A VLSI NEURAL NETWORK WITH CAPACITIVE SYNAPSES

M. Verleysen, D. Martin, P. Jespers

Université Catholique de Louvain Belgium
Michel Verleysen is working under an IRSIA fellowship

INTRODUCTION

Since a few years, considerable progress has been made in the field of neural networks. The recent developments of neurophysiology and in the understanding of how the human brain is working led to the idea that artificial devices whose structure is similar to the brain could share some of its properties. Conventional computers can perform efficiently complex but clearly defined tasks, such as resolution of mathematical problems, games, or computer-aided design,... However, a 5-years child can easily solve vision and perception problems that would be very difficult to treat by a computer. The need for special structures able to perform dedicated perception tasks is thus obvious.

The main properties of neural networks are their speed, their fault-tolerance, and their ability to solve optimization problems. They consist in several processors, called neurons, which are connected together through a coupling network. The power of neural networks comes from these coupling devices, called synapses; when a neural network is used as a content-addressable memory, the informations are stored in the whole network; this is called "distributed representation".

In the fields of vision, optimization and pattern recognition, neural nets seem to bring interesting properties, and they will probably be intensively used in industrial applications within a few years. However, all the advantages of neural networks are not fully exploited when these are simulated on conventional computers. Special hardware is thus necessary for the realization of industrial or research applications of neural networks.

VLSI REALIZATIONS OF NEURAL NETWORKS

Neural networks are generally characterized by a very regular structure. For the implementation, several designs can be used. The simplest one is to use resistors as synaptic connections, like Howard et al (1); the value of the resistor will then determine the synaptic weight. However, resistors are difficult to implement in standard VLSI technology, and take a large area on the chip. Graf and de Vegtar (2) use also current sources, but the matching between the different components contained in the chip is not easy to obtain.

Other techniques using current sources have also been presented by Verleysen et al (3). The problem of the matching between the different types of components (for example p-type and N-type MOS current sources) is replaced by a matching between the same type of transistors. However, the technological processes involved in the realization of VLSI chips make it impossible to obtain exactly the same electrical characteristics on the whole circuit; minor parameter variations can thus be observed, and the maximum number of neurons and synapses which can be put together on the same chip is thus limited.

The architecture proposed in this paper uses capacitors in place of resistors and current sources. The advantage with this sort of implementation is that capacitors realized in VLSI technology are accurate (more precisely, their ratios can be adjusted with a high precision); moreover, the techniques used in switched-capacitors filters to enhance the matching between the different components, like those presented by Droettboom and Greenes (4) can be used. These considerations will make neural networks using capacitors as synapses more robust to an increase of the number of interconnected neurons, and thus more suited to industrial applications.

HOPFIELD'S FULLY INTERCONNECTED NETWORK

For recognition or vision tasks, several neural architectures can be used: multi-layer perceptrons, Hopfield's fully interconnected networks,... This paper will not discuss about the various possibilities, nor about their particularities and their ability to solve specific problems. For the simplicity, we will consider in this paper only fully interconnected networks: the regularity of their structure make them the best candidates for an efficient solution.

A Hopfield fully interconnected network consists in N neurons, each one being connected to each other by one of the N² synapses (figure 1).

The output value of neuron i, V_i, can be either +1 or -1. In this model, we limit the possible connection values T_{ij} to +1, 0 and -1. Learning algorithms coping with this restriction gave about the same results than with continuous synaptic weight values; an example of them is given by Sirletti et al (5). The function of each synapse is to multiply the synaptic weight T_{ij} by the output value V_j of the neuron to which the synapse is connected. The output values V_j of the synapses in the same column are then summed by the neuron, and the results go through a non-linearity so that the output of the neuron is +1 if its input is positive, and -1 in the opposite case. The output of each neuron is then fed back into the synapses.

Although the architecture described here is an Hopfield network, the design used to implement the circuit is general and can be adapted for example to perceptrons; the only differences will be in the connections between neurons and synapses, while their design will remain identical.
CAPACITIVE VLSI ARCHITECTURE

The new architecture proposed in this paper is shown in figure 2. In this fully interconnected network, the neurons (at the bottom of the figure) sum the contribution of each synapse. The result of each sum is then compared to a threshold, latched and finally fed back into the square plane formed by the synapses. The reason to clock the whole circuit is to examine the intermediate states before the stable output is reached.

Each neuron consists in three separate parts: an integrator, a comparator and a latch. The integrator collects all the charges coming from the C_{ij} capacitors (for all j) and sums them into C_i. This scheme of integrator is in fact independent of any parasitic capacitor. Indeed, possible parasitic capacitors are shown in figure 3 (C_{p1} ... C_{p3}). The capacitors C_{p1} and C_{p2} have no influence on the integration because they are connected to low-impedance nodes. The case of C_{p3} is different. Before any integration the whole system is reset and therefore the potential of node A will be V_{ref}. During the integration period, the potential on this node will vary but will finally converge to the same potential as the non-inverting input of the operational amplifier: V_{ref}. This comes from the fact that the operational amplifier built with a feedback loop will minimize the difference between its two inputs. Since C_{p3} will not see different potentials between the beginning and the end of integration, this capacitor will have no influence on the result. The main speed limitation will come from the settling time of the operational amplifier charged by a capacitor.

