SPECIFICATION AND IMPLEMENTATION OF A DIGITAL HOPFIELD-TYPE ASSOCIATIVE MEMORY WITH ON-CHIP TRAINING

A. JOHANNET, L. PERSONNAZ, G. DREYFUS
Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris
Laboratoire d'Electronique
10, rue Vauquelin
75005 PARIS - FRANCE

J.-D. GASCUEL, M. WEINFELD
Ecole Polytechnique
Laboratoire d'Informatique
91128 PALAISEAU Cedex- FRANCE

ABSTRACT

This paper addresses the definition of the requirements for the design of a neural network associative memory, with on-chip training, in standard digital CMOS technology. We investigate various learning rules which are integrable in silicon, and we study the associative memory properties of the resulting networks. We also investigate the relationships between the architecture of the circuit and the learning rule, in order to minimize the extra circuitry required for the implementation of training. We describe a 64-neuron associative memory with on-chip training, which has been manufactured, and we outline its future extensions. Beyond the application to the specific circuit described in the paper, the general methodology for determining the accuracy requirements can be applied to other circuits and to other auto-associative memory architectures.

1. INTRODUCTION

The present paper describes the specification and the silicon integration of a Hopfield neural network designed (i) to operate as an associative memory, and (ii) to be trainable on-line, i.e. without a host computer; the latter point is especially important, since *adaptive* neural networks, i.e. neural networks which undergo continual training, are gaining popularity; although the network which is presented here is not intended for adaptive operation, its design is an interesting starting point in that direction. In addition, integrating training on

the chip gives much more autonomy, and opens a way towards the design and operation of associations of networks.

When designing a neural network in digital technology, the choice of the number of bits for encoding the synaptic weights is of central importance, since it involves a tradeoff between the efficiency of the network after training, and the circuit area. When training is performed on the chip itself, an additional problem arises, namely, the accuracy of the arithmetics used for computing the coefficients. The present paper reports a detailed investigation of this problem, and describes the specifications which resulted from this study.

The first part of the paper is devoted to a short presentation of the Hopfield net and of its operation as an associative memory. Next, we investigate specific iterative learning rules (Widrow-Hoff rule, Perceptron rule, Minover rule) whose silicon implementations do not require much overhead in terms of additional connections and/or additional arithmetic operators; this is a crucial point for the implementation of on-chip training. The accuracy of the synaptic weights and of the arithmetics used for training is discussed in detail. The last section describes the circuit which has been designed and manufactured.

2. HOPFIELD-TYPE NETWORK AND ASSOCIATIVE MEMORY

2.1 - Neural network model:

We consider a Hopfield-type network (without external inputs) consisting of an assembly of n fully connected neurons ; at time t, each neuron computes a weighted sum of its inputs and adjusts its output at time $t+\tau$ according to:

$$s_i(t+\tau) = f(\Sigma_j C_{ij} s_j(t)) = f(v_i(t)) \qquad (1)$$

where $s_i(t)$ is the state of the neuron i at time t,

$v_i(t)$ is the potential of the neuron i at time t,

C_{ij} is the synaptic weight of neuron i receiving information from neuron j,

f is the activation function of the neuron.

The dynamics of the network is fully defined by the values of the synaptic weights C_{ij} and by function f.

In the following, we consider the case of binary neurons (whose activation function is the signum function), whose state is denoted by $\sigma_i(t)$.

If the neurons are updated in a parallel, synchronous way, the decision-making rule of the network can be formulated in matrix form :

$$\underline{\sigma}(t+\tau) = \text{sign}\,(C\,\underline{\sigma}(t)) = \text{sign}\,(\underline{v}(t)) = [\text{sign}(v_1(t)), ..., \text{sign}(v_n(t))]^T \qquad (2)$$

where $\underline{\sigma}(t)$ is the n-vector state $\{-1, +1\}^n$ of the network at time t, and C is the (n,n) synaptic matrix.

2.2 - Auto-associative memory:

An associative memory is a device which is trained to perform associations between source and target items; once training is completed, the memory is able to retrieve the target information when presented with an incomplete or distorted version of the source information. When the source and target items are identical, the memory is termed *auto*-associative. In general, training is performed non-adaptively, i.e. the learning phase is completely separated from the retrieval phase.

The associative memory properties of Hopfield networks, based on the existence of attractor fixed points in state space, have been extensively studied [1-3]. The problem of designing an auto-associative memory can be summarized as follows.

Consider a given set of p binary prototype vectors $\{\underline{\sigma}^k, k=1 \text{ to } p\}$, coding for a set of items, and a network as defined above ; the problem consists in finding a matrix C so that (i) the patterns $\{\underline{\sigma}^k\}$ are attractor fixed points ; (ii) no cycle attractors exist. The stability of the prototypes is expressed by the n.p inequalities :

$$v_i^k \sigma_i^k > 0, \text{ for } 1 = 1 \text{ to } n, \text{ and } k = 1 \text{ to } p. \qquad (3)$$

A solution is obtained by solving the n.p equations :

$$v_i^k \sigma_i^k = a_i^k, \text{ with } a_{ik}>0, \text{ for } i = 1 \text{ to } n, \text{ and } k = 1 \text{ to } p. \qquad (4)$$

The parameters a_i^k determine the stability margin of the prototypes.

