7.2 A NEW CMOS ARCHITECTURE FOR NEURAL NETWORKS

Presented at the "Workshop on VLSI for Artificial Intelligence", Oxford, 20-22/07/88.

Michel Verleysen, Bruno Sirletti and Paul Jespers

Published in "VLSI for Artificial Intelligence", J G. Delgado-Frias, W. R. Moore, ed.r, Kluwer Academic Publ., 1989, pp 209-217

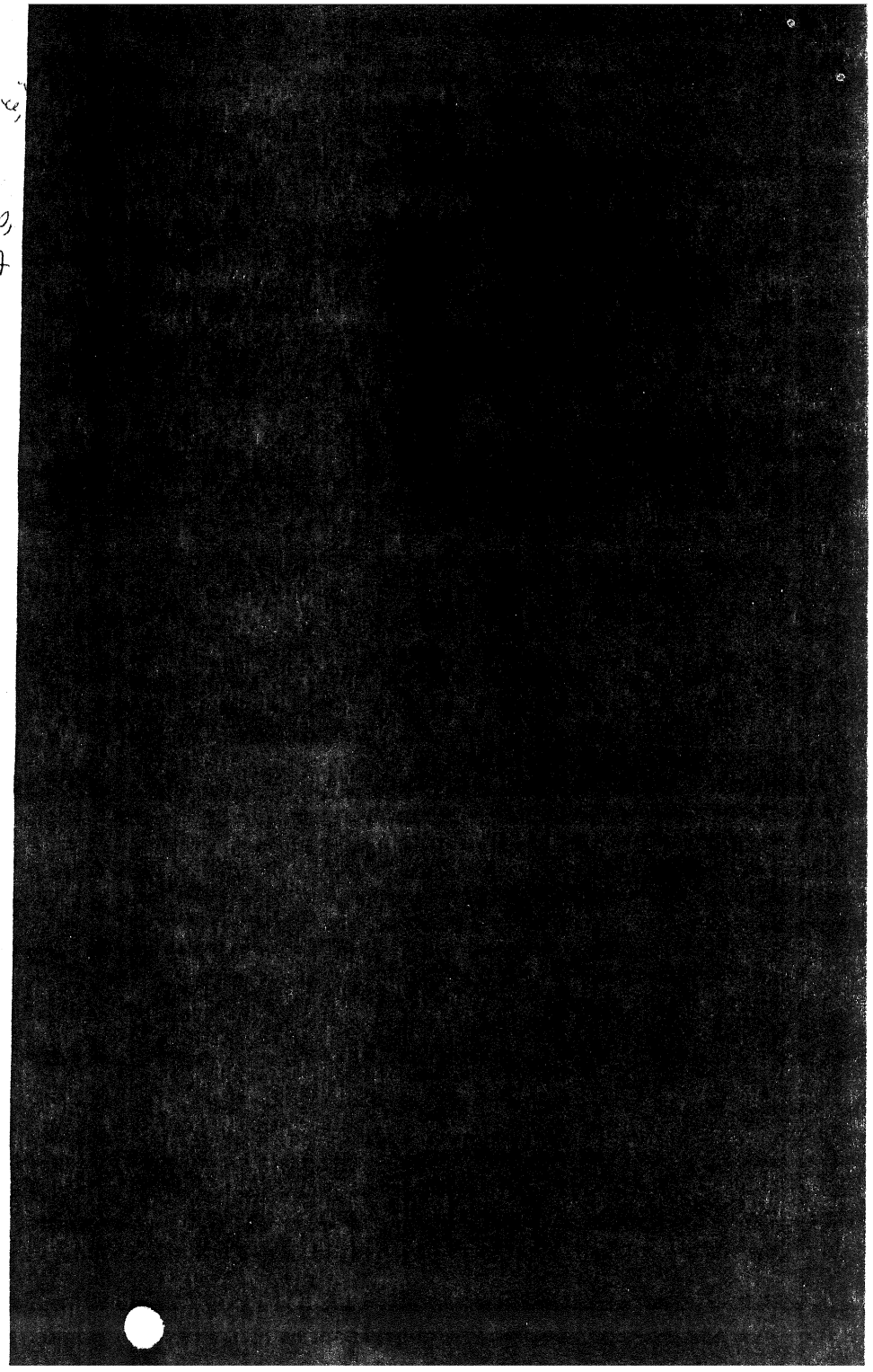
INTRODUCTION

Conventional computer architectures and artificial neural networks have radically different structures and utilities. While computers are used to quickly perform complex but clearly defined tasks, neural networks are particularly adapted to solve more blurred problems like those encountered in perception, recognition and optimization. The hope is that neural networks might share some of the processing capabilities of the human brain. This is especially interesting in perception problems which are very long to compute by conventional computers because of the need for powerful sequential algorithms. The tradeoff between speed and efficiency implies the use of more powerful computers in order to run the complex algorithms needed to solve real-time perception problems. Nevertheless sequential solutions are not "natural" for this class of problems in which a large amount of simple but repetitive computation is needed.

In a human brain, about 10^{12} neurons form a three-dimensional, highly interconnected network. Each neuron can be connected up to 10^4 other neurons (Rumelhart and McClelland 1986); the huge computational power of the brain resides only in this highly parallel structure: although the propagation times between the neurons and the neuron's delay time are very long, a complex vision problem can be solved in less than 500 milliseconds. It seems therefore obvious that no complex algorithm is used inside the brain.

