Artificial Neural Networks
T. Kohonen, K. Mäkisara, O. Simula and J. Kangas (Editors)

STOCHASTIC COMPUTATIONS IN VLSI ANALOG NEURAL NETWORKS

Michel Verleysen and Paul Jespers
Université Catholique de Louvain - Microelectronics Laboratory
3, pl. du Levant
Louvain-la-Neuve, Belgium

In some applications of neural networks (associative memories, classifiers,...), the size of the device is very important. Its performances are directly related to the number of synapses or neurons which can be used in a single network. When the network is implemented on a VLSI chip, the number of cells depends on their size and on their precision. A compromise between the precision of digital implementations and the density of analog ones can be obtained by using stochastic operations in the network. This paper shows how to use stochastic computations in sum-of-products, and their applications to neural networks.

1. INTRODUCTION

Neural networks can be used in applications where the processing rules are not clearly defined, such as recognition tasks (speech, image,...) or optimization problems. Some particular networks require only a small number of synapses and/or neurons. An example is the separation of uncorrelated sources [1], which can be performed by a two-neurons network for a two-sources problem. But in most applications of neural networks, the number of neurons and synapses is important for the performances of the device. For example, the bad capacity of an Hopfield associative memory can be compensated by the number of cells used in the network; if more cells, and thus more neurons, are available in the network, the way of how the information is represented in terms of neuron states can be defined so that for the same amount of data a certain redundancy can be obtained. It is obvious that this redundancy will help the learning algorithm to find appropriate weights, and will also help the network to converge to the desired stable state; we can hope that the storage capacity will be increased by this way. A crucial point is thus to reduce the size of neurons and synapses when the networks are implemented on a VLSI chip.

Digital processors, or neurons in the case of neural networks, can easily be designed; the required precision, which depends on the algorithm and the application, can be reached with as many bits as necessary. The size of each cell

* Part of this research has been carried out under the EEC ESPRIT-BRA project n°3049 NERVES ("Innovative Architectures for Neurocomputing Machines and VLSI Neural Networks").

(neuron or synapse), however, grows with this precision and can quickly become too big to consider chips with hundreds of neurons. In digital neural networks, some operations are thus generally multiplexed; the disadvantage of this method is that the computation times are increased because of this multiplexing. For example, systolic arrays can be used to implement neural network architectures. F.Blayo [2] showed that all methods for multiplexing and circulation of data in a systolic neural net can be unified to a "general systolic ring" model, in which the number of steps required for a complete matrix product is independent of the chosen architecture. This number of steps obviously depends on the size of the network, but is in all cases much larger than the number of computation steps in an analog architecture, which can be reduced to one!

Analog neurons are then preferred in most cases when speed and size are simultaneously required. The precision that can be obtained with analog computations is however limited by the precision of the components used in the chip [3]. Some matching techniques exist to overcome this limit [4], but such methods require in general much larger areas on silicium. For example, two current sources can be quite perfectly matched (less than 1% error) if four transistors are designed on the four corners of a square, and if opposite transistors are connected together; however, the silicium area occupied by these four transistors can be as much as 10 times the area occupied by two transistors of a doubled size! Moreover, it is extremely difficult to match more than two devices...

Mixed solutions are then considered, to decrease the influence of these errors [5,6]. In such implementations, one- or two-bits digital products are digitally performed, while analog sums are realized by summing currents in order to decrease the number of connections between synapses and neurons (fig.1).

The main problem of this architecture is that in most networks analog products must be realized in the synapses instead of digital products. In most cases indeed, the learning algorithms result in analog values for the synaptic weights. Analog products are then performed, the results are summed and the activation function performed on the sum results also in an analog value. The digital input-outputs used in the last mentioned architecture, together with the two-bit products implemented, restrict the use of this circuit to specific algorithms.

The above mentioned problems encountered when replacing digital values by analog ones may lead to the conclusion that this architecture is not suited for networks when analog products are needed. However, the idea of analog sums (fig.1) is quite interesting since it considerably reduces the required wirings between synapses and neurons. A good solution would thus be to keep these analog sums but with multipliers which could cope with analog values.

In order to consider only digital values on the gates of the current sources (fig.1) even with analog products in the synapses, stochastic computations can be used [7]. The advantage of this method is that a greater precision than with analog products can be obtained, while the disadvantage is the increased computation times.

