LABORATOIRE DE MICROÉLECTRONIQUE

Louvain-la-Neuve

UCL
Université catholique de Louvain

VLSI implementations of artificial neural networks

Michel Verleysen

Thèse annexe présentée en vue de l'obtention du grade d'agrégé de l'enseignement supérieur.

décembre 2000

Contents

VLSI implementations of artificial neural networks

Introduction

Artificial neural networks (ANN) are parallel algorithms. Their inherent parallelism makes them particularly suited to parallel VLSI implementations, i.e. the development of dedicated circuits or architectures that are able to perform many operations in parallel. As ANN involve similar operations to be performed in parallel, it is particularly easy to develop dedicated parallel architectures: most of them are based on the repetition of identical devices, each of them performing one of the operations.

Parallel architectures for ANN are justified by the fact that most ANN learning algorithms involve a high computational load. Who never waited days (of computing time) for the learning of a Multi-Layer Perceptron does not know what neural networks are! Since the early beginning of the neural networks research, one has naturally been tempted to develop specialised machines, in order to lighten the computational load.

Another nice feature of ANN is that most algorithms mainly involve simple operations. Indeed, those algorithms with biological inspiration naturally use the same type of operations as found in biological cells. For example, early algorithms based on McCullogh and Pitts model of the neuron mostly use sum-of-products. Of course, there are exceptions: it is not because pseudo-inverse of matrices may be found in "neural" algorithms that we could expect our brain to perform large matrix inversions…

Inherent parallelism, simplicity of operations and high computational load are the reasons why VLSI implementations of ANN became so popular. Since the mid 80s, many attempts have bee made towards the realisation of VLSI circuits, or larger machines, suited to the simulation of ANN. The development of VLSI circuits implementing large numbers of operations in parallel is interesting on the point of view of the circuit designer: as large number of cells have to be integrated, one must carefully optimise them. Furthermore, connectivity, transmission of information, and maximisation of the use of resources are interesting problems generated by such circuits. But are parallel implementations really needed for the user of neural technology? Are computational power requirements still a challenge? Are there other reasons to develop specific architectures? The following paragraphs try to answer to these questions, and emphasise on the rapid evolution of this research area.

Analog versus digital

Since the early developments of specific VLSI implementations of ANN, there has been a long debate concerning the respective advantages and drawbacks of analog and/or digital circuits. We do not intend to enter here this debate. Everybody now acknowledges that both issues are of interest, in different situations though. We limit ourselves here to a rough list of advantages of both issues, to set the basis for the following discussion.

Analog implementations rely on the fact that ANN involve simple operations. As many of these operations are non-linear, simple analog cells using a few transistors only are able to implement them efficiently. The required precision of ANN algorithms is usually limited and compatible with analog computations. As simple cells can be developed, integrating a large number of these cells in parallel is possible, leading to a high number of operations per second (MIPS). Moreover, real-world signals being analog, these implementations are able to process the incoming signals directly, without the need for convertors, filters, etc.

Digital implementations do not permit parallelism to the same extent. While it is typical to find hundreds or thousands of analog cells working in parallel on a chip, digital cells, being much complex hence much larger, are limited in number to a few tens usually. Nevertheless, digital cells are more flexible, programmable, and thus may be developed for a wider range of applications. Concerning the speed of operations, there is no general rule differentiating analog and digital ones. Nevertheless, reminding that the number of MIPS is modulated by the degree of parallelism, digital implementations are less advantageous on this point.

Evolution of the field - historical facts

Following the evolution of the specific domain of neural network hardware is rather easy. Unlike may other branches of the neural network field, there exists one main conference (MicroNeuro) dealing with this topic. Other papers may of course be found in most neural network conferences and journals, but the vast majority of essential works are presented at least to MicroNeuro.

Looking to the evolution of the MicroNeuro conference is thus informative. In 1990 (first conference), most presentations concern ideas on how parallel implementations could be realized. Some circuits are presented, but never concern an application realized to the end. In fact, almost none of the papers speaks about applications!

