USC–SIPI REPORT #268

Performance Optimization in Cellular Neural Networks and Associated VLSI Architectures

by

Sa Hyun Bang

August 1994

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

To my parents,

Soon M. Bang

Bong N. Choi,

and to my wife,

Ann J. Park,

in gratitude of their love

Acknowledgment

I would like to express the sincerest thanks to my research advisor Professor Bing J. Sheu, Director of VLSI Signal Processing Laboratory, for his guidance and support throughout the Ph.D. research work. I wish to extend my deep appreciation to Professor Robert A. Scholtz, Department of Electrical Engineering, and Professor Fred Browand, Department of Aerospace Engineering, for serving as my dissertation committee members. I would also like to thank Professor Alice Parker and Professor Zhen Zhang for serving as my qualifying examination committee members.

I am very grateful to Professor Leonard Silverman, Dean of the Engineering School, Professor Hans H. Kuehl, Chairman of the Electrical Engineering - Electrophysics Department, Professor Melvin A. Breuer, Chairman of the Electrical Engineering-Systems Department, Professor Ted Berger, and Ms. Ramona Gordon and Ms. Anna Fong in the Electrical Engineering Program, for providing me with great research environment. This research work was conducted through connections with several research organizations in the University of Southern California including Center for Neural Engineering (CNE) with Professor Michael Arbib as Director, Signal and Image Processing Institute (SIPI) with Professor Jerry Mendel as Director, MOSIS Service of Information Science Institute (ISI), and industrial institutions such as Samsung Electronics Co., Ltd., Kiheung, Korea, and Samsung Information Systems America (SISA), San Jose, CA. Information exchange with Professor Leon O. Chua of Electrical Engineering and Computer Sciences Department at University of California, Berkeley, and Professor Tamas Roska of Computer and Automation Institute of the Hungarian Academy of Sciences, are very beneficial.

I am thankful to Professor Wonhoo Kim and the faculty of Electrical Engineering Department and Communications Engineering Department at Hankuk Aviation University, Koyang, Korea, for their guidance in my Bachelor- and Master-degree studies. I am also indebted to Dr. Dongkoo Kim, currently at the Yonsei University, Seoul, Korea, for valuable discussions on communications.

It will be the greatest pleasure to remember the period of my stay in the VLSI Signal Processing Laboratory at University of Southern California. I am very grateful to Dr. Bang W. Lee, currently at Samsung Electronics Co. Ltd., Buchun, Korea, for his pioneering work in the neural networks, Dr. Wai-Chi Fang, currently at Jet Propulsion Laboratory, for his work in image processing application of neural networks, Dr. Sudhir M. Gowda, currently at IBM T. J. Watson Research Center, for his helpful insights on the VLSI modeling and simulation, Dr. Josephin C.-F. Chang, currently at Raytheon Semiconductors, for her helpful support, and Dr. Oscal T.-C. Chen for providing good computing facility and assistance. I would like to give my special thanks to Dr. Joongho Choi, currently at IBM T. J. Watson Research Center, for helpful collaboration in my neural network study and analog VLSI implementation. I deeply appreciate Mr. Robert C. Chang and Mr. Tony Wu, for their enthusiasm toward advanced knowledge.

I would like to express special thanks and appreciation to my parents, Soon M. Bang of blessed memory and Bong N. Choi, for their love and patience. I also would like to thank my brothers, Dae Hyun, Yi Hyun, Woo Hyun, and sisters, Choon Wha, Choon Hee. I deeply appreciate my wife, Ann J. Park, for devoted support, encouragement, and love during my Ph.D. study, my son, Steven Heejo, and daughter, Elizabeth Heejin, for their understanding and patience.

Contents

List of Tables

List of Figures

Abstract

A systematic annealing method of finding optimal solutions in recurrent associative neural networks is presented. The Hopfield neural network and cellular neural network are very promising in solving many scientific optimization problems by the use of their collective computational properties. However, the neural networks for optimization are subject to sub-optimal solutions due to the local-minimum problem as in engineering optimization problems. Various techniques and neural networks for global optimization have been suggested. The stochastic methods such as simulated annealing and Boltzmann machine require a tremendous amount of computational resources in executing the algorithm on digital computers.

Paralleled hardware annealing exploited in this dissertation is a highly efficient method of finding globally optimal solutions in recurrent associative neural networks. It is a paralleled, hardware-based realization of effective mean-field annealing while achieving the effects of the matrix nonconvexity method. The speed of convergence can be faster than those of the stochastic methods by several orders of magnitude, and is suitable for those of typical analog very large-scale integration (VLSI) neuroprocessor implementation. The process of global optimization can be described by the eigenvalues of a time-varying dynamic system. The generalized energy function, which serves as the cost function to be optimized, of the network is first increased by reducing the voltage gain of neurons. Then, the hardware annealing searches for the globally minimum energy state by continuously increasing the gain of neurons. The proposed annealing technique is described by applying it to a basic two-neuron network followed by a Hopfield analog-to-digital decision network in which the desired optimal solutions are exactly known. The powerful cellular neural networks are examined to understand the effectiveness of the hardware annealing in solving the problems of energy barriers. In many applications other than optimization, hardware annealing also provides adequate stimulation to frozen neurons caused by ill-conditioned initial states. As a practical example of the neural-based combinatorial optimization, the maximum-likelihood sequence estimation of digital data in communications is successfully investigated. In addition, efficient computing architectures for VLSI and detailed circuit design are presented.

Preface

The organization of this dissertation consists of the following chapters.

Chapter I describes classical methods and neural network approaches for solving combinatorial optimization problems. In addition, several optimization techniques in classical mechanics, optimization, and neural network are reviewed in relation to the proposed paralleled hardware annealing.

In *Chapter II*, the fundamentals of recurrent associative neural networks, Hopfield and cellular neural networks, are reviewed, and then their local minima problem in optimization problems is mentioned. By using a basic and generalized neural network models, paralleled hardware annealing technique for obtaining the optimal solutions is derived and verified.

In *Chapter III*, the hardware annealing is applied to a multi-level Hopfield analog-to-digital decision network, which is a combinatorial optimization circuit with the local minima problem, to show the effectiveness of the proposed method.

More detailed description of the hardware annealing and its application to cellular neural network are given in *Chapter IV*. Network stability and annealing parameters are derived and discussed by using the expressions of generalized network energy and eigenvalues associated with linearized dynamic system. Simulation results of the proposed method in achieving optimal solutions and improved performance are also provided.

In *Chapter V*, a neural network approach for the maximum-likelihood (ML) detection of signals in digital communications is presented. The mapping of the ML problem onto a combinatorial optimization problem is discussed. In addition, the hardware annealing is provided to improve the ML detection performance. By using different networks and neuron models, the performances of the ML detection in basic inter-symbol interference channels are compared.

Hardware design of both plain and annealed cellular neural networks is given in *Chapter VI*. An efficient network architecture and detailed circuit elements for VLSI implementation are designed and simulated.

Chapter I

Introduction

1.1 Classical Optimization Problems

An optimization problem consists of looking for a set of variables $s = \{s_1, s_2, \cdots, s_n\}$ which minimizes a cost function $J(s)$ while obeying a number of constraints $g_k(s)$, $k = 1,2,\cdots,m$. Many optimization problems involve the minimization of a quadratic cost function;

$$\min_{x \in P} f(x) = \min_{x \in P}\left(c^T x + \frac{1}{2} x^T \mathrm{Q} x\right), \tag{1.1}$$

subject to the inequality constraint $\mathrm{A}x \le b$ or equality constraint $\mathrm{A}x = b$. Here, P is the bounded polyhedron [1], Q is an $n \times n$ symmetric matrix, A is an $n \times m$ matrix, and $x, c, b \in R^n$. Based on the characteristics of the matrix Q, optimization problems can be divided into three kind of objective functions as;

1. convex,
2. indefinite, or
3. concave.

The convex objective function $f(x)$ is a bowl-shaped $(n+1)$-dimensional surface with n degree of freedom represented by the state of x. This surface is characterized by a unique minimum. There are many convex quadratic optimization problems in adaptive signal processing. For example, solving the Wiener-Hopf equation for linear optimal filter is equivalent to finding the minimum of the mean-squared error, which is a quadratic function of the filter tab weights. If the matrix Q is indefinite, there may exist multiple minima and the solution to indefinite quadratic problem occurs on the boundary point of the hypercube, but not necessarily a vertex. For a positive definite Q, the energy function $f(x)$ is a scalar-valued, concave function of the state vector x and can be visualized as an inverted bowl-shaped $(n+1)$-dimensional surface. The solution to a concave quadratic problem always occurs at a vertex of P. Many engineering problems belong to the optimization of concave objective functions in which the domain x_k, $k=1,2,\cdots,n$, takes a finite number of values in a set, e.g., -1 or $+1$.

In many branches of science and technology one often encounters difficult optimization problems which have combinatorial complexity. For such problems there is a large but finite set of possible solutions, of which we want to find one that globally minimizes the cost function involved. Those include the map coloring problem [25], graph partition [16-20,23-25], assignment problem [25], and the traveling salesman problem (TSP) [11-14,25]. The combinatorial optimization problems can be mapped onto neural networks by constructing suitable energy functions and then transforming the problem of their minimization into associated systems of differential or difference equations.

In general, the computational energy function in optimization can often be expressed as sum of two terms:

$$E = E_{\text{cost}} + E_{\text{constraint}}, \tag{1.2}$$

where E_{cost} is the cost energy function and $E_{\text{constraint}}$ is the cost energy associated with constraint satisfaction. These two terms of the energy function compete with each other. To obtain the minimum of E of (1.2), we need to minimize the cost function while simultaneously satisfying the constraint as much as possible. Moreover, the constraint energy may consist of several terms and (1.2) can be rewritten as

$$E(x) = E_C(x) + \sum_{i=1}^{M} \lambda_i E_i(x), \tag{1.3}$$

where $x = [x_1\ x_2 \cdots x_n]^T$, $x \in \{-1,+1\}^n$, $E_C(x)$ is the cost energy function, and $E_i(x)$ is the penalty function term representing the violation for the i-th constraint. The penalty parameters $\lambda_i > 0$, $i = 1,2,\cdots,m$, ensure an appropriate balance between two simultaneous minimizations of the energy functions. For a wide class of optimization problems, the computational energy function (1.3) can be transformed into a quadratic energy function

$$E(x) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j - \sum_{i=1}^{n}\alpha_i x_i = -\frac{1}{2}x^T\mathbf{W}x - x^T\boldsymbol{\alpha}, \tag{1.4}$$

where

$$x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T,\ x_i \in \{-1,+1\},$$
$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \end{bmatrix}^T,$$
$$\mathbf{W} = \left[w_{ij},\ 1 \le i,j \le n\right].$$

The weight matrix $\mathbf{W}$ and external input vector $\boldsymbol{\alpha}$ are completely determined by the associated energy function to be minimized. Let us define an n-dimensional unit hypercube as $D^n = \{x \in R^n : -1 \le x_i \le +1, i = 1,2,\cdots,n\}$. Depending on the optimization problem to be solved, the matrix $\mathbf{W}$ can be 1) negative definite (semi-definite), 2) indefinite, or 3) positive definite (semi-definite) which corresponds to convex, indefinite, or concave quadratic optimization, respectively.

1.2 Combinatorial Optimization using Neural Networks

The range of applications of the theory of recursive neural networks is not limited to biology but could include those for solving optimization problems as well. The artificial neural network for optimization problem has been pioneered by Tank and Hopfield [2-4]. They showed that certain highly-interconnected networks of nonlinear analog neurons can be very effective in solving difficult optimization problems. The information pertaining to the problem to be solved is built into the network as an energy function which is to be optimized. The mapping of an optimization problem onto a neural network is to build an energy landscape $E(s)$ in the phase space of neuronal states such that the deepest minima of E are solutions to the problem. Once the mapping has been achieved, one lets the system evolve according to the neuronal dynamics until a steady state is obtained. The great advantage of this method is that a physical device for solving the problem can be constructed to generate a solution without the necessity of learning iterations.

The neuronal dynamics of Hopfield network can be described by a set of differential equations;

$$C\frac{du_i}{dt} = -T_i u_i + \sum_j W_{ij} v_j + \theta_i, \quad i = 1,2,\cdots,n, \tag{1.5}$$

$$v_i = sign(u_i) = sign\left(\sum_j W_{ij} v_j + \theta_i\right), \quad v_i \in \{-1,+1\}.$$

The Lyapunov [5] or generalized energy function $E(t)$ associated with (1.5) is defined as

$$E = -\frac{1}{2}\sum_i \sum_j W_{ij} v_i v_j - \sum_i \theta_i v_i, \tag{1.6}$$

which is to be a non-increasing function of time provided that;

1. the dynamics is serial, i.e., one neuron is updated at a time,
2. the interaction is symmetrical, i.e., $W_{ij} = W_{ji}$, $\forall i, j$, and
3. the self-connections vanish, i.e., $W_{ii} = 0$, $\forall i$.

In (1.5) and (1.6), u_i and v_i is the states of the i-th neuron input and output, respectively, W_{ij} is the amount of interaction or weight between the i-th neuron input and j-th neuron output, and θ_i is the external input including the bias. The energy function (1.5) can be rewritten in vector and matrix form as

$$E = -\frac{1}{2}v^T\mathbf{W}v - v^T\boldsymbol{\theta}, \tag{1.8}$$

where $v = [v_1 \ v_2 \ \cdots \ v_n]^T \in \{-1,+1\}^n$

$\mathbf{W} = [T_{ij}, 1 \le i,j \le n] = \mathbf{W}^T$, $T_{ii} = 0$

$\boldsymbol{\theta} = [\theta_1 \ \theta_2 \ \cdots \ \theta_n]$.

The equation (1.8) is a scalar-valued, quadratic function of v and has the same form as the cost functions in optimization problems. The collective computational properties of the Hopfield neural network for seeking a stable equilibrium can be utilized in solving many of difficult optimization problems. When an optimization problem is appropriately mapped onto the energy function of the form (1.8), it can be solved autonomously and an output $v \in D^n$ that minimizes (1.8) will be produced in the steady state. Here, we restrict ourselves to the combinatorial optimization problems in which each element of the desired output has a finite number of discrete values in a set, e.g., two-level bipolar output $v_i \in \{-1,+1\}$, or m-ary unipolar output $v_i \in \{0,1,\cdots,m-1\}$, etc. Many science and engineering optimization problems belong to these categories. On the other hand, the optimization problem with inequality constraints, i.e., linear or nonlinear programming typically requires an additional network for constraints satisfaction [4,27-31].

Many different neural network approaches and techniques have been proposed to solve a large variety of combinatorial optimization problems. Among them the most popular and promising are the following approaches;

1. Hopfield neural networks [2-4,12,16-20,27],
2. cellular neural networks (CNN) [59-63,112,113],
3. simulated annealing (SA) [37-42],
4. mean field annealing (MFA) [43-46], and
5. competitive neural networks [25].

The CNN is basically a multi-dimensional array of locally-connected cells and has the collective computational properties similar to those of the Hopfield network. Together with the Hopfield neural network, it will be explored in detail in this dissertation.

1.3 Global Optimization

In general, an indefinite or concave quadratic function of the form (1.8) in combinatorial optimization problems, contains a large number of local minima [1] (Figure 1-1). For instance, in a neural network of binary neurons, any of at most 2^n vertices of the unit hypercube D^n can be a solution to the problem, for which the energy function is locally minimized [4,32-35]. In some applications such as the content addressable memory (CAM), those minima are utilized to store the patterns which can be recalled by the corresponding recall keys. However, many optimization problems require the global minimization of the energy function over all possible solutions. In this case, a neural network may fail to find the desired solution if no provision is made on avoiding those minima at which the solutions are sub-optimal. In Table 1.1, the performance of a neural network approach is compared with those of several conventional algorithms in solving 50, 100, and 200-city Traveling Salesman Problems (TSP) [11]. In all cases, the neural network has little longer average tour lengths than others except simulated annealing algorithm, which may be a direct consequence of local minima. On the other hand, the conventional algorithms are designed to deal with combinatorial optimization problems.

The annealing technique in combinatorial optimization problems and neural networks is originated from the statistical mechanics which is the central discipline of condensed matter physics. Figure 1-2 shows the relationships among various techniques related to the annealing theory, in the areas of statistical mechanics, optimizations, and neural networks. The followings are brief review of these techniques.

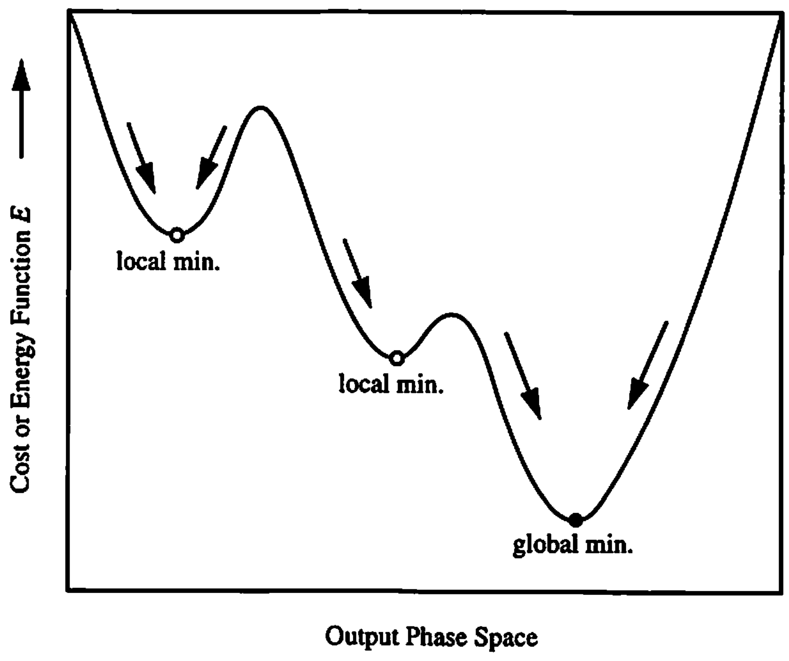


Figure 1-1: Local minima in optimization problems.

Table 1.1 Comparison of various optimization algorithms on N-city Traveling Salesman Problem (TSP).

Optimization Algorithms	Average Tour Lengths		
	N=50	N=100	N=200
Genetic Algorithm	5.58	7.43	10.49
Simulated Annealing (SA)	6.80	8.68	12.79
Elastic Net	5.62	7.69	11.14
Hybrid (SA+Search)	-	7.48	10.53
Neural Network [1]	6.61	8.58	12.66

1. Potts neural net (mean-field approximation) with fixed temperature [46].

1.3.1 Statistical Mechanics

A fundamental question in statistical mechanics concerns what happens to the system in the limit of low temperature. In practical contexts, low temperature is not a sufficient condition for finding ground states of matter. To determine the low-temperature state of a material, experiments are done by first melting the substance, lowering the temperature slowly, and then spending a long time at temperatures in the vicinity of the freezing point. At a high temperature, all particles or atoms of a metal lose the solid phase so that the positions are random according to statistical mechanics. The particles of the molten metal tend toward the minimum energy state at which the state of metal is highly ordered and has the structure of a defect-free crystal lattice, but high thermal energy prevents this. In order to achieve the defect-free crystal, the metal is first heated to an appropriate temperature above the melting point and then cooled slowly until the metal freezes into a defect-free crystal. The slow cooling is usually necessary to avoid dislocations and other crystal lattice disruptions. The process of controlling the temperature for the global energy minimization is called metallurgical annealing.

Metropolis [36] introduced a simple algorithm that can be used to provide an efficient simulation of a collection of atoms in equilibrium at a given temperature. In each step of the algorithm, an atom is given a small random displacement and the resulting change, ΔE, in the energy of the system is computed. If $\Delta E \leq 0$, the displacement is accepted, and the configuration with the displaced atom is used as the starting point of the next step. The case $\Delta E > 0$ is treated probabilistically. The probability that the configuration is accepted is $p(\Delta E) = \exp(-\Delta E / k_B T)$, where k_B is the Boltzmann's constant. A uniformly distributed random number N is selected in the interval (0,1) and compared with $p(\Delta E)$. The new configuration is retained if $N < p(\Delta E)$, and the original

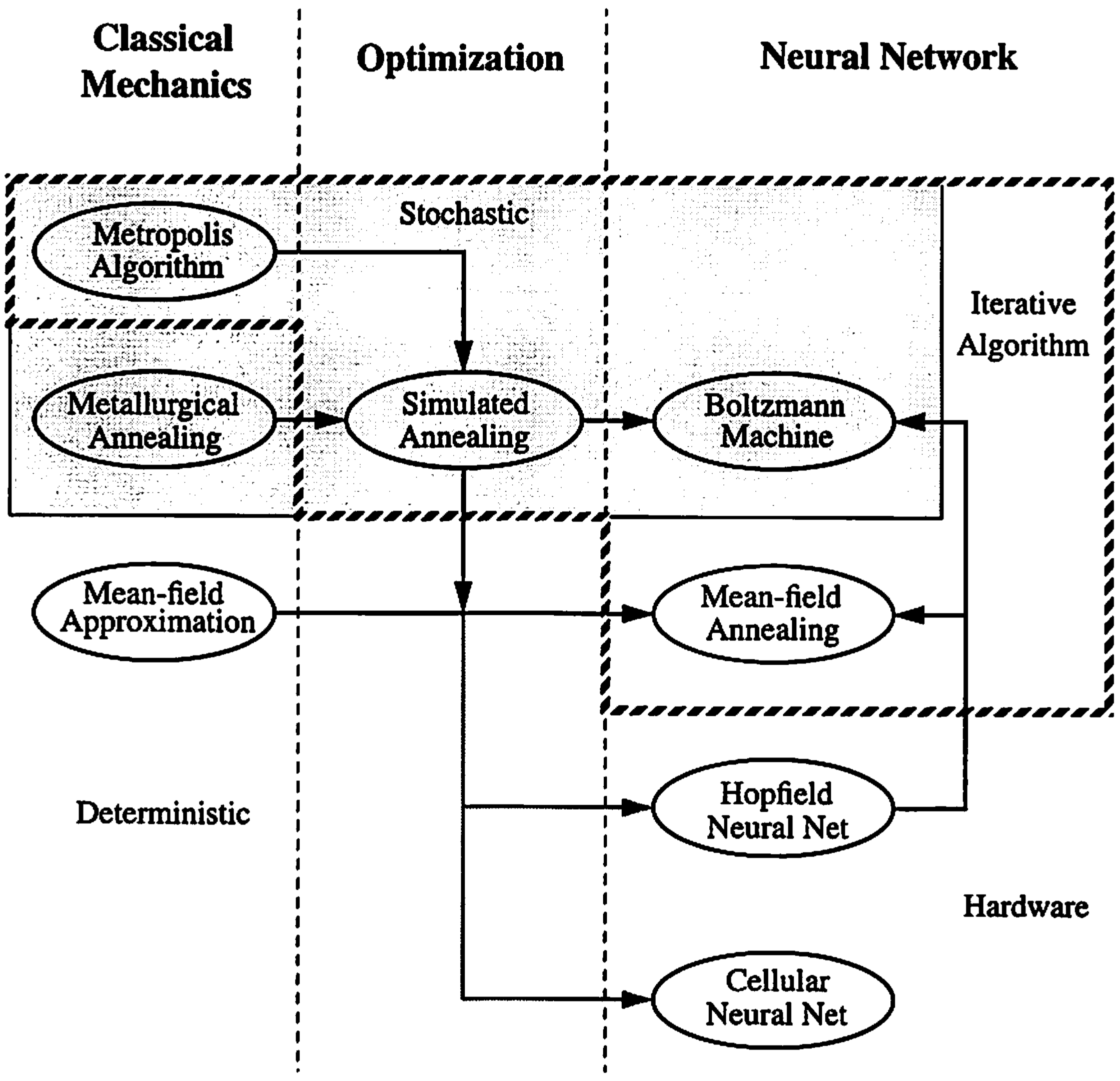


Figure 1-2: Relationships among optimization techniques.

configuration is used to start the next step, otherwise. By repeating the step many times, one simulates the thermal motion of atoms in thermal contact with a heat bath at a given temperature. The choice of $p(\Delta E)$ has a consequence that the system evolves into a Boltzmann distribution

$$p(v) = \frac{e^{-E(v)/T}}{\sum_{v} e^{-E(v)/T}}, \tag{1.9}$$

where T is a controlling parameter called computational temperature.

1.3.2 Simulated Annealing (SA) and Boltzmann Machine

Using the cost function in place of the energy function and defining configurations by a set of parameters $\{v_i\}$, $i = 1,2,\cdots,n$, in optimization problems, simulated annealing (SA) implements the Metropolis procedure with artificial thermal noise which is gradually decreased in time. This noise allows occasional hill-climbing interspersed with descents. At a high temperature, the probability of moves to higher energy levels is large. But at a low temperature, the probability is low, i.e., fewer uphill moves are allowed as the temperature decreases. The possible chances of getting stuck in a worse local minimum are avoided in a controlled fashion by an annealing schedule. This strategy has an analogy to the physical behavior of metallurgical annealing. More importantly, gross features of the eventual state of the system appear at high temperatures but fine details develop at lower temperatures. Interesting features of the SA are its general applicability and its ability to obtain solutions close to an optimum. It has been applied to a wide variety of combinatorial problems in such diverse areas as engineering, VLSI design, and operational research [40-42].

The Boltzmann machine [23,26,40] is a kind of stochastic feedback neural network consisting of binary neurons connected mutually by symmetric weights. It is an energy minimization network consisting of statistical neurons which appear probabilistically in one of two states -1 or $+1$. It uses the energy function of the form (1.8) and the simulated annealing approach to locate energy function minima. Simulated annealing in the Boltzmann machine is a stochastic strategy for searching the state of neurons corresponding to the global minimum of the energy function. For the energy function (1.8), the probability of a particular neuron output being -1 or $+1$ is given by [23,35]

$$p(v_i = \pm 1) = \frac{e^{\pm u_i/T}}{e^{+u_i/T} + e^{-u_i/T}}, \tag{1.10}$$

where u_i is the neuron input. Random displacement of the network state is done by using the number N_i selected from a uniformly distributed interval (0,1) as the external input θ_i in (1.8). Therefore, at time step k, the new neuron output v_i^k for the neuron input u_i^{k-1} is given by

$$v_i^k = \tanh\left(\gamma u_i^{k-1}\right) = \tanh\left\{\gamma\left(\sum_{j=0}^{n-1} W_{ij} v_j^{k-1} + N_i\right)\right\}, \tag{1.11}$$

where the gain γ is high enough so that (1.11) approximates the signum function.

1.3.3 Mean-Field Annealing

In physics, the behavior of systems of particles or spins in thermal equilibrium is often simplified by an analytic expression called the mean-field approximation [46]. In a similar manner, the stochastic simulated annealing can be approximated or emulated by an analytic version, the mean-field annealing (MFA). The MFA can be based on a spin glass model [46] or the Hopfield neural network model [43,44]. In the latter case, the stochastic binary neurons (1.11) in the Boltzmann machine are replaced by analog neurons with continuous outputs $v = \left[v_1 \ v_2 \ \cdots \ v_n\right]^T$, $-1 \le x_i \le +1$. The energy function called the free energy for the continuous output value is given as

$$E(v) = -\frac{1}{2}\sum_i \sum_j w_{ij} v_i v_j - \sum_i \theta_i v_i . \tag{1.12}$$

The continuous variable v_i of analog neuron will represent the average value of the binary variable $x_i = -1$ or $+1$ at the temperature T. From (1.10), the MFT approximation takes the form

$$\begin{aligned} v_i &= p(v_i = +1) - p(v_i = -1) \\ &= \tanh\left(\frac{u_i}{T}\right) = \tanh\left\{\frac{1}{T}\left(\sum_{j=0}^{n-1} W_{ij} v_i + \theta_i\right)\right\}. \end{aligned} \tag{1.13}$$

In this way, the complex stochastic process of simulated annealing has been approximated by a system of nonlinear deterministic equations (1.13) and the gain is varied to obtain a similar effect of controlling temperature in the metallurgical annealing on the deterministic network. The level of optimization achievable by MFA is comparable to that of SA, but 1-2 orders of magnitude fewer relaxation iterations are required [46].

1.3.4 Parallel Algorithms

The SA is a stochastic optimization technique based on the classical metallurgical annealing and its performance has been shown to be good in many applications [40,44]. As is common to most algorithms based on randomization techniques, the algorithm usually requires large amounts of computation time. Paralleled implementations of SA such as a systolic array algorithm and a clustering algorithm can reduce the computational burden significantly. However, for large-scale problems, the use of these algorithms on massively parallel multiprocessor systems becomes impractical and communication bottlenecks drastically reduce the efficiency [44].

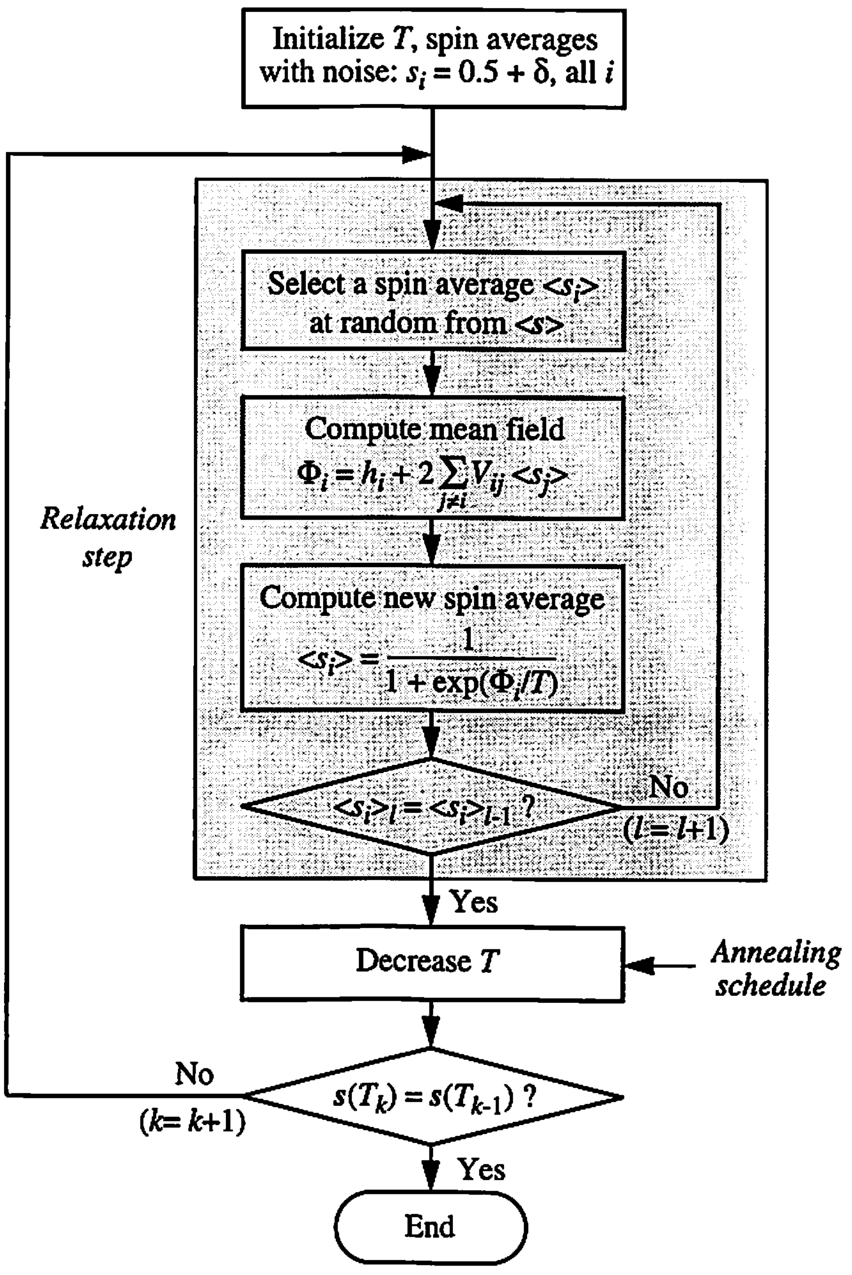
The characteristic features of neural network approaches to combinatorial optimization problems are;

- *distributed memory* with reduction of communication bottlenecks, and
- *massive parallelism* for fast computing.

The Boltzmann machine belongs to the class of neural network models and is a representative of connectionist models. It provides a computational model that is suitable for massively parallel execution of the SA algorithm. The MFA algorithm, a deterministic version of SA, has been successfully ported to parallel multiprocessor systems such as a ZIP array processor with near-linear speedup [46]. However, these algorithms are also iterative procedures for solving a set of stochastic or deterministic equations. Figure 1-3 shows the flowchart of a MFA algorithm [46] using Ising Hamiltonian of spins, the mean-field approximation model of spin glass. A relaxation step indicated by a shaded area consists of many iterations at a specific *T* for finding a stable equilibrium, and is repeated many times until a global minimum is obtained. On the other hand, the proposed hardware-based annealing is directly incorporated with a neural network. It involves a single network evolution that corresponds to the relaxation step in MFA, therefore can achieve a very fast convergence speed via true parallelism. Moreover, the convergence speed of an annealed neural network hardware is almost independent of the network size.

Ising Hamiltonian of N spins: $s = \{s_1, s_2, ..., s_N\}$

$$H(s) = \sum_i h_i s_i + \sum_i \sum_{j \neq i} V_{ij} s_i s_j, \quad V_{ij} = V_{ji}, s_i = 0 \text{ or } 1$$



Initialize T, spin averages with noise: $s_i = 0.5 + \delta$, all i
Select a spin average $<s_i>$ at random from $<s>$
Compute mean field $\Phi_i = h_i + 2\sum_{j \neq i} V_{ij} <s_j>$
Compute new spin average $<s_i> = \frac{1}{1 + \exp(\Phi_i/T)}$
$<s_i>_l = <s_i>_{l-1}$?
No
($l = l+1$)
Yes
Relaxation step
Decrease T
Annealing schedule
$s(T_k) = s(T_{k-1})$?
No
($k = k+1$)
Yes
End

Figure 1-3: Flowchart of mean-field annealing (MFA) algorithm [46] from naive mean field approximation model of spin glass.

Chapter II

Recurrent Associative Neural Networks

An associative neural network is a network with essentially a single functional layer designed to map one set of vectors $x_1, x_2, \cdots, x_L$ into another set of vectors $y_1, y_2, \cdots, y_L$, where $x_k \in R^n$ and $y_k \in R^m$ for $k = 1,2,\cdots,L$. In a *recurrent* associative network, the output signals of the processing elements of the layer are connected to those elements as input signals. The insertion of the vector x entered through separate input connections to the processing elements, causes the feedback loop of the network to become active. It possesses the following properties [26]. First, given any initial state, the network should always converge to some stable state. Second, the stable state to which the network converges should be the one closest to the initial state, as measured by some metric. Third, it should be possible to have as many stable states as desired.

2.1 Hopfield Neural Networks

2.1.1 Continuous-Valued Neuron Model

The Hopfield neural network belongs to the class of recurrent associative network in which the dynamics of evolution plays an important role. It has been developed by using a electrical circuit model consisting of resistors, capacitors, and nonlinear amplifiers as neurons as shown in Figure 2-1. By applying the Kirchhoff's Current Law, the network can be described by a set of nonlinear differential equations

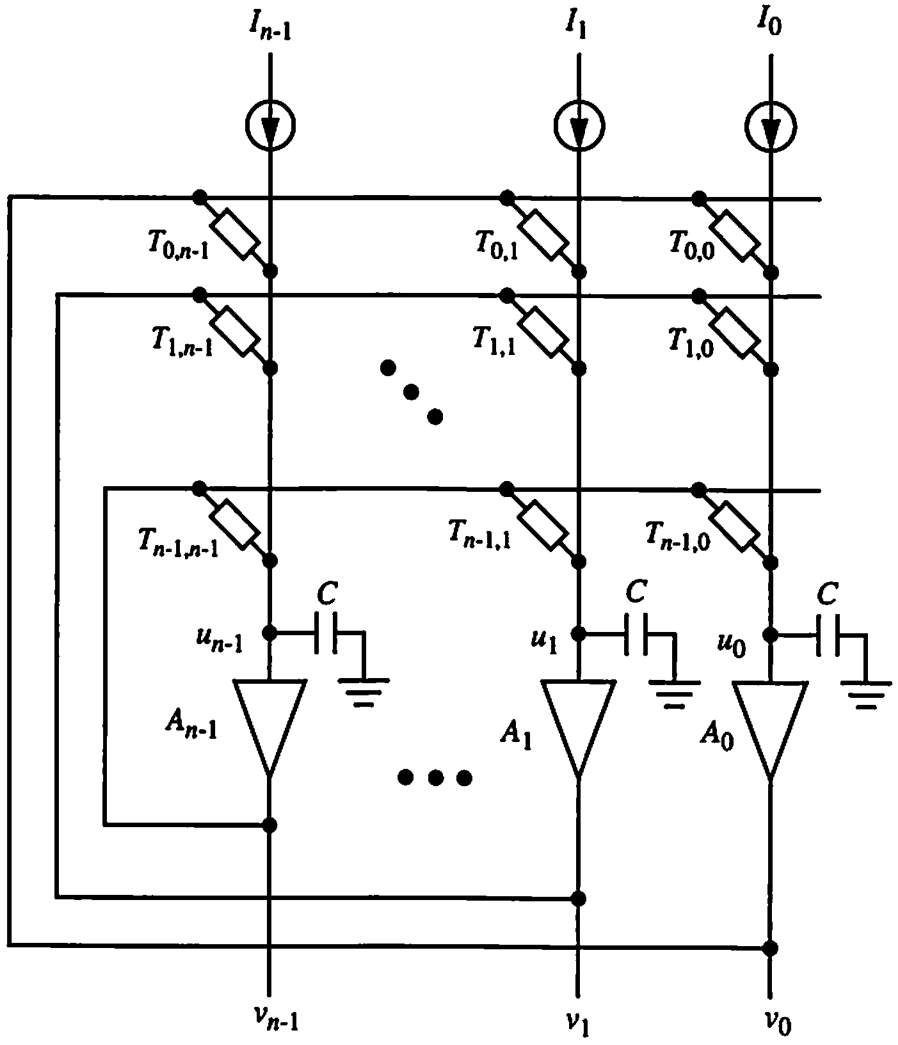


Figure 2-1: Hopfield neural network.

$$C_i \frac{du_i(t)}{dt} = -\frac{1}{R_i} u_i(t) + \sum_{j=1}^{n} T_{ij} v_i(t) + I_i, \qquad i = 1,2,\cdots,n \tag{2.1}$$

where $u_i(t)$ is the neuron input, $v_i(t) = f(u_i(t))$ is the corresponding output through a nonlinear transfer function $f(\cdot)$, $T_{ij} = 1/R_{ij}$ is the transconductance between the i-th neuron input and the j-th neuron output, R_i is the equivalent input resistance, and I_i is the external bias input. Without loss of generality, it is assumed that $y = f(x)$ is monotonically increasing and differentiable, i.e., $df/dx \geq 0$, $\forall x \in \boldsymbol{R}$, and its inverse function $x = f^{-1}(y)$ exists in the possible range of x. Figure 2-2 (a) shows the transfer characteristics of the sigmoid function which is widely used as the nonlinearity. Note that if the ideal current source I_i is replaced by a voltage source with a resistance $R_{i0} = 1/T_{i0}$, then $T_i = 1/R_i = \sum_{j=1}^{n} |T_{ij}| + |T_{i0}|$.