The idea which lead to the design of artificial neural networks is the emulation of the brain's structure in order to take advantage of its perception properties. In the brain, each neuron can be seen as a single processing element which performs a weighted sum of its inputs (the outputs of the other neurons). The neuron turns on if the result is greater than an internal threshold (Fahlman and Hinton 1987). The difficulty in the realization of artificial neural networks resides in performing this weighted sum efficiently .

In image processing, to obtain interesting results, the network must be large enough to perform recognition tasks on a sufficient number of pixels. Since the power of parallel solutions in general, and neural networks in particular, comes from their ability to solve complex problems quickly, the processing speed of the networks must also be high enough to meet real time requirements. The recent advancements in the VLSI technology make possible the implementation of large fast networks due to the increasing number of transistors which can be integrated on a single chip.



ARTIFICIAL NEURAL NETWORKS

An electronic neural network can be seen as an array of simple processing elements (*neurons*) connected through a coupling network (a single connection between two neurons is called *synapse*). Each pair of neurons can be connected by a synapse which is either excitatory (positive) or inhibitory (negative) (Figure 1). All the information contained in the network resides in the connection values and is thus distributed within the whole system. The neurons only perform a weighted sum of all their inputs and a comparison between this sum and a fixed threshold.

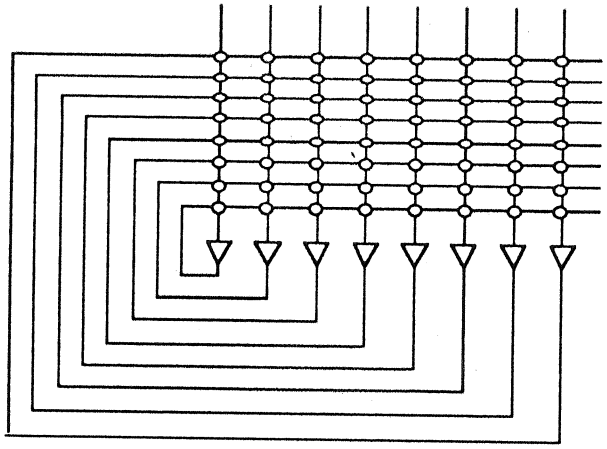


Figure 1 Fully interconnected neural network

In Hopfield's model, the neurons are first set to the input pattern values (Hopfield 1982). The neuron values are then reassigned by computing for each neuron the sum of all its inputs weighted by the synapse strengths; this processus is repeated until a stable state is reached.

