Analog Neural Programmable Optimizers in CMOS VLSI Technologies

R. Domínguez-Castro, A. Rodríguez-Vázquez, J. L. Huertas and E. Sánchez-Sinencio

Abstract—**A 3-μm CMOS IC is presented demonstrating the concept of an analog neural system for constrained optimization. A serial time-multiplexed general-purpose architecture is introduced for the real-time solution of this kind of problem in MOS VLSI. This architecture is a fully programmable and reconfigurable one exploiting SC techniques for the analog part and making extensive use of digital techniques for programmability.**

I. Introduction

THE constrained optimization problem can be defined as one of minimizing a multivariable cost function $\Phi(\boldsymbol{x})$ subjected to a set of constraints $F_k(\boldsymbol{x}) \geqq 0$, $1 \leqq k \leqq Q$. It has been recently demonstrated that the constrained optimization problem can be solved in real time by using analog neural feedback networks [1]–[3]. These networks show potential for those applications where online optimization is required as in robotics, satellite guidance, etc.

Up to now, demonstration of the different proposals for analog neural optimizers has been made either by discrete prototypes [1]–[3] or by simulation results [3]. Previously reported optimizers are, on the other hand, intended to just demonstrate the concept via the solution of problems with fixed weights. Weight programmability is not realistically covered by any of the above-mentioned proposals; this represents an important drawback for the practical use of this kind of circuit.

This paper discusses the implementation of digitally programmable analog neural optimizers using switched-capacitor (SC) integrated circuit techniques. The use of switched capacitors for neural network algorithms has been considered by different authors [3]–[5]. However, the proposed architectures are of the parallel type and, as a consequence, large-area occupation can be expected in case binary-weighted capacitor arrays are used for digital programmability. In this paper we first introduce a parallel SC neural optimizer architecture and discuss area limitations due to the incorportion of programmability issues. Due to these limitations this architecture is only suitable for low dimension problems. Then, a serial time-multiplexed architecture is presented which allows digital control of the weight values with reasonable area figures. Finally, a 3-μm CMOS SC prototype is reported demonstrating the concept of SC analog neural optimizers via an integrated circuit.

Manuscript received December 3, 1991; revised March 3, 1992.

R. Domínguez-Castro, A. Rodríguez-Vázquez, and J. L. Huertas are with the Department of Design of Analog Circuits, Centro Nacional de Microelectrónica, University of Seville, 41012 Sevilla, Spain.

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

IEEE Log Number 9200273.

II. Mathematical Model and Basic SC Architecture

There are many useful optimization problems that can be formulated as linear and quadratic problems. Consider the problem of minimizing a quadratic cost function with linear constraints:

$$\text{minimize } \Phi(\boldsymbol{x}) = \sum_{i=1}^{N} a_i x_i + \frac{1}{2}\left\{\sum_{i=1}^{N}\sum_{j=1}^{N} g_{ij} x_i x_j\right\}, \quad g_{ij} \doteq g_{ji}$$

$$\text{subjected to } F_k(\boldsymbol{x}) = \sum_{i=1}^{N} b_{ki} x_i + b_{k0} \geq 0, \quad 1 \leq k \leq Q. \tag{1}$$

Its solution can be found by using the modified external penalty technique [3], [6]. To this purpose an equivalent unconstrained optimization problem must first be defined having the following cost function:

$$\Psi(\boldsymbol{x}) = U(\boldsymbol{F})\phi(\boldsymbol{x}) + \mu \sum_{k=1}^{Q} U(-F_k)\,|F_k(\boldsymbol{x})| \tag{2a}$$

where μ is a scale factor called penalty multiplier and $U(\boldsymbol{F})$ and $U(-F_k)$ are threshold operators defined as follows:

$$U(-F_k) = \begin{cases} 1, & F_k < 0 \\ 0, & \text{otherwise} \end{cases}$$

$$U(\boldsymbol{F}) = \begin{cases} 1, & F_k \geq 0\ \forall k \\ 0, & \text{otherwise.} \end{cases} \tag{2b}$$

