A CMOS Analog Adaptive BAM with On-Chip Learning and Weight Refreshing

Bernabé Linares-Barranco, *Member, IEEE*, Edgar Sánchez-Sinencio, *Fellow, IEEE*, Angel Rodríguez-Vázquez, *Member, IEEE*, and José L. Huertas, *Member, IEEE*

Abstract—**In this paper we will extend the transconductance-mode (T-mode) approach [1] to implement analog continuous-time neural network hardware systems to include on-chip Hebbian learning and on-chip analog weight storage capability. The demonstration vehicle used is a 5+5 neurons bidirectional associative memory (BAM) prototype fabricated in a standard 2-μm double-metal double-polysilicon CMOS process (through and thanks to MOSIS). Mismatches and nonidealities in learning neural hardware are supposed not to be critical if on-chip learning is available, because they will be implicitly compensated. However, mismatches in the learning circuits themselves cannot always be compensated. This mismatch is specially important if the learning circuits use transistors operating in weak inversion. In this paper we will estimate the expected mismatch between learning circuits in the BAM network prototype and evaluate its effect on the learning performance, using theoretical computations and Monte Carlo Hspice simulations. Afterwards we will verify these theoretical predictions with the experimentally measured results on the test vehicle prototype.**

I. Introduction

An artificial neural network is usually characterized by, first a basic topology that defines how the neurons are interconnected and how they transmit information from one to another, and second a learning rule that defines how the strengths of the inter-neuron connections (called *"synapses"*) change with time as different external stimuli are provided. Engineering researchers have proposed and studied many artificial neural networks [2]–[13], which are ultimately defined by differential or difference equations [14]. The way these artificial neural networks are implemented in practice is usually by software simulations on a conventional computer of the equations that define their behavior. Although this might be enough to validate the mathematical model, this approach is obviously not sufficient to obtain all the benefits inherent in highly parallel processing systems, and therefore some type of special purpose hardware technique needs to be devised to fully exploit the potentials of neural processing.

Many researchers are proposing digital type VLSI implementations [15]–[21] of neural networks, probably due to the enormous experience and success of this field achieved during the last decades in building all types of computing machines. The main advantages of digital circuit design techniques are high precision, ease of information storage, ease of sequencing and multiplexing, ease of interfacing, ease of design, and high reliability. However, there is a big drawback for this technique, which is excessive area consumption. Another disadvantage, however less critical, of digital circuit techniques concerns the processing delays characteristic of complex operations like multiplications, extensively used in neural network processing. This together with the fact that in neural networks high precision is not needed and that sequencing or multiplexing is not absolutely necessary if silicon area could be drastically reduced, has made some VLSI researchers with experience in analog design to propose some interesting alternatives for neural network hardware realizations [21]–[38]. However, there are still some issues that have not been clearly solved so far and still need more effort investments. One of them is the absence of an efficient and reliable analog storage mechanism, which has made many researchers to rely on off-chip digital memory storage with periodic refresh of on-chip capacitively stored analog weight voltages [21], [31], [36]–[39]. Another feature that seldom has been tackled by people building analog neural network VLSI (except, for example, [24]) is the on-chip learning capability. Lately some relatively large analog (or mixed analog/digital) learning neural network processing systems have been reported, but they perform learning through off-line conventional computing host machines [36]–[39].

The work we present in this paper can be considered as a further step in analog neural network hardware research, in the sense that it describes a modest size neural processing test prototype that performs on-chip learning and has an on-chip analog weight storage scheme able to keep the learned weights as long as power supply is available [40]. The chip we present, which corresponds to an adaptive bidirectional associative memory (BAM) [9], [10], is designed using an already proposed analog circuit design technique for continuous-time general neural networks, and that has been proven to be able to implement Hopfield, BAM, Winner-Take-All, simplified ART1, and optimization networks [1]. Here we will extend this technique to include an on-chip Hebbian learning rule, as well as an on-chip analog weight refreshing scheme.

Besides the area efficiency, high speed rates, on-chip learning, and on-chip weight storage capability, there is another characteristic that a neural network hardware implementation technique should have if it is intended for a vast variety of arbitrary size applications, namely the *modular capability*. This is an issue that already has been under consideration by several researchers [19], [24], [31], [37]–[39]. The approach

Manuscript received June 29, 1992; revised September 19, 1992.

B. Linares-Barranco, A. Rodriguez-Vazquez, and J. L. Huertas are with Centro Nacional de Microelectronica (CNM), Department of Analog Design, Ed. CICA, 41012 Seville, Spain.

E. Sanchez-Sinencio is with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77843.

IEEE Log Number 9206618.

we present in this paper is modular, and it is in such a way that it does not need any interfacing circuitry in order to assemble larger systems [1].

In the next section we will describe the architecture and topology of the BAM system prototype and give the circuit descriptions of the different building blocks in the chip. In Section III we will consider some aspects related to circuit mismatch that will affect the learning efficiency of the system, and in Section IV we will provide experimental results of the prototype fabricated in a standard 2-μm double-metal double-poly CMOS process (MOSIS).

II. System Architecture and Circuit Description

The continuous-time description[1] of most of the neural network algorithms available in the literature have a short-term memory (STM) described by the following set of nonlinear first-order differential equations,

$$C\dot{x}_i = -\alpha x_i + \sum_{j=1}^{N} w_{ji} f(x_j) + I_i, \qquad i = 1, \ldots N \quad (1)$$

where x_i is the activity of neuron i, w_{ji} is the weight of the synaptic interconnection that goes from neuron j to neuron i, I_i is the external input to neuron i, α, and C are positive constants, and $f(\cdot)$ is a nonlinear, monotonically increasing function with a maximum and a minimum saturation values (also called *"sigmoidal"* function). $f(x_j)$ represents the output of neuron j.

A. General T-mode Circuit Implementation

Elsewhere [1] we have shown that the description of (1) is functionally equivalent to the following mathematical description:

$$C\dot{y}_i = -g(y_i) + \sum_{j=1}^{N} w_{ji} y_j + I_i, \qquad i = 1, \ldots N \quad (2)$$

where now y_i is the activity and output of neuron i, and the nonlinear function $g(\cdot)$ is defined by

$$g(y) = \alpha f^{-1}(y). \quad (3)$$

The descriptions of (1) and (2) are totally equivalent in the quasi-stationary case ($\dot{x}_i \approx 0, \dot{y}_i \approx 0$). In [1] we showed that if the dynamics of the system described by (2) are a perturbation of the one described by (1), and if the weight matrix is invertible, then the same equivalent initial condition for both descriptions would yield the same equivalent final steady state.

Fig. 1 shows a T-mode circuit implementation for the mathematical model of (2). The synaptic elements are built using the simple transconductance multipliers of Fig. 2, the external current sources I_i with the circuit implementation shown in Fig. 3, and the nonlinear resistor $g(\cdot)$ represents the parallel association of the output resistances of all the synaptic multipliers in Fig. 1 with the circuit depicted in Fig. 4.

[1] Grossberg provides a method [14] to map a discrete-time description of a neural system into a continuous-time one, and vice-versa. Therefore, the neural network algorithms reported with discrete-time dynamics can also be represented by (1).

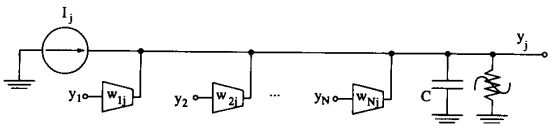

Fig. 1. T-mode implementation for neural networks.