The Hopfield's model allows the realization of a content-addressable memory by setting appropriate connection values. A simple rule inspired from biological models, called the Hebb's rule (Hebb 1949), is used to compute the different connection weights. It consists in increasing the strength between two neurons if they have the same state in the pattern to memorize, and in decreasing it if they don't. This simple rule can be easily formalised:

$$T_{ij} = \sum_{p} V_{ip} V_{jp}$$

where T_{ij} is the connection strength between neurons i and j

V_{ip} is the value of neuron i (+1 or -1) in the pattern p to memorize.

Although this rule is very simple, it allows the realization of a content-addressable memory which can store up to 0.15N patterns of N bits (Lippmann 1987). More complex rules increase this 0.15N limit (Personnaz 1986).

VLSI NEURAL NETWORKS

Synapses using resistors

By analogy with the theoretical models of neural nets, the first implemented networks used resistors to realize the coupling matrix (Howard *et al* 1987). Each synapse contains a resistor whose value determines the strength of the connection. In order to allow positive and negative values, the resistors are connected either to the non-inverting or inverting output of the neuron.

Although this is the simplest and most powerful way to realize these networks, two disadvantages appear because of the use of resistors. First, the different connection strengths need resistors with different values which occupy various areas; this prevents the network from having an highly regular structure where each synapse would occupy the same area on the chip. Secondly, because the connection values must be set during the design of the chip, no programming and no learning are allowed. Since in most applications the patterns to be stored are not previously known, chips with fixed connection values do not cover the whole range of applications we can expect from neural networks.

These two restrictions strongly limit the use of resistors networks.

Synapses using current sources

A new design for VLSI neural networks has been proposed in recent papers (Graf and de Vegvar 1987, Tsividis 1987). In those circuits, each synapse is a programmable current source controlled by the output of the neuron to which the synapse is connected; following the sign of the connection and the one of the connected neuron, the synapse sources or sinks current to the input line of the second neuron to which the synapse is connected. In such implementations, the synapse possible values are generally limited to +1, 0 or -1. Indeed, if more values were allowed, more memory points would be necessary to store the connection weight; the area of one synapse would be larger, and less neurons could be implemented on a single chip. The current trend is to reduce the area of the synapse by using algorithms which show no performance decrease when they are used in a network with three-values synapses.

In the Hopfield's model, a XOR function must be realized in each synapse between the output of the connected neuron and the memorized weight. According to the result of this XOR function, a current is sourced by a P-type transistor or sunk by a N-type one (Figure 2). All the sourced and sunk currents are summed on the input line of the connected neuron. The logical function of the neuron is to detect if more or less synaptic currents are sourced than sunk. Since the only input of the neuron is the sum of all the synaptic currents, we must detect if this sum is greater or less than the threshold (0 in our case). A problem arises immediatly: the P-type and N-type current sources will never be exactly equal because of the mobility differences between the two types of charges. This mismatching is multiplied by the number of active synapses, and can quickly reach the value of one synaptic current; this limits considerably the number of neurons which can be connected together. Figure 3 shows the maximum number of neurons versus the ratio mismatching/single synaptic current. It shows that with the physical feasible values for this ratio (about 3-5 %) (Nicollian and Brews 1982), the number of neurons is strongly limited.