If we impose all parameters a_i^k to be equal to 1, the orthogonal projection matrix of the state space into the subspace spanned by the prototype vectors is a solution of the np equations (4). It has been shown [4] that this matrix gives satisfactory auto-asociative properties to the neural network. In this case, the network complies with the previous requirements even when correlated patterns are learnt, within the limit p<n. Additional (sometimes termed "spurious") attractors appear, whose basins of attraction are related to ambiguous patterns.

Various methods are available for computing the projection matrix: it can be computed either directly, or iteratively. The direct matrix computation requires a finite number of operations, which is $O(p^3)$; iterative procedures involve several presentations of the prototypes, and the total number of operations is infinite, in principle. Therefore, direct computation might seem preferable; however, the total number of operations is not the only criterion, since ease of silicon implementation, in terms of circuit regularity and transistor count, is of central importance. We show in this paper that iterative methods are preferable in the present case.

The projection matrix is not the only synaptic matrix which complies with the above requirements for auto-associative operation of Hopfield nets; in the present paper, we use it as a reference, because the properties of the networks whose synaptic matrix is the projection matrix are well known and are satisfactory.

3. IMPLEMENTATION OF THE LEARNING RULES

The integration of three iterative learning rules, whereby the synaptic weights are modified at each presentation of the prototpyes, has been investigated: the Widrow-Hoff rule [5], the Perceptron rule [6], and the Minover rule [7]. These rules are local because the computation of the variation of the synaptic weights of a neuron requires arithmetic operations whose operands are the potential of that neuron and the state of the network; in the case of the architecture of the circuit described in Section 4, the components of the state vector are made available to the neuron in sequence, so that no extra data flow is required.

As mentioned above, the main issue in the integration of the training algorithm is the word size of the synaptic coefficients, and 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 encoded with a small number of bits. The present paper shows how one can capitalize on this property to use integer arithmetics with limited accuracy. The accuracy required to implement the Widrow-Hoff learning rule so as to give satisfactory retrieval properties to the network after training is determined in two steps : (i) formulation of the rules with integer arithmetics; (ii) investigation of the network behaviour during the learning and retrieval phases as a function of the number of bits used for encoding the weights, and as a function of the accuracy of the computations. The same analysis has been carried out for the Perceptron and the Minover rule; it is briefly summarized here.

3.1 - Widrow-Hoff rule:

The Widrow-Hoff rule is a gradient-type learning rule [5] which allows the iterative computation of the projection matrix [8]; the procedure is as follows :

- matrix C is initialized to 0,
- when the prototype sk is presented to the network at iteration k, matrix Ck-1 (computed at iteration k-1) is available, so that each neuron-processor i computes :
  $\Delta C_{ij}^k = (1/n)(\sigma_i^k - v_i^k)\sigma_j^k$ for j=1 to n, (5)
  with $v_i^k = \sum_j C_{ij}^{k-1}\sigma_j^k$,
- all prototypes are presented several times in sequence until convergence, i.e. until:
  $|\sigma_i^k - v_i^k| < \varepsilon$, ε being an arbitrarily small quantity, for all i and all k ($a_i^k=1$ for all i and all k).

In this section, we first investigate the speed of convergence of the Widrow-Hoff rule as a function of the number of neurons and of the mean correlation between prototypes, and we determine the dynamic range of the coefficients and potentials during training; as a subsequent step towards a silicon implementation, we reformulate the learning rule with integer arithmetics, and show that this formulation introduces an integer multiplicative parameter m; we determine, as a function of m, the number of bits which are necessary for

encoding the coefficients and the potentials in order to preserve the convergence of the training procedure. Finally, we investigate the auto-associative properties of the resulting networks; this leads to the final determination of the parameter m, hence the determination of the number of bits necessary for encoding the coeficients and potentials.

3.1.1 - Speed of convergence:

The Widrow-Hoff procedure produces a sequence of synaptic matrices which converges to the projection matrix, when the prototypes are presented in sequence to the network, even if they are correlated [8]. The speed of convergence can be estimated by observing the evolution of the difference between the synaptic matrix and the projection matrix, or, more interestingly, the evolution of the potentials. When the prototypes are mutually orthogonal (zero correlation between prototypes), the Widrow-Hoff rule yields the projection matrix after a single presentation of the set of prototypes. Thus, it may be conjectured that the required number of presentations of the training set will increase with increasing correlations between prototypes, and with an increasing number of neurons: this fact is exemplified by Figure 1, which shows the number of presentations Π of the training set that is required in order that:

$$|v_i^k - \sigma_i^k| \leq (1/n) \qquad i=1 \text{ to } n,\ k=1 \text{ to } p \qquad (8)$$

as a function of the number n of neurons, for random patterns and two different values of the correlations. The components of the prototype vectors are generated with the following law: Probability(σ_i^k=+1)=x, Probability(σ_i^k=-1)=1-x, with 0<x<1; in the following, the correlations between the resulting patterns is expressed by the mean value of the cosine of the angles between prototypes (also termed "mean overlap"), denoted by <cos>; uncorrelated patterns, i.e. patterns whose mean overlap is equal to zero (<cos>=0), are obtained for x=1/2.

For uncorrelated patterns, the number of presentations is roughly independent of the number of neurons. For correlated patterns, the number of presentations of the training set increases quasi-quadratically in the range of n investigated (8≤n≤512). The graphs shown on Figure 1 were obtained for a ratio α=p/n equal to 0.25, and for mean overlaps of 0 (uncorrelated patterns) and 0.16 (correlated patterns).