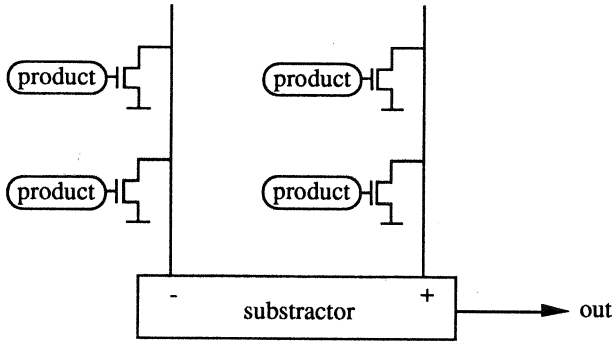

FIGURE 1
Analog sum-of-products with current sources

2. STOCHASTIC PRODUCTS

The main idea of stochastic products is to replace the analog signals at the input of the multiplier by digital signals whose probability to be "high" is proportional to the ratio between the analog value and the full-scale voltage [8]. These digital signals are generated by comparison between the analog value and a time-dependant function whose probability density is constant over the whole voltage dynamics; examples of such functions are triangular signals with constant slope or white noise (fig.2).

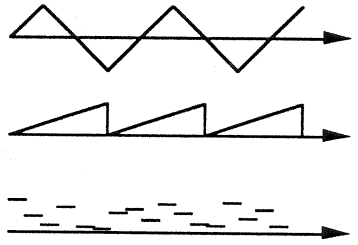

FIGURE 2
Examples of functions used in stochastic products

The two time-dependant signals can then be digitally multiplied by an AND gate. Integration of the result is finally performed to obtain an analog value which is the product of the two analog inputs (fig.3). The only restriction on the two time-dependant input signals (triangular, random,...) is that they must be uncorrelated. This can be proved by the following:

If: in_1 and in_2 are the two analog values to multiply, each of them being included in the interval [0,Vmax]
$r_1(t)$ and $r_2(t)$ are the two time-dependant functions with constant probability in the interval [0,Vmax]

$$p(r_1(t) \le V) = p(r_2(t) \le V) = \frac{V}{V_{max}}$$

Thus

$$p(r_1(t) \le in_1) = \frac{in_1}{V_{max}} \quad \text{and} \quad p(r_2(t) \le in_2) = \frac{in_2}{V_{max}}$$

$$p[r_1(t) \le in_1 \ \& \ r_2(t) \le in_2] = \frac{in_1 \cdot in_2}{V_{max}^2}$$

if the two probabilities are uncorrelated, i.e. $r_1(t)$ and $r_2(t)$ are independent.

The input of the integrator has thus a probability to be "high" proportional to in_1 and in_2; the integration will convert this probability to a voltage proportional to the product of inputs.

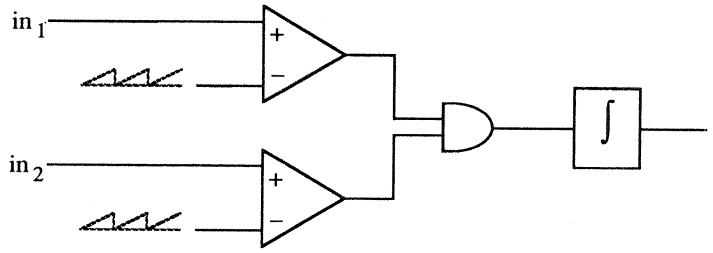

FIGURE 3
Stochastic product

3. APPLICATION TO NEURAL NETWORKS

The architecture described in figure 3 can be used in synapses by connecting in_1 to the input of the synapse and in_2 to a capacitor or memory point whose analog value represents the weight stored in the synapse. If a precision of n bits is required for the products, 2^n values are allowed on the capacitor and at the input of the synapse (in_1 and in_2), and 2^{n-1} levels must be generated at the inputs of the two comparators.

One method (among many others) to generate these signals is to use two digital n-bits random generators followed by D-A convertors. The generators may be n-bits

shift registers, with appropriate XOR feedback loops; different feedbacks lead to different sequences of numbers, and with well chosen feedbacks two uncorrelated random signals can be obtained, with constant probability over their dynamics.

It is possible to store the analog synaptic weight in each synapse by different ways; several digital memory points can be used, or only one analog memory. However, full advantage of stochastic products can only be exploited with this last solution: analog values can then be compared by very simple amplifiers, with reduced area on silicium.

