Microelectronics for Neural Networks
U. Ramacher, ed. (Elsevier, 1991)

Silicon integration of learning algorithm and other auto-adaptive properties in a digital feedback neural network

P.Y.Alla[1], G.Dreyfus[2], J.D.Gascuel[1], A.Johannet[2], L.Personnaz[2], J.Roman[1], M.Weinfeld[1]

1 Ecole Polytechnique
Laboratoire d'Informatique
91128 Palaiseau Cedex, France

2 Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris
Laboratoire d'Electronique
10 rue Vauquelin 75005 Paris, France

Introduction

In the past few years, a lot of efforts has been devoted to the integration of neural networks [1,2], with much emphasis on the implementation of the network itself, leaving the burden of learning to a host, possibly parallel, computer [3]. However, the idea of implementing training on the chip itself is attractive for two reasons : (i) the learning phase is usually very time-consuming ; (ii) on-chip learning makes the network more autonomous and opens the way to building elaborate assemblies of networks. The present paper discusses the capabilities of a neural network chip, fully connected with feedback, using binary neurons with parallel synchronous dynamics, intended to be used as an associative memory [4] ; the chip integrates a learning algorithm and also some additional, potentially useful features such as sel identification of correct relaxation on a stored vector (a prototype), and to the discussion of the mair silicon implementation issues.

The main architectural and technological choices

The first integrated neural networks were based on analog technology, with the advantage of speed an compactness, but with severe limitations in terms of programmability and precision of the synapti coefficients, as well as some sensitivity to crosstalk and spurious noise [5-14]. Nowadays, many innovativ efforts are dedicated to various techniques to overcome these deficiencies, and the issue of analog versu digital technology is far from being settled. Nevertheless, digital circuitry has theoretically none of tl aforementioned drawbacks : it allows the implementation of any function (for network relaxation as we as for in-circuit learning), with arbitrary precision, is strongly immune to noise, and has no specil interfacing problems. The real difficulties arise from the connectivity issues and the larger number transistors per function (memory or arithmetic units), leading to rather large silicon areas. Even though less numerous, there are or have been several attempts to design digital networks [15-20].

A solution to the connectivity problem lies in trading off parallelism with computation time, using some kind of multiplexing scheme. We have chosen a linear systolic architecture [21,22], in which each neuron has its own synaptic coefficients stored locally, in a circular shift register (see figure 1). This choice limits and simplifies the data lines connecting each neuron to the rest of the network, but leads to a large area for each neuron, which is mainly devoted to memory. Taking into account the possibilities of present technologies, we decided to focus our efforts towards a network of 64 fully-connected binary neurons with feedback, with a chip surface of about one square centimeter, which is conservative enough to give reasonable yields. Designing such a circuit consists mainly in designing the elementary neuron, which can be duplicated as a macrocell with automatic connection of the signal ports.

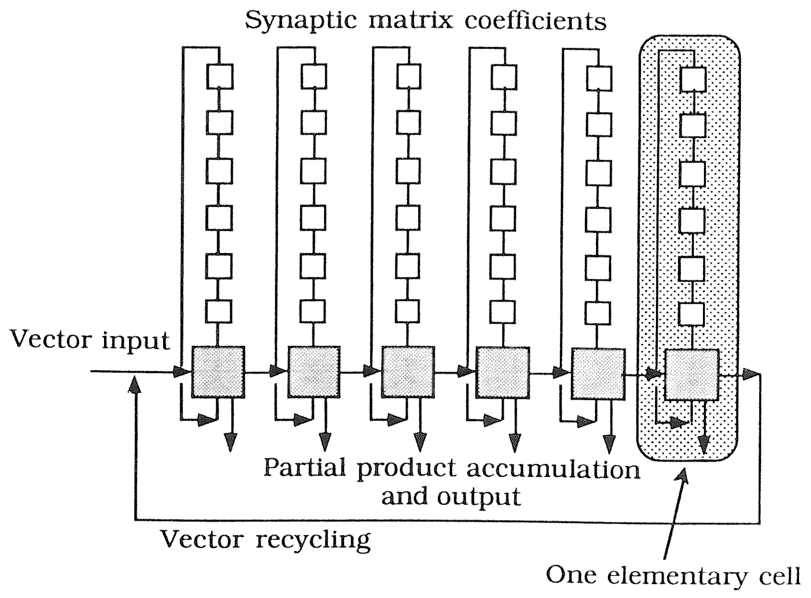

Synaptic matrix coefficients
Vector input
Partial product accumulation and output
Vector recycling
One elementary cell

Figure 1

One of the advantages of digital architecture lies in the fact that training can be integrated on the chip, at no great expense in terms of additional complexity or silicon area, provided an appropriate learning rule is used. As we shall see later, other features can be rather easily added to the basic architecture, providing supplementary possibilities in view of multi-network architecture implementation.

In addition, it should be noticed that the same architecture, with minor alterations, can be used to implement multilayer networks of binary or multivalued neurons [17], which are becoming increasingly popular with the emergence of constructive methods allowing a rational use of such networks for classification [23].