3.1.2 - Magnitude of the coefficients and of the potentials during training:

We first consider the projection matrix. One can easily prove that:

$$|C_{ij}| \leq 1 \qquad \text{for } i=1 \text{ to } n,\ j=1 \text{ to } n,$$

and $$|v_i^k| = 1 \qquad \text{for } i=1 \text{ to } n,\ k=1 \text{ to } p.$$

These conditions, however, are not necessarily valid *during* training by the Widrow-Hoff rule; therefore, we have observed the evolution of the potentials and of the coefficients during training. Simulations have shown that the values of the coefficients never exceed 1.

Figure 2 shows the evolution of the average value of the diagonal coefficients (which are dominant) during training (N_p is the number of prototype vectors which have been presented), for n = 64, and 16 prototypes. It converges to the value corresponding to the projection matrix (p/n). The behaviour is similar for correlated and for uncorrelated prototypes. The curves shown on Figure 2 are averaged over 100 different sets of prototypes.

The evolution of the magnitudes of the potentials during learning is shown on Figure 3 with the same parameters as in Figure 2. The potentials converge to the desired values (±1) corresponding to the projection matrix. During learning, the potentials may exceed their final value ; the magnitude of the overshoot depends on the correlation between the prototypes. The curves shown on Figure 3 are averaged over 100 different sets of prototypes and over the potentials of all neurons. In the range of parameters investigated, the overshoot of the potentials may reach 150%: therefore, provision must be made, in the encoding of the potentials, for variations of the latter in the range -1.5 to +1.5 at least.

3.1.3 - Widrow-Hoff rule with integer arithmetics:

The Widrow-Hoff procedure can be expressed with integer quantities by multiplying the synaptic increment (hence the synaptic matrix) by a integer parameter m, multiple of n. We denote by J the Widrow-Hoff integer matrix and by u the integer potential; if m>>n, we have: $J_{ij} \approx m\, C_{ij}$ (for all i and j), $\Delta J_{ij} \approx m\, \Delta C_{ij}$, and $u_i \approx (m/n)\, v_i$.

Denoting the saturation function resulting from the computation with integers by S, denoting the truncation function by T, and denoting the integer potential by u, the Widrow-Hoff rule can be rewritten as :

$$\Delta J_{ij}^k = S[(m/n)\, \sigma_i^k - u_i^k]\, \sigma_j^k,$$

with $u_i^k = T\{(1/n)\, S[\Sigma_j\, J_{ij}^{k-1}\, \sigma_j^k]\}$.

Note that the result of the summation $S[\Sigma_j\, J_{ij}^{k-1}\, \sigma_j^k]$ depends on the implementation, in particular on the order in which the saturations occur.

The sum itself is an integer, but the division by n introduces a roundoff error.

After convergence of the algorithm, all ΔJ_{ij} are equal to zero, and all potentials are equal to ±m/n. Given the discretization and saturation errors introduced by the use of integer arithmetics, the Widrow-Hoff matrix computed with integer arithmetics (hereinafter referred to as the Integer Widrow-Hoff matrix or IWH matrix) will not converge exactly to the projection matrix, in general; specifically, the IWH matrix will not be strictly symmetrical, whereas the projection matrix is symmetrical. Therefore, we are faced with the following problem: how 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.

3.1.4 - Encoding of the synaptic coefficients and of the potentials:

The IWH training rule has been investigated for n=32, 64, 128. Two cases must be considered:

- if m>>n, the IWH network has the same behaviour as the Projection network. Thus, since the coefficients do not exceed the value of 1 for the projection matrix, and since they can take on m times this value with integer coding, we encode them on $\beta = \log_2 m+1$ bits (including the sign bit) to prevent saturation. In this case, the synaptic matrix J converges to m.C.
- if m≈n, saturations and truncations occur frequently during training, thus introducing strong non-linearities which may prevent the algorithm from converging. Two types of undesirable behaviours may occur : cycles or divergence of the values of the coefficients. Extensive simulations have shown that, for random correlated or uncorrelated patterns, and for ratios p/n lower than 0.3, the learning procedure converges provided that :
  - the coefficients are encoded on $\beta_J = \log_2 m +1$ bits,
  - the potentials are encoded on $\beta_u = \beta_J + 2$ bits,

  with m > n.

  The second case (m>≈n) must be considered in order to implement minimum-size coefficients on the circuit.

This implementation of the IWH rule and potentials shows two interesting properties: (i) for a network with a given number of neurons n, *the number of bits necessary to encode the potentials and the synaptic weights is determined by a single parameter, m*; (ii) *the training procedure converges even with a reduced arithmetic accuracy (m≈n)*, but it does not converge exactly to the projection matrix: the next section is devoted to the investigation of the retrieval properties of IWH networks, which allows the final determination of the parameter m, hence the determination of the number of bits necessary for encoding the coefficients and potentials.

3.1.5 - Network behaviour during the retrieval phase:

In the present section, the auto-associative properties of the IWH network are compared with the well-known properties of the network whose matrix is computed with the Projection rule in standard floating-point arithmetics (hereinafter referred to as the Projection net). The IWH matrix is computed for various values of m/n and, as mentioned above:

- the coefficients J_{ij} are encoded on $\beta_J = \log_2 m + 1$ bits,
- the potentials are encoded on $\beta_u = \beta_J + 2$ bits.