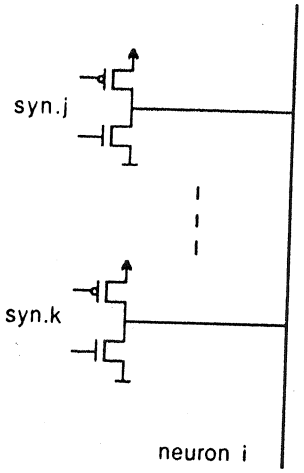


Figure 2 P and N current sources

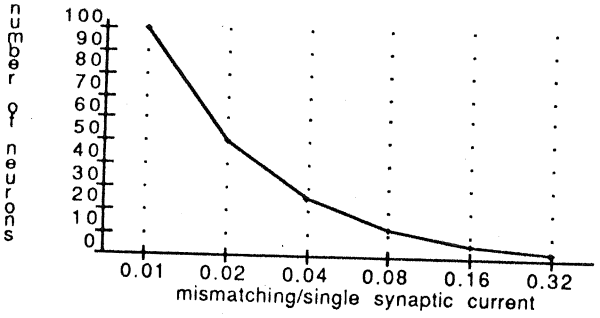


Figure 3 Maximum number of neurons

We thus need a method which suppresses these drawbacks in order to allow the implementation of a large number (hundreds) of neurons on a single chip.

THE PROPOSED CIRCUIT

Introduction

The main purpose of the proposed circuit is to suppress these drawbacks to allow the full interconnection of hundreds of neurons on a single chip. In order to eliminate the mismatching between the P-type and N-type current sources, all the synaptic currents will

be sourced by N-type transistors only, but the positive currents will be sourced on one line and the negative currents on another one. Two different sums are then realized (one for the positive currents and one for the negative ones); the function of the neuron is to compare the total currents on the two different lines; the neuron will switch on if the total positive current is greater than the total negative one; otherwise it will switch off. The new values for each neuron are then fed back into the network through the different synapses. When all of the neuron values don't change anymore, a stable state is reached and the set of the neuron values forms the output of the network.

Synapses

Since a N neuron fully interconnected network requires N^2 synapses, the main part of the chip area will be occupied by the array of connections. In order to increase the neuron density on the chip, we will thus try to reduce as much as possible the synapse area. The logic diagram of a synapse is reproduced in Figure 4.

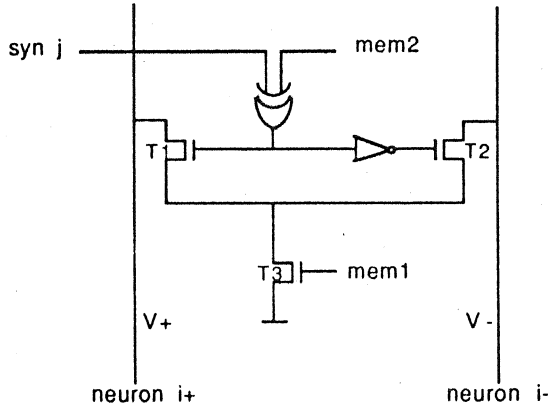


Figure 4 Synapse

As we saw previously, two values are stored in each synapse: "mem1" represents the absolute value of the connection while "mem2" represents its sign. The synapse can thus take the three different values -1, 0 and +1.
The output of the neuron j to which the synapse is connected is multiplied by a XOR function with the content of "mem2". This determines the sign of the current. "Mem1" commands the gate of a current source to determine if a current is to be sunk in this synapse. It is obvious that a single synaptic current has to be as small as possible because N currents can be summed on a neuron input. To avoid voltages in mem1 other than 0 and 5V, we use a long transistor (T3) as current source. The three permitted different combinations of mem1 and mem2 allow thus to sink a current on neuron i+, on neuron i-, or to sink no current at all.

Neurons

The function of the neuron is to compare the two differents currents on lines neuron i+ and neuron i-. First, the two currents are converted into voltages by transistors T4 and T5 (Figure 5). These voltages are themselves compared in the reflector formed by T6 to T9.

