This implies that $|v_{n+1}(t)|$ is bounded in the time interval of existence. Based on (11), $|v_i(t)|$ is also bounded for all $1 \le i \le n$. Therefore, $v(t)$ and $\psi(t)$ can be extended to $[0, +\infty)$. From (11), we have

$$\lim_{t \to \infty} \left| \frac{v_i(t)}{v_{n+1}(t)} \right| = 0, \qquad 1 \le i \le n. \tag{15}$$

From (12), we have

$$\lim_{t \to \infty} \; v_{n+1}(t) = \frac{-1}{c_{n+1,n+1}} \tag{16}$$

and hence

$$\lim_{t \to \infty} \; \psi(t) = \frac{-\boldsymbol{c}_{n+1}}{c_{n+1,n+1}}. \tag{17}$$

Therefore, $-\boldsymbol{c}_{n+1}/c_{n+1,n+1}$ is a local stable point of (1) in the case of $\psi_{n+1}(0) = -1$. ■

In the following two theorems, we assume that $c_{n+1,n+1} > 0$ and $c_{i,n+1} \ge 0$ for all $1 \le i \le n$ without loss of generality (when $c_{i,n+1} < 0$, we can simply replace $\boldsymbol{c}_i$ by $-\boldsymbol{c}_i$).

Theorem 4: If $\psi_{n+1}(0) = -1, \boldsymbol{c}_{n+1}^T \psi(0) < 0$ and $\boldsymbol{c}_i^T \psi(0) \le 0$ for all $1 \le i \le n$, then the solution $\psi(t)$ of (1) can be extended to $[0, +\infty)$ and $\lim_{t \to +\infty} \; \psi(t) = -\boldsymbol{c}_{n+1}/c_{n+1,n+1}$.

Proof: If the above condition holds, $v_{n+1}(0) < 0$ and $v_i(0) \le 0$ for $1 \le i \le n$, and hence $v_{n+1}(t) < 0$ and $v_i(t) \le 0$ $(1 \le i \le n)$ for all t in the time interval of existence. From (12), we have

$$-\frac{1}{v_{n+1}(t)} \ge c_{n+1,n+1} \tag{18}$$

and hence $v_{n+1}(t)$ is bounded in the time interval of existence. From (11), we see that $v_i(t)$ is also bounded for all $1 \le i \le n$. Therefore, $v(t)$ and $\psi(t)$ can be extended to $[0, +\infty)$. Similar to the proof for Theorem 3, we have $\lim_{t \to +\infty} \; \psi(t) = -\boldsymbol{c}_{n+1}/c_{n+1,n+1}$. ■

Theorem 5: If $\psi_{n+1}(0) = -1, \boldsymbol{c}_{n+1}^T \psi(0) < 0$ and $\boldsymbol{c}_i^T \psi(0) \ge 0$ for all $1 \le i \le n$, then the solution of $\psi(t)$ of (1) can be extended to $[0, +\infty)$ and $\lim_{t \to +\infty} \; \psi(t) = -\boldsymbol{c}_{n+1}/c_{n+1,n+1}$.

Proof: If the above condition holds, $v_{n+1}(t) < 0$ and $v_i(t) \ge 0$ $(1 \le i \le n)$ for all t in the time interval of existence. From (12), we have

$$c_{n+1,n+1} \ge -\frac{1}{v_{n+1}(t)}. \tag{19}$$

Since $|\zeta_i(t)|$ is a decreasing function of t and $\zeta_i(t) \le 0$ for all $1 \le i \le n$ [see (12)], $v_{n+1}(t)$ is an increasing function of t, and hence $v_{n+1}(t)$ is bounded. As shown in the proof for Theorem 3, $\psi(t)$ can be extended to $[0, +\infty)$ and $\lim_{t \to +\infty} \; \psi(t) = -\boldsymbol{c}_{n+1}/c_{n+1,n+1}$. ■

REFERENCES

[1] K. Gao, M. O. Ahmad, and M. N. S. Swamy, "A constrained anti-Hebbian learning algorithm for total least-square estimation with application to adaptive FIR and IIR filtering," *IEEE Trans. Circuits Syst. II*, vol. 41, pp. 718–729, Nov. 1994.

[2] L. Ljung, "Analysis of recursive stochastic algorithms," *IEEE Trans. Automat. Contr.*, vol. AC-22, no. 4, pp. 551–575, 1977.