The coefficients of both networks are computed with the same set of prototypes, for $0.25<p/n<0.33$, and n=32, 64, 128.

The performance of the associative memory may be assessed in various ways. The first performance criterion stems from the following considerations : during the retrieval phase, the network is initialized into a state which codes for an unknown pattern, and is left to evolve until it reaches a stable state or a cycle. The performances of the networks can be compared by observing their evolutions when initialized in the same state. Figure 4 shows the proportion of initial states yielding different evolutions with the Projection network and with the IWH network, for various values of β_J, with n=64, and p=16 uncorrelated patterns. Each point is an average over 10,000 random states and 20 sets of prototypes. The diagram shows that the difference between the networks is smaller than 10 % provided the coefficients are encoded on 13 bits or more.

It can be argued that the really important issue is not the behaviour 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 performance of a network, for the task of auto-association, by estimating the size of the basins of attraction of the stored prototypes. 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 state which is at a Hamming distance H_f of that prototype. A distance H_f equal to zero corresponds to perfect retrieval. Figure 5 shows the results obtained with a Projection network of 64 neurons with 16 uncorrelated prototypes; 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. These results will be used below for comparisons with the properties of other networks.

As a first comparison, Figure 6 shows the results obtained for the IWH network with β_J = 7 bits (m/n = 1) and β_J = 9 bits (m/n = 4) respectively, with the same prototypes as before. When training is performed with 7-bit arithmetics, the behaviour of the IWH network is similar to that of the Projection network, illustrated on Figure 5. This result can also be compared with the result shown on Figure 4: clearly, the required accuracy is lower when we consider the neighborhood of the prototypes than when we take into account the whole state space.

The behaviour of the network has also been investigated with correlated prototypes; Figure 7 shows the histograms obtained with correlated prototypes, whose mean overlap lies in the range 0 to 0.64. The behaviours of the IWH net and of the Projection net are similar provided that one has $\langle\cos\rangle < 0.3$; otherwise, the behaviours differ by more than 10%. These results are obtained with networks of 64 neurons, α=0.3, β_J = 9 (m/n=4), and averaged over five sets of prototypes.

Clearly, for 64 neurons, the auto-associative properties of the IWH network whose synaptic matrix is encoded on 9 bits are quite similar to those of the Projection network

computed with standard floating-point arithmetics. Although the synaptic matrix is not exactly symmetrical, no cycle has been observed with this choice of parameters.
The required number of bits for the coefficients has also been investigated as a function of the number of neurons. The same comparisons as above show that the required coefficient size increases with the number of neurons as follows :

$$\beta_J = \log_2 n + 3,$$

with $m/n = 4$.

To summarize, the behaviour of the IWH network approaches the behaviour of the Projection network when the accuracy increases: first, the prototype vectors are stabilized; then they become attractors; finally, the behaviours of both networks become identical. These results are valid for random correlated prototypes. They make the digital implementation of a Hopfield network and its learning rule on a single chip possible: *for a 64-neuron network with on-chip training - whose implementation will be described below - satisfactory auto-associative properties are obtained if the coefficients are encoded on β_J = 9 bits and the potentials on β_u=11 bits.*

3.2 - Perceptron rule and Minover rule:
The Widrow-Hoff rule requires the computation of the quantity $\sigma_i^k - v_i^k$ for the evaluation of the coefficient increments ; in contrast, the Perceptron rule and the Minover rule involve increments which have fixed values. In the present section, we present the main results of an investigation of these rules, in the same spirit as the above investigation of the Widrow-Hoff rule: we first reformulated the rules with integers, and subsequently investigated the storage and retrieval properties as functions of the encoding of the coefficients and potentials is investigated.
We have investigated the behaviour of networks trained by the improved version of the Perceptron rule introduced by Diederich and Opper [8]. The rule can easily be reformulated for application to integer quantities. If we denote by J the integer synaptic matrix, and by *a* the threshold, the rule becomes :
- matrix J is initialized to 0,
- at the presentation of prototype $\underline{\sigma}^k$, for each neuron i :

if $v_i^k \sigma_i^k \leq a$: $\Delta J_{ij} = \sigma_i^k \sigma_j^k$, for j=1 to n.
if $v_i^k \sigma_i^k > a$: $\Delta J_{ij} = 0$, for j=1 to n,

where *a* is a positive threshold. The procedure stops when the condition $v_i^k \sigma_i^k > a$ is true for all prototypes and all neurons. Since the modifications of the coefficients are integer quantities, the matrix is automatically encoded with integers. Therefore, no roundoff error is introduced during learning.

The Minover rule can also be reformulated for application to integer quantities. If we denote by J the integer synaptic matrix, and by *a* a positive threshold value, the rule becomes:

- matrix J is initialized to zero
- for each neuron i:

find the prototype $\underline{\sigma}^{\mu}$ which satisfies :

$(\underline{J}_i \underline{\sigma}^{\mu})\ \sigma_i^{\mu} = \min \{(\underline{J}_i \underline{\sigma}^{k})\ \sigma_i^{k}\}$, k = 1 to p, $\underline{J}_i$ being row i of the synaptic matrix,

modify the synaptic coefficients of neuron i :

if $\sigma_i^{\mu}\ v_i^{\mu} \leq a$: $\Delta J_{ij} = (\sigma_i^{\mu}\ \sigma_j^{\mu})$, j = 1 to n,

if $\sigma_i^{\mu}\ v_i^{\mu} > a$: $\Delta J_{ij} = 0$.