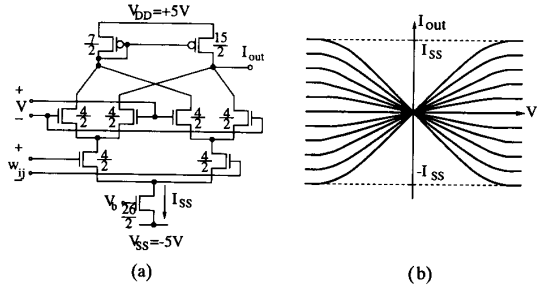

Fig. 2. (a) Actual schematic of transconductance multipliers. (b) Transfer curves.

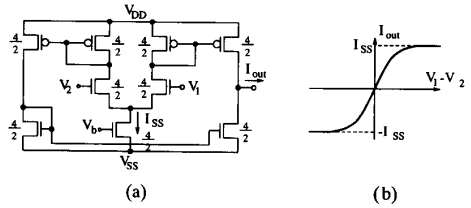

Fig. 3. (a) Circuit implementation of transconductance amplifier. (b) Transfer curves.

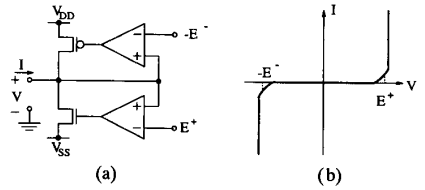

Fig. 4. (a) Nonlinear resistor circuit implementation. (b) Transfer curve.

B. BAM Architecture and T-Mode Realization

A bidirectional associative memory (BAM) is a two layer network in which all neurons in a layer are connected to all the neurons in the other layer through bidirectional synaptic connections. Its continuous-time version, already studied by Kosko [9]–[10], is described by the following set of nonlinear first order differential equations:

$$C\dot{a}_i = -\alpha a_i + \sum_{j=1}^{M} w_{ji} f(b_j) + I_i, \qquad i = 1, \ldots N$$

$$C\dot{b}_j = -\alpha b_j + \sum_{i=1}^{N} w_{ji} f(a_i) + J_j, \qquad j = 1, \ldots M \quad (4)$$

where a_i denotes the activity of the ith neuron in the first layer, b_j that of the jth neuron in the second layer, I_i and J_j are the corresponding external inputs, $f(\cdot)$ is a sigmoidal function, α and C are positive constants and w_{ji} is the weight

of the bidirectional synapse connecting neuron i in the first layer with neuron j in the second layer.

Using now the T-mode circuit design technique, the system of (4) results in the network depicted in Fig. 5. Note that this topology can be partitioned into several chips, so that modular capability is achieved with the penalty of increasing the node capacitances at each neuron. Fig. 6 illustrates the modularity issue, where it can be seen that no additional interfacing circuitry among chips is needed [1], [40].

C. Learning Circuitry for the T-mode BAM Network

The advantages of including an on-chip learning circuitry are listed as follows,

- Learning will be performed in situ, hence compensating for nonidealities present in the physical hardware of which the short term memory (STM) is made.
- No host computer is needed to perform the learning task off-line, hence speeding it up and simplifying the interface to the neural system in a practical real world application.

Different learning rules (also called long term memories (LTM)) can be used for an adaptive BAM [10], the simplest one being the one originally used by Kosko when he introduced the *"adaptive BAM"* [9], which is called the *"Hebbian learning rule"*. Inclusion of the Hebbian learning rule into the network of Fig. 5 is done by implementing, for each bidirectional synapse, the following nonlinear first order differential equation:

$$C_w \dot{w}_{ji} = -\beta w_{ji} + \kappa a_i b_j \qquad \begin{cases} i = 1, \ldots N \\ j = 1, \ldots M \end{cases} \tag{5}$$

where C_w, β, and κ are positive parameters. Note that in the T-mode circuit implementation of Fig. 5 each bidirectional synapse has locally available the output voltages, a_i and b_j, of the two neurons it interconnects. This makes very simple the circuit realization of the Hebbian learning rule for each synapse, as is depicted in Fig. 7. Multiplier *M3* has the same circuit than multipliers *M1* and *M2* (see Fig. 2), except that the tail bias current is two orders of magnitude less in order to decrease the time constant of the LTM. Therefore, *M3* will have to be biased with all its transistors operating in their weak inversion region of operation. The load resistance β is implemented with the same *M3* multiplier circuit, connected as a transconductance amplifier with negative feedback, hence emulating a positive resistance, and partially compensating its nonlinear transfer characteristics with those of multiplier *M3*.

Learning is performed by sequentially applying the set of training pairs of patterns at the inputs I_i and J_j of the BAM system. These pairs of patterns will have to be presented at a rate such that the voltage variations at the output of all *M3* multipliers (i.e., w_{ji}) are sufficiently smooth. This rate will depend on the time constants associated to the learning rule (LTM) and the one of the STM. After a few cycles of input patterns presentation a steady state will be reached for each synaptic weight w_{ji}. At this point learning can be considered to be complete and the capacitors C_w of each synapse can be isolated from their learning circuits (multiplier *M3* and resistor β), and used to store the learned weight voltage w_{ji} that biases

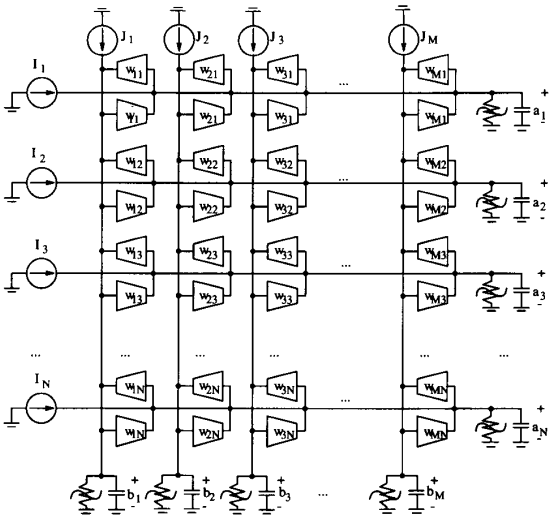

Fig. 5. T-mode circuit implementation of BAM algorithm.

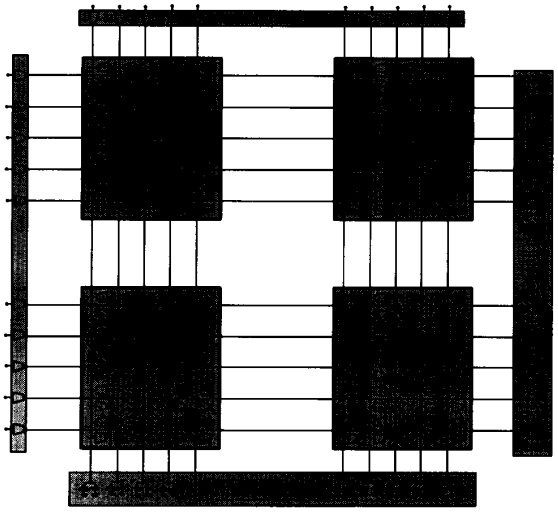

Fig. 6. Illustration of modular capability of T-mode circuit BAM implementation.

multipliers *M1* and *M2* of the STM. However, due to leakage currents present in the switches, the weight voltages would vanish in a few seconds if no kind of weight refreshing scheme is provided.