One possibility is to store the synaptic weights by injecting charges on a capacitor. The unavoidable leakage currents are compensated by a periodic refreshment of the analog value on the capacitor: the value is compared to a set a predefined quantized levels, and is refreshed to the next upper level (for negative leakage currents). If the voltage decrease due to the leakage currents between two refreshments is less than the voltage drop between two levels, the analog value can be kept constant at the precision of the quantization [9]. Other methods to obtain analog synaptic weights include EEPROMs or external storage in RAMs.

Values coming from all the synapses connected to a single neuron are summed before the integration. This can be done with current sources as in [5] or [6] (fig.4.a). However, mismatchings and precision problems can occur since the values are converted into analog signals again. Such problems can be reduced by connecting only one synaptic activation at the same time (fig.4.b, where all switches are successively on); only digital values are then going in the neurons, whose layout may be more sophisticated to eliminate as much mismatching errors as possible during the integration.

In the first case, the total current, which represents the sum of all currents coming from the synapses connected to one single neuron, is directly integrated on a capacitor. In the second case, a first-order filter can be used to convert the total current into a voltage which can be used at the input of another network (for example in the case of multi-layer perceptrons).

4. CONCLUSION

Many layouts have already been published in the field of VLSI neural networks. The performances of all realizations rely on precise characteristics, such as the amount of multiplexed computations or the precision of the internal values. Depending on these characteristics, the networks are used in different applications; the use of stochastic computations in VLSI neural networks offer a good compromise between the precision of digital networks and the density of analog ones. Moreover, the precision obtained during the computations can reach the precision of digital neural nets; finally, because most of the functions used in the chip are common, the layouts and cells can be very standard, and the effort to design and implement the complete neural net is strongly reduced.

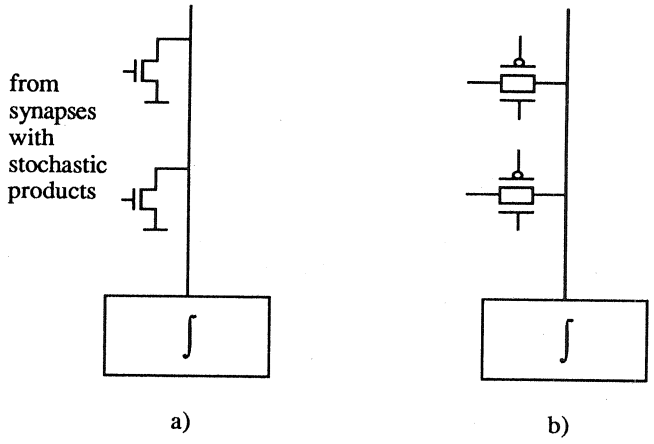

Figure 4
Sum before integration

REFERENCES

[1] Vittoz, E., and Arreguit, X., CMOS Integration of Hérault-Jutten Cells for Separation of Sources, in: Proceedings of the workshop on "Analog VLSI and Neural Networks", (Portland, May 1989).
[2] Blayo, F., and Marchal, P., Extension of Cellular Automata to Neural Computation, in: Proceedings of the International Joint Conference on Neural Networks, (Washington, June 1989).
[3] Verleysen, M., and Jespers, P., Precision of Sum-of-products in Analog Neural Networks, in: Goser, K., Ramacher, U., and Rückert, U., (ed.), Proceedings of the 1st International Workshop on Microelectronics for Neural Networks, (Dortmund, Germany, June 25-26 1990).
[4] Gregorian, R., and Temes, G.C., Analog MOS Integrated Circuits for Signal Processing, (J.Wiley & Sons, 1986).
[5] Graf, H.P., and de Vegvar, P., A CMOS Implementation of a Neural Network Model, in: Losleben, P., (ed.), Proceedings of the 1987 Stanford Conference on Advanced Research in VLSI, (MIT Press, Cambridge, 1987).
[6] Verleysen, M., et al., Neural Networks for High-Storage Content-Addressable Memory: VLSI Circuit and Learning Algorithm IEEE JSSC, (1989) 562-569.
[7] Van den Bout, D., and Miller, T., A Stochastic Architecture for Neural Nets, in: Proceedings of the IEEE IJCNN 1988, (Washington, 1988).
[8] Jespers, P., Windal, M., and Watteyne, T., An Integrated Binary Correlator Module IEEE JSSC, (1983) 286-290.
[9] Macq, D., and Jespers, P., Analog Storage in a Standard CMOS Technology, in print.