The matrix is automatically encoded with integers ; the algorithm terminates when all $\sigma_i^{\mu}\ v_i^{\mu}$ are above the threshold.

The properties of networks of 64 neurons, trained with either rule with *a*=256. It was found that, in both cases, the coefficients have to be encoded with 9 bits and the potentials with 11 bits (both including sign) in order to avoid saturation. Interestingly, the values are the same as the values determined for the Widrow- Hoff and Perceptron rules. The attractivity of correlated prototypes stored by the Minover rule is slightly higher than the attractivity of the same prototypes stored in an IWH network, whereas the attractivity of correlated prototypes stored with the Perceptron rule is not significantly different from that of the same prototypes stored in an IWH network

3.4 - Conclusion:

The present study is the first systematic investigation of the accuracy required to compute and to express the synaptic coefficients, for a given network architecture and a given task. Three training procedures have been investigated : the Widrow-Hoff rule, the Perceptron rule with a threshold *a*, and the Minover rule. *All three rules were shown to require the same accuracy for the coefficients and the potentials, namely, 9 and 11 bits respectively for a network of 64 neurons, and to give approximately the same attractivity to correlated prototypes*. The Minover rule requires a special procedure for choosing the prototype with minimal overlap, which makes its implementation more complex. The Perceptron rule is the simplest rule for implementation, but the properties of the resulting synaptic matrix are not as well known as those of the projection matrix, which is obtained by application of the Widrow-Hoff rule. Since the implementation complexity of the latter is not much higher than the implementation complexity of the Perceptron rule, the Widrow-Hoff rule was deemed to be the best choice.

4. IMPLEMENTATION AND FUTURE EXTENSIONS

4.1 - Basic choices:

Most published designs [9] describe analog implementations of fully connected networks. However, these circuits exhibit the problems inherent to analog circuitry, especially in feedback systems, such as instability and noise sensitivity. Still more important, the above designs do not allow easy adjustments of the synaptic weights : they have limited learning ability. In addition, it might seem desirable to implement neural networks in standard proven technologies; these facts led us to the design of a fully digital custom circuit with on-chip learning.

Various digital architectures have been discussed previously in the literature [9]; the main differences between these designs stem from various tradeoffs between silicon area and computation time. Since full parallelism would require n^2 operators, it cannot be achieved efficiently in digital technology with a significant amount of neurons, given the current area limitations; the preferred organization is that of a systolic ring architecture, requiring only n neuron-processors operating in parallel [10]. Thus, the speedup which can be achieved is only O(n), whereas it would be $O(n^2)$ with a fully parallel architecture.

In that architecture, one neuron-processor i has a memory of n synaptic coefficients (C_{ij}, j=1,n). It computes its potential in n successive multiplications and additions. Figure 9 shows the general organization of the network in a ring architecture: to initiate the computation, each neuron-processor receives one component of the state vector. During retrieval, the components of the state vector are shifted in the ring, so that each neuron-

processor receives the corresponding component to compute the product $C_{ij}\,\sigma_i$, and to update the partial sum. All information transfers are performed in n steps with n links between n neuron processors.

This architecture is particularly interesting if binary neurons are implemented: in such a case, the synapse operation is reduced to an elementary boolean operation, and one single bit is transferred between processors. Therefore, the circuit area depends basically on the memory requirements for the synaptic matrix, which grows as n^2: a special effort has to be devoted to reducing the corresponding silicon area, by reducing the number of bits of the synaptic weights.

In addition to on-chip training, two special features have been implemented; they will be briefly described in the following sections.

4.2 - Automatic detection of spurious states:

In addition to its auto-associative memory properties, the network has the ability of signalling whether it has succeeded or not in recognizing a pattern (since a network always gives an output for any input stimulus): a binary signal is produced, indicating whether the final state is a stored pattern or a spurious state. The vectors presented to the network for training are divided into two fields: the longer contains the pattern itself, and the smaller, six-bit wide, field (called the *label*), contains the result of the coding of the first field through a cyclic error correcting code. The whole vector (pattern concatenated to its label) is stored. In the retrieval phase, the attractor to which the network has converged is submitted to the same coding: if the information-carrying field is consistent with the label-carrying field, this is a strong indication (≈98% confidence with a 6-bit label) that this attractor is a stored pattern, and vice-versa.

4.3 - A coarse "annealing" mechanism for improving the success of retrieval:

To improve the quality of the retrieval, provision has been made for adding a limited amount of noise to the input pattern if the network fails to converge to a prototype. This operation can be repeated several times, either until success, or until non-recognition is assessed. The circuitry uses two coupled 6- and 11-bit linear feedback shift registers for generating a 64-bit string with a fixed number of bits (1 or 2) set to 1 in random positions. This perturbation vector is xored with the input pattern if necessary, while the linear systolic register is loaded prior to the retrieval phase. This feature can be triggered automatically by the output of the spurious state detection described in the previous section, and has been shown to allow up to 25 to 30% improvement to the overall recognition capabilities of the network. Both this mechanism and the spurious state detection are to be described in more detail in a further paper.

