A Multilevel Neural Network for A/D Conversion

Jen-Dong Yuh, *Student Member, IEEE,* and Robert W. Newcomb, *Fellow, IEEE*

***Abstract*—In this paper we introduce a multilevel neuron and show its use in a neural network multilevel A/D converter. An energy function suited for multilevel neural networks is defined for which local minima problems for A/D conversion are removed by modifying Lee and Sheu's method. This energy function extends others in the sense that it allows one to consider more than two discrete levels in the neuron output and threshold settings. We also demonstrate how to build and implement multilevel nonlinearities, and a way of implementing a multilevel neural network for A/D conversion by taking advantage of BiCMOS technologies. Computer simulations are included to illustrate how this design functions and individual component VLSI chips measurements for multilevel A/D conversion are presented to show how each component operates.**

I. Introduction

NEURAL networks have been shown to have the capability to handle problems in a number of applications and in a variety of areas [1]–[5]. They are able to deal with some optimization problems more efficiently than digital computers. The function of each neuron is simple and primitive. However, collectively they can be organized to solve complicated problems [1].

Typically, a neural system will need a large number of weights which makes VLSI implementation inconvenient. Thus a means to reduce these weights and still achieve desired goals should be of major interest for neural network design. Multilevel neurons have the potential of reducing the number of weights and when properly designed they may be implemented in VLSI. Therefore, we investigate here a multilevel neural network.

While most neural networks in the literature can be considered as "two state" neural networks [3], [4]. There are a couple of works using multilevel neurons [6], [7]. In [6], Banzhaf did some simulations involving multistate neural associative memory and introduced what he called an energy function. However, he did not show what kind of nonlinearity should be used, what is the possibility to implement that nonlinearity, and necessary properties and his energy function. In [7], Si and Michel did analysis and synthesis of discrete-time neural networks with multilevel threshold functions using a pseudoinverse and without using energy functions. In this paper, we consider these properties by examining multilevel neural networks starting from Tank and Hopfield's optimization neural networks [2], [3]. They designed an A/D converter based on analog processors and showed that these analog processors should be promising in optimization applications [2]. Here we extend the results of [2] to multilevel neural processors. An energy function is given for multilevel neural networks. This energy function is also adequate for binary neural networks defined by Tank and Hopfield [2], since their network is a special case of that presented here. A BiCMOS nonlinearity building block (NBB) circuit suited for multilevel neural network is designed to apply it to multilevel A/D converters. SPICE3 simulations of multilevel neural A/D converters with MOSIS BiCMOS analog process parameters are included to demonstrate the feasibility of this implementation. The reason that we use BiCMOS technology is to approximate mathematical equations for multilevel nonlinearities as closely as possible. As a result, by means of computer simulations, and formal analysis, we show that neural A/D converters can be obtained by using multilevel neurons. The chip is fabricated via MOSIS using BiCMOS technology and individual component measurements for multilevel neural A/D conversion are shown in this paper.

This paper proceeds as follows. In Section II, multilevel nonlinearities are introduced, and with these nonlinearities, we construct multilevel neurons. Then, a fully connected multilevel network is organized. Also, we demonstrate a way of implementing these multilevel neurons, and implementation of neuron weights is also shown. In Section III, an energy function is proposed for these neural networks. We use this energy function for designing multilevel neural A/D converters with local minima elimination in Section IV. some useful notations are introduced to help eliminating local minima for multilevel neural A/D converters. A circuit implementation for multilevel neural A/D converters is also demonstrated through SPICE3 simulations and measurements on a chip. Finally, we compare our circuit implementation with Lee and Sheu's, discuss some advantages and disadvantages of our implementation, and make our conclusions in Section V.

II. Multilevel Nonlinearities

A typical artificial neuron presently most popular in the literature [3], [4], [8] can be modeled by a set of weights which multiply inputs to the neuron. The neuron sums the weighted inputs and passes the result through a nonlinearity. Hard limiters, saturating linear elements, and sigmoidal functions are three common types of nonlinearities [3], [4]. The nonlinearities can be considered as two-level nonlinearities since classification is into one of two neuron output values. In order to represent a b-level nonlinearity, a multilevel function $M(\cdot)$ is introduced as

$$M(x) = \sum_{j=0}^{b-1} a_j f_j(x - \theta_j) \qquad (1)$$

Manuscript received July 15, 1992; revised August 31, 1992. This work was supported by the National Science Foundation under Grant MIP 891122.

The authors are with the Department of Electrical Engineering, University of Maryland, College Park, MD 20742.

IEEE Log Number 9206627.

where the basis functions $f_j(\cdot)$ are chosen among any of a number of monotonically nondecreasing step-type functions, mapping from the reals R onto a finite interval I. Examples of such $f_j(\cdot)$'s are the unit step function $1(x)$, the special sigmoid function $1/(1+e^{-\lambda x})$ and $\tanh(\lambda x)$ for which the intervals I are $[0,1]$, $[0,1]$, and $[-1,1]$ respectively. In (1), a_j are positive finite constants for $0 \le j \le b-1$. θ_j is a threshold point where the neuron output is considered to transit from one level to the next level, and $\theta_j < \theta_{j+1}$; we take $\theta_0 = -\infty$ and $\theta_b = +\infty$ to simplify the following formulas. There are generally b output levels of this multilevel neuron nonlinearity with b, a positive integer greater than 1.

By choosing the $f_j(\cdot)$ to all be a unit step function, we have the multilevel hard limiter $M_h(\cdot)$ function shown in Fig. 1(a) and described by

$$\begin{aligned} M_h(x) &= \sum_{j=0}^{b-1} a_j 1(x-\theta_j) \\ &= \sum_{j=0}^{i} a_j; \qquad \theta_i \le x < \theta_{i+1}, \qquad 0 \le i < b-1. \end{aligned} \tag{2a}$$

The subscript h denotes the hard limiter function. The nonlinearity is called "discrete" because its output generally assumes the discrete values, $m_0, m_1, \ldots, m_{b-1}$, where $m_i = a_0 + \ldots + a_i$. Although discrete hard limiters are appropriate for computer simulations, they are discontinuous and, consequently, not the most appropriate to implement in hardware. Therefore, another type of continuous multilevel nonlinearity is needed. The special sigmoid function has been widely used in continuous neural networks [4], [9], [10]. Hence, it is an appropriate choice to use as the basis functions in which case (1) gives

$$\begin{aligned} M_S(x) = &\frac{a_0}{\left(1+e^{-\lambda(x-\theta_0)}\right)} + \frac{a_1}{\left(1+e^{-\lambda(x-\theta_1)}\right)} + \cdots \\ &+ \frac{a_{b-1}}{\left(1+e^{-\lambda(x-\theta_{b-1})}\right)}. \end{aligned} \tag{2b}$$

For the same reason, subscript s denotes $f(\cdot)$ to be the special sigmoid function. This is the nonlinearity that the hardware implementation will closely approximate. Similarly, a multilevel $\tanh(\cdot)$ nonlinearity can also be built by letting the $f_j(\cdot)$ to be $\tanh(\cdot)$ functions.

A. Multilevel Neurons

Neuron models can be roughly classified as continuous time models [9]–[12] and discrete time models [3], [8]. In [9], a typical continuous time neuron is defined by (3)

$$c_1 \frac{dx_i}{dt} = \sum_{j=1}^{n} w_{ij} v_j - G_i x_i + I_i \tag{3a}$$

$$v_i = g_i(x_i) \tag{3b}$$

where neuron output v_i, in a finite interval range, is a continuous and monotone-increasing function $g_i(\cdot)$ of the input state variable x_i to neuron i, w_{ij} is the connection weight between the jth neuron output and the ith neuron input, and G_i and c_i are the input conductance and the input capacitance of neuron i, respectively. For our multilevel neuron, we modify the above description to a mathematical model defined by (3a) and (4), illustrated in Fig. 2; that is the characteristic function $g(\cdot)$ is replaced by a multilevel nonlinearity

$$v_i = M_i(x_i). \tag{4}$$

Here, $M_i(\cdot)$ is the ith multilevel nonlinearity as per (1) and $U_i(x_i) = G_i x_i$. The function $U_i(x_i)$ is considered to introduce a negative feedback in terms of the neuron state x_i as shown in Fig. 2. Equation (3a) means that the activity of node i, i.e., the state variable x_i, increases if interneuron input to the neuron, which is the sum and I_i in (3a), exceeds the intrinsic feedback function $U_i(x_i)$. As we can see in Fig. 2, if $U_i(x_i)$ is omitted, the output of the integrator, x_i, may be unbounded as long as dx_i/dt keeps either a positive or a negative value for a sufficient period of time.