D. Weight Refreshing Scheme

The weight refreshing scheme used in our learning BAM is based on an A/D conversion followed by a D/A conversion of the weight value [41]. Periodically, each weight voltage is read and transformed through an A/D followed by a D/A conversion to a discretized version which is written back to the storage capacitor C_w. This way the weight voltages are kept within a finite interval. The synaptic weights in the chip are refreshed sequentially. The corresponding weight refreshing circuit is shown in Fig. 8. The A/D and D/A pair needs to be

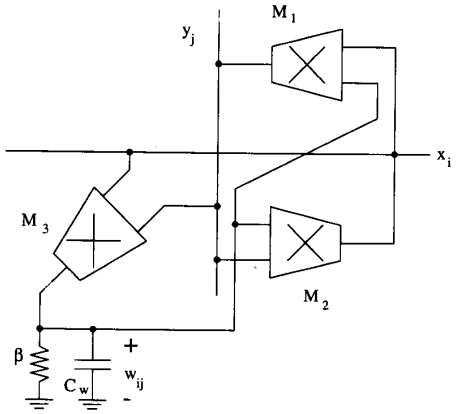

Fig. 7. Learning BAM synaptic T-mode circuit.

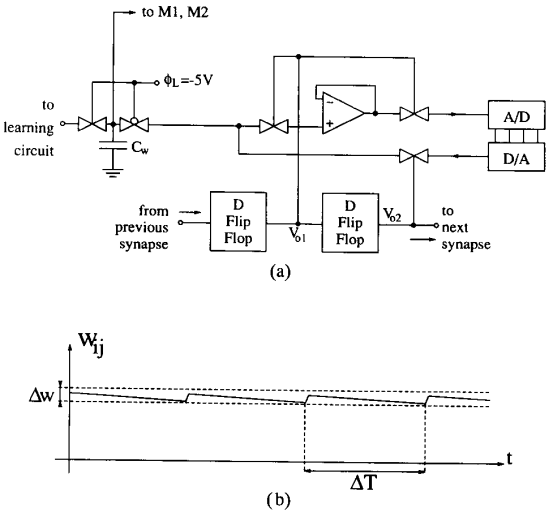

Fig. 8. (a) Refreshing circuit diagram. (b) Weight time waveform.

implemented only once per chip. For the modest size prototype we will present in Section IV no high resolution is needed for the weight values [9]. In our case, a discretization of the weight range into seven steps is more than sufficient. This will enable us, for this particular case, to simplify the circuitry of the A/D and D/A converters pair and use the one shown in Fig. 9. The circuit used for each D-flip-flop in Fig. 8(a), which is driven by two non-overlapping clock phases, is depicted in Fig. 10. The D-flip-flops of all the synapses are connected sequentially, so that a large D-flip-flop chain is formed inside the chip. Of this chain only one D-flip-flop at a time has an active output, so that only one synapse of the chip is selected at a time, and is connected either to the input or to the output of the A/D–D/A converter pair.

The weight storage capacitor C_w is connected to the learning circuitry of Fig. 7 during the learning phase, or to the refreshing circuit of Fig. 8 during the performing phase. Control signal ϕ_L (see Fig. 8) makes capacitor C_w to be connected to one or the other circuitry. Extra care needs to be taken, so that when ϕ_L is switched from the learning mode to the refreshing

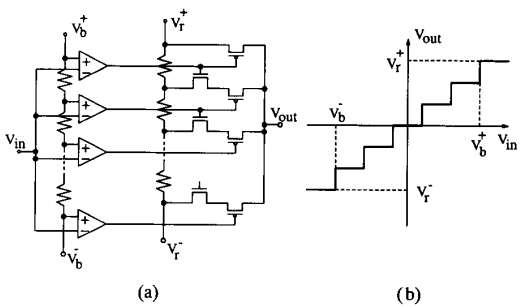

Fig. 9. (a) A/D–D/A converters pair circuit. (b) Transfer characteristics.

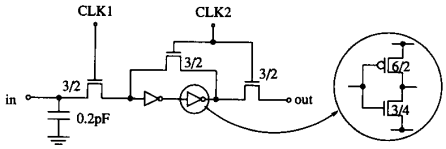

Fig. 10. Circuit diagram of D-flip-flop.

mode no synapse is selected at this moment, and the arbitrary voltage value present at the output of the D/A converter is not loaded into one of the synaptic weight storage capacitors.

Capacitor leakage rate values, l_r, are typically below 40 mV/s (for a $C_w = 2$ pF capacitor with connections to p^+ and n^+ diffusions). If Δw is the weight voltage precision required for a specific application and ΔT is the refreshing period for each synapse, then it must be

$$\Delta T \leq \frac{\Delta w}{l_r}. \tag{6}$$

Since all synapse weights have to be refreshed during the period ΔT, then the following condition has to be satisfied as well:

$$\Delta T \geq N_s \tau \tag{7}$$

where N_s is the total number of chip synapses and τ is the time employed to refresh a single synapse weight voltage.

III. Effect of Synaptic Mismatch on Learning Performance

We have already mentioned that one of the most important advantages of neural network systems with on-line learning is that, as they learn, they compensate for imperfections of nonidealities present in the STM. However, what about the imperfections and nonidealities present in the LTM? If the learning circuitry has defects, second order deviations, or even is nonoperative for some synapses, who is going to take care of this? At the time being, there is no supra-learning mechanism that will teach the LTM to modify its learning rule. However, the fact that the LTM is distributed all over the synaptic matrix makes the whole system to be more tolerant to local deviations in each synaptic learning circuitry. Nevertheless, there is a limit on the deviation that a system can tolerate for each synapse. As we will see, for a BAM system the maximum tolerable weight value deviations depend on the size of the

network, the number of patterns that have to be stored, and the hamming distance between the stored patterns.

In the T-mode BAM implementation described in the previous section the main nonideal effect of the learning circuitry that will affect the learning performance is the mismatch between the learning circuits of the synapses. This results in the dominant effect because the learning circuit components (multiplier *M3* and resistor β in Fig. 7) have to be operated with their transistors biased in weak inversion. Mismatch between MOS transistors is a function of their gate-to-source bias voltage and becomes maximum when they perform in their weak inversion region of operation [42].

Mismatch of a property P between two identical transistors is a function of their gate area and the "distance" between them [42]

$$\sigma^2(\Delta P) = \frac{A_p^2}{WL} + S_p^2 D_x^2 \tag{8}$$

where W and L are the widths and lengths of both transistors, D_x the distance between them projected on a maximum deviation gradient axis, and A_p and S_p are empirically determined parameters. Pelgrom *et al.* [42] verified that for small size transistors the distance dependent component of the deviation is negligible with respect to the area component. Since in our learning circuit all transistors have relatively small sizes we will consider that $S_p \approx 0$. The dominant transistor mismatch sources in our case are the mismatches between the threshold voltages V_T and the one between the mobility and gate oxide products $K' = \mu C_{ox}$ [43]. According to the data provided by Pelgrom *et al.* [42] and Lakshmikumar *et al.* [43], and taking into account the geometries of our NMOS and PMOS transistors, the following deviations seem to be reasonable to estimate the mismatch between transistors in the synaptic learning circuit we are using:

- NMOS:

$$\sigma(V_T) = 10 \text{ mV}$$
$$\sigma(K') = 0.5\ \mu\text{A/V}^2$$

- PMOS:

$$\sigma(V_T) = 15 \text{ mV}$$
$$\sigma(K') = 0.5\ \mu\text{A/V}^2 .$$

These deviations were used to perform a Monte Carlo Hspice [44] simulation of multiplier *M3* loaded with resistor β and capacitor C_w (see Fig. 7) while connecting one of the multiplier inputs to its maximum value and the other one to a variable duty cycle square wave signal. Depending on the duty cycle of the square wave signal a certain steady state voltage should be developed at capacitor C_w, which represents the learned weight of a synapse. This learned weight will suffer a deviation from synapse to synapse, due to the mismatch in the transistors that constitute multipliers *M3* and resistors β. The deviation of the final learned weight depends on the duty cycle of the signal used. Fig. 11 shows the result of several Monte Carlo Hspice simulations of the learning circuit for different duty cycles. The deviation in the steady state weight voltage