The stability of a nonlinear dynamical system is described by the Lyapunov function [2,3] or generalized energy function. For the system of (2.1), it can be given as

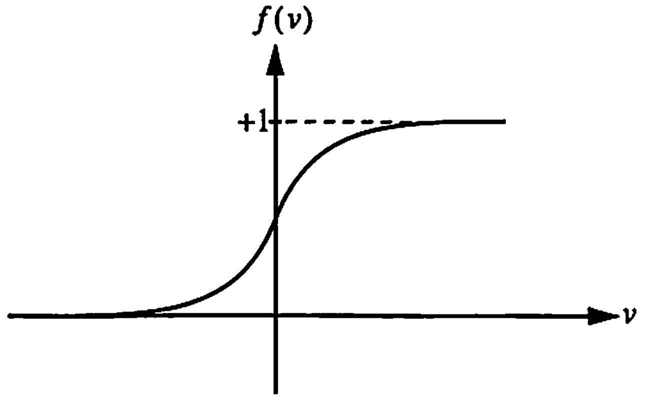
$$E(t) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} T_{ij} v_i(t) v_j(t) + \sum_{i=1}^{n} T_i \int_0^{v_i} f^{-1}(v) dv - \sum_{i=1}^{n} I_i v_i(t). \tag{2.2}$$

Each component of the energy function consists of three terms corresponding to the feedback of neuron output, neuron itself, and input. The integral expression in the second term represent the area of the function $x = f^{-1}(y)$ when integrated from $y = 0$ to $y = v_{yij} \leq 1$. Assuming that the connection is symmetric $T_{ij} = T_{ji}$, the time derivative of the energy function (2.2) is given by

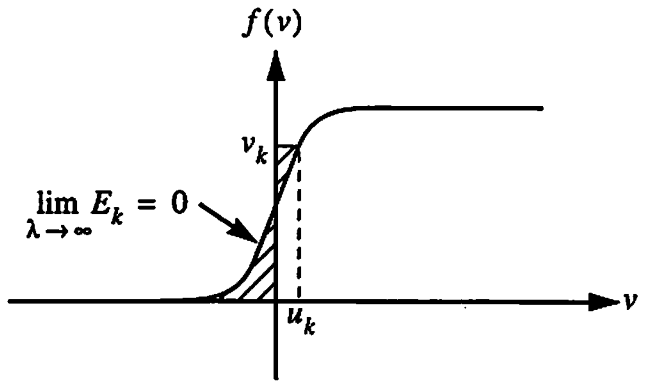
$$\frac{dE}{dt} = \sum_{i=1}^{n} \frac{\partial E}{\partial v_i} \frac{dv_i}{dt} = \sum_{i=1}^{n} \left(-\sum_{j=1, j\neq i}^{n} T_{ij} v_j + T_i f^{-1}(v_i) - I_i \right) \frac{dv_i}{dt}. \tag{2.3}$$

By using (2.1), $f^{-1}(v_i) = u_i$, and $dv_i/dt = (\partial f/\partial u_i) \times (du_i/dt)$, (3.6) is shown to be

$$\frac{dE}{dt} = -\sum_{i=1}^{n} C_i \frac{du_i}{dt} \frac{dv_i}{dt} = -\sum_{i=1}^{n} C_i \frac{\partial f}{\partial u_i} \left(\frac{du_i}{dt} \right)^2 \leq 0, \tag{2.4}$$



(a) Binary sigmoid nonlinearity.



(b) Neuron Energy: $E_k = \int_0^{v_k} f^{-1}(\omega)\, d\omega$

Figure 2-2: Sigmoid function and its 'generalized energy'

because $C_i > 0$ and the neuron transfer function is non-decreasing, i.e., $\partial f/\partial u_i \geq 0,\ \forall i$. Therefore, the state of the Hopfield neural network changes in such a way that the energy function is decreased. In the steady-state, the network reaches a stable equilibrium, at which $du_i/dt = 0$ and $dE/dt = 0$. For a guaranteed binary saturated output in the steady state, the continuous-value neuron model is replaced by an infinite-gain, threshold neuron for which the second term in (2.2) vanishes. In this case, the self-connections need to be zero for stable operation, i.e., $T_{ii} = 0,\ \forall i$ [2,3].

2.1.2 Threshold Neuron Model

When the neuron gain is very high, the continuous-valued neuron acts like a threshold neuron and the second term in (2.2) is negligible. In hardware implementations, this is always the case because the hard-limiting device has smooth transitions. It is related to the nature of the neuronal dynamics;

$$v_i = sign\left(\sum_j W_{ij} v_j + \theta_i\right), \qquad v_i \in \{-1,+1\}. \tag{2.5}$$

The energy $E(t)$ associated with (2.5) is defined as

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} v_i v_j - \sum_{i=1}^{n} \theta_i v_i . \tag{2.6}$$

In (2.5) and (2.6), v_i is the state of the i-th neuron output, W_{ij} is the amount of interaction or weight between the i-th neuron input and j-th neuron output, and θ_i is the external input including the bias. The incremental energy ΔE_i for the output change Δv_i of a specific neuron i is easily shown to be

$$\Delta E_i = -\Delta v_i \left(\sum_{j=1}^{n} W_{ij} v_j + \theta_i\right) - \frac{1}{2} W_{ii} (\Delta v_i)^2, \tag{2.7}$$

where the symmetry condition $W_{ij} = W_{ji}$ is used. If $W_{ii} = 0$, then the second term is zero and $\Delta E_i < 0$ because, from (2.5), Δv_i and the quantity in the parenthesis have the same sign. Therefore, for a simultaneous change of all neurons, $\Delta E = \sum_{i=1}^{n} \Delta E_i < 0$ and the network always operates in such a way that the energy function (2.6) is decreased.

The symmetry of synaptic weights with vanishing diagonal elements is a sufficient condition for the stability of a Hopfield neural network with threshold neurons. However, only the symmetry condition is required in the network with continuous-valued neurons.

2.1.3 Local Minima in Hopfield Neural Networks

We can re-write the energy function (2.2) in vector and matrix form as

$$E = -\frac{1}{2} v^T \mathbf{W} v + \mathbf{T}\kappa - v^T \theta, \tag{2.8}$$

where $v = [v_1 \ v_2 \ \cdots \ v_n]^T \in D^n = \{v \in R^n : -1 \le v_i \le +1, i = 1,2,\cdots,n\}$

$\mathbf{W} = [T_{ij}, 1 \le i,j \le n] = \mathbf{W}^T$, $T_{ii} = 0$

$\mathbf{T} = diag(T_i)$

$\kappa = [\kappa_1 \ \ \kappa_2 \ \ \cdots \ \ \kappa_n]^T$, $\kappa_i = \int_0^{v_i} f^{-1}(\upsilon) d\upsilon$

$\theta = [I_1 \ \ I_2 \ \ \cdots \ \ I_n]$.

The equation (2.8) is a scalar-valued, quadratic function of v and has the same form as the cost functions in the optimization problems. The collective computational properties of the Hopfield neural network for seeking a stable equilibrium can be utilized in solving many of difficult optimization problems. When an optimization problem is appropriately mapped onto the energy function of the form (2.8), it can be solved autonomously and an output $v \in D^n$ that minimizes (2.8) will be produced in the steady state. Here, we restrict ourselves to the combinatorial optimization problems in which each element of the desired output has a finite number of discrete steps, e.g., two-level bipolar output $v_i \in \{-1,+1\}$, or m-ary unipolar output $v_i \in \{0,1,\cdots,m-1\}$, etc. Many science and engineering optimization problems belong to these categories. On the other hand, the optimization problem with inequality constraints, i.e., linear or nonlinear programming typically requires an additional network for constraints satisfaction [25,27-31]. From (2.8), the gradient of E with respect to $\boldsymbol{v}$ is given by

$$\begin{aligned}\nabla_{\boldsymbol{v}} E = \frac{dE}{d\boldsymbol{v}} &= -\frac{1}{2}\left(\mathbf{W}^T \boldsymbol{v} + \mathbf{W}\boldsymbol{v}\right) + \mathbf{T}\boldsymbol{v} - \boldsymbol{\Theta} \\ &= -\mathbf{W}\boldsymbol{v} + \mathbf{T}\boldsymbol{v} - \boldsymbol{\Theta} = -C\frac{d\boldsymbol{u}}{dt}. \end{aligned} \tag{2.9}$$

Here, $d\kappa_i/dv_i = f^{-1}(v_i) = u_i$ is used. The equation (2.9) indicates that the dynamics of the network is equivalent to the directional change of the energy function and a stable equilibrium $\boldsymbol{v} = \boldsymbol{v}_0$ is reached when $\nabla_{\boldsymbol{v}} E = d\boldsymbol{u}/dt = \mathbf{0}$, where $\boldsymbol{v}_0$ is the solution of $\nabla_{\boldsymbol{v}} E = \mathbf{0}$.

Since $\mathbf{W}$ is a real symmetric matrix, it can be diagonalized by a set of orthonormal vectors

$$\mathbf{W} = \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T \tag{2.10}$$

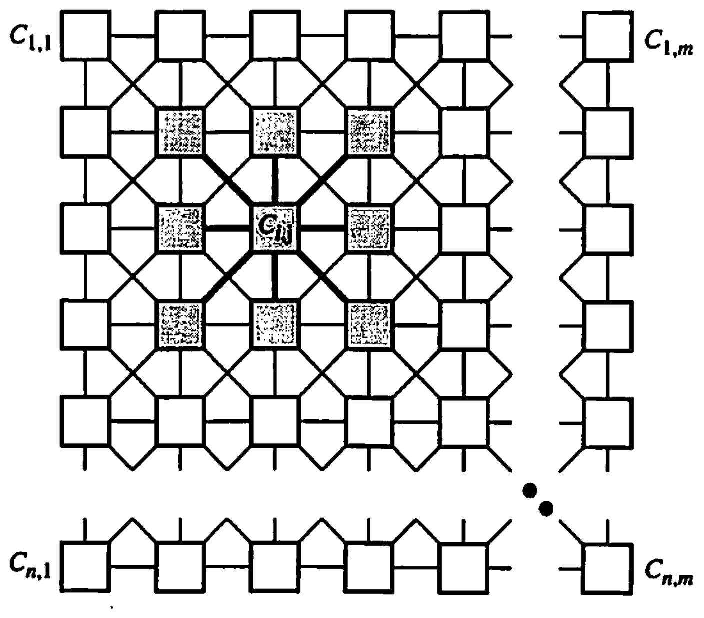
where $\boldsymbol{\Lambda} = diag(\lambda_k)$ is an n-by-n diagonal matrix with eigenvalues λ_k, $k = 1,2,\cdots,n$, on the diagonal and $\mathbf{E} = [e_1 \,\vert\, e_1 \,\vert \cdots \vert\, e_n]$ is an n-by-n matrix with the corresponding eigenvectors $\boldsymbol{e}_k$, $k = 1,2,\cdots,n$, in each column. It is well known from the theory of linear algebra that for diagonalizable W, the sum of the diagonal elements is equal to the sum of eigenvalues, i.e., $\sum_{i=1}^{n} w_{ii} = \sum_{i=1}^{n} \lambda_i$. Therefore, for vanishing diagonal elements of W in the Hopfield neural network, there exist some positive and negative eigenvalues whose sum is identically zero. The energy E is an indefinite function of $\boldsymbol{v}$ in a compact set D^n. In some directions, the energy value decreases while it increases in the other directions.

As a result, the Hopfield neural network can be a tool for an indefinite optimization problem, in which two important observations can be made. First, the stable equilibrium v_0 is not guaranteed to be at the corners of the unit hypercube, i.e., $v_0 \notin \{-1,+1\}^n$ in general. However, if the neuron gain is high, v_0 is close to the corner of the hypercube [2,3] as can be seen from (2.9). Secondly, for given $\mathbf{W}$ and $\boldsymbol{\theta}$, some corners of D^n may not be the stable equilibria because the energy value at the corner is higher than that of its vicinity.

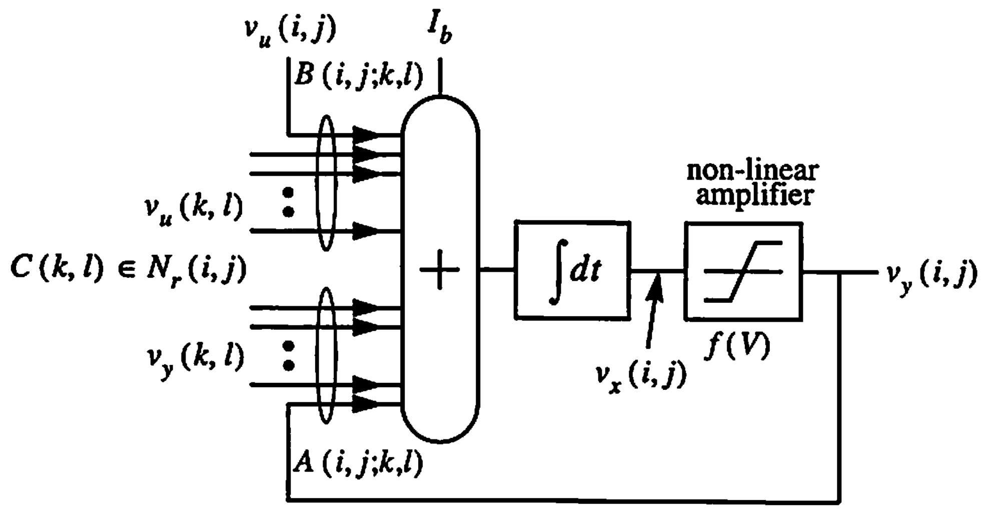
2.2 Cellular Neural Networks

2.2.1. General architecture

A cellular neural network (CNN) is a continuous- or discrete-time artificial neural network that features a multi-dimensional array of neuron cells and local interconnections among the cells. The basic CNN proposed by Chua and Yang [59,60] in 1988 is a continuous-time network in the form of an n-by-m rectangular-grid array where n and m are the numbers of rows and columns, respectively. Each cell in a CNN corresponds to an element of the array. However, the geometry of the array needs not to be rectangular and can be such shapes as triangle or hexagon [65]. A multiple of arrays can be cascaded with an appropriate interconnect structure to construct a multi-layered CNN. The r-th neighborhood cells $N_r(i,j)$ of a cell $C(i,j)$, $1 \le i \le n$, $1 \le j \le m$, are defined as the cells $C(k,l)$, $1 \le k \le n$, $1 \le l \le m$, for which $|k-i| \le r$ and $|l-j| \le r$. The cell $C(i,j)$ has the direct interconnections with $N_r(i,j)$ through two kinds of weights, i.e., the feedback weights $A(k,l;i,j) / A(i,j;k,l)$ and feedforward weights $B(k,l;i,j) / B(i,j;k,l)$, where the indices pair $(k,l;i,j)$ represents the direction of signal from $C(i,j)$ to $C(k,l)$. The cell $C(i,j)$ communicates directly with its neighborhood cells $C(k,l) \in N_r(i,j)$. Since



(a) An n-by-m cellular neural network on rectangular grid (shaded boxes are the neighborhood cells of $C(i,j)$).



(b) Functional block diagram of neuron cell.

Figure 2-3: Cellular neural network (CNN).

the cells $C(k,l)$ have their neighborhood cells, it also communicates with all other cells $C(k,l) \notin N_r(i,j)$ indirectly. Figure 2-3 (a) shows an n-by-m CNN with $r = 1$. The cells filled with dashed lines represent the neighborhood cells $N_1(i,j)$ of $C(i,j)$, including $C(i,j)$ itself.

The block diagram of a cell $C(i,j)$ is shown in Figure 2-3 (b). The external input to the cell is denoted by $v_{uij}(t)$ and typically assumed to be constant $v_{uij}(t) = v_{uij}$ over a operation interval $0 \le t < T$. The input is connected to $N_r(i,j)$ through the feedforward weights $B(i,j;k,l)$'s. The output of the cell, denoted by $v_{yij}(t)$, is coupled to the neighborhood cells $C(k,l) \in N_r(i,j)$ through the feedback weights $A(i,j;k,l)$'s. Therefore, the input signals consist of the weighted sum of feedforward inputs and weighted sum of feedback inputs. In addition, a constant bias term is added to the cell. If the weights represent the transconductance values among the cells, the total input current $i_{xij}(t)$ to the cell is given by

$$i_{xij}(t) = \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{ukl} + I_b, \tag{2.11}$$

where I_b is the bias current. The equivalent circuit diagram of a cell is shown in Figure 2-4, where R_x and C are the equivalent resistance and capacitance of the cell, respectively. For the simplicity, I_b, R_x, and C are assumed to be the same for all cells throughout the network. All inputs are represented by dependent current sources and summed at the state node. Due to the capacitance C and resistance R_x, the state voltage $v_{xij}(t)$ is developed at the summing node and satisfies a set of differential equations

$$\begin{aligned} C\frac{dv_{xij}(t)}{dt} &= -\frac{1}{R_x}v_{xij}(t) + i_{xij}(t) \\ &= -\frac{1}{R_x}v_{xij}(t) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{ukl} + I_b, \\ &\qquad 1 \le i \le n, 1 \le j \le m, \end{aligned} \tag{2.12}$$

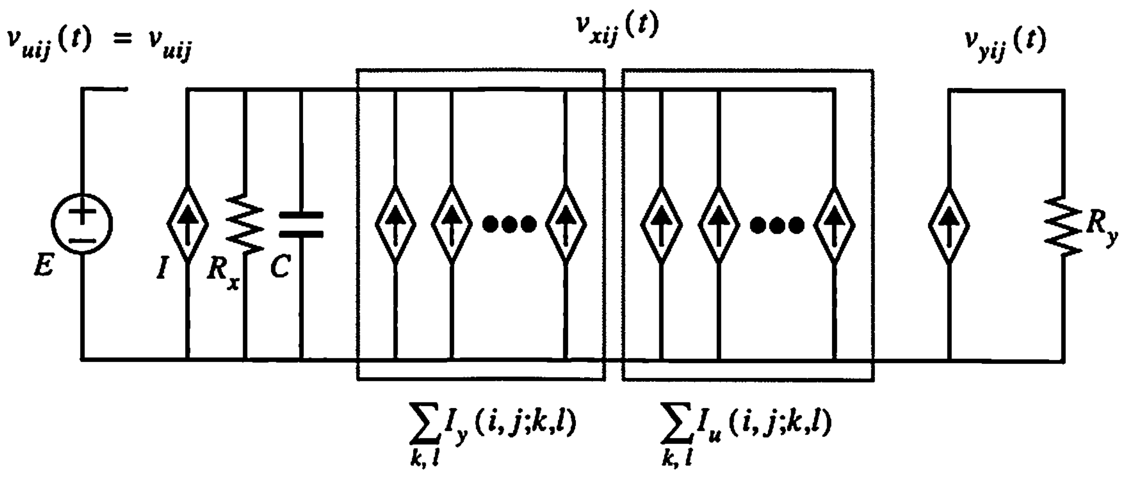
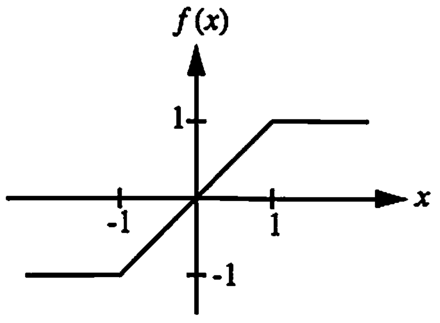


Figure 2-4: Equivalent circuit diagram of cell.



$$f(x) = \frac{1}{2}(|x+1| - |x-1|)$$

Figure 2-5: Piecewise-linear function.

The cell contains a nonlinearity between the state node and the output and its input-output relationship is represented by $v_{yij}(t) = f(v_{xij}(t))$. The nonlinear function used in a CNN can be any differentiable, non-decreasing function $y = f(x)$, provided that $f(0) = 0$, $df(x)/dx \geq 0$, and $f(+\infty) \rightarrow +1$ and $f(-\infty) \rightarrow -1$. Two widely used nonlinearities are the piecewise linear and sigmoid functions as given by

$$y = f(x) = \begin{cases} \frac{1}{2}(|x+1| - |x-1|) & ;\text{piecewise} - \text{linear} \\ \frac{1-e^{-\lambda x}}{1+e^{-\lambda x}} & ;\text{sigmoid.} \end{cases} \tag{2.13}$$

Here, the parameter λ is proportional to the gain of the sigmoid function. For a unity neuron gain at $x = 0$, $\lambda = 2$ may be used for the sigmoid function. The gain of neurons in a Hopfield neural network is very large so that the steady-state outputs are all binary-valued. However, if the positive feedback in the CNN cell is so strong that the feedback factor greater than one, the gain of the cell needs not to be large for guaranteed binary output in the steady state. Typically, a unity gain $df(x)/dx|_{x=0} = 1$ is used in CNNs. The transfer characteristics of the piecewise-linear function is shown in Figure 2-5. The piecewise-linear function provides a mathematical tractability in the analysis, while the sigmoid-like nonlinearity can be easily obtained as a by-product of electronic circuits such as operational amplifier. The shift-invariant CNNs have the interconnections that do not depend on the position of cells in the array except at the edges. The shift-invariant property of CNN is the most desirable feature when implementing a large-size electronic network such as VLSI chip. The weights of a shift-invariant CNN can be represented by the $(2r+1) \times (2r+1)$ feedforward and feedback cloning templates

$$\begin{aligned} \mathbf{T}_A &= \left[a_{p,q}, -r \leq p,q \leq +r\right] \\ \mathbf{T}_B &= \left[b_{p,q}, -r \leq p,q \leq +r\right]. \end{aligned} \tag{2.14}$$

Let $N = n \times m$ be the number of cells in a CNN. By using the vector and matrix notations, (2.12) can be re-written as

$$C\frac{dx}{dt} = -\frac{1}{R_x}x + \mathbf{A}y + \mathbf{B}u + I_b w, \tag{2.15}$$

where

$$\begin{array}{ll}
x = [x_1 \ x_2 \ \cdots \ x_N]^T = [v_{x1}(t) \vdots v_{x2}(t) \vdots \cdots \vdots v_{xn}(t)]^T, & ; N\text{-by-}1 \\
y = [y_1 \ y_2 \ \cdots \ y_N]^T = [v_{y1}(t) \vdots v_{y2}(t) \vdots \cdots \vdots v_{yn}(t)]^T, & ; N\text{-by-}1 \\
u = [u_1 \ u_2 \ \cdots \ u_N]^T = [v_{u1} \vdots v_{u2} \vdots \cdots \vdots v_{un}]^T, & ; N\text{-by-}1 \\
\mathbf{A} = toeplitz((\mathbf{A}_0 \vdots \mathbf{A}_1 \vdots \cdots \vdots \mathbf{A}_r \vdots 0 \vdots \cdots), (\mathbf{A}_0 \vdots \mathbf{A}_{-1} \vdots \cdots \vdots \mathbf{A}_{-r} \vdots 0 \vdots \cdots)), & ; N\text{-by-}N \\
\mathbf{B} = toeplitz((\mathbf{B}_0 \vdots \mathbf{B}_1 \vdots \cdots \vdots \mathbf{B}_r \vdots 0 \vdots \cdots), (\mathbf{B}_0 \vdots \mathbf{B}_{-1} \vdots \cdots \vdots \mathbf{B}_{-r} \vdots 0 \vdots \cdots)), & ; N\text{-by-}N \\
w = [1 \ 1 \ \cdots \ 1]^T. & ; N\text{-by-}1
\end{array}$$

Here,

$$\begin{array}{ll}
v_{xk}(t) = [v_{xk1}(t) \ v_{xk2}(t) \ \cdots \ v_{xkm}(t)], & ; 1\text{-by-}m \\
v_{yk}(t) = [v_{yk1}(t) \ v_{yk2}(t) \ \cdots \ v_{ykm}(t)], & ; 1\text{-by-}m \\
v_{uk} = [v_{uk1} \ v_{uk2} \ \cdots \ v_{ukm}], & ; 1\text{-by-}m \\
\mathbf{A}_k = toeplitz((a_{k,0} \ a_{k,1} \ \cdots \ a_{k,r} \ 0 \ \cdots), (a_{k,0} \ a_{k,-1} \ \cdots \ a_{k,-r} \ 0 \ \cdots)), & ; m\text{-by-}m \\
\mathbf{B}_k = toeplitz((b_{k,0} \ b_{k,1} \ \cdots \ b_{k,r} \ 0 \ \cdots), (b_{k,0} \ b_{k,-1} \ \cdots \ b_{k,-r} \ 0 \ \cdots)), & ; m\text{-by-}m
\end{array}$$

and $toeplitz(\mathbf{a}, \mathbf{b})$ is defined as the Toeplitz matrix with **a** in the first row and **b** in the first column. Note that the submatrices $\mathbf{A}_k$ and $\mathbf{B}_k$ are Toeplitz, but **A** and **B** are not. The elements of $\mathbf{T}_A$ and T_x are often normalized to the scale of T_x, e.g., 10^{-3}. The notations of voltages $v_x(t)/v_y(t)$ and the state variables x/y will be used interchangeably hereafter. Because $-1 \le y_k \le +1$, $\forall k$, the output variable y is confined within the N-dimensional hypercube so that $y \in D^N = \{y \in R^N : -1 \le y_k \le 1; k = 1, 2, \cdots, N\}$. The cloning templates are called symmetric if $A(i,j;k,l) = A(k,l;i,j)$ and $B(i,j;k,l) = B(k,l;i,j)$. In this case, **A** and **B** are symmetric matrices and the stability of the network is guaranteed as will be shown later on. In fact, the symmetry of **A** is a sufficient condition for stability.

Under the constraint conditions $\left|v_{xij}(0)\right| \le 1$ and $\left|v_{uij}\right| \le 1$, $\forall i,j$, the shift-invariant CNN always produces a stable output in the steady state. Moreover, if $A(i,j;i,j) > 1/R_x$, then the saturated binary outputs are guaranteed.

In any CNN's, all states $v_{xij}(t)$, $\forall t \ge 0$, are bounded and the bound $v_{x,\max}$ can be given by [59]

$$v_{x,\max} = 1 + R_x|I_b| + R_x \max_{\substack{1\le i\le n,\\ 1\le j\le m}} \left(\sum_{C(k,l)\in N_r(i,j)} \left(|A(i,j;k,l)| + |B(i,j;k,l)| \right) \right). \tag{2.16}$$

The terms in (2.16) account for the initial value, bias, feedback, and feedforward interactions, respectively. Therefore, the operating range of the circuits for summing and integration in Figure 2-3 (b) must be at least $-v_{x,\max} \le v_{xij}(t) \le v_{x,\max}$.

2.2.2 Stability

The stability of a nonlinear dynamic system including CNNs is described by the Lyapunov [5] or generalized energy function. For the CNN with the piecewise-linear function, it is given by [59]

$$\begin{aligned} E(t) = &-\frac{1}{2}\sum_{i,j} \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)v_{yij}(t)v_{ykl}(t) + \frac{1}{2R_x}\sum_{i,j}\left(v_{yij}(t)\right)^2 \\ &-\sum_{i,j} \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{yij}(t)v_{ukl} - \sum_{i,j} I\, v_{yij}(t). \end{aligned} \tag{2.17}$$

For the sigmoid nonlinearity, the second term of (2.17) is replaced by

$$\frac{1}{R_x}\sum_{i,j} \int_0^{v_{yij}(t)} f_{ij}^{-1}(v)dv. \tag{2.18}$$

The expression (2.18) can be used for arbitrary nonlinearity $y = f(x)$ if its inverse function $x = f^{-1}(y)$ can be well-defined over the range of x. It can be interpreted as the area of the function $x = f^{-1}(y)$ when integrated from $y = 0$ to $y = v_{yij} \le 1$. The piecewise-linear function used in (2.17) is a special case of this general expression (2.18). For the piecewise-linear function, $x = f^{-1}(y) = y$, $-1 \le y \le 1$, and

$$\int_0^{v_{yij}(t)} f^{-1}(y)dy = \int_0^{v_{yij}(t)} y dy = \frac{1}{2}\left(v_{yij}(t)\right)^2, \tag{2.19}$$

which is consistent with the one in (2.17). In the vector and matrix forms, (2.17) is a scalar-valued quadratic function of output vector y,

$$\begin{aligned} E &= -\frac{1}{2}y^T\mathbf{A}y + \frac{1}{2R_x}y^Ty - y^T\mathbf{B}u - Iy^T\mathbf{w} \\ &\equiv -\frac{1}{2}y^T\mathbf{M}y - y^T\mathbf{b}, \end{aligned} \tag{2.20}$$

where $\mathbf{M} = \mathbf{A} - (1/R_x)\mathbf{I}$ and $\mathbf{b} = \mathbf{B}u + I_b\mathbf{w}$. The stability of the network can be tested by checking the behavior of the energy function after the network is activated at time $t = t_0$. By using the chain rule, the time derivative of E can be given by a scalar product of two vectors

$$\frac{dE}{dt} = \frac{\partial E}{\partial y}\cdot\frac{dy}{dt} = \sum_{k=1}^{N}\frac{\partial E}{\partial y_k}\frac{dy_k}{dt}. \tag{2.21}$$

where $\partial E/\partial y = \nabla_y E$ is the gradient of E with respect to y. From (2.20), we have

$$\nabla_y E = \frac{\partial}{\partial y}\left(-\frac{1}{2}y^T\mathbf{M}y - y^T\mathbf{b}\right) = -\frac{1}{2}\left(\mathbf{M}y + \mathbf{M}^Ty\right) - \mathbf{b}. \tag{2.22}$$

If $\mathbf{A}$ is symmetric, so is $\mathbf{M}$ and $\mathbf{M} = \mathbf{M}^T$. Therefore,

$$\nabla_y E = -(\mathbf{M}y + \mathbf{b}) = -\left(\mathbf{A}y - \frac{1}{R_x}y + \mathbf{b}\right). \tag{2.23}$$

Assume that the network is activated at $t = t_0$ and the constraint condition $\left|v_{xij}(t_0)\right| \le 1$ is satisfied. Then, it begins to operate in the linear region because $v_{yij}(t_0) = v_{xij}(t_0)$ and as time increases, some of the cells become saturated such that $|y_l| = 1$ for some l. Note that if $|x_l| > 1$, then $dy_l/dt = 0$ and the corresponding terms in (2.20) vanish. If we consider only nonzero terms, then for $k \neq l$, $y_k = x_k$, $dy_k/dt = dx_k/dt$, and from (2.12) and (2.23) $\partial E/\partial y_k = -C(dx_k/dt)$. Therefore,

$$\frac{dE}{dt} = -C\sum_{k\neq l}\frac{dx_k}{dt}\frac{dy_k}{dt} = -C\sum_{k\neq l}\left(\frac{dx_k}{dt}\right)^2. \tag{2.24}$$

Since $C > 0$, the energy E decreases as time elapses such that $dE/dt \leq 0$, $\forall t \geq t_0$. When all the cells become saturated, $dE/dt = 0$ and the network results in a stable binary output for which the energy function (2.20) is locally minimized. Note that the state x is stabilized after the stable equilibrium is reached through $dE/dt = 0$. If we use other neuron transfer characteristics $y = f(x)$, for which the inverse function $x = f^{-1}(y)$ is well defined in the range of x, (2.20) can be written as

$$E = -\frac{1}{2}y^T\mathrm{A}y + \frac{1}{R_x}\int_0^y f^{-1}(v)dv - y^T\mathbf{b}, \tag{2.25}$$

where the second term is simply an N-by-1 vector with the integral expression in each element. In this case, $f^{-1}(y) = x$ and

$$\nabla_y E = -\mathbf{A}y + \frac{1}{R_x}f^{-1}(y) - \mathbf{b} = -\mathbf{A}y + \frac{1}{R_x}x - \mathbf{b} = -C\frac{dx}{dt}, \tag{2.26}$$

from which it follows that

$$\frac{dE}{dt} = \nabla_y E\cdot\frac{dy}{dt} = -C\frac{dx}{dt}\cdot\frac{dy}{dt} = -C\sum_{k=1}^{N}\frac{\partial f}{\partial x_k}\left(\frac{dx_k}{dt}\right)^2 \leq 0. \tag{2.27}$$

The information to be processed can be passed into the network as a form of the input v_{uij} or the initial state $v_{xij}(0)$, or both of them. In any cases, the initialization of the state voltage $v_{xij}(t)$ is required at the beginning of each operation, such that $|v_{xij}(0)| \leq 1$, $\forall i, j$. Otherwise, the undesirable situation $E(t = 0) < E(t = +\infty)$ may occur. Local interconnection and simple synaptic weights are the most attractive features of the CNN for VLSI implementation in high-speed, real-time applications.

2.2.3 Discrete-time CNNs

Discrete-time cellular neural networks (DTCNNs) are special kind of feedback threshold network where the local interconnections and the shift-invariant weights are transferred from continuous-time CNNs. They are completely describes by a recursive algorithm. The dynamic behavior is based on the feedback of clocked, binary outputs and a single cell is influenced by the inputs and outputs of neighboring cells. The architecture is closely related to cellular automata, but differs from them in having continuous-valued inputs and weights.

The DTCNN is the discrete-time version of (2.12) and defined by the state equation

$$x_{ij}(k) = \sum_{k,l} A(i,j;k,l) y_{kl}(k) + \sum_{k,l} B(i,j;k,l) u_{kl} + I_b, \tag{2.28}$$

and the output equation

$$y_{ij}(k) = sign\left(x_{ij}(k-1)\right) = \begin{cases} -1 & \text{if } x_{ij}(k-1) < 0 \\ +1 & \text{if } x_{ij}(k-1) \geq 0 \end{cases} \tag{2.29}$$

for a cell $C(i,j)$, $i = 1,2,\cdots,n$, $j = 1,2\cdots,m$, in an $n \times m$ rectangular-grid array. Here, $y_{ij}(k) \in \{-1,+1\}$, $u_{ij} \in D$, $A(i,j;k,l) \in R$, and $B(i,j;k,l) \in R$, where D is defined as $D = \{d \in R: -1 \leq d \leq +1\}$. The advantage of DTCNNs is the description of the next output through a set of linear inequalities in discrete-time fashion. The fact that it does not include a sophisticated integration algorithm, allows simple implementation of the algorithm on general-purpose digital computer. In a hardware viewpoint, the operation is quite insensitive to noise and parameter variations caused by fabrication tolerances and environmental effects. Thus, the interconnections among cells in a network or among

VLSI chips in a large scale, multi-chip system, can be simplified. In addition, because the discrete-time operation is controlled by clock signal, a robust control over the propagation speed and testability is provided.

The energy function of a DTCNN can be defined by the use of the Lyapunov theory for discrete-time systems [6,64]

$$\begin{aligned} E(k) = &-\sum_{i,j}\sum_{k,l} A(i,j;k,l) y_{kl}(k-1) y_{ij}(k) \\ &-\sum_{i,j}\sum_{k,l} B(i,j;k,l) y_{kl}(k)\big(y_{ij}(k)+y_{ij}(k-1)\big) \\ &-\sum_{i,j} I_b\big(y_{ij}(k)+y_{ij}(k-1)\big). \end{aligned} \tag{2.30}$$

Assuming the symmetric feedback weights $A(i,j;k,l) = A(k,l;i,j)$, the differential energy $\Delta E \equiv E(k+1) - E(k)$ is given by

$$\begin{aligned} \Delta E &= -\sum_{i,j}\big(y_{ij}(k+1) - y_{ij}(k-1)\big)\left(\sum_{k,l} A(i,j;k,l) y_{kl}(k) + \sum_{k,l} B(i,j;k,l) u_{kl} + I_b\right) \\ &= -\sum_{i,j}\big(y_{ij}(k+1) - y_{ij}(k-1)\big) x_{ij}(k). \end{aligned} \tag{2.31}$$

Rewriting (2.29) as $y_{ij}(k+1) = x_{ij}(k) / |x_{ij}(k)|$ and substituting it into (2.31), we can get

$$\Delta E = -\sum_{i,j} |x_{ij}(k)| \big(y_{ij}(k+1)^2 - y_{ij}(k-1) y_{ij}(k+1)\big). \tag{2.32}$$

Therefore, $\Delta E = 0$ if $y_{ij}(k-1) = y_{ij}(k+1)$, $\forall i,j$, and $\Delta E < 0$ otherwise. The energy function E decreases as k increases and the condition $\Delta E = 0$ for a stable state can be reached. However, for the condition $\Delta E = 0$ there exist two possible cases, i.e., for all i and j, $y_{ij}(k) = y_{ij}(k-1)$, and $y_{ij}(k) \neq y_{ij}(k-1)$ but $y_{ij}(k+1) = y_{ij}(k-1)$. The first case obviously corresponds to a stable state, while the second case represents a two-cycle oscillation between two different outputs. Thus, the stable operation is not guaranteed in a DTCNN with symmetric feedback templates. For some other class of templates, the DTCNN is shown to be always stable [64].

