VLSI IMPLEMENTATION OF A TRANSCONDUCTANCE MODE CONTINUOUS BAM WITH ON CHIP LEARNING AND DYNAMIC ANALOG MEMORY

B. Linares–Barranco[1], *E. Sánchez–Sinencio,*[1], *A. Rodríguez–Vázquez*[2] *& J. L. Huertas*[2]
[1] Texas A&M University, College Station, TX 77843–USA
[2] Universidad de Sevilla, 41012–Sevilla, Spain

Abstract–In this paper we present a complete VLSI Continuous–Time Bidirectional Associative Memory (BAM). The short term memory (STM) section is implemented using small transconductance four quadrant multipliers, and capacitors for the integrators. The long term memory (LTM) is built using an additional multiplier that uses locally available signals to perform Hebbian learning. The value of the learned weight is present at a capacitor for each synapse. After learning has been accomplished the value of the stored weight voltage can be refreshed using a simple A/D–D/A conversion, which if done fast enough, will maintain the weight value within a discrete interval of the complete weight range. Such a discretization still allows good performance of the STM section after learning is finished.

I. Introduction

In 1987 Bart Kosko published its Bidirectional Associative Memory (BAM) Algorithm [1,2]. He proposed two different versions. The first one is discrete–time, characterized by finite difference equations and appropriate for simulation programs and discrete systems in general. The second one is continuous–time, characterized by differential equations and thus, potentially possible to be implemented with analog continuous–time circuit techniques.

Kosko's Continuous–Time BAM can be viewed as a simplification of Carpenter and Grossberg's ART Associative Memory [3], in the sense that it does not incorporate the orienting subsystem, attentional vigilance and the gain control features, and therefore does not have the self–scaling, self–adjusting memory search and attentional priming characteristics of an ART system. However, it still can be used as an efficient associative memory and its simplicity makes it very attractive as a good candidate for a VLSI implementation. One of the goals in a VLSI neural network implementation is to obtain the highest possible density on silicon. Therefore, analog circuit techniques are very suitable for this purpose. Also, analog circuits are able to implement easily differential equations. For this reason it seems appropriate to implement Kosko's continuous BAM using analog circuit techniques.

In this paper we will first give an implementation of the STM (short term memory) section using transconductance mode (T–mode) analog circuits techniques. This consists of a circuit that realizes Kosko's STM continuous differential equations. After this, and for each synapse, an implementation of the LTM (long term memory) differential equation will be given, so that learning may be performed. The value of each weight will be stored as a voltage on a capacitor. However, in CMOS VLSI capacitors there are leakage currents that discharge them in a few milliseconds. Therefore, if the training process is stopped, the learned weights will vanish in a short time. For this reason, once learning is accomplished, the circuit will be switched to a refresh mode, in which the voltage of each weight capacitor will be dynamically refreshed [4]. This is done by reading the voltage in one of these capacitors with an A/D converter, transforming this value again with a D/A converter, and restoring it into the same capacitors. This refreshing technique implies a discretization in the values of the weights. We have verified through computer simulation that, after learning, a certain granularity is allowed for keeping proper operation of the BAM.

In the following Sections, implementation details of the STM, the LTM in the learning mode, and the LTM in the refreshing mode will be given. Simulation results confirm proper operation of the system. Chips that contain different subparts of a complete BAM have been sent for fabrication. Simulation results will be presented.

II. The Short Term Memory

Fig. 1. represents Kosko's continuous BAM architecture which is described [1] by the following set of differential equations:

$$C\dot{a}_i = -\alpha a_i + \sum_j S(b_j)m_{ij} + I_i$$

$$C\dot{b}_j = -\alpha b_j + \sum_i S(a_i)m_{ij} + I_j \qquad (1)$$

where a_i is the activity of neuron i in the top layer, b_j is the one of neuron j in the bottom layer, and $S(\cdot)$ is a nonlinear sigmoidal type function.

For each neuron (a_i or b_j), its corresponding differential equation can be physically implemented using transconductance devices, as shown in Fig. 2 for neuron a_i.

The shape of the sigmoidal function $S(a_i)$ is shown in Fig. 3(a). Suppose now that we make $S(a_i)$ to have a slope of 1 in its central region, and that a_i is not allowed to exceed the limits $+V_L$, $-V_L$. We have verified, through numerical simulation, that the overall behavior of the system is still the same after this modification. In this case we can eliminate the sigmoidal element of Fig. 2 and replace the linear resistor α by a nonlinear resistor that would limit the voltage a_i to the range $[-V_L, +V_L]$. For each neuron, the circuit would become as shown in Fig. 4, and the driving point characteristics of each nonlinear resistor would be as illustrated in Fig. 3(b).