This equivalent unconstrained problem can then be minimized by applying a discrete time gradient strategy,

$$x_i[(n+1)T_C] = x_i(nT_C) - \frac{1}{\tau_0}\left.\frac{\partial \Psi}{\partial x_i}\right|_{t=nT_C}, \quad i = 1, 2, \cdots, N \tag{3}$$

yielding, per state variable,

$$x_i(n+1) = x_i(n) - \frac{1}{\tau_0}\left[U(\boldsymbol{F})\left\{a_i + \sum_{j=1}^{N} g_{ij} x_j(n)\right\}\right.$$

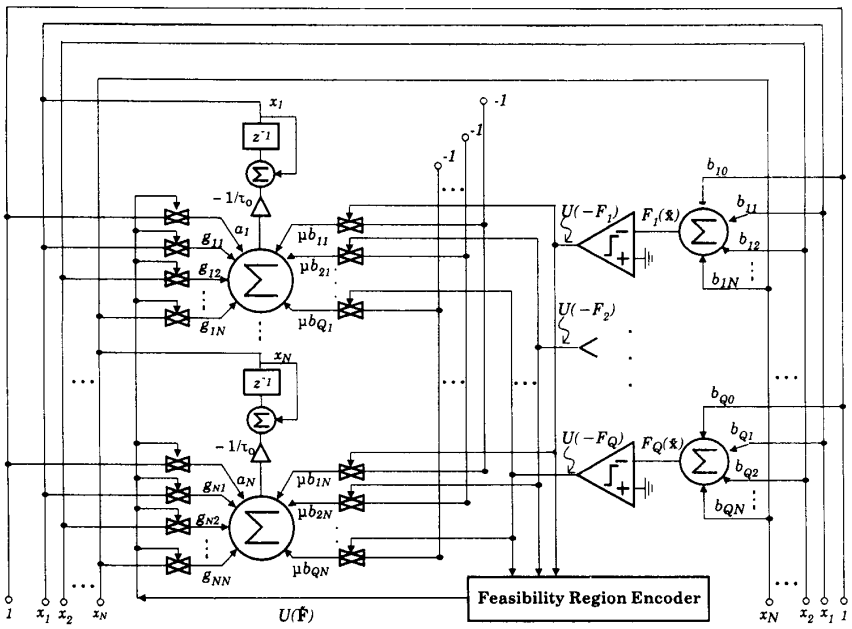

Fig. 1. Fully interconnected parallel neural optimizer architecture.

$$- \mu \sum_{k=1}^{Q} U(-F_k) b_{ki} \Bigg], \qquad i = 1, 2, \cdots, N \tag{4}$$

where T_C, the time interval between consecutive time instances, has been eliminated for simplicity.

The solution of (1) can either be inside the region defined by the problem constraints (feasibility region) or on its border. This solution is obtained by the companion dynamic gradient system in (3) and (4) via the process of seeking the minima of $\Psi(x)$. Notice that the penalty (second term in (2a)) makes $\Psi(x)$ have a valley floor centered around the feasibility region, the valley walls becoming steeper as the penalty multiplier increases. Thus, any trajectory starting outside this region is forced to enter it. Once the trajectory enters the feasibility region, the term due to $\Phi(x)$ in (3) makes the system seek the problem solution point. In case this point is inside the region, the evolution is asymptotic according to the gradient of $\Phi(x)$. Otherwise, a steady state is reached where the trajectory remains confined inside a small interval around the nominal solution point. The size of this interval can be controlled by design [6].

Fig. 1 shows a parallel fully interconnected architecture for the implementation of (4), containing N integrating neurons (shown on the left in Fig. 1) and Q constraint blocks (shown on the right). Fig. 2(a) and (b) shows parasitic insensitive SC schematics for the ith integrating neuron and the kth constraint block, respectively. Notice that these schematics by themselves do only allow positive weight values. Thus, in order for both negative and positive values to be implemented, each input signal has to be multiplied by the sign of the corresponding weight, as is indicated in Fig. 2(a) and (b), where sgn(·) holds for the sign function.

In Fig. 2(a) some switches are controlled by the clock phases while others are controlled by the signals resulting from the logical AND operation between the even clock phase and threshold operator outputs (indicated in the figure by using superscript e). This is done so as to allow modulation of the weights by threshold operators, as required for (4). The controlling signals $U(-F_k)$ are obtained at the outputs of the schematics of Fig. 2(b), while $U(F)$ can be obtained via the logical NOR operation among the different $U(-F_k)$ (represented by the block labeled feasibility region encoder in Fig. 1).