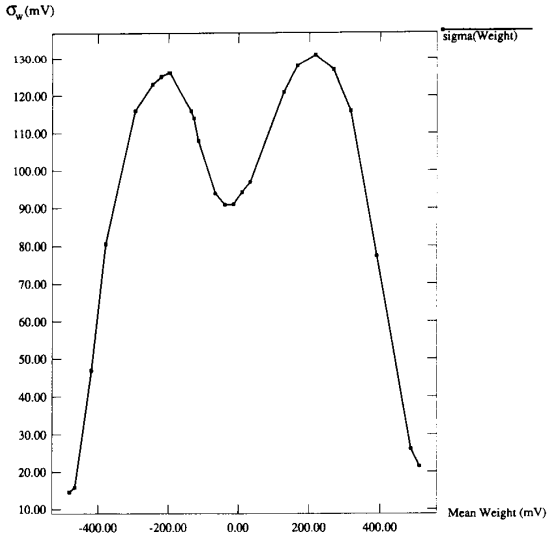

Fig. 11. Weight deviations (σ_w) as a function of nominal weight for the learning circuit of Fig. 7, obtained from Monte Carlo Hspice simulations.

is represented as a function of the mean weight voltage for each Monte Carlo simulation.

In order to estimate the weights deviation σ_w from the nominal values a T-mode BAM network can tolerate and still operate properly, we simulated numerically the mathematical equations that define the operation of the system. These are basically the equations given in (2) for the STM, and the ones in (5) for the LTM, to which we have added the nonlinear characteristics inherent in the multipliers:

$$C\dot{a}_i = -g(a_i) + \sum_{j=1}^{M} \omega_{\text{STM}}(w_{ji}, b_j) + I_i, \qquad i = 1, \ldots N$$
$$C\dot{b}_j = -g(b_j) + \sum_{i=1}^{N} \omega_{\text{STM}}(w_{ji}, a_i) + J_j, \qquad j = 1, \ldots M$$
$$C_w \dot{w}_{ji} = -\beta w_{ji} + \omega_{\text{LTM}}(a_i, b_j) \tag{9}$$

where the nonlinear functions are defined as follows:

$$g(x) = \begin{cases} +\infty & \text{if } x > V_L \\ \alpha x & \text{if } -V_L \le x \le V_L \\ -\infty & \text{if } x < V_L \end{cases} \tag{10}$$

$$\omega(y, x) = I_1(y, x) - I_2(y, x) \tag{11}$$

with

$$I_1(y,x) = \begin{cases} I_a(y) & \text{if } x > \sqrt{\frac{I_a(y)}{K_p}} \\ xK_p\sqrt{\frac{2I_a(y)}{K_p} - x^2} & \text{if } |x| \le \sqrt{\frac{I_a(y)}{K_p}} \\ -I_a(y) & \text{if } x < -\sqrt{\frac{I_a(y)}{K_p}} \end{cases} \tag{12}$$

$$I_2(y,x) = \begin{cases} I_b(y) & \text{if } x > \sqrt{\frac{I_b(y)}{K_p}} \\ xK_p\sqrt{\frac{2I_b(y)}{K_p} - x^2} & \text{if } |x| \le \sqrt{\frac{I_b(y)}{K_p}} \\ -I_b(y) & \text{if } x < -\sqrt{\frac{I_b(y)}{K_p}} \end{cases} \tag{13}$$

and the functions $I_a(y)$ and $I_b(y)$ defined by

$$\text{if } |y| \le \sqrt{\frac{I_{ss}}{K_p}} \Rightarrow \begin{cases} I_a(y) = \frac{K_p}{2}\left[\sqrt{\frac{I_{ss}}{K_p} - \frac{y^2}{2}} + \frac{y}{\sqrt{2}}\right]^2 \\ I_b(y) = \frac{K_p}{2}\left[\sqrt{\frac{I_{ss}}{K_p} - \frac{y^2}{2}} - \frac{y}{\sqrt{2}}\right]^2 \end{cases}$$

$$\text{if } y > \sqrt{\frac{I_{ss}}{K_p}} \Rightarrow \begin{cases} I_a(y) = I_{ss} \\ I_b(y) = 0 \end{cases}$$

$$\text{if } y < -\sqrt{\frac{I_{ss}}{K_p}} \Rightarrow \begin{cases} I_a(y) = 0 \\ I_b(y) = I_{ss}\,. \end{cases} \tag{14}$$

The values for the different parameters were

$$\alpha = 5 \times 10^{-7}\ \text{A/V}$$
$$V_L = 300\ \text{mV} \tag{15}$$

and for $\omega_{\text{STM}}(\cdot)$ we had

$$K_p = 2.25 \times 10^{-5}\ \text{A/V}^2$$
$$I_{ss} = 2.00\ \mu\text{A} \tag{16}$$

while for $\omega_{\text{LTM}}(\cdot)$ it was

$$K_p = 2.00 \times 10^{-7}\ \text{A/V}^2$$
$$I_{ss} = 50.0\ \text{nA} \tag{17}$$

which approximately corresponds to the conditions present in the experimental prototype we will see in the next section.[2]

Using this mathematical model to emulate the operation of a BAM, the system was trained for different network sizes and number of patterns. The learned weight values were then subjected to normally distributed random weight perturbations of different deviation (σ_w) values, and the resulting weight was discretized like it would be by the refreshing circuit. In order to test if a perturbated set of weights is still acceptable the following procedure was used: without providing any external inputs, and for each pair of patterns stored, the system was set to the initial condition that corresponds to one of the stored pairs of patterns and let to settle. If all neuron outputs remained in the same state, the perturbated set of weights was considered to be acceptable, and a new set of weights was computed with a larger σ_w. Fig. 12 shows this maximum tolerable weight deviation (σ_w) as a function of the number of neurons in each layer (the two layers were supposed to have the same number of neurons) and the number of patterns stored in the network, for a certain set of pairs of patterns. Naturally, the curves in Fig. 12 are highly patterns dependent. Actually, they would change dramatically as the hamming distances between the patterns decreases. Nevertheless, Fig. 12 provides us with a flavor of how the maximum tolerable weight deviations (σ_w) change as the number of neurons in the network or the number of stored patterns are altered. In the next section we will consider a particular case of BAM with a particular set of patterns, and we will compute, using the mathematical model described in this section, the maximum tolerable weight deviation (σ_w) for that particular case, and also we will take into account the dependence of this deviation with the nominal weight value, as shown in Fig. 11. Afterwards we will contrast these theoretical expectations with the experimental results observed in our test prototype.

[2]Note that, although multiplier $M3$ is biased in weak inversion we are using the equations that describe its operation when it works in strong inversion. However, with a proper selection of parameters K_p and I_{ss} the difference between both descriptions is small enough to render the same final result.