2.3 Paralleled, Hardware-based Annealing

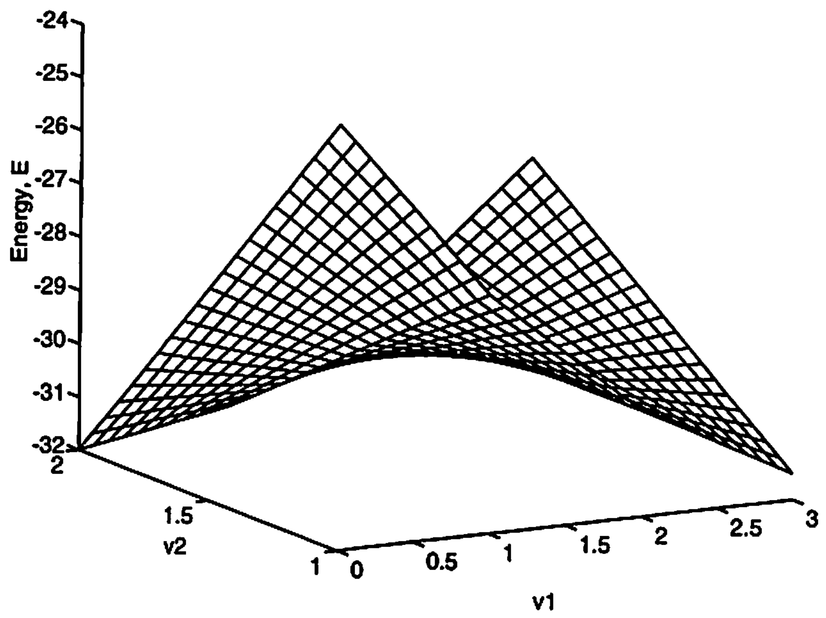
The Boltzmann machine or MFA described in the previous chapter is an iterative algorithm for solving a set of stochastic or deterministic equations. On the other hand, when the problem to be optimized is appropriately mapped onto the energy function, the Hopfield neural network or CNN can be by itself a tool for solving the problem. The Hopfield network was originated from the network of probabilistic threshold neurons and the circuit theory, while the CNN is a continuous-valued variation of cellular automata with similar collective computational properties to those of the Hopfield network [2]. They are both described by the model of electronic circuits in a nonlinear system, i.e., differential equation (1.8) by capacitor and resistors and the energy function (1.9) by the Lyapunov theory [5,6]. These networks can be realized by physical hardware circuits for very fast, real-time processing of real-world data or signals. Therefore, for the global optimization of the energy function, these neural networks need to be equipped with means of avoiding local minima problems.

The hardware-based annealing technique has an analogy to the metallurgical annealing in the metallurgy and simulated annealing in the Boltzmann machine, which are the optimal stochastic procedures. It is a paralleled, electronic version of the deterministic mean-field learning rule [43-46] directly incorporated with the Hopfield neural network or CNN. However, as will be shown in the later, it can be generalized as a hardware-based optimization technique in which the globally optimal solution is obtained through the process of changing the network energy from a convex function to an indefinite or concave function. In contrast to the simulated annealing or other optimization techniques on serial or parallel digital computer, it can be easily implemented in a massively-paralleled hardware for real-time applications. For example, in solving the global minimum of a 12-neuron, binary Hopfield neural network using the circuit partitioning technique [19], it took about 5 minutes on a SUN-SPARC machine. In many applications of the simulated annealing, the major obstacle has been shown to be the slow convergence speed toward a near-optimal solution [23,40,46]. On the other hand, the speed of microseconds or fractional microseconds can be achieved in VLSI neural networks [47-51].

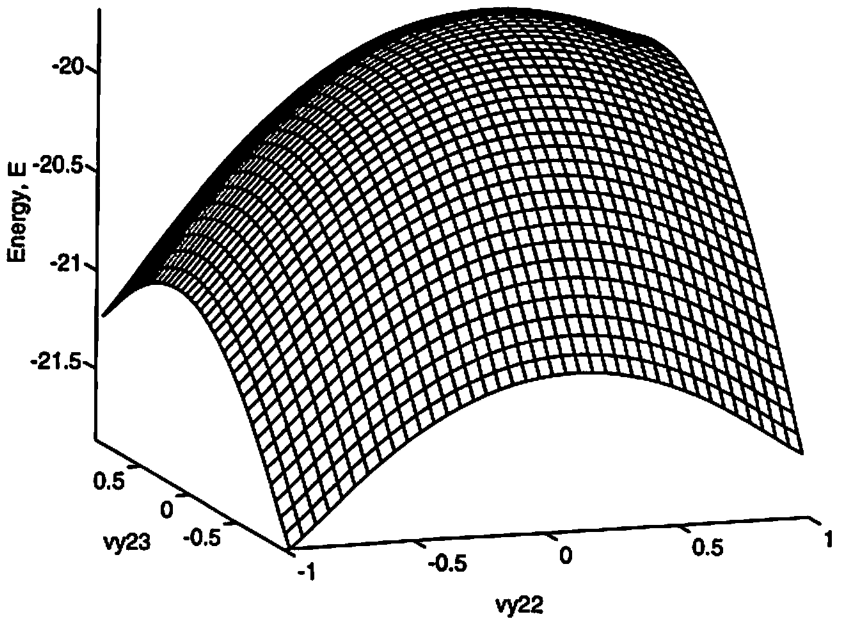
2.3.1 Local Minima

Let us examine the local minima in the energy function of a Hopfield neural network and CNN. Typical energy functions of a Hopfield neural network and a CNN are shown in Figure 2-6 (a) and (b), respectively, for comparison. In the plots, the x- and y-axes represent two outputs of specific neurons in the networks as the free variables and the z-axis indicates the magnitude of the energy E. In a Hopfield neural network, the diagonal elements of the transconductance matrix are zero and due to the infinite neuron gain the energy term that corresponds to the second term in (2.16) vanishes [2,3]. Some eigenvalues are always negative such that the sum of all eigenvalues is equal to zero and therefore the networks have the saddle-shaped energy surfaces [33-35,113] as shown in Figure 2-6 (a). In Figure 2-6 (b), the energy is plotted as the functions of v_{y22} and v_{y23} in a 4×4 CNN with $R_x = 10^3\Omega$, $I = 0$ and the cloning templates

$$\mathbf{T}_A = \begin{bmatrix} 0 & -0.25 & 0 \\ -0.25 & 2 & -0.25 \\ 0 & -0.25 & 0 \end{bmatrix} \text{ and } \mathbf{T}_B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.33}$$



(a) Indefinite energy function of Hopfield neural network.



(b) Concave energy function of CNN.

Figure 2-6: Typical energy landscapes of Hopfield neural network and CNN.

The eigenvalues of M corresponding to these parameters are evaluated and listed in Table 2.1. The matrix M in this case is positive definite and all corners of $[-1,+1]^2$ are the minima as shown in the figure.

Consider a simple two-neuron cellular neural network shown in Figure 2-7 (a). It contains self-feedbacks and cross-interactions. For simplicity, assume that the neuron uses the piecewise-linear function, the capacitance C is normalized to a unity, and interactions are symmetric $A_{1,1} = A_{2,2} = A_0 > T_x = 1/R_x$, $A_{1,2} = A_{2,1} = A_1$. Then, the Lyapunov function is given by

$$\begin{aligned} E &= -\frac{1}{2}\sum_{i=1}^{2}\sum_{j=1}^{2} A_{j,i} v_{yi} v_{yj} + \frac{T_x}{2}\sum_{i=1}^{2}\left(v_{yi}\right)^2 - \sum_{i=1}^{2} v_{yi} v_{ui} \\ &= -\frac{1}{2}\underbrace{\begin{bmatrix} v_{y1} \\ v_{y2} \end{bmatrix}^T}_{y}\underbrace{\begin{bmatrix} A_0 - T_x & A_1 \\ A_1 & A_0 - T_x \end{bmatrix}}_{M}\begin{bmatrix} v_{y1} \\ v_{y2} \end{bmatrix} - \begin{bmatrix} v_{y1} \\ v_{y2} \end{bmatrix}^T \underbrace{\begin{bmatrix} v_{u1} \\ v_{u2} \end{bmatrix}}_{u}. \end{aligned} \quad (2.34)$$

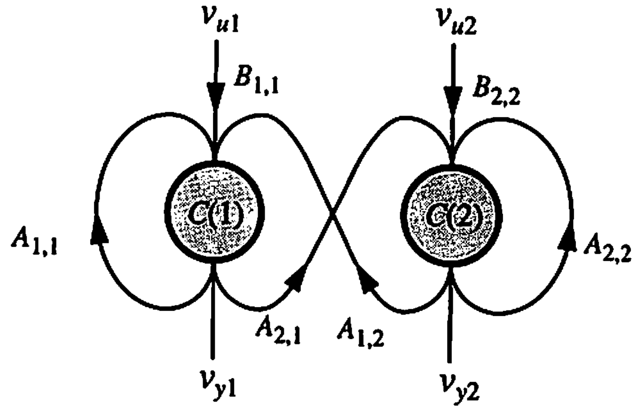
where $-1 \le v_{y1}, v_{y2} \le +1$. By solving the equation $Mx = \lambda x$, the eigenvalues and eigenvectors of M are given by $\lambda_1 = A_0 + A_1 - T_x$, $\lambda_2 = A_0 - A_1 - T_x$ and $x_1 = [+1\ +1]^T$, $x_2 = [+1\ -1]^T$, respectively. Obviously, E has multiple minima at the corners of $[-1,+1]^2$. If the eigenvalues λ are positive and the bias is relatively small, then the minima occur at all corners as given by

$$E = \begin{cases} -(A_0 - T_x + A_1) - (v_{u1} + v_{u2}) & ; v_{y1} = v_{y2} = +1 \\ -(A_0 - T_x - A_1) + (v_{u1} - v_{u2}) & ; v_{y1} = -1, v_{y2} = +1 \\ -(A_0 - T_x - A_1) - (v_{u1} - v_{u2}) & ; v_{y1} = +1, v_{y2} = -1 \\ -(A_0 - T_x + A_1) + (v_{u1} + v_{u2}) & ; v_{y1} = v_{y2} = -1 \end{cases} \quad (2.35)$$

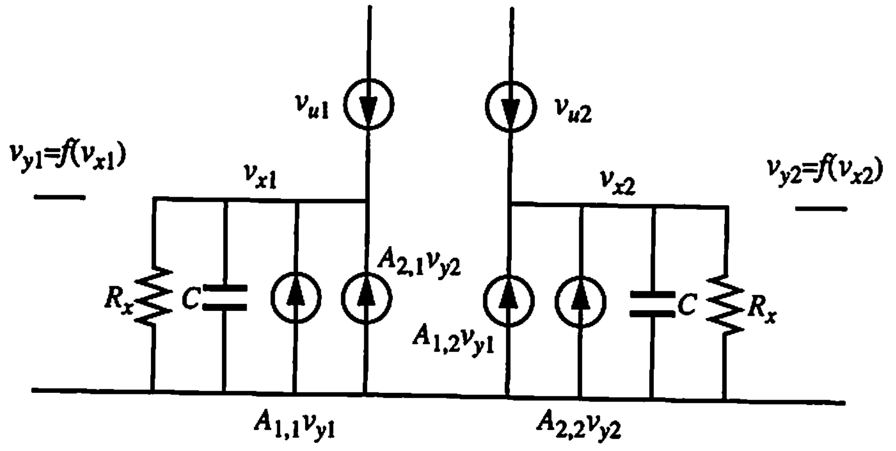
Depending on the values of A_1, v_{u1}, and v_{u2}, each minimum has a different value from others. The lowest value corresponds to the global minimum and others the local minima. Figure 2-8 (a) shows contour plot of the energy function when $A_0 = 2$, $A_1 = -0.5$, and

Table 2.1 List of eigenvalues of positive definite **M** for cloning templates of (2.33).

Eigenvalue	Magnitude	Multiplicity
$\lambda_1 - \lambda_4$	+ 1.000	4
λ_5, λ_6	+ 1.250	2
λ_7, λ_8	+ 0.750	2
λ_9, λ_{10}	+ 0.441	2
$\lambda_{11}, \lambda_{12}$	+ 1.559	2
λ_{13}	+ 0.691	1
λ_{14}	+ 1.309	1
λ_{15}	+ 0.191	1
λ_{16}	+ 1.809	1
$\sum_{k=1}^{16} \lambda_k$	+ 16.00	

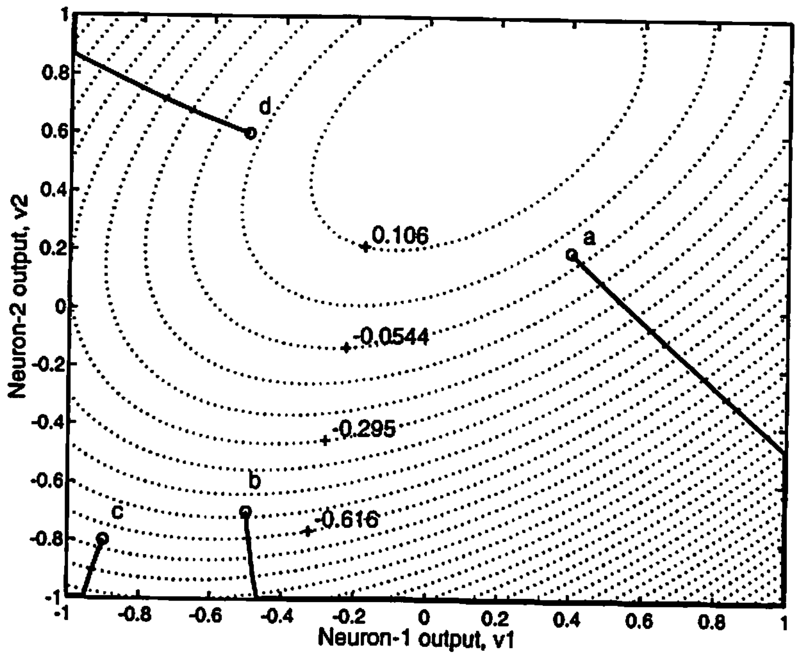


(a) Two-neuron cellular neural network.

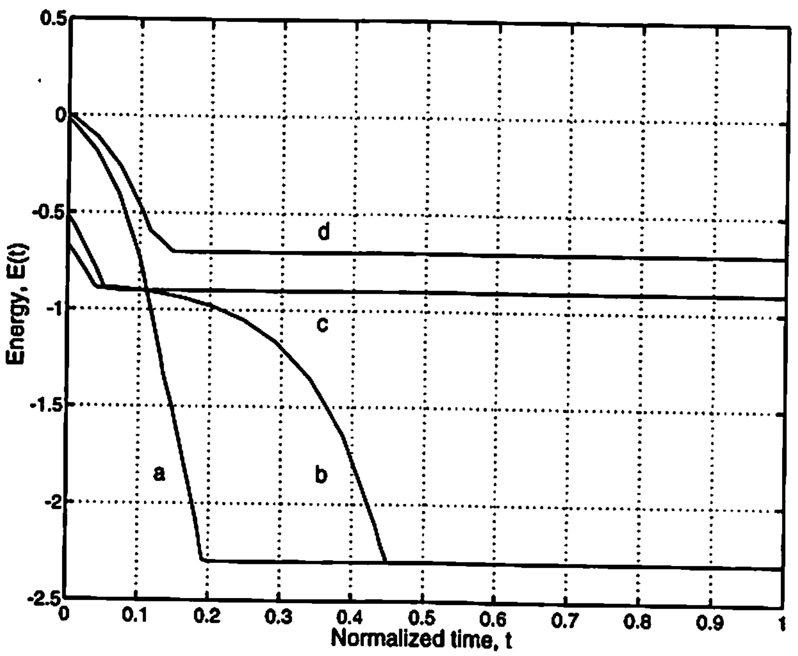


(b) Equivalent circuit diagram.

Figure 2-7: Two-neuron neural network.



(a) Energy contour and trajectories of outputs for several initial conditions.



(b) Corresponding energy functions E(t) during network evolution.

Figure 2-8: Multiple minima in concave energy function of two-neuron CNN.

$[v_{u1}, v_{u2}] = [0.2, -0.5]$. The point $[+1, -1]$ is the global minimum while the points $[-1, -1]$ and $[-1, +1]$ are the local minima. Four initial outputs *a*, *b*, *c*, and *d* follow the corresponding trajectories when the network is allowed to evolve. The steady-state outputs for *a* and *b* are both $[v_{y1}, v_{y2}] = [+1, -1]$ which correspond to the global minimum. The initial locations *c* and *d* result in the local minima in the steady state. The energy levels during the network evolution and steady state for these four cases are shown in Figure 2-8 (b). Note that two local minima happen to have the same energy level. The energy landscape E is determined by the interactions and input. However, there is no direct correspondence between the initial value and steady-state output in a given network.

2.3.2 Global Optimization

Now consider a CNN with $n \times m = N$ neurons. The neuron have the piecewise-linear transfer function and its gain is variable as shown in Figure 2-9. The gain is controlled by a monotonically increasing function $g(t)$ such that

$$v_y = f(gv_x) = \begin{cases} +1 & ; v_x > +1/g \\ gv_x & ; -1/g \le v_x \le +1/g \\ -1 & ; v_x < -1/g \end{cases} \tag{2.36}$$

In this case, the energy function (2.17) can be written as

$$\begin{aligned} E &= -\frac{1}{2}\sum_{i,j}\sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l) v_{yij} v_{ykl} + \frac{1}{2gR_x}\sum_{i,j}(v_{yij})^2 \\ &\quad -\sum_{i,j}\sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l) v_{yij} v_{ukl} - \sum_{i,j} I_b v_{yij}. \\ &= -\frac{1}{2} y^T \left(\mathrm{A} - \frac{T_x}{g}\mathrm{I} \right) y - y^T b \\ &= -\frac{1}{2} y^T \mathbf{M}_g y - y^T b, \end{aligned} \tag{2.37}$$

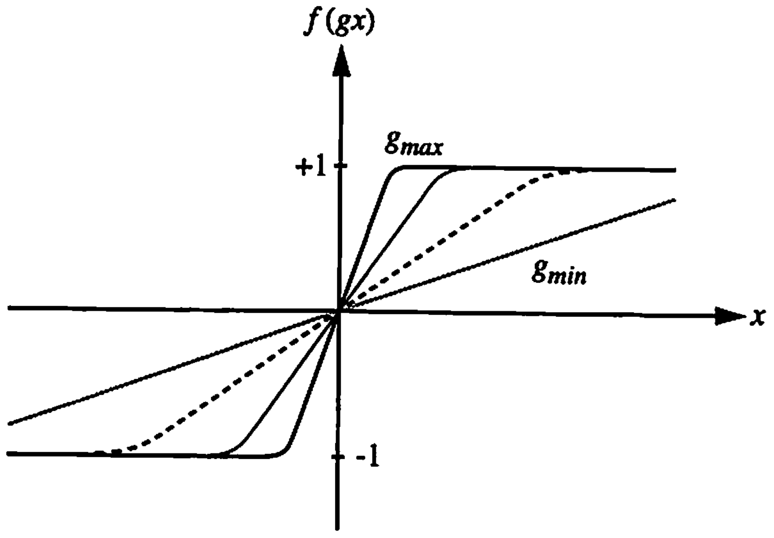


Figure 2-9: Variable-gain piecewise-linear function for hardware annealing.

where the factor $1/g$ in the second term stems from the energy associated with the piecewise-linear function with a neuron gain other than unity. When $\mathbf{M}_g$ is diagonalized as $\mathbf{M}_g = \mathbf{Q}\Lambda_g\mathbf{Q}^T$, the matrix $\mathbf{Q}$ consisting of orthonormal set of eigenvectors e_k's is independent of neuron gain because $\mathbf{M}$ and $\mathbf{M}_g$ commute. As the annealing gain g increases from a small initial value $0 < g_0 << 1$, the eigenvalues of the time-varying matrix $\mathbf{M}_g \equiv \mathbf{A} - (T_x/g)\mathbf{I}$ change from negative to positive values and the energy function E given in (2.37) gradually changes from a convex function to concave function.

Let λ_k and λ_k^1 denote the k-th eigenvalue of $\mathbf{M}_g$ and $\mathbf{M}$, respectively. Also let $y = \beta_1 e_1 + \cdots + \beta_N e_N$, where β_k, $1 \le k \le N$, are scalar constants. Then, $\beta = \mathbf{Q}^T y = \sum_k \mathbf{Q}^T(\beta_k e_k) = [\beta_1 \quad \beta_2 \quad \cdots \quad \beta_N]^T$ and (2.37) can be rewritten as

$$E = -\sum_{k=1}^{N}\left(\frac{1}{2}\beta_k^2\lambda_k + \beta_k e_k^T b\right) = \sum_{k=1}^{N} E_k. \tag{2.38}$$

From (2.38), it can be seen that the network attains its equilibrium y_0 when $\beta_k = -e_k^T b/\lambda_k \equiv \beta_{0,k}$, $\forall k$. It also can be found by solving

$$\frac{\partial E}{\partial y} = -\left(\mathbf{A} - \frac{T_x}{g}\mathbf{I}\right)y - b = -\mathbf{M}_g y - b = 0, \tag{2.39}$$

where the symmetric property of $\mathbf{M}_g$ is used in the partial differentiation. Thus we have the same result

$$y_0 = -\mathbf{M}_g^{-1} b = -\sum_{k=1}^{N}\left(\frac{e_k^T b}{\lambda_k}\right)e_k = -\sum_{k=1}^{N}\beta_{0,k}e_k. \tag{2.40}$$

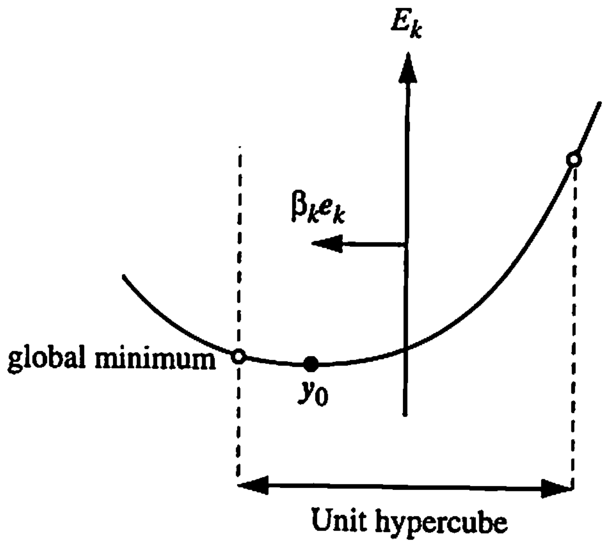
The global minimization of (2.20) can be explained by observing the trajectory of the equilibrium y_0 during the process of the proposed hardware annealing. The energy minimum for $g = 1$ can be obtained by simultaneously minimizing quadratic energy terms E_k's in (2.38) and should occur at a vertex of the hypercube $[-1,+1]^N$ where the following conditions are met

$$\operatorname{sgn}(\beta_k) = \begin{cases} -\operatorname{sgn}\left(-e_k^T b/\lambda_k^1\right) & ;\lambda_k^1 > 0 \\ +\operatorname{sgn}\left(-e_k^T b/\lambda_k^1\right) & ;\lambda_k^1 < 0 \end{cases} \tag{2.41}$$

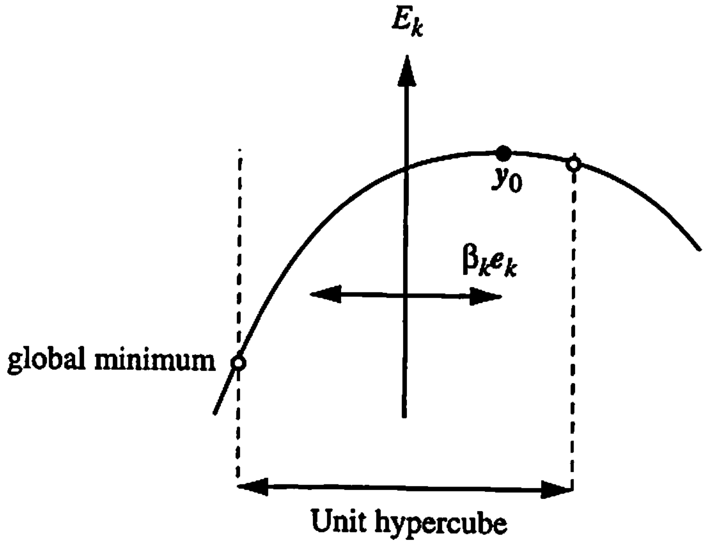
from the symmetric property of E_k as a function of β_k. Figure 2-10 graphically shows the condition (2.41), the relationship between the output for the global minimum and the equilibrium y_0 at $g=1$ along the eigenvector e_k. Of course, the conditions (2.41) just indicate the direction of the global minimum around the origin, but neither an absolute location in a whole space R^N nor the direction of the movement as g increases.

For the gain increase from g_0 to a fixed value g_C at which $\lambda_1 = 0$, all eigenvalues are negative. During this period, we may observe the following facts: 1) $\mathbf{M}_g$ is negative definite (i.e., $-\mathbf{M}_g$ is positive definite), and y_0 is a unique stable equilibrium so that $y \cong y_0$; and 2) because the magnitude of $-e_k^T b/\lambda_k$ in (2.40) increases as g increases, y_0 moves from $y_0(0) \cong 0$ toward the boundary of D^N while satisfying the condition (2.41). In other words, the annealing process forces the network such that: 1) the output stays in a stable equilibrium condition, i.e., $y \cong y_0$, and 2) the output moves toward the global minimum energy state. A neuron output is bounded by a value between -1 and +1, some of neuron outputs become saturated and the linear operation described above does not hold. However, those neurons with saturated outputs have fixed energy values in (2.20), they no longer contribute to the energy minimization operation and the process continues with a reduced dimensionality. As $\lambda_1 > 0$, E becomes an indefinite function and energy barriers begin to stand out, preventing the network in the globally minimum energy state from moving back to local minima.

The critical neuron gain g_C similar to the critical temperature [11,46] in the mean-field annealing, may be defined as the neuron gain at which $\lambda_1 = 0$. By noting that $\mathbf{M}_g = \mathbf{A} - (T_x/g)\mathbf{I} = \mathbf{M} + \left((g-1)T_x/g\right)\mathbf{I}$, it is given by



(a) Negative eigenvalye $\lambda_k^1 < 0$.



(b) Positive eigenvalue $\lambda_k^1 > 0$.

Figure 2-10: Directional relationship between equilibrium y_0 and global minimum in one-dimensional phase space of E_k.

$$g_C = \frac{T_x}{\lambda_1^1 + T_x}. \tag{2.42}$$

where λ_1^1 is the maximum eigenvalue of $\mathbf{M} = \mathbf{M}_g(g=1)$. The critical gain (2.42) and another important gain value will be further investigated in Chapter IV.

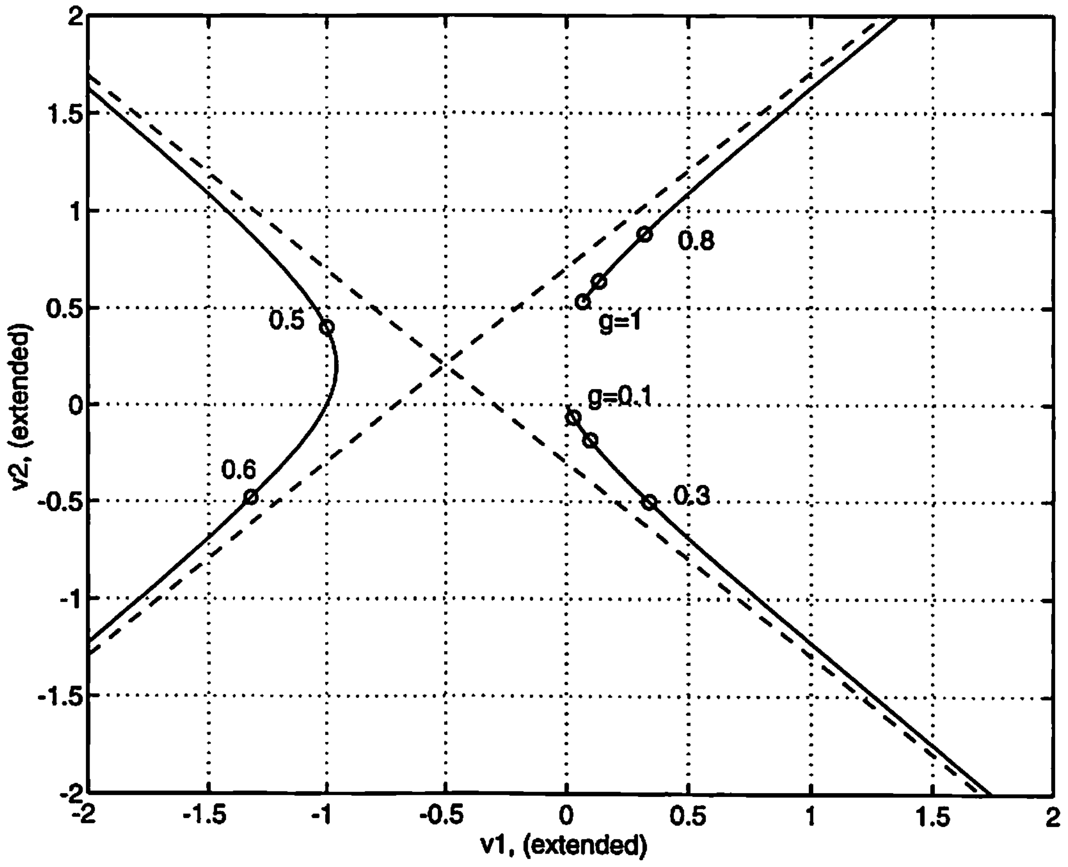
The process of global optimization by hardware annealing is discussed heuristically. Now let us apply the proposed method to two-neuron CNN described early in this section.

$$\begin{aligned} E &= -\frac{1}{2}\begin{bmatrix} v_{y1} \\ v_{y2} \end{bmatrix}^T \begin{bmatrix} A_0 - T_x/g & A_1 \\ A_1 & A_0 - T_x/g \end{bmatrix} \begin{bmatrix} v_{y1} \\ v_{y2} \end{bmatrix} - \begin{bmatrix} v_{y1} \\ v_{y2} \end{bmatrix}^T \begin{bmatrix} v_{u1} \\ v_{u2} \end{bmatrix} \\ &= -\frac{1}{2} y^T \mathbf{M}_g y - y^T u. \end{aligned} \tag{2.43}$$

Until a saturation occurs in one or both neurons, the network is a linear time-varying system that can be described by the equivalent circuit of Figure 2-7 (b). By diagonalizing the matrix $\mathbf{M}_g$ as $\mathbf{M}_g = \mathbf{Q}\Lambda\mathbf{Q}^T$ where $\Lambda = diag(\lambda_1, \lambda_2)$ and $\mathbf{Q} = [e_1 \mid e_2]$ is the corresponding eigenvector matrix, the eigenvalues are $\lambda = A_0 \pm A_1 - T_x/g$. The equilibrium is given by

$$y_0 = -\mathbf{M}_g^{-1} v_u = -\frac{e_1^T v_u}{\lambda_1} e_1 - \frac{e_2^T v_u}{\lambda_2} e_2, \quad \lambda_1, \lambda_2 \neq 0. \tag{2.44}$$

As λ_1 approaches zero, the first term of (2.44) dominates and the direction of y_0 approaches that of the eigenvector e_1. Note that the desired output $y = (+1, -1)^T$ is toward the direction of e_1, the eigenvector that corresponds to the largest eigenvalue. As g further increases, y_0 travels the directions of $-e_1$, $-e_2$, and finally e_2. In Figure 2-11, the trajectory of y_0 and the eigenvectors e_1 and e_2 shifted by u are shown in solid and dotted lines, respectively, when g varies from 0 to 1. Figure 2-12 shows plots of the energy landscape (2.43) for increasing neuron gain values. The energy function is convex, indefinite, and concave when the gain is in the range $0 < g < 0.4$, $0.4 < g < 2/3$,



(dashed lines: directions of eigenvectors e1 & e2, u=[-0.5,0.2])

Figure 2-11: Trajectory of equilibrium point for increasing annealing gain. (g=0-0.4: stable eq., g=0.4-2/3: saddle point, g=2/3-1: unstable eq.)

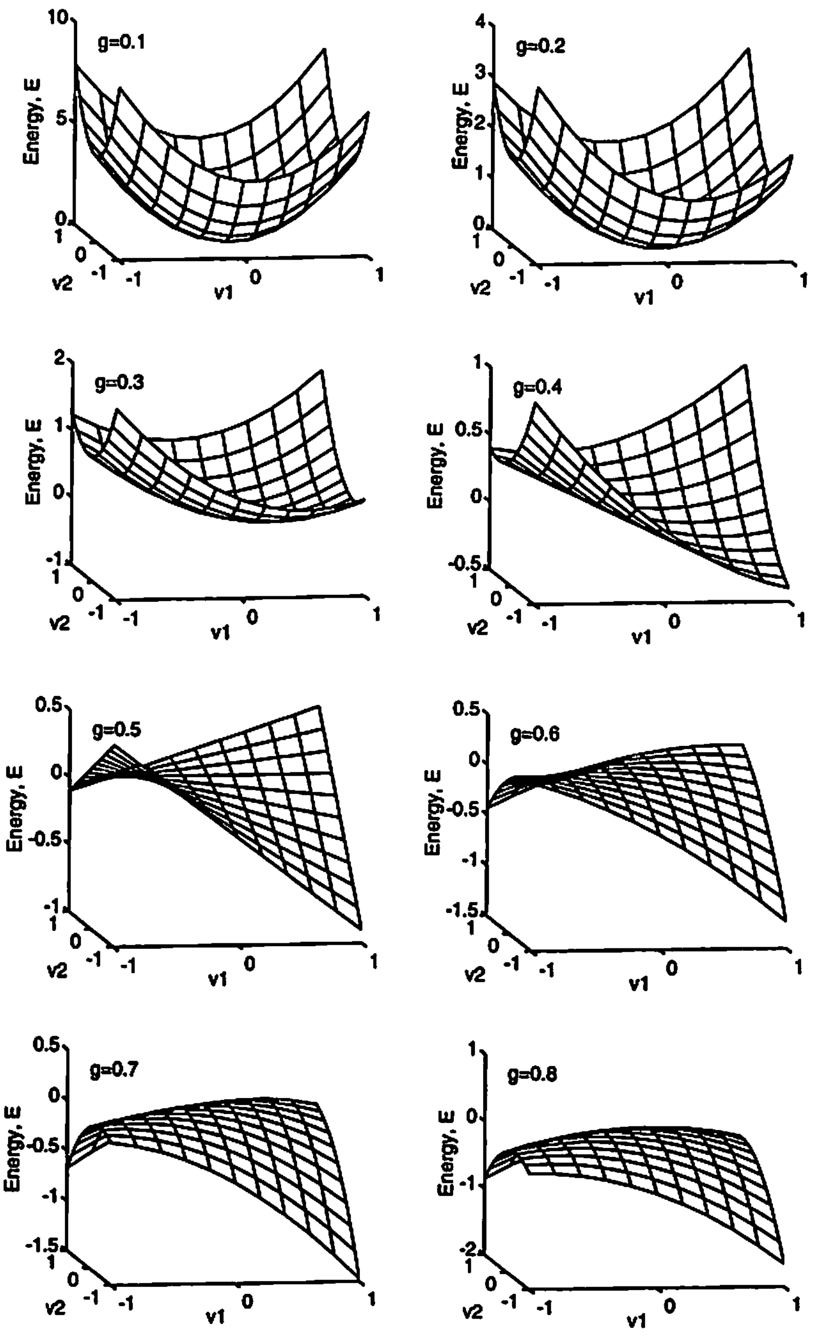


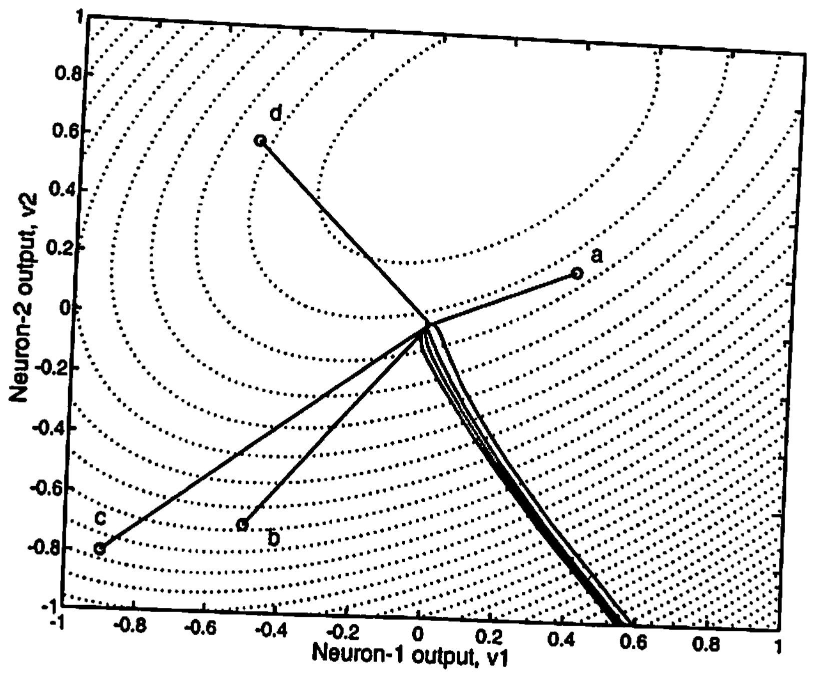
Figure 2-12: Dynamic changes of energy landscape for increasing neuron gain from g=0.1 to g=0.8.

and $2/3 < g \leq 1$, respectively. From the figure, it can be seen that the global minimum is reached during the gain increase from 0 to 0.4 with $g_C = 0.4$. In Figure 2-13, the trajectories of the outputs are plotted when the proposed annealing is applied to the network conditions of Figure 2-8. Regardless of initial state values, the network results in the optimal solution at which its energy is minimized globally.