[3] F. Verhulst, *Nonlinear Differential Equations and Dynamic Systems*. Berlin, Germany: Springer-Verlag, 1985.

A New Synthesis Procedure for a Class of Cellular Neural Networks with Space-Invariant Cloning Template

Zanjun Lu and Derong Liu

***Abstract*— This paper presents a new synthesis procedure (design algorithm) for cellular neural networks (CNN's) with a space-invariant cloning template with applications to associative memories. In the present synthesis procedure, the design problem is formulated as a set of linear inequalities, and the inequalities are solved using the well-known perceptron training algorithm. When desired memory patterns are given by a set of bipolar vectors, it is guaranteed that a cellular neural network with a space-invariant cloning template can be designed using the design algorithm developed herein. An algorithm is also provided to design CNN's with space-invariant cloning templates and with symmetric connection matrices to guarantee the global stability of the network. Two specific examples are included to demonstrate the applicability of the methodology developed herein.**

***Index Terms*—Associative memories, cellular neural networks, perceptron training.**

I. INTRODUCTION

Cellular neural networks, first introduced in 1988 [4], are of great interest due to the fact that such networks are among the easiest to implement in hardware. Cellular neural networks include the class of feedback neural networks with local interconnections and they are also suitable for very large-scale integration (VLSI) implementations of associative memories. In this paper, we consider the realization of associative memories via the class of (zero-input) cellular neural networks introduced in [4]. The goal of associative memories is to store a set of desired patterns as stable memories such that a stored pattern can be retrieved when the initial pattern contains sufficient information about that pattern. In practice the desired memory patterns are usually represented by bipolar vectors (or binary vectors). Cellular neural networks have been considered for associative memories in the literature [2], [5], [8], [9], [11], [14]. In these works, cellular neural networks with space-varying cloning templates are considered.

In [6], the synthesis of cellular neural networks with *space-invariant* cloning templates is considered; a design algorithm based on the eigenstructure method [12] is developed. In the present paper, a *new* synthesis procedure (design algorithm) for cellular neural networks with space-invariant cloning templates will be developed. The present design algorithm is developed based on the well-known perceptron training algorithm, and the convergence of the design algorithm is guaranteed, i.e., it is certain to obtain a space-invariant cloning template for cellular neural networks using the present design algorithm. The present paper also provides a solution to the cloning template design of cellular neural networks with *symmetric* interconnections.

A class of *two-dimensional, continuous-time, zero-input cellular neural networks* is described by equations of the form (cf. [3], [4])

$$\begin{cases} \dot{v}_{xij} = -\dfrac{1}{R_x} v_{xij} + \displaystyle\sum_{C(k,l) \in N_r(i,j)} W_{ij,kl} \; v_{ykl} + I_{ij} \\ v_{yij} = \mathrm{sat}(v_{xij}) \end{cases} \tag{1}$$

Manuscript received April 28, 1997; revised April 10, 1998. This work was supported by Stevens Institute of Technology. This paper was recommended by Associate Editor J. M. Zurada.

The authors are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030-5991 USA.

Publisher Item Identifier S 1057-7130(98)08500-0.

where $1 \le i \le M$, $1 \le j \le N$, $R_x > 0$, $\mathrm{sat}(v_{xij}) = \frac{1}{2}(|v_{xij}+1| - |v_{xij}-1|)$, and v_{xij} represents the states of the network, $\dot{v}_{xij}$ denotes the derivative of v_{xij} with respect to time t, and v_{yij} represents the outputs of the network. The basic unit in a cellular neural network is called a *cell*. In (1), there are $M \times N$ such cells arranged in an $M \times N$ array. The cell in the ith row and the jth column is denoted by $C(i,j)$, and an *r-neighborhood* $N_r(i,j)$ of the cell $C(i,j)$ for a positive integer r is defined by $N_r(i,j) \triangleq \{C(k,l)\colon \max\{|k-i|, |l-j|\} \le r,\ 1 \le k \le M,\ 1 \le l \le N\}$. $W_{ij,kl}$ denotes the connection weight from cell $C(k,l)$ to cell $C(i,j)$. I_{ij} is the bias term for cell $C(i,j)$.