4.4 - VLSI implementation:

4.4.1 General architecture choices:

As described below, the synaptic memory has a particular mode of operation, and has been custom-designed to save space. The rest of the neuron (control, arithmetic and logic unit) has been designed with standard cells. Figure 10 shows the block diagram of the circuit.

Since the neuron states are coded on one bit only, the operating part is very simple: it uses fixed-point signed arithmetics. The only operations implemented are addition, complementation and arithmetic shift. Provision is made to take into account addition over or underflow, which is transformed into saturations, in order to avoid errors.

For convergence detection, 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, in the retrieval phase. The local results in each neuron are just combinatorially ored, giving a low true signal when the whole network is stable. Thus, an extra cycle is required for asserting the convergence, where no updating actually takes place, but during which the serial state vector is converted into a parallel output. In the training phase, since each neuron computes a synaptic coefficient increment, the six most significant bits are ored to give a local convergence signal (null increment), which is in turn ored through all neurons, the same way as in the retrieval phase (it is not necessary to use the other less significant bits since a subsequent division trims them anyway).

Input/output operations are performed in parallel, the serialization of the data taking place inside the chip. For use with external 16 bits buses, the 64 bits can be multiplexed into four 16-bit 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 other three signals are kept in a high-impedance state. In addition to these I/O control signals, there are several main external control signals, plus some internal state signals, mainly for debugging purposes.

4.4.2 - The commands:

Eight commands are implemented; when properly chained, they drive all the operations, in the retrieval phase as well as in the learning phase. They are summarized in Table 1.

Some signals are provided to the output, for interfacing or debugging purposes :

φ_1, φ_2 : non overlapping internal clocks,

C_0 and C_{63} : markers of the first and last cycle in a macro-cycle,

*busy_* : the state of the command FIFO; the circuit can accept a new command when it is set,

outputValid : indicates that the validity of the results at the end of a macro-cycle,

Cvg : bit indicating the convergence. In the retrieval phase, this means that an attractor has been reached. In the learning phase, it indicates that the synaptic matrix is stable after presentation of a prototype,

codeValid : indicates the result of the spurious state detection.

A *Rotate* or *Reset* command can occur alone, and at any time", but the other commands are part of standard sequences, corresponding to the various modes of operation. During their execution, the command FIFO guarantees a 100% duty cycle to the circuit. In the case of excessive delay in loading this FIFO, the necessary amount of *Rotate* commands are automatically inserted so as to refresh the memory.

The retrieval sequence is made of a *Shift* to load the state register, followed by as many *Evol* as necessary for the *Cvg* to be asserted.

When learning, the sequence *Shift, Learn1, Learn2* is used with each presentation of one prototype. When all p prototypes have been presented, if *Cvg* has been asserted for each of them, the learning phase is terminated.

4.4.3 - The elementary computation cycle:

The external clock frequency being 25 MHz, the duration of an internal cycle is 80 ns. The memory is fully synchronous, using a biphase clock φ_1 and φ_2. On the rising edge of φ_1, the data is valid at the memory output. In the case of a write operation, the data must be ready on the falling edge of φ_2. For a read-calculate-write cycle, the available time is the interval between these two edges (approximately 75 ns).

The commands broadcast to all neurons are thus distributed so as to be valid at least during this interval. Three signals C_0, C_{63} and C_{58to63} allow to specify the commands on a macro-cycle (64 basic cycles). They all change state on the rising edge of φ_1. C_0 marks the first cycle of a macro-cycle, C_{63} marks the last, C_{58to63} marks the last six cycles, corresponding to the part of the state vector which contains the cyclic redundancy code bits required for spurious state detection.

The state vector circulates through the whole circuit : parallel-serial input register, cyclic code insertion, the 64 neurons, random generator, serial-parallel output register. It is implemented by a shift register synchronized on the falling edge of φ_2.

4.4.4 - The representation of numbers:

The internal representation in the ALU of each neuron uses 12-bit fixed point, signed, two’s complement arithmetics. The MSB carries the sign, the two following bits code for the integer part, the 9 remaining bits code for the fractional part. Thus, the numbers are in the range -4.000 to +3.998. In case of over- or underflow resulting from an addition, a saturation mechanism replaces the erroneous result by the appropriate boundary value. In the memory, we only implement the requested minimal precision of 9 bits, thus using

boundary values of -0.500 and +0.498. The same saturation mechanism is used in the learning phase to avoid storing wrong coefficients.

4.4.5 - The neuron memory:

The 9-bit wide memory of the network is distributed in n lines of the synaptic matrix, each one being local to one neuron. There are two kinds of memory access : read-only during the retrieval or the first learning phase, read-write during the second learning phase. During each of these operations, a neuron *k* accesses its data following always the same cyclic order :

k, k+1, k+2, ..., n, 1,... k-1 .

Thus, in the design of the memory, it seemed appropriate take advantage of this property to avoid unnecessary addressing overhead. We have compared several solutions, including a classical RAM structure: a simple shift register turned out to be the most compact and the less demanding in terms of power consumption. Of course, these conclusions might not be valid in the case of significantly larger memories. The memory is thus implemented in the form of nine identical one-bit shift registers using dynamic tristate circuitry. Since the synaptic coefficients are computed locally, they are automatically stored at the right place, where they can be retrieved without addressing: this is an additional advantage of in-circuit learning.