Notice the output of Fig. 2(a) is only valid during the even clock phase at any iteration, while this output has to be used during the odd clock phase at the subsequent iterations. Thus, we must provide a way for the output signals to be stored from the even to the odd clock phases. Also, a mechanism must be provided to get $-x_i$ from x_i, as required for Fig. 2(a) and (b). These functions can be implemented by using the circuit of Fig. 2(c). In a similar way the memory circuit of Fig. 2(d) is required since the outputs of the constraint blocks are obtained using the odd clock phases and used during the even clock phases.

Each weight in the blocks of Fig. 2(a) can be made programmable by just substituting the associated input capacitor by a binary-weighted capacitor array (BWCA), as illustrated in Fig. 2(e) for an M-bit codification of a generic weight w, where $w_{\max}$ is a scaling factor for the weight values. For the parallel inputs approach, however, full digital programmability has the drawback that the capacitor area occupation increases proportionally to $N(N+$

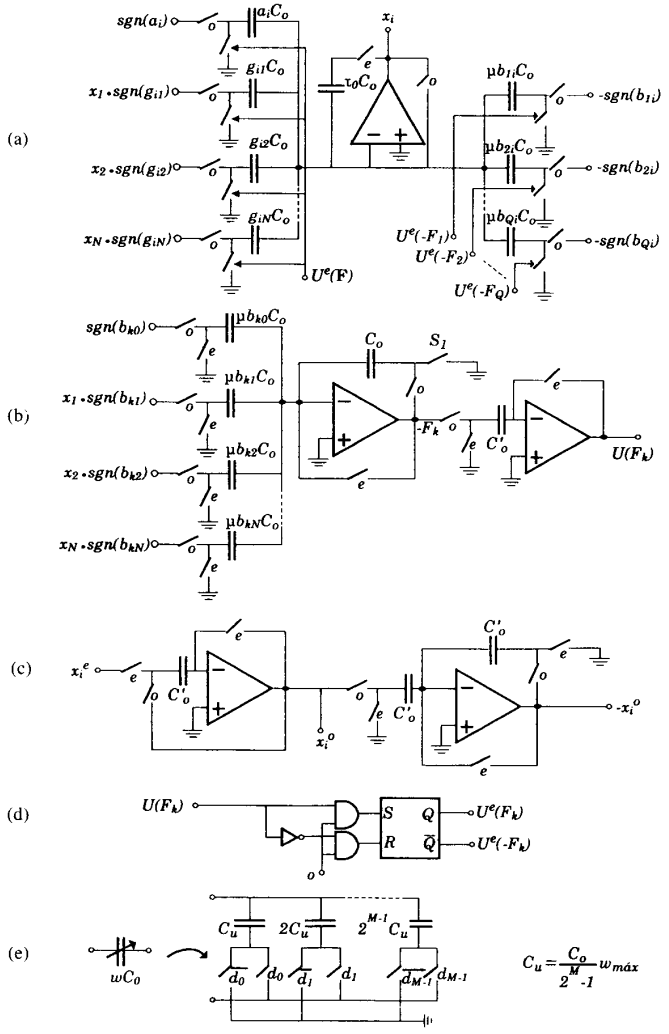

Fig. 2. SC building blocks for constrained optimization circuits.

$2Q)2^M$ (M being the number of bits of the digital word codifying the weights). Also, since $N + Q$ routing lines per neuron and Q per constraint evaluation block are required, routing area increases as $2N^3 + 5N^2Q + 2NQ^3$. As a matter of fact, the latter area term may become the dominant one even for low dimension problems. Taking into account both area terms, we can see that area limitations are severe and hence that, for reasonable yield, a constraint will exist on either the problem dimension or the length of the digital word.

Operation speed introduces additional constraints to the parallel architecture. Notice each neuron output must simultaneously drive $N + Q$ BWCA's as well as large parasitics associated with the routing lines. Hence, operation speed will be penalized when trying to achieve a reasonable power consumption for large dimension problems. On the other hand, stability analysis [3], [6] shows that the model time constant (τ_0 in (4)) must be made to increase as the number of neuron inputs increases, which further penalizes the operation speed for large dimensions.