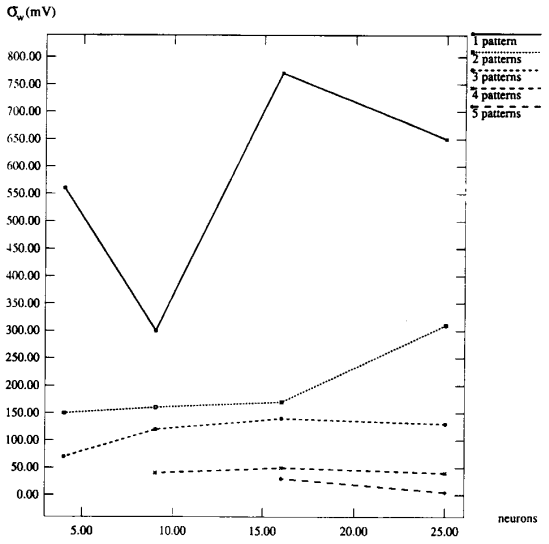

Fig. 12. Maximum tolerable weight deviations (σ_w) as a function of network size and number of stored patterns, obtained from theoretical statistical computations.

IV. Experimental Results

In a previous paper [1] we provided extensive experimental demonstrations of the performance and potential of the STM of programmable systems built using the T-mode approach. In this paper we are going to concentrate on the LTM performance, and therefore give experimental results that are more related to the learning circuit aspects.

We have fabricated a BAM network with on-chip Hebbian synaptic learning and on-chip weight refreshing in a 2-μm double metal double polysilicon standard CMOS process (MOSIS). The BAM network has five neurons per layer, and the chip contains, therefore, 25 bidirectional synapses with their corresponding learning and refreshing circuitry. The A/D and D/A converters pair is implemented only once per chip, and is shared sequentially by all the synapses on the chip.

In order to test the performance of the learning circuit, a circuit composed of multiplier *M3*, resistor β, and capacitor C_w (see Fig. 7) was set up. The inputs to multiplier *M3*, a_i, and b_j, were connected to external signal sources. Input b_j was set to its maximum value, and input a_i was connected to a variable duty-cycle square wave signal. After a transient response, a steady state voltage value develops at capacitor C_w, which is the learned weight value. Fig. 13 represents this weight value as a function of the duty-cycle of signal a_i. In a practical situation, during a training stage, the sequence of training patterns is presented iteratively at the external inputs I_i and J_j. Therefore, all neuron outputs a_i and b_j will be time

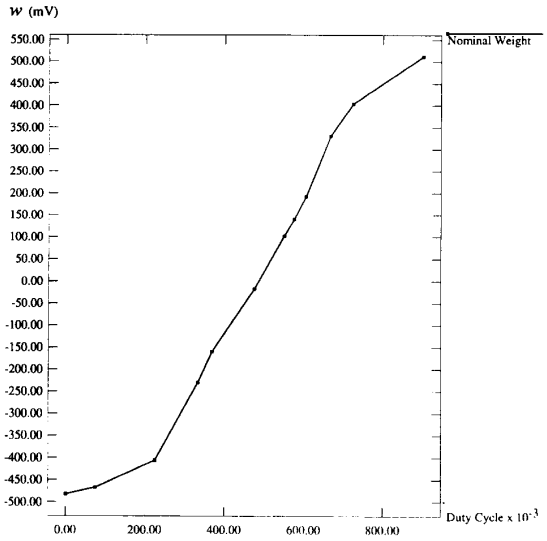

Fig. 13. Steady state weight value as a function of duty-cycle.

variant in general, but still the steady state weight value will be a function of the product of the effective duty-cycles[3] of the two inputs of multipliers *M3*.

For the circuits related to the refreshing operation, we first measured the leakage rate at the storage capacitor C_w. In order to minimize the leakage rate complementary switches were used to connect capacitor C_w to either the learning or the refreshing circuit. This way, both PMOS and NMOS transistors would be connected to the capacitor, and the effective leakage current the capacitor sees is the difference between the leakage currents at the p^{++} and n^{++} diffusions. This leakage rate has a slight dependence with the voltage at capacitor $C_w = 2$ pF, and the experimentally measured rates have always been below 34 mV per second. In our prototype weights were refreshed every 8 ms, which is much faster than what is needed to keep the resolution we have. The resolution is given by in how many steps the A/D–D/A converters pair discretizes the weight values range. Fig. 14 shows the measured voltage transfer function for the A/D–D/A pair. In this case, the weight range has been divided into seven steps. This would allow us to store up to six different patterns [9], [10], [45] if the BAM network had this capacity. Empirical BAM storage capacity experiments [45] reveal that in order to successfully store six different patterns in a BAM, it would need an approximate number of 40 neurons per layer. In our prototype the BAM has 5 neurons per layer. The storage capacity of such a network, if the patterns have a sufficiently large hamming distance between them, could be of two or three pairs of patterns. Therefore, the weight resolution we have implemented is far beyond our needs.

In order to evaluate the learning performance of the complete BAM system we trained it for different pairs of patterns. First, let us consider the case of storing a single pattern pair,

[3]Note that a_i and b_j will not be square-wave signals.

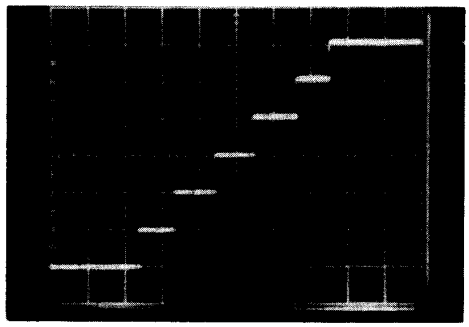

Fig. 14. Measured transfer characteristics of A/D–D/A converters pair. Horizontal and vertical scales are 200 mV/div.

for instance,[4]

$$\vec{a} = (+1, -1, +1, -1, +1)$$
$$\vec{b} = (-1, +1, -1, +1, -1) \tag{18}$$

where $\vec{a}$ represents the outputs of the neurons of the first layer, and $\vec{b}$ the ones of the second layer. According to the Hebbian learning rule, this should generate the following normalized weight matrix.

$$\tilde{W} = \begin{bmatrix} -1 & +1 & -1 & +1 & -1 \\ +1 & -1 & +1 & -1 & +1 \\ -1 & +1 & -1 & +1 & -1 \\ +1 & -1 & +1 & -1 & +1 \\ -1 & +1 & -1 & +1 & -1 \end{bmatrix}. \tag{19}$$

Fig. 15 shows the measured output of the D/A converter during the refreshing operation, once the pair of (18) was stored into the network. This figure represents the sequence of the 25 weight values of the matrix (19). While this weight matrix was being refreshed, we provided several input vectors to either the inputs I_i or J_j or both. The network did converge to either the pair of patterns in (18) or to its complementary, depending on which of the two had a smaller hamming distance to the external inputs. Fig. 16 shows how the signals a_i and b_j converge to the pattern pair of (18) in one of the cases. In order to guarantee that the system had reached a stable local energy minimum, the external currents I_i and J_j were made zero after the steady state was reached, and it was verified that the system remained in the same state.

Fig. 17 shows the measured weight matrix for the case pattern pair

$$\vec{a} = (+1, -1, +1, -1, +1)$$
$$\vec{b} = (+1, +1, -1, +1, +1) \tag{20}$$

was used for training the BAM network. The corresponding normalized weight matrix is

$$\tilde{W} = \begin{bmatrix} +1 & -1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 \\ -1 & +1 & -1 & +1 & -1 \\ +1 & -1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 \end{bmatrix}. \tag{21}$$

[4]These values are normalized with respect to the complete neuron output voltage range, which was 1 V.