In the next conferences, some emphasis is put on an interesting aspect of the field. It becomes clear that despite the adequacy of ANN to parallel implementations, some restrictions exist concerning the precision required in the algorithms. On the point of view of the circuits, analog ones are inherently limited in precision, and digital ones require a compromise between precision and complexity (hence size). Interesting works then concern the development of algorithmic methods that are designed to cope with the restrictions of hardware implementations: at equal performances, one prefers algorithms with a lower need for precision.

In the early 90s, most papers pretend to increase the speed of processing for neural network algorithms, but few quantify the gain with respect to a pure sequential simulation on a conventional computer! There is a balance between the number of papers dealing with analog and digital implementations; pulse-stream based computations are also used. Some exotic papers can also be found (quantum devices for ANN, etc.). The first papers about neurocomputers may be found (by neurocomputers, we mean complete boards or computers where the core implements a parallel neural algorithm, but that also deals with the handling of inputs and outputs, the programmability issues, etc.; most neurocomputers are based on digital circuits).

In the mid 90s, one may find the first papers dealing with applications of the implementations. Most of the applications concern computer vision (velocity sensors, motion detection, filtering, scene segmentation, etc.). The parallelism of visions systems coupled with the possibilities of ANN in this domain makes them the first candidate for parallel implementations.

After the mid 90s, other specific applications appear, as very fast detection systems in high-energy physics experiments. What is interesting to mention is that, as the developments concern specific applications rather than generic solutions, analog circuits are more and more used. This is in accordance with the fact that digital implementations are more flexible, thus more generic, while analog ones may be designed to be more powerful in specific cases.

Another interesting fact is that acceleration is no more the first objective. In many applications, the portability of the system implemented is the main goal, making the device more *adapted* to the application, even when it is not more *powerful* than a simulation on a standard PC.

Despite this evolution towards more specific applications, there is no doubt that a majority of developments in this domain are still motivated by the challenge of the design, rather than by the expected use of the device.

Need for parallel architectures

Are parallel architectures really needed to implement neural networks? This is the important question; we will try to give some answer in the next paragraphs.

Even for inherently parallel algorithms, it must be clear that parallel implementations are not an advantage in itself. What can be an advantage is the resulting increase in performances generated by the use of parallel algorithms, or the portability, or the possibility to handle input signals without conversion and filtering, etc. The advantages of parallelism are subject to the possibility of efficiently using the computing cells working in parallel; but many typical neural network, like associative memories, Multi-Layer Perceptrons, Radial-Basis Function Networks, Self-Organizing Maps, etc., fit this condition.

"Chip-in-a-loop" is sometimes used in parallel implementations of neural networks. "Chip-in-a-loop" means that the parameters of the neural network are adapted through a learning realized on the chip, with the same components that will calculate the outputs of the network in a subsequent use. The advantage of the "chip-in-a-loop" feature is that possible imperfections of components or deviations of their nominal characteristics may be compensated to some extent. "Chip-in-a-loop" may thus be nicely used with analog circuits, the inherent precision of analog components being usually limited. Again, "chip-in-a-loop" is not an advantage in itself, but its consequences on the performances of the computation may be attractive.

While we already mentioned other possible advantages of parallel implementations, speed increase is usually cited as the main one. Nevertheless, as stated in the previous section, very few authors take the risk of comparing the speed of simulation obtained on a dedicated architecture, with respect to the speed of the same simulation on a conventional computer. The reason is simple: the technology of standard computers evolves so fast that any comparison would be obsolete within a few months. The drama is here. Any dedicated implementation takes months or years to develop, and makes use of a technology that is in general one step behind the current technology used for standard processors; volumes of fabrication are the reason for the use of "old" technologies for dedicated circuits (note that "old" usually means a few months or one year…).

Also programmable components may be an interesting alternative to dedicated circuits. Powerful signal processors now exist that may be used to simulate neural networks; programmable FPGA and other similar components are possible alternatives too. There are two strong arguments in favour of these programmable components. First, they are cheap, compared to the huge costs of dedicated electronic realisations (in particular of integrated circuits) and immediately available (they do not need long fabrication delays). Secondly, these commercial chips are designed with all facilities concerning their inputs and outputs. As already mentioned, inputs-outputs are often neglected in research prototypes of integrated circuits, leading sometimes to tremendous difficulties to use a powerful device!