4.4.6 - The neuron ALU:

As mentioned above, the ALU has to perform the computation of the following quantities :

In the retrieval phase :

$$u_i := \sum_{j=1}^{n} J_{ij}\sigma_j \quad .$$

In learning phase 1 :

$$A := \sigma - u \quad .$$

In learning phase 2 :

$$J_{ij} := J_{ij} + \frac{1}{n} A_i \ \sigma_j \quad .$$

Figure 11 shows the simplified schematics of one neuron. The ALU consists of a 12-bit parallel adder associated with a 12-bit accumulator, and of a divider by n. Since n is an integer power of two, this divider is just an arithmetic right shifter.

The ALU can perform the following operations :

Acc ← 0 - Mem

Acc ← 0 + Mem

Acc ← Acc + Mem

Acc ← Acc -Mem

Acc ← 1-Mem

Acc ← -1 + Mem
Mem ← Mem + 1/n Acc
Mem ← Mem - 1/n Acc

where *Acc* and *Mem* are the contents of the accumulator and of the current memory cell respectively.

4.5 - Fabrication data:

The circuit has been manufactured using an industrial 1.2 μm CMOS double metal technology (European Silicon Structures ECDP12 process). The design of the circuit has been performed on a workstation, using the Cadence software. The final circuit totals 420,000 transistors; memory blocks are custom designed, and operating parts use standard cells provided by the founder. Figure 12 shows a micrograph of the die, which has an overall area of 1 cm^2. The area of each neuron is approximately equally divided between the memory of the synaptic coefficients and the local ALU and controls, although the memory uses about three times as many transistors as the ALU. The die is encapsulated in a PGA 176 chip carrier.

4.6 - Performances:

The overall duration of one updating cycle (macro-cycle), in the training as in the retrieval phase, is the product of the time required to accumulate an increment of the neuron potential (internal neuron computation delay) plus the time needed to shift the state vector by one bit, multiplied by the total number of neurons. With the basic cycle of 80 ns, this leads to a complete macro-cycle in 5.4 μs. In the retrieval phase, let us suppose that three cycles are needed (this number depends on the initial Hamming distance of the stimulus to the corresponding stored pattern, and is generally smaller than three). Counting the extra macro-cycle during which convergence is assessed, and the initial loading macro-cycle, one pattern recognition is performed in less than 30 μs. In the training phase, the speed is limited by two factors: (i) the computation of the coefficient increments requires two cycles, as explained above; (ii) one has to reach the convergence of the coefficients to stable values corresponding to the exact learning rule: the number of presentations required increases with the number of stored patterns. Thus, in the same conditions as above, learning 15 moderately correlated patterns ($p/n \approx 0.25$) takes approximately 3 ms (the exact time depending on the correlation between prototypes). This is approximately 1500 times faster than an MC68030 workstation, and 6 times faster than a CM2 Connection Machine™, with equivalent operating parameters.

These speeds (tens of microseconds in retrieval, some milliseconds in training) are obviously high enough for many real-time operations.

4.7 - A flexible simulator:

The circuits are wired on Apple Macintosh™ MCP NuBus™ interface cards, to be configured into various multi-network structures by software. Several such cards can be used, possibly in different microcomputers, as long as they are linked by the standard LocalTalk™ network. This very convenient way of using many circuits for experimentation, under control of a simple man-machine interface, helps assess various higher level architectures, which would otherwise be impossible to simulate just by software, except on costly and bulky superparallel mainframes.

4.8 - Future extensions:

The scaling rules for this kind of architecture are straightforward. The size of the die grows roughly as $n^2 \log_2 n$, and the speed decreases as 1/n. Therefore, if a network of 256 neurons (which is an interesting size e.g. for image processing) is required, an area of about 20 cm^2 (using the same technology) would be required. Since this is out of reach, there are two solutions. The first is to use a technology which would be four times as small, which has the additional advantage of compensating for the decrease in speed, but 0.3 mm is still far from reach nowadays. The second solution consists in the so-called silicon hybridation technology (or multi-chip module), where specifically designed chips are laid on a common silicon substrate which provides interconnections (especially for power and common signals), the rest of the connections being wired between the chip pads, as in classical chip carrier bonding. This technology seems to be the most promising for the near future; therefore, we have decided to design a network based on the same architecture as the one described here, having four times its size. Each chip will then contain only 16 neurons arranged in open chain, each of them having a 256 word memory, 11-bit wide. An array of 16 of these chips, connected serially to form the systolic ring, will implement the whole network. Another advantage of this technique is that it allows the use of tested chips only, which alleviates the design and reliability issues, as compared to Wafer Scale Integration.

5. CONCLUSION

One of the central - and often overlooked - problems in the VLSI design of neural networks in digital technology is the determination of the accuracy of the synaptic coefficients; this is especially important when integrating the training algorithm itself. Clearly, the accuracy requirements depend to a great extent on the number of neurons, on the architecture of the network, on the data to be processed, and on the training algorithm. The present paper focusses on the determination of the accuracy, in the case of the design of an auto-

associative memory, for the storage of correlated patterns, with on-chip training. Various learning rules, which are integrable in silicon, have been investigated, and the associative memory properties of the resulting networks have been studied in detail by means of extensive simulations. The relationships between the architecture of the circuit and the learning rule have been investigated in order to minimize the extra circuitry required to implement training. Finally, the implementation of a 64-neuron associative memory with on-chip training, which has been manufactured, has been described, together with its future extensions.

