

A Networked FPGA-Based Hardware Implementation of a Neural Network Application

Héctor Fabio RESTREPO, Ralph HOFFMANN, Andres PEREZ-URIBE,
Christof TEUSCHER, and Eduardo SANCHEZ

Logic Systems Laboratory, Swiss Federal Institute of Technology
CH – 1015 Lausanne, Switzerland
E-mail: {name.surname}@epfl.ch

Abstract

This paper describes a networked FPGA-based implementation of the FAST (Flexible Adaptable-Size Topology) architecture, a Artificial Neural Network (ANN) that dynamically adapts its size. Most ANN models base their ability to adapt to problems on changing the strength of the interconnections between computational elements according to a given learning algorithm. However, constrained interconnection structures may limit such ability. Field programmable hardware devices are very well adapted for the implementation of ANN with in-circuit structure adaptation. To realize this implementation we used a network of Labomat 3 boards (a reconfigurable platform developed in our laboratory), which communicate with each other using TCP/IP or a faster, direct hardware connection.

1 Introduction

Artificial neural network models offer an attractive paradigm: learning to solve problems from examples. They achieve fast parallel processing via massively parallel non-linear computational elements. While software implementations on conventional von Neumann machines are very useful for investigating the capabilities of neural network models, a hardware implementation is essential to fully profit from their inherent parallelism, as well as for real-time processing in real-world problems. Well-known ANN, such as multi-layer Perceptrons [1], have proved capable of solving various problems, but because of their relatively high complexity, they are not well suited for a hardware implementation. Thus, we chose the FAST (**F**lexible **A**daptable-**S**ize **T**opology) algorithm [2], which is very well suited for such an implementation. This algorithm has been implemented on a Labomat 3 board, developed in our laboratory as a teaching and research tool. It is a flexible board, providing FPGA technology and a microcontroller with an Ethernet interface.

2 The FAST Algorithm

The **FAST** neural network [2, 3] is an unsupervised learning network with **F**lexible **A**daptable-**S**ize **T**opology. The network is feed-forward, fully connected, and consists of two layers: an input layer and an output layer. Essentially, the aim of the network is to cluster or categorize the input data. Clusters must be determined by the network itself based on correlations of the inputs (unsupervised learning). The network's size increases by adding a new neuron to the output layer when a *sufficiently distinct* input vector is encountered and decreases by deleting an operational neuron through the application of probabilistic deactivation.

3 Labomat 3 Description

The Labomat 3 [4] board uses FPGA technology, and is therefore a reconfigurable platform. Developed by our laboratory, the features of this board makes it a unique teaching and research tool. Our aim is to offer a powerful, all-round platform that is easy to understand and simple to use.

4 FAST Implementation on Labomat 3

The FAST neural network architecture was implemented using a network of Labomat 3 boards and was completely described in generic VHDL. The system is completely scalable because we can add to the network as many Labomat 3 boards as we want. The user communicates with and configures the system via the telnet protocol.

5 Results

The final implementation consists of a network of up to eight Labomat 3 boards. Each board contains two FAST neurons supporting 8-bit computation and two-dimensional vectors.

The maximal working frequency for this design was approximately 10 MHz. The same implementation using an FPGA speed grade of -1 (XC4013E-1) would allow a

working frequency of 16 MHz. The working frequency of this implementation could be increased by implementing only a single neuron into the FPGA circuit: we would then have enough place to implement more adders and multipliers working in parallel rather than sequentially, as is the case in our current implementation. Of course, this approach requires sixteen Labomat 3 boards instead of eight. Thanks to the scalability characteristics of our implementation, however, this increase does not pose any particular problem, and we are currently developing this approach.

The average number of clock cycles needed to process one vector is approximately 64, which translates to a processing time of $6.4\mu s$ per vector for a 10 MHz clock. Unfortunately, the whole process is software-driven (each hardware subprocess is started by the processor, based on the last subprocess results), which introduces an important software overhead.

For the application tests, we presented to the neural network a set of 10000 two-dimensional vectors randomly chosen from a database composed of 40000 vectors. Several tests have been made on the following platforms (Table 1 summarizes the obtained results):

1. eight Labomat 3 boards using TCP/IP;
2. eight Labomat 3 boards using direct hardware communication;
3. software simulation on the 68360 processor available on the board and a Sun SPARC Ultra 1 workstation.

The first observation are the poor results of the TCP/IP-based configuration. This is due to the way we use TCP: there is only one byte sent or received per TCP packet.

# of neurons	TCP/IP	Hardware	68360	SPARC Ultra
4	197ms	0.650ms	$60\mu s$	$2.10\mu s$
8	575ms	1.470ms	$80\mu s$	$2.80\mu s$
12	955ms	2.250ms	$100\mu s$	$3.51\mu s$
16	1362ms	3.040ms	$120\mu s$	$4.21\mu s$

Table 1: Hardware-based

By using the direct hardware connections between the boards, we obtain better results, but the software overhead is very important ($290\mu s$ for a communicationless, single board configuration). To understand the main differences between the hardware processing time (about $6.4\mu s$) and the final processing time, we measured the time spent in function calls using a C cross-compiler, and obtained a value of $1.5\mu s$ (time to call, then return from a function). The software stack involves an important amount of function calls, which is not the case for the software simulation.

The best results were obtained with software simulations on powerful but widely available platforms.

6 Conclusion

We have used a network of Labomat 3 boards to successfully realize an adaptable-size neural network architecture, which was completely implemented in generic VHDL. Our initial experiments with this recently completed system indicate that high performance can be achieved on pattern clustering tasks.

As an example, we have applied our FAST ANN to an image segmentation and recognition problem. It has been found that the results obtained with this methodology closely resemble those obtained with other neural algorithms which are not suitable for hardware implementation.

At the beginning of our implementation, a hardware driver featuring network capabilities was developed, unfortunately this driver was quite bulky and complex, requiring fully half of the resources of the Xilinx XC4013 device. For this reason we used the software device driver instead, leading to very disappointing results.

The working frequency and the speed of our implementation could be increased by implementing a single neuron into the FPGA circuit: we would then have enough place to implement more adders and multipliers working in parallel rather than sequentially, as well as the aforementioned hardware-driven communication process.

The hardware approach would become competitive with a completely hardware-driven process (which is not the case in our current implementation) and with more neurons per chip, implying a greater amount of parallelism, an approach that may be possible with larger FPGAs like the new Xilinx Virtex family, or even better, ASIC technology.

References

- [1] P. Liang N.K. Bose. *Neural Network Fundamentals with graphs, Algorithms, and Applications*. Electrical and Computer Engineering Series. McGraw-Hill, Inc., 1996.
- [2] A. Pérez-Urbe. *Structure-Adaptable Digital Neural Networks*. PhD thesis, Swiss Federal Institute of Technology-Lausanne, Lausanne, EPFL, 1999. (to appear).
- [3] A. Pérez-Urbe and E. Sanchez. FPGA Implementation of an Adaptable-Size Neural Network. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN96)*, volume 1112 of *Lecture Notes in Computer Science*, pages 383–388. Springer-Verlag, Heidelberg, 1996.
- [4] C. Teuscher, J.-O. Haenni, F. J. Gomez, H. F. Restrepo, and E. Sanchez. A Tool for Teaching and Research on Computer Architecture and Reconfigurable Systems. In *Proceedings of the 25th Euromicro Conference*, volume 1, pages 343–350. IEEE Computer Society, Milan, Italy, September 8-10 1999.