In order to write (1) in a matrix-vector format, we denote $W = [W_{ij,kl}] \in \Re^{MN \times MN}$, where $W_{ij,kl}$ is defined in (1), and we let $T = W = [T_{ij}] \in \Re^{n \times n}$, where $n = M \times N$. System (1) can now be written as

$$\begin{cases} \dot{x} = -x + T\ \mathrm{sat}(x) + I \\ y = \mathrm{sat}(x) \end{cases} \tag{2}$$

where $x = [v_{x11}, v_{x12}, \cdots, v_{x21}, \cdots, v_{xMN}]^T \in \Re^n$ is the state vector, $y = [v_{y11}, v_{y12}, \cdots, v_{y21}, \cdots, v_{yMN}]^T \in D^n \triangleq \{x \in \Re^n\colon -1 \le x_i \le 1, i = 1, \cdots, n\}$ is the output vector, $T = [T_{ij}] \in \Re^{n \times n}$ is the coefficient (or connection) matrix, $I \in \Re^n$ is a bias vector, and $\mathrm{sat}(x) = [\mathrm{sat}(x_1), \cdots, \mathrm{sat}(x_n)]^T$. Note that in (2), without loss of generality, we choose $R_x = 1$. Also note that in (2), T is a sparse matrix depending on the neighborhood radius r (see examples in [8] and [9]). Using the notation in system (2), it is implied that the boundaries of the $M \times N$ array are not periodic, i.e., cells on the boundaries will have fewer connections than cells in the interior of the array.

The dynamic rule of a cellular neural network can be completely specified by a $(2r+1) \times (2r+1)$ matrix which is called *cloning template*, or simply, *template*, and a bias term. For example, for a cellular neural network with a neighborhood radius $r = 1$, the cloning template of this network is of size 3×3; thus, in order to completely describe the dynamics of the network, we only need to specify the *nine* connection weights in the template (cf. [4]) and *one* bias term. In this case, the cloning template and the bias term are *space-invariant*. One of the advantages of considering a space-invariant cloning template for cellular neural networks is the small number of different connection weights required for describing a cellular neural network which can simplify the hardware implementation process [1], [3].

In Section II, we develop algorithms for space-invariant cloning template design. In Section III, two specific examples are considered to demonstrate the applicability of the present results.

II. Space-Invariant Cloning Template Design

A vector α will be called a *(stable) memory vector* (or simply, a *memory*) of system (2) if $\alpha = \mathrm{sat}(\beta)$ and if β is an asymptotically stable equilibrium point of system (2). Use B^n to denote the set of n-dimensional *bipolar vectors*, i.e., $B^n \triangleq \{x \in \Re^n\colon x_i = 1 \text{ or } -1, i = 1, \cdots, n\}$. For $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_n]^T \in B^n$, define $C(\alpha) = \{x \in \Re^n\colon x_i \alpha_i > 1, i = 1, \cdots, n\}$. A result for bipolar memories established in [8] and [9] states the following.

Lemma 2.1: If $\alpha \in B^n$ and if

$$\beta = T\alpha + I \in C(\alpha) \tag{3}$$

then (α, β) is a pair of stable memory vectors and an asymptotically stable equilibrium point of (2). ■

The following describes the design problem of cellular neural networks for associative memories.

Design Problem: Given a neighborhood radius r for the cellular neural networks described by (2) and m vectors $\alpha^1, \cdots, \alpha^m \in B^n$, choose T and I such that $\alpha^1, \cdots, \alpha^m$ are stable memory vectors of system (2). ■

To solve the design problem, one needs to determine T and I from (3) with $\alpha = \alpha^k, k = 1, 2, \cdots, m$, i.e., one needs to determine T and I from

$$T\alpha^k + I \in C(\alpha^k), \quad \text{for } k = 1, 2, \cdots, m. \tag{4}$$

Condition (4) can be equivalently written as

$$\begin{cases} T_i\alpha^k + I_i > 1, & \text{if } \alpha_i^k = 1 \\ T_i\alpha^k + I_i < -1, & \text{if } \alpha_i^k = -1 \end{cases} \tag{5}$$

for $k = 1, 2, \cdots, m$ and for $i = 1, 2, \cdots, n$; where T_i represents the ith row of T, I_i denotes the ith element of I, and α_i^k is the ith entry of α^k. From (5) [or equivalently, from (4)], a synthesis algorithm based on the Perceptron Training Algorithm [13] is obtained in [7]. From (5), one can see that the design problem of neural networks is essentially equivalent to designing n threshold logic functions of $n+1$ variables ($T_{i1}, T_{i2}, \cdots, T_{in}$ and I_i for $i = 1, 2, \cdots, n$). This observation motivates the choice of the perceptron training algorithm over other techniques (cf. [2], [8], [9], [11], and [14]) for solving the neural network design problem in [7] and in the present paper.

