** placement of the logic cell groups; ***iii)*** routing of the logic cells and I/O pads; and ***iv)*** implemented circuit edition.

4.1. ANN Design System

4.1.1. Sizing of the Neural Net

The sizing of the neural net is made by the user who specifies it to the software: ***i*)** the number of the net inputs; ***ii*)** the number of each pyramid levels; and ***iii*)** the number of each neuron inputs. Once the user has made the above described definitions, the system builds the neural net automatically.

4.1.2. The Net Training

In this phase, the system teaches the neural network to recognize the patterns delivered by the user. The user can define the output pattern to each input pattern previously presented to the network through a file. The net training includes the validation and learning modes, which permit to identify the memory locations that can learn a new pattern presented to the net, and effectively teach the net how to acquire a new pattern.

4.1.3. Logic Mapping of the Neurons

Each neural network neuron is mapped as a *black-box*. The system excites a neuron with all possible input combinations and identifies the output corresponding to each excitation. With this procedure it is possible to construct a true table that leads to a logic description of the neuron. This logic description leads to a schematic diagram of each neuron.

4.1.4. VHDL Description

Once the logic mapping of each neuron is completed, the system delivers a VHDL description of all network to the user. This enables the prototyping of the net. The VHDL description permits a direct interaction of the ANN with commercial FPGA prototyping systems.

4.1.5. Network Simulation

The system allows the recall mode simulation with noise inclusion (the inversion of some bits) in the user proper patterns or the ones used to train the neural network. With the simulator utilization, a preliminary evaluation of the recall level is feasible, permitting a fast test and validation of possible design modifications.

4.2. Automatic Integrated Circuit Generation

4.2.1. Logic Gates Mapping

With the utilization of the presented ANN design system, a VHDL description of each neuron was provided. The neurons weights were not stored in internal memory cells, but they were direct mapped in the neuron VHDL description. This solution provides a minimization of the logic gates for just the significant connections are taken into account. It reduces the circuit flexibility since it has to be redesigned if the network has to learn a new pattern.

The next step of the neural network development with the FLECHA matrix was the programmable logic cells specification that will implement all the neurons. At the mapping phase, the input description language, containing the specification of the IC to be projected, is adapted and optimized to fit the FLECHA matrix logic cells [11].

4.2.2. Placement of the Logic Cell Groups

The next phase is the placement of the defined logic cell groups under the FLECHA array. Genetic Algorithm (GA) based techniques were studied and projected to perform input data pre-processing in order to make use of its evolution characteristics to improve the distribution of the implemented circuit functional blocks [12].

Experimental results of this technique have demonstrated that GA can be successfully applied to VLSI global placement strategy [3]. The fix FPGA internal architecture allows the employment of the same GA algorithm to optimize all IC applications.

4.2.3. Routing of the Logic Cells and I/O Pads

The routing phase makes the interconnection between logic cells and I/O Pads through pre-defined channels (interconnection lines) in the matrix. These channels are programmable and their configuration is part of the global FPGA programming. The matrix routing process is fully automatic and it is made by a standard procedure, created to reach the most complex communication necessities. Therefore, this project phase does not need any iteraction with the user. As the matrix has a fixed and regular structure, a neural network based routing algorithm [2] was successfully applied. This routing algorithm uses a Hopfield model based neural network [1], which allows a more uniform interconnection distribution and a higher utilization of the matrix capacity [2]. This neural network based algorithm deals with each interconnection individually, using the parallel processing characteristic of the neural networks.

An important characteristic of the Hopfield model is that it can only reach local minimal, i.e., the attained solution may not be the best one. In spite of that, for many practical problems it is better to have a good solution in a short period of time, instead of having the best solution within a great delay. Amazing results were obtained with the developed solution: the space occupied by the interconnections in the tested circuits was reduced more than 20%.

4.2.4. Implemented Circuit Edition

The edition software is divided into several windows. In the project window, the user can view and edit with eight blocks of five logic cells that compose the matrix or with each block at a time. In both cases, the scroll bar is used to navigate through the matrix. The project window allows the user to manually interfere in the matrix configuration, to verify internal connections (bus conflicts), to simulate the designed IC and to generate the file that will configure the FPGA [5]. The logic cells can be programmed in groups or one by one. The user can also label logic cells, I/O pads and interconnection switches to facilitate the project view. It is possible to print the entire matrix, the configuration file and a simulation report.

The software was written in Borland C++ for Windows using the Object Oriented Programming (OOP) paradigm. According to this paradigm, the windows, menus, bitmaps and dialog boxes are all interpreted as objects [14]. Each object has its own local variables that represent the object state. These objects talk with other objects and the operating system through messages [15]. In the FLECHA software tool, the project window and the cell window are objects, and as so, communicate through messages. Therefore, the object behavior is the way with which it answers the system or other objects messages. The OOP paradigm makes program modification more reliable [16]. Each object in the software can be easily modified without changing the entire program. The OOP paradigm was chosen because of its facility in: ***i)*** communication with the graphic environment; ***ii)*** software reuse; and ***iii)*** program modifiability. Presenting these characteristics, the software tool can easily be adapted to include new FPGA families once they are developed in the future.