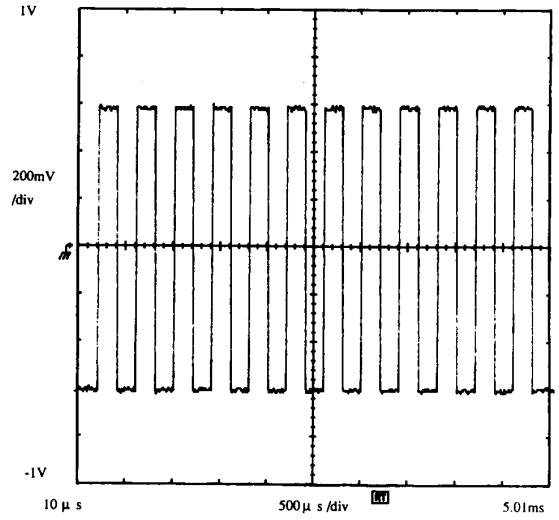

Fig. 15. Experimentally measured sequence of refreshed weight values for the pattern pair of (16).

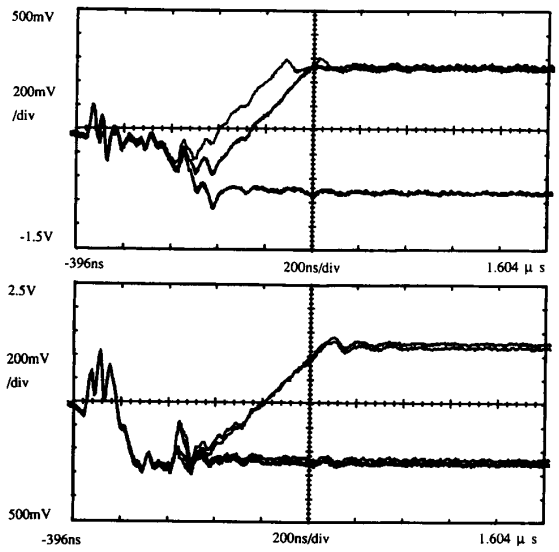

Fig. 16. Convergence to pattern of (16). Top traces correspond to neurons a_i, bottom traces to neurons b_j.

The experimentally measured convergence to the pattern pair of (20) is shown in Fig. 18.

We also trained the network to recall the following two pairs of patterns.

$$\text{Pattern } A: \begin{cases} \vec{a} = (+1,-1,+1,-1,+1) \\ \vec{b} = (-1,+1,-1,+1,-1) \end{cases}$$

$$\text{Pattern } B: \begin{cases} \vec{a} = (+1,+1,+1,-1,-1) \\ \vec{b} = (-1,-1,-1,-1,-1). \end{cases} \quad (22)$$

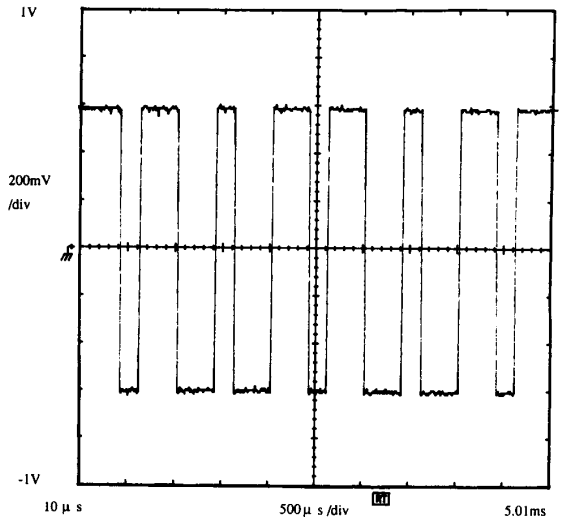

Fig. 17. Experimentally measured sequence of refreshed weight values for the pattern pair of (18).

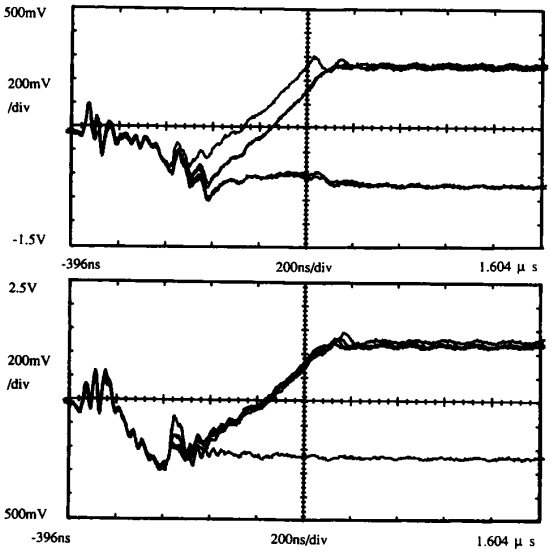

Fig. 18. Convergence to pattern of (18). Top traces correspond to neurons a_i, bottom traces to neurons b_j.

The corresponding normalized weight matrix is

$$\tilde{W} = \begin{bmatrix} -1 & 0 & -1 & +1 & 0 \\ 0 & -1 & 0 & 0 & +1 \\ -1 & 0 & -1 & +1 & 0 \\ 0 & -1 & 0 & 0 & +1 \\ -1 & 0 & -1 & +1 & 0 \end{bmatrix}. \quad (23)$$

The measured weight matrix for this case is given in Fig. 19. After learning was accomplished and the system was switched to the refreshing mode, the network did always converge to either patterns A, B, or their complementaries, depending on which of them had the smallest hamming distance to the external input pattern pair $(\vec{I}, \vec{J})$, where $\vec{I} = (I_1, I_2, I_3, I_4, I_5)$ and $\vec{J} = (J_1, J_2, J_3, J_4, J_5)$. As an illustration, Fig. 20 shows

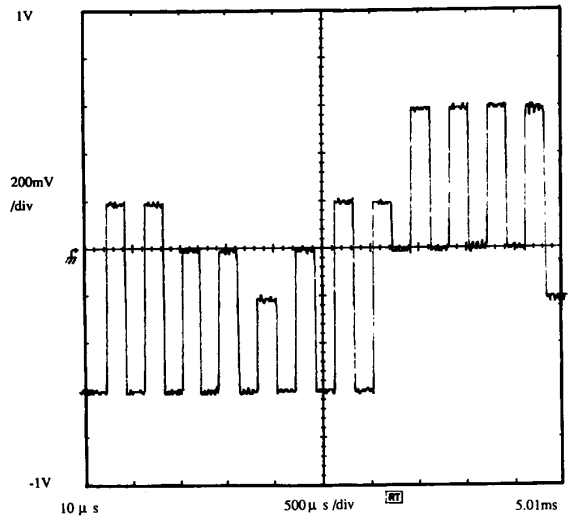

Fig. 19. Experimentally measured sequence of refreshed weight values for the pattern pairs of (20).

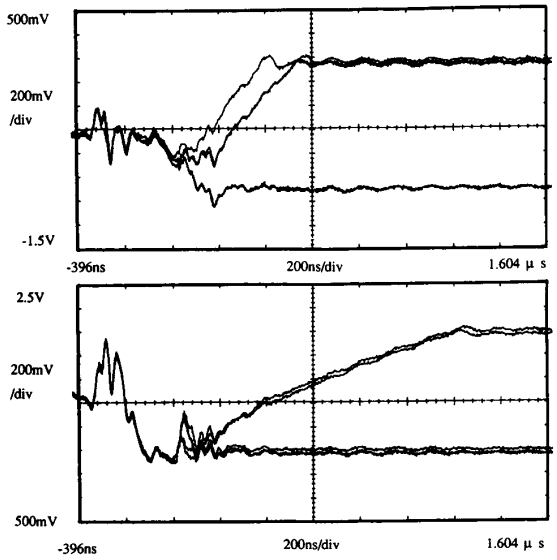

Fig. 20. Convergence to the pattern A of (20). Top traces correspond to neurons a_i, bottom traces to neurons b_j.

the convergence to pattern A in one of those cases. When no external input current was provided $\vec{I} = \vec{J} = 0$ the BAM preferred to converge to pattern B, as is shown in Fig. 21. All stored patterns are stable, in the sense that if the external input currents are made zero (after the network converges to one of the stored patterns or complementaries) the system remains at the same pattern.