For a cellular neural network described by (2) with $n = M \times N$ cells and with neighborhood radius r, one needs a $(2r+1) \times (2r+1)$ (space-invariant) cloning template to describe the connection matrix T. Use $p_l, l = 1, 2, \cdots, (2r+1)^2$ to denote these elements and write

$$P_T = \begin{bmatrix} p_1 & p_2 & \cdots & p_{(2r+1)} \\ p_{(2r+1)+1} & p_{(2r+1)+2} & \cdots & p_{2\times(2r+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & p_{(2r+1)^2} \end{bmatrix} \in \Re^{(2r+1)\times(2r+1)}.$$

If a cellular neural network (2) has a cloning template P_T, the connection matrix T will consist of only zeros and p_l's. Given a cloning template P_T, the connection matrix T can be easily generated according to the interconnecting structure specified by the neighborhood radius r.

From the given $\alpha^k, k = 1, 2, \cdots, m$, define

$$\gamma^{ki} = \left[\gamma_1^{ki}, \gamma_2^{ki}, \cdots, \gamma_{(2r+1)^2}^{ki}\right]^T \tag{6}$$

for $k = 1, 2, \cdots, m$ and for $i = 1, 2, \cdots, n$ with

$$\gamma_l^{ki} = \begin{cases} \alpha_j^k, & \text{if the } l\text{th connection weight of} \\ & \text{the } i\text{th cell corresponds to the } j\text{th} \\ & \text{element on the } i\text{th row of } T \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

If the connection matrix T has a space-invariant cloning template P_T and a space-invariant bias term b, (5) can be written as

$$\begin{cases} [p_1, p_2, \cdots, p_{(2r+1)^2}]\gamma^{ki} + b > 1, & \text{if } \gamma_c^{ki} = 1 \\ [p_1, p_2, \cdots, p_{(2r+1)^2}]\gamma^{ki} + b < -1, & \text{if } \gamma_c^{ki} = -1 \end{cases} \tag{8}$$

for $k = 1, 2, \cdots, m$ and for $i = 1, 2, \cdots, n$, where γ_c^{ki} denotes the center element of γ^{ki} and

$$c = \frac{(2r+1)^2 + 1}{2}. \tag{9}$$

We note that (8) is simply a rearrangement of (5) where T and I are determined from P_T and b.

The following is a synthesis algorithm for the design of cellular neural networks with space-invariant cloning templates.

Synthesis Algorithm 2.1: Using the Perceptron Training Algorithm, obtain one perceptron $W = [w_1, w_2, \cdots, w_{(2r+1)^2+1}]$, such that

$$\begin{cases} W\overline{\gamma}^{ki} > 0, & \text{if } \gamma_c^{ki} = 1 \\ W\overline{\gamma}^{ki} < 0, & \text{if } \gamma_c^{ki} = -1 \end{cases}$$

for $k = 1, 2, \cdots, m$ and for $i = 1, 2, \cdots, n$, where

$$\overline{\gamma}^{ki} = \begin{bmatrix} \gamma^{ki} \\ \cdots \\ 1 \end{bmatrix}$$

and γ^{ki} are determined by (6) and (7).

For $l = 1, 2, \cdots, (2r+1)^2$, choose $p_l = w_l$ if $l \neq c$, $p_c = w_c + \mu$ with $\mu > 1$, and set $I = [I_1, I_2, \cdots, I_n]^T$ with $I_i = w_{(2r+1)^2+1}$ for $i = 1, 2, \cdots, n$, where c is given in (9). Then construct T from P_T according to the interconnecting structure specified by the neighborhood radius r. ■

Remark 2.1: The above synthesis algorithm will certainly result in a space-invariant cloning template and a space-invariant bias term with any learning rate $\eta > 0$ in the Perceptron Training Algorithm. Considering the fact that all the desired patterns are bipolar vectors, the convergence of the training iteration can also be proved following similar steps as in [7]. For the design problem considered in the present paper, a trivial solution, i.e., a template with only a center nonzero element, always exists. Our simulation shows that one can always obtain nontrivial solutions by choosing either zero initial weights or random initial weights in the perceptron training algorithm. ■