The resulting complete BAM network is shown in Fig. 5, where the voltage controlled current sources of Fig. 4 have been substituted by a transconductor symbol of transconductance m_{ij}. The corresponding differential equations become

$$\begin{aligned} C\dot{a}_i &= -i_{NL}(a_i) + \sum_j m_{ij}b_j + I_i \\ C\dot{b}_j &= -i_{NL}(b_j) + \sum_i m_{ij}a_i + J_j \qquad (2) \end{aligned}$$

We have computer simulated the two sets of differential equations (1) and (2) to reproduce Kosko's results in [1] and no significant difference was appreciated between the two approaches. Furthermore, when incorporating learning to the simulation program, the learned weights were approximately the same for the two cases. Even when making the synaptic transconductance devices of Fig. 5 to have a saturating nonlinearity the whole network kept working properly and the learned weights still were very similar to the ones obtained previously. Therefore, in a practical CMOS implementation we can replace each transconductance of Fig. 5 by a single non–linearized multiplier, like the one shown in Fig. 6. Its total area is $52\mu m \times 42\mu m$.

Note that two different grounds (GND1 and GND2) are used. This allows the elimination of a level shifter at the gates of M_1 and M_2, and therefore, keeping a small area for the complete multiplier.

III. The Long Term Memory

In order to incorporate learning to the continuous BAM of Fig. 5, the values of the weights have to be able to change in time, while different sets of input pairs $\{I_i, J_j\}$ are being switched iteratively. A very simple learning algorithm, that was used by Kosko [1] is the Hebbian learning. This means, that for each synapse the following differential equation is applied

$$C_M\dot{m}_{ij} = -\beta m_{ij} + Ka_ib_j \qquad (3)$$

where C_M, β and K are positive parameters. A circuit that would implement this equation is shown in Fig. 7 for one synapse.

The value of the learned weight is stored in a capacitor C_M. Since there exists a leakage current in parallel with each capacitor, the time constant of the learning process C_M/β has to be much faster than the one associated with the leakage, so that learning can be accomplished during the training process. Note that the leakage current has been neglected in (3).

Once the network has been trained and all m_{ij} reach their steady state, the capacitor C_M can be switched out of the LTM circuit and remain connected only to the gates of the synaptic multipliers. However, due to the leakage currents present, the voltage stored in capacitor C_M would vanish in a few miliseconds. To avoid this we use a dynamic analog refreshing technique [4]. This technique is based on the fact that once learning has been accomplished the value of m_{ij} allows a certain degree of granularity, i.e., its value can be discretized within its dynamic range and the BAM still remembers all the stored pairs. The number of discrete steps allowed in the dynamic range is given by the maximum number of pattern pairs that can be stored. If N is this maximum number, then there must be a minimum of $N+1$ discrete steps within the dynamic range of m_{ij}. The load resistance β at node m_{ij} is implemented according to the circuit in Fig. 8, where the value β can be adjusted through V_β, and the feedback amplifier at the gates of PMOS transistors assures that the resting potential for m_{ij} is ground for any value of V_β.

IV. Dynamic Analog Refreshing Technique

Consider the circuit shown in Fig. 9. During the training process ϕ_L is high and ϕ_R and ϕ_W are low, so that the capacitor C_M of each synapse is connected as shown in Fig. 7. Once training is over and all the weights m_{ij} have reached their stationary values ϕ_L goes down and the refreshing circuit comes into play. The refreshing circuit consists of a large shift register of D–Flip–Flops. Each synapse has 3 of these Flip–Flops, and by connecting in series all of the Flip–Flops of the synapses on a chip, the whole shift register is formed. At a certain time, only one Flip–Flop in the chip has its output high. These Flip–Flops control the read–write sequence for the refreshing of the synapse capacitors. The A/D–D/A pair is shared by all of the synapses in the chip. Its circuit is as shown in Fig. 10. It is a flash–

type conversion, and the comparators are built using current-mode techniques.

V. Simulation Results

We have performed simulations, using HSPICE, of the learning process and of the refreshing circuit, independently. For the learning process, consider first a BAM that has only one neuron per layer.

During the training process and after the presentation of each pair of patterns, the node voltages a_i and b_j have to be reset. In the 1×1 BAM, since its capacity is only one pair of patterns, there is no need to reset a_i and b_j between patterns. However, we did reset it the simulations in order to obtain a feeling of how a bigger circuit would work. If $m_{11} = +1$, the pattern stored is either $a_1 = 1$, $b_1 = 1$ or $a_1 = -1$, $b_1 = -1$. If $m_{11} = 1$, it is either $a_1 = +1$, $b_1 = -1$ or $a_1 = -1$, $b_1 = +1$. Fig. 11 shows the evolution of m_{11}, a_1 and b_1 for the first case.