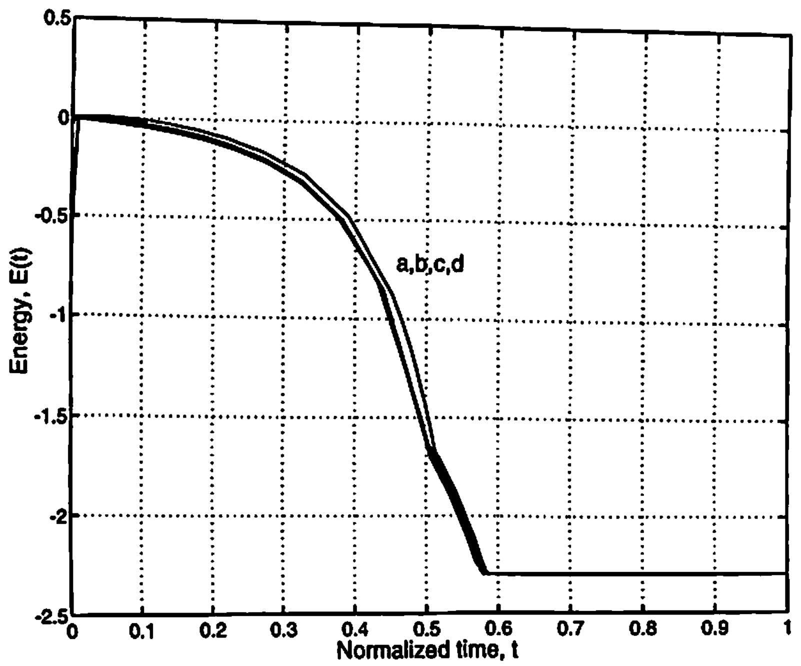
2.3.3 Ill-conditioned Initial Conditions

As indicated in the section 2.2.1, the constraint conditions on the initial state and input values must be satisfied for the stable operation of the network. The steady-state values of states is $1 < v_{xij}(t = +\infty) \leq v_{x,max}$, $\forall i, j$, as can be seen from (2.16). Therefore, the state values must be initialized to a value $-1 \leq v_{xij}(0) \leq +1$ at the beginning of each operation. Figure 2-14 (a) shows two initial conditions in the two-neuron neural network with and without satisfying the constraint conditions. For $[v_{x1}(0), v_{x2}(0)] = [-1.5, +1.2]$, the output never changes and frozen to the initial value $[v_{y1}(t), v_{y2}(t)] = [-1, +1]$, $\forall t \geq 0$. This is the case if there is no stable equilibrium with a lower energy value than $E(t = 0)$. Even when the constraint conditions are satisfied, external excitations may not be so strong as to compete with the initial state [59].

The hardware annealing is shown to be an efficient way of overcoming such ill-conditioned initial states. It first forces the output to move to around the origin and then lets the network evolve according to the dynamics described in the previous section. Figure 2-14 (b) shows the correct operation of the annealed network in finding the global minimum for both initial conditions used in Figure 2-14 (a). If the initial states are not used as means of supplying the external input, they can be left uninitialized in each operation of an annealed network.

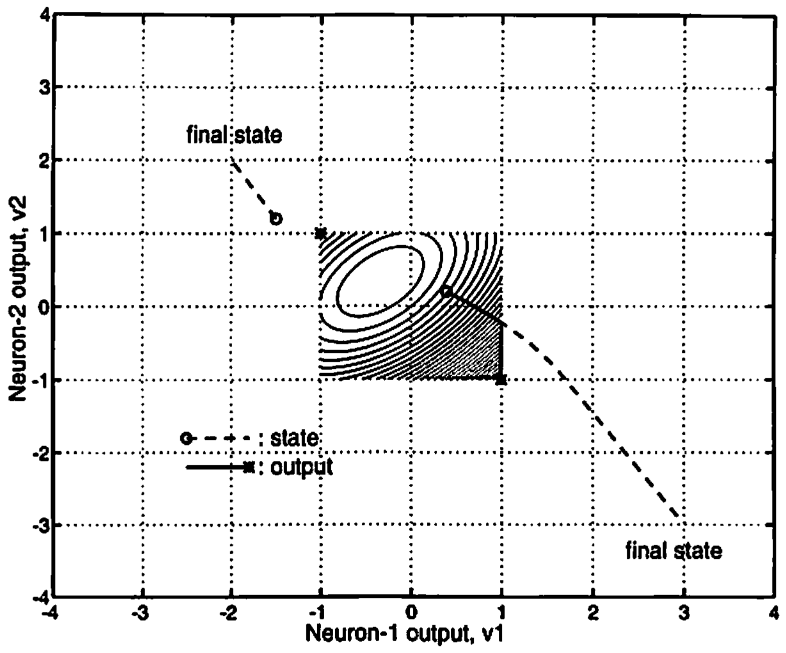


(a) Trajectories of outputs during annealed operation.

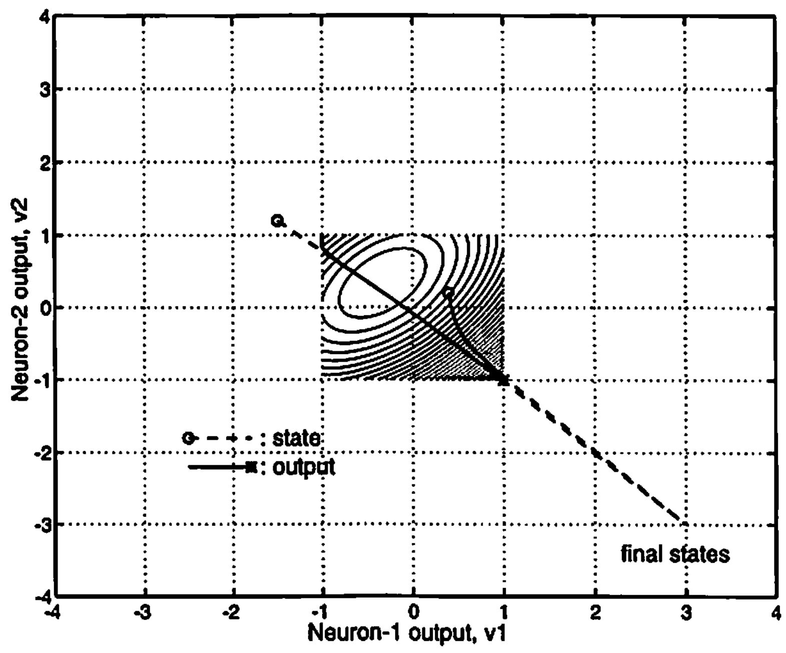


(b) Corresponding energy functions E(t).

Figure 2-13: Global optimization: Annealed network operations for conditions of Figure 2-8.



(a) Ill-conditioned initial state can freeze neuron outputs.



(b) Trajectories of states during annealed operation.

Figure 2-14: Ill-conditioned initial states and hardware annealing.

Chapter III

Paralleled Hardware-based Annealing in Hopfield Neural Networks

3.1 Introduction

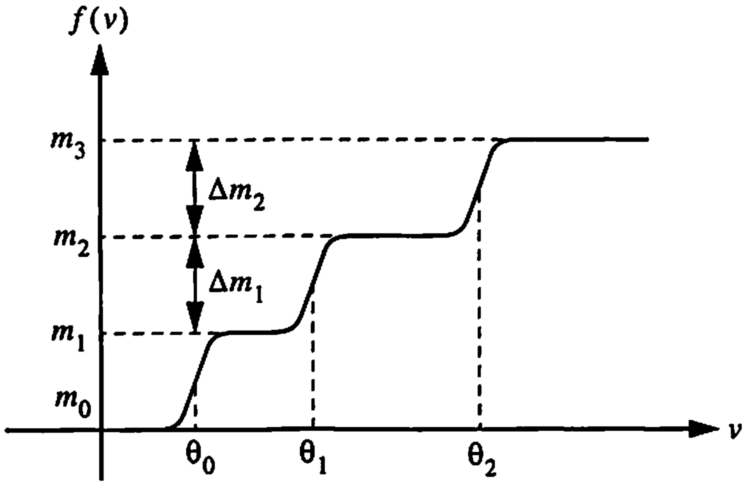
Search of fast optimization in scientific and engineering applications has drawn researchers' interest in the field of engineering neural networks. Simulated annealing [37-40] is an important method for searching for the optimal results on digital computers. It is a stochastic process modeled after the metallurgical annealing in which the slow decrease of the cooling temperature and the natural tendency towards a minimum energy state are the key factors. Due to a slow cooling schedule in software execution, the simulated annealing method on digital computers requires a lot of computing time for a complex optimization problem. By constructing a hardware-based parallel annealing technique in analog electronics, the processing speed can be significantly improved.

An analog-to-digital (A/D) converter based on the Hopfield neural network [4,34,35] is a computational network for obtaining a digital representation of analog input by minimizing the squared error between them. The properties of the Hopfield network together with the mapping of the cost function onto the energy function were presented as an illustrative example of optimization problem. The result, however, is somewhat discouraging because some of digital outputs are sub-optimal in representing the input values, due to local minima problem. The first part of this chapter is devoted to the hardware annealing technique for avoiding such minima in the A/D conversion neural network with multi-level neurons, which is an extension of binary Hopfield network. An A/D conversion neural network is first chosen because it typically consists of small number of neurons and the local minima can be easily computed.

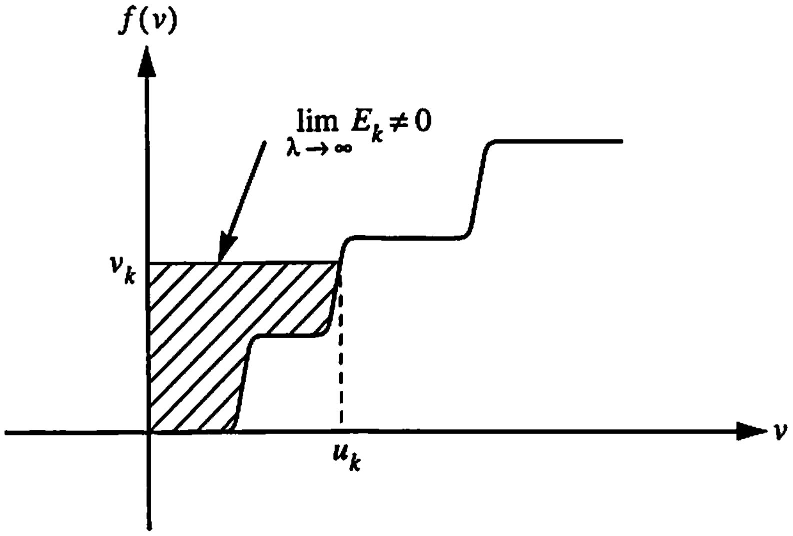
The characteristics of a multi-level neuron is first reviewed in the next section. The energy function expression of a multi-level neuron is described. The technique of hardware annealing is discussed the section 3.1.3. In the section 3.1.4, hardware annealing is applied to a multi-level neural A/D converter to help the result to quickly reach the optimal solution. Simulation results of the energy contour of a multi-level neural network at various stages of the hardware annealing process are given.

3.2 Hopfield Neural Network with *m*-ary Neurons

Most neural networks use two-level binary neurons in either a continuous-valued or threshold logic model. If a combinatorial problem is no longer a binary-valued optimization such as the quadratic 0-1 programming, it is in general difficult to accommodate the task with a network of two-level neurons. A multi-level Hopfield neural network [56] is an m-ary extension of the binary model. An m-ary content addressable memory (CAM) model has been shown [52] to be a novel technique for recovering heavily distorted images. Obvious advantage of the multi-level version over the binary Hopfield neural network is the ability to handle more information with a smaller number of neurons. A two-level neuron produces output with 1-bit accuracy while an m-level neuron produces output with $\log_2 m$-bit accuracy. An analog-to-digital (A/D) converter using a n-neuron Hopfield network with $m = 2^p$ levels is equivalent to $(p \cdot n)$-bit binary network. Therefore, a multi-level neuron is quite desirable for VLSI



(b) Transfer characteristic of multilevel neuron



(b) Neuron energy: $E_k = \int_0^{v_k} f^{-1}(\omega)\, d\omega$

Figure 3-1: Multilevel neuron and its 'generalized energy'.

implementation for complex optimization problems. The activation shown in Figure 3-1 (a) represents a multi-level thresholding operation defined as

$$f_m(x) = a_k,\ \theta_k < x \le \theta_{k+1}, \qquad k = 0,1,\cdots,m-1 \tag{3.1}$$

where θ_k and a_k are the threshold and output values, respectively. We also define the threshold and output step sizes for $k = 1,2,\cdots,m-1$, as $\Delta\theta_k = \theta_k - \theta_{k-1}$ with $\Delta\theta_1 = \theta_1$, and $\Delta a_k = a_k - a_{k-1}$. If a specific neuron in a network, e.g., the neuron-*i*, needs to be referenced, then two indices are used such as $\theta_{i,k}$ and $a_{i,k}$. A typical two-level neuron can be modeled by $y = f_b(x-\theta)$ where θ is the threshold value of the neuron. To represent a multi-level neuron with m threshold values θ_k, $k = 0,1,\cdots,m-1$, a neuron transfer function of the form

$$f_m(x) = \sum_{k=1}^{m} c_k f_{b,k}(x-\theta_k) \qquad \theta_k < \theta_{k+1} \tag{3.2}$$

can be used. Here, the constant c_k is a scaling factor for the *k*-th level, and if $a_1 = 0$ then $a_k = c_1 + c_2 + \cdots + c_{k-1}$. Therefore, the output level has a step size of $\Delta a_k = a_{k+1} - a_k = c_k$. As in the binary neuron case, we can use a differentiable, monotonically increasing accumulation function, instead of an ideal staircase function $f_m(x)$. The sigmoid function has been widely used in the neural networks. This function is a bounded, differentiable real function that is defined for all real input values, and it has a non-negative value everywhere. By replacing $f_i(\cdot)$ with the sigmoid function, (3.9) becomes

$$f_m(x) = \sum_{k=0}^{m-1} \frac{c_k}{1+e^{-\lambda_k(x-\theta_k)}}. \tag{3.3}$$

where λ_k the gain control parameter of the *k*-th binary activation function. For the shake of simplicity, $\lambda_k = \lambda$ and $c_k = 1$ are used for all *k*.

The governing equation for the *i*-th neuron in an *n*-neuron network can be expressed as a nonlinear differential equation

$$C_i \frac{du_i(t)}{dt} = -T_i u_i(t) + \sum_{j=1, j\neq i}^{n} T_{ij} v_j(t) + I_i(t), \tag{3.4}$$

where C_i and T_i are the equivalent capacitance and conductance at the input node of the i-th neuron, and T_{ij} is the conductance between the i-th and the j-th neurons. The voltage at the input node of the i-th neuron is $u_i(t)$ and the output voltage of the j-th neuron is $v_j(t)$. The energy function associated with the network can be expressed as

$$E = -\frac{1}{2}\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} T_{ij} v_i v_j + \sum_{i=1}^{n} T_i \int_0^{v_i} f_{m,i}^{-1}(v) dv - \sum_{i=1}^{n} I_i v_i \tag{3.5}$$

where $f_{m,i}(\cdot)$ is the transfer function of the i-th multi-level neuron. The time derivative of the energy function is

$$\frac{dE}{dt} = -\sum_{i=1}^{n} \frac{dv_i}{dt}\left(\sum_{j=1, j\neq i}^{n} T_{ij} v_j - T_i u_i + I_i \right). \tag{3.6}$$

Substituting (3.4) into (3.6), we can obtain

$$\frac{dE}{dt} = -\sum_{i=1}^{n} \frac{dv_i}{dt}\left(C_i \frac{du_i}{dt} \right) = -\sum_{i=1}^{n} C_i f_{m,i}^{-1}(v_i) \left(\frac{du_i}{dt} \right)^2 . \tag{3.7}$$

Since $C_i > 0$, $\forall i$, $dE/dt \leq 0$ if $f_m(\cdot)$ is chosen to be a monotonically increasing function. The operation of the network evolves in the direction of decreasing the energy of the network until the equilibrium state is reached. As a matter of fact, all underlying equations (3.4)-(3.7) are the same form as those of the binary Hopfield network. However, there exists one fundamental difference in the energy function. In the binary case, the second term in (3.5) vanishes as the neuron gain approaches an infinity. As noted in the previous chapter, it represents a closed area indicated in Figure 3-1 (b). For a multi-level neuron, it approaches a non-zero function $\Psi(v)$ as the neuron gain becomes an infinity, i.e., all binary activation functions $f_k(x-\theta_k)$, $\forall k$, in (3.2) approach step functions. Assuming that $a_0 = 0$ and $\theta_k \geq 0$, $\forall k$,

$$\Psi(v) \equiv \lim_{\lambda\to\infty}\left(\int_0^v f_m^{-1}(v)dv\right) = \sum_{l=1}^{k}\Delta a_l\theta_l + (v-a_k)\theta_{k+1}$$

$$= \sum_{l=1}^{k}\sum_{p=1}^{l}\Delta a_l\Delta\theta_p + (v-a_k)\theta_{k+1}, \qquad a_k < v \le a_{k+1}. \tag{3.8}$$

3.3 Neural Network Analog-to-Digital Converter: A Combinatorial Optimization with Local Minima

The one-dimensional A/D decision network is suitable for illustrating the properties of the multi-level Hopfield neural network. In this case, a neuron with m discrete levels may be realized by m-1 amplifiers connected in parallel with their output currents summed together as indicated by (3.2). The summed current can be converted to the output voltage of the neuron by a current-to-voltage converter. The output voltages are connected back to the neuron inputs through interconnection conductances. An analog input value and reference voltage are applied to all neurons. A general schematic diagram shown in Figure 2-1 also represents a Hopfield neural network with multi-level neurons. We use multi-level neurons with integer values $a_{i,k} = k$, $k = 0,1,2,3$, for all i. The synapse weight values can be determined by minimizing the squared value of the difference between the input analog value and the corresponding digital representation,

$$E_0 = \left(x_a - \sum_{i=0}^{n-1} m^i v_i\right)^2, \qquad v_i \in \{0,1,2,3\}. \tag{3.9}$$

The analog input x_a has a normal range of $-0.5 \le x_a \le 63.5$ and may be scaled to the range of 0 to 5V if electronic hardware implementation is considered. After expanding (3.9) and discarding the irrelevant constant term, we obtain

$$\hat{E}_0 = -\frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0, j\ne i}^{n-1}\left(-m^{i+j}\right)v_i v_j - \sum_{i=0}^{n-1} m^i x_a v_i + \frac{1}{2}\sum_{i=0}^{n-1} m^{2i} v_i^2. \tag{3.10}$$

Without loss of a generality, we assume uniform spacing between the neighboring threshold values and the output levels in a specific neuron, i.e., $\Delta\theta_{i,k} = \theta_{i,k} - \theta_{i,k-1} = \Delta\theta_i$, and $\Delta a_{i,k} = a_{i,k} - a_{i,k-1} = \Delta a_i$, for all k in the neuron-i. In this case, (3.8) is simplified as

$$\Psi(v) = \frac{1}{2}v(v+1)\Delta a_k \Delta\theta_k + (v - a_k)\Delta\theta_{k+1} \qquad a_k \le v < a_{k+1}. \tag{3.11}$$

The assumption $a_k = k$ is used in the first term of (3.11). If we define the first term as $\Psi_0(v) = 0.5v(v+1)\Delta a_k \Delta\theta_k$, $a_k \le v < a_{k+1}$, then $\Psi(v) \ge \Psi_0(v)$ where the equality holds only if $v = a_k = k$. Hence, the second term of (3.11) can be regarded as the biased energy for the constraint $v \in \{0, 1, 2, 3\}$. In Figure 3-2, the exact and quadratic approximation are shown. As shown in Figure 3-2 (b), the energy landscape of concave function remains unchanged with the approximation. This is generally true even when the neuron gain is finite but large. Note that, in contrast to the intentional addition of constraint energy in conventional optimization problems [1,25] or in binary neuron case [4], it is already included in the energy function (3.8). Therefore, over integer values $v_i \in \{0,1,\cdots,m-1\}$, $i = 0,1,\cdots,n-1$, the minimization of (3.5) is equivalent to the minimization of

$$\overline{E} = -\frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0,j\ne i}^{n-1} T_{ij}v_i v_j - \sum_{i=0}^{n-1}\left(I_i - \frac{1}{2}T_i\Delta\theta_i\Delta a_i\right)v_i + \frac{1}{2}\sum_{i=0}^{n-1} T_i\Delta\theta_i\Delta a_i v_i^2. \tag{3.12}$$

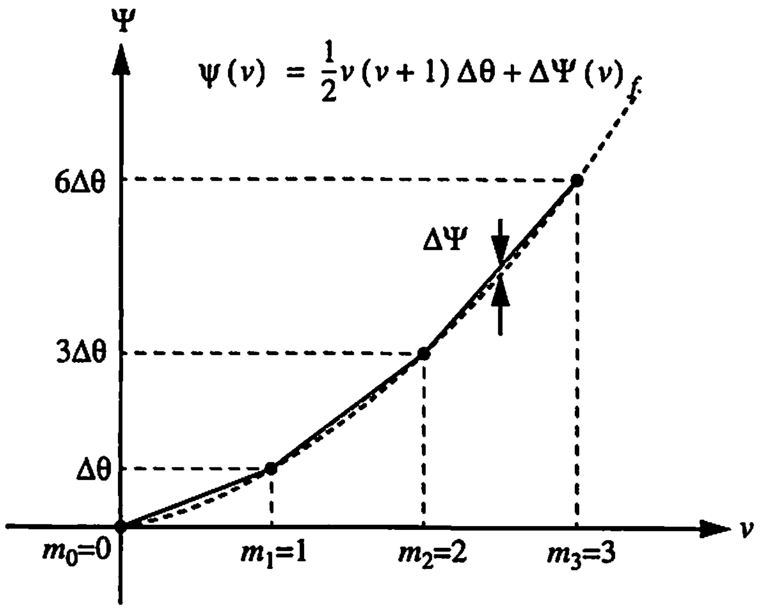
Note also that $E = \overline{E}$ only if $v_i = a_{i,k}$, $\forall i$. By equating corresponding terms in (3.10) and (3.12), we can obtain

$$T_{ij} = -m^{i+j}, \tag{3.13}$$

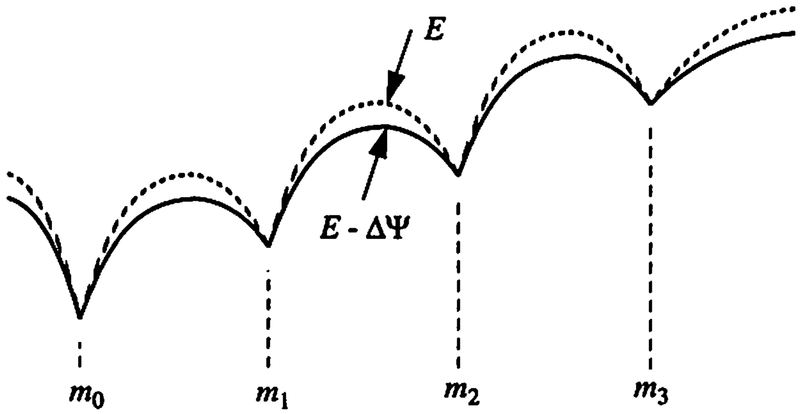
$$I_i \equiv I_{ai} + I_{ri} = m^i x_a + \frac{1}{2}T_i\Delta\theta_i\Delta a_i \equiv T_{ai}x_a + T_{ri}x_r,$$

and

$$T_i\Delta\theta_i\Delta a_i = m^{2i}.$$



(a) Approximation of E_f by subtracting residual constraint energy $\Delta\Psi$ with $\Delta\Psi=0$ at m_k, $k=0,1,2$.



(b) Corresponding changes in total energy function.

Figure 3-2: Non-zero 'generalized energy' term and its approximation.

If an ideal current source is used for the external bias term $I_{ri} = 0.5 T_i \Delta\theta_i \Delta a_i = 0.5 m^{2i}$, then

$$T_i = \sum_{j=0, j\neq i}^{n-1} \left| T_{ij} \right| + T_{ai} = m^i \left(1 + \sum_{j=0, j\neq i}^{n-1} m^j \right). \tag{3.14}$$

where the input transconductance of an amplifier itself is neglected. A 4-ary, 3-neuron Hopfield neural network for an 6-bit equivalent A/D conversion is shown in Figure 3-3. By using (3.14) and $\Delta a_{i,k} = 1$, $\forall i,k$, the threshold values are given by vectors $\boldsymbol{\theta}_0 = (1/21)[1, 2, 3]$ for the least-significant bit neuron, and $\boldsymbol{\theta}_2 = (8/3)[1, 2, 3]$ for the most-significant bit neuron, respectively, as shown in the figure. If a bias voltage $x_{ri} = 1$, $\forall i$, is used, then an additional term $T_{ri} = 0.5 m^{2i}$ should be added to the equivalent input conductance given in (3.14). This change in T_i and correspondingly the change in $\Delta\theta_i = m^{2i}/T_i$ reflect different effects of an ideal current bias I_{ri} and voltage bias x_{ri} with a resistance $R_{ri} = 1/T_{ri}$ connected to the neuron input. The amount of currents that flow into the neuron input are I_{ri} and $T_{ri}(x_{ri} - u_i)$, respectively. However, if the weighted summation $\sum_{j=0, j\neq i}^{n-1} T_{ij} v_j + I_i$ is performed by a summing circuit and its input is maintained at a virtually grounded potential, two current values are the same and the equivalent conductance T_i is solely determined by the summing circuit.

Let us carefully examine the transfer function of the A/D decision network. The steady state equilibrium is reached when $du_i/dt = 0$ for all i. From (3.4), we can obtain the steady-state solution for the neuron input voltage as

$$u_i = \frac{\sum_{j=0, j\neq i}^{n-1} T_{ij} v_j + T_{ri} x_{ri} + T_{ai} x_a}{T_i}, \qquad i = 0, 1, \cdots, n-1. \tag{3.15}$$

Let the value of u_i be in the range

$$\theta_{i,k} \leq u_i \leq \theta_{i,k+1} \tag{3.16}$$

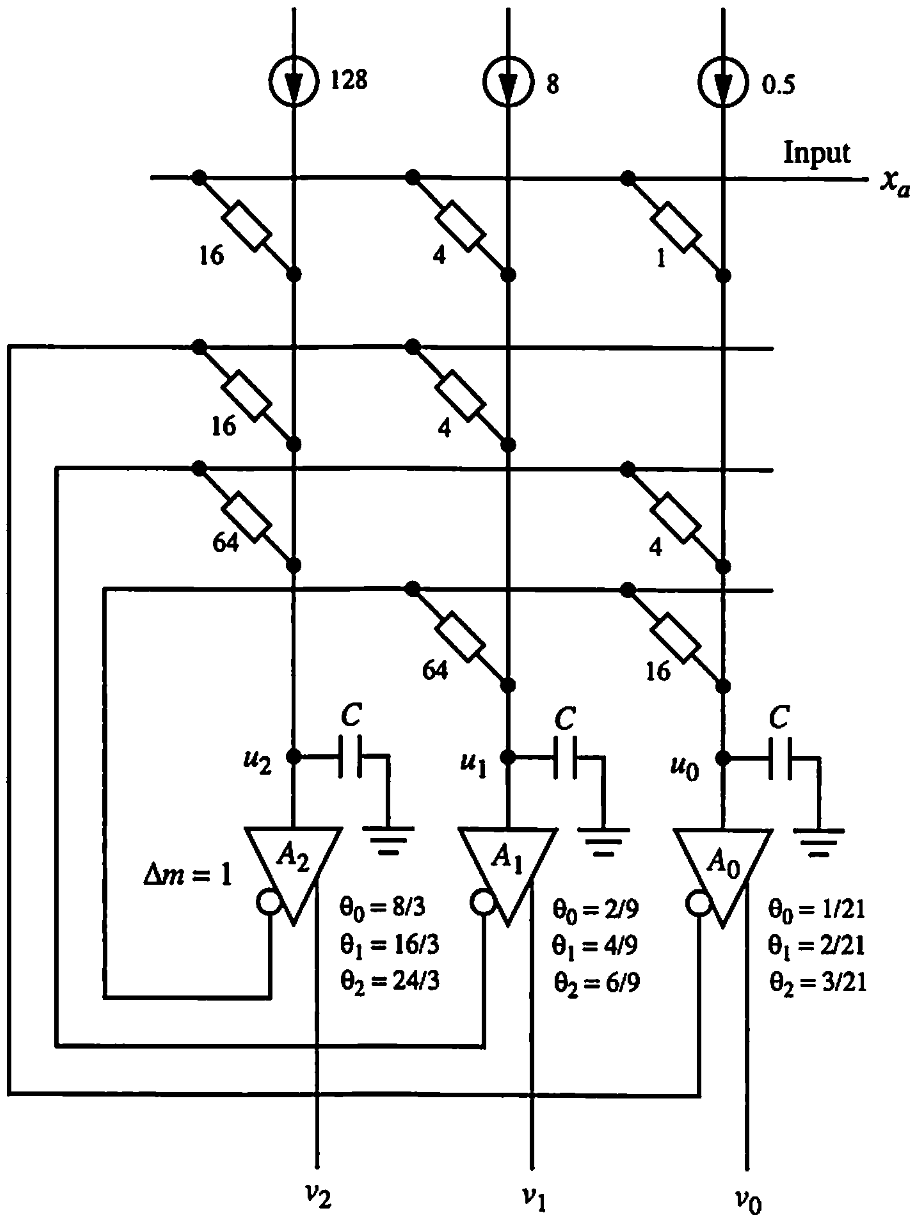


Figure 3-3: 4-Level, 3-neuron Hopfield A/D converter.

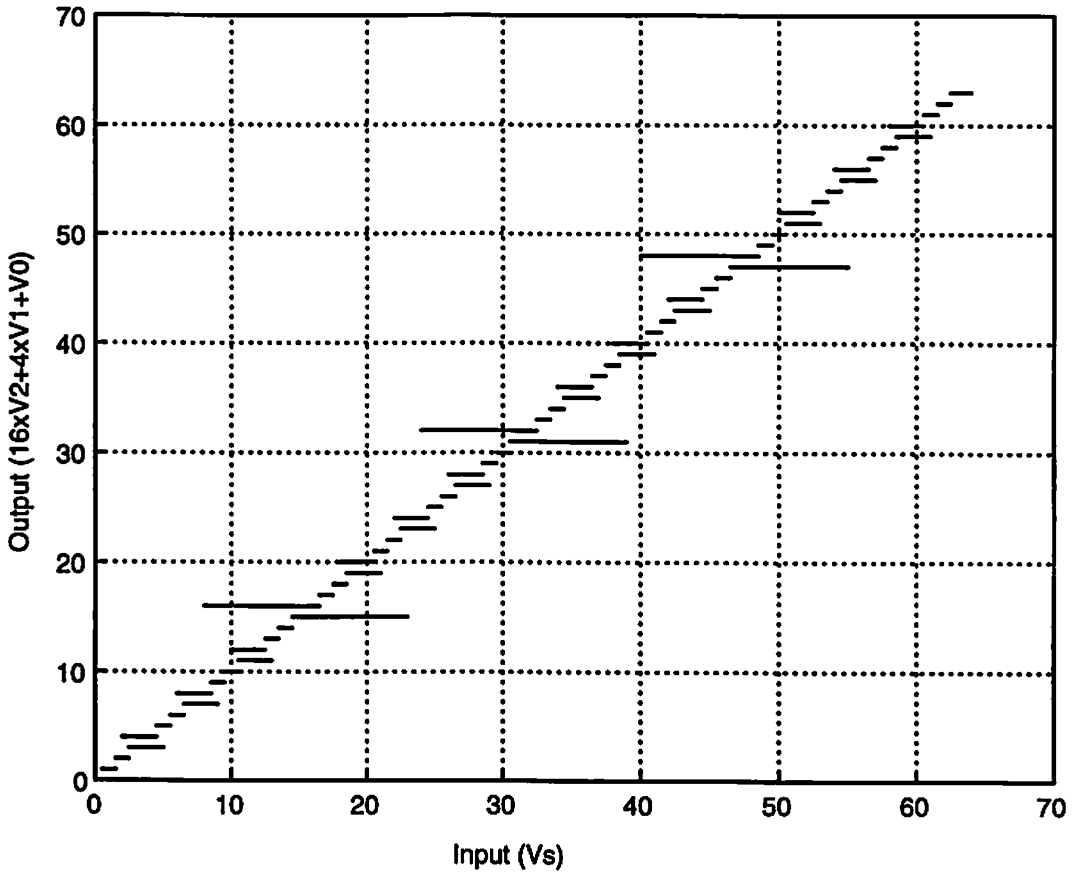
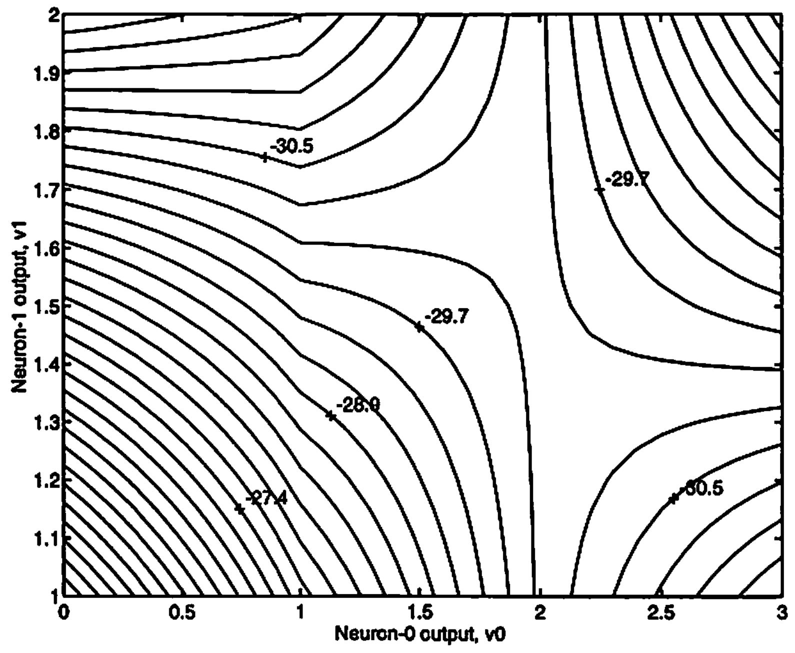
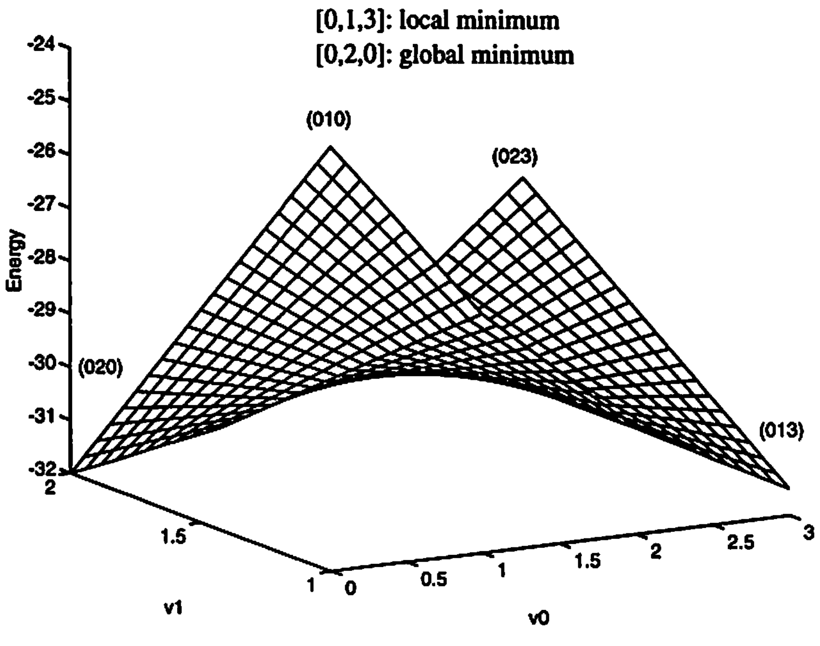


Figure 3-4: Transfer characteristics of 4-level, 3-neuron Hopfield A/D converter.



(a) Contour plot of energy landscape.



(b) Energy landscape.

Figure 3-5: Local minima problem, g=infinity, v2=0, Xa=8.

such that the i-th neuron output is $v_i = a_{i,k}$. Substituting (3.15) into (3.16), we can obtain all possible ranges of input value at which the steady-state solution is given by (3.16) as

$$\frac{-\sum_{j=0,j\neq i}^{n-1} T_{ij}v_j + T_i\theta_{i,k} - T_{ri}x_{ri}}{T_{ai}} \leq x_a \leq \frac{-\sum_{j=0,j\neq i}^{n-1} T_{ij}v_j + T_i\theta_{i,k+1} - T_{ri}x_{ri}}{T_{ai}}. \tag{3.17}$$

Figure 3-4 shows the plots of all possible steady-state solutions (3.17) in the A/D decision network. For some ranges of input value, there exist a multiple of digital representations for the same input x_a. Only one of them is the optimal solution and all others are sub-optimal in terms of the squared-errors (3.9). The actual output out of those multiple candidates is dependent on the initial condition $u_i(0)$ of the dynamical equation (3.4). For all zero initial conditions, the energy value is quite large, i.e., $E = 0$ from (3.5), but the optimal solution can not be guaranteed. Figure 3-5 shows the computed energy function versus v_0 and v_1 with $v_2 = 0$, $u_i(0) = 0$, $i = 0,1,2$, and $x_a = 8$. There are two minima at the vertices $[v_2 \ v_1 \ v_0] = [0 \ 1 \ 3]$ and $[0 \ 2 \ 0]$ which represent decimal numbers 7 and 8, respectively. The vertex $[0 \ 2 \ 0]$ is exactly the desired output with a lower energy level. At some input values, two local minima can exist. For example, if $x_a = 20$, the possible outputs are 15, 19, and 20.