Remark 2.2: It is shown in [7] that when the diagonal elements of the connection matrix $T_{ii} \geq 1$, the system (2) will have no stable memory points except the vertices of the n-dimensional hypercube. It is also shown in [7] that when $T_{ii} \leq 1$, the system (2) exhibits some good characteristics. For instance, none of the vectors $\gamma \in B^n$ with Hamming distance $H(\gamma, \alpha) = 1$ from any stable memory α can be stable when $T_{ii} \leq 1$. Therefore, one reasonable solution to reduce the number of spurious memories is to have $T_{ii} = 1$. Experimental results show that neural networks which satisfy the constraints $T_{ii} = 1$ usually have fewer spurious (parasitic) memories than networks which do not satisfy these constraints (cf. examples in [7] and Example 2 in Section III of the present paper). ■

Remark 2.3: It can easily be proved that a cloning template design with the diagonal elements of the connection matrix T satisfying $T_{ii} = 1$, for $i = 1, 2, \cdots, n$, can be obtained if and only if $w_c < 0$ from the Synthesis Algorithm 2.1. If the algorithm ends up with $w_c \geq 0$, then one can repeat the perceptron training with an initial weight W chosen as $W = [w_1, \cdots, w_{c-1}, -\xi, w_{c+1}, \cdots, w_{(2r+1)^2+1}]$ with $\xi > 0$. Experiments show that this will usually end up with $w_c < 0$ if such a solution exists. ■

The Synthesis Algorithm 2.1 presented above will usually result in a nonsymmetric coefficient matrix T since the resulting cloning template is usually not with the special structure to render a symmetric T. A symmetric design is of great interest since a cellular neural network (2) will be globally stable when T is symmetric (cf. [4], [9]). We will develop in the following a design algorithm of cellular neural networks with symmetric connections and with a space-invariant cloning template.

In the sequel, the following notation will be used. For a matrix $F = [f_{ij}] \in \Re^{n \times n}$, denote

$$F^r \triangleq BFB \tag{10}$$

where $B = [b_{ij}] \in \Re^{n \times n}$ with

$$b_{ij} = \begin{cases} 1, & \text{if } i + j = n + 1 \\ 0, & \text{otherwise.} \end{cases}$$

The following result provides a simple criterion for symmetric cellular neural networks to have a space-invariant cloning template. Its proof is straightforward.

Theorem 2.1: The connection matrix T with space-invariant cloning template P_T is symmetric if $P_T^r = P_T$. ■

For the P_T determined from the Synthesis Algorithm 2.1 with $\mu > 1$, choose $\Delta P_T = (P_T^r - P_T)/2$. Then, $\overline{P}_T \triangleq P_T + \Delta P_T = (P_T + P_T^r)/2$ satisfies $\overline{P}_T^r = \overline{P}_T$; and the matrix T_s which is computed from $\overline{P}_T$ will be symmetric (Theorem 2.1). From [8] and [10], one knows that if $\|\Delta T\|_\infty = \|T_s - T\|_\infty < \nu - 1$, where ν is obtained as

$$\nu = \min_{1 \leq k \leq m} \{\delta(\beta^k)\} > 1 \text{ with } \beta^k = T\alpha^k + I, k = 1, \cdots, m \tag{11}$$

the neural network (2) with a symmetric connection matrix T_s will also store all the desired patterns as memories. The notation $\delta(x) = \min_{1 \leq i \leq n} \{|x_i|\}$ for $x \in \Re^n$ is used above.

Based on the above observation, after obtaining the cloning template P_T and the bias vector I from the Synthesis Algorithm 2.1, one can proceed as follows to get a symmetric design.

Symmetric Design Algorithm 2.2:

1) Use the Synthesis Algorithm 2.1 to obtain a space-invariant cloning template P_T.
2) Compute P_T^r as in (10) from the cloning template P_T. If $P_T^r = P_T$, stop.
3) Generate the connection matrix T using P_T. Generate T_1 using P_T^r in the same way.
4) Compute $\beta^k = T\alpha^k + I$ for $k = 1, 2, \cdots, m$ and let $\nu = \min_{1 \leq k \leq m} \{\delta(\beta^k)\}$.
5) If $\nu \leq 1 + \eta$, stop, where η is a small positive number (e.g., $\eta = 0.01$). Otherwise, goto step 6).
6) Compute $\Delta T = T_1 - T$. If $\|\Delta T\|_\infty < \nu - 1$, choose $\lambda = 1$. Otherwise, choose $\lambda = (\nu - 1)/\|\Delta T\|_\infty - \varepsilon$, where ε is a small positive number (e.g., $\varepsilon = 0.001$).
7) Determine the cloning template $\overline{P}_T$ as $\overline{P}_T = P_T + \lambda(P_T^r - P_T)/2$.
8) Replacing P_T by $\overline{P_T}$, go to step 2.