Towards multi-network architectures : some additional properties

A common approach to applications of neural networks has been the use of one single network to perform the whole task. It can be argued that a hierarchical approach might be more efficient : therefore, one should try to find ways of assembling relatively small networks, strongly connected, into higher-level architectures, with sparser connections, as biology would suggest it. However, this strategy requires that the basic building blocks have the ability of making decisions about information processing and routing autonomously, including learning, thereby alleviating the need for control from an external superviser [24]. Any external supervision, downloading synaptic coefficients or assessing the results of a relaxation, for instance, may induce excessive overhead and impair drastically the interest of these specific integrated components.

Implementing a learning rule

The choice of a learning rule depends strongly on the task that the network is supposed to perform, and on the architecture of the network. In the present case, we want to take advantage of the storage and retrieval properties of a Hopfield network of binary neurons. Therefore, we need a learning rule which has the ability of storing binary patterns as attractors of the dynamics of the network. In addition, the learning rule should be easily integrable, which means that it has to be local (contained in each neuron), not needing additional signal routing or global external calculations to be broadcast to each cell. [25]

The projection rule [26] was shown to allow the storage and retrieval of any set of prototype patterns, but its direct implementation on an integrated network is very difficult. It has been proved [27] that the Widrow-Hoff iterative learning rule can be used to train a fully connected network intended for use as an associative memory : if the number of prototypes is smaller than the number of neurons, the synaptic matrix obtained by the Widrow-Hoff rule is identical to the projection matrix. In addition, the procedure is iterative and local : the computation of the variation of the weight of the connection between neurons i and j, in neuron i, requires a multibit information (the potential) which is locally available, in addition to the state of neuron j. Thus, it is a good candidate for integration on the same chip as the network itself.

Since the use of standard floating-point arithmetic in each neuron is ruled out because of its excessive complexity, the main issues in the integration of the training algorithm on the network itself are (i) the accuracy of the synaptic coefficients used in the retrieval phase, and (ii) the accuracy required for the computation of the coefficients. The first issue is related to the fault tolerance of the network : Hopfield-type networks exhibit some degree of fault tolerance, in the sense that the retrieval properties of the network degrade gracefully if the synaptic coefficients are expressed on a small number of bits [28]. Thus, one can capitalize on this property to use integer arithmetic with limited accuracy. To the best of our knowledge, the second issue, namely the precision required for the training phase itself, has never been investigated in depth.

We first recall the Widrow-Hoff rule :

$$\Delta C_{ij} = \frac{1}{N}\left(\sigma_i^k - v_i^k\right)\sigma_j^k$$

where C_{ij} is the coefficient of the synapse transmitting information from neuron j to neuron i, N is the number of neurons, $\sigma_i{}^k$ is the i-th component of prototype number k, and $v_i{}^k$ is the potential of neuron i when the state of the network is the k-th prototype ; $v_i{}^k$ is given by :

$$v_i^k = \sum_{j=1}^{N} C_{ij}\,\sigma_j^k \quad .$$

It is easy to notice that this rule is local, since each neuron only needs the knowledge of the states of the others, coded on one bit, as would be the case with Hebb's rule.

The matrix obtained by the Widrow-Hoff rule converges to the projection matrix provided it is initialized as the zero matrix.

Speed of convergence of the learning procedure

The first issue to be discussed is the speed of convergence of the iterative rule. When the prototypes are orthogonal, the Widrow-Hoff procedure reduces to Hebb's rule, and it converges after a single presentation of the prototypes. Thus, it may be conjectured that the speed of convergence will decrease with increasing correlations between prototypes, and with a larger number of neurons. Indeed, the number of presentations of the prototypes set increases quadratically in the range of N investigated. Figure 2 shows the number of presentations of the training set required in order to have

$$\left|1 - \sigma_i^k v_i^k\right| < \frac{1}{N} \quad \forall i, \forall k \ ,$$

as a function of the number of neurons. Clearly, the number of presentations for a small number of neurons is manageable. For random uncorrelated patterns, the number of presentations is roughly independent of the number of neurons. For random correlated patterns, the number of presentations increases quadratically in the range investigated. The graphs shown on Figure 2 were obtained for a ratio $\alpha = p/N$ (where p is the number of prototypes) equal to 0.25, and for average overlaps of 0 (uncorrelated patterns) and 0.16 (correlated patterns).

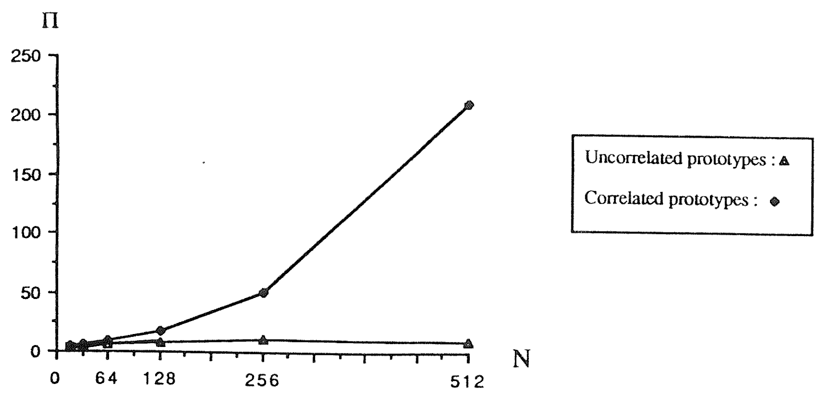


Figure 2

The accuracy required in fixed point arithmetics

We now turn to the influence of the computation with limited accuracy on the learning phase. We assume that the synaptic weights are coded on $\beta = \log_2 M + 1$ bits (including the sign bit), and, for simplicity, we assume that M in a multiple of N. Thus, the Widrow-Hoff rule can be written as :