The output of the integrator is then compared to V_{ref} and the result is latched by Φ. The latch provides two complementary logic outputs V_j and \bar{V}_j which are fed back into the synapse plane.

As described in the previous section, each synapse has to perform the product T_{ij}v_{ij}. This is implemented by transmission gates which will route to C_{ij} either V_j, \bar{V}_j or V_{ref} respectively corresponding to the weights +1, -1 and 0.

The combination of weights is typical for each problem that can be solved by the network. Therefore, for each specific application, the weights have to be loaded on the circuit.

The architecture proposed here allows to program the weights by the means of memory points connected to the nodes M_i0 and M_i1. When equal to 01, 10 or 00 the weights are respectively +1, -1 or 0. The combination 11 is illegal since it would shorten V_j to V_j.

After programming the memory points, the circuit is ready for computing. Computing is done on digital input data introduced through the C_{ij}. Evaluation all the capacitors have to be discharged. This is accomplished with the reset command. When high, all the C_j are shortened. The non-inverting input of each operational amplifier is set to V_{ref} so one terminal of each C_{ij} is virtually connected to V_{ref}. The other terminal of those capacitors is also connected to V_{ref} through multiplexers activated by the reset line; all the C_{ij} are thus discharged. The input capacitors C_{in} are reset in the same fashion.

After resetting the capacitors, inputs can be applied to the C_{in}. When this occurs, care has to be taken for the reset command. The Ci and C_{in} may not be shortened otherwise charges coming from V_{in} would not be properly integrated in C_i. The signal "reset" has thus to be disabled shortly after applying the inputs. On the other hand, the C_{ij} have to be shortened otherwise the results of a previous evaluation could influence the integrators' values. Indeed, the output of each integrator would then be given by:

\[ V_{in} \cdot C_{in}/C_i + \sum_{j=1}^{n} T_{ij} \cdot V_j \cdot C_{ij}/C_i \]  \hspace{1cm} (1)

where V_j is the output of neuron j corresponding to a previous computation. For this reason, the multiplexers connected to the C_{ij} are activated by a different signal (reset2). This one will be logic high one more clock cycle to assure that V_j (output of neuron i) corresponds to (and only to) V_{in}. A timing diagram is represented in figure 4.

After this initialization phase the convergence process can begin: each clock pulse will correspond to a new computation of the neuron values, and the network will have converged when all tapped neuron values are stable. This usually occurs after 10-50 cycles for networks with 10-100 fully interconnected neurons.

CONCLUSION

A VLSI architecture for artificial neural networks has been described. The example shown here presents an Hopfield's fully interconnected network, but the same design can be adapted to any neural architecture. In comparison with designs using resistors or current sources to realize the synapses, this architecture offers several advantages: the accuracy that can be reached with capacitors is increased, and the number of synapses that can be connected to the same neuron is greater. Furthermore, only a few transistors are imbedded in each synapse; and since only the relative values of the capacitors are important, their size can be reduced to very small values. The total area occupied by one synapse is thus small too, and a great number of them can be integrated on the same chip. While the minimum values for the capacitors to avoid interference by parasitic ones are still to be determined, estimations show that a density of about 40-50 synapses/mm² in a 3-microns technology can be considered.

In order to prove the feasibility of such architecture, a 8-neurons network has been realized with discrete components. The value of synaptic and input capacitors are 4.7 nF,
and the value of integration capacitors is 10 nF. This network is programmed as a content-addressable memory. Since the synaptic connections can only take three different values (+1, 0 and -1), a learning algorithm adapted to this restriction had to be used. The classical Hebb's rule presented by Hopfield (6) shows a very low storage capacity (about 0.15N vectors of N bits in a N-neurons network); furthermore, this capacity is slightly reduced when the synaptic weights are truncated to +1, 0 and -1. The pseudo-inverse rule from Personnaz (7) gives a better storage capacity if the synaptic weights can take any value; on the opposite, if only three values are allowed, the storage capacity is strongly decreased, and unacceptable for useful applications. Another learning rule (5), coping with this restriction, has thus been used. This algorithm maximizes the stability of the stored vectors by maximizing the difference between the input of each neuron and its threshold, and this for all the patterns to memorize. This is done by an off-line resolution of a simplex system, before storing the weights into the network. The storage capacity obtained with this algorithm is about the same as the one obtained with the pseudo-inverse rule with continuous synaptic weights. However, the weights are restricted to +1, 0 and -1 in the resolution of the simplex system itself, and this learning rule is thus more adapted to electronic realizations. The possibility to suppress the clocking in the circuit has also been added, in order to check that the results obtained with an asynchronous convergence are identical to the ones obtained with a clocked circuit.