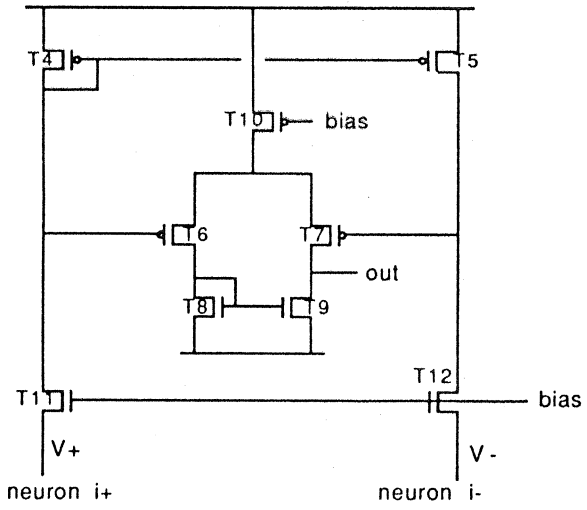


Figure 5 Neuron

Since a large number of synapses can be connected to the same neuron, we must add two transistors (T11 and T12) to keep the voltages V+ and V- as fixed as possible. Indeed, without these transistors, these voltages would decrease when we increase the number of active synapses; with a great number of neurons, we would quickly reach a point where the addition of one synapse will have no more effect on the neuron current because of the V+ and V- voltage diminution. Figure 6 shows the voltages on the output line (out) when there are N/2 active synapses on line neuron i+ and N/2+1 on line neuron i-, or vice versa. If we add a buffer to this output, we see that at least 500 synapses are allowed on the same line.

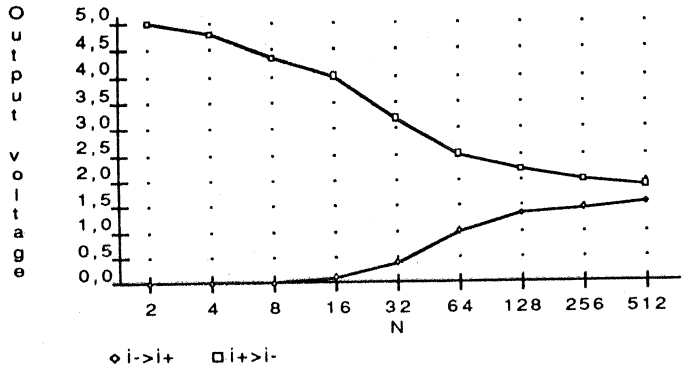


Figure 6 Neuron discrimination

Global architecture of the circuit

As described in the previous section, the proposed architecture can be used to implement a large number of neurons. However, to verify the theoretical predictions and to measure the computation speed of the network, we designed a 3x3 mm test chip in a CMOS 3 micron technology. The chip we describe here is a fully interconnected Hopfield network with only 14 neurons and 196 synapses.

Two memory points are contained in each synapses. To program this memory, a decoder selects the address line (i.e. the column) in which we want to write; the 28 memory points are then programmed simultaneously through 28 input pads. In the computation mode, 14 of these 28 pads are used as the outputs of the neurons. The loop between the neuron outputs and the synapse inputs can be interrupted to compute the neuron values step by step for test purpose.

Layout

Figure 7 shows the layout of the 14 neuron test circuit. It is divided in 14 neurons (1), 196 synapses (2), the RAM decoder (3), 14 input/output pads (4) used either to program the RAM or as output of the circuit and 14 input pads (5) used to program the RAM.

The chip has been realized in a CMOS 3 microns technology with single metal and single poly. Obviously the use of a large chip with a smaller technology will increase the number of neurons which can be put together on a single chip.