$$\Delta J_{ij}^k = \frac{1}{N}\left[M\,\sigma_i^k - \sum_{m=1}^{N} J_{im}^{k-1}\sigma_m^k \right]\sigma_j^k \;,$$

where $J_{ij} = M\,C_{ij}$.

The first term within brackets is an integer. The summation itself is also an integer, but the division by N introduces a truncation error. Thus, the Widrow-Hoff procedure with integer arithmetic can be expressed as :

$$\Delta J_{ij}^k = \frac{1}{N}\left[M\,\sigma_i^k - \overline{\sum_{m=1}^{N} J_{im}^{k-1}\sigma_m^k} \right]\sigma_j^k$$

where the bar over an expression means the integer part of it.

We define the integer potential as :

$$u_i = \frac{1}{N}\overline{\sum_{j=1}^{N} J_{ij}\sigma_j} \;.$$

After convergence of the algorithm, all ΔJ_{ij} are equal to zero, and all potentials are equal to $\pm M/N$.

Given the discretization and truncation errors introduced by the use of integer arithmetics, the Widrow-Hoff matrix will not converge exactly to the projection matrix, in general ; specifically, the matrix obtained by the Widrow-Hoff rule will not be strictly symmetrical. The problem is to get a synaptic matrix which conveys satisfactory associative memory properties to the network, while using a limited number of bits, compatible with the goal of minimal silicon area.

There are several ways of assessing quantitatively the "quality" of the associative memory. The first quality criterion stems from the following considerations : in the retrieval phase, the network is initialized into a state which codes for the unknown patterns, and is left to evolve until it reaches a stable state ; therefore, one can compare the quality of two networks by comparing their evolutions when initialized in the same state. Figure 3 shows the proportion of initial states yielding different evolutions with a network built by the projection rule with standard floating-point accuracy, and for a network built by the Widrow-Hoff rule, with coefficients coded on β bits. Each point is an average on 10000 random states and 20 sets of prototypes.

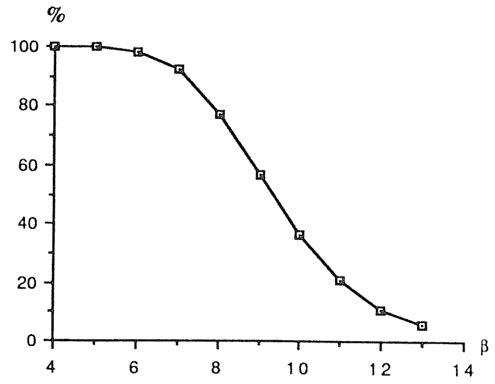


Figure 3

The diagram shows that a coding on 13 bits gives less than 10 % difference between the behaviour of a network of 64 neurons, with a synaptic matrix computed by the projection rule, and the behaviour of the network with a synaptic matrix computed by the Widrow-Hoff rule. (16 uncorrelated prototypes).

It can be argued that the really important issue is not the behavior of the network when initialized in any random state, since the relevant parts of state space are the basins of attraction of the prototypes. Thus, one can also assess the quality of a learning rule, for the task of auto-association, by investigating the size of the basins of attraction of the stored prototypes. In order to determine this size, the following procedure is used : after completion of the learning phase, the state of the network is initialized at a Hamming distance H_i of one of the prototypes, and the network is left to evolve until it reaches a stable sate which is at a Hamming distance H_f of that prototype. A distance H_f equal to zero corresponds to a perfect retrieval. Figure 4 shows the results obtained with the projection rule, on a network of 64 neurons ; 16 prototypes, with equally probable random components +1 or -1, were stored ; the histograms show the distribution of the final normalized distances $h_f = H_f/N$, for two values of the normalized initial Hamming distance $h_i = H_i/N$, equal to 0.125 and 0.25.