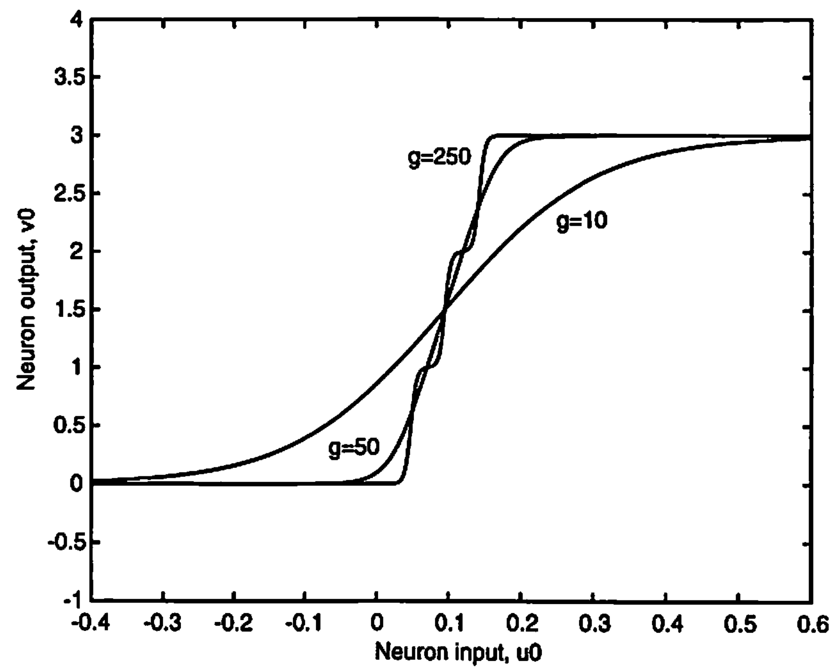
3.4 Hardware-Based Annealing

The hardware annealing is a paralleled, hardware-based optimization technique that can be coupled with the recurrent associative neural networks, in which the energy function can be expressed as a quadratic function of output variables. The voltage gains of the neurons are gradually increased in a continuous manner to achieve a similar effect as changing temperature in simulated annealing. If.the multi-level neuron is achieved by a combination of binary activation functions as in (3.3), the gain control of the composite function $f_m(x,\lambda)$ can be performed by controlling the individual gain value λ_k of $f_k(x,\lambda_k)$, $k=1,2,\cdots,m-1$, simultaneously. For the simplicity, the gain control parameter are the same for all binary and multi-level activation functions, i.e., $\lambda_{i,k}=\lambda$, $\forall i,k$. The gain is controlled by a smooth function of time as given by

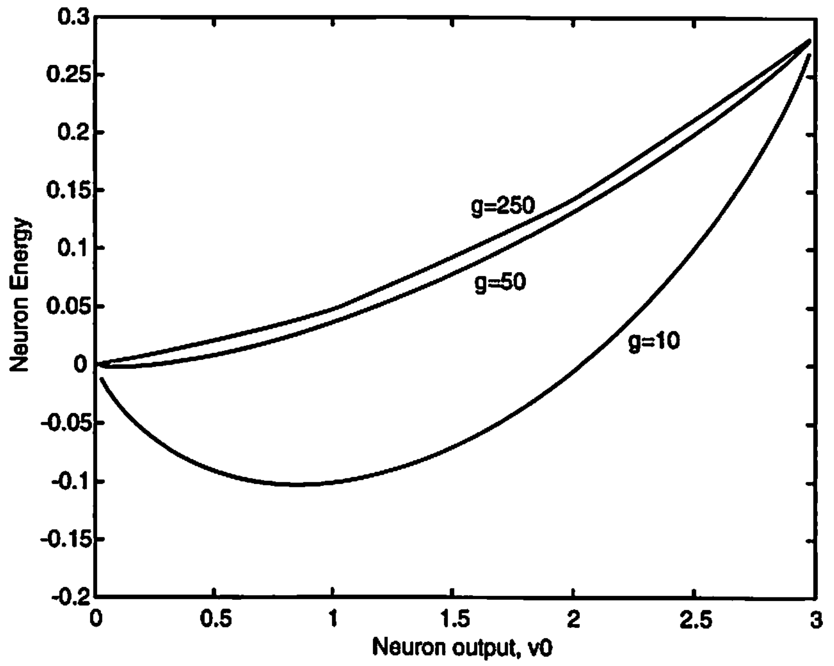
$$\lambda = g(t) \geq 0, \qquad \forall t \geq t_0 \tag{3.18}$$

where t_0 is the time at which the network begins to operate with initial state voltages $u_i(t_0)$, $i=0,1,\cdots,n-1$, and a initial neuron gain $g(t_0)$. Typically, with $t_0=0$, the zero initial condition $u_i(t_0)=0$, $\forall i$, is used and $g(t_0)$ is chosen to be a small positive number such that $0<g(t_0)<1$. As will be shown later in the cellular neural network case in which the collective computational properties are similar to those of the Hopfield network, the annealing gain $g(t)$ also need to be a monotonically increasing function of time for a stable operation. Figure 3-6 shows the transfer curves and the corresponding energy terms of the first neuron (neuron-0) for gain values $g=10$, 50, and 250. Since the incremental threshold value $\Delta\theta$ is small, a multi-level appearance of neuron characteristics is clear at high gain. In Figure 3-7, the third neuron (neuron-2) is examined at the same gain values. Note that the threshold θ_k and output level a_k are not affected by this gain control operation, although in some cases they are not clearly distinguishable when the gain is small, as in Figure 3-6. Although the multi-level neuron is highly nonlinear device, it can be approximated by a linear amplifier at its operating region if the neuron gain is relatively small. Therefore, during the first half of the annealing process, the argument of eigenvalues in the linearized system model can be applied in the same way as described in the section 2.3.

By using two-level activation function as a basic element, Figure 3-8 shows a possible realization of a variable-gain multi-level neuron for hardware annealing. The input x is first subtracted by a threshold value θ_k, $k=1,2,\cdots,m-1$, in each of m-1 branches. The

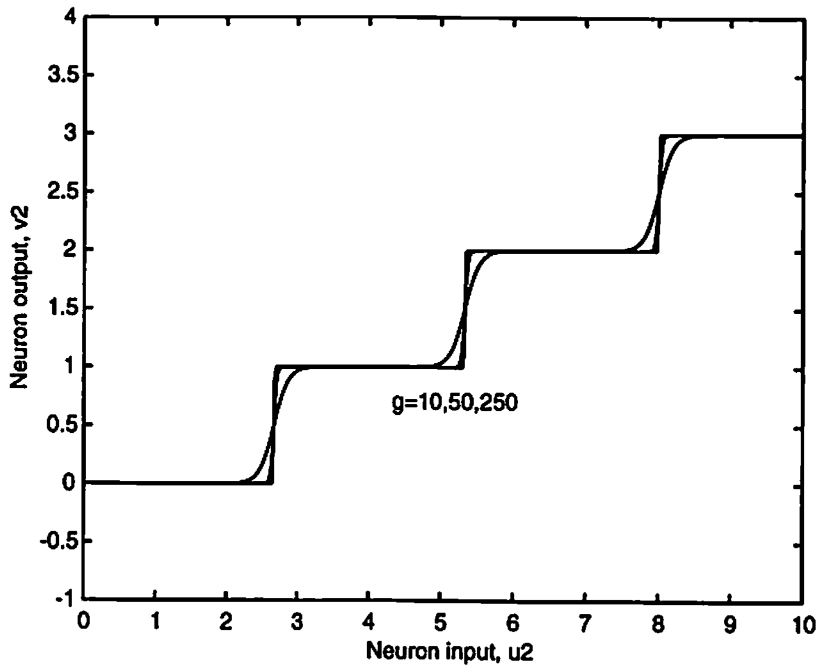


(a) Transfer curves of the 1st neuron for different gain values.

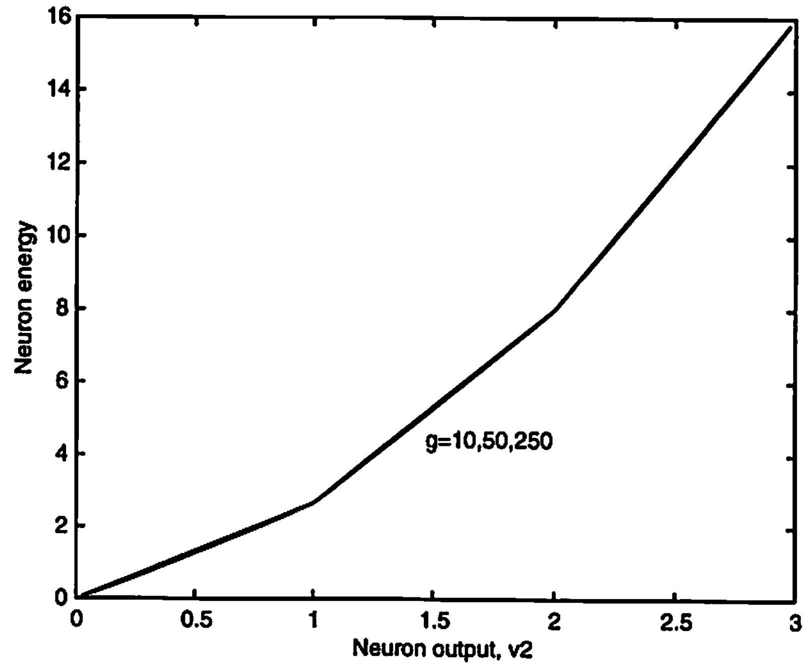


(b) Corresponding energy curves.

Figure 3-6: Transfer functions and energies of the 1st neuron.

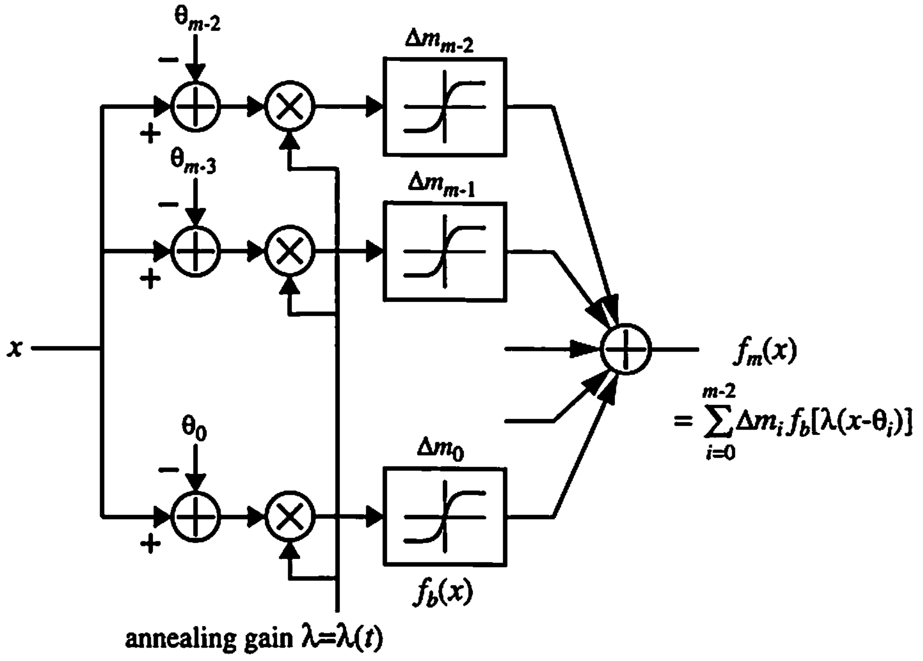


(a) Transfer curves of the 3rd neuron for different gain values.

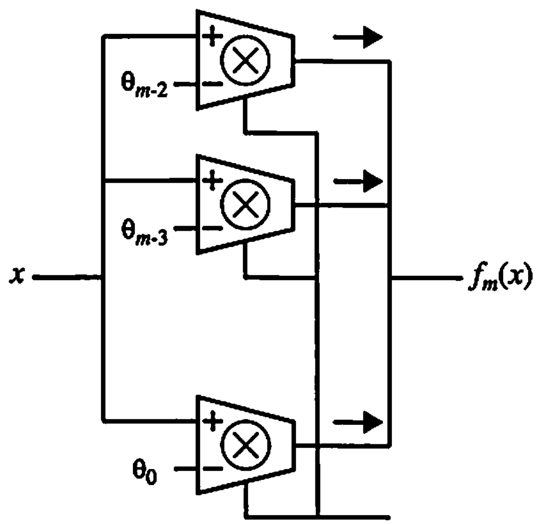


(b) Corresponding energy curves.

Figure 3-7: Transfer functions and energies of the 3rd neuron.



(a) Block diagram.



(b) Use of m-1 transconductance multipliers.

Figure 3-8: Variable-gain multilevel neuron.

resulting values are then amplified by the gain factor $\lambda = g(t)$ in variable-gain amplifiers or two-quadrant multipliers. The fixed-gain nonlinear amplifiers following the multipliers have the sigmoid-like transfer functions. The output values are then summed together to produce the desired characteristics (3.3) with a time-varying neuron gain $\lambda = g(t)$ and $c_k = 1$, $\forall k$. With this scheme, the binary representation of a multi-level value is readily available from the outputs of two-level nonlinear amplifiers. By using $m-1$ transconductance multipliers with differential input, the variable-gain m-ary neuron can be realized as shown in Figure 3-8 (b).

3.5 Simulation results

In order to examine the hardware annealing for the A/D conversion network described above, the same input value $x_a = 8$ and zero initial condition are used in the simulation. By solving the equations (3.4), the network is first operated in the normal high-gain condition and then annealed by a linearly increasing schedule given by

$$g(t) = \alpha t, \qquad \alpha > 0. \tag{3.19}$$

The resulting waveforms for neuron states $u_i(t)$, $i = 0,1,2$, together with the gain control function are shown in Figure 3-9. Here, the dotted lines in Figures 3-9 (b) - (d) represent the threshold values of each neuron. The output from the unannealed operation ($0 \le t < 1$ in normalized time-scale) is the sub-optimal solution $[0\ 1\ 3]$ as mentioned in the previous section, while the desired output $[0\ 2\ 0]$ is reached in the annealed operation ($1 \le t < 3$). In Figures 3-10 (a) and (b), the trajectory of (v_0, v_1)-pair and output waveforms during the unannealed and annealed operations are plotted, respectively. Note that the steady-state values $u_i(1)$, $i = 0,1,2$, from the previous unannealed operation are used as the initial condition of the annealed operation. As described in Chapter I, the global minimization of the energy function is achieved by dynamical transformation of the energy landscape

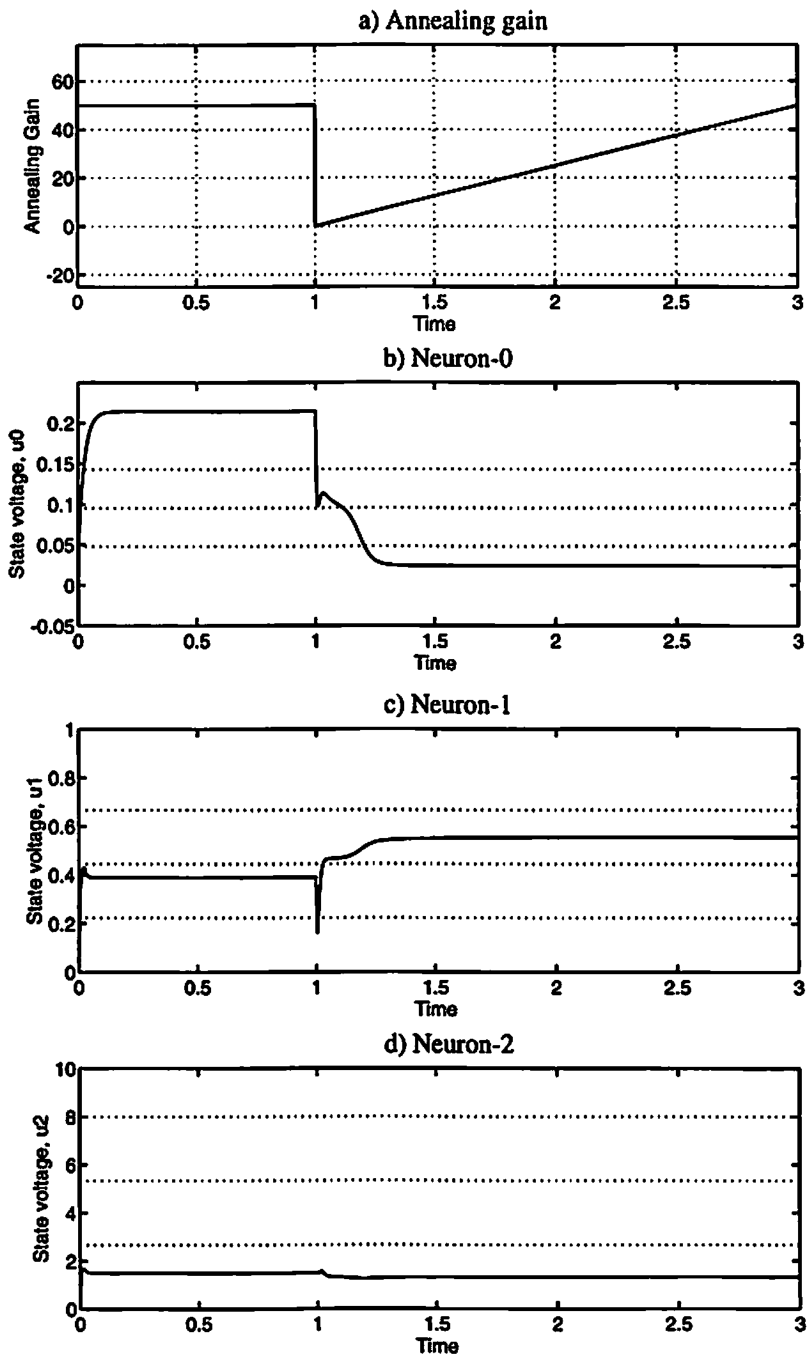
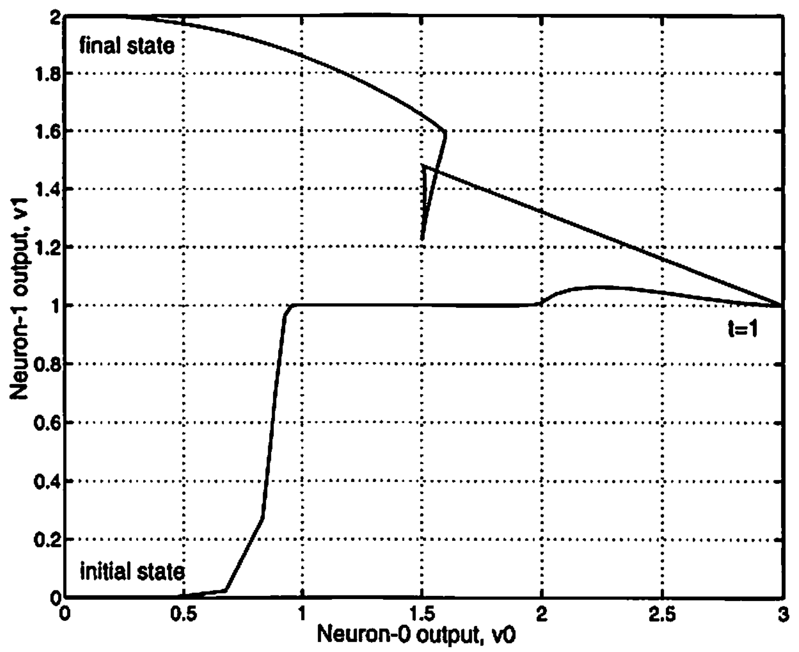
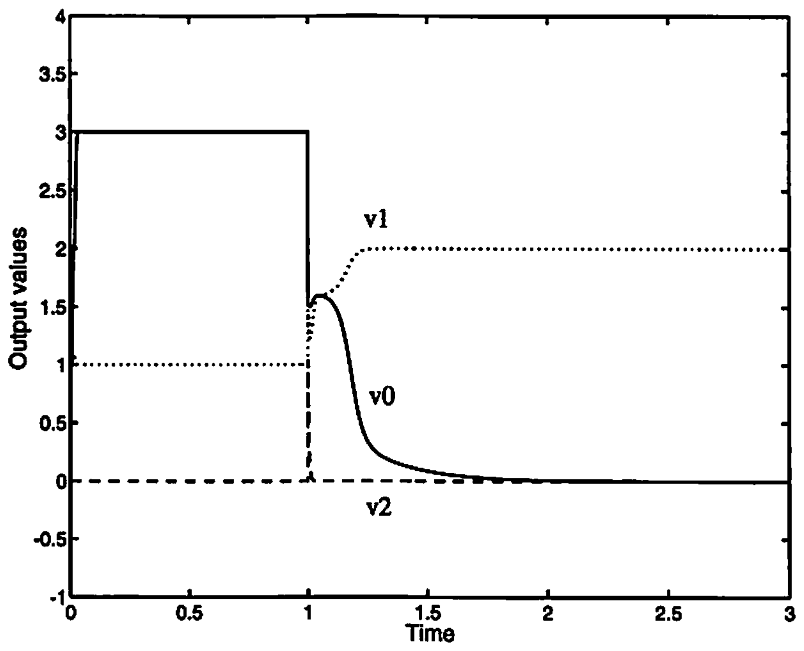


Figure 3-9: Annealing gain and state values for unannealed (0<t<1) and annealed (1<t<3) operations.



(a) Trajectories of v0 & v1 for zero initial condition.



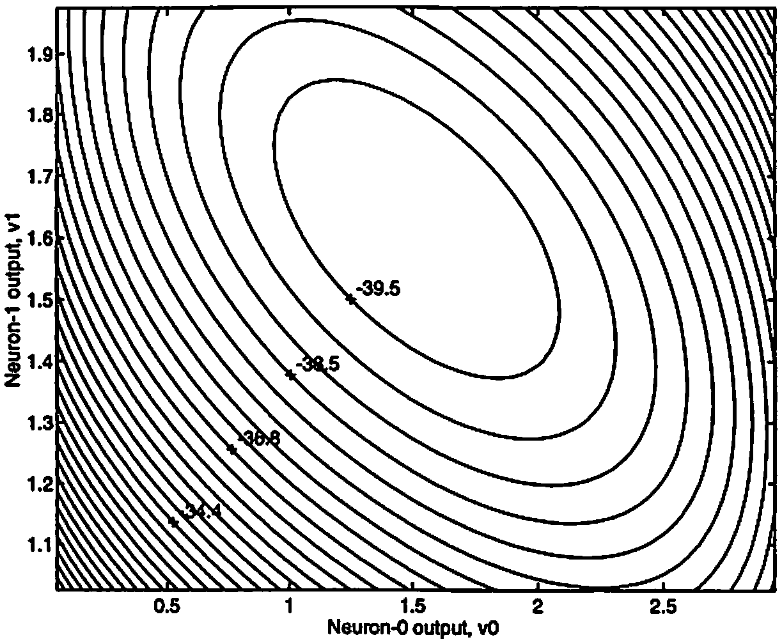
(b) Output waveforms.

Figure 3-10: Neuron outputs for unannealed and annealed operations.

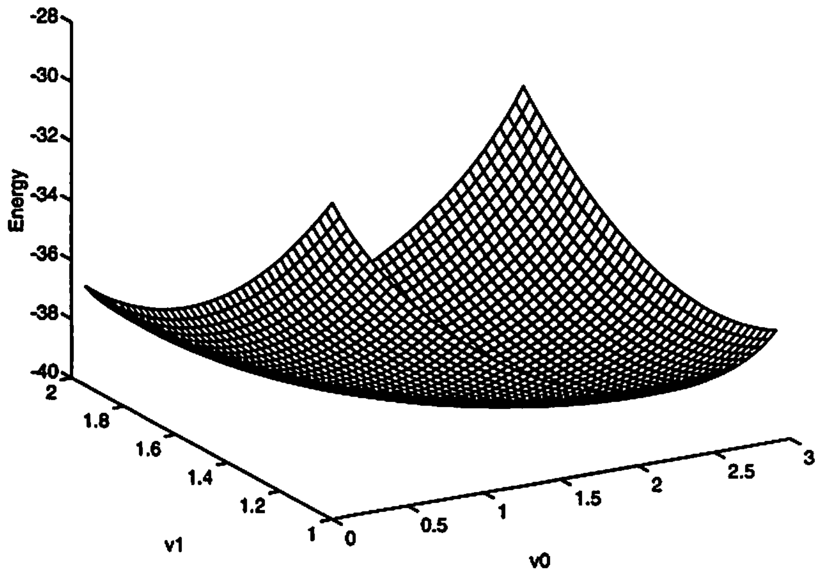
from a convex function to indefinite or concave function. Figure 3-11 shows an appearance of E for the intermediate neuron gain of 10, at which E is convex and has a unique minimum. At a higher gain value $g = 100$, the energy landscape is a saddle-shaped, indefinite function of neuron outputs, as shown in Figure 3-12. Similar results are obtained at different input values if there exist multiple minima.

With zero initial condition, the transfer characteristics of A/D conversion network is shown in Figure 3-13 (a). It was obtained by applying integer input values $i = 0,1,\cdots,63$ in an unannealed network and taking the steady-state outputs in decimal number format. When the proposed annealing technique is added, an ideal transfer curve shown in Figure 3-13 (b) can be obtained. If the input is close to values $i + 0.5$, $i = 0,1,\cdots,63$, below and above which the output differs by one, the A/D conversion network may result in systematic errors as shown in Figure 3-14. The input values used are $x_a = i + 0.49$, $i = 0,1,\cdots,63$. The conversion errors of unannealed and annealed A/D network for two cases are shown in Figure 3-15, where the comparisons are made with desired integer values instead of continuous function. For the second case, both unannealed and annealed operations produce additional errors of maximum one, which may be related to simulation tolerance but not to the global optimization technique.

As an illustrative example of global optimization techniques, the hardware-based annealing theory is applied to a Hopfield neural network for A/D conversion to facilitate a quick search for the optimal solution. The voltage gain of the neurons is gradually increased from a certain low value to a critically high value. The energy landscape was modified by the annealing process and the natural gradient-descent operation at various gain values help the solution to reach the optimal state. In addition, a multi-level m-ary neuron model is used in the Hopfield network to accommodate non-binary combinatorial optimization problems.

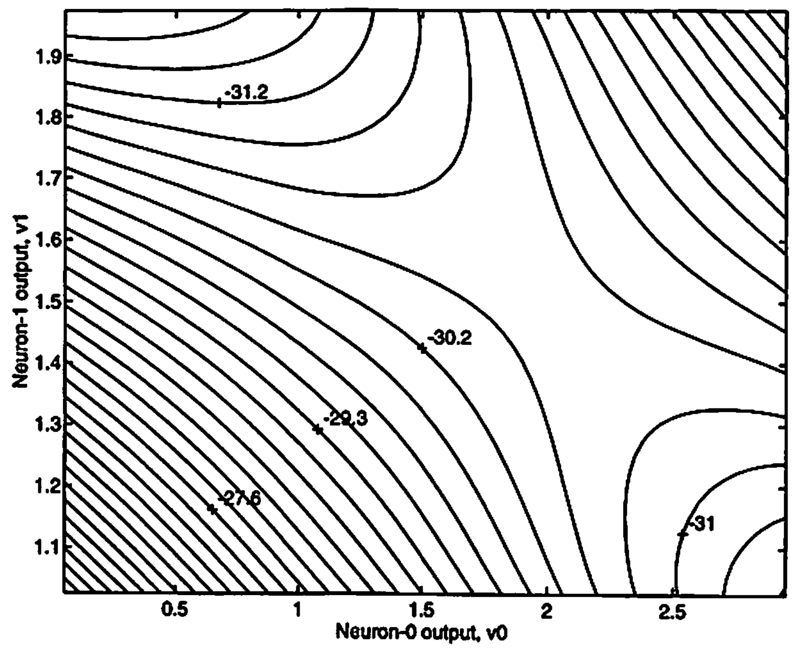


(a) Contour plot of energy landscape.

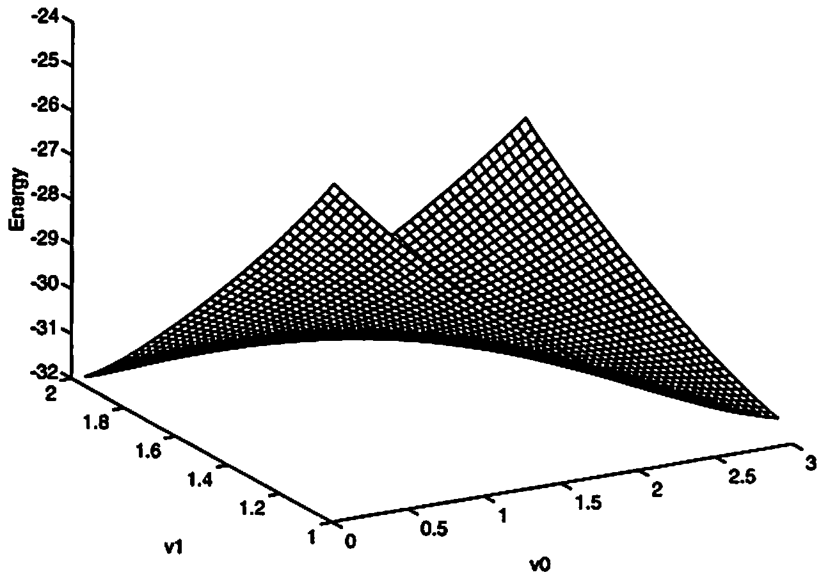


(b) Energy landscape.

Figure 3-11: Convex energy function at low annealing gain, g=10, v2=0, Xa=8.

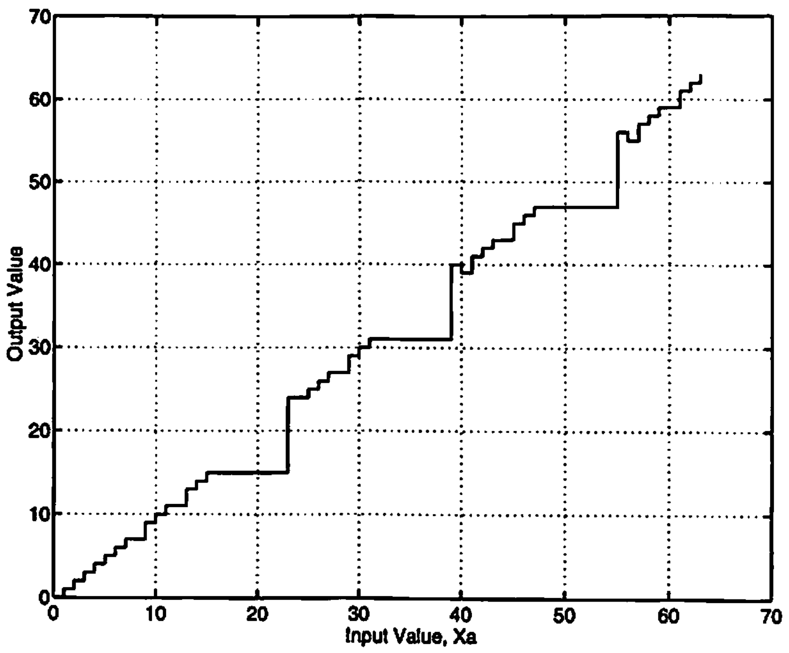


(a) Contour plot of energy landscape.

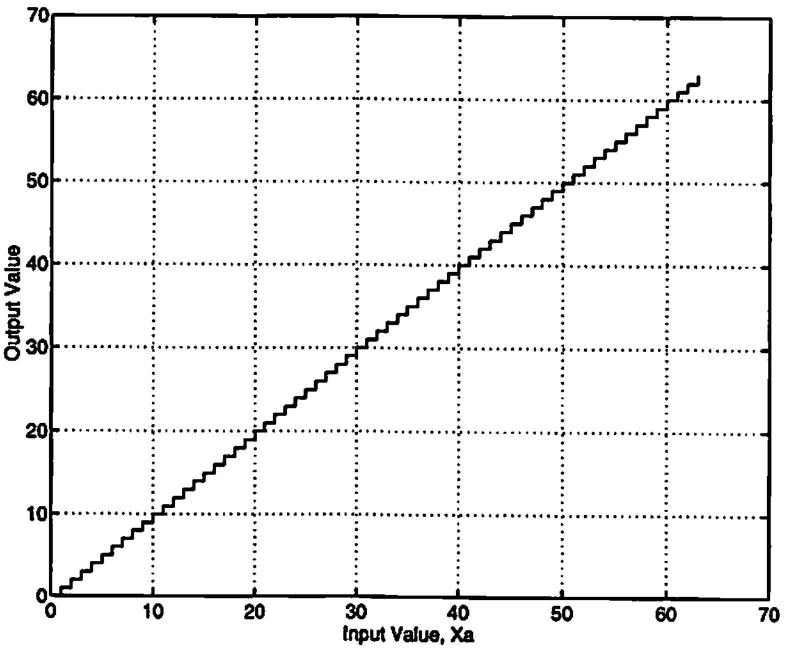


(b) Energy landscape.

Figure 3-12: Indefinite energy function at high annealing gain, g=100, v2=0, Xa=8.

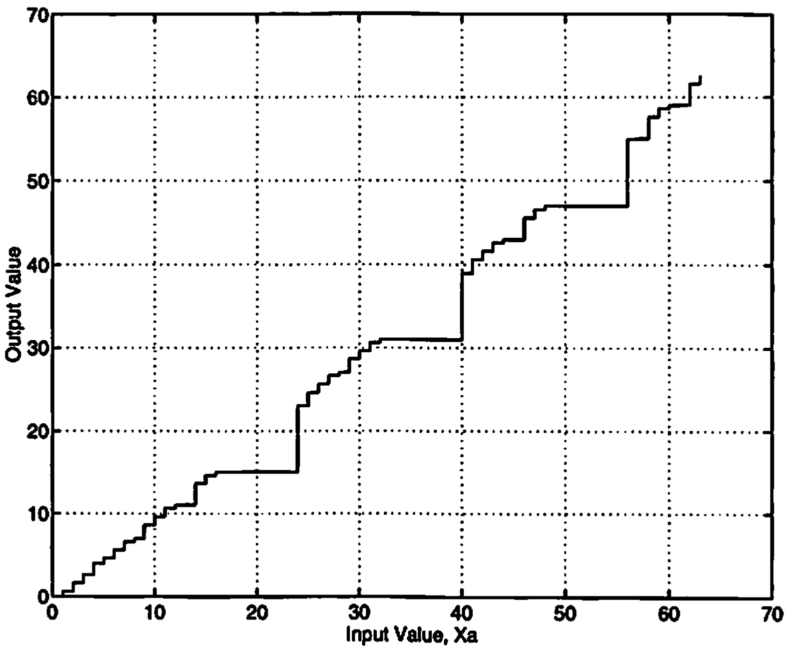


(a) Unannealed A/D conversion.

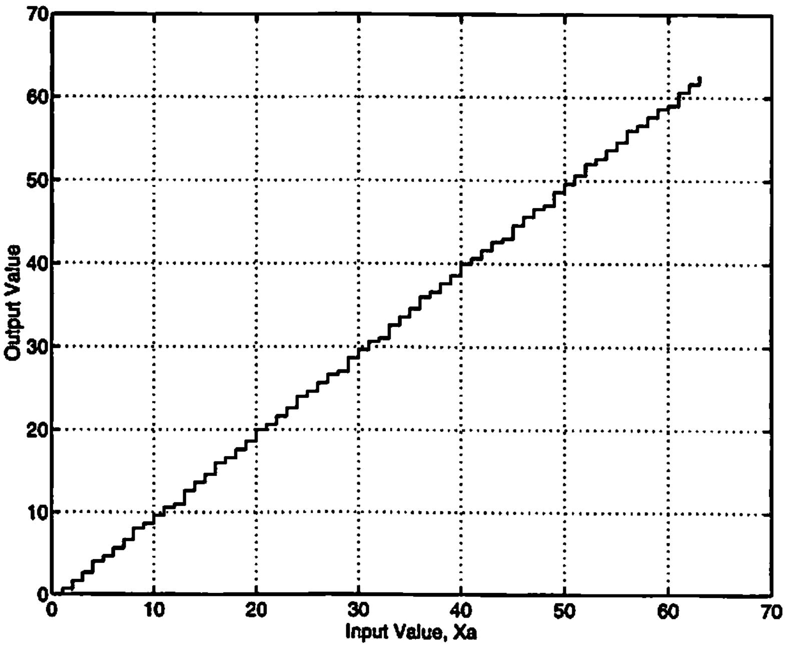


(b) Annealed A/D conversion.

Figure 3-13: Transfer curves of Hopfield A/D converter, Xa=integer values.

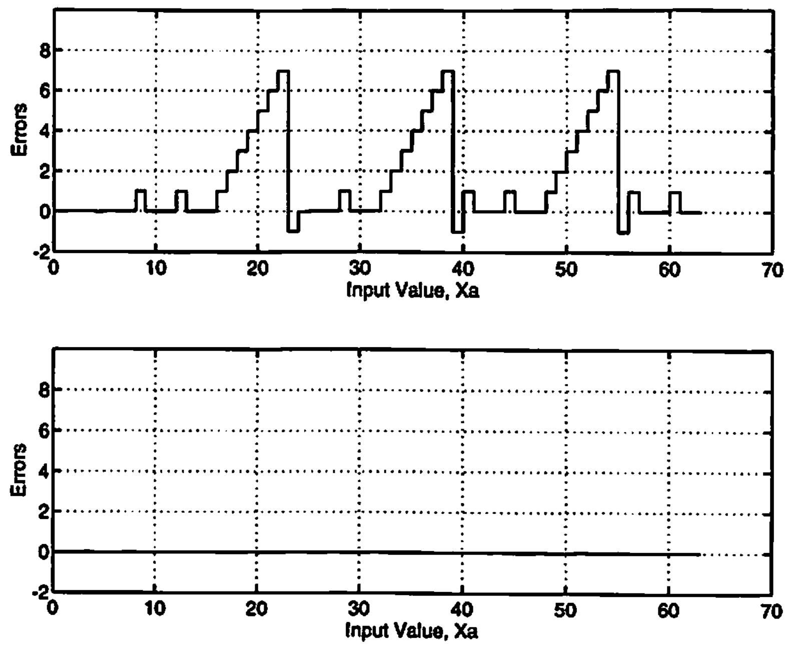


(a) Unannealed A/D conversion.

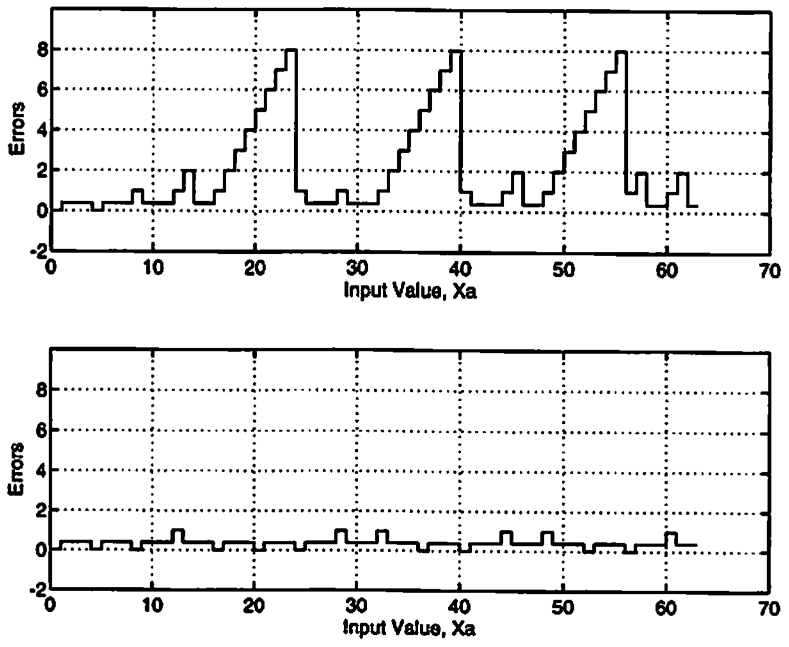


(b) Annealed A/D conversion.