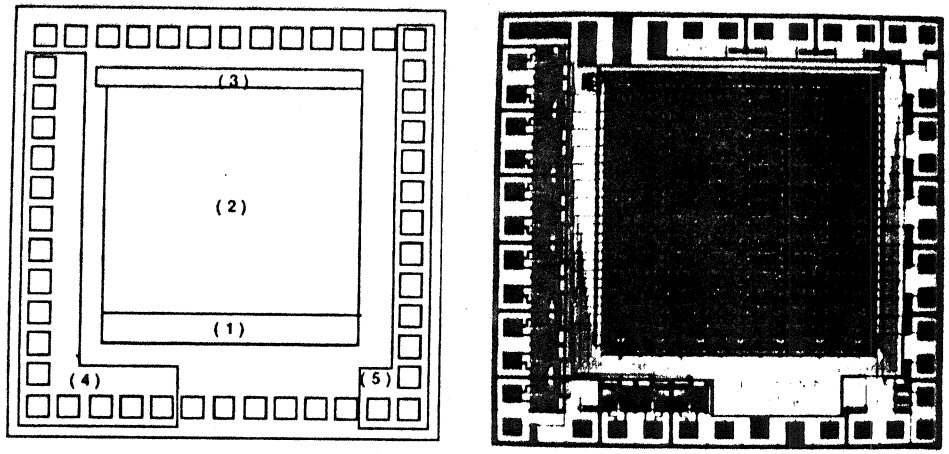


Figure 7 Layout

CONCLUSION

We have described a new VLSI architecture for the implementation of neural networks. Classical VLSI approaches present an important problem: computation errors appear when we increase the number of connected neurons. To suppress these drawbacks, we designed neurons and synapses where the comparison between currents is much more accurate than in circuits based on N-type and P-type current sources. This architecture allows the implementation of a fully interconnected network with hundreds of neurons, or a layered network with many more neurons.

We suppose here that only three values are allowed for each connection; the scope is to minimize the synapse area, since only two memory points are necessary to store the connection strength. However, classical CAM algorithms such as Hebb's rule need more values for each connection. There are two solutions to this problem: either we increase the number of possible strengths in each synapse, but its area will also increase, or we develop new algorithms more adapted to this architecture (it is possible to obtain better results than the Hebb's rule even with only three allowed connection values (Sirletti *et al.*, 1988)).

To design very large arrays of synapses, we are obliged to reduce their areas. We think that the use of different memory points, for example DRAM in place of SRAM, will considerably reduce the synapse area. We can compare the actual state-of-the-art in neural networks with the early developments of memories and look forward to very useful arrays of neurons with millions of synapses...

Acknowledgement

M. Verleysen and B. Sirletti acknowledge the support of IRSIA.

References

Fahlman, S. E. and Hinton, G. E., "Connectionnist Architectures for Artificial Intelligence", *IEEE Computer*, pp. 100-109, Jan 1987.

Graf, H.P. and de Vegvar, P., "A CMOS Implementation of a Neural Network Model", in *Proc. 1987 Stanford Conference in Advanced Research in VLSI*, pp. 351-367, 1987.

Hebb, D.O., *The Organization of Behavior*. New York: Wiley, 1949.

Hopfield, J.J., "Neural networks and physical systems with emergent collective computational abilities", in *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554-2558, April 1982.

Howard, R.E., Schwartz, D.B., Denker, J.S., Epworth, R.W., Graf, H.P., Hubbard, W.E., Jackel, L.D., Straughn, B.L. and Tennant D.M., "An Associative Memory Based on an Electronic Neural Network Architecture", *IEEE Transactions on Electron Devices*, vol. ED-34, pp. 1553-1556, 1987.

Lippmann, R.P., "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, pp. 4-22, April 1987.

Nicollian, E.H. and Brews J.R., *MOS: Physics and Technology*. New-York:John Wiley & Sons, Inc.,1982.

Personnaz, L., "Etude de Réseaux de Neurones Formels, Propriétés et Applications", Doctoral Thesis, University of Paris, June 1986.

Rumelhart, D.E., McClelland, J.L. and the PDP Research Group, *Parallel Distributed Processing*, vol. 1&2. Cambridge: MIT Press, 1986.

Sirletti, B., Verleysen, M. and Jespers, P.G.A., *A New Learning Algorithm for Content-Addressable Memories Using Hopfield's Neural Networks*, UCL Internal Report, April 1988.

Tsividis, Y. and Satyanarayana, S., "Analogue Circuits for Variable-Synapse Electronic Neural Networks", *Electronics Letters*, Vol. 23 N° 24, November 1987.