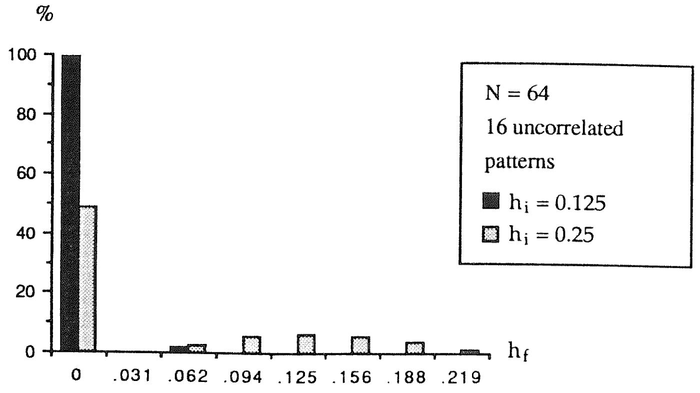
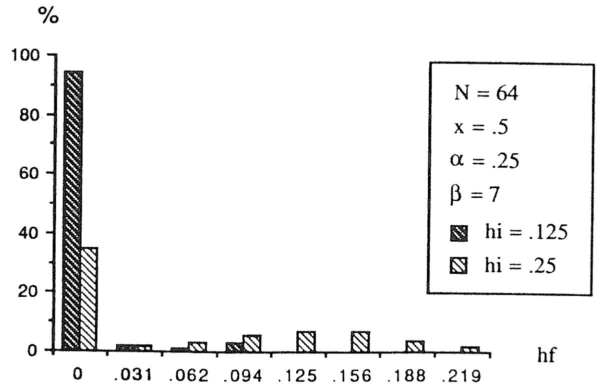


Figure 4

Figure 5 shows the same results obtained with the Widrow-Hoff procedure on integers with $\beta = 7$ bits and $\beta = 9$ bits respectively. Clearly, the auto-associative properties of the synaptic matrices coded on 9 bits are quite similar to those obtained when making use of the projection rule with floating-point accuracy. Although, as stated above, the synaptic matrix is not exactly symmetrical, no cycles have been observed with this choice of parameters.



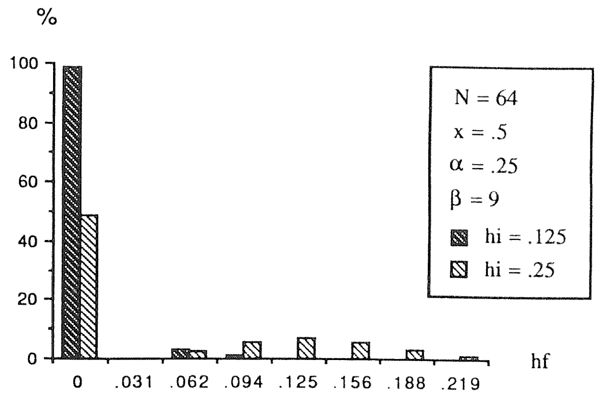


Figure 5

The present study is the first systematic study of the accuracy required to express the synaptic coefficients, for a given network architecture, a given learning rule and a given task. Much effort is still required in order to get a general framework for tackling such problems, which will be of central importance if artificial neural networks ever come into actual industrial use. We now turn to other important implementation issues.

Self identification of successful retrieval

In the context of auto-associative memory applications, it seems desirable to give to the network the ability of signalling by itself whether it has or not succeeded in recognizing a pattern, since a network gives always an output (whether it is a feedforward or a feedback network) for every input stimulus. This intrinsic property of neural networks in general, whatever their architecture, is nearly always overlooked, but it seems clear that it must be taken into account when real applications must be implemented.

We have devised a mechanism which allows the automatic recognition of "good" convergence, producing a binary signal indicating whether the final state reached is a prototype or a spurious state. The vectors presented for storing by the network are divided into two fields : the longer contains the prototype itself, and the other one, called the label, contains the result of the coding of the first through a cyclic error correcting code [29]. This coding uses the primitive polynomial x^6+x+1, which is well adapted to hardware generation by linear feedback shift registers, in good agreement with the general architecture of the network. The whole vector (prototype concatenated to its code) is stored. In the retrieval phase, the attractor on which the network has relaxed is submitted to the same coding: if the information-carrying field is coherent with the label-carrying field, there is a strong probability that this attractor corresponds to a prototype, and vice-versa. Figure 6 shows schematically how this mechanism operates.

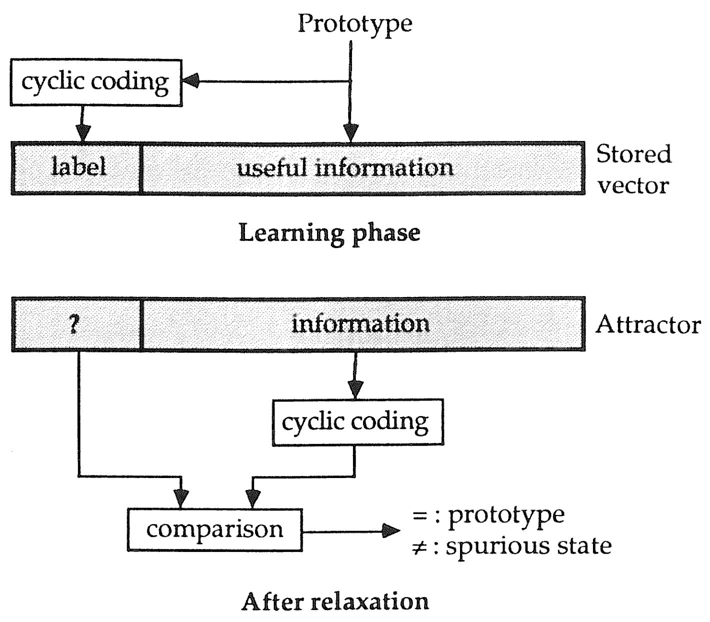


Figure 6 : schematic diagram of attractor labeling, during the learning phase, and after retrieval.