1) Multilevel Neural Networks: A general multilevel network can be organized as a fully connected network as shown in Fig. 3, following the structure of Hopfield [8], [9]. Each $M(\cdot)$ is assumed to be the sigmoid multilevel type shown in Fig. 1(b). Then the neural network is defined in the following

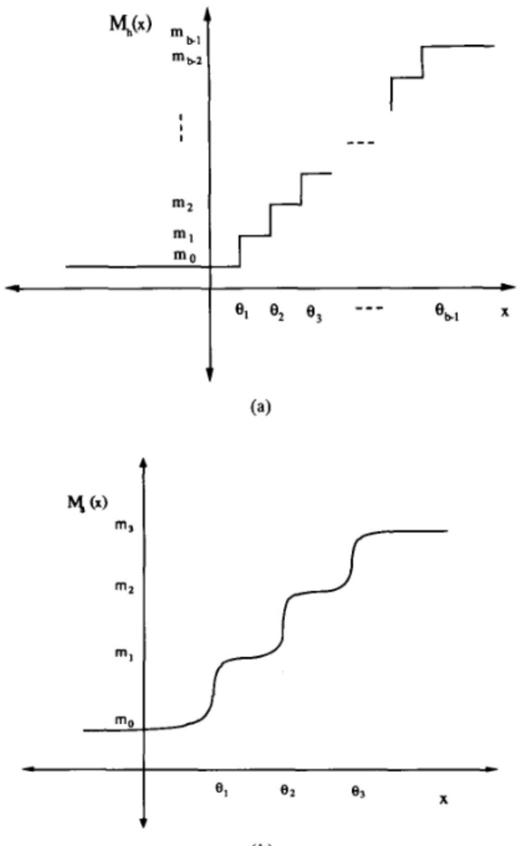


Fig. 1. A hard-limiter multilevel nonlinearity. (b) A sigmoidal multilevel nonlinearity.

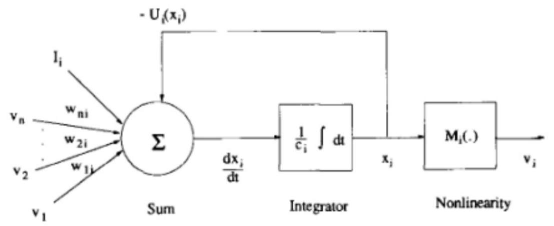


Fig. 2. A function block diagram for a multilevel neuron.

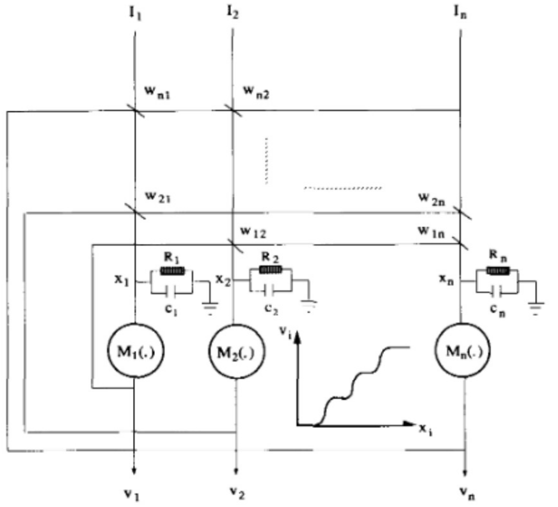


Fig. 3. A multilevel neural network; $M_i(\cdot)$ is the ith multilevel nonlinearity.

matrix form of (3):

$$C\dot{X} = WV - GX + I \tag{5a}$$

$$V = M(X) \tag{5b}$$

where, using superscript T to denote matrix transpose, $X = [x_1, x_2, \cdots, x_n]^T$ consists of the voltages across input capacitors, C is a $n \times n$ diagonal matrix with the ith diagonal capacitance value c_i, and W is the $n \times n$ matrix of connection weights. $V = [v_1, v_2, \cdots, v_n]^T$ is the neuron output vector. G is also a diagonal matrix with the ith diagonal value G_i. $I = [I_1, I_2, \cdots, I_n]^T$ is the vector of external inputs. Here, $\dot{X}$ denotes the derivative of X.

B. Multilevel Neuron Circuit Implementations

We wish to implement the typical neuron model in multilevel form. For this, a block diagram for a circuit implementation is shown in Fig. 4 by using a multilevel nonlinearity of (1). It consists of three parts, current sources formed via weights on neuron outputs and as external inputs, an equivalent input resistor and capacitor pair, and a multilevel nonlinearity. The weight w_{ij} may be implemented by the geometrical ratio W/L of two MOS current mirrors [13]. In Fig. 4, we see that $U_i(x_i) = G_i x_i$, and Kirchhoff's current equation gives (3a) and (4). Furthermore, in the two level neuron case, the threshold point is rarely mentioned and is normally taken to be zero. But in the multilevel case, the choice of threshold points is critical. These threshold points are shown in Fig. 4 explicitly, and we will show how to choose these threshold points in Section IV where a multilevel neural A/D converter design is given.

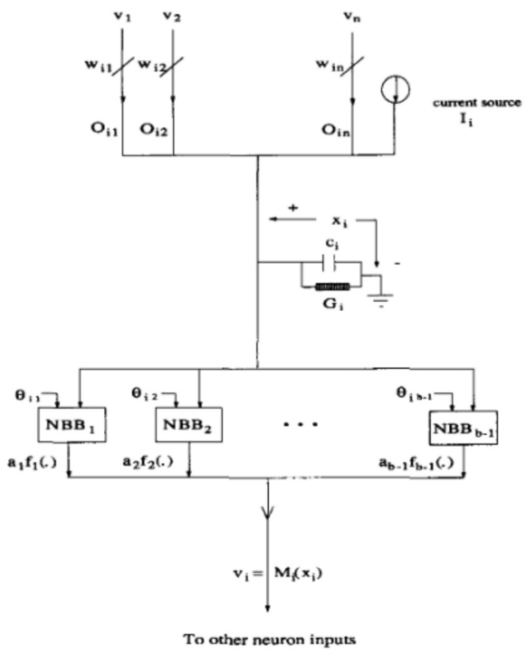


Fig. 4. A block diagram for a sigmoidal multilevel neuron.

1) Multilevel Nonlinearity Implementations: In order to show how to build multilevel nonlinearities, implementation of the nonlinearity building block (NBB) is demonstrated first. A NBB can be implemented by a bipolar emitter coupled pair and MOS transistor current mirrors as shown in Fig. 5. In analyzing Fig. 5, we assume that Q1 and Q2 are matched and I_{c1} is described by (6) [14, pp. 445],

$$I_{c1} = \frac{\alpha_F I_{EE}}{1 + \exp\left(-\frac{v_d}{v_T}\right)} \tag{6}$$

where $v_d = x - \theta$, α_F is the large signal forward current gain of bipolar transistor common base configuration, I_{EE} is the current source for an emitter coupled pair, and V_T is the thermal energy. The NBB shown in Fig. 5 builds a function $a_j f(x - \theta)$. Let the geometry ratio of PMOS transistors M_1 and F_j be $R_{Fj} = (W/L_{Fj})/(W/L_{T1})$. Assuming that the channel modulation effect is relatively insignificant. Then I_{dj} is approximately equal to $R_{Fj} I_{ds1}$ by applying current mirror techniques [14, pp. 346] in which case we have the following:

$$a_j f(v_d) = R_{Fj} I_{c1} = \frac{R_{Fj} \alpha_F I_{EE}}{1 + \exp\left(-\frac{v_d}{v_T}\right)}. \tag{7}$$

For building multilevel nonlinearities, these $b - 1$ NBB's collectively generate b output levels in accordance with the input voltage x_i in Fig. 4. In order to implement the summation

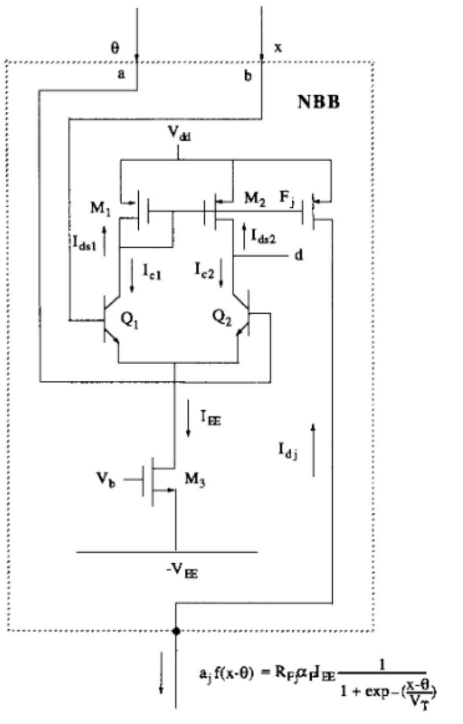


Fig. 5. $1/[1+\exp(-\lambda(x-\theta))]$ Current mode circuit implementations of NBB's with $\lambda = 1/V_T$ using both bipolar and CMOS transistors.

of (1) for a multilevel nonlinearity, we use to advantage that the output of a NBB is a current. As a consequence, by tying together outputs of $b-1$ NBB's with different threshold points, as shown in Fig. 4, we have a general multilevel nonlinearity given by (1). As an example, the following homogeneous multilevel nonlinearities, $M_S(x)$, where only one kind of nonlinearity is used to build a multilevel nonlinearity, are provided for illustration:

$$\begin{aligned} M_S(x) &= a_1 f(x-\theta_1) + a_2 f(x-\theta_2) + \ldots \\ &\quad + a_{b-1} f(x-\theta_{b-1}) \\ &= \alpha_F I_{EE}\left(\frac{R_{F1}}{1+\exp\left(-\frac{v_d^{(1)}}{v_T}\right)} + \frac{R_{F2}}{1+\exp\left(-\frac{v_d^{(2)}}{v_T}\right)} \right. \\ &\qquad \left. + \ldots + \frac{R_{Fb-1}}{1+\exp\left(-\frac{v_d^{(b-1)}}{v_T}\right)}\right) \end{aligned} \tag{8}$$

where $v_d^{(k)} = x - \theta_k$, and $M_S(\cdot)$ is a multilevel nonlinearity with threshold settings, $\theta_1, \theta_2, \ldots \theta_{b-1}$. The above multilevel nonlinearity uses the special sigmoid function. Other multilevel nonlinearities can be built in a similar way by means of constructing NBB's. As shown in [15], tanh$[x]$ is used as NBB's to build multilevel nonlinearities. Output levels of these multilevel nonlinearities can be scaled by choices $R_{F1}, \ldots, R_{Fb-1}$, and I_{EE}.

