# A Comparison Between 3 Different GSN Model Hardware Implementations with the Appliance of an ANN Fast Prototyping System

Eduardo do Valle Simões, Luís Felipe Uebel & Dante Augusto Barone
Universidade Federal do Rio Grande do Sul - Porto Alegre - Brazil
E-mail: EDSIM, UEBEL, BARONE@inf.ufrgs.br

## Abstract

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 comparison between two implementation strategies of ANN hardware design: the VLSI-Full Custom approach and FPGA. For that reason, three alternatives will be described. The first one is the DIANNE integrated circuit [7-8]. It is a full digital-asynchronous implementation of the GSN Boolean model, proposed by Edson Filho *et al*. [6]. The GSN differs from the PLN model because it can present on its output an indefinite state U. The GSN network consists of pyramids that can be connected with different organizations, making integrated circuit (IC) design easier. It presents a lower number of interconnections between neurons; its simple structure permits asynchronous neurons design, allowing massive parallel processing; and its regular pyramidal structure gives good flexibility to the designer.

The DIANNE circuit consists of a fixed architecture, presenting 4 GSN pyramids with 60 neurons at all. The learning mode is not included in DIANNE because it occupies a large silicon area. Therefore, this circuit can only operate in the recall mode. After the software tool has trained a neuron net, the weights are stored in internal RAM memories that configure the chip to recognize the trained patterns. The relationship between area, performance and number of neurons presents a good solution that improves system speed (full software implementation) more than 300%.

The second alternative presented in this article is a fast prototyping system for Boolean neural network design within a proper FPGA matrix, named FLECHA. In such context, considering just the GSN Boolean model, the software 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, that will program the FLECHA matrix to work according the trained neural network. The same VHDL description of the neural net trained neurons can be implemented by some commercial FPGA design tools, such as the ALTERA MAXII+PLUS [10]. This is the third alternative presented in this paper.

This article contains a brief description of how the proposed ANN design system works, as well as a brief presentation of the DIANNE circuit and the FLECHA matrix, and finally, the results obtained from the prototyping of some GSN neural network applications within the DIANNE circuit, the FLECHA matrix and the ALTERA commercial EPLD.

# 1. Brief Presentation of the DIANNE Circuit

The DIANNE circuit is a CMOS Full Custom ASIC, implemented with ES2 1.2μm technology. The DIANNE 40-pin chip has 3100μm x 2800μm with 17,000 transistors and it is capable to perform up to 600 million synapses by second, considering a critical path of 200ns. The chip has 16 input pins and 8 output pins. Therefore each implemented ANN is able to recognize an input pattern grid of 4 x 4 pixels. Each pyramid output request 2 Pads because it is necessary to use 2 bits to represent the three possible logic values 0, 1, U. The indefinite value U is a proper characteristic of the GSN model and needs to be implemented by hardware.

The DIANNE has a fixed architecture of just 4 GSN pyramids with 60 neurons and sometimes it may cause problems if it is necessary to implement specific applications. The FPGA utilization is more versatile and allows specific implementations with different number of pyramids and neurons. This is an excellent solution for many practical cases.

# 2. Brief 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 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.

The 48 pin DIL-type package allowed the design of an array with 40 logic cells and 40 I/O Pads, performing 1000 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 6.9 ns.

# 3. A Software System Dedicated to ANN Fast Prototyping

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, 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 trained; 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.

The first phase of the ANN Design System is the sizing of the neural net. It is made by the user who specifies: ***i*)** the number of the net inputs; ***ii*)** the number of each pyramid level; and ***iii*)** the number of each neuron input. Once the user has made the above described definitions, the system builds the neural net automatically.

In the training 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.

The system also 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.

With the utilization of the presented ANN design system, a VHDL description of each neuron was provided. If the DIANNE solution was chosen, the neurons weights are directed stored in the DIANNE internal memory cells. Thus, the DIANNE circuit is ready to operate. If the FLECHA matrix was chosen, the weights have to be mapped into the neurons VHDL description. Therefore, the FLECHA solution provides a minimization of the logic gates and 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.

Each neural network neuron is then 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.

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.

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

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 [9]. 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.

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 interaction 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. Amazing results were obtained with the developed solution: the space occupied by the interconnections in the tested circuits was reduced more than 20%.

The edition software is divided into several windows. In the project window, the user can view and edit the eight blocks of five logic cells that compose the matrix. Scroll bars are 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 [11]. Each object has its own local variables that represent the object state. These objects talk with other objects and the operating system through messages [12]. 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. 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.

# 4. Practical Results

The same data-set of many GSN circuits were developed in the DIANNE circuit, the FLECHA matrix and the ALTERA commercial EPLDs [10], to provide performance comparisons. The utilization of FLECHA and ALTERA design tool also permits that each neuron is directly introduced by its VHDL description.

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 (presenting a delay of 40ns) have shown an average delay of 36.8ns in the FLECHA matrix. Table 1 presents some of the results obtained with the implementation of many trained GSN ANN with the DIANNE circuit, the FLECHA matrix and the ALTERA EPM5032. The table shows chip speed related to critical path delay of two GSN architectures trained with different saturation degrees.

The results obtained with the DIANNE circuit and the FPGA systems have shown high recognition average rates: 97%, with the logic inversion of one bit in the input pattern, and 93%, with two bits presenting logic inversion. In these cases, 1000 patterns were used in the training phase. The recognition rates and performances show that these three kinds of ANN hardware are suitable to real-time image recognition applications.

**Table 1 - Comparison between 3 ANN Hardware Implementations**

| Examples | DIANNE | FLECHA | ALTERA |
|---|---|---|---|
| | 5MHz / 200ns | 28.98MHz / 34.5ns | 25MHz / 40ns |
| 4 Pyramids | 5MHz / 200ns | 36.23MHz / 27.6ns | 25MHz / 40ns |
| 60 Neurons | 5MHz / 200ns | 20.70MHz / 48.3ns | 25MHz / 40ns |
| 4 x 4 pixel Grid | 5MHz / 200ns | 18.11MHz / 55.2ns | 14.28MHz / 70ns |
| | 5MHz / 200ns | 16.10MHz / 62.1ns | 14.28MHz / 70ns |
| 6 Pyramids | ---- | 14.49MHz / 69ns | 14.28MHz / 70ns |
| 240 Neurons | ---- | 13.17MHz / 75.9ns | 10.52MHz / 95ns |
| 5 x 5 pixel Grid | ---- | 12.07MHz / 82.8ns | 10.52MHz / 95ns |

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

The FLECHA design tool has shown good performance in rapid circuit prototyping. It was developed with a modern programming strategy and a powerful graphic interface that allows the user to quickly project and edit integrated circuits very easily.

The design of a direct VHDL interface between the GSN software simulator and the FLECHA or other commercial FPGA prototyping tool has permitted a significant reduction in the network implementation time. The modification of the output data format to the VHDL language permits that many commercial systems recognize directly the constitution of the network neurons without the necessity of the user interventions.

# 6. 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; Barone, D.A.C. **Design of a Novel FPGA Matrix: Internal Routing and Control Device**. Primer Workshop Iberchip, Cartagena de Indias, Colombia, Feb., 1995.

[6] 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.

[7] 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.

[8] 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.

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

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

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

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