Simulations with random uncorrelated or partially correlated prototypes have shown that the probability of correct identification increases with the length of the label, and that a six-bit label is sufficient to give a good rate of success with a network size of 64 cells. The following diagram (figure 7) illustrates the efficiency of the labeling scheme for identifying prototype convergence, as a function of the number of stored prototypes. As can be expected, the failure rate of the labeling scheme increases both with the initial distance from the studied attractor and with the learning ratio of the network. However, staying within realistic conditions, that is to say an initial distance such that the initial stimulus lies in the basin of attraction of the prototype, the diagram shows that an identification accuracy of about 98% or more can be guaranteed. In addition, simulations showed that most spurious states, wich are known to be thresholded linear combinations of the prototypes, are well identified as such. A similar approach, using a less elaborate coding, was suggested earlier for classification tasks using similar networks [30].

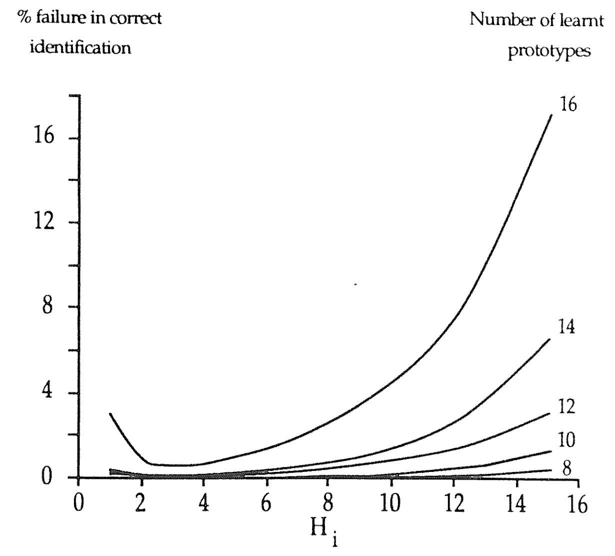


Figure 7 : error rate for identifying a prototype, as a function of initial Hamming distance of the stimulus from the prototype. The parameter is the number of learnt prototypes, for a 64 cells network.

A simple but efficient pseudo-annealing mechanism

Another interesting feature would be the use a probabilistic rule for the neuron decision. It is known that this improves the attractivity of the stored prototypes, but it is also known that it slows down the relaxation of the network [31]. Implementing a real probabilistic threshold function inside each neuron would have added an unacceptable complexity. We chose to implement a very crude mechanism, consisting in flipping a fixed number of neuron states (this number being the remote equivalent of a temperature parameter), the flipped neurons being drawn at random at each updating cycle of the network. The particular random generator required must be able to place a fixed number of bits at random locations in a 64-bit wide vector. It works during the updating cycle of the network, generating a bit-string which is serially xored with the network state vector. This simple mechanism is easy to implement into digital circuitry, and represents only an slight additional overhead to the basic architecture.

Although the added noise is independent of the state of the neuron, simulations have shown that it does lead to a noticeable improvement in prototype attractivity, as shown later, but the number of perturbed neurons must stay low enough. The decision of annealing must be taken only in case of unsuccessful "cold"relaxation of the network, since this situation provides the fastest convergence, and should be tried first. For this purpose, the self-identification mechanism is very useful, since it can be used without supervision to trigger this annealing.

We have made several simulations with various values of α (8, 16 or 24 prototypes for 64 neurons), and measured the percentage of convergence on a prototype, starting from various Hamming distances from this prototype. This is a measure of the prototype attractivity. The experiments have been performed first on cold networks, with the self-identification being used to trigger annealed retries (no more than three) in case of convergence on a spurious attractor. The annealing consisted in flipping randomly 2 or 4 neuron states at each parallel iteration of the network. The results are shown on the following table. The figures are the overall success rate, in percent, of exact retrieval of the prototype (this is a rather stringent criterion), under various learning conditions, and as a function of the pseudo-temperature. It is easy to see that the pseudo-annealing, however coarse, can enhance the performances significantly, two bits being enough in almost all cases, and even a maximum for heavily loaded networks (high values of α). Increasing the number of perturbed neuron states is apt to lead the state of the network out of the basin of attraction of the prototype.

p	H_i	t = 0	t = 2	t = 4
8	16	78.8	93.4	94.2
8	20	36.5	63.8	65.0
16	10	67.3	88.9	89.0
16	14	28.2	53.9	52.6
24	6	38.3	60.6	56.3

Table 1 : effect of pseudo-annealing on the overall retrieval success. p and H_i are respectively the number of learnt prototypes and the initial Hamming distance, and t indicates the number of flipped neurons. The results are the percentage of exact convergence on a prototype.