Note that 6 out of the 25 weights (see Fig. 19) suffer a deviation from their normal value. This is a consequence of the mismatch between the learning circuits of the different synapses. However, the BAM network was still able to successfully retrieve all stored patterns (and their complementaries). According to the Monte Carlo simulation of Fig. 11 for

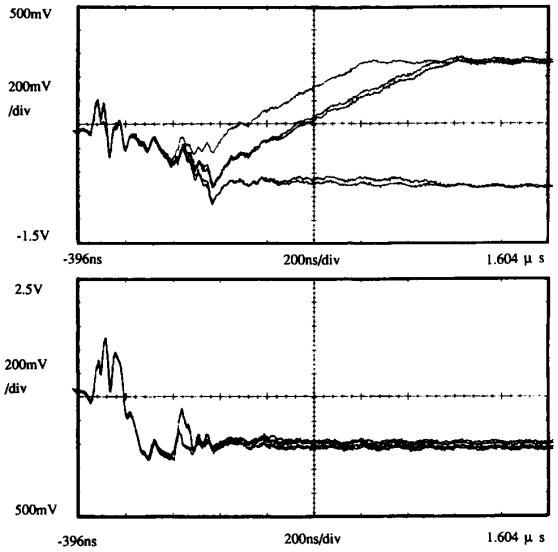

Fig. 21. Convergence to the pattern B of (20) when no external input currents are provided. Top traces correspond to neurons a_i, bottom traces to neurons b_j.

our synaptic learning circuit, the expected weight deviations for the matrix (23) would be,

$$w_{ji}\,|_{\text{normalized}} = 0 \rightarrow \sigma_w = 93\ \text{mV}$$
$$w_{ji}\,|_{\text{normalized}} = \pm 1 \rightarrow \sigma_w < 25\ \text{mV}. \qquad (24)$$

Using the mathematical model of (9)–(16) for this particular BAM network with the particular patterns of (22), and neglecting σ_w when $w_{ji}\,|_{\text{normalized}} = \pm 1$, the maximum tolerable σ_w for $w_{ji}\,|_{\text{normalized}} = 0$ was

$$\sigma_{w=0}\,|_{\max} = 130\ \text{mV}. \qquad (25)$$

Therefore, this predicts that with our learning circuit we should be able to store the two pairs of patterns of (22), and which was verified experimentally as shown previously.

In order to compute now the maximum tolerable $\sigma_{w=0}$ for the case of storing three pairs of patterns, we used again the mathematical model of (9)–(16) as described in Section III. The computed value, for a set with the maximum possible hamming distances between patterns, was

$$\sigma_{w=0}\,|_{\max} = 20\ \text{mV}. \qquad (26)$$

This means that with our learning circuitry we should not be able to store three pairs of patterns in our BAM network. We tried to train the BAM prototype for that set of three pairs of patterns. The network was then able to converge to all three of them and their complementaries, provided the external inputs $(\vec{I}, \vec{J})$corresponded exactly to those patterns. However, if once the steady state was reached and these currents were made zero, the system moved to a slightly different state for some of the patterns. Therefore, the local energy minimums did not exactly correspond to the stored set of patterns. Hence, our BAM network is not able to store three pairs of patterns, which was predicted by our theoretical computations and Monte Carlo simulations.

V. Conclusions

We have designed, fabricated and tested a 5+5 neurons BAM network prototype with on-chip Hebbian learning and on-chip analog weight storage. The prototype was designed using analog continuous-time circuit design techniques based on the use of transconductance synaptic multipliers and nonlinear neural resistors, as well as capacitors (T-mode approach). Due to the fact that the learning circuitry of the synapses uses transistors biased in their weak inversion region of operation, the effects of mismatch between learning circuits of different synapses were studied. It was verified that the most important effect of this mismatch is on the learning performance, so that the maximum theoretical storage capacity of the network is degraded. The grade of storage degradation was predicted theoretically through the use of a mathematical model of the network and Monte Carlo Hspice simulations, and then was experimentally verified with the fabricated prototype.

The work presented in this paper validates the use of the T-mode approach to implement analog learning hardware neural network systems and demonstrates the high potential of this technique for the implementation of low cost, small area, high speed neural hardware.

References

[1] B. Linares-Barranco, E. Sánchez-Sinencio, A. Rodríguez-Vázquez, and J. L. Huertas, "A modular T-mode design approach for analog neural network hardware implementations," *IEEE J. Solid-State Circuits,* vol. 27, May 1992.

[2] J. J. Anderson, "A simple neural network generating interactive memory," *Math. Biosciences,* vol. 14, pp. 197–220, 1972.

[3] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing,* vol. 37, pp. 54–115, 1987.

[4] G. E. Hinton and R. J. Sejnovski, "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing,* D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: M.I.T. Press, 1986, vol. 1, ch. 7.

[5] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci.,* vol. 79, Apr. 1982, pp. 2554–2558.

[6] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci.,* vol. 81, May 1984, pp. 3088–3092.

[7] J. J. Hopfield and D. W. Tank, "Neural computation of decisions optimization problems," *Biol. Cybern.,* vol. 52, pp. 141–152, 1985.

[8] T. Kohonen, "The 'neural' phonetic typewriter," *Computer,* pp. 11–22, vol. 21, Mar. 1988.

[9] B. Kosko, "Adaptive bidirectional associative memories," *Appl. Opt.,* vol. 26, pp. 4947–4960, Dec. 1987.

[10] B. Kosko, *Neural Networks and Fuzzy Systems.* Englewood, NJ: Prentice-Hall, 1992.

[11] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.,* Apr. 1987.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature 323,* pp. 533–536, 1986.

[13] D. W. Tank and J. J. Hopfield, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.,* vol. 33, pp. 533–541, May 1986.

[14] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Networks,* vol. 1, no. 1, pp. 17–61, 1988.

[15] L. E. Atlas and Y. Suzuki, "Digital systems for artificial neural networks," *IEEE Circuits Devices,* pp. 20–24, 1989.

[16] M. Ysunaga *et al.,* "Design, fabrication and evaluation of a 5-inch wafer scale neural network LSI composed of 576 digital neurons," in *Proc. of the 1990 Int. Joint Conf. Neural Networks (IJCNN'90),* vol II, 1990, pp. 527–535.

[17] M. Duranton, J. Gobert, and N. Mauduit, "A digital VLSI module for neural networks," in *Proc. Euro88, IDSET,* Paris, France, 1988, pp. 720–724.

[18] R. Hecht-Nielsen, *Neurocomputing.* Boston, MA: Addison-Wesley, 1990.

[19] P. Treleaven, M. Pacheco, and M. Vellasco, "VLSI architectures for neural networks," *IEEE Micro,* pp. 8–27, Dec. 1989.

[20] D. E. Van Den Bout and T. K. Miller III, "A digital architecture employing stochasticism for the simulation of Hopfield neural nets," *IEEE Trans. Circuits Sys.,* vol 36, pp. 732–738, May 1989.

[21] J. E. Tomberg and K. K. K. Kaski, "Pulse-density modulation technique in VLSI implementations of neural network algorithms," *IEEE J. Solid-State Circuits,* vol. 25, pp. 1277–1286, Oct. 1990.