A test chip in a CMOS 3-microns VLSI technology has also been realized (figure 5). It contains only two neurons and four synapses, since the purpose of this chip was only to verify the architecture and to measure some parameters (values of capacitors, ...). The capacitors are very big in this chip, but can easily be reduced in larger networks since only the ratios of their values is important.

The VLSI implementations of neural networks will open the way to industrial applications in the next few years. An architecture has been presented in this paper; the trend is of course to integrate larger networks, with hundreds and perhaps thousands of fully interconnected neurons. The development of VLSI neural networks can be compared with the first memories: beginning with only a few hundreds of bits, chips now reach the capacity of 4 million bits; perhaps in some years, associative memories with thousands of stored patterns will be available...

REFERENCES


Figure 1 Hopfield's fully interconnected network

Figure 3 integrator's parasitic capacitors

Figure 4 reset clocking diagram

Figure 5 photomicrograph of test chip
This paper describes a silicon implementation of an artificial neural network using analog techniques. The problems arising during chip design are analysed and it is seen how the constraints on the accuracy provided by an analog implementation can be relaxed by a proper selection of the learning rule. A chip containing an array of 32 neurons with 32 inputs is actually under development.

**INTRODUCTION**

The large interest in Artificial Neural Systems (ANS) finds one of its main motivations in the possibility to build artificial neurons by means of VLSI integrated circuits. Today's technology seems to offer good chances to obtain results that some years ago were not feasible. The real problem is that no efficient model of the brain behavior is yet available although several approaches have been attempted. Three different methods of implementation can be considered: biological, optical and electrical. The biological approach is perhaps the most attractive one, but it cannot be considered a workable solution for the next few years. Also optical computing seems to have interesting perspectives but state of the art technology is still far from the scale of integration required by a neural network. The electrical is today the only feasible solution because of the high density of integration on a single silicon die and the device speed is rather high, although slower than competitive technologies such as the Gallium Arsenide or the optical.

Several implementations of Artificial Neural Systems have been proposed in the last years, and we can distinguish among three different classes:

* Software approaches, using traditional sequential or parallel computers (1).
* Dedicated hardware approaches with special architectures implementing directly the computational kernel of ANS.
* Ad-hoc approaches, where specific hardware emulates some typical functions of real brain (e.g. the retina (2)).

Currently the major trend towards implementation is to use simulation software running on general-purpose computers either with or without digital accelerators. Little work has been done on dedicated silicon devices (3), (4) and (5) which implement directly in hardware artificial neural models.

This work shows the possibility to build silicon neural cells. Multi-layer perceptrons have been analyzed by means of computer simulations to find the best solutions for an integrated circuit. Our analysis deals with the physical requirements of an artificial neuron and therefore relates to problems such as the number of neurons required, the accuracy and resolution of the synapses, the dynamic range of weights, the neuron speed and the number of connections. The main topic of our study was the behavior of the neural cell during the learning phase and the retaining of the knowledge (Long Term Memory).

**MULTI-LAYER PERCEPTRON**

We use as reference a classical theoretical model, namely the multi-layer perceptron (6) shown in Fig. 1 because this architecture shows interesting features for a hardware implementation using dedicated integrated circuits. The structure of our reference uses two layers of Processing Elements, each one with full interconnection. The inputs of the first layer are fully digital while all the other inputs and outputs are continuous. The outputs of the network may then be converted into a digital form to be used by a digital system. The internal behavior is therefore fully analog as the human brain is.

Generally speaking, a multi-layer perceptron is composed of \( n \) layers with \( M \) inputs and \( M \) outputs respectively, where \( z = 1 \) is the first hidden layer and \( z = 2 \) the last (output) layer. Every layer is composed of an array of neurons and each one computes a weighted sum of the inputs \( I_i \) to produce the outputs \( O_j \):

\[
O_j^z = F \sum_{i=0}^{N_z} \left( W_{ij}^z I_i + N_z O_j^{z+1} \right)
\]

where \( I_i \) and \( O_j \) are respectively the \( i \)-th input and the \( j \)-th output of the \( z \)-th layer. \( W_{ij} \) and \( \Theta_j \) are the synaptic weights and the neuron threshold, respectively. The term within brackets is passed through a non-linear activation function \( F(x) \), usually a sigmoid or similar function:

\[
F(x_{i,j}^z) = \frac{1}{1 + e^{-\frac{x_{i,j}^z}{\gamma}}}
\]

where \( x_{i,j}^z \) represents the weighted sum of the inputs and the threshold as from formula 1. The interconnectivity between adjacent layers of the network can be expressed as follows:

\[
I_i^{z+1} = O_j^z
\]

The learning rule is a back-propagation delta rule. The weight changes are computed according to the difference between the network outputs and the target pattern. A slightly different formula applies to the output layer and to all hidden layers. Namely, for the output layer each weight is modified by means of the following law:

\[
\Delta W_{ij}^z(t+1) = -\eta O_j^z \sum P_i \delta_j^z + \alpha \Delta W_{ij}^z(t)
\]