Using it all together

Even with a network alone, a certain amount of auto-adaptivity is made possible by the self-identification property. For instance, some pattern that is not recognized can be automatically stored for later learning. This property may open the way to associations of networks. For instance, if several networks are wired in parallel (i.e., have the same inputs), each one having learnt various representations of a given symbol (for example the same letter with various positions and shapes), the whole device may operate as a classifier, the output of which is using only the self-identification bit of one network to signal the recognized letter; the particular prototype on which the network has converged being of no interest, is not used as an output. One potential advantage is that learning or relearning is local to each network, and does not involve the whole classifier. Another advantage would be that it might be possible to deal with some invariance difficulties (translation or size), not having to be treated by the learning rules themselves. Simulations are underway on a Connection Machine to assess the properties of various multiple network couplings.

The silicon implementation

Figure 8 shows the fundamentals of the network architecture.

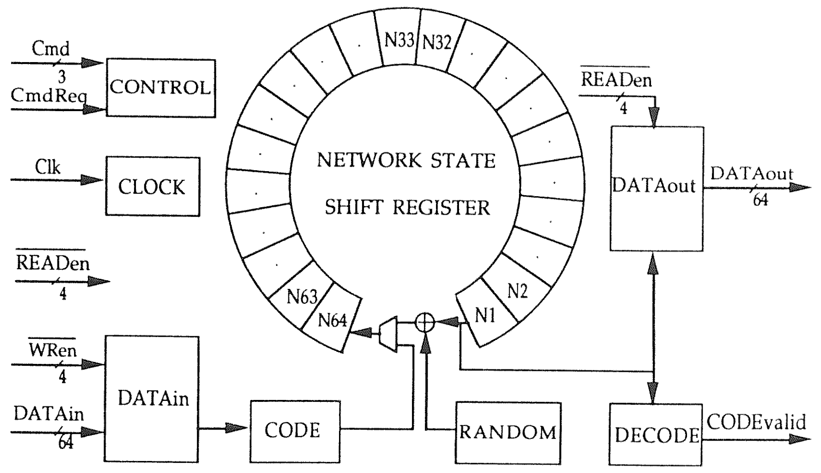


Figure 8

We have made conservative choices, using a state of the art industrial technology, namely two-metal 1.2μm CMOS. The main efforts have been devoted to the design of the synaptic memory, since it represents the main part of the neuron. For a low number of stored words (64 times 9 bits in each neuron memory), using a standard RAM would have been wasteful because of the relatively important amount of ancillary circuitry (read-write, addressing, etc...). Since the synaptic coefficients are always stored or read in a serial fashion, we designed a custom 9 bit wide circular shift register, with a power and area lower than that of conventional silicon-compiler generated RAM, using semi-static tristate circuitry. On the other hand, the other elements of the neuron (control, arithmetic and logic unit) have been designed using a normal standard cell approach, which simplifies the design, and allows parametrization to some extent. It uses fixed-point arithmetic, twelve-bit wide including the sign bit. Provision is made to take into account over- or underflows, which are transformed into saturations, so as to avoid errors.

Convergence detection is rather straightforward. During the relaxation phase, each neuron stores its previous state in a one bit local register, which is xored with the new state bit at the end of an updating cycle. The local results in each neuron are just combinatorially ored serially, giving a low true signal when the whole network is stable. Thus, an extra updating cycle is needed, in which no updating actually takes place, just for asserting the convergence. During the learning phase, since each neuron calculates a synaptic coefficient increment, the six most significant bits are ored to give a local convergence signal (null increment), which is in turn ored serially through all neurons, the same way as in the relaxation stage.

Since a subsequent division trims the lower bits, there is no point in making use of these less significant bits for convergence detection.

Input/output operations are performed in parallel, the necessary serialization of the data taking place inside the chip. It is possible, for use with 16 bits buses such as the MC68000's, to multiplex the 64 bits in four 16 bits blocks : this is the reason why the WRen (write enable) and the READen (read enable) are four-bit wide, each bit enabling the corresponding block while the three others are kept at high impedance. In addition to these I/O control signals, the main external control signals are three command bits (Cmd), a clock, and a command request (CmdReq) entering the chip, a network ready (end of relaxation), an end of learning (all synaptic coefficients stable), a command accept, and finally the prototype identification (CODEvalid), one bit each, plus some state signals, mainly for debugging purposes.

Performances

The overall duration of one updating cycle (in the learning as in the retrieval phase) is the product of the time required to accumulate an increment of the neuron potential (internal neuron computing delay) plus the time needed to shift the state vector by one bit, by the total number of neurons. As we use a 1.2µm technology, we expect a basic cycle of 100ns. This gives an internal computing delay of 75ns, a shift delay (including various housekeeping tasks) of 25ns ; thus, for a complete updating cycle we expect a duration of 6.4 µs, as already shown by functional simulations. In the retrieval phase, if we suppose that for instance four cycles are needed, counting the extra cycle during which convergence is assessed, pattern recognition is performed in approximately 30µs. In the learning phase, the speed is lower for two reasons. The first reason is that the calculation of the coefficients increments needs two cycles : one for the calculation of the potentials, and the second for the calculation of the increments themselves. The second reason is that one has to reach the convergence of the coefficients to stable values corresponding to the exact learning rule : the numbers of iterations needed is roughly proportional to the square of the number of prototypes, as seen earlier. Thus, in the same conditions as above, learning 15 prototypes may take approximately 5 ms, which is probably an absolute maximum, corresponding to a rather high learning ratio. Expressed in the so-called exotic unit of "connections updates per second" (or "cups"), this is equivalent to 2.10^8 cups in our circuit. Whatever unit is used, the overall speed of the circuit is certainly high enough for research purposes, and probably also for most real-life applications.