Acknowledgments:
This work was supported by DRET under contract 87-187, by EEC under BRAIN contract ST27-0312-C, by G.CIS of CNRS, and by PRC "Architectures de Machines Nouvelles".

FIGURE CAPTIONS

Figure 1: Speed of learning with the Widrow-Hoff rule.
Number of presentations (Π) of the prototype set *vs*. the number of neurons (n) in order to have $|v_i^k - \sigma_i^k| \leq (1/n)$, for i=1 to n, and for all prototypes σ^k ; p/n = 0.25.

Figure 2: Mean value and standard-error of the diagonal coefficients ($<C_{ii}>$), averaged over 100 different sets of prototypes, *vs*. the number of prototype vectors which have been presented (N_p), for n = 64 and p/n = 0.25. The behaviour is similar for uncorrelated or for correlated prototypes, but the speed of convergence decreases when the <cos> increases.

Figure 3: Evolution of the mean and standard-error of the magnitude of the potentials, averaged over 100 different sets of prototypes, *vs*. the number of prototype vectors which have been presented (N_p), for n = 64 and p/n = 0.25.

Figure 4: Comparison between the behaviour of the Projection network and the behaviour of the IWH network during retrieval: the curve shows the proportion of identical initial states leading to different stable states with the Projection network, and with the IWH network, for various values of β_J (number of bits for encoding the coefficients); p/n = 0.25. This curve is the average over 10 000 initial states and over 20 prototype sets.

Figure 5: Behaviour of the Projection network: attractivity of the prototypes; the histogramm shows the distribution of the final normalized Hamming distances from one prototype (after retrieval) when the Projection network is initialized in a state at a normalized Hamming distance of 0.125, and 0.25 from this prototype.

Figure 6: Behaviour of the IWH network: attractivity of the prototypes for two values of β_J : a) β_J =7 bits ; b) β_J =9 bits.

Figure 7: Comparison between the Projection network and the IWH network: attractivity of the prototypes for various correlations between prototypes.

Figure 8: Projection network with integer coefficients: attractivity of the prototypes for two values of b.

Figure 9 : Schematics of the chip: most signals have self-explanatory names. The *CODE* and *DECODE* blocks and the *CODEvalid* signal implement the spurious state detection, the *RANDOM* block implements the random bit string generation for annealing purposes.

Figure 10 : Simplified schematics of one neuron: σ is the component of the network state vector (sequential), Cvg is the convergence signal (combinatorial), the C_{ij} are the synaptic coefficients. Common command and clock signals have been omitted for clarity.

Figure 11 : View of the chip: the neurons are arranged in 4 rows of 16. The surface of each neuron is divided into two parts of approximately equal areas: the memory (darker) and the ALU. The I/O and control circuitry are visible on the left-hand side of the chip.

LITERATURE REFERENCES

[1] J.J.Hopfield: "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. Nat. Acad. Sci. USA, vol.79, pp.2554-2558, 1982.

[2] J.J.Hopfield, D.W.Tank, "Neural computation of decisions in optimization problems", Biol.Cybern., vol. 52, pp.141-152 (1985).

[3] For an overview, see for instance: L. Personnaz, E. Bienenstock, G. Dreyfus, "A Cursory Introduction to the Physicists' Neural Networks", J. de Physique, vol. 50, pp. 207-208 (1989).

[4] L.Personnaz, I.Guyon, G.Dreyfus, "Collective computational properties of neural networks : New learning mechanisms", Phys. Rev. A, vol. 34, pp. 4217-4223 (1986).

[5] B.Widrow, S.D.Stearns, "Adaptive Signal Processing", Prentice-Hall Signal Processing Series, A.V.Oppenheim ed, 1985.

[6] F. Rosenblatt, "Principles of Neurodynamics: Perceptrons and the theory of brain mechanisms", Spartan Books, Washington (1962).

[7] W.Krauth, M.Mézard, "Learning algorithms with optimal stability in neural networks", J.Phys. A, vol. 20, pp. L745-L752, (1987).

[8] S.Diederich , M.Opper, "Learning of correlated patterns in spin-glass networks by local learning rules", Phys. Lett. Rev., vol. 58, pp.949-952, (1987).

[9] For an overview of silicon implementations of neural networks, see for instance:
Microelectronics for Neural Networks, U. Ramacher, ed. (Springer, 1991)
Silicon Architectures for Neural Nets, M.G. Sami, ed. (Elsevier, 1991).

[10] S.Y. Kung, J.N. Hwang, "Parallel Architectures for Artificial Neural Nets", Proceedings of the IEEE Conference on Neural Networks, vol. 2, pp. 165-172 (1988).

M. Weinfeld, "A Fully Digital CMOS Integrated Hopfield Network Including the Learning Algorithm", in VLSI for Artificial Intelligence, J.G. Delgado-Frias and W.Moore eds. (Kluwer Academic, 1989).

J.-D.Gascuel, M.Weinfeld, "A 64 Neuron Digital CMOS Fully Connected Neural Network with Properties Intended for Building Higher Level Architectures", International Joint Conference on Neural Networks, Seattle, 1991.