2) Implementation of Neuron Weights: Neuron weights are implemented via the L/W ratios of MOS current mirrors. In Fig. 6, a bidirectional current mirror [16] is shown that allows current to be mirrored no matter whether it flows into or out of the input node; this is desirable if a multilevel nonlinearity takes both signs. A positive weight is defined for Fig. 6 when the output current O is the same direction as the input current $M_S(x)$. Thus an inverting bidirectional current mirror is shown in Fig. 6, part (a), since O is negative when $M_S(\cdot)$ is positive. For simplicity for that figure, assume that channel modulation effects are relatively insignificant, and therefore,

$$\begin{aligned} O &= -(I_{r-} + I_{d-}) \\ &= -\left(K^{(1)} I_{ds5} + K^{(2)} I_{ds4}\right) = w^{(-)} M_S(\cdot) \end{aligned} \tag{9a}$$

where $K^{(1)} = (W/L_{r-})/(W/L_5)$, $K^{(2)} = (W/L_{d-})/(W/L_4)$, $-K^{(1)} = -K^{(2)} = w^{(-)}$, and $I_{ds4} + I_{ds5} = M_S(\cdot)$ are shown in Fig. 6 part (a); (9a) is to give a relationship between O and $M_S(x)$, $O = w^{(-)} M_S(x)$, where $w^{(-)}$ is negative, designating a negative weight. A noninverting bidirectional current mirror is obtained in Fig. 6, part (b) by cascading two inverting bidirectional current mirrors. Similarly, let T_7 and T_5, and T_6 and T_4 be equal sizes, we get $O = w^{(+)} M_S(x)$, where $w^{(+)} = (W/L_{r+})/(W/L_9) = (W/L_{d+})/(W/L_8)$. Therefore, weights of either sign for a multilevel nonlinearity $M_S(x)$ are given by the following equations:

$$w^{(+)} = (W/L_{r+})/(W/L_9) = (W/L_{d+})/(W/L_8) \tag{9b}$$

$$w^{(-)} = (W/L_{r-})(W/L_5) = (W/L_{d-})/(W/L_4). \tag{9c}$$

The above implementation is a general way to obtain weights of either sign for currents of either sign for multilevel nonlinearities. However, for some special cases, we can implement weights in a way that uses less transistors. If $M_S(x) \geq 0$ for all x, positive weights are implemented by current sources and negative weights are implemented by current sinks. Similarly, if $M_S(x) \leq 0$ for all x, positive weights are implemented by current sinks and negative weights are implemented by current sources.

III. Energy Functions for Multilevel Neural Networks

For the system defined by (5), we define the following energy function,

$$E = -\frac{1}{2} V^T W V + \int_0^x GZ(V)\, dV - V^T I \tag{10a}$$

where $Z = [z_1, z_2, \ldots, z_n]^T$ is just a dummy vector of integration for the vector variable X, and E is a scaler. Expanding 10(a), we have, where n is the number of neurons:

$$\begin{aligned} E(x) = \frac{1}{2} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} v_i(x_i) v_j(x_j) + \sum_{i=1}^{n} G_i \int_0^{x_b^a} z_i \, dv_i\,(z_i) \\ - \sum_{i=1}^{n} I_i v_i(x_i) \end{aligned} \tag{10b}$$

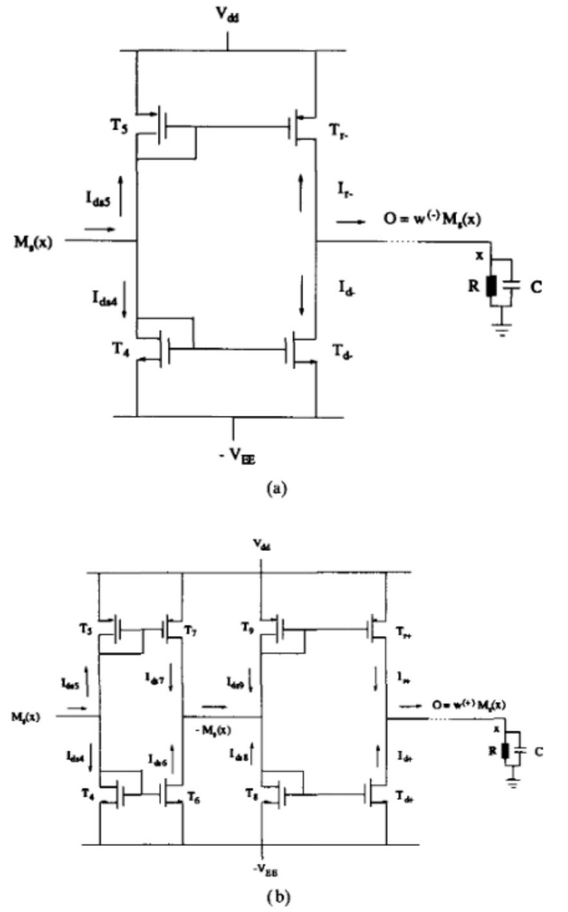


Fig. 6. A neuron weight implementation for $M_S(x)$ using bidirectional current mirrors.

Equation (10b) is a energy function when W is symmetric since

$$\frac{dE(x)}{dt} = \sum_{i=1}^{n} \frac{dE(x)}{dv_i(x_i)} \cdot \frac{dv_i(x_i)}{dx_i} \cdot \frac{dx_i}{dt}$$
$$= -\sum_{i=1}^{n} \frac{1}{c_i} \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} v_j - G_j x_j + I_i \right)^2 \cdot \frac{dv_i}{dx_i}. \tag{10c}$$

By definition, an energy function should be bounded with a trajectory derivative that is less than zero. In our case, $dE(x)/dt \leq 0$ as long as dv_i/dx_i is positive since c_i is assumed positive. As indicated in Section II-A, $v_i(x)$ is bounded for all x, and I_i, G_i are all finite values. Therefore, for showing the boundedness of the energy function, what we need to investigate is the integral term in (10b):

$$\sum_{i=1}^{n} G_i \int_0^{x_i} z\, dv(z) = \sum_{i=1}^{n} G_i E_\theta^i(v). \tag{11a}$$

The integral is denoted by $E_\theta(v)$ and developed as

$$E_\theta(v) = \int_0^{x} z\, dv(z) = \int_{m_0}^{v} M^{-1}(y)\, dy. \tag{11b}$$

Here, normally $M_i^{-1}(\cdot)$ can not be expressed in a closed form. Therefore, we use the Stieltjes integral form of the first term and get the following, assuming $M_S(\cdot)$ of (2b):

$$E_\theta(x) = \sum_{j=1}^{b-1} \frac{1}{\lambda} \log_e\left(1 + e^{\lambda\theta_j}\right)$$
$$- \sum_{j=1}^{b-1} \left[\frac{x e^{-\lambda(x-\theta_j)}}{1 + e^{-\lambda(x-\theta_j)}} + \frac{1}{\lambda} \log_e\left(1 + e^{-\lambda(x-\theta_j)}\right) \right]. \tag{11c}$$

The first term of (11c) is a constant and the last two terms go to bounded values as x approaches infinity. That is

$$F(x) = \sum_{j=1}^{b-1} \left[\frac{x e^{-\lambda(x-\theta_j)}}{1 + e^{-\lambda(x-\theta_j)}} + \frac{1}{\lambda} \log_e\left(1 + e^{-\lambda(x-\theta_j)}\right) \right]$$
$$\text{has } \lim_{x \to \infty} F(x) = 0, \text{ and}$$
$$\lim_{x \to -\infty} F(x) = \sum_{j=1}^{b-1} \theta_j. \tag{11d}$$

Therefore, $E_\theta(x)$ is bounded even if x approaches infinity. Here, $M_i(x_i) = v_i$ and we assume that $M_i(x_i)$ is a monotonically continuous function as defined in (2b). Therefore, (10b) is indeed an energy function since $E(x)$ is bounded and v_i is monotonically increasing in x_i.

Equation (10) provides the foundation for multilevel Hopfield style neural systems. We will demonstrate how they can be applied to a multilevel neural A/D converter in Section IV.