Using the networks in a real environment

We are currently designing a testbed for the chip, and also for forthcoming multichip structures. It is based on a special prototyping card called MCP™, supplied by Apple Computer for interfacing in a Macintosh II NuBus. This card includes a real-time multitasking system using a local MC68000 processor, 1 Mbyte of RAM and auxiliary circuitry. This setup provides a high level interface between any particular proprietary VLSI chip sitting on the card and the host computer operating system. We intend to use the well known ergonomy of applications running in the Macintosh to drive the network, that is to say supplying global control signals and data, and asynchronously reading the results from the network. The

only function of the host computer is that of an intelligent input/output device for the chip, with no "neural computations" to perform. Several prototyping cards (up to 5) may be inserted in the bus slots, and can work in parallel, allowing a flexible multi-chip architecture. In order to implement larger multi-network architectures, we intend to design a whole external card, able to host more chips, with mixed hard-soft reconfiguration capabilities.

Future integrated networks with many more neurons : will this be possible ?

Sixty four neurons are not enough to cope with some real-life problems, such as image processing. Networks four or even sixteen times larger would be probably more interesting in the future. A 256 cells network is clearly out of reach at present times if we want to keep the same architecture, the same technology, and still design the network in one chip. The area scales roughly like the square of the numbers of neurons, because the synaptic coefficients must be coded on a greater width (two more bits), and the arithmetic unit must also be made wider acordingly. We can design cascadable chips, each containing 16 neurons arranged in an open loop, but each with a 256-word 11-bit wide memory, occupying approximately the same area as our present 64-neuron chip. Sixteen such chips could be arranged using "silicon hybridation" in a four by four array using a surface of about $20cm^2$, which is manageable. The main drawback is the following : since the updating cycle is longer, all the characteristic times (learning as well as relaxation) are increased proportionally to the number of neurons. Apart from finding more efficient architectures, if they exist, the only solution that we see at present is to use smaller technologies, thus increasing the clock rates. Roughly, if the technology is shrinked by a quantity k (k>1), the clock rate should be increased by this factor. For a 0.6µm process, the loss in speed would then be only by a factor of two for the 256 cell network with respect to the 64 cell network, with the important advantage of approximately dividing the silicon area by four, possibly allowing a monolithic implementation. Of course, some important issues such as the connection delays between chips have not yet been addressed in detail. Finally, it is not unlikely that a 1024 cell network could be designed, having an acceptable speed, using technologies below 0.5µm and multichip silicon hybridation, by the mid of the decade.

REFERENCES

[1] L.Personnaz, A.Johannet, G.Dreyfus : Problems and trends in integrated neural networks; in Connectionism in Perspective, R. Pfeifer, Z. Schreter, F. Fogelman-Soulié, eds (Elsevier, 1989).

[2] M.Weinfeld : Neural networks as specialized integrated circuits : an academic exercise or the promise of new machines ? ; International Conference "Neural networks : biological computers or electronic brains", Lyon, 6-8 march 1990.

[3] A.Petrowski, L.Personnaz, G.Dreyfus, C.Girault : Parallel implementations of neural network simulations; in Hypercube and Parallel Computers, A. Verjus, F. André, eds. (North Holland, 1989).

[4] J. J.Hopfield: Neural networks and physical systems with emergent collective computational abilities; Proc. Natl. Acad. Sci. USA, vol. 79, pp. 2554-2558, 1982

[5] J.Alspector, R.B.Allen, V.Hu, S.Satyanarayana : Stochastic learning networks and their electronic implementation. IEEE Conf. on Neural Information Processing Systems, Natural and Synthetic, Denver (USA), 1987; in: Neural Information Processing Systems, Natural and Synthetic, D.Z.Anderson, ed., (American Institute of Physics, 1988).

[6] M.Holler, S.Tam, H.Castro, R.Benson: An electrically trainable artificial neural network with 10240 "floating gate" synapses; Proceedings of International Joint Conference on Neural Networks, Washington D.C., June 1989

[7] A.Moopenn, H.Langenbacher, A.P.Thakoor, S.K.Khanna: A programmable binary synaptic matrix chip for electronic neural networks. IEEE Conf. on Neural Information Processing Systems, Natural and Synthetic, Denver (USA), 1987; in: Neural Information Processing Systems, Natural and Synthetic, D.Z.Anderson, ed., (American Institute of Physics, 1988).