If the algorithm ends up with $P_T^r = P_T$, a solution has been found for the symmetric design problem. If the algorithm ends up with $\nu \leq 1 + \eta$ and $P_T^r \neq P_T$, the symmetric design algorithm is not successful in solving a symmetric T for the given problem. ■

Remark 2.4: The robustness analysis result of [8] and [10] is applied iteratively in the Symmetric Design Algorithm 2.2. The idea is to gradually approach the cloning template which renders a symmetric connection matrix [steps 6) and 7)]. The robustness analysis result of [8] and [10] guarantees that the desired patterns $\alpha^1, \alpha^2, \cdots, \alpha^m$ will still be stable memory vectors of the neural network after the parameter perturbation given by $\lambda(P_T^r - P_T)/2$ in step 7. ■

III. Examples

Example 1: The same example as in [11] will be used. Consider a design of a cellular neural network (2) with 36 neurons ($n = 36$) with $M = 6$, $N = 6$, and $r = 1$. The objective is to store the six patterns shown in Fig. 1 as stable memories (black = 1 and white = -1).

Case I: Nonsymmetric Design. The Synthesis Algorithm 2.1 is utilized to design a cellular neural network. The matrix T obtained has a cloning template

$$P_T = \begin{bmatrix} 3.6 & 3.2 & 5.4 \\ 1 & 14.5 & 0 \\ 5.2 & 2.4 & 6.3 \end{bmatrix}$$

with $I = [4.3, 4.3, \cdots, 4.3]^T$.

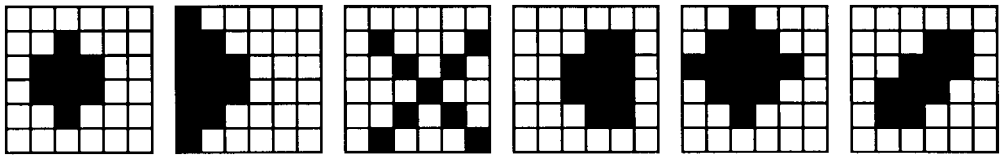

Fig. 1. The six desired memory patterns (6 × 6) for Example 1.

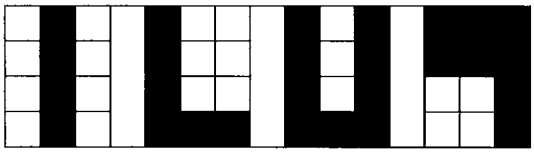

Fig. 2. The four desired memory patterns (4× 3) for Example 2.

A comparative study between the present space-invariant cloning template design and the space-varying designs [9], [11] indicated that the cellular neural network with the space-invariant cloning template has smaller basins of attraction for each desired pattern than those with space-varying cloning template designed as in [9] and [11].

Case II: Symmetric Design. Starting from the cloning template obtained in Case I, a symmetric design is obtained using the Symmetric Design Algorithm 2.2. Matrix T in this case has a cloning template given by

$$P_T = \begin{bmatrix} 4.95 & 2.85 & 5.3 \\ 0.5 & 14.5 & 0.5 \\ 5.3 & 2.85 & 4.95 \end{bmatrix}. \tag{12}$$

It can be verified by Lemma 2.1 that the six desired patterns are also stable memory vectors for system (2) with a symmetric connection matrix generated from the template (12). Notice that the structure of the cloning template in (12) is a typical one for $r = 1$ which renders a *symmetric interconnection matrix*. We can see that this cloning template requires symmetric connections in vertical, horizontal, as well as diagonal directions from any cell unit in the network.

Simulation results of this example indicate that the present results are comparable to the ones reported in [6] even though space-varying bias terms are used in [6]. The next example, Example 2, shows that the present synthesis methods can also control the diagonal elements of the connection matrix T which cannot be done using the design method developed in [6]. ■

Example 2: The objective of this example is to demonstrate the capability of the Synthesis Algorithm 2.1 to satisfy the constraints on the diagonal elements (Remark 2.2) if such a solution exists. It also shows the effect of diagonal elements on the number of spurious memories. Results are compared to fully connected neural networks designed using the algorithm of [7].