In Fig. 12, we show the 16 node voltages m_{ij} of a 4×4 BAM, in which 2 pair of patterns have been stored. Note that the steady state m_{ij} values group themselves around three positions. If the m_{ij} capacitors are substituted by voltage sources with their corresponding m_{ij} steady-state values of Fig. 12, simulation shows that originally stored patterns are retrieved.

Conclusions

In this paper we presented a technique for building a VLSI Bidirectional Associative Memory (BAM) that incorporates Hebbian learning and is able to keep the learned patterns after learning for an indefinite amount of time. This storage is based on the fact that learned values allow a certain granularity, and therefore an analog dynamic refreshing technique based on an A/D followed by a D/A conversion is possible. We have verified the models discussed in this paper through computer simulation. Test prototypes of the VLSI implementation have been sent for fabrication and experimental results will soon be reported.

References

[1] B. Kosko, "Adaptive Bidirectional Associative Memories", *Applied Optics*, Vol. 26, No. 23, pp. 4947–4960, 1 December 1987.

[2] B. Kosko, "Bidirectional Associative Memories", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 18, No. 1, pp. 49–60, January/February 1988.

[3] G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", *Computer Vision, Graphics & Image Proc.*, 37, pp. 54–115, 1987.

[4] D. J. Weller and R. R. Spencer, "A Process Invariant Analog Neural Network IC with Dynamically Refreshed Weights", *Proc. Midwest Symposium on Circuits and Systems*, Calgary, 1990.

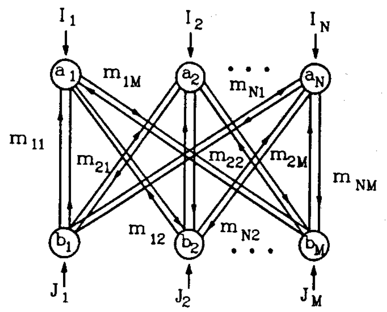

Fig. 1. Basic Kosko's BAM STM Architecture

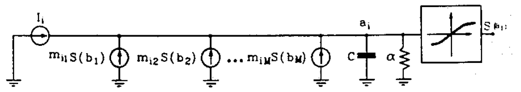

Fig. 2. T-Mode Implementation for Each Neuron's Differential Equation

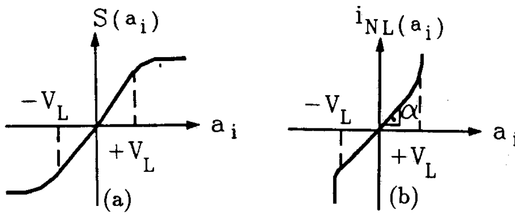

Fig. 3. (a) Shape of Sigmoidal Function
(b) Driving Point Characteristics for each Nonlinear Resistor