III. A Serial Architecture for SC Optimizers

Previously discussed drawbacks make fully programmable parallel input architectures, as the one in Fig. 1, appropriate only for low dimension optimization problems (about $N + Q < 10$). For larger dimension problems requiring programmability, a serial input architecture such as the one shown at the conceptual level in Fig. 3(a) can be used. In this architecture, a number ($(N + Q)$ in the more general case) of single-input single-output identical analog processing units are connected together in a neural feedback scheme. Fig. 4(a) shows an SC parasitic insensitive conceptual schematic for the processing units. Notice that just one BWCA per unit is required. Each pro-

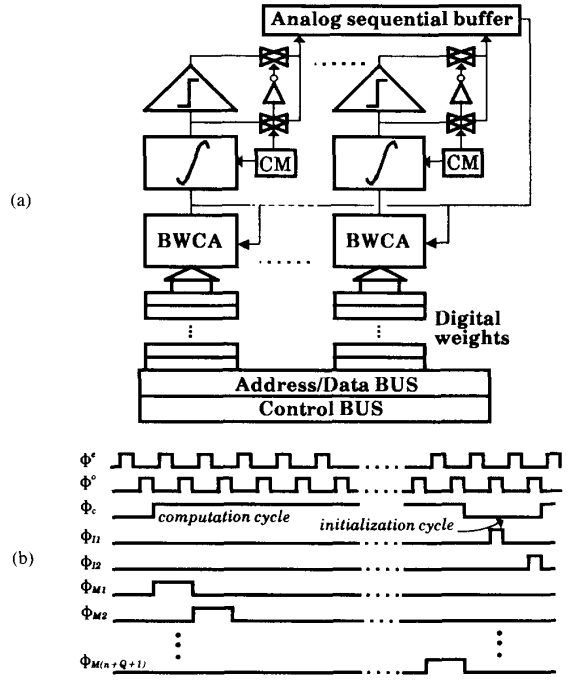

Fig. 3. Concept for a programmable time-multiplexed analog optimizer circuit.

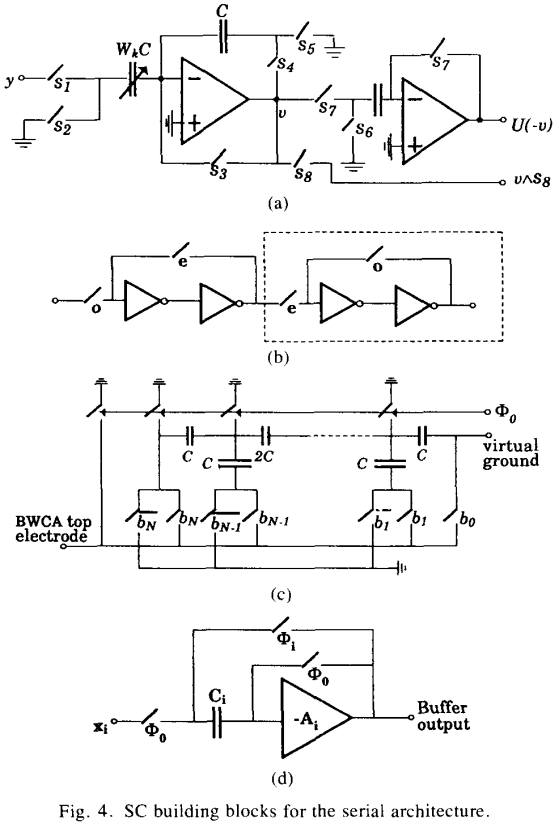

Fig. 4. SC building blocks for the serial architecture.

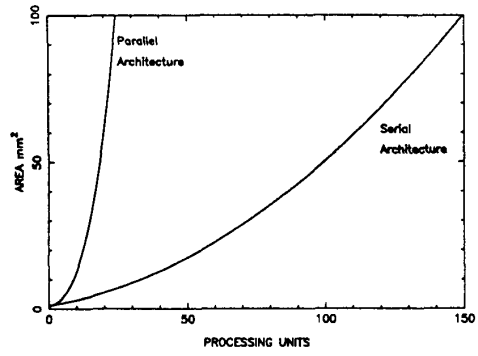

Fig. 5. Comparative area figures for the parallel and the serial architectures.

cessing unit can be configured via the control signals S_1 to S_7 to perform either as an integrating neuron or as a constraint evaluation block.