Figure 3-14: Transfer curves of Hopfield A/D converter, Xa=integer values-0.49.



(a) Comparison of errors for Figure 3-13.



(b) Comparison of errors for Figure 3-14.

Figure 3-15: Comparisons of conversion errors (integer only).

Chapter IV

Optimal Solutions for Cellular Neural Networks by Hardware Annealing

4.1 Introduction

A cellular neural network (CNN) [59-63] is a massively parallel, locally connected, nonlinear dynamic system. The ability to seek a stable point in a multidimensional phase space, at which the generalized energy function of the network is locally minimized, makes it possible to use the CNN's in many areas of image-related signal processing [60,75-85]. Moreover, due to the space-independent and locally interconnected architecture, CNNs are suitable for VLSI implementation for high-speed, real-time applications. Since its systematic introduction in 1988 by Chua and Yang [59,60], many issues regarding architecture [64-67], stability [67-74], application [75-88], and circuit implementation [89-100] of CNNs have been addressed and remarkable results have been achieved. Under the mild conditions, a CNN is always stable and finds a locally optimum output in the steady state. However, the output so obtained may not be a globally optimum solution in terms of the generalized network energy, because there may exist multiple minima at which the energy is minimized locally. This might, in turn, cause sub-optimal network operations as long as the objective is to map the input signal in one space to the output in another space by minimizing the cost function involved.

In optimization-oriented applications such as nonlinear programming problems and feature recognition tasks, a quick search of the optimal solution is highly desirable. It is important to avoid any local minimum which can result in a sub-optimal solution or completely unwanted result. Existence of local minima is common in the solutions for many optimization-oriented artificial neural networks such as perceptrons and Hopfield networks. There has been various suggested procedures for the network to avoid getting stuck in the local minima. Frequently used schemes include simulated annealing [37-40] for stochastic hill climbing, mean field annealing [43-46] to deform the energy barrier profile and Boltzmann-machine technique with the addition of intentional noises. These techniques are quite capable of escaping the local minima. However, they can not guarantee to reach a globally optimal solution. The mean-field learning method uses a set of deterministic equations instead of extensive calculation of probabilities at each temperature as in the Boltzmann machine.

By adopting a hardware-based parallel annealing in analog electronics, the processing speed can be significantly increased. The hardware annealing is an efficient electronic version of mean-field annealing in which the equivalent temperature parameter of the network is increased to a predetermined high value and then decreased continuously down to a critical low temperature value. In fact, the local minima in recurrent associative networks such as binary or multilevel Hopfield networks [29] and cellular neural networks can be successfully eliminated by applying the hardware annealing directly to the analog neural networks. It does not require any stochastic procedure. Once the energy of the network is increased by reducing the gain of neurons to make the neuron outputs at the non-saturated region, the hardware annealing quickly searches for the globally minimum energy state, because the nonlinear system problem has been effectively linearized.

The organization of this chapter is as follows: In the next section, the characteristics of the energy in a CNN is described. Section 4.3 describes the principle and dynamical behavior of the hardware annealing. Several simulation results are presented in Section 4.4. The final conclusions and remarks are given in Section 4.5.

4.2 Local Minima and Energy Barriers in CNN

Figure 4-1 shows the three-dimensional diagram of a 4×4 CNN. The input and output are directing along the z direction, and the interconnections among cells are arranged in the (x,y) plane. A CNN has the Lyaponov function and collective computational properties similar to those of a Hopfield neural network. Thus, in addition to the basic architecture and its properties covered in Chapter II, it is also important to examine how the Lyapunov function of (2.17) is affected by given cloning templates, initial state, and input values. For the shake of simplicity, a basic CNN model with $r=1$ is used in this chapter, instead of general model described in Chapter II. Then, the matrix A can be defined as

$$\mathbf{A}=\begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & \mathbf{0} & \\ \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{A}_1 & & \\ & \mathbf{A}_1 & \mathbf{A}_0 & \ddots & \\ & & \ddots & \ddots & \mathbf{A}_1 \\ & \mathbf{0} & & \mathbf{A}_1 & \mathbf{A}_0 \end{bmatrix}, \tag{4.1}$$

where $\mathbf{A}_0$ and $\mathbf{A}_1$ are two $m\times m$ Toeplitz matrices with elements determined by a given cloning template. When the feedforward cloning template is written as

$$\mathbf{T}_A=\begin{bmatrix} a_2 & a_1 & a_2 \\ a_1 & a_0 & a_1 \\ a_2 & a_1 & a_2 \end{bmatrix}, \tag{4.2}$$

then $\mathbf{A}_0 = toeplitz([a_0\ a_1\ 0\cdots 0])$ and $\mathbf{A}_1 = toeplitz([a_1\ a_2\ 0\cdots 0])$.

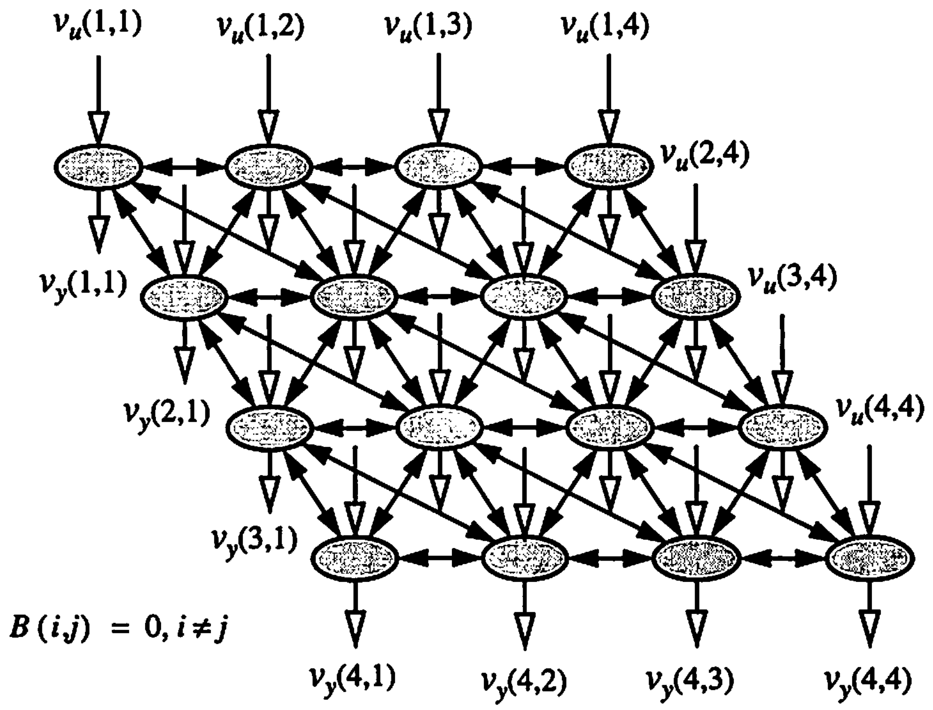


Figure 4-1: A 4-by-4 cellular neural network with scalar feedforward operator.

4.2.1 Energy Landscape

The matrix M of (2.20) can be diagonalized by an orthonormal set of eigenvectors as $\mathbf{M} = \mathbf{Q}\Lambda\mathbf{Q}^T$ where Λ is a diagonal matrix with eigenvalues of M on the diagonal and Q is an orthonormal matrix with the corresponding eigenvectors as columns. To find the maximum or minimum of $E(y)$ in R^N, we set the gradient of $E(y)$ equal to zero vector $\nabla_y E = \mathbf{0}$, where

$$\nabla_y E = \frac{\partial E}{\partial y} = -(\mathbf{A} - T_x \mathbf{I})\boldsymbol{y} - \boldsymbol{b} = -\mathbf{M}\boldsymbol{y} - \boldsymbol{b}. \tag{4.3}$$

If $A(i,j;i,j) \neq T_x$, then M is nonsingular and (4.3) has a unique solution $\boldsymbol{y}_0 = -\mathbf{M}^{-1}\boldsymbol{b}$. If the forcing function $\boldsymbol{b}$ is identically equal to zero, then (2.20) can be re-written as

$$E = -\frac{1}{2}\boldsymbol{y}^T\mathbf{M}\boldsymbol{y} = -\frac{1}{2}(\mathbf{E}^T\boldsymbol{y})^T\Lambda(\mathbf{E}^T\boldsymbol{y}) = -\frac{1}{2}\sum_{k=1}^{N}\lambda_k(\bar{y}_k)^2. \tag{4.4}$$

where for the linear transformation $\mathbf{Q}^T$, $\mathbf{Q}^T\boldsymbol{y} \equiv \bar{\boldsymbol{y}} = [\bar{y}_1\ \bar{y}_2 \cdots \bar{y}_N]^T$. First, consider the case when $\lambda_k > 0$, $\forall k$. Then M is positive definite and $E \leq 0$ for all $\boldsymbol{y} \in \boldsymbol{D}^N$, where the equality holds only at the origin $\boldsymbol{y} = \mathbf{0}$. Thus the energy denoted by $E(\boldsymbol{y})$ is a concave function of $\boldsymbol{y}$ in $\boldsymbol{D}^N$, with the maximum value $E_{\max} = E(\mathbf{0}) = 0$. Since $|y_k| = 1$, $1 \leq k \leq N$, in the steady state, all corners of the N-dimensional hypercube are possible minima having the energy values of (4.4). From (4.4), the steady-state value of E depends only on the matrix M. Thus the global minimum is unique for a given cloning template $\mathbf{T}_A$. If $\boldsymbol{b} \neq \mathbf{0}$, the energy E is maximized at $\boldsymbol{y} = \boldsymbol{y}_0$ where $\boldsymbol{y}_0$ is the solution of the equation (4.3) and given by

$$\boldsymbol{y}_0 = -\mathbf{M}^{-1}\boldsymbol{b} = -\sum_{k=1}^{N}\frac{\boldsymbol{e}_k^T\boldsymbol{b}}{\lambda_k}\boldsymbol{e}_k. \tag{4.5}$$

Here, e_k is an orthonormal eigenvector of $\mathbf{M}$ associated with the eigenvalue λ_k, $1 \le k \le N$. By substituting (4.5) into (2.20), the maximum value of E is given by

$$E_{\max} = E(y_0) = \frac{1}{2} b^T \mathbf{M}^{-1} b = \frac{1}{2} \sum_{k=1}^{N} \frac{b_k^2}{\lambda_k}. \tag{4.6}$$

If the location of $E_{\max}$ is inside the hypercube, i.e., $y_0 \in D^N$, then all corners are possible minima as in the case when $b = \mathbf{0}$. However, if $y_0 \notin D^N$, some of the corners are not the equilibrium states and hence can not be reached in the steady state. Note that from (4.5), the number of stable equilibria depends on magnitudes of eigenvalues as well as the input and bias. Secondly, let us consider the case when some of the eigenvalues of $\mathbf{M}$ are negative while satisfying a condition,

$$\sum_{k=1}^{N} \lambda_k = \sum_{k=1}^{N} m_{k,k} = N\big(A(i,j;i,j) - T_x\big) > 0. \tag{4.7}$$

where $m_{i,j}$ is the (i,j)-th element of $\mathbf{M}$ with $m_{k,k} = A(i,j;i,j) - T_x = a_0 - T_x$. From the theory of linear algebra [110], the eigenvalues of $\mathbf{M}$ are inside a circle in the z-plane represented by

$$\left| z - m_{k,k} \right| \le \max_k \sum_{\substack{i=1 \\ i \ne k}}^{N} \left| m_{k,i} \right| = 4\big(|a_1| + |a_2|\big) \tag{4.8}$$

Here a_0, a_1, and a_2 are the elements of the cloning template as defined in (2.14). Thus, if $a_0 - T_x < 4(|a_1| + |a_2|)$, there may exist some negative eigenvalues. In this case, the matrix $\mathbf{M}$ is indefinite, and the energy function $E(y)$ is convex in some directions and is concave in others. The saddle-point occurs at the origin $y = \mathbf{0}$ when $b = \mathbf{0}$, and at y_0 from (4.5) when the forcing function is not zero. If $y_0 \in D^N$, the initial locations of the equilibria are not the corners of D^N. However, the indefinite $\mathbf{M}$ still guarantees the saturated outputs in the steady state, provided that it satisfies a condition that will be discussed in the next section. For the piecewise-linear neuron, the condition is given by $A(i,j;i,j) > T_x$ as in

(4.7). The above eigenvalue analysis, however, is valid only for the time interval $0 \le t \le t_1$, where t_1 is the time beyond which the network is unstable so that $v_{yij}(t) = +1$ or -1 for some i and j. When the matrix **M** is indefinite, the dynamics of the CNN is somewhat different. The neurons corresponding to the positions of positive eigenvalues are saturated first and a system with reduced dimension consisting of the sub-matrix $\mathbf{M}_S$ of **M** is produced. Because $\mathbf{M}_S$ has all positive elements on the main diagonal, it is again an indefinite or a positive definite matrix. If $\mathbf{M}_S$ is indefinite, the reduction of dimension is repeated. Finally, the stable equilibrium is reached once $\mathbf{M}_S$ is positive definite. The indefiniteness of **M**, however, does not imply that it takes more time to reach the steady state. For given input and bias, the magnitudes of eigenvalues of **M** determine the speed of convergence. Assume that if a neuron, say $C(p,q)$, is in saturation, the output value of +1 or -1 is maintained for $t \ge t_1$. Then the contribution of this specific neuron output to the neighborhood cells $C(k,l) \in N_r(p,q)$ for $t \ge t_1$ is simply $+A(k,l;p,q)$ or $-A(k,l;p,q)$ which can be regarded as another bias or input. Beyond this time point, the network is governed by the same equation as (2.20) with the cell $C(p,q)$ removed. The eigensystem $\mathbf{M}_s$ may vary as neurons become saturated. However, the initial $\mathbf{M}_s$, i.e., **M**, which is based on the linear network model through the constraint condition $|x_k(0)| \le 1$, $\forall k$, plays a major role in the operation of CNN.

4.2.2 Effects of Initial States

In addition to the characteristics of the energy function discussed above, the initial value of the energy which is determined by the initial state as well as given templates and the forcing function, plays an important role in the operation of the network. Given input u and initial state $x(0)$, the output vector y moves toward the closest stable equilibrium in the boundary $v_y \in D^N$ and stays there. First, note that the initial and final values of the energy function are given by

$$E(0) = -\frac{1}{2}x(0)^T[\mathbf{A} - T_x\mathbf{I}]x(0) - x(0)^T[\mathbf{B}u - I_b w] \tag{4.9}$$

and

$$E(+\infty) = -\frac{1}{2}y_\infty^T\mathbf{M}y_\infty - y_\infty^T b, \tag{4.10}$$

respectively, where y_∞ is the steady-state output. From the constraint condition $|v_{xij}(0)| \le 1, \forall i,j$, the relationship $y(0) = x(0)$ is used in (4.9) for the neuron with a piecewise linear transfer function. Because the network always operates in a direction so as to decrease the energy as time elapses, the following inequality holds,

$$E(0) \ge E(t) \ge E(+\infty) \qquad \forall t > 0. \tag{4.11}$$

Given $E(0)$, the trajectory of the network output is confined to the basin of attraction in D^N, which may consist of many disjoint units. The state in one unit can not reach others because there exist energy barriers among them. So the neuron is frozen to the state determined by the initial value $x(0)$. This local minima problem was observed [59] in applications where the global minimization of E is not necessarily required.

Let N_0, N_1, and N_2 denote the number of corners of D^N, the number of possible equilibria in D^N, and the number of minima for which the inequality (4.11) holds, respectively. Then, the relationship $N_2 \le N_1 \le N_0$ and $N_0 = Q^N$ hold in general, where Q is the number of levels that the neuron can have in the steady state. For binary neurons, $Q = 2$. The objectives of the proposed annealing method are to maximize the number N_2 as close to N_1 as possible and to minimize the energy E for a given system of (2.17). In some applications such as content addressable memory, the maximization of N_1 by a proper choice of the templates may be desirable for increasing the capacity.

4.3 Hardware-annealed CNN

The hardware annealing is performed by controlling the gain of the neuron $g(t)$, which is assumed to be the same for all neurons throughout the network for simplicity of analysis and illustration. The initial gain at time $t = 0$ can be set to an arbitrarily small, positive value such that $0 \le g_0 << 1$, and after the annealing process for t_A period of time the final gain $g(t_A) = 1$ is maintained until the next operation. When the hardware annealing is applied to a CNN by increasing the neuron gain $g(t)$, the transfer function can be described by

$$v_{yij}(t) = \begin{cases} +1 & v'_{xij}(t) \ge 1 \\ v'_{xij}(t) & -1 < v'_{xij}(t) < 1 \\ -1 & v'_{xij}(t) \le -1 \end{cases} \tag{4.12}$$

where $v'_{xij}(t) = g(t)v_{xij}(t)$. Figure 4-2 shows the transfer characteristics of the piecewise nonlinearity for several gain values. Note that the saturation level is still $y = +1$ or -1 and only the slope of $f(x)$ around $x = 0$ varies. After the state is initialized, i.e., $x = x(0)$, $g(t)$ increases linearly from $g_{\min} = g_0$ to $g_{\max} = 1$ for $0 \le t \le T_A$. Then, the maximum gain is maintained during $T_A < t \le T$, during which the network is stabilized. Figure 4-3 shows the block diagram of a neuron cell for use in the annealed CNN. An analog multiplier is placed between the summing and nonlinear circuits. Because $g > 0$, a multiplier with two-quadrant operation may suffice.

4.3.1 Dynamical Behavior of Annealed CNN

The dynamics of a CNN is described by a set of the nonlinear differential equations,

$$C\frac{dv_{xij}(t)}{dt} = -\frac{1}{R_x}v_{xij}(t) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{ukl} + I_b$$

$$1 \le i \le n,\ 1 \le j \le m \tag{4.13}$$

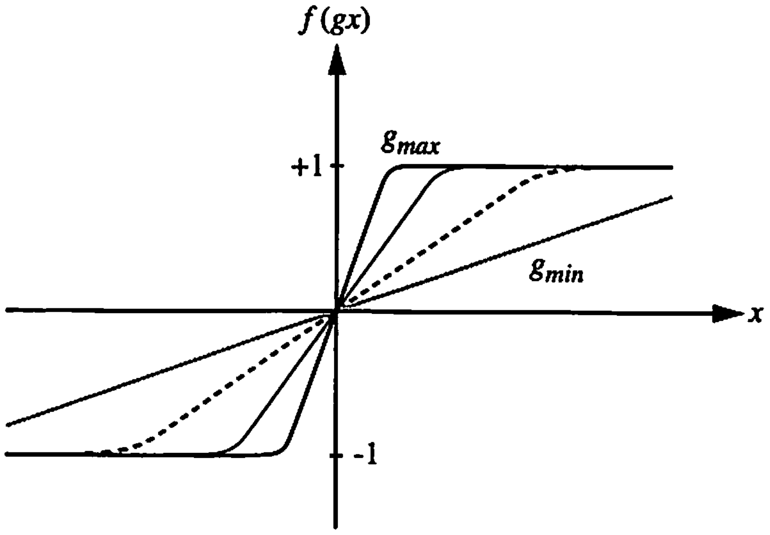


Figure 4-2: Transfer characteristics of variable-gain piecewise-linear function.

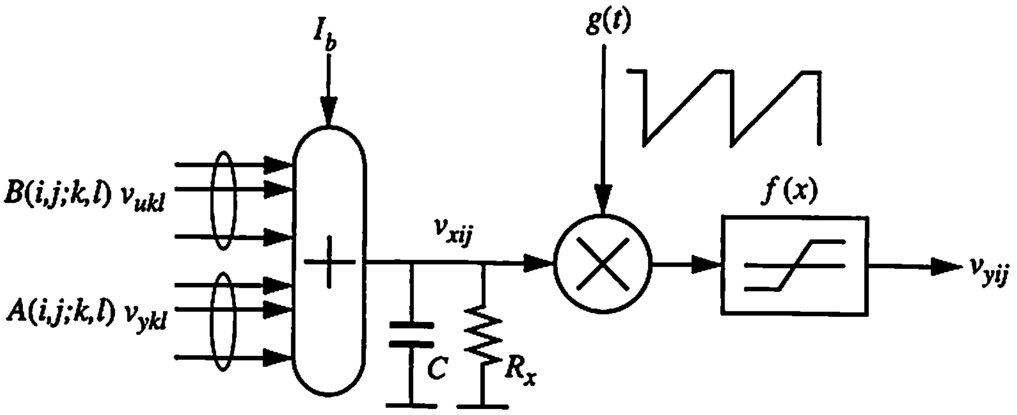


Figure 4-3: Block diagram of variable-gain neuron cell with two-quadrant analog multiplier.

where $v_{yij} = f(gv_{xij})$ and C is the equivalent capacitance at the state node of each neuron. In a compact form,

$$C\dot{x} = -T_x x + \mathrm{A}y + b, \tag{4.14}$$

where $y = f(gx)$ and $b = \mathrm{B}u + I_b w$. When the neuron gain is not equal to one, the Lyapunov function (2.17) can be written as

$$\begin{aligned} E = &-\frac{1}{2}\sum_{i,j}\sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)v_{yij}(t)v_{ykl}(t) + \frac{1}{2gR_x}\sum_{i,j}\left(v_{yij}(t)\right)^2 \\ &-\sum_{i,j}\sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{yij}(t)v_{ukl} - \sum_{i,j} I_b v_{yij}(t). \end{aligned} \tag{4.15}$$

where $v_{yij} = f(gv_{xij})$. For the sigmoid nonlinearity, the second term of (4.15) can be expressed as

$$\frac{1}{gR_x}\sum_{i,j}\int_0^{v_{yij}(t)} f_{ij}^{-1}(v)dv, \tag{4.16}$$

which can be used for arbitrary nonlinearity $v_{yij}(t) = f(v_{xij}(t))$ if its inverse function $gv_{xij}(t) = f^{-1}(v_{yij}(t))$ can be defined. As stated in Chapter II, the piecewise-linear function used in (4.15) is a special case of this general expression. For variable neuron-gain, this is still the case. For the piecewise-linear function, $x = f^{-1}(y) = y/g$, $-1 \leq y \leq 1$, and

$$\int_0^{v_{yij}(t)} f^{-1}(y)dy = \frac{1}{g}\int_0^{v_{yij}(t)} ydy = \frac{1}{2g}\left(v_{yij}(t)\right)^2, \tag{4.17}$$

which is consistent with (4.15). If $g = 1$ then it is the same as the one in the equation (2.17). By using vector and matrix notations, (4.15) can be re-written as

$$E = -\frac{1}{2}y^T \mathrm{A}y + \frac{T_x}{2g}y^T y - y^T b. \tag{4.18}$$

The process of global optimization can be understood by observing the eigenvalues of the time-varying system matrix of a CNN during the annealing. When the neuron gain g is decreased to its small initial value, the corresponding output is simply given by $y = gx$. For use with arbitrary initial condition instead of the constraint condition $|x(0)| \le 1$, we assume that $g_0 < 1/|x_{\max}|$, where $|x_{\max}| \ge 1$ is the maximum magnitude of neuron state, such that all neurons are in the linear region. For other nonlinearities, we also can assume that the system of (4.13) is linear near the operating regions. Note that in (4.13) $v_{yij} = f(gv_{xij}) \ne gf(v_{xij})$ is true in general. However, if $|v_{xij}| < 1/g$, then $f(gv_{xij}) = gv_{xij}$ and the gain control function $g(t)$ can be combined with the weight matrix $\mathbf{A}$ in (2.15) to produce a time-varying matrix $\mathbf{M}_g = \mathbf{A} - (T_x/g(t))\mathbf{I}$ and the equation (4.14) can be written as

$$C\dot{x} = (g\mathbf{A} - T_x\mathbf{I})x + \mathbf{B}u + I_b w = g\mathbf{M}_g x + b. \tag{4.19}$$

Since the matrix $\mathbf{M}_g$, $0 < g \le 1$, commutes with $\mathbf{M}$ such that $\mathbf{M}_g\mathbf{M} = \mathbf{M}\mathbf{M}_g$, it shares the same eigenvectors with $\mathbf{M}$ and

$$\begin{aligned}\mathbf{M}_g &= \mathbf{A} - \frac{T_x}{g}\mathbf{I} = (\mathbf{A} - T_x\mathbf{I}) + \frac{(g-1)T_x}{g}\mathbf{I} \\ &= \mathbf{Q}\Lambda\mathbf{Q}^T + \frac{(g-1)T_x}{g}\mathbf{Q}\mathbf{Q}^T = \mathbf{Q}\left[\Lambda + \frac{(g-1)T_x}{g}\mathbf{I}\right]\mathbf{Q}^T.\end{aligned} \tag{4.20}$$

Therefore, the eigenvalues of $\mathbf{M}_g$ are given by

$$\lambda_k = \lambda_k^1 + \frac{g-1}{g}T_x, \qquad k = 1,2,\cdots,N, \tag{4.21}$$

where λ_k^1 is the k-th eigenvalue when the neuron gain is 1. During the annealing process, each eigenvalue starts at the initial value $\lambda_k^1 + (g_0 - 1)T_x/g_0$ and increases toward the value λ_k^1 until one or more neurons are saturated so that the eigensystem (4.19) is modified. If $g_0 = 0$, then all eigenvalues begin at $-\infty$. Initially, all eigenvalues of (4.21) are negative so that the matrix $-\mathbf{M}_g$ is positive definite. The energy E decreases along the trajectory of y in (4.18) toward the minimum point y_0' which is the solution of the equation $\nabla_y E = \mathbf{0}$ where

$$\nabla_y E = \frac{\partial E}{\partial y} = -\mathbf{A}y + \frac{T_x}{g}y - b = -\mathbf{M}_g y - b. \tag{4.22}$$

For a small g_0, the initial value of y at $t = 0$, $y(0)$, and the solution of (4.22)

$$\begin{aligned} y_0' &= -\mathbf{M}_g^{-1} b \\ &= -\mathbf{Q}\left[diag\left(\frac{1}{(\lambda_k^1 + T_x)g - T_x}, k = 1,2,\cdots,N \right) \right] \mathbf{Q}^T b \end{aligned} \tag{4.23}$$

are both close to the origin as verified by

$$\lim_{g_0 \to 0} \|y_0'\| = \lim_{g_0 \to 0} \|y(0)\| = 0. \tag{4.24}$$

Therefore, the output variable y quickly approaches to the equilibrium y_0' at which the energy value is given by

$$E(y_0') = \frac{1}{2} b^T \mathbf{M}_g^{-1} b. \tag{4.25}$$

Once $y = y_0'$, the trajectory of y follows the minimum y_0' as the neuron gain g increases, because it is the only stable equilibrium while the matrix $-\mathbf{M}_g$ is positive definite. Let λ_k, $k = 1,2,\cdots,N$, be the maximum eigenvalue of $\mathbf{M}_g$ with possible multiplicity of n. As the gain increases further, λ_k becomes positive, making the equilibrium y_0' unstable. Beyond this time point, the output begins to move from the unstable equilibrium y_0' to one of the stable equilibria that have been newly developed as $-\mathbf{M}_g$ becomes indefinite. Note that as the gain increases, the equilibrium y_0' moves toward the point y_0 of (4.5), at which the energy of the network with $g = 1$ is maximized. By recalling that E is a quadratic function of y, the direction of the movement is the same as the direction to the local minimum and y_0' moves away of the global minimum. As λ_k approaches zero, the speed at which the corresponding output y_k follows the equilibrium y_0' also approaches zero. In the linear operating region, $y = gx$ and

$$\nabla_y E = -\mathrm{A}y + \frac{T_x}{g} y - b = -\mathbf{M}_g x - b = -C\frac{dx}{dt}. \tag{4.26}$$

Substituting (4.23) and $y = y_0' + \varepsilon$ where $\|y - y_0'\| \le \varepsilon$ for a small positive ε into (4.26), we can obtain

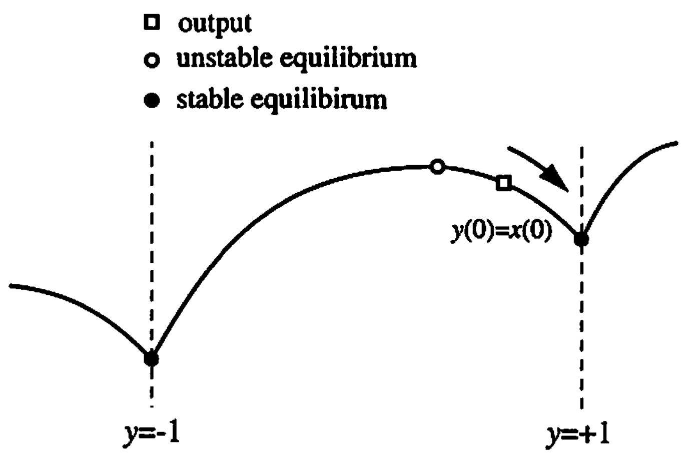
$$\begin{aligned}\nabla_y E &= -\mathbf{M}_g(y_0' + \varepsilon) - b = -\mathbf{M}_g\left(-\mathbf{M}_g^{-1} b + \varepsilon\right) - b \\ &= -\mathbf{M}_g \varepsilon.\end{aligned} \tag{4.27}$$

Therefore, for small λ_k, $dy_k/dt = g(dx_k/dt)$ is also small. On the other hand, the speed of movement of y_0' is determined by $g(t)$. Therefore, at the time when $-\mathbf{M}_g$ becomes indefinite, y is in the opposite side of the local minimum around y_0', and as λ_k becomes positive, y now begins to move to the global minimum. As the other eigenvalues become positive with further gain increase, the same procedures may take place. Figure 4-4 (a) shows the one-dimensional plot of the energy function in an unannealed CNN. For the given initial state, $y = +1$ is the stable equilibrium that can be reachable in the steady state. The annealing procedure described above is graphically shown in Figure 4-4 (b).

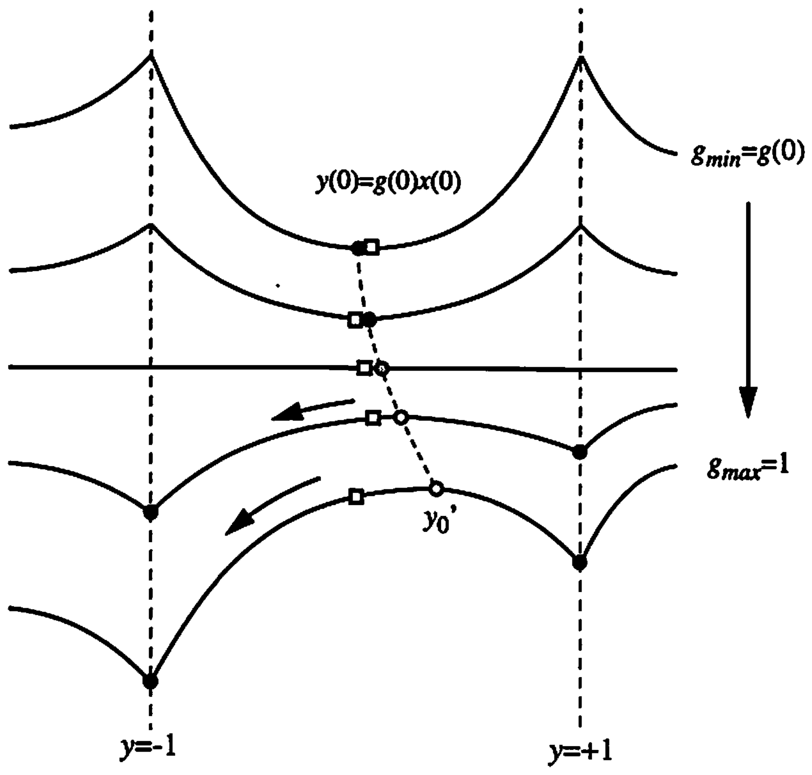
The argument of eigenvalues in describing the annealing procedure can be verified through the time-domain analysis. Although $\mathbf{M}_g$ is a function of time through $g = g(t)$, the equivalence transformation $\mathbf{Q}^{-1} = \mathbf{Q}^T$ is time-invariant. Therefore, the dynamical equation (4.14) is equivalent to

$$\dot{z} = g\Lambda_g z + \mathbf{Q}^T b, \tag{4.28}$$

where $z = \mathbf{Q}^T x$ and $\Lambda_g = \mathbf{Q}^T \mathbf{M}_g \mathbf{Q}$. Without loss of generality, it is assumed that $C = 1$ in (4.28). Then the diagonal matrix Λ_g allows us to find the state transition matrix [111] of (4.28) during $0 \le t \le T_A$,

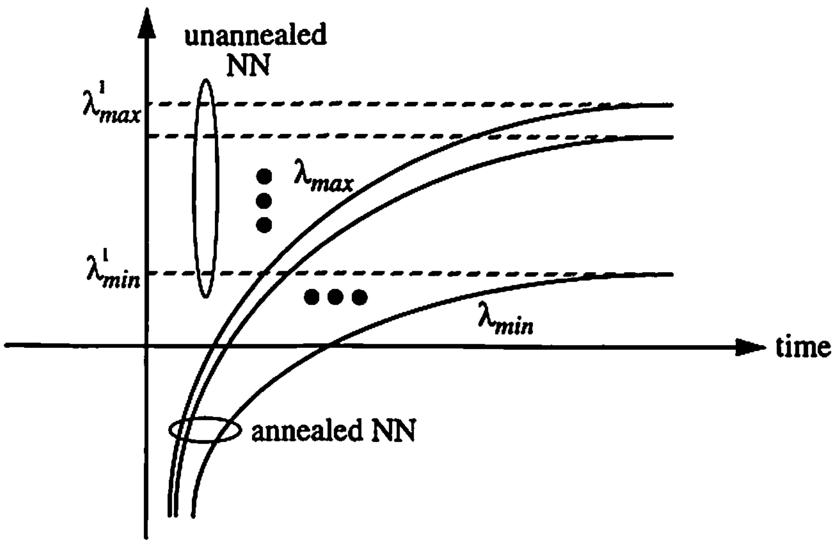


(a) Unannealed CNN: Steady-state output y=+1 (local minimum).

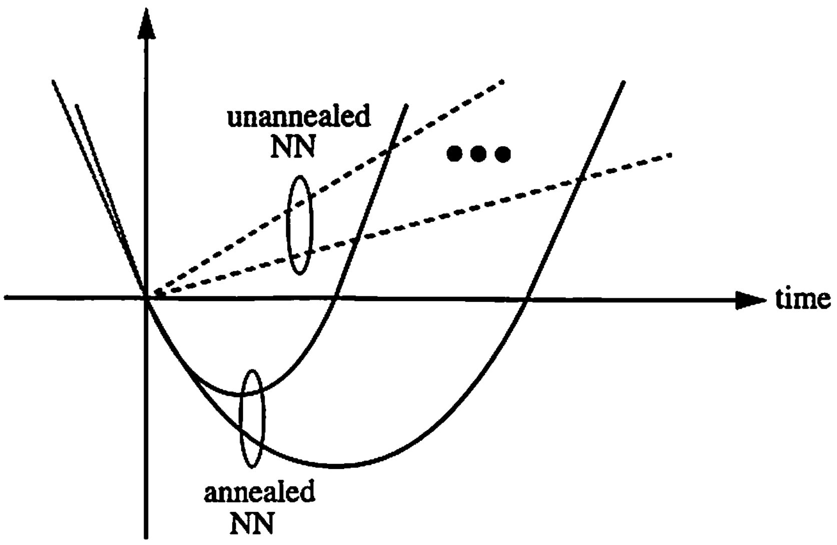


(b) Dynamic process of finding global minimum: Change of *E* from convex function to concave function.

Figure 4-4: Energy functions of unannealed and annealed CNNs.



(a) Eigenvalues of **M** and $\mathbf{M_g}$.



(b) Exponents of state transition matrix $\Phi(t,0)$.

Figure 4-5: Eigenvalues of $\mathbf{M_g}$ and exponents of state transition matrix $\Phi(t,0)$ (for eigenlavues $\lambda^1_k > -T_x$).

$$\Phi(t,t_0) = \exp\left[\int_{t_0}^{t} g(\tau)\Lambda(g(\tau))d\tau\right]$$
$$= diag\left[\exp\left\{\int_{t_0}^{t}\left[(\lambda_k^1 + T_x)g(\tau) - T_x\right]d\tau\right\}, k = 1,2,\cdots,N\right], \tag{4.29}$$

where $t_0 = 0$. If $g(t) = 1$ for $t \geq 0$, then $\Phi(t,0) = e^{\Lambda t} = diag[e^{\lambda_k^1 t},\ k = 1,2,\cdots,N]$ as expected. The difference between the dynamics of an annealed CNN and that of a unannealed network can be understood by comparing the exponents $\lambda_k^1 t$ for the unannealed CNN and those in (4.29) for the annealed network. For a linearly increasing cooling schedule $g(t) = \alpha t$, where $\alpha = 1/T_A$ is a constant, the exponent is a quadratic function of time as can be seen from

$$\int_0^t \left[(\lambda_k^1 + T_x)g(\tau) - T_x\right]d\tau = \frac{\lambda_k^1 + T_x}{2T_A}t^2 - T_x t, \quad k = 1,2,\cdots,N. \tag{4.30}$$

Notice that each exponent is negative for $0 \leq t \leq 2T_x T_A/(\lambda_k^1 + T_x)$ during which the output y approaches and follows the stable equilibrium y_0'. In Figure 4-5, the eigenvalues and exponents of the state transition matrix (4.29) for the unannealed and annealed CNN are plotted as a function of time or annealing gain. Note that once one or more neuron is saturated, the dynamics remains the same but with different set of values due to reduced system dimension.