[22] P. Mueller *et al.,* "Design and fabrication of VLSI components for a general purpose analog neural computer," *IEEE Workshop on Analog VLSI Implementation of Neural Systems,* C. Mead and M. Ismail, Eds. Boston: Kluwer, 1989.

[23] C. Mead, *Analog VLSI and Neural Systems.* Boston: Addison-Wesley, 1989.

[24] Y. Arima, K. Mashiko, K. Okada, T. Yamada, A. Maeda, H. Notani, H. Kondoh, and S. Kayano, "A 336-neuron, 28K-synapse, self-learning neural network chip with branch-neuron-unit architecture," *IEEE J. Solid-State Circuits,* vol. 26, pp. 1637–1644, Nov. 1991.

[25] K. A. Boahen, P. O. Pouliquen, A. G. Andreou, R. E. Jenkins, "A heteroassociative memory using current-mode MOS analog VLSI circuits," *IEEE Trans. Circuits Syst.,* vol. 36, pp. 747–755, May 1989.

[26] S. Eberhardt, T. Duong, and A. Thakoor, "Design of parallel hardware neural network systems for custom analog VLSI 'building block' chips," in *Proc. of Int. Joint Conf. on Neural Networks, II,* IEEE Press, New York, June 1989, pp. 183–190.

[27] L. D. Jackel, H. P. Graf, W. Hubbard, J. S. Denker, D. Henderson, and I. Guyon, "An application of neural net chips: Handwritten digit recognition," in *Proc. IEEE Int. Conf. on Neural Networks, II,* New York, July 1988, pp. 107–115.

[28] B. W. Lee and B. J. Sheu, *Hardware Annealing in Analog VLSI Neurocomputing.* Boston, MA: Kluwer Academic 1991.

[29] J. Meador *et al.,* "Programmable impulse neural circuits," *IEEE Trans. Neural Networks,* vol. 2, pp. 101–109, Jan. 1990.

[30] J. Meador, D. Watola, and N. Nintunze, "VLSI implementation of a pulse-Hebbian learning law," *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'91),* Singapore, 1991, pp. 1287–1290.

[31] A. Hamilton, A. F. Murray, D. J. Baxter, S. Churcher, H. M. Reekie, and L. Tarassenko, "Integrated pulse stream neural networks: Results, issues, and pointers," *IEEE Trans. Neural Networks,* vol. 3, pp. 385–393, May 1992.

[32] M. E. Robinson, H. Yoneda, and E. Sánchez-Sinencio, "A modular CMOS design of a hamming network," *IEEE Trans. Neural Networks,* vol. 3, pp. 444–456, May 1992.

[33] A. Rodríguez-Vázquez, R. Domínguez-Castro, A. Rueda, J. L. Huertas, and E. Sánchez-Sinencio, "Nonlinear switched-capacitor 'neural' networks for optimization problems," *IEEE Trans. Circuits and Sys.,* vol. 37, pp. 384–398, Mar. 1990.

[34] J. P. Sage, K. Thompson, and R. S. Withers, "An artificial neural network integrated circuit based upon MNOS/CCD principles," in *Neural Networks for Computing,* J. S. Denker, Ed., AIP Conf. Proc., 151, Am. Inst. of Physics, New York, 1986, pp. 381–385.

[35] S. W. Tsay and R. W. Newcomb, "VLSI implementation of ART1 memories," *IEEE Trans. Neural Networks,* vol. 2, pp. 214–221, Mar. 1991.

[36] B. E. Boser, E. Sackinger, J. Bromley, Y. L. Cun, and L. D. Jackel, "An analog neural network processor with programmable topology," *IEEE J. Solid-State Circuits,* vol. 26, pp. 2017–2025, Dec. 1991.

[37] S. Satyanarayan, Y. P. Tsividis, and H. P. Graf, "A reconfigurable VLSI neural network," *IEEE J. Solid-State Circuits,* vol. 27, pp. 67–81, Jan. 1992.

[38] J. Van der Spiegel, P. Mueller, D. Blackman, P. Chance, C. Donham, R. Etienne-Cummings, and P. Kinget, "An analog neural computer with modular architecture for real-time dynamic computations," *IEEE J. Solid-State Circuits,* vol. 27, pp. 82–92, Jan. 1992.

[39] R. Mason, W. Robertson, and D. Pincock, "An hierarchical VLSI neural network architecture," *IEEE J. Solid-State Circuits,* vol. 27, pp. 106–108, Jan. 1992.

[40] B. Linares-Barranco, "Analog neural network VLSI implementations," Ph.D. dissertation, Texas A&M University, College Station, TX, Dec. 1991.

[41] P. B. Brown, R. Millecchia, and M. Stinely, "Analog memory for continuous-voltage, discrete-time implementation of neural networks," in *Proc. IEEE ICNN'87,* vol. 3, pp. 523–530, 1987.

[42] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching properties of MOS transistors," *IEEE J. Solid-State Circuits,* vol. 24, pp. 1433–1440, Oct. 1989.

[43] K. R. Lakshmikumar, R. A. Hadaway, and M. A. Copeland, "Characterization and modeling of mismatch in MOS transistors for precision analog design," *IEEE J. Solid-State Circuits,* vol. SC-21, pp. 1057–1066, Dec. 1986.
[44] Meta-Software, HSPICE User's Manual H9007, Meta-Software, Inc., 1300 White Oaks Road, Campbell, CA 95008.
[45] Y. F. Wand, J. B. Cruz, and J. H. Mulligan, "On multiple training for bidirectional associative memory," *IEEE Trans. Neural Networks,* vol. 1, pp. 275–276, Sep. 1990.

Bernabé Linares-Barranco was born in Granada, Spain, on November 26, 1992. He did his elementary studies in Germany until 1975. He received the B.Sc. degree in electronic physics in June 1986, and the M.S. degree in microelectronics in September 1987, both from the University of Seville, Sevilla, Spain. He received his first Ph.D. degree in high-frequency OTA-C oscillator design in June 1990 from the University of Seville, Spain, and the second Ph.D. degree in analog neural network design in December 1991 from Texas A&M University, College Station.

Since September 1991 he has been a Senior Researcher at the Analog Design Department of the Microelectronics National Center (CNM), Sevilla, Spain. His research interests are in the area of nonlinear analog and neural network microelectronic design.

Edgar Sánchez-Sinencio (S'72–M'74–SM'83–F'92) for a photograph and biography please see page 386 of this issue.

Angel Rodriguez-Vázquez (M'80) received the Licenciado en Física degree in 1977, and the Doctor en Ciencias Físicas degree in 1983, both from the University of Seville, Spain.

Since 1978 he has been with the Department of Electronics and Electromagnetism at the University of Seville where he is an associate professor. He is also at the Department of Analog Circuit Design of the Centro Nacional de Microelectrónica. His research interests are in the fields of analog/digital integrated circuit design, analog integrated neural and nonlinear networks, and modeling of analog integrated circuits.

José L. Huertas (M'74) received the Licenciado en Física degree in 1969 and the Doctor en Ciencias Físicas degree in 1973, both from the University of Seville, Spain. From 1970 to 1971 he was with the Philips International Institute, Eindhoven, The Netherlands, as a postgraduate student.

Since 1971 he has been with the Department of Electronics and Electromagnetism at the University of Seville where he is a professor. He is also at the Department of Analog Circuit Design of the Centro Nacional de Microelectronica. His research interests are in the fields of multivalued logic, sequential machines, analog circuit design, and nonlinear network analysis and synthesis.