Each processing unit performs weighted summations in a serial way, under the control of the set of clock signals of Fig. 3(b). At the end of the nth operating cycle (understood as the time required for one iteration of (4)) the corresponding integrating neuron outputs $x_i(n)$ and constraint threshold signals $U\{-F_k(n)\}$ are made available and are stored in the analog sequential buffer. Also, the feedback capacitor of the processing units performing as constraint evaluation blocks is zeroed. At the subsequent operating cycle, the $N + Q$ analog values stored in the sequential buffer are one by one serially provided to the BWCA's driving the integrators, together with their corresponding digital weight codes. For this purpose the digital weights are stored in a cyclic memory. Weighted summations required to evaluate $x_i(n + 1)$ and $U\{-F_k(n + 1)\}$ according to (4) are hence sequentially completed after $N + Q$ clock subcycles.

The digital memory storing the weight codes consists of M shift registers each of length $N + Q$. The schematic for a single stage in these registers is shown in Fig. 4(b). On the other hand, although not explicitly shown in Fig. 4(a), the integrator time constants have to be electrically controlled as well; this is needed for proper solution of the stability versus speed trade-off. Fig. 4(c) shows the schematic to be used for this purpose, consisting of a C-$2C$ ladder which is on the one side connected to the top electrode of the binary-weighted array and, on the other, to the virtual ground of the op amp. With this scheme the feedback capacitor around op amp $OA1$ of the processing unit can be fixed to a small value. This solution results in important area savings as compared to the use of an additional BWCA to control τ_0.

The analog buffer in Fig. 3 is composed of $N + Q$ identical units, one per basic processing unit, hence ensuring complete modularity of the architecture. Fig. 4(d) is a schematic for the ith section of this buffer.

Observe that since the BWCA's in the serial architec-

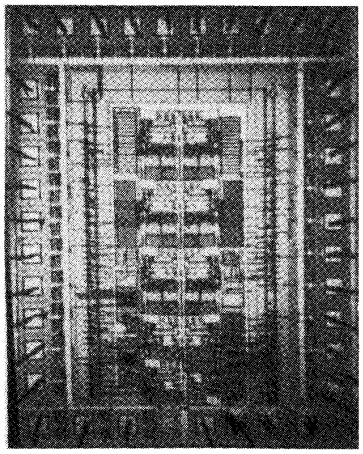

Fig. 6. 3-μm CMOS demonstrator.

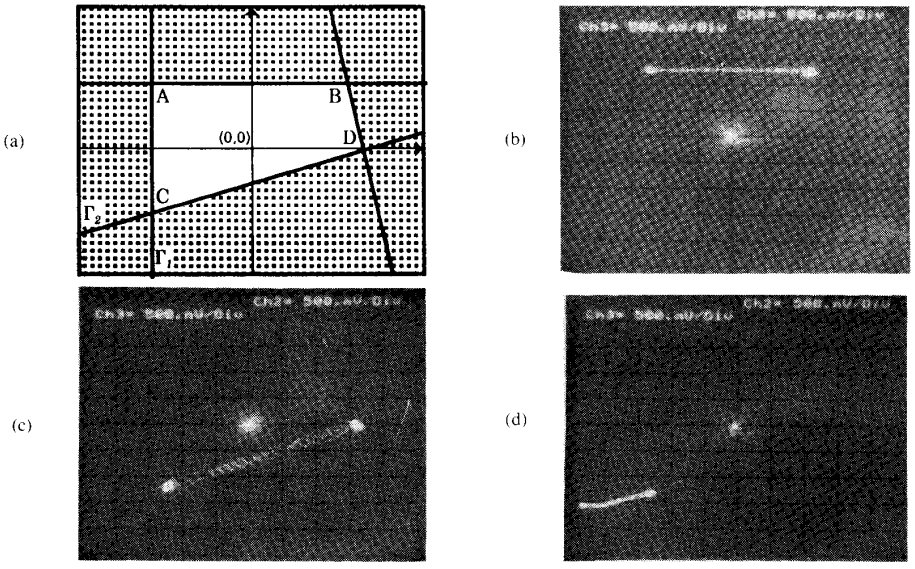

Fig. 7. Experimental results for Fig. 6.

ture are time multiplexed, the capacitor area term for this architecture is proportional to $(N + Q)2^M$. Also, area due to routing increases just as $N + Q$. Fig. 5 shows comparative area figures for the parallel and the serial architectures. Area has been estimated assuming identical op amps and capacitors for both architectures. As can be seen, the area is much lower for the serial architecture than for the parallel one.