4.3.2 Stability of Annealed Network

Now let us consider the stability of the annealed CNN by checking the behavior of $E(t)$ for $t \geq 0$. By using the chain rule,

$$\frac{dE}{dt} = \frac{\partial E}{\partial y}\cdot\frac{dy}{dt} + \frac{\partial E}{\partial g}\frac{dg}{dt} \tag{4.31}$$

where the symbol $\cdot$ denotes the scalar product and for a continuously increasing neuron gain, $dg/dt > 0$. Substituting (4.26) into (4.31), we can obtain

$$\frac{dE}{dt} = -\left(\mathrm{A}y - \frac{T_x}{g}y + b\right)^T \frac{dy}{dt} - \frac{T_x}{2g^2}\frac{dg}{dt}y^T y \tag{4.32}$$

The second term in (4.32) is a simple quadratic function and is a non-positive quantity. The first term which is a scalar product of two vectors, can be expressed as a sum of N terms. If $|y_l| = 1$ for some l, then $dy_l/dt = 0$ and the corresponding values in the summation vanish. If we consider only nonzero terms $y_k = gx_k < 1$, $k \neq l$, then we have

$$\frac{dE}{dt} = -C\sum_k \left(\frac{dx_k}{dt}\frac{dy_k}{dt}\right) - \frac{T_x g'}{2g^2}\sum_{k=1}^{N}(y_k)^2 . \tag{4.33}$$

Recall that in an unannealed CNN with $g = 1, \forall t \geq 0$, the second term is zero and (4.33) is simplified to (2.24). However, for an annealed CNN the gain control function g is also a function of time and

$$\frac{dy}{dt} = \frac{\partial y}{\partial x}\frac{dx}{dt} + \frac{\partial y}{\partial g}\frac{dg}{dt} = g\frac{dx}{dt} + g'x \tag{4.34}$$

where the second equality holds only if $y < 1$. By defining t_C as the time when $g = g_C$, the proof of stability of annealed CNN's may consist of two steps.

(A) $0 \leq t < t_C$: During this interval, M_g is negative definite and the system is asymptotically stable [111] with the output y approaching the stable equilibrium y_0' given in (4.23). Assume that the annealing gain g increases slowly such that $y = y_0'$. Then, from (4.25)

$$E(y_0') = \frac{1}{2}b^T \mathrm{M}_g^{-1} b \leq 0, \qquad \forall b \in R^N . \tag{4.35}$$

(B) $t_C \leq t < +\infty$: From the equations (4.14), (4.33), and (4.34),

$$\begin{aligned}\frac{dE}{dt} &= -\left(\mathrm{A}y - \frac{T_x}{g}y + b\right)^T \left(g(\mathrm{A}y - T_x x + b) + g'x\right) - \frac{T_x g'}{2g^2} y^T y \\ &= -g\|\mathrm{A}y - T_x x + b\|^2 - \frac{g'}{g}\left(y^T \mathrm{A}y - \frac{T_x}{2g} y^T y + y^T b\right) \\ &= -g\|\mathrm{A}y - T_x x + b\|^2 + \frac{g'}{g}E - \frac{g'}{2g} y^T \mathrm{A}y. \end{aligned} \tag{4.36}$$

For $g > 0$ and $g' = dg/dt \geq 0$, the first and third terms are not positive. Thus if $E \leq 0$, $\forall t \geq 0$, then $dE/dt \leq 0$, $\forall t \geq 0$. Initially, the positive definiteness of $-\mathrm{M}_g$ guarantees $y = y_0'$, at which $E \leq 0$ as shown in (4.35). If we use other neuron transfer characteristics $y = f(gx)$, for which the inverse function $x = (1/g)f^{-1}(y)$ is well defined in the range of x, (4.18) can be written as

$$E = -\frac{1}{2}y^T \mathrm{A}y + \frac{T_x}{g}\kappa^T w - y^T b \tag{4.37}$$

where $\kappa = [\kappa_1 \ \kappa_2 \ \cdots \ \kappa_N]^T$, $\kappa_i = \int_0^{y_i} f^{-1}(v)dv$. In this case, $f^{-1}(y) = gx$ and

$$\nabla_y E = -\mathrm{A}y + \frac{T_x}{g}f^{-1}(y) - b = -\mathrm{A}y + T_x x - b = -C\frac{dx}{dt} \tag{4.38}$$

from which it follows that

$$\frac{dE}{dt} = \nabla_y E \cdot \frac{dy}{dt} = -C\frac{dx}{dt}\cdot\frac{dy}{dt} = -C\sum_{k=1}^{N}\frac{\partial f}{\partial x_k}\left(\frac{dx_k}{dt}\right)^2 \leq 0. \tag{4.39}$$

Despite of the time-varying nature of the hardware annealing, the stability of the network is still maintained as long as the gain control function $g = g(t)$ are non-negative.

4.3.3 Critical Neuron Gains and Annealing Schedule

Two gain values are involved in the annealing operation. First, the critical gain g_C defined in the section 2.3.2 represents the value at which the phase of search for an optimal solution is completed. In relation to this critical gain, two conditions must be satisfied;

1. the initial gain g_0 is a small positive number less than g_C, and
2. an enough amount of time is allowed for the network state to reach the basin of attraction to which the global minimum of E belongs, before the gain reaches g_C.

From (4.21), the critical gain is given as in (2.42) by

$$g_C = \frac{T_x}{\lambda^1_{\max} + T_x}, \tag{4.40}$$

where $\lambda^1_{\max}$ denotes the maximum eigenvalue of E at $g = 1$. Since the value $\lambda^1_{\max}$ in (4.40) is difficult to compute in general, the upper bound can be used instead. From (4.8),

$$g_C \approx \frac{T_x}{\lambda^U_{\max} + T_x} = \frac{T_x}{\max_k\left(\sum_{i=1}^N |m_{k,i}|\right) + T_x}, \tag{4.41}$$

where $m_{i,j}$ is the (i,j)-th element of M. For a CNN with cloning templates (2.14) and $r = 1$, $g_C \approx T_x / \left(a_0 + 4(|a_1| + |a_2|)\right)$.

One or more eigenvalues with positive real parts is a sufficient condition for the system instability in the sense of bounded-input/bounded-output (BIBO). Therefore, for the condition $g > g_C$, at least one neuron output is saturated as time elapses. However, it does not guarantee binary-valued outputs for all neurons in the steady state. As much the same way as the critical gain g_C is determined, a saturation gain g_S must be defined as the annealing gain beyond which all saturated binary outputs are guaranteed. Note that for $g > g_C$, the linearized system model is no longer valid and the network operation is dependent on neuron nonlinearity model. Therefore, one may introduce the Jacobian of the system (4.19). If we rewrite (4.19) as $\dot{x} = C\mathrm{X}(x)$ where $\mathrm{X}(x) = -T_x x + \mathrm{A} f(gx) + b$, then the Jacobian matrix of $\mathrm{X}(x)$ is given by

$$\mathbf{J}_{\mathrm{X}}(x) = \mathbf{A}\,\mathbf{J}_{\mathrm{F}}(x) - T_x \mathbf{I}, \tag{4.42}$$

where $\mathbf{J}_{\mathrm{F}}(x)$ is the Jacobian matrix of $f(gx)$ defined as

$$\mathbf{J}_{\mathrm{F}}(x) = diag\left[\frac{\partial f(gx)}{\partial x_k}, k = 1,2,\cdots,N\right].$$

When the neurons with the piecewise-linear transfer function are used, $v_y = f(gv_x) = gv_x$, $\mathbf{J}_{\mathrm{F}}(x) = g\mathbf{I}$, and $\mathbf{J}_{\mathrm{X}}(x) = g\mathrm{A} - T_x\mathbf{I}$. Note that the Jacobian matrix $\mathbf{J}_{\mathrm{X}}(x)$ is not symmetric in general because for $i \neq j$, $\partial f(gx)/\partial x_i \neq \partial f(gx)/\partial x_j$ and $(\mathbf{A}\mathbf{J}_{\mathrm{F}}(x))^T = \mathbf{J}_{\mathrm{F}}(x)\mathrm{A} \neq \mathbf{A}\mathbf{J}_{\mathrm{F}}(x)$. Let $x_{\min}$ be the minimum value of the state such that $y_{\min} = f(x_{\min})$ is the minimum neuron output for logic +1, and let $\Delta^N = \{y \in R^N : |y_k| < f(x_{\min}), k = 1,2,\cdots,N\}$. To ensure the instability of the system in Δ^N, the Jacobian matrix $\mathbf{J}_{\mathrm{X}}(x)$ must have at least one eigenvalue with positive real part.

The system of (4.19) can be examined in a piecewise-linear fashion. Assume that $q = N - p$, $1 \leq q \leq N-1$, neurons become saturated at time $t = t_0 > t_C$ so that the corresponding outputs are binary-valued. Then for $t \geq t_0$, we can define a $p \times p$ submatrix $\mathbf{J}_{p\times p}$ of $\mathbf{J}_{\mathrm{X}}(x) \equiv \mathbf{J}_{N\times N}$ as the Jacobian of a p-dimensional system constructed by discarding the rows and columns that correspond to the neurons with saturated outputs. The Jacobian $\mathbf{J}_{p\times p}$ must be either indefinite or positive (semi-)definite. There may exist stable equilibrium in Δ^p, otherwise. With repeated applications of dimensionality reduction, the final neuron or neurons will be saturated in the steady state if the corresponding Jacobian is positive definite. Therefore, to ensure all binary-valued outputs in the steady state, the Jacobian $\mathbf{J}_{k\times k}$, $k = 1,2,\cdots,N$, can not be negative (semi-)definite. Since the network dynamics depends on the initial state and the external forcing function as well, it is difficult to anticipate the order in which the neurons are saturated. Thus, a sufficient condition can be given as all positive values along the main diagonal of $\mathbf{J}_{\mathbf{X}}(x)$,

$$A(i,j;i,j)\frac{\partial f(gx)}{\partial x_k} - T_x > 0 \qquad k = 1,2,\cdots,N. \tag{4.43}$$

If we define

$$\eta = \min_k\left(\frac{\partial f(gx)}{\partial x_k}\right) \tag{4.44}$$

for $y = f(gx) \in \Delta^N$ then (4.43) is satisfied if $\eta > T_x/A(i,j;i,j)$. For the piecewise-linear transfer function, $x_{\min} = 1/g$ and $\eta = g$. In this case, the saturation gain g_S is simply

$$g_S = \frac{T_x}{A(i,j;i,j)}. \tag{4.45}$$

For the sigmoid function, $\eta = df(x)/dx\big|_{x=x_{\min}}$ and $y_{\min} = f(x_{\min}) = f_1(g\cdot x_{\min})$, where $f_1(x) = f(x)\big|_{g=1}$. Therefore,

$$\eta = \frac{df_1(g\cdot x)}{dx}\bigg|_{x=x_{\min}} = g\frac{df_1(w)}{dw}\bigg|_{w=f_1^{-1}(y_{\min})} > \frac{T_x}{A(i,j;i,j)} \tag{4.46}$$

and

$$g_S = \frac{T_x}{A(i,j;i,j)}\left(\frac{df_1(w)}{dw}\bigg|_{w=f_1^{-1}(y_m)}\right)^{-1}. \tag{4.47}$$

If we choose $y_{\min} = f_1(1) = 0.7616$, then $g_S = 2.381\cdot T_x/A(i,j;i,j)$.

In summary, the hardware annealing may consist of the following operation phases;

1. Search phase ($g_0 \le g < g_C$) : The annealing process forces the network state to move to the basin of attraction to which the global minimum of E belongs, i.e., $y(g_0) \in D^N \to y(g_C) \in \Omega^N$ where Ω^N is an N-dimensional manifold.

2. Attraction phase ($g_C \le g < g_S$) : Once $y(g_C) \in \boldsymbol{\Omega}^N$, a further gain increase makes the output approach the global minimum rapidly. In addition, the energy barriers begin to stand out so that $\boldsymbol{\Omega}^N$ becomes disjoint from others and occasional jumps to local minima can be prevented.
3. Completion phase ($g_S \le g \le 1$) : In this interval, the formation of the original energy landscape is completed and all saturated binary outputs are obtained.

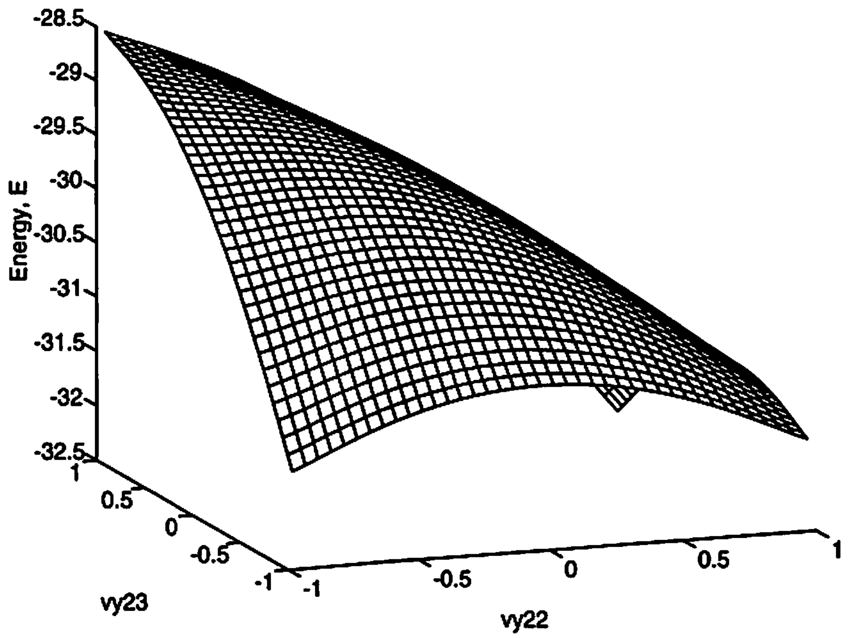
Thus, the annealing schedule must be chosen such that $g_{\min} = g_0 < g_C = g(t_C)$ and the time period t_C for search phase is long enough. The qualitative analysis of the value t_C for the successful search phase, is subject to further investigation in the future.

4.4 Simulation Results

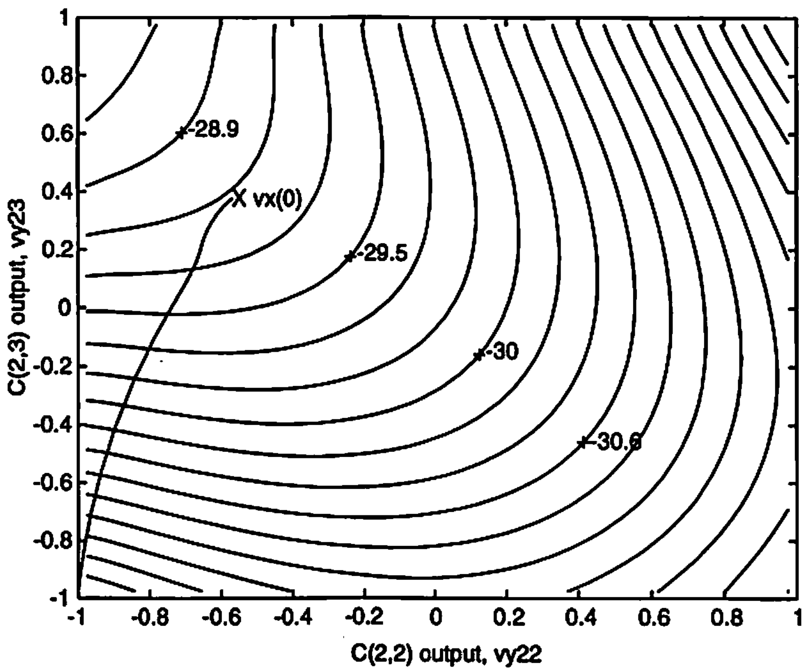
Figure 4-6 (a) shows the energy landscape of a 4×4 CNN in the steady state as the functions of v_{y22} and v_{y23}. The CNN uses the bipolar sigmoid function as the neuron nonlinearity and has the parameters: $R_x = 10^3\Omega$, $I = 0$, and the cloning templates of

$$\mathbf{T}_A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ and } \mathbf{T}_B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.48}$$

For these parameters, the matrix **M** has six negative eigenvalues and each of three distinct eigenvalues has the multiplicity of 2, as given in Table 4.1. In the figure, all neurons other than $C(2,2)$ and $C(2,3)$ have fixed, steady-state output values determined by the network operation. Notice that, because $y_0 \notin D^{16}$ where y_0 is given by (4.5), there are three minima at the locations $[v_{y22}, v_{y23}] = [-1,-1]$, $[+1,-1]$, and $[+1,+1]$. Depending on the value of $v_x(0)$ and v_u, the output $[v_{y22}, v_{y23}]$ will be attracted to one of vertices $[-1,-1]$, $[+1,-1]$, and $[+1,+1]$ in the steady state. The points $[v_{y22}, v_{y23}] = [-1,-1]$ and $[+1,-1]$ are the local minima and $[v_{y22}, v_{y23}] = [+1,+1]$ is the global minimum with the



(a) Concave energy landscape for neuron outputs [vy22,vy23].



(b) Trajectory of [vy22,vy23] from initial value (X) to local minimum.

Figure 4-6: Local minima problem in CNN.

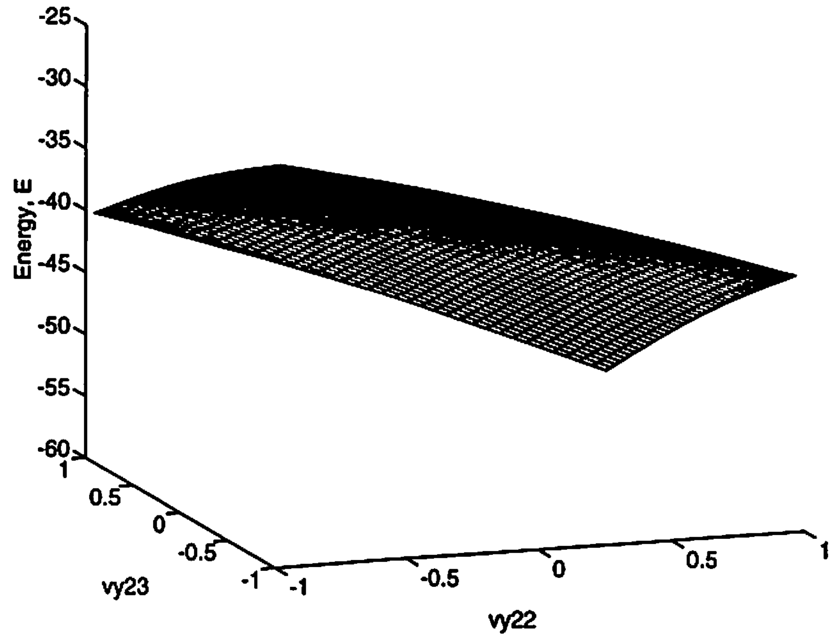
Table 4.1 List of eigenvalues of indefinite **M** for cloning templates of (4.44).

Eigenvalues	Magnitude	Multiplicity
λ_1, λ_2	+ 0.618	2
λ_3, λ_4	+ 1.000	2
λ_5, λ_6	- 0.2361	2
λ_7, λ_8	- 1.0000	2
λ_9, λ_{10}	- 1.6180	2
$\lambda_{11}, \lambda_{12}$	+ 4.2361	2
λ_{13}	+ 0.3820	1
λ_{14}	+ 0.1459	1
λ_{15}	+ 2.6180	1
λ_{16}	+ 6.8541	1
$\sum_{k=1}^{16} \lambda_k$	+ 16.00	

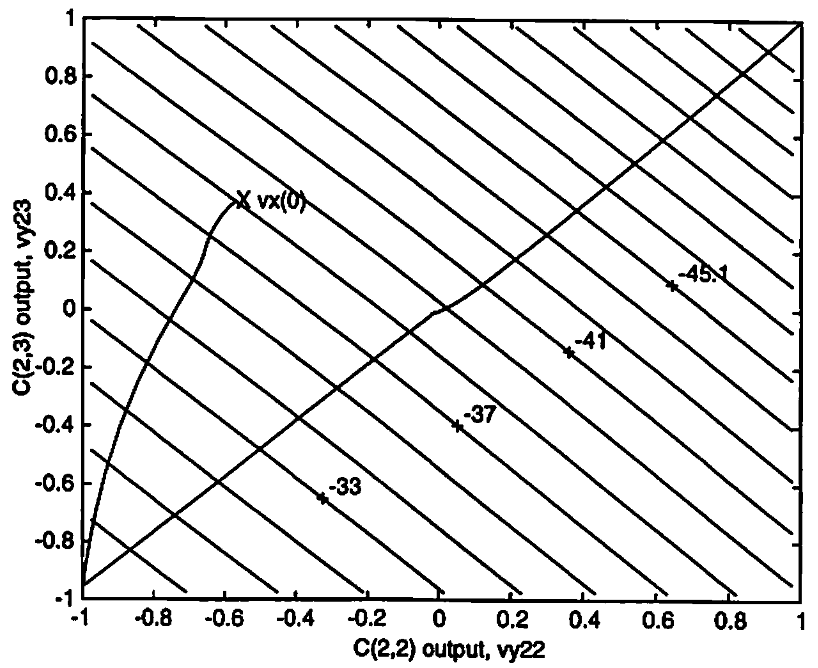
lowest energy value. Figure 4-6 (b) shows the contours of the energy function and the trajectory of the output values during the network operation. The figure shows that the initial output values $[v_{y22}(0), v_{y23}(0)]$ indicated by X is attracted to the point $[-1,-1]$, which is one of the local minima.

The hardware annealing is performed by controlling the neuron gain $g(t)$, which is assumed to be the same for all neurons throughout the network. The initial gain at time $t=0$ can be set to an appropriately small, positive value such that $0 \leq g(0) << 1$. After the annealing process for t_A period of time, the final gain $g(t_A)=1$ is maintained until the next operation. Figure 4-7 shows the results of hardware annealing for the same network parameters including the initial state as in Figure 4-6, and $g(t)=\alpha t$ where α is a constant. In Figure 4-7 (a), the local minima do not exist and the output [+1,+1] which was the global minimum before annealing, became the only minimum that can be reached by annealing. However, this does not mean that the local minima were removed by the hardware annealing. As a matter of fact, the local minima still exist at the locations they do all the time, and as the result of the annealing process two-dimensional subspace D^2 spanned by $[v_{y22}, v_{y23}]$ moved to different location in D^{16} at which the global minimization of E can be achieved. In Figure 4-8, $E(t)$ is plotted for unannealed and annealed operations. The annealing operation is activated at time $t=1$ (normalized) with the initial state $x(1^+)$ equal to the final state of the unannealed operation. Note that for all k, $1 \leq |x_k(1^+)| \leq v_{x,\max} \approx R_x \sum_{i,j=1}^{3} |T_A(i,j)| + |v_u| = 11$. For comparison purpose, the dynamics of $E(t)$ is shown for three different cooling schedules

$$g(t)=\begin{cases} \alpha_1 t^2 & ;\ \text{schedule A} \\ \alpha_2 t & ;\ \text{schedule B} \\ \alpha_3 \sqrt{t} & ;\ \text{schedule C} \end{cases} \tag{4.49}$$



(a) Energy surface for neuron outputs [vy22,vy23] after annealing.



(b) Trajectory of [vy22,vy23] from local [-1,-1] to global minimum [+1,+1].

Figure 4-7: Result of hardware annealing for Figure 4-6.

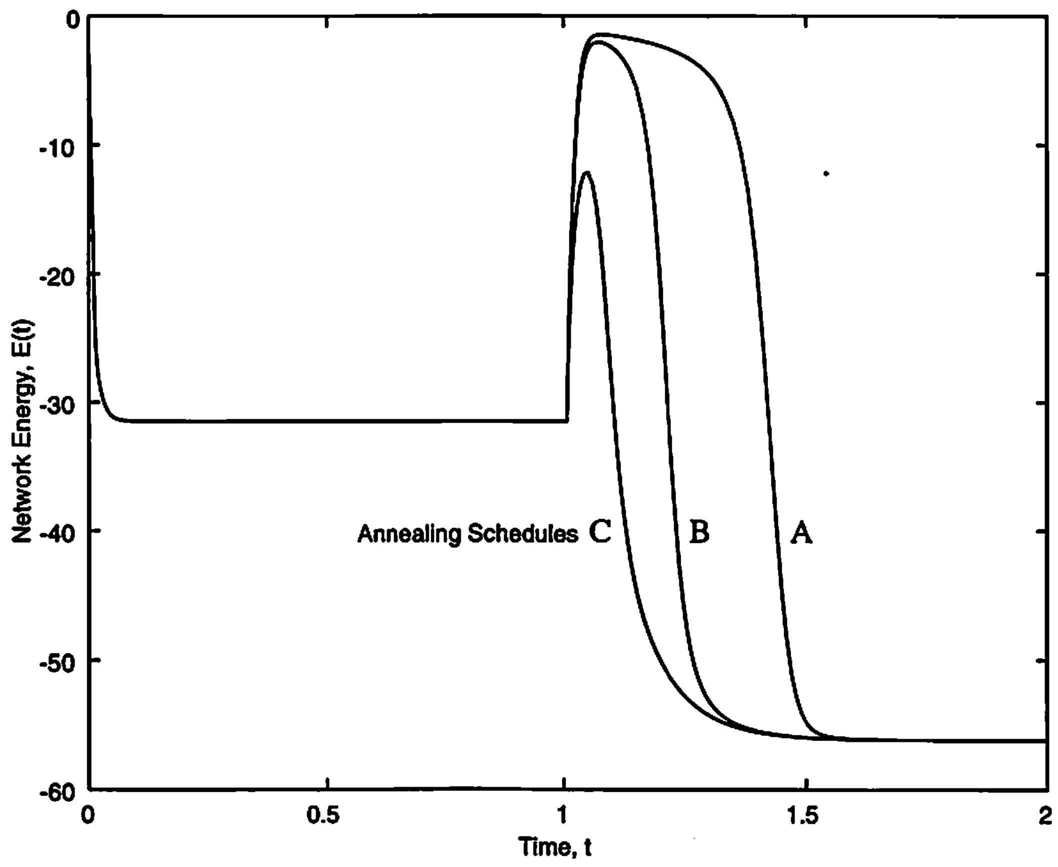


Figure 4-8: Comparison of energy E(t) with and without annealing for different schedules (Annealing is activated at t=1, A:square, B:linear, C:square-root).

where α_1, α_2, and α_3 are constants. In Figure 4-9, the corresponding waveforms for the states and outputs of four cells $C(2,1) - C(2,4)$ are shown. The steady-state value v_{y24} is not affected by the annealing but the polarities of the other outputs changed.

Figure 4-10 shows how often the local minima problems are encountered, and how effectively the hardware annealing solves them in 50 independent experiments with the parameters given in (2.33). In Figure 4-10 (a), each row denotes an experiment and each neuron is numbered as a column. In each experiment, $\boldsymbol{v}_x(0)$ and $\boldsymbol{v}_u$ are chosen randomly in D^{16} so that the constraint conditions and the independency of the experiments are satisfied. In the figure, neurons with the black color indicate the changes in steady-state outputs as the result of hardware annealing. Figure 4-10 (b) shows the comparison of the energy levels for unannealed and annealed conditions. Similar experiments are conducted for two special cases when $\boldsymbol{v}_x(0) = \mathbf{0}$ and $\boldsymbol{v}_u = \mathbf{0}$. Figure 4-10 (c) shows the energy levels when $\boldsymbol{v}_x(0) = \mathbf{0}$. In this case, the initial value of E is zero as given by (4.7). This energy value is quite high compared to those in the steady state and the basin of attraction can be as large as a whole D^N. However, from (4.6) it can be seen that

$$E_{\max} = \frac{1}{2}\boldsymbol{b}^T\mathbf{M}^{-1}\boldsymbol{b} \geq E(0) = 0 \tag{4.50}$$

for positive definite M, as given in Table 2.1. Correspondingly, the network still can stay at the nearest local minimum in the steady state and the zero initial condition does not always provide the optimal solution. On the other hand, the annealing process not only increases the value of energy function to near zero initially, but also force it to reach $E_{\max}$ during the relaxation process as described in the previous section. In Figure 4-10 (d), the external forcing function $\boldsymbol{b}$ is set to zero in which case the energy is solely determined as $E = -0.5\boldsymbol{y}^T\mathbf{M}\boldsymbol{y}$. The figure shows that the hardware annealing provides the globally optimal solution in each experiment. By using the exact steady-state values $|y_k| = 1$, $\forall k$, the

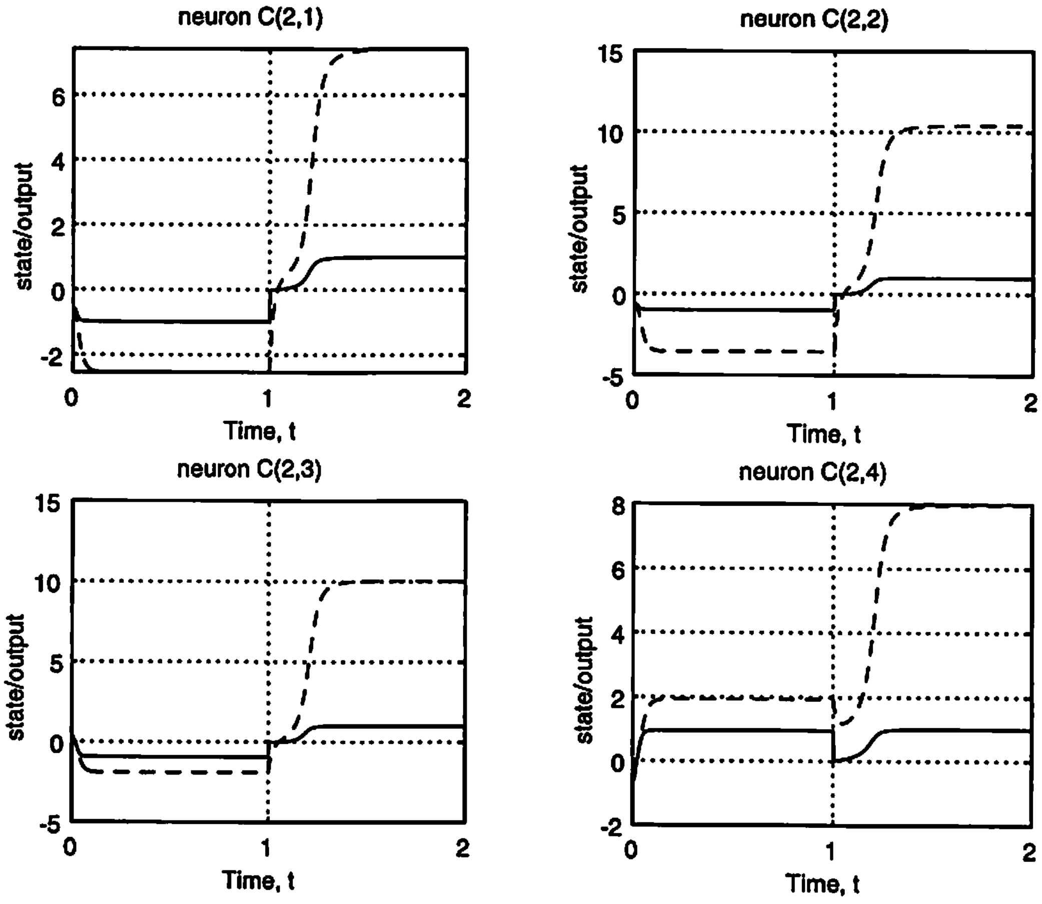
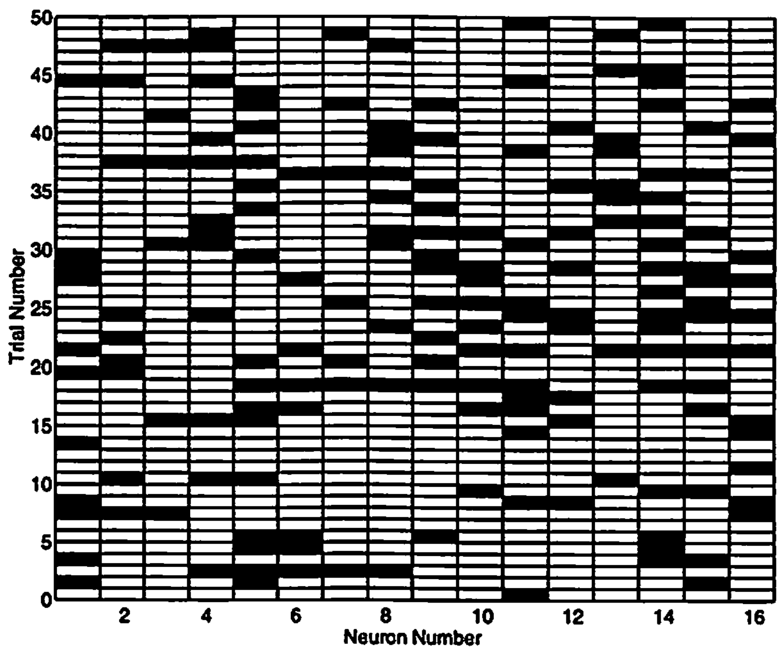
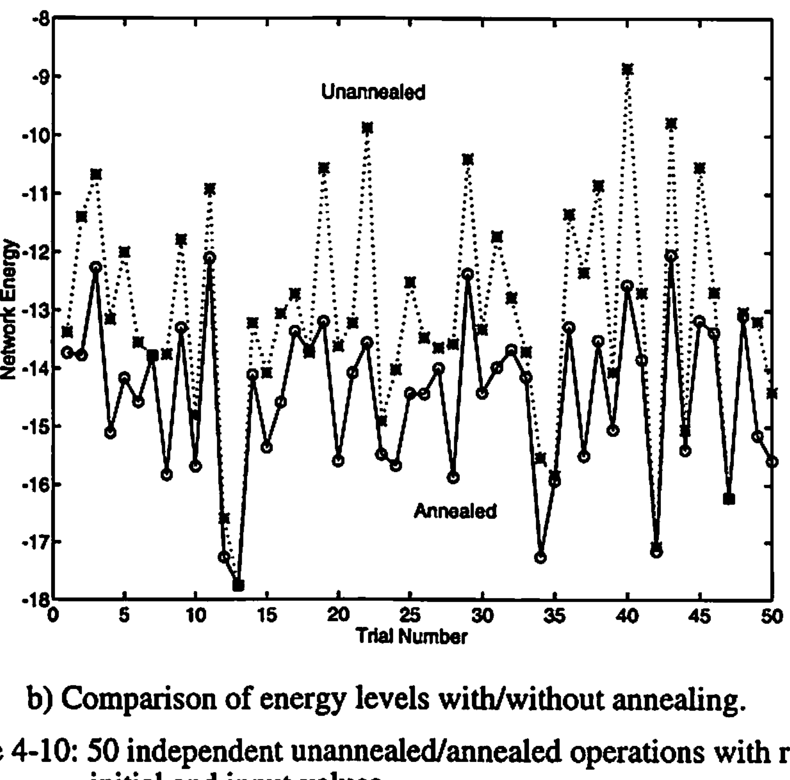


Figure 4-9: Evolution of neuron states and outputs during unannealed and annealed operations (dashed lines: state vx2i, solid lines: output vy2i, i=1,2,3,4).

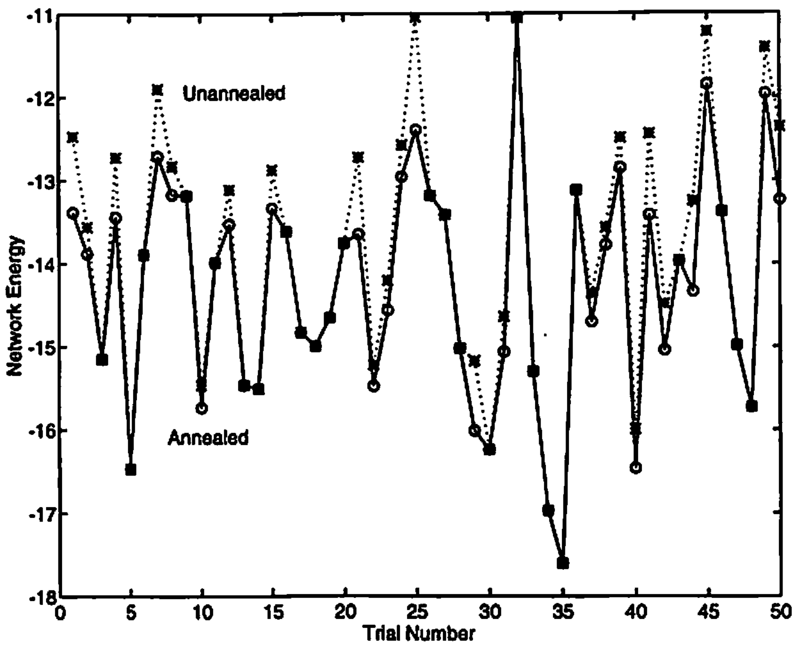


a) Frequencies of sub-optimal solutions and their optimizations by annealing.

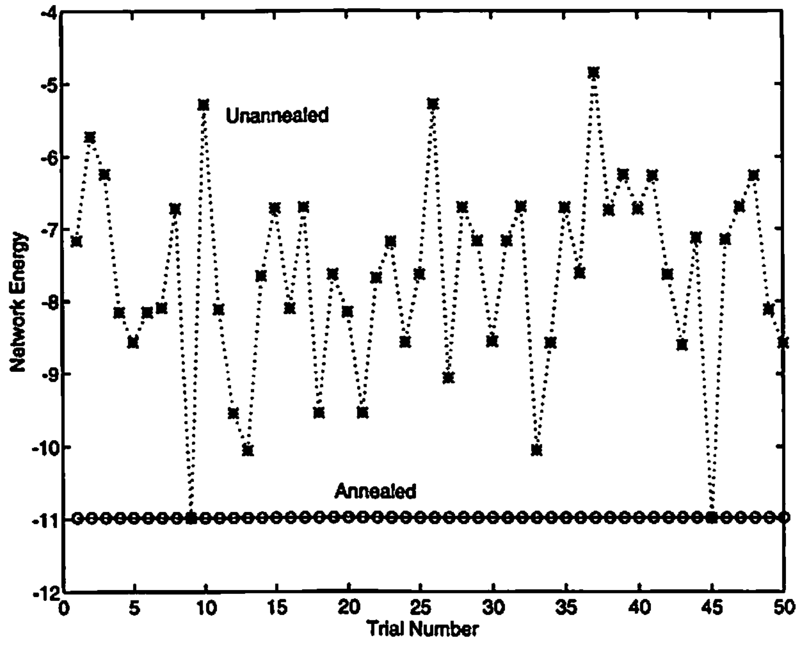


b) Comparison of energy levels with/without annealing.

Figure 4-10: 50 independent unannealed/annealed operations with random initial and input values.



(a) Zero initial state, vxij(0)=0, i,j=1,2,3,4.



(b) Zero input, vuij=0, i,j=1,2,3,4.