In this example, a cellular neural network (2) with 12 neurons $(n = 12)$ is required for the realization of an associative memory. The cellular neural network is arranged in a 4 × 3 array. Four desired patterns (Fig. 2) are to be stored in the network designed by using the Synthesis Algorithm 2.1 combined with Remark 2.3. A cloning template whose center element is one can be found as

$$P_T = \begin{bmatrix} -2.205 & 5.98 & -2.55 \\ 5.255 & 1 & 4.755 \\ -5.345 & 7.12 & -3.99 \end{bmatrix}$$

and $I_i = 1.825$ for $i = 1, 2, \cdots, n$. The above cloning template guarantees that $T_{ii} = 1$ for $i = 1, 2, \cdots, n$.

The performance of this network is illustrated by means of a typical simulation run of (2), shown in Fig. 3. In this case, the (noisy) initial pattern is generated by reversing five bits in the desired pattern. Convergence occurs in 14 steps with step size $h = 0.04$.

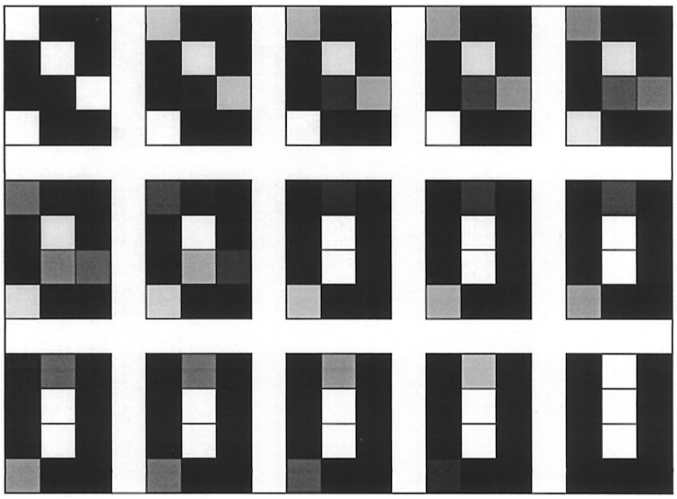

Fig. 3. A typical evolution of pattern 3 of Fig. 2.

TABLE I

T_{ii}	0	1	2	3	4
Total number of spurious memories (the present algorithm)	22	22	50	81	111
Total number of spurious memories (algorithm in [7])	4	4	4	5	6

Choosing the diagonal component T_{ii} to be 0, 1, 2, 3, and 4, respectively, the number of the spurious memory vectors for each case can be computed using the results in [9]. One can see that the number of spurious memory vectors decreases as $T_{ii} \geq 1$ decreases. Table I records the experimental results. It is noted that in Table I, the number of spurious memories is for the entire 12-dimensional space (4 × 3 array). The comparison with fully connected neural networks designed using the algorithm of [7] is also shown in the table. One can see that the number of spurious memories in a fully connected neural network is less than that in a cellular neural network.

We note that in Table I, in order for the comparison to be meaningful, we choose to design neural networks using the two algorithms with the same μ, which is a parameter to control the robustness of the design (see [7] for details about the parameter μ). We also note that when $T_{ii} < 1$ in Table I, nonbipolar spurious memories do exist in the designed cellular neural networks. Although the performance of the cellular neural network (2) is excellent (compared to [6]) with diagonal element equal to one, the one-diagonal design is not always obtainable. The conditions under which such a solution will exist are currently under investigation by the authors.

REFERENCES

[1] M. Anguita, F. J. Pelayo, A. Prieto, and J. Ortega, "Analog CMOS implementation of a discrete time CNN with programmable cloning