IV. Multilevel Neral A/D Converters

The strategy for designing a multilevel A/D converter is similar to that used on Tank and Hopfield's A/D converter [2]. However, in our development, the energy function (given in (17)) and the $M_S(\cdot)$ multilevel nonlinearity are used for designing multilevel neural A/D converters. Multilevel neural A/D conversion is considered as a simple optimization problem. In order to do this, we use the *square difference* between an analog input x and its neuron output representation, $[V]$, in the following:

$$E_{\text{sd}} = \frac{1}{2}(x - [V])^2, \quad \text{where } [V] = \sum_{k=1}^{n} b^{k-1} v_k \tag{12}$$

where V is the neuron output vector. What we want is to minimizing E_{sd} so that we can obtain the best digital representation for a given input x. In order to have a neural network perform this job, we need a neural network energy function which is in a form similar to (12) a well as (10b). Also two criteria must be met. The first is to have v_i as close as possible to one of the values in $L = \{0, 1, 2, \cdots b-1\}$, and the second is to minimize this error energy function E_{sd} by use of multilevel neural network so that each analog input

x will map to a base b representation $[V]$ of neural outputs with minimal E_{sd}. Since (10b) is a multilevel energy function, what we want is to arrange E_{sd} in a form similar to (10b). Expanding (12) and dropping the constant term x^2, we have the following:

$$E'_{\mathrm{sd}} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(-b^{i+j-2}\right)v_i v_j - \sum_{i=1}^{n}\left(xb^{i-1}\right)v_i + \sum_{i=1}^{n} b^{2(i-1)}\frac{1}{2}\left(v_i\right)^2. \tag{13}$$

As we compare (10b) with (13), the integral term in (10b) needs more treatment to obtain a similar term as the last term in (13). To accomplish this, we examine equation E_θ of (11c). Assume that λ is a large number, then $E_\theta(v)$ is illustrated for $b = 4$ in Fig. 7(b). The $E_\theta(v)$ portion of $E(x)$ is the area under the function $M_i^{-1}(v)$. We will assume homogeneous threshold settings, i.e., $\theta_{j+1} = \theta_j + \Delta\theta$ for all j, and $\Delta\theta = \theta_1$. In a similar way, we also assume homogeneous neuron output levels, i.e., $m_{k+1} = m_k + \Delta m$ for all k. To obtain a term similar to the last term of (13), we define

$$E_\theta^d(v) = \Delta\theta\Delta m\,\frac{(v)^2 + v}{2} \quad \text{and} \quad E_{\theta\_\mathrm{err}}(v) = E_\theta(v) - E_\theta^d(v) \tag{14}$$

and then we express $E_\theta(v)$ as

$$E_\theta(v) = E_\theta^d(v) + E_{\theta\_\mathrm{err}}(v) \tag{15}$$

which is illustrated in Fig. 7(b). Substitute $E_\theta(v)$ of (15) into (10b) to replace the integral term in (10b), and we have

$$E(V) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_{ij}v_i v_j - \sum_{i=1}^{n} I_i v_i + \sum_{i=1}^{n} G_i\left[E_\theta^d(v_i) + E_{\theta\_\mathrm{err}}(v_i)\right] \tag{16}$$

where V is the neural network output vector. Then, substitute $E_\theta^d(v_i)$ of (14) into (16), and rearrange (16) as

$$E(V) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq 1}}^{n} w_{ij}v_i v_j - \sum_{i=1}^{n}\left(I_i - \frac{1}{2}G_i\,\Delta\theta_i\,\Delta m_i\right)v_i + \frac{1}{2}\sum_{i=1}^{n}(v_i)^2 G_i\,\Delta\theta_i\,\Delta m_i + \sum_{i=1}^{n} G_i E_{\theta\_\mathrm{err}}(v_i). \tag{17}$$

here, (17) is in a similar form as (14) except for the last term in (17). To (14), in order to have a term similar to the last term of (17), we add a term E_D, which favors discrete neuron outputs, and have

$$\begin{aligned} E_{AD} &= E'_{\mathrm{sd}} + E_D \\ &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(-b^{i+j-2}\right)v_i v_j - \sum_{i=1}^{n}\left(xb^{i-1}\right)v_i \\ &\quad + \frac{1}{2}\sum_{i=1}^{n}(v_i)^2 b^{2(i-1)} + E_D. \end{aligned} \tag{18}$$

Now, (18) is in a similar form to (17). But, E_D is like a parasitic term when we use a neural network energy function for A/D conversion. To know the effect of E_D in E_{AD}, we examine $E_{\theta\_\mathrm{err}}(v_i)$ since E_D is in a summation form of $E_{\theta\_\mathrm{err}}(v_i)$. As shown in Fig. 7(b), we know the last term of (17), $E_{\theta\_\mathrm{err}}(v_i)$, is bigger than or equal to 0 and approaches zero for the ith neuron output close to one of the values in L. As a result, the effect of $E_{\theta\_\mathrm{err}}(v_i)$ is to favor discrete neuron outputs. Therefore, (18) satisfies our two criteria to design a neural A/D converter. There is no need to know exactly what E_D should be because it does not affect us in deciding neuron connection weights, external inputs, and thresholds for each neuron. By comparing (18) and (17), connection weights, inputs, and thresholds for each neuron can be obtained as

$$w_{ij} = -b^{i+j-2} \tag{19a}$$

$$I_i = x\cdot b^{i-1} + 1/2G_i\,\Delta\theta_i\,\Delta m_i \tag{19b}$$

$$G_i\,\Delta\theta_i\,\Delta m_i = b^{2(i-1)}. \tag{19c}$$

Connection weight entries are completely determined by (19a). Here, x is the analog input voltage. Without losing generality by choosing $G_i\,\Delta m_i = 1$, I_i is determined by (19b) and we obtain $\Delta\theta_i = b^{2(i-1)}$ from (19c). Threshold for each neuron are determined by $\Delta\theta_{i,j+1} = \theta_{i,j} + \Delta\theta_i$; note that $\Delta\theta_i = \theta_{i,1}$. A neural A/D converter circuit design will be shown in Section IV-A.

A. Local Minimal Energy Problems for Neural A/D Converters

The multilevel neural A/D converter shown in Section IV suffers from local minimal energy problems. In order to phrase the means to eliminate these local minima, in Section IV-B, we introduce some useful notation. Since energy functions are expressed in terms of neuron outputs V, our analysis of local minimal problems for A/D conversion is based on neuron outputs too. We assume that there are only discrete neuron output levels for simplicity.

A normal A/D converter is a function that takes an analog input value and gives a finite number of digital outputs. It is defined as a function, $F_{a2d}(x)$, which maps the analog interval $R_x = (r^-, r^+)$ into a n-vector whose entries are digits in $L = \{0, 1, \cdots, b-1\}$, where n is the number of neurons. This function is only a function of analog input x. In contrast, we define an inverse neural A/D converter function, that is

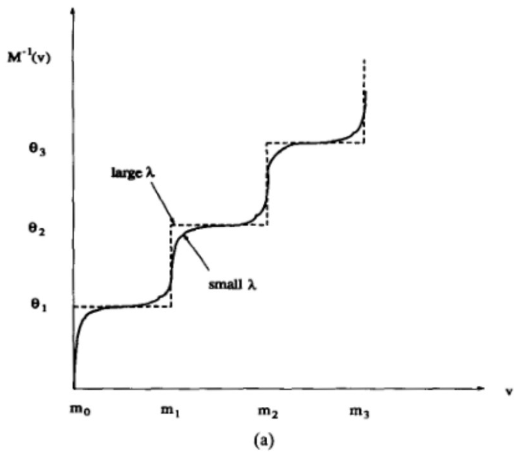
$$f_{a2d}^{-1}(C_i) = \{x \mid \forall\, C_j \in L^n, f_{a2d}(x, C_j) = C_i, C_i \in L^n\} \tag{20a}$$

where $f_{a2d}^{-1}(\cdot)$ is a function mapping from a digital output code C_i to the analog input set giving the code. By definition, a neuron output code is a vector $C_i = [v_{ci,1}, v_{ci,2}, \cdots, v_{ci,n}]^T$, where $v_{ci,k}$ is the kth neuron output, $i \in \{0, 1, \cdots, b^n - 1\}$. The analog input range of code C_i is

$$X_{ci} = f_{a2d}^{-1}(C_i). \tag{20b}$$

The limits on the set X_{ci} are

$$X_{c_i}^{1} = \min_x\left\{f_{a2d}^{-1}(C_i)\right\}, \qquad X_{c_i}^{h} = \max_x\left\{f_{a2d}^{-1}(C_i)\right\}. \tag{20c}$$



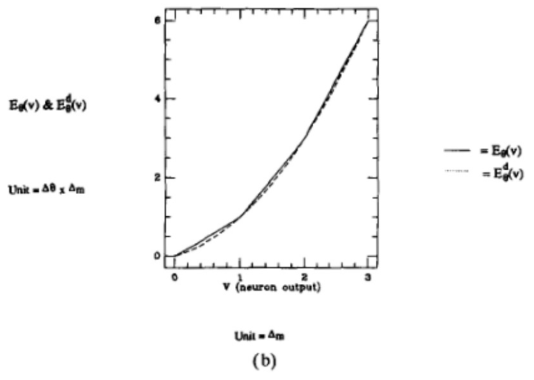


Fig. 7. (a) Inverse function $M^{-1}(v)$ in terms of neuron output, where $b = 4$. (b) Energy functions of $E_\theta(v)$ and $E_\theta^d(v)$ in terms of a neuron output; $E_{\theta\_err}(v)$ is the difference between these two curves.

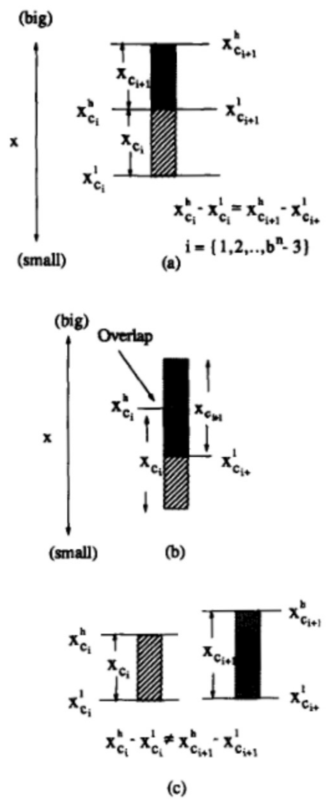


Fig. 8. (a) without local minimum problem; (b) an overlap; (c) different analog input ranges for two adjacent codes.