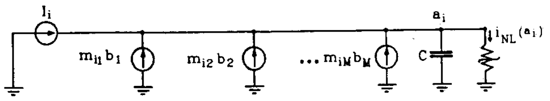

Fig. 4. Modified T-Mode Implementation for Each Neuron's Differential Equation

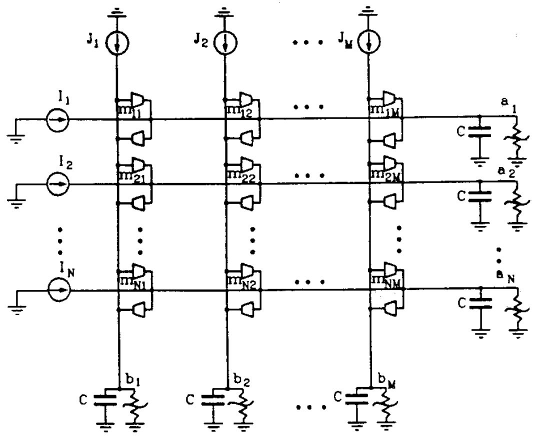

Fig. 5. STM Section of T-Mode Continuous BAM Architecture

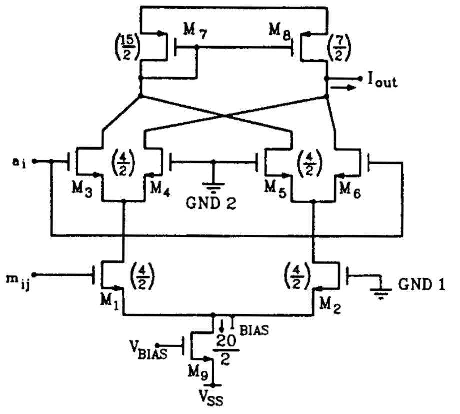

Fig. 6. Transconductance Multiplier, Using an Area of $52\mu m \times 42\mu m$

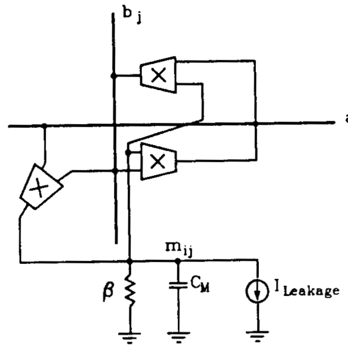

Fig. 7. Circuit for LTM Implementation of Each Synapse

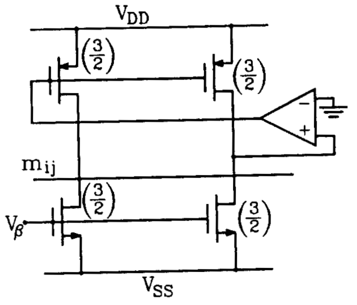

Fig. 8. Load Resistance β for CTM Equation Implementation

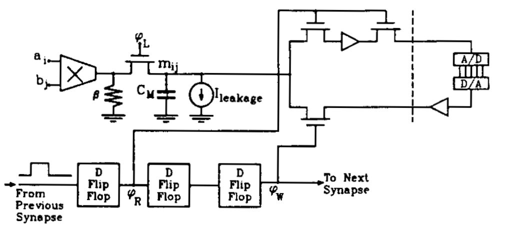

Fig. 9. Dynamic Analog Refreshing Scheme for BAM Weights

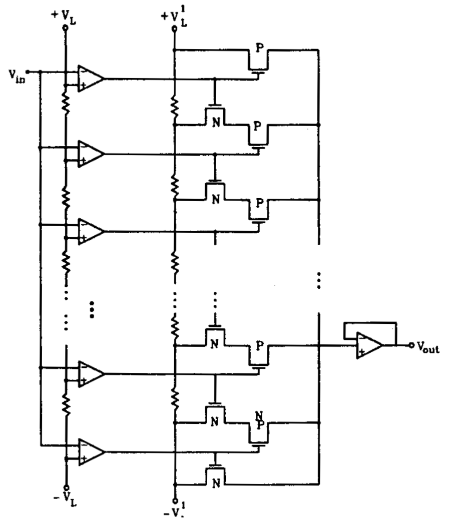

Fig. 10. Complete Current-Mode Flash A/D-D/A Pair.

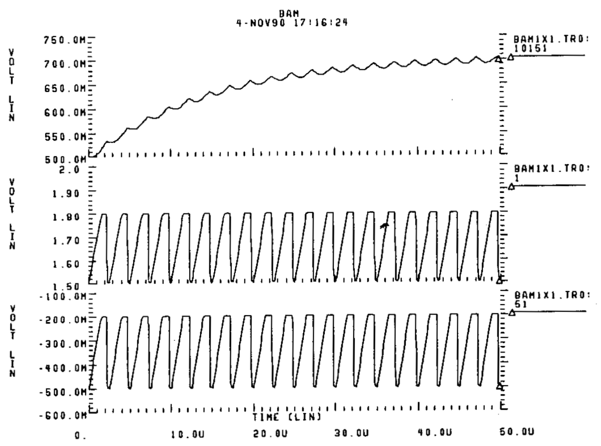

Fig. 11. Voltages for the 1×1 BAM. Top trace is m_{11}, bottom traces are a_1 and b_1.

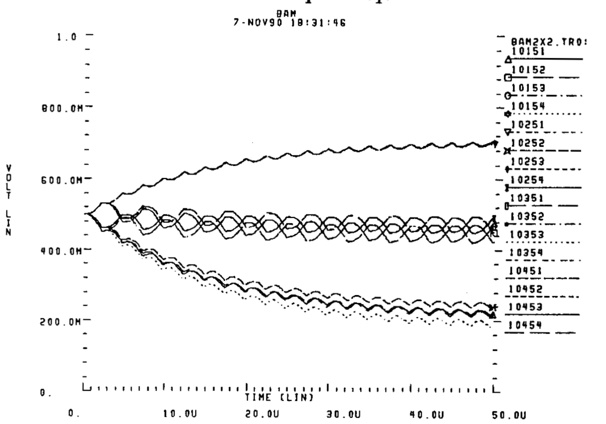

Fig. 12. m_{ij} Node Voltages for a 4×4 BAM