[8] M.Sivilotti,M.R.Emerling, C.Mead: A novel associative memory implemented using collective computation; in: Proc. Chapel Hill Conf. on VLSI, pp. 329-342, (Computer Science Press 1985).

[9] S.Tam, M.Holler, G.Canepa: Neural networks synaptic connections using floating gate non-volatile elements; Neural Networks for Computing, Snowbird, 1988.

[10] D.B.Schwartz, R.E.Howard, J.S.Denker, R.W.Epworth, H.P.Graf, W.Hubbard, L.D.Jackel, B.Straughn, D. M.Tennant : Dynamics of microfabricated electronic neural networks; Appl. Phys. Lett., vol. 50, pp. 1110-1112, 1987.

[11] J.P.Sage, K.Thompson, R.S.Withers : An artificial network integrated circuit based on MNOS/CCD principles; in: Neural networks for computing, J.S.Denker ed., (American Institute of Physics, 1986).

[12] D. B.Schwartz, R. E.Howard : Analog VLSI for adaptive learning; Neural Networks for Computing, Snowbird, 1988.

[13] Y.P.Tsividis, D.Anastassiou : Switched capacitor neural network; Electronic Lett. vol. 23, pp. 958-959, 1987.

[14] U.Rückert, K.Goser : VLSI design of associative networks. International Workshop on VLSI for artificial intelligence, Oxford (GB), July 1988; in : VLSI for artificial intelligence, J.G. Delgado-Frias and W.Moore eds., (Kluwer Academic, 1989).

[15] A.F.Murray, V.W.Smith, Z.F.Butler : Bit-serial neural networks , IEEE Conf. on Neural Information Processing Systems, Natural and Synthetic, Denver (USA), 1987; in: Neural Information Processing Systems, Natural and Synthetic, D.Z.Anderson, ed., American Institute of Physics, 1988

[16] S.Jones, M.Thomaz, K.Sammut : Linear systolic neural network machine; IFIP Workshop on Parallel Architectures on Silicon, Grenoble, 1989.

[17] J.Ouali, G.Saucier, J.Trilhe : A flexible wafer scale network; ICCD Conference, Rye Brook, (USA), September 1989

[18] V.Peiris, G.Columberg, B.Hochet, G.van Ruymbeeke, M.Declercq : A versatile digital building block for fast simulation of neural networks; IFIP Workshop on Parallel Architectures on Silicon, Grenoble, 1989

[19] U.Ramacher : Systolic architectures for neurocomputing; IFIP Workshop on Parallel Architectures on Silicon, Grenoble, 1989

[20] M.Duranton, J.Gobert, N.Mauduit, : A digital VLSI module for neural networks. nEuro'88 conference, Paris, June 1988; in: Neural networks, from models to applications. L.Personnaz and G.Dreyfus, eds., (IDSET, 1989).

[21] S.Y.Kung, J.N.Hwang : Parallel architectures for artificial neural nets; Proc. IEEE International Conf. on Neural Networks, vol. II, pp. 165-172, 1988.

[22] M. Weinfeld : A fully digital CMOS integrated Hopfield network including the learning algorithm; International Workshop on VLSI for Artificial Intelligence, Oxford, 1988, in VLSI for artificial intelligence, J.G. Delgado-Frias and W.Moore eds. (Kluwer Academic, 1989).

[23] S. Knerr, L. Personnaz, G. Dreyfus : Single-layer learning revisited : a stepwise procedure for building and training a neural network; Proc. NATO Advanced Workshop on Neurocomputing, F. Fogelman, J. Hérault, (Springer Verlag, 1990).

[24] M.Weinfeld : Integrated artificial neural networks : components for higher level architectures with new properties; Proc. NATO Advanced Workshop on Neurocomputing, F. Fogelman, J. Hérault, (Springer Verlag, 1990).

[25] L.Personnaz, A.Johannet, G.Dreyfus, M.Weinfeld : Towards a neural network chip: a performance assessment and a simple example; in Neural networks, from Models to Applications, L.Personnaz and G.Dreyfus, eds. (IDSET, Paris, 1989).

[26] L.Personnaz, I.Guyon, G.Dreyfus : Collective computational properties of neural networks : new learning mechanisms; Phys. Rev. A, vol. 34, pp. 4217-4228, 1986.

[27] S.Diederich, M.Opper : Learning of correlated patterns in spin-glass networks by local learning rules; Phys. Rev. Lett., vol. 58, no. 9, pp. 949-952, 1987.

[28] H.Sompolinsky : Neural networks with non linear synapses and a static noise ; Phys.Rev. A, vol. 34, no. 3, pp. 2571-2574, 1986.

[29] W.W.Peterson, E.J.Weldon : Error correcting codes; MIT Press, 1972

[30] I.Guyon, L. Personnaz, P. Siarry, G.Dreyfus : Engineering applications of spin-glass concepts ; Lecture Notes in Physics, vol. 275, J.L.van Hemmen, I. Morgenstern, eds (Springer, 1986)

[31] P.Peretto, J. J.Niez : Stochastic dynamics of neural networks; IEEE Trans. on Systems, Man and Cybernetics, vol. SMC-16, pp. 73-83, 1986.