Also, C_{i+1} denotes the one step higher code from C_i, as shown in Fig. 8, and $C_0 = [0, 0, \cdots, 0]$, $C_1 = [1, 0, \cdots, 0], \cdots, C^n b - 1 = [b-1, \cdots, b-1]$. Here, index i represents a discrete D/A output level, and $i = [C_i]$, where $[\cdot]$ is as defined as (12). In order for the neural converter to function correctly as an A/D converter, the following conditions should be satisfied.

$$\text{(P1)} \quad f_{a2d}^{-1}(C_i) \cap f_{a2d}^{-1}(C_{i+1}) = \phi \tag{21a}$$

$$\text{(P2)} \quad X^h_{c_i} - X^1_{c_i} = X^h_{c_{i+1}} - X^1_{c_{i+1}};$$

$$i = \{1, 2, \cdots, b^2 - 3\} \tag{21b}$$

where ϕ is the empty set. Condition (P1) means that there is no analog input overlap between two adjacent codes. Condition (P_2) gives an appropriate analog input range for each neuron output code except boundary codes, C_0 & C_{b^n-1}. The local minimum problem for A/D conversion is illustrated by Fig. 8(a), (b), and (c). Fig. 8(a) is what is desired for the A/D converter to function correctly. Fig. 8(b) shows violation of the condition (P1), and Fig. 8(c) shows violation of the condition (P2). These conditions are used to illustrate local minima for neural A/D converters. Neural A/D converters with local minimum problems violate condition (P1) and/or (P2). We will use these two conditions helping us to eliminate local minima of neural A/D converters.

B. Elimination of Local Minima for Multilevel Neural A/D Converters

To rectify problems mentioned in Section IV-A, the concept proposed by Lee and Sheu [17], [18] is exploited and generalized to multilevel neural A/D converters. The overlapped input range between two adjacent digital codes can be corrected by adding compensation currents $I^*(V)$ to the neuron inputs. Basically, the function of this compensating circuit is to fill up the wells of local minima, so that only a global minimum exists [18].

$$\begin{aligned} E^*(V) &= E(V) - \sum_{i=1}^{n} I_i^*(V) v_i \\ &= -\frac{1}{2} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} v_i v_j - \sum_{i=1}^{n} (I_i + I_i^*(V)) v_i \\ &\quad + \frac{1}{2} \sum_{i=1}^{n} v_i(v_i + 1) G_i \, \Delta\theta_i \, \Delta m_i + \sum_{i=1}^{n} G_i E_{\theta\_err}(v_i) \end{aligned} \tag{22}$$

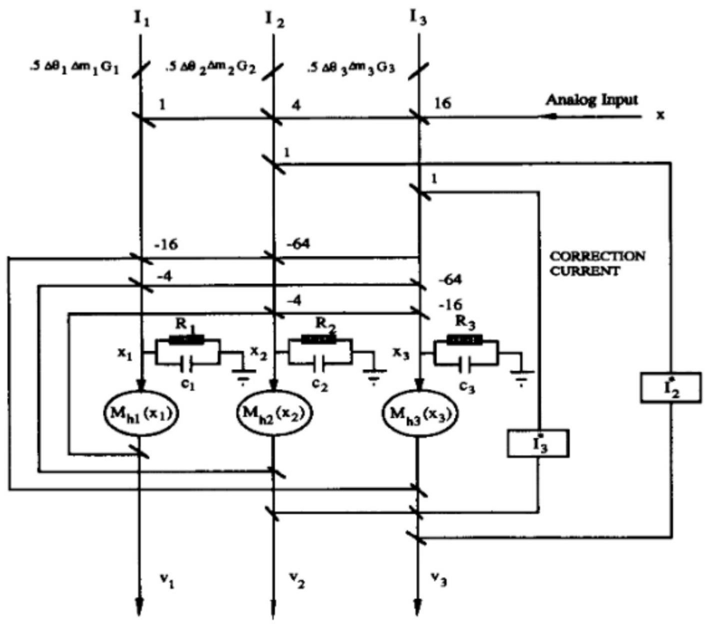


Fig. 9. A multilevel neural A/D converter without local minimal problem; Functions of I_2^* and I_3^* are shown in Table I; $M_{hi}(\cdot)$ is the ith multilevel nonlinearity for neuron i.

where $I^*(V) : L^n \Rightarrow R^n$ is the compensation input current vector, and R is the set of real numbers; n and L are as defined in Section IV-A. Then $E^*(V)$ is the corrected energy function. In order to determine $I^*(V)$ for the neuron output codes C_i of the last section, we need to know overlap input ranges between adjacent neuron output codes. An overlap input range between any two adjacent neuron output codes is defined by

$$D = X^1_{c_{i+1}} - X^h_{c_i}. \tag{23}$$

If $D < 0$, a given analog input value may map to more than one neuron output code. For neural A/D converters, $D \le 0$ indicates violation of condition (P1). Our objective is to have $D = 0$ for each set of adjacent codes so that local minima can be eliminated. To examine this problem appropriately, we denote two consecutive codes as $C_{i+1} = [0, \cdots, 0, p, a, \cdots, a]$ and $C_i = [b-1, \cdots, b-1, p-1, a, \cdots, a]$, and index $i = [C_i]$ as mentioned in Section IV-A. Here, $a \in L$ and $p \in L - \{0\} \cdot p$ & $p-1$ sit in the kth position of the neuron output vector, and for a given i, k is given by

$$k = \max_y\{y | \forall\, y \in \{1, 2, \cdots n\}, (i+1) \bmod b^{y-1} = 0\}. \tag{23a}$$

Here, mod is the mod function that gives the remainder as a dividend. As an example for (23a), $i = 15$ & $b = 4$, then $k = 3$; to satisfy $(15+1) \bmod 4^{y-1} = 0$ in (23a), y could be 1, 2, or 3 and then we obtain $k = 3$ by taking the maximum value over y. Furthermore, the kth neuron decides the high and low limit of an analog input range between two adjacent neuron output codes. This is the key point that we should keep in mind.

As a first step towards correcting local minimum problems, the analog input voltage range causing transition of individual neuron outputs from one level to another will be studied. On the kth neuron, using (19b), the original external current input would be $I_k = xb^{k-1} + 1/2G_i\, \Delta\theta_i\, \Delta m_i$ but to correct local minimum problems, we need to add a current (given in (28)) for which we denote the correction current as $I_k^*(V)$. To determine the correction $I_k^*(V)$ for the kth neuron,

$$c_k \frac{dx_k}{dt} = \left[\sum_{\substack{j=1\\ j\neq k}}^{n} w_{kj} v_j(x_j) - x_k G_k + xb^{k-1} + 1/2G_k\, \Delta\theta_k\, \Delta m_k + I_k^*(V)\right]. \tag{24a}$$

Here, x is the analog input voltage for A/D conversion, and x_k is the kth neuron input potential. Setting its left hand side equal to zero gives the steady state A/D conversion. Rearranging (24a) we have

$$x = \frac{\left[x_k G_k - \sum\limits_{\substack{j=1\\ j\neq k}}^{n} w_{kj} v_j(x_j) - 1/2G_k\, \Delta\theta_k\, \Delta m_k - I_k^*(V)\right]}{b^{k-1}}. \tag{24b}$$

Setting $x_k = \theta_{p-1,k}$ for the lower limit and then to $\theta_{p,k}$ for the higher limit when the kth neuron output is $v_k = p - 1$, the analog input voltage range is given by the following

TABLE I
CORRECTION LOGIC FUNCTIONS, F2 AND F3, FOR LOCAL MINIMAL ELIMINATION IN FIG. 9

$V_3V_2V_1$		F2		F3
h 0 0	(d 0 0)	0	(−1)	−1
d 0 1		0		0
d 0 2		0		0
d 0 3		1		0
d 1 0		−1		0
d 1 1		0		0
d 1 2		0		0
d 1 3		1		0
d 2 0		−1		0
d 2 1		0		0
d 2 2		0		0
d 2 3		1		0
d 3 0		−1		0
d 3 1		0		0
d 3 2		0		0
h′ 3 3	(d 3 3)	0	(1)	1

inequality

$$\frac{G_k\theta_{p-1,k} - \sum_{\substack{j=1\\j\neq k}}^{n} v_j W_{kj} - 1/2G_k\,\Delta\theta_k\,\Delta m_k - I_k^*(V)}{b^{k-1}} < x$$
$$< \frac{G_k\theta_{p,k} - \sum_{\substack{j=1\\j\neq k}}^{n} v_j w_{kj} - 1/2G_k\,\Delta\theta_k\,\Delta m_k - I_k^*(V)}{b^{k-1}}. \tag{25}$$