Notice also in the serial architecture that there is only one analog signal line simultaneously driving $N + Q$ BWCA's. It allows a much more flexible power versus speed trade-off than in the parallel input case. In fact, for a given power consumption, global operation speed for both the parallel and the serial input architecture may be of the same order of magnitude [6]. Finally, since only one analog input and one analog output are required, the serial architecture makes the layout easier for multichip circuits and wafer-scale integration.

IV. A 3-μm CMOS Demonstrator

Fig. 6 is a microphotograph for a 3-μm CMOS SC neural optimizer prototype. It can be reconfigured for different second-order quadratic and linear problems. Four linear constraints have been implemented defining a trapezoidal feasibility region whose boundaries can also be externally modified. Fig. 7(b)–(d) shows measured trajec-

tories for a case where the feasibility region is as shown in Fig. 7(a). The cost function is externally controlled in such a way that the solution (which corresponds to the equilibrium of the model algorithm) can be on any of the corners of the feasibility region (points labeled A, B, C, and D in Fig. 7(a)). For all the experimental pictures, the point at the origin corresponds to the autozero phase of the SC circuits. In the experiment of Fig. 7(b) the cost function is initially set for its solution at point A. The circuit is allowed to reach equilibrium in this situation. An external control signal is then used to change the cost function in such a way the new theoretical solution coincides with point B. Fig. 7(b) illustrates the evolution of the circuit in the dynamic process of seeking the new solution point starting from the original one.

As theoretically predicted by the model algorithm of (4), the evolution is on the upper border of the feasibility region. Fig. 7(c) is similar to the previous one, the initial and final solution points now being C and D, respectively. Dynamic evolution is now on the lower border of the feasibility region, as expected. Fig. 7(d) is qualitatively different. It corresponds to a case where the cost function is controlled for the theoretical solution to coincide with point C and shows the dynamic circuit evolution from an initial point outside the feasibility region until reaching equilibrium at the theoretical solution point. As can be seen, the trajectory is forced to enter into the feasibility region by first following a direction perpendicular to the unfulfilled constraint border Γ_1 until Γ_2 is reached. From this point, evolution towards equilibrium at C is on Γ_2. Maximum operating frequency for the circuit was about 200 kHz with the time required for the circuit to reach equilibrium in the cases shown in Fig. 7 being about 500 μs for this maximum frequency.

V. Conclusions

As opposed to the general class of networks, the functionality of analog neural-like optimizers does not emerge exclusively from system complexity. Hence, small dimension as well as large dimension problems can be interesting from the application point of view. Results in this paper show the possibility of designing optimizer IC's by using analog/digital circuit techniques. Although the presented prototype is in CMOS technology, the proposed architecture can be implemented as well in any other technology supporting switched-capacitor applications. Optimization strategy in the proposed architectures is based in the use of a penalty gradient technique. However, the architectures can be very easily modified for other strategies such as, for instance, Lagrange multipliers [6]. From a more general point of view, the proposed serial architecture can also be applied to the design of mixed-mode coprocessors for matrix-by-vector multiplication with moderate accuracy (≤ 10 b).

References

[1] D. A. 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. CAS-33, pp. 533–541, May 1986.

[2] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. 35, pp,. 554–562, May 1988.

[3] A. Rodríguez-Vázquez, *et al.*, "Nonlinear switched-capacitor "neural" networks for optimization problems," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 384–397, Mar. 1990.

[4] Y. Tsividis and D. Anastassiou, "Switched-capacitor neural networks," *Electron. Lett.*, pp. 958–959, Aug. 1986.

[5] B. J. Maundy and E. I. El-Masry, "Feedforward associative memory switched capacitor artifical neural networks," *Analog Integrated Circuits and Signals Processing*, vol. 1, pp. 321–338, Dec. 1991.

[6] R. Domínguez-Castro *et al.*, "Modeling and design of analog VLSI "neural" optimizers," in *Proc. 1991 ECCTD*, pp. 489–497.