5. Practical Results

The same data-set of many GSN circuits were developed in both the FLECHA System and the ALTERA Commercial System [13], to provide performance comparisons. The utilization of the ALTERA EPLDs (Erasable Programmable Logic Devices) design tool (MAX+PLUSII) also permits that each neuron is directly introduced by its VHDL description. With the previous refereed data, MAX+PLUSII performs the global logic synthesis, the partitioning, the timing analysis and chooses the most adequate EPLD to project each application.

The ALTERA logic synthesis and choice of the appropriate circuit average speed lasted 25 seconds, in a 486DX33 MHz platform. The maximum propagation delay in some circuits has reached the 42ns mark, representing then a high performance to the ALTERA implemented network.

The FLECHA solutions differ from the ALTERA ones, since the FLECHA logic cells have individual low capacity leading to the necessity of grouping many cells to treat each output. Nevertheless, the faster ALTERA implementations have presented a maximum delay of 30ns in the FLECHA matrix.

The results obtained with the two systems have showed high recognition average rates: 95%, with the logic inversion of one bit in the input pattern, and 80%, with two bits presenting logic

inversion. In these cases, 100 patterns were used in the training phase. The recognition rates and performances show that this kind of network is suitable to real-time image recognition applications.

6. Conclusions

The programmable FPGA structures allow a network optimization linked directly to the learning patterns, reducing its generality, its design costs, and increasing its performance. The complete re-design of the circuit can be made in a short time delay, enabling the circuit to be tested and validated under different conditions imposed to the hardware system.

In the logic synthesis phase, the network digital circuit suffered a drastic minimization. Many inputs are irrelevant to calculate the pyramid outputs, in specific applications, and they were disregarded by the system. This represents a high redundancy characteristic of the GSN based ANN.

This software tool has a very good perfomance in the rapid circuit prototyping using the FLECHA matrix. It was developed with a modern programming strategy and a powerful graphic interface which allows the user to quickly project integrated circuits very easily.

The design of a direct interface between the GSN software simulator and the FLECHA prototyping tool, using the VHDL, has permitted a significant reduction in the network implementation time. The modification of the output data format delivered by the GSN simulator to the VHDL language permits that this prototyping tool recognizes directly the constitution of the network neurons without the necessity of the user interventions.

7. References

[1] Hopfield, J.J., Tank, D.W. **"Neural" Computing of Decisions Optimization Problems**. In: Biol. Cyber 1985. v. 52, p. 141-152.

[2] Shih, P.H.; Chang, K.E.; Feng, W.S. Neural Computation Network For Global Routing. **Computed-Aided Design**, Surrey, Inglaterra, Oct. 1990.

[3] Hwee, G.B.; Hiot, L.M..: **Genetic Algorithm for VLSI Floorplan Design Problem**, 5th International Symposium on IC Technology, Systems & Applications, Proceedings, Hyatt Regency, Singapore, Setember 1993, pp. 475-479.

[4] Simões, E.V., Uebel, L.F., Barone, D.A.C. **Fast Prototyping of Artificial Neural Networks: GSN Digital Implementation.** IV International Conference on Microelectronics for Neural Networks and Fuzzy Systems, Torino-Itália, setembro de 1994.

[5] Simões, EV. **FLECHA-A FPGA Directed to Integrated Circuit Fast Prototyping**, Msc thesis, Federal University of Rio Grande do Sul, Brazil, may, 1994.

[6] E. C. D. B. Filho, D. L. Bisset and M. C. Fairhurst, **A Goal Seeking Neural for Boolean Neural Networks**, Proc. International Neural Network Conference, Paris, France, Vol. 2, July 1990, pp. 894-897.

[7] E. C. D. B. Filho, M. C. Fairhurst and D. L. Bisset, **Analysis of Saturation Problem in RAM-Based Neural Network**, Electronics Letters, Vol. 28, No. 4, February 1992, pp. 345-346.

[8] I. Aleksander and W. K. Kan, **A Probabilistic Logic Neuron Network for Associative Learning**, Proc. IEEE First Intl. Conf. on Neural Networks, San Diego, California, Vol. II, June 1987, pp. 541-548.

[9] Luís Felipe Uebel and Dante Barone, **DIANNE: A GSN Neural Network VLSI Implementation**, 5th International Symposium on IC Technology, Systems & Applications, 1993, pp. 678-682.

[10] Luís Felipe Uebel and Dante Barone, **Implementation of a GSN Neural Network on Integrated Circuit**, ISIC'93 - International Symposium on IC Techinology, Systems & Applications, 5., 1993. Singapore. **Proceedings...** Singapore, 1993. pp. 459-468.

[11] Chen, Y.; Tsai, W.K.; Kurdahi, F.J. A Logic Synthesis System Based on Global Dynamic Extraction and Flexible Cost. In: ISIC'93 - International Symposium on IC Techinology, Systems & Applications, 5., 1993. Singapore. **Proceedings...** Singapore, 1993. p.299.

[12] Goldemberg, D.E..: Genetic Algorithms in Search, Optimization, and Machine Learnig, Addison-Wesley publishing company, Inc., pp. 1-214, 1989.

[13] Altera Corporation, **MAX+PLUS II - Getting Started**, 1992, 137 p.

[14] Murray, W.H. and Pappas, C.H. *Windows Programming: An Introduction*, Osborne McGraw-Hill, 1990

[15] Dershem, H., and Jipping, M.J. *Programming Languages: Structures and Models*, Wadsworth Publishing Company, California, 1990.

[16] Mullin, M. *Object Oriented Program Design*, Addison-Wesley, 1989.