In order to know the overlap taking place between two consecutive codes, we substitute $C_i = [V_{ci,1}, V_{ci,2}, \cdots V_{ci,n}]$ and $C_{i+1} = [V_{ci+1,1}, V_{ci+1,2}, \cdots V_{ci+1,n}]$ into (25) upper limit, and (25) lower limit, respectively, and then we have the following inequalities:

$$x < \frac{G_i\theta_{p,k} - \sum_{\substack{j=1\\j\neq k}}^{n} v_{ci,j}w_{kj} - 1/2G_k\,\Delta\theta_k\,\Delta m_k - I_k^*(C_i)}{b^{k-1}} = x^h_{C_i} \tag{26a}$$

$$x > \frac{G_k\theta_{p,k} - \sum_{\substack{j=1\\j\neq k}}^{n} v_{ci+1,j}w_{kj} - 1/2G_k\,\Delta\theta_k\,\Delta m_k - I_k^*(C_{i+1})}{b^{k-1}} = x^1_{C_{i+1}}. \tag{26b}$$

As defined in Section IV-A, $v_{ci,j}$ is the jth neuron output for code C_i and $v_{ci+1,j}$ is the jth neuron output for code C_{i+1} and k is defined as (23a). From (26), the difference of (26a) and (26b) gives the following:

$$D = x^h_{C_i} - x^1_{C_{i+1}} = \frac{-(b-1)\sum_{j=1}^{k-1} w_{kj} + I_k^*(C_{i+1}) - I_k^*(C_i)}{b^{k-1}}. \tag{27}$$

Because of the condition (P2), the analog input range for each neuron output code should be corrected to be at the middle point of the overlapped range. Thus we let $I_k^*(C_i) =$

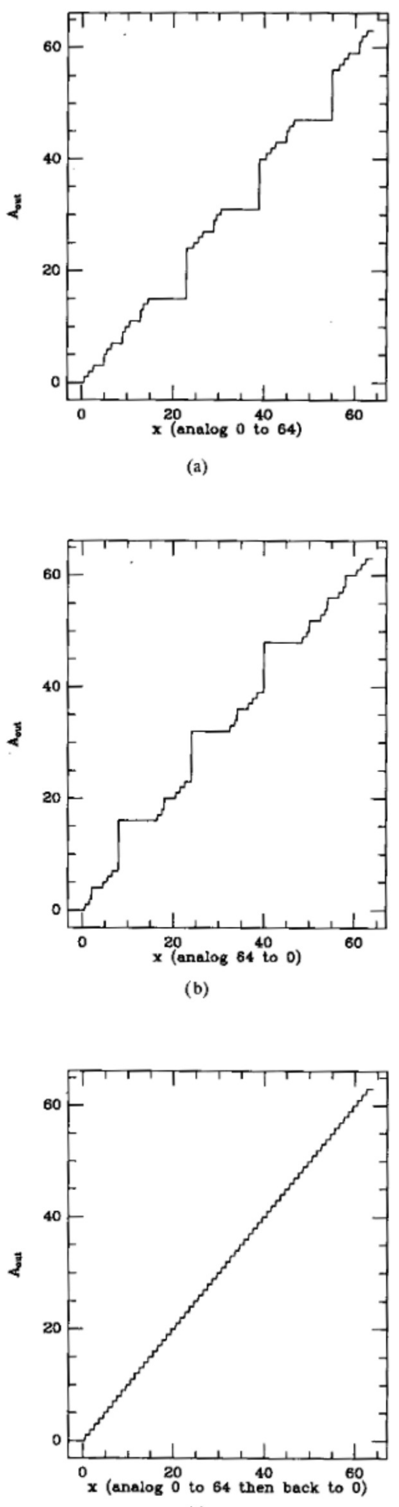
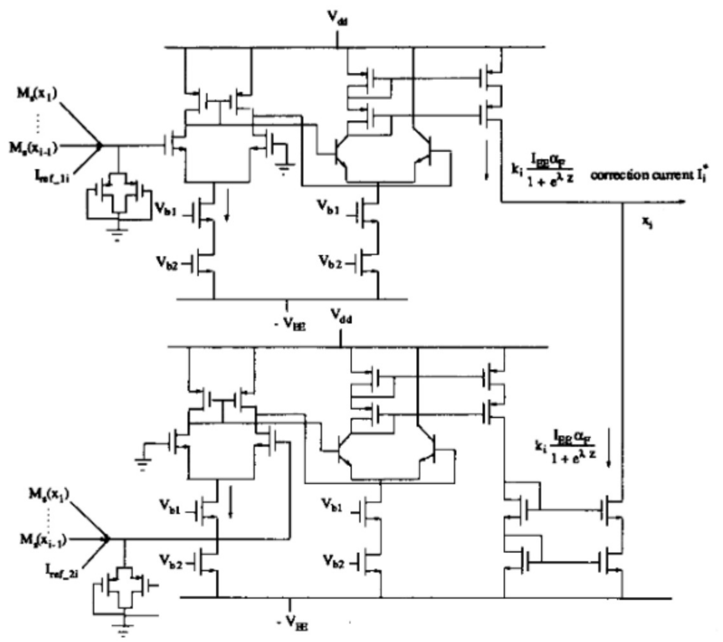
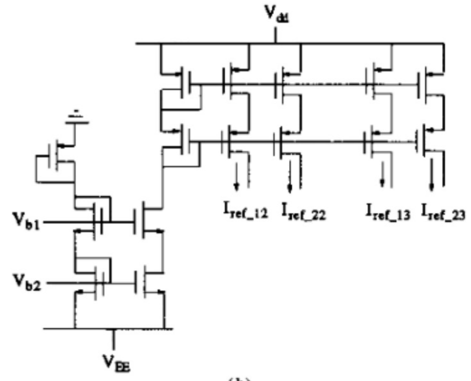


Fig. 10. Three discrete-time simulations of D/A output versus analog input x; part (a) and part (b) without correction; part (c) with correction.



(a)



(b)

Fig. 11. (a) A BiCMOS circuit implementation for I_i^*, where $i = 2, \cdots, n$; (b) A circuit for V_{b1} and V_{b2} and $I_{\rm ref}$'s, where $n = 3$.

$-I_k^*(C_{i+1})$, and $D = 0$ in (27), and have

$$I_k^*(C_i) = -\frac{1}{2}(b-1)\sum_{j=1}^{k-1} w_{kj} \quad \text{and}$$
$$I_k^*(C_{i+1}) = \frac{1}{2}(b-1)\sum_{j=1}^{k-1} w_{kj}. \tag{28}$$

From (28), if two adjacent codes differ in the least significant digit, $k = 1$, no overlap will occur to the input range of each code since $D = 0$, and $I_1^*(C_i) = I_1^*(C_{i+1}) = 0$. For $k > 1$, correction currents at neuron inputs are calculated by (28). As an example, for three multilevel neurons and $b = 4$, the corrected multilevel neural A/D converter circuit diagram is shown in Fig. 9. Also, the correction logic function is listed in Table I. In the table, d denotes don't care, $h \in L - \{0\}$ and $h' \in L - \{3\}$. Other entries not listed in this table are all 0's. In order to facilitate circuit implementation, correction logic simplification has been made by using the values in the parentheses; other entries in the table without parentheses remain unchanged. These changes have been verified by Spice3e1 simulations to check the correct functionality.

C. Simulations for Neural Multilevel A/D Converters

For our discrete-time neural A/D system, we do simulations by using a synchronously deterministic iteration; this is