Speed can remain an attractive feature in so-called *neurocomputers*, i.e. complete systems (usually extension boards or full computers) that integrate a powerful computing device, but also handle all issues concerning input-outputs and programming. Nevertheless, it appears that such systems have few applications. Many such neurocomputers have been developed in the early and mid 90s by commercial companies. Most of the big manufacturers in computing equipment (Intel, Siemens, etc.) developed original solutions at large cost. Without generalizing, one can say that most of these research and development programmes are abandoned, due to the lack of commercial feedback. Again, the tremendous increase in performances of conventional computers does not help.

But if speed of parallel implementations is no more an advantage in most applications, then what is looked for? The answer is undoubtedly portability. Many industrial applications require (or prefer) a single component to perform a specific task, rather than a whole computer (large, heavy, difficult to maintain without crash, etc.). Simplicity of use is certainly a keyword that severely contrasts with complex softwares installed on generic computers. There is thus a need for simple, small and easy-to-use components, performing specific tasks, in our case the simulation of artificial neural networks. To give an example, one of the first applications we had the opportunity to come across is the development of small, stand-alone meteorological stations, disseminated in the Swiss mountains. Meteorological prediction in Switzerland is crucial. Moreover, the landscape makes that many measurement points are necessary; it is impossible to have all of them complex and attended. Stand-alone automatic systems were then developed, including circuits implementing neural networks for the processing of the data.

New trends in neural network implementations

Beside a few exceptions (parallel coprocessors and stand-alone programmable chips), most of the current attempts to design and fabricate hardware for neural networks concern application-oriented systems. The main reason for this is that speed increase is no more the primary goal of implementations, but rather the compactness, portability, and ability to process signals without the excessive use of interfaces.

For these reasons, the tendency is to integrate as much as possible the computing devices with analog circuits. Although analog circuits have many advantages over digital ones in the context of neural network implementations, there is no doubt that programming and reconfiguration according to the needs of applications are much more difficult than with digital circuits. Analog implementations are thus naturally closer from the applications, and therefore application-dependent.

On the other side, more and more neural network techniques concern some kind of signal processing. An example of this trend is the study of Independent Component Analysis (ICA). Signal processing applications are more favourable to implementations, because of the nature itself of the data to process. Currently developed applications of signal processing by neural networks concern electronic noses, spectrometers, or more traditionally speech processing.

What is clear is that the old-fashioned concept of "developing a nice implementation, and then look for possible use" is over. We strongly advocate the development of application-specific implementations, in order to reach portable, stand-alone easy-to-use systems. The developments have of course to follow the needs of the applications. Naturally, such developments are long and costly, and thus restricted to specific either large-scale either with high added value applications.

The reason for this tendency is the "full integration" concept. The interest of developing specific circuits that must still be surrounded by a lot of annex devices is low. An interesting concept is for example to integrate a full electronic nose into a single circuit (or into a few ones), including the sensors, the processing of the signals, etc. Such fully integrated systems with sensors are commonly called *smart sensors*. Smart sensors integrate signal sensing, handling (amplification, filtering) and processing (classification, source separation, etc.). As an example, some Independent Component Analysis algorithms are well suited to analog implementations. We are convinced that this is one of the future possible trends in VLSI implementations of neural networks.

Classification algorithms at the output of sensor systems are another possible trend. This could be applied for example to spectrometers, optical recognition systems, speech analyzers, etc.

To summarize our view, we are convinced that the next few years will confirm an important evolution in the field of neural network implementations. The future is to application-dependent systems, making it possible to integrate the whole application on one or few components; for most applications, in particular those related to sensors and signal processing, analog solutions seem to have determining advantages. This is why we think that the domain of VLSI implementations of neural networks will slowly break up, leading to a situation where the possibility to implement algorithms into silicon will be viewed as a tool rather than a discipline. Rather than a negative view, we think that taking more into account the needs of real applications is advantageous for a prosperous future of this fascinating research area.