ANALOG VLSI SYNAPSE MATRIX WITH ENHANCED STOCHASTIC COMPUTATIONS

Michel Verleysen and Paul Jespers

Université Catholique de Louvain
Microelectronics laboratory
3, pl. du Levant
1348 Louvain-la-Neuve
Belgium

Most of the applications of neural networks require large arrays of synapses and neurons. Generally, neural networks are used in problems when classical algorithms are so complex that it is difficult or impossible to use them efficiently in real-time applications. When implementing the network on silicium, digital techniques can be used, with a great precision in the computations but with large cells which are to be multiplexed, or analog ones, which are faster and smaller, but which present drawbacks in their precision and cascadability because of the mismatching between components. This paper presents an architecture which offers a good compromise between analog and digital layouts, by using stochastic computations in small synapses with good precision.

1. Introduction

In most neural network applications, the size of the network is an important criterion. In image recognition for example, two possibilities are commonly used. Either each pixel of the image is associated to a neuron, and the size can quickly increase for real problems; either some kind of feature extraction is made before using a neural network for the recognition, but then a great number of characteristics are generally involved, and the problem of the dimension of the network is quite identical. The same problem arises in other kinds of applications : speech processing or preprocessing, time series prediction, data recognition or clustering, associative memories,...

When implementing neural networks with VLSI architectures, one of the main objectives must thus be the size of the network to realize. Two main possibilities generally exist. The first one consists in implementing digital processors, or neuron and synapses in the case of neural networks. As in a classical microprocessor, any desired precision can be reached; in order to achieve this precision, more elementary

cells can be put in parallel (adders, multipliers, memories, registers,...). Another possibility is to realize some kind of multiplexing, in order to keep a reasonable area on the chip; the drawback of course is that the computation times are increased with this multiplexing.

F. Blayo [1] 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! Digital solutions can be used when no or almost no time requirement exists. For example, an interesting application of Kohonen self-organizing maps is process control in chemistry [2]. In this problem, data flow is very low, but requires an important precision; a digital chip, or even a simulations on a computer, is thus appropriate.

In other applications, precision is not so important, but well time. In image compression for example, data can come from some kind of television or animated graphic, and the speed of the total computation made on the picture is thus a crucial point. Analog VLSI cells are generally faster than their digital equivalent. Furthermore, the small size of analog cells makes it possible to integrate parallelism, i.e. to put an important number of cells on the same chip, in order to compute several data simultaneously. The precision that can be obtained with analog computations is however limited by the precision of the components used in the chip [3]. For example, the differences obtained with current sources spread over 4 square millimeters of silicium can be as much as 12% (measurements made on UCL fabrication line, 3µm process, on big transistors - W/L=18/12 µm -, with same size, same orientation, same shape, same voltages and all precautions taken to have quite perfectly matched current sources). 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...

2. Mixed analog-digital circuits

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 [6].

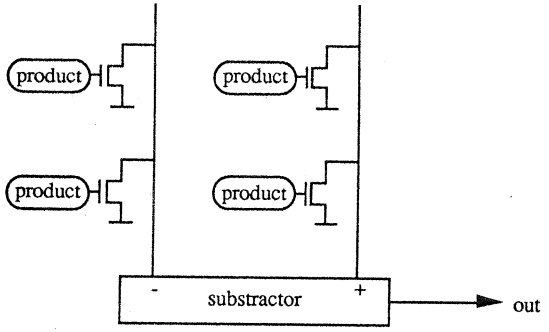

Figure 1. digital sum-of-products

Only replacing digital products by analog ones in figure 1 is not permissible. Two problems can indeed occur : first if only discrete values are present at the input of the substractor, which is the case with digital products (if n sources are connected, n+1 levels are possible), these values can be regenerated to one of the n+1 allowed discrete levels, even if the analog value was corrupted by some mismatch (the analog value is in fact rounded to the next upper or lower discrete level); this is a way to get rid of the imperfections of the analog components used. Of course, the number of levels is limited by the speed and the precision of the regeneration convertor; this kind of regeneration is obviously no more possible if analog products are used.

The second problem is the linearity : with the architecture of figure 1, each synapse, with its current source, must be linear (i.e. the current sourced by the synapse must be a linear product of its two inputs). In the case of digital products, there are only two possibilities (current or no current), and it is thus non-sense to speak about linearity. In the case of analog products, this is quite more complicated!

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.

3. 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).

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.

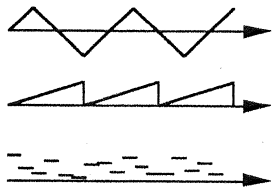

Figure 2. Example of functions used in stochastic products

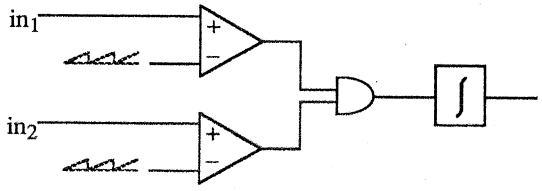

Figure 3. Principle of stochastic products

The result of the product 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 the inputs.

4. Synapses with stochastic products

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 [9]. 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.