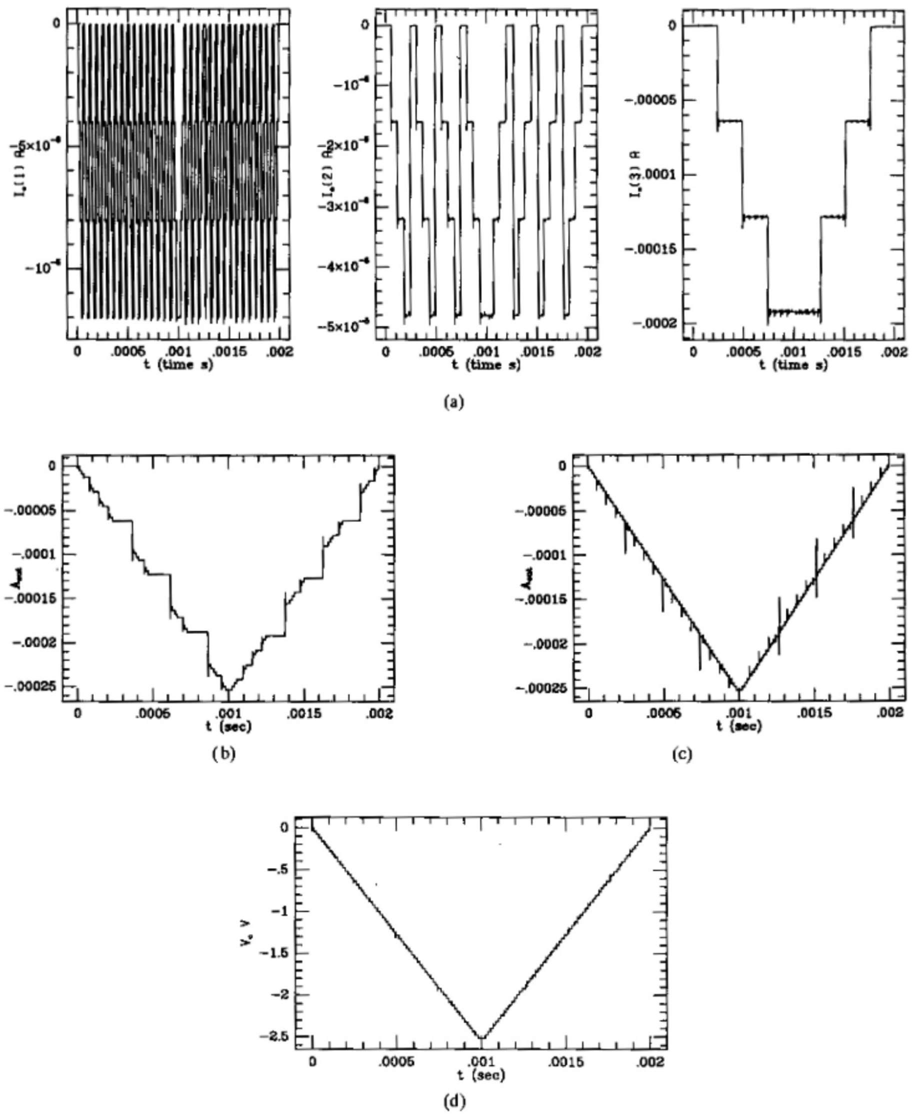


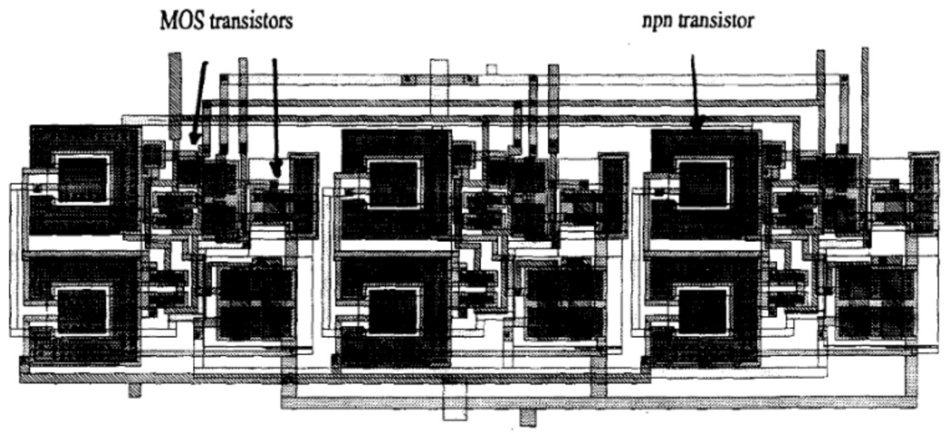
Fig. 12. (a) SPICE3 simulation of a multilevel neural A/D converter with correction circuits. (b) The D/A output without correction circuit is shown. (c) The D/A output of Fig. 12(a) with correction circuit is shown (d) The voltage V_0, shown in the figure is the D/A voltage output after neuron current outputs past low pass filters.

represented by (29)

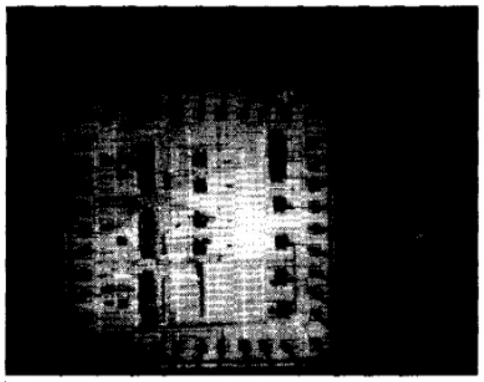
$$v_i^{k+1} = M_h\left(\sum_{j=1}^{n} w_{ij}v_j^k + I_i\right) \qquad (29)$$

where superscript k denotes a discrete time variable.

We have done discrete-time and SPICE3 circuit simulations with and without correction circuits for comparison. Discrete-time simulations are all with 0.1 step voltage analog increment (or decrement) ramp input signal. Basically, discrete-time simulations represent ideal case computer simulations. For discrete-time analog neurons with multilevel nonlinearities, the results we have obtained are shown in Fig. 10. The D/A output with correction, A_{out}, is shown in part (c) of Fig. 10 and the D/A outputs A_{out} without correction is shown in part (a) and (b) of Fig. 10. The analog input voltage for part (a) without correction is from 0 to 64; the analog input for part (b) without correction is from 64 to 0; the analog input for part (c) with correction is from 0 to 64 then back to 0; A_{out}, shown in part (a), (b) and (c), is the D/A output for each run; v_3 is the most significant digit, and v_1 is the least significant digit. As we can see in parts (a) and (b) of Fig. 10, without correction, the D/A output formed by summing three neuron outputs currents directly exhibits a very nonlinear characteristic. Note that the D/A outputs shown in Fig. 10 parts (a) and (b) are different since part (a) is for analog input values between 0 and 64, and



(a)



(b)

Fig. 13. (a) A magic layout of multilevel nonlinearity circuit. (b) Microscope photo of the chip for multilevel neural A/D conversion.

part (b) is for analog input values from 64 to 0, respectively. However, with correction, part (c) in Fig. 10 shows the ideal output with analog input values varying from 0 to 64 then back to 0.

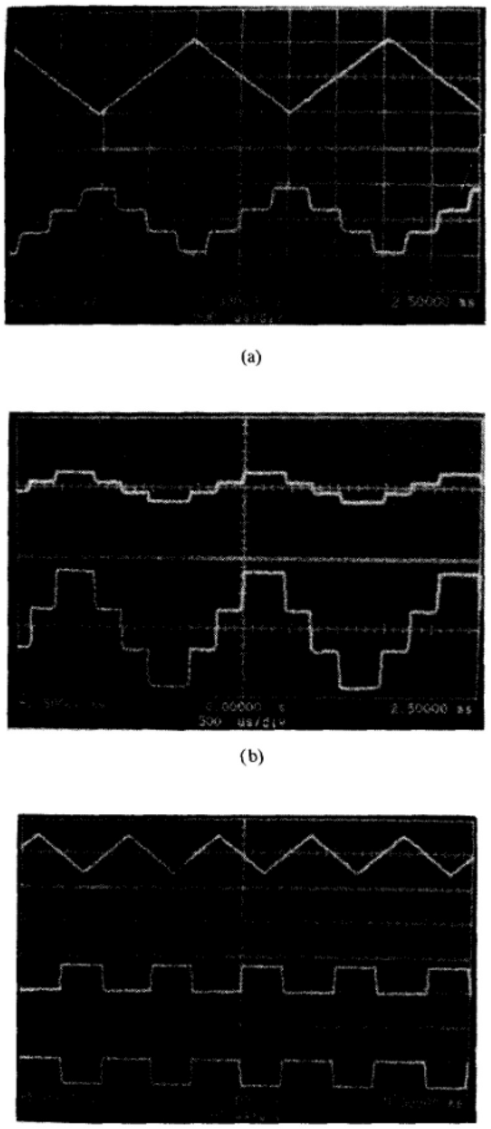
The circuit implementation of Fig. 9 for multilevel neural A/D conversion is addressed in the following; the multilevel nonlinearity (MVN) circuit is implemented by applying the NBB circuit and each NBB is added with a differential pair preamplifier for increasing voltage gain; neural weight implementation is implemented via cascode current mirrors [14, pp. 349]. An implementation of correction circuits for A/D conversion is shown in Fig. 11 part (a), where I_{ref} currents are constant current sources shown in Fig. 11 part (b). The values of I_{ref} currents are set by Table I. Comparing neuron outputs with I_{ref} currents, correction circuits decide whether to furnish correction currents to neuron input nodes or not. When neuron current outputs reach the preset points, I_{ref} currents, the correction circuits are activated to eliminate local minima. Simple two transistor nonlinear current to voltage converters as shown in Fig. 11 part (a) are used for correction circuits. SPICE3e1 simulations are performed with level 2 parameters obtained from MOSIS for a 2-μm, n-well, double-poly, analog BiCMOS process (run N15S of June 18th, 1991).

The SPICE3e1 circuit simulation of a three neuron A/D converter with local minimum correction via the circuit of Fig. 11 is shown in Fig. 12(a), where the curves are for the current outputs of neurons 1, 2, 3, $I_0(1)$, $I_0(2)$, $I_0(3)$, in SPICE3e1 simulations. All circuit simulations are with an upward then downward 4 V peak value ramp input signal over a total time of 2 ms. The least significant digit is shown in the leftmost block, $I_0(1)$, and -4 μA current is used as an unit in our multilevel neural A/D converter design. There are four logic levels in each neuron output. Here, we use negative current outputs for positive number representation. The value representing logic one for neuron 1 is $I_0(1) = -4$ μA, for neuron 2 is $I_0(2) = 4$ times -4 μA, and for neuron 3 is $I_0(3) = 16$ times -4 μA. The D/A outputs of neuron outputs without correction circuit and with correction circuit are shown in Fig. 12(b) and (c) via summing neuron current outputs, $I_0(1)$, $I_0(2)$, and $I_0(3)$, respectively. Fig. 12(b) without the correction circuit is not as desired since it shows a nonlinear stairway curve. However, from Fig. 12(b) and (c), the D/A output of Fig. 12(a) is a linear stairway curve if we ignore pulses in Fig. 12(c). The curve of Fig. 12(c) for the multilevel neural A/D converter with the correction circuit is a quantized downward then upward ramp signal. If we ignore pulses in Fig. 12(c) this is the desired curve that we want since our input analog signal is an upward then downward ramp input signal.