templates," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 215–218, Mar. 1993.
[2] M. Brucoli, L. Carnimeo, and G. Grassi, "Discrete-time cellular neural networks for associative memories with learning and forgetting capabilities," *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 396–399, July 1995.
[3] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 147–156, Mar. 1993.
[4] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257–1272, Oct. 1988.
[5] B. Ling and F. M. A. Salam, "A cellular network formed of Hopfield networks," in *Proc. 35th Midwest Symp. Circuits Systems*, Washington, DC, Aug. 1992, pp. 256–259.
[6] D. Liu, "Cloning template design of cellular neural networks for associative memories," *IEEE Trans. Circuits Syst. I*, vol. 44, pp. 646–650, July 1997.
[7] D. Liu and Z. Lu, "A new synthesis approach for feedback neural networks based on the perceptron training algorithm," *IEEE Trans. Neural Networks*, vol. 8, pp. 1468–1482, Nov. 1997.
[8] D. Liu and A. N. Michel, *Dynamical Systems with Saturation Nonlinearities: Analysis and Design*. New York: Springer-Verlag, 1994.
[9] ____, "Sparsely interconnected neural networks for associative memories with applications to cellular neural networks," *IEEE Trans. Circuits Syst. II*, vol. 41, pp. 295–309, Apr. 1994.
[10] ____, "Robustness analysis and design of a class of neural networks with sparse interconnecting structure," *Neurocomputing*, vol. 12, pp. 59–76, June 1996.
[11] G. Martinelli and R. Perfetti, "Associative memory design using space-varying cellular neural networks," in *Proc. 2nd Int. Workshop on Cellular Neural Networks and Their Applications*, Munich, Germany, Oct. 1992, pp. 117–122.
[12] A. N. Michel and J. A. Farrell, "Associative memories via artificial neural networks," *IEEE Contr. Syst. Mag.*, vol. 10, pp. 6–17, Apr. 1990.
[13] M. L. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: M.I.T. Press, 1969 and 1988, expanded version.
[14] G. Seiler, A. J. Schuler, and J. A. Nossek, "Design of robust cellular neural networks," *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 358–364, May 1993.

An 8-bit CMOS 3.3-V–65-MHz Digital-to-Analog Converter with a Symmetric Two-Stage Current Cell Matrix Architecture

Ji Hyun Kim and Kwang Sub Yoon

***Abstract*—This paper describes a 3.3-V–65-MHz 8-bit CMOS digital-to-analog converter (DAC) with two-stage current cell matrix architecture which consists of a 4-MSB and a 4-LSB current matrix stage. The symmetric two-stage current cell matrix architecture allows the designed DAC to reduce not only a complexity of decoding logics, but also a number of high swing cascode current mirrors. The designed DAC with an active chip area of 0.8 mm^2 is fabricated by a 0.8-μm CMOS n-well standard digital process. The experimental data shows that the rise/fall time, the settling time, and integral nonlinearity/differential nonlinearity (INL/DNL) are 6 ns, 16 ns, and a less than 0.8 LSB, respectively. The designed DAC is fully operational for the power supply down to 2.0 V, such that the DAC is suitable for a low voltage and a low power system application. The power dissipation of the DAC with a single power supply of 3.3 V is measured to be 34.5 mW.**

***Index Terms*—Current cell matrix, DAC, high-swing cascode current mirror, low power.**

I. Introduction

As more progress of very large scale integration (VLSI) technology has been made, many mixed-signal integrated circuits have been implemented on a single chip, including data conversion circuits such as analog-to-digital converters (ADC's) and digital-to-analog converters (DAC's). Recently, many high-speed DAC's have been reported utilizing a single stage current cell matrix [1]–[7]. The DAC architecture with a single stage matrix cell is able to achieve a fast settling time. However, the drawback of this architecture is that it requires a complex decoding logic circuit, which results in a large power dissipation and a large chip area. The other high-speed DAC proposed in the literature [8] is the one that utilizes a resistor string with a high-speed output voltage buffer to drive a low resistance load. Although this DAC is able to achieve a good linearity due to the well-matched resistors, a high-speed output voltage buffer and a resistor string requires a large power dissipation and a large chip area, respectively.

This paper proposes a CMOS high-speed 8-bit DAC with the symmetric two-stage current cell matrix architecture which consists of a 4-MSB (most significant bit) current cell matrix and a 4-LSB (least significant bit) current cell matrix. This symmetric two stage current cell matrix architecture allows the DAC to reduce not only a complexity of decoding logic circuits, but a number of current sources compared with a single-stage current cell matrix DAC.

II. The DAC Architecture with a Symmetric Two-Stage Current Cell Matrix

The proposed DAC circuit diagram is shown in Fig. 1. The input

Manuscript received March 5, 1997; revised April 10, 1998. This work was supported by the Academic Research Fund of the Ministry of Education, Republic of Korea, through Inter-University Semiconductor Research Center (ISRC 97-E-2028) in Seoul National University. The paper was recommended by Associate Editor V. Porra.

J. H. Kim is with the Memory BU Design Team #2, LG Semicon Company, Korea.

K. S. Yoon is with the Department of Electronic Engineering, Inha University, Inchon 402-751, Korea.

Publisher Item Identifier S 1057-7130(98)08499-7.