In a matrix product, since in_1 is common for the entire line of synapses, the comparison between in_1 and the first random generator can be made only once; figure 4 shows the synapse, which contains only one comparator, one AND gate and one current source (10 transistors if the AND gate is replaced by a NAND, the problem of sign inversion being solved in the neuron). The refreshing system for the capacitor is not showed,, and must be added in the synapse; tendency is now to have a small capacitor, periodically refreshed through some kind of regenerating system as above described [10]. In such system, it is possible 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 [10]. Other methods to obtain analog synaptic weights include EEPROMs or external storage in RAMs.

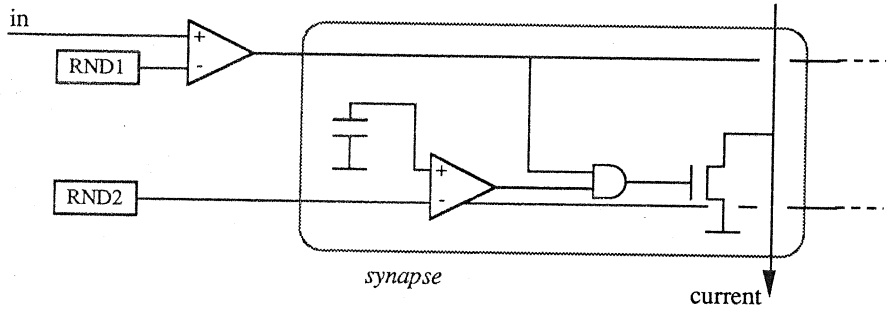

Figure 4. Synapse with stochastic product

We introduce here a variation of the stochastic product method; figure 5 shows its principle.

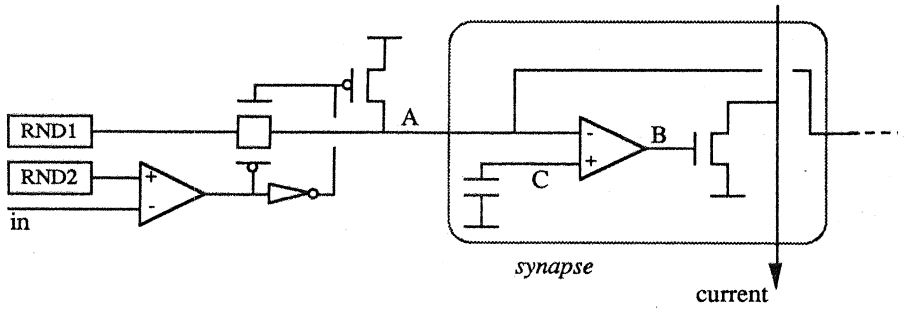

Figure 5. Enhanced method

Voltage A is defined by:

A = 1	with a probability 1 - in/max
A = RND1	with a probability in/max

where max represents the maximum of the dynamics for voltages in, C, RND1 and RND2.

Voltage C is the weight stored on the capacitor.
Voltage B is the output of the synaptic comparator.

We then have:

$$P(B=1) = P(C>A) = P(C>A \mid A=1) \cdot P(A=1) + P(C>A \mid A=RND1) \cdot P(A=RND1)$$

$$= 0 \cdot \left(1 - \frac{in}{max}\right) + \frac{C}{max} \cdot \frac{in}{max} = \frac{C \cdot in}{max^2}$$

Voltage B has thus a probability to be high proportional to the product of in and C, exactly as in the previous architecture. When integrating the sum of the currents, the correct value for the analog sum-of-products will thus be retrieved.

Without taking into consideration the refreshing system for the capacitor, the new synapse contains only 6 transistors; the number of horizontal lines crossing the synapse is also reduced to two (one showed, one for the selection of the capacitor). Comparing this cell with the one presented in [9], a first approximation of the gain in the cell area is 40%. As in the previous architecture, currents are summed before integration, so only one integrator per neuron is needed.

5. Conclusion

We presented an architecture for VLSI neural networks, where stochastic products are used in the synapses. Despite an increase in the computation times, in comparison with purely analog matrix products, this method has the advantage of offering a better precision in the computations.

If n bits precision are required (depending of the type of network, the learning algorithm and the application), 2^n steps are needed for a complete computation of the matrix product, but first approximation can be obtained earlier with well-chosen time constants in the low-pass filter included in the neuron to realize the integration. In standard applications, this is however much less than the number of steps required for a similar purely digital processor, where at least as many iterations as the number of synapses are required. In comparison with analog synapses, the precision which can be obtained is much better, since it almost no more depends on the precision of the components (only for the maximum number of connected synapses). The proposed architecture offers thus a good compromise between the precision of digital implementations and the size and speed of analog ones.

6. References

[1] 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).

[2] Tryba, V., and Goser, K., Self-organizing feature maps for process control in chemistry, Proceedings of the International Conference on Neural Networks, (Helsinky, Finland, June 24-28 1991).

[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] Verleysen, M., and Jespers, P., Stochstic computations in VLSI analog neural networks, Proceedings of the International Conference on Neural Networks, (Helsinky, Finland, June 24-28 1991).

[10] Macq, D., and Jespers, P., Analog Storage in a Standard CMOS Technology, in print.