These simulations show us that the multilevel A/D converters are more feasible than binary ones. As compared to the same capacity for representing number, the structure is simpler, and a smaller number of resistors will be needed.

D. VLSI Implementaiton-Chip Design and Measurements

The circuits for multilevel neural A/D converters were designed and integrated via MOSIS using BiCMOS n-well, 2-μ double-poly and double metal technology. Since MOSIS does not optimize the process for npn transistors, the area of npn transistors in this process is much bigger than for comparable CMOS transistors as can be shown in the Magic layout of a multilevel nonlinearity in Fig. 13(a). Also the collector resistance of a MOSIS fabricated npn transistor is large when compared with ordinary npn transistors. However, we can increase a transistor's size to reduce the collector parasitic resistance. The microscope photo for multilevel neural A/D converter with 3 neurons and 6 weights is shown in Fig. 13(b). The chip is a Tiny Chip with 40 pins and die size 2250 μm $\times$ 2220 μm. The supply voltages are $+5$ V and -5 V. Measurements for a four-level neuron output of a 4-level neuron nonlinearity versus input voltage are shown in Fig. 14(a). The input voltage, of from 0 to 2 V is shown in the upper part and the output current is shown in the lower part with a peak-peak current of 48 μA. Since mismatching effects in fabrication need to be considered, the neural weights are laid out as a multiple number of a basic cascode CMOS transistor current mirror. The size ratios of MOS transistors for neural weights are designed to have equal lengths and equal widths. This design tries to reduce the mismatching effect resulting from the fabrication process. A weight of value 4 is calculated as the ratio and two signals in Fig. 14(b). The upper portion of Fig. 14(b) is a multilevel signal input current of peak-to-peak value of 12 μA and the lower portion is the current output of peak-to-peak value 48 μA. About 2% mismatch was observed in this measurement. The correction output currents versus input voltage are shown in Fig. 14(c). The range of input voltage is from -0.5 V to $+0.5$ V. When the input voltage crosses 0, the middle curve shows a positive 6 μA correction current and the bottom curve shows a negative 6 μA correction current.


(a)

(b)

(c)

Fig. 14. Typical A/D component traces (all horizontal scales: time as indicated). (a) A multilevel nonlinearity current output; vertical scale: upper trace, 1 V/div; lower trace, 25 μA/div. (b) A weight of 4 using a current mirror. The vertical scale for both curves is 3.5 μA/div. (c) The correction current outputs versus input voltage are shown here; upper trace: 0.25 V/div ($\pm$1 V full scales); middle and lower traces: 2 μA/div ($\pm$8 μA full scales).

V. Discussion and Conclusions

We have introduced a multilevel neuron and multilevel nonlinearities for its realization. For then, an energy function suited for discrete multilevel neural networks has been proposed and we have shown that multilevel neural networks are feasible. By applying the energy function proposed here, we have demonstrated an appropriate way of designing multilevel neural A/D converters. An implementation of this neuron using current mode transistors is provided, and the application of BiCMOS technology in neural networks is exploited. We have avoided using weight resistors in our circuit since resistors usually need large area and accurate conductance ratios for which a big neural network would be difficult to implement. As Barkan, Smith, and Persky point out in [19], when resistors are used to realize the weights, conductance weights will be affected by voltage x_i across capacitor c_i. Thus an equivalent conductance weight no longer corresponds to a single coupling weight, and each coupling weight influences all the weights.

Our implementation avoid this problem since neuron weighs are decided by MOS transistor geometry ratio.

As indicated by Lee and Sheu [17], most of their VLSI chip is occupied by resistors and switches in so doing. In our multilevel A/D converter, we eliminate the number of weight resistors needed and simplify the correction circuit. The compensation for that is the use more amplifiers and transistors. We consider these will reduce chip area needed since amplifiers and transistors need less chip area than the resistors they replace. In our implementation, we need 3 neurons and 6 weights to implement a 6-bit A/D converter. However, for Lee and Sheu's implementation, 6 neurons and 30 weights would be needed to do the same job. The penalty for our implementation is that the area of multilevel nonlinearity is bigger than regular sigmoidal nonlinearity. However, for a big neural network, the reduced synapse weights should overcome this drawback. Also, some quantum effect devices [21] shown promising in multivalued logic applications may apply to multilevel neural network for reducing the complexity of multilevel nonlinearity implementation. All CMOS circuits are also feasible to design a multilevel neural networks. Since npn transistors are big in chip area for MOSIS fabrication process, we think all CMOS implementation will reduce the chip area needed. We use p-diffusion resistors for input resistors. These resistors occupy a small area on the chip of Fig. 13(b) and the maximum resistor ratio is only 4. The weights connected to the input of each individual neuron shown in Fig. 8 can be scaled, as Lee and Sheu did in [17], so that the maximum weight ratio (the largest weight/the smallest weight) can be reduced. For A/D conversion, weight matrixes are always symmetrical even for a mixed radix number system.

As we can see in Fig. 12(c), some pulses are shown during transitions from one digital level to the next higher digital level. These noises are able to be eliminated after we use simple low pass filters, at current mirror neuron outputs to get rid of pulses in Fig. 12(c). As shown in Fig. 12(d), we obtain the D/A voltage output from the current output Fig. 12(c) by using pairs of resistor and capacitor as low pass filters.

Measurements on individual components for an A/D converter are demonstrated. The next work is to make the full multilevel neural A/D converter.

REFERENCES

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

[2] 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. CAS-33, pp. 3088–3092, May 1986.

[3] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, pp. 4–21, Apr. 1987.

[4] A. N. Michel, and J. A. Ferrel, "Associative memories via artificial neural networks," *IEEE Contr. Syst. Mag.*, pp. 6–17, Apr. 1990.

[5] B. Linares-Barranco, E. Sanchez-Sinencio, R. W. Newcomb, A. Rodriguez-Vazquez, and J. L. Huertas, "A novel CMOS analog neural oscillator cell," in *Proc. IEEE Int. Conf. Circuits and Syst.*, May 1989, pp. 794–797.

[6] W. Banzhaf, "A network of multistate units capable of associative memory and pattern classification," *Phys. D*, vol. 34, pp. 418–426, 1989.

[7] J. Si and A. N. Michel, "Analysis and synthesis of discrete-time neural networks with multi-level threshold-functions," in *Proc. IEEE Int. Symp. Circuits Syst.*, The Westin Stamford and Westin Plaza, Singapore, June 1991, pp. 1461–1464.

[8] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. National Academy of Science*, vol. 79, Apr. 1982, pp. 2554–2558.

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

[10] J. H. Li, A. N. Michel, and W. Porod, "Qualitative analysis and synthesis of a class of neural networks," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 976–986, Aug. 1988.

[11] S. Grossberg, and M. Cohen, "Absolute stability of global pattern information and parallel memory storage by competitive neural networks," *IEEE Trans. Syst. Man, Cybern.*, Sept. to Oct. 1983, pp. 815–826.

[12] R. W. Newcomb, "Neural-type microsystems: Some circuits and considerations," in *Proc. IEEE Conf. Circuits and Computers*, Oct. 1980, vol. 2, pp. 4072–4074.

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

[14] R. L. Geiger, P. E. Allen, and N. R. Strader, *VLSI Design Techniques for Analog and Digital Circuits.* New York: McGraw-Hill, 1990, pp. 318–372, 431–449.

[15] J. Yuh and R. W. Newcomb, "Circuits for multilevel nonlinearities," in *Proc. Int. Joint Conf. Neural Networks*, Baltimore, MD, June 1992, vol. II, pp. 27–32.

[16] J. J. Clark, "Current mode implementation of a self-organization feedforward network," in *Proc. Int. Joint Conf. Neural Networks*, vol. II, Washington, DC, 1990, pp. 118–121.

[17] B. W. Lee and B. J. Sheu, "Design of a neural-based A/D converter, using modified Hopfield network," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1129–1135, Aug. 1989.

[18] B. W. Lee and B. J. Sheu, "Modified Hopfield neural networks for retrieving the optimal solution," *IEEE Trans. Neural Networks*, vol. 2, pp. 137–142, Jan. 1991.

[19] O. Barkan, W. R. Smith, and G. Persky, "Design of coupling resistor networks for neural network hardware," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 756–765, June 1990.

[20] G. Moon, M. E. Zaghloul, and R. W. Newcomb, "An enhancement-mode MOS-controlled linear resistor with large dynamic range," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1284–88, Oct. 1990.

[21] F. Capasso, S. Sen, F. Beltram, L. Lunardi, A. Vengurlekar, P. Smith, N. Shan, R. Malik, and A. Cho, "Quantum functional devices: Resonant-tunneling transistors, circuits with reduced complexity and multi-valued logic," *IEEE Trans. Electron Dev.*, vol. 36. pp. 2065–2081, Oct. 1989.

Jen-Dong Yuh received the B.S. degree from Chung-Yuan Christian University, Taiwan, Republic of China, in 1982, and M.S. degree from University of Maryland, College Park, MD, in 1987 both in electrical engineering. He is currently a Ph.D. candidate in electrical engineering at University of Maryland, College Park, MD.

His current research interest include multilevel neural networks, neural network computing, implementations of neural networks, and VLSI design.

Mr. Yuh is an associate member of Sigma Xi.

Robert W. Newcomb (S'52–M'56–F'72), for photograph and biography please see page 386 of this issue.