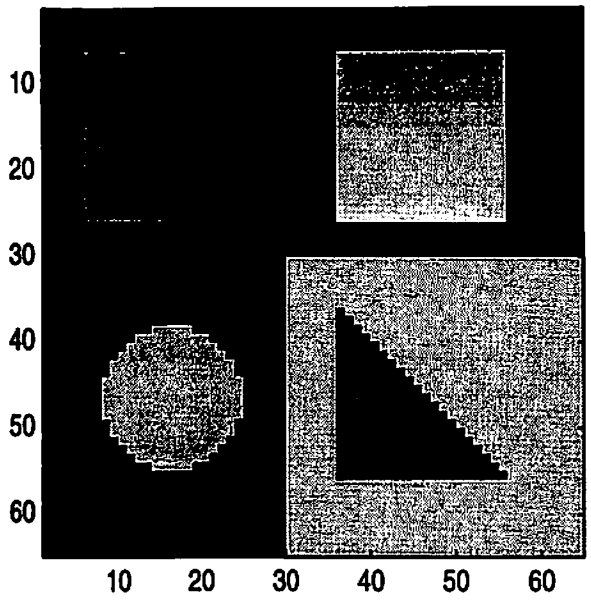
Figure 4-10 (cont.): Experiments with zero-initial and zero-input conditions.

computer-generated solution for the global minimum of E is shown to be -10.8. Note that for the piecewise linear function this value is -14 because the 2nd term in (2.16) is 0.5, instead of 0.7 for the sigmoid nonlinearity.

Figure 4-11 shows the edge detection results of a 64-by-64 CNN for unannealed and annealed conditions. Figure 4-11 (a) shows the original gray-scale image. The sigmoid function is used as the neuron nonlinearity and the cloning templates given in Figure 4-11 (b). See also [2, Fig. 18]. Figures 4-11 (c) shows the CNN outputs due to the annealing effect. Here, the image is applied to the input only and $v_x(0) = \mathbf{0}$. Two networks resulted in the same, correct outputs. When the image is applied to both the input and initial state, the output is not correct as shown in Figure 4-11 (d). It shows the correct output when the annealing operation is applied. In Figure 4-11 (e), the constraint condition $|v_{xij}(0)| \leq 1, \forall i, j$, is removed and random values of the initial state between 1 and 5 are used. As a result, the output of unannealed network contains many neurons that are not able to toggle the states. However, as shown in the figure, the hardware annealing provides enough stimulation to those frozen neurons caused by such ill-conditioned initial states. Next, a small amount (standard deviation of 0.1) of white Gaussian noise is added to the input image. The annealed network produces a clean output as shown in Figure 4-11 (f). However, as noise level increases the effect of noise is not negligible in an annealed network.

4.5 Conclusions and Remarks

The hardware annealing is a very effective method of overcoming local minima in the CNN. Instead of using stochastic optimization as in simulated annealing, or Boltzmann machine, the proposed method continuously reconfigures the energy surface of the network so that the network state easily finds the global minimum. The process of global



(a) Original 64-by-64 gray-scale image.

0	0	0
0	2	0
0	0	0

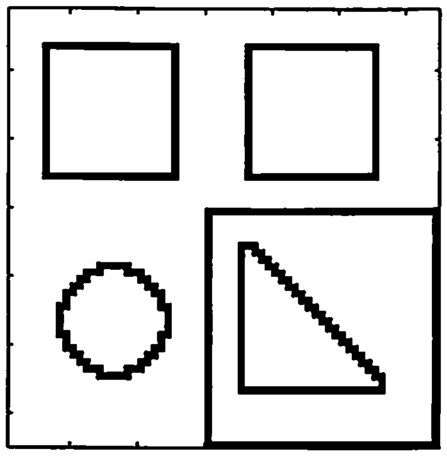
T_A

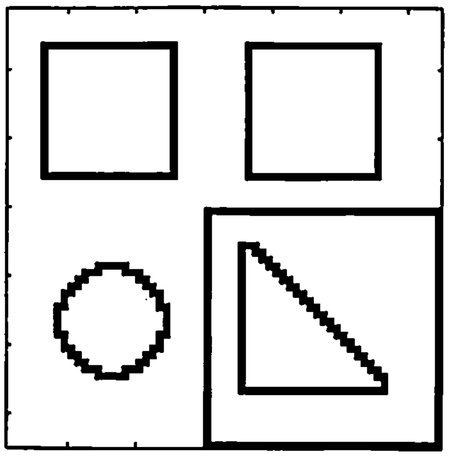
-0.25	-0.25	-0.25
-0.25	2	-0.25
-0.25	-0.25	-0.25

T_B

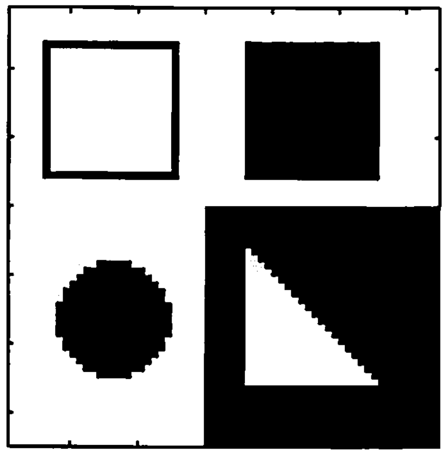
(b) Feedback (T_A) and feedforward (T_B) cloning templates.

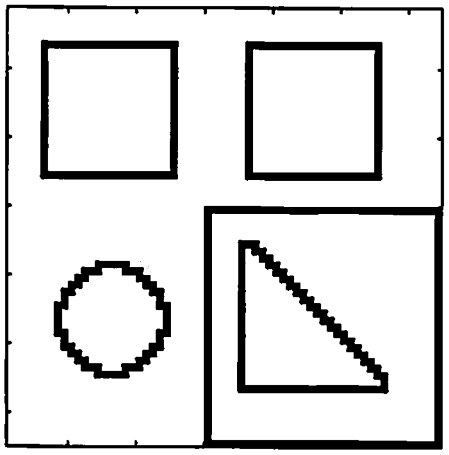
Figure 4-11: Edge detection application of 64-by-64 CNN.


(unannealed operation)


(annealed operation)

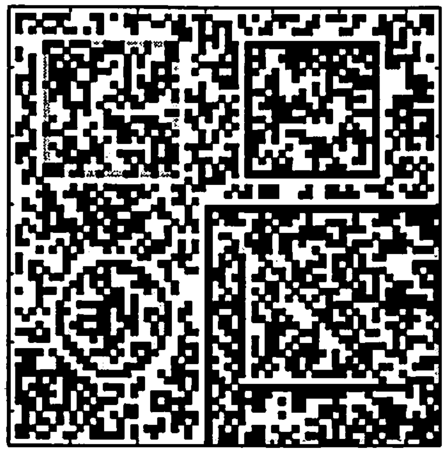
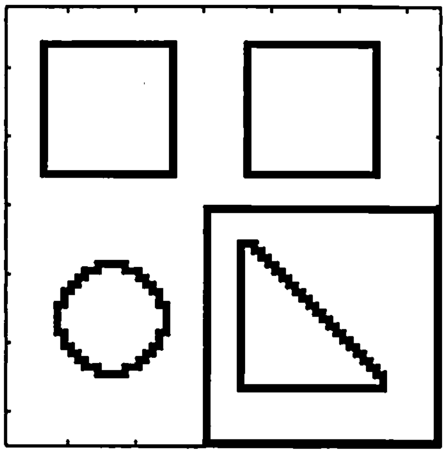
(c) Zero-initial condition vx(0)=0, vu=input image.


(unannealed operation)


(annealed operation)

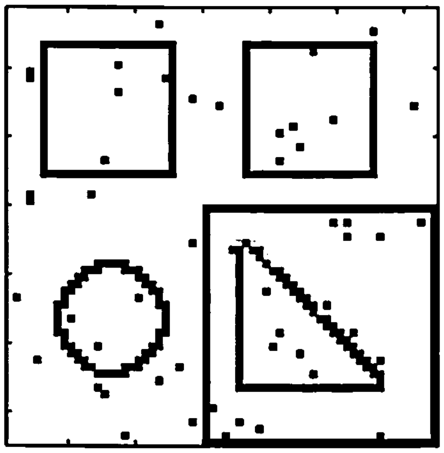
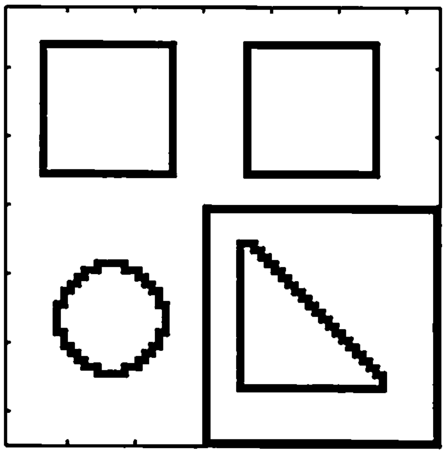
(d) vx(0)=vu=input image.

Figure 4-11 (cont.): Edge detection application of 64-by-64 CNN.

(unannealed operation) (annealed operation)

(e) Ill-initial conditions |vx(0)|>1, vu=input image.

(unannealed operation) (annealed operation)

(f) vx(0)=vu=image+Gaussian noise (standard deviation=0.1).

Figure 4-11: Edge detection application of 64-by-64 CNN.

optimization is explained by the use of eigenvalues of the network. In applications other than optimization, the hardware annealing is also useful in that it provides enough stimulation to overcome the energy barriers among the local minima. Because the annealing can be performed by the direct control of the paralleled hardware, the speed of searching for optimal solutions can be very fast, compared to those of the software-based algorithms.

Chapter V

Maximum-likelihood Sequence Estimation by Neural Networks

5.1 Introduction

The performance of digital communication systems is largely affected by an ability to overcome the channel impairments introduced during signal propagation. Information-carrying signals in wireless mobile communications are often distorted severely due to the noise, limited bandwidth, and multi-path propagations through surrounding objects. Traditionally, detection of signals in such environments was performed by an optimization method for minimizing the probability of detection errors. Several demodulation techniques such as adaptive equalization and maximum-likelihood sequence estimation (MLSE) have been developed and used extensively in such applications. The MLSE system provides an optimum performance in terms of error probability. The numerically efficient version of the MLSE, known as the Viterbi algorithm [114,119] can be implemented in VLSIs [115-118], and the researches toward more efficient hardware implementations continue in both academy and the industry.

The artificial neural networks have been great promise in solving many complex signal processing and optimization problems that can not be addressed satisfactorily with conventional approaches. Supervised/unsupervised learning and massively parallel architectures inherent in the artificial neural networks provide attractive means of optimization and fast problem solving. The neural network approaches in communications have been motivated by the adaptive learning capability to process real-world signals. Well designed neural networks have the ability to perform the error correction of error-control codes [120,121], and equalization of transmission channels [135-139]. Performing maximum likelihood decoding of linear block error-correcting codes is shown to be equivalent to finding a global minimum of the energy function associated with a neural network. Given a code, a neural network can be constructed in such a way that there exists one-to-one correspondence between every codeword and every local minimum of the energy function. Decoding techniques using neural networks can be a useful tool for solving problems of maximization of polynomials over multi-dimensional space.

In this chapter, the Hopfield neural network or CNN with hardware annealing is used to solve the optimization problem involved in the MLSE. In an optimization point of view, the MLSE is a combinatorial minimization of the cost function over all possible sequences of a finite length. The signaling alphabet $\boldsymbol{\alpha} = \{\alpha_k\}$, $k = 1,2,\cdots,M$, and the length of sequence $s_n = \{s_i\}$, $i = 0,1,\cdots,n-1$, correspond to the set of numbers and the dimension of a problem, respectively. There are M^n possible combinations over which the MLSE computes the cost function. The best estimate of transmitted sequence is the one that has the minimum cost. A computationally efficient method is the Viterbi algorithm in which the redundant computations involved in the MLSE are avoided. As an alternative approach to such optimal algorithms, the MLSE by hardware-annealed neural network is proposed and examined as a real-time machine for combinatorial optimization problems.

Although the areas of applications of analog circuits are limited due to inaccuracies caused by parameter variations and noise, the neural network approach using combined discrete-time and continuous-time signal processing techniques has the following advantages over others;

1) *Speed*: The most powerful property of a recurrent associative neural network is the collective or massively-paralleled computations of the solution in a single operation. The best estimate of sequences of length n can be found in a single evolution interval which could be a fraction of microseconds by using semiconductor technologies.
2) *Complexity*: The network complexity increases linearly with the length of the sequence to be estimated and/or the number of channel memory. In addition, the structure of the MLSE mapping entails a modular architecture for large-scale hardware implementation. For the detection of long sequences, multiple modules can be cascaded to construct a large network.
3) *Numerical stability*: Under mild conditions, the neural network exhibits an absolute stability. Conventional algorithms sometimes suffer from the numerical stability, e.g., a large eigenvalue spread in the gradient-descent method [124].

5.2 MLSE using Neural Network

An optimum method of detecting digital data symbols transmitted over time-dispersive, time-varying channels in the presence of additive white Gaussian noise has been known as the maximum-likelihood sequence estimation (MLSE). As a class of nonlinear receivers, it exhibits superior error rate performance compared to its linear counterparts. However, it is often impractical to construct due to the computation intensity and complexity required for the signal processing functions. A more efficient computational method of implementing MLSE is the Viterbi algorithm. An alternative implementation of MLSE function is to use neural networks as a massively parallel computing machine. It can be readily shown that the cost function to be minimized in the

MLSE has the same form as the Lyapunov (energy) function associated with the Hopfield neural network, and if the cost function is mapped onto the network then the desired estimate is obtained at the output.

Figure 5-1 shows the baseband model of a digital communication system over inter-symbol interference (ISI) and additive Gaussian noise channel. The actual ISI channel together with the baseband Nyquist filters in transmitter and receiver can be modeled as a finite impulse response (FIR) filter of length $L+1$. The channel is represented by the impulse response $h(k)=h_k$ and its z-transform $H(z)$. Note that $h(k)=0$ for $k<0$ and $k>L$. The received signal *r(t)* is the convolution of $u(k)=\sum_i u_i\delta(k-i)$ with $h(k)$ where $\delta(k)$ is the Kronecker delta function, plus white Gaussian noise $n(k)$ of zero-mean and variance σ^2,

$$r(k)=\sum_i u_i\, h(k-i)+n(k). \tag{5.1}$$

The maximum likelihood sequence estimation is an optimum receiving system in a sense that it minimizes the probability of errors [114,119]. The maximum-likelihood sequence estimator selects a sequence as a best estimate of transmitted sequence, that maximizes the conditional *a posterior* probabilities $p(r_n|u_n)$, where $r_n=\{r_0,r_1,\cdots,r_{n-1}\}$ and $u_n=\{u_0,u_1,\cdots,u_{n-1}\}$ are the received and transmitted sequences of length *n*, respectively. For a sufficiently large *n*, the MLSE algorithm is to choose a sequence that maximizes a scalar cost function

$$J=--\sum_{k=0}^{n-1}\left(r_k-\sum_{i=0}^{L}h_i^* u_{k-i}\right)^2=-\sum_{k=0}^{n-1}\left(r_k-\sum_{i=0}^{n-1}h_{k-i}^* u_i\right)^2 \tag{5.2}$$

for all possible combinations of sequences of length *n*. The cost function (5.2) is simply the sum of squared-errors between received sample and the output of the channel for the input u_n to be estimated. The evaluation of values given by (5.2) must be performed over

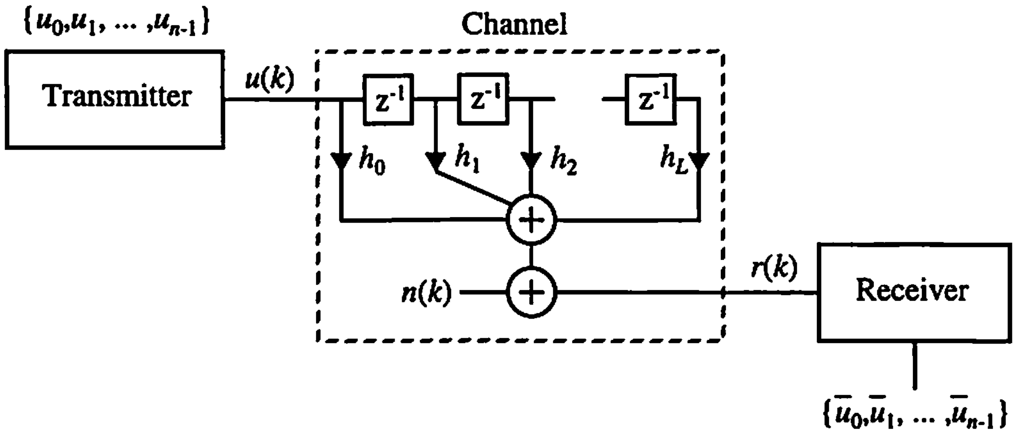


Figure 5-1: Baseband digital communication system model.

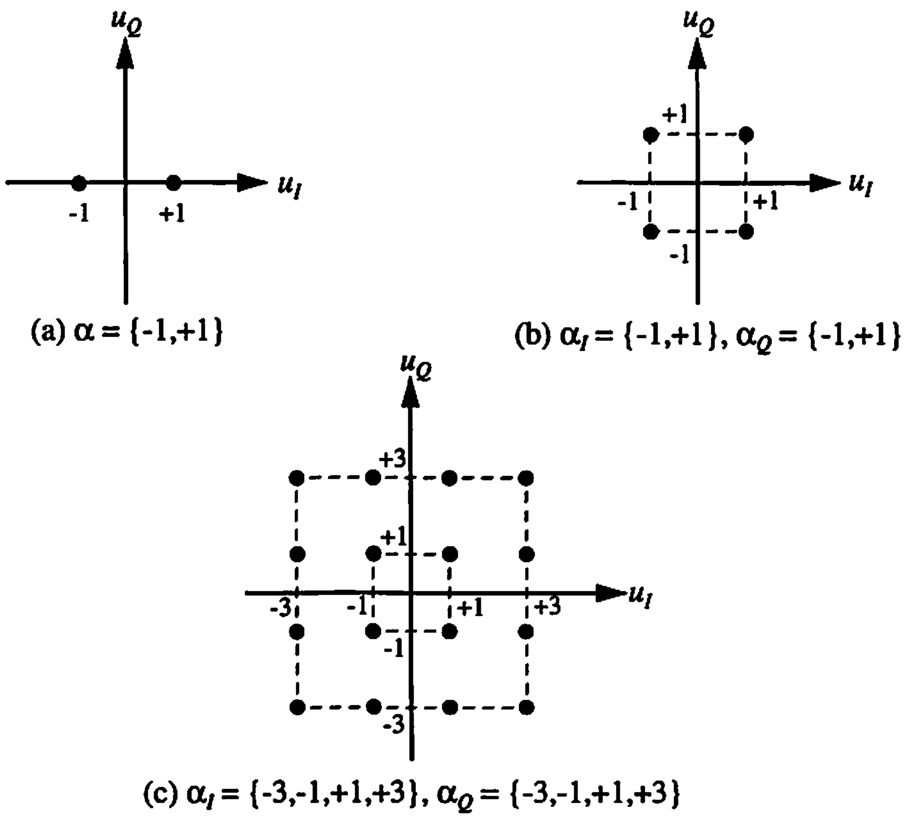

u_Q
u_I
-1
+1
(a) $\alpha = \{-1,+1\}$
(b) $\alpha_I = \{-1,+1\}$, $\alpha_Q = \{-1,+1\}$
+3
-3
(c) $\alpha_I = \{-3,-1,+1,+3\}$, $\alpha_Q = \{-3,-1,+1,+3\}$

Figure 5-2: Examples of signal constellations; (a) binary, (b) QPSK, and (c) 16-ary QAM.

all possible sequences of $u_n = \{u_0, u_1, \cdots, u_{n-1}\}$. Therefore the algorithm complexity is proportional to M^n, where M is the number of signaling alphabets, i.e., $u_k \in \{\alpha_1, \alpha_2, \cdots, \alpha_M\}, \forall k$ and n is the length of the sequence to be estimated. In typical data communications in which the length of a sequence is not given explicitly, the number n can be arbitrarily large and in principle could be infinity. When (5.2) is expanded, the first term can be discarded because it is a constant for given input r_n. Then, by changing the sign of the resulting cost function and dividing by 2, the MLSE is equivalent to minimizing the quantity

$$\begin{aligned}\hat{J}_n &= -\mathrm{Re}\left\{\sum_{i=0}^{n-1}\left(\sum_{k=0}^{n-1} r_k h_{k-i}^*\right)u_i\right\} + \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left(\sum_{k=0}^{n-1} h_{k-i}^* h_{k-j}\right)u_i u_j^* \\ &= \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} x_{i-j} u_i u_j^* - \mathrm{Re}\left\{\sum_{i=0}^{n-1} y_i u_i\right\},\end{aligned} \tag{5.3}$$

where

$$y_i \equiv \sum_{k=0}^{n-1} r_k h_{k-i}^*$$

$$x_l \equiv \sum_{k=0}^{n-1} h_k^* h_{k+l} = \sum_{k=0}^{L} h_k^* h_{k+l}.$$

The y_i is the cross-correlation between the received signal and $h(k)$, while x_l is the auto-correlation of $h(k)$. If the channel is considered to be time-invariant during n symbol intervals, then $x_{-k} = x_k^*$, $k = 1, 2, \cdots, L$. In vector and matrix forms, (5.3) can be written as

$$\hat{J}_n = \frac{1}{2} u^H \mathrm{X} u - \mathrm{Re}\{u^H y\} \tag{5.4}$$

where $u = [u_0 \; u_1 \; \cdots \; u_{n-1}]^T = u_I + j u_Q$, $\qquad u \in \{\alpha_1, \alpha_2, \cdots, \alpha_M\}^n$, $\alpha_i \in C$

$y = [y_0 \; y_1 \; \cdots \; y_{n-1}]^T = y_I + j y_Q$, $\qquad y \in C^n$

$$\mathbf{X}=\begin{bmatrix} x_0 & x_{-1} & \cdots & x_{-n+2} & x_{-n+1} \\ x_1 & x_0 & \cdots & x_{-n+3} & x_{-n+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n-2} & x_{n-3} & \cdots & x_0 & x_{-1} \\ x_{n-1} & x_{n-2} & \cdots & x_1 & x_0 \end{bmatrix}=\mathbf{X}_I+j\mathbf{X}_Q,\quad x_k\in\boldsymbol{R}.$$

Here, the channel is assumed to be stationary at least for n symbol intervals. If the channel is non-stationary, $x_k \neq x_{-k}$ in general.

In general, a data communication system transmits and receives a sequence of complex-valued data symbols $\{u_k\}$, where $u_k = u_{I,k} + ju_{Q,k}$, $u_{I,k} \in \boldsymbol{\alpha}_I$, $u_{Q,k} \in \boldsymbol{\alpha}_Q$. For a physical realization of the MLSE, the received baseband signal must be separated into the real and complex parts in (5.4). The signaling alphabet $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \cdots, \alpha_M\}$ depends on the modulation techniques employed. Therefore, consideration is made on the binary signaling alphabet $\boldsymbol{\alpha} = \{-1,+1\}$, for which M=2 and all the quantities in (5.7) are real. The followings are some of typical sets in normalized antipodal formats [119];

$\boldsymbol{\alpha} = \{-1,+1\}$,	binary PSK or QAM
$\boldsymbol{\alpha}_I, \boldsymbol{\alpha}_Q = \{-1,+1\}$,	QPSK or 4-ary QAM
$\boldsymbol{\alpha}_I, \boldsymbol{\alpha}_Q = \{-3,-1,+1,+3\}$,	16-ary QAM.

Figure 5-2 shows the constellation diagrams of these signaling alphabets. Of course, there exist other constellations in QPSK and 16-ary QAM techniques, and other techniques such as 256-QAM modulation. The major attention is paid to the binary and QPSK for which the binary neurons can accommodate the signals. However, a neural network with multi-level neurons can be used for more complicated signal constellations, e.g., 4-level bipolar neurons for 16-ary QAM.

The correlation matrix X is Hermitian, Toeplitz, and positive semi-definite [124], which imply that $\mathbf{X}_I^T = \mathbf{X}_I$ is symmetric and $\mathbf{X}_Q^T = -\mathbf{X}_Q$ is skew symmetric from the equality $\mathbf{X}^H = \mathbf{X}_I^T - j\mathbf{X}_Q^T = \mathbf{X} = \mathbf{X}_I + j\mathbf{X}_Q$, and all eigenvalues of X are real and non-negative. In almost any situations except the case $x_0 = 0$, X is positive definite. The cost function $\hat{J}_n$ given by (5.3) is real and needs to be represented by real quantities in the right hand side for the physical realization of the equation. By using the properties of symmetric and skew-symmetric matrices, (5.4) can be expanded and reformulated as

$$\begin{aligned}\hat{J}_n &= \frac{1}{2}\left(u_I^T \mathrm{X}_I u_I + 2u_Q^T \mathrm{X}_Q u_I + u_Q^T \mathrm{X}_I u_Q\right) - \left(u_I^T y_I + u_Q^T y_Q\right) \\ &= \frac{1}{2}\left[u_I^T \,\vdots\, u_Q^T\right]\left[\begin{array}{c|c} \mathrm{X}_I & \mathrm{X}_Q^T \\ \hline \mathrm{X}_Q & \mathrm{X}_I \end{array}\right]\left[\begin{array}{c} u_I \\ \hline u_Q \end{array}\right] - \left[u_I^T \,\vdots\, u_Q^T\right]\left[\begin{array}{c} z_I \\ \hline z_Q \end{array}\right] \\ &\equiv \frac{1}{2}\bar{u}^T \overline{\mathrm{X}} \bar{u} - \bar{u}^T \bar{y}. \end{aligned} \tag{5.5}$$

Here, the matrix $\overline{\mathrm{X}}$ is again symmetric and positive semi-definite. Let $x^T = \left[x_1^T \,\vdots\, x_2^T\right]$ where $x_1, x_2 \in R^n$. Then

$$x^T \overline{\mathrm{X}} x = \left[\begin{array}{c} x_1 \\ \hline x_2 \end{array}\right]^T \left[\begin{array}{c|c} \mathrm{X}_I & \mathrm{X}_Q^T \\ \hline \mathrm{X}_Q & \mathrm{X}_I \end{array}\right]\left[\begin{array}{c} x_1 \\ \hline x_2 \end{array}\right] = x_1^T \mathrm{X}_I x_1 + x_1^T \mathrm{X}_Q^T x_2 + x_2^T \mathrm{X}_Q x_1 + x_2^T \mathrm{X}_I x_2. \tag{5.6}$$

Now, since X is Hermitian and positive semi-definite, the quantity $\tilde{x}^H \mathrm{X} \tilde{x}$ is real and non-negative for any $\tilde{x} \in C^{2n}$. Using $\tilde{x} = x_1 + jx_2$, we have

$$\begin{aligned}\tilde{x}^H \mathrm{X} \tilde{x} &= \left(x_1 + jx_2\right)^H \left(\mathrm{X}_I + j\mathrm{X}_Q\right)\left(x_1 + jx_2\right) \\ &= x_1^T \mathrm{X}_I x_1 - x_1^T \mathrm{X}_Q x_2 + x_2^T \mathrm{X}_Q x_1 + x_2^T \mathrm{X}_I x_2 \geq 0. \end{aligned} \tag{5.7}$$

From the skew-symmetric property of X_Q, $x_1^T \mathrm{X}_Q^T x_2 = -x_1^T \mathrm{X}_Q x_2$. Therefore, (5.6) and (5.7) are the same and $x^T \overline{\mathrm{X}} x \geq 0$ for any $x \in R^{2n}$.

5.3 System Model

The artificial neural network considered here for implementation of the MLSE function is based on the work done by Hopfield [2-4]. The similarity between the cost function given in (5.5) and the energy function of Hopfield network allows a neural network implementation of the MLSE function for digital communications. First, consider the Hopfield type energy function of the form

$$E = -\frac{1}{2} v^T W v - v^T \theta, \tag{5.8}$$

where $v \in \{-1,+1\}^{2n}$, $W \in R^{2n \times 2n}$, and $\theta \in R^{2n}$. This energy function corresponds to two-level threshold neurons. If neuron gain is finite but very large, (5.8) is still valid without any significant error [2,3]. In this case, v is the continuous-valued output with $v \in D^{2n} = \{v \in R^{2n}: -1 \le v_i \le +1, i = 0,1,\cdots,2n-1\}$. A direct comparison of (5.5) with (5.8) reveals that the desired estimate $\hat{u}_n$ can be obtained at the output of a Hopfield neural network if

$$W = -\overline{X} = -\left[\begin{array}{c|c} X_I & X_Q^T \\ \hline X_Q & X_I \end{array}\right] \text{ and } \theta = \overline{y} = \left[\begin{array}{c} y_I \\ \hline y_Q \end{array}\right]. \tag{5.9}$$

In other words, the cost function $\hat{J}_n$ is mapped onto a neural network constructed by the transconductance matrix $W = \overline{X}$ and input vector $\theta = \overline{y}$.

Figure 5-3 shows the block diagram of the neural network MLSE receiver. The received signal $r(t)$ is first separated into two baseband signals, i.e., in-phase signal $r_I(t)$ and quadra-phase signal $r_Q(t)$. The signals are then sampled at $t = kT$ where T is the duration of a symbol, and the resulting discrete-time signals $r_I(k)$ and $r_Q(k)$ are correlated with the channel impulse response $h(k)$. The correlation filter matched to channel impulse response $h(k)$ is approximated by a FIR filter, whose tab coefficients are updated sequence by sequence. In general, because the channel characteristics is not known, its impulse response is also estimated by using the received reference signals. Thus, the estimate $\hat{h}(k)$ is used instead. Note that the channel estimation is equivalent to finding a set of filter coefficients that minimizes the cost function or the mean-squared errors between received and desired signals. Therefore, it corresponds to a convex

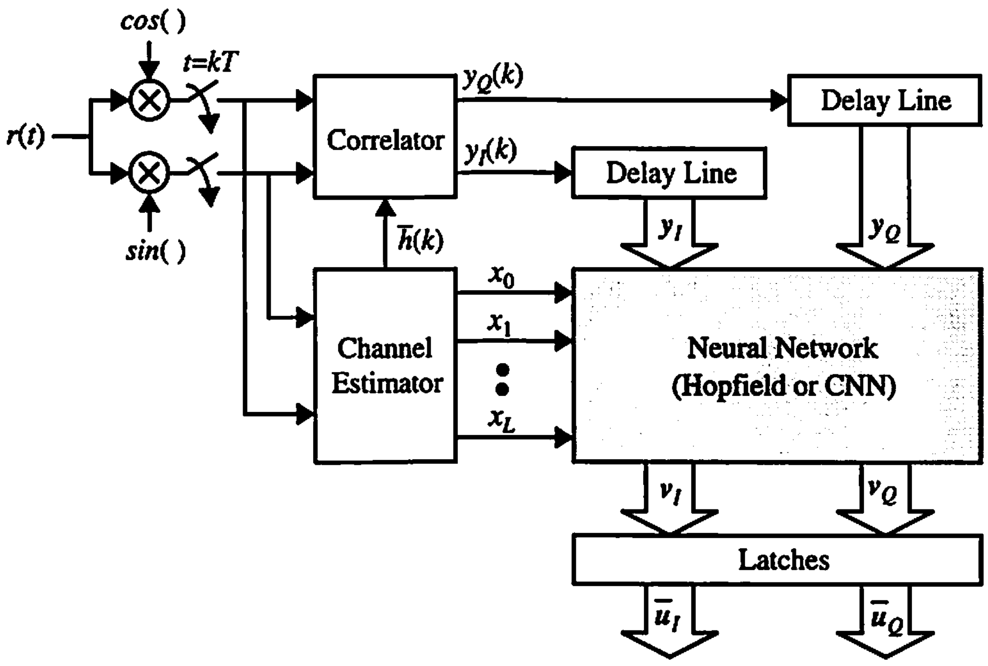


Figure 5-3: Block diagram of neural network MLSE receiver.

optimization problem that can be efficiently solved by a neural network [1,25]. However, such a approach is not considered and the structure of the channel estimator in Figure 5-3 is left undecided in this thesis. The in-phase and quadra-phase signals from the correlators are shifted into two separate tapped delay lines of length n. The outputs of tapped-delay lines are given by

$$\begin{aligned} y_I &= \left[y_I(0)\ y_I(1)\ \cdots\ y_I(n-1)\right]^T = \left[y_{I0}\ y_{I1}\ \cdots\ y_{I(n-1)}\right]^T \\ y_Q &= \left[y_Q(0)\ y_Q(1)\ \cdots\ y_Q(n-1)\right]^T = \left[y_{Q0}\ y_{Q1}\ \cdots\ y_{Q(n-1)}\right]^T. \end{aligned}$$

If a binary modulation technique is used, all the signals are real and only one receiving path for signal preprocessing is required. Either a Hopfield neural network or CNN can be used as the core of nonlinear signal processing for the MLSE as shown in the figure. If the neural network produces saturated binary or multi-level values in the steady-state, the outputs represent the MLSE of received sequence, i.e., $\hat{u}_n = \{\hat{u}_0\ \hat{u}_1\ \cdots\ \hat{u}_{n-1}\}$. Since the cost function to be minimized is unique, the resulting Hopfield neural network and CNN are basically the same except the neuron models. In the Hopfield network case, the network is reset to all zero initial state values at the beginning of each operation. On the other hand, the input $\bar{y}$ in a CNN architecture can be served as the input to the network, initial state or both. After n symbols are shifted into the delay lines, the network performs the MLSE of an n-symbol sequence through an autonomous evolution of its internal state for $0 \le t < T_M$. If the shift operations of delay lines are pipelined, the network can estimate n/T_M symbols per second. For example, if $T_M = 1$ µsec. and $n = 100$, then a real-time operation of symbol rate up to 1×10^8 symbols/sec. is readily achievable.

***A) High-gain Neurons and Hopfield Model*:**

Since the matrix X is positive semi-definite, E is a convex function of output v and has a unique minimum. However, the MLSE is a combinatorial minimization of $\hat{J}_n$ over the set of signaling alphabets. In the bipolar binary case, $\bar{u} \in \{-1,+1\}^{2n}$ while $v \in D^{2n} = \{v \in R^{2n}: -1 \le v_i \le +1, i = 0,1,\cdots,2n-1\}$. In order to transform the convex optimization into a concave equivalence, we add the constraint energy $E_C = \mu(v+\phi)^T(v-\phi)$, where ϕ is a $2n$-by-1 unity vector and μ is a constant. If we neglect the constant term $\mu\phi^T\phi = \mu(2n)$, the cost function (5.5) is mapped onto the neural network with a modified energy function

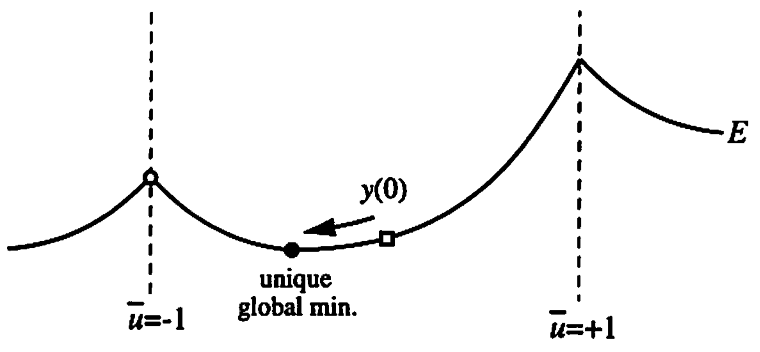
$$\begin{aligned}\hat{E} &= E\big|_{\mathbf{W}=\overline{\mathbf{X}},\,\mathbf{\theta}=\bar{y}} + \mu v^T v \\ &= -\frac{1}{2}v^T(\overline{\mathbf{X}}-2\mu\mathbf{I})v - v^T\bar{y} \equiv -\frac{1}{2}v^T\overline{\mathbf{W}}v - v^T\overline{\mathbf{\theta}}, \qquad \mu < -\frac{\lambda_{\max}}{2},\end{aligned} \tag{5.10}$$

where $\mathbf{I}$ is a $2n$-by-$2n$ unity matrix and $\lambda_{\max} > 0$ is the maximum eigenvalue of $\overline{\mathbf{X}}$. The second term of (5.10) corresponds to the constraint energy satisfying $E_C = \mu(v+\phi)^T(v-\phi) \ge 0$, $v \in D^{2n}$, where the equality holds only if $v \in \{-1,+1\}^{2n}$. The parameter μ controls the shape of energy landscape. If $\mu < -\lambda_{\max}/2$, then $\hat{E}$ is a concave function of v by the negative definite matrix $-\overline{\mathbf{W}} = \overline{\mathbf{X}} + 2\mu\mathbf{I}$, and saturated binary output in the steady state is guaranteed such that $v(t=\infty) \in \{-1,+1\}^{2n}$. The maximum eigenvalue $\lambda_{\max}$, on the other hand, is difficult to compute and may vary sequence by sequence. The eigenvalues of $\overline{\mathbf{X}}$ are real and bounded by [110];

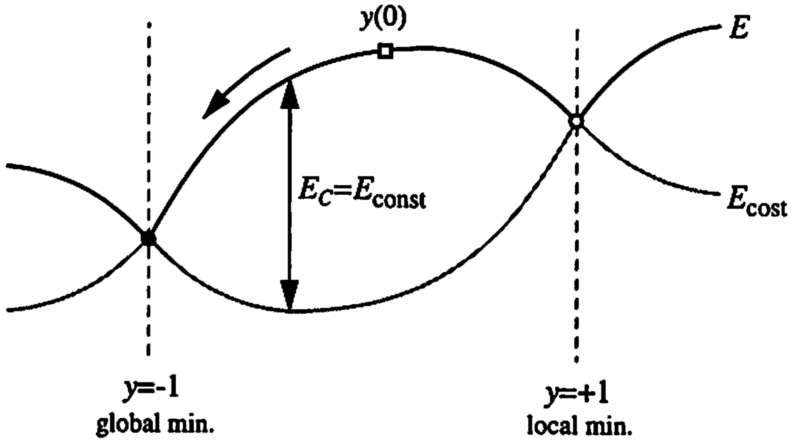
$$x_0 - 2\sum_{i=1}^{L}|x_i| \le \lambda \le x_0 + 2\sum_{i=1}^{L}|x_i|. \tag{5.11}$$

Therefore, the parameter μ can be chosen such that $\mu < -x_0/2 - \sum_{i=1}^{L}|x_i|$. Note that the positive semi-definite X, i.e., $\lambda \ge 0$, does not imply $x_0 \ge 2\sum_{i=1}^{L}|x_i|$ in (5.11). Figure 5-4 shows typical energy landscapes of the original and modified energy functions.

Figure 5-5 shows the network diagram when $n = 6$ and $L = 2$. In this case, the matrix $\overline{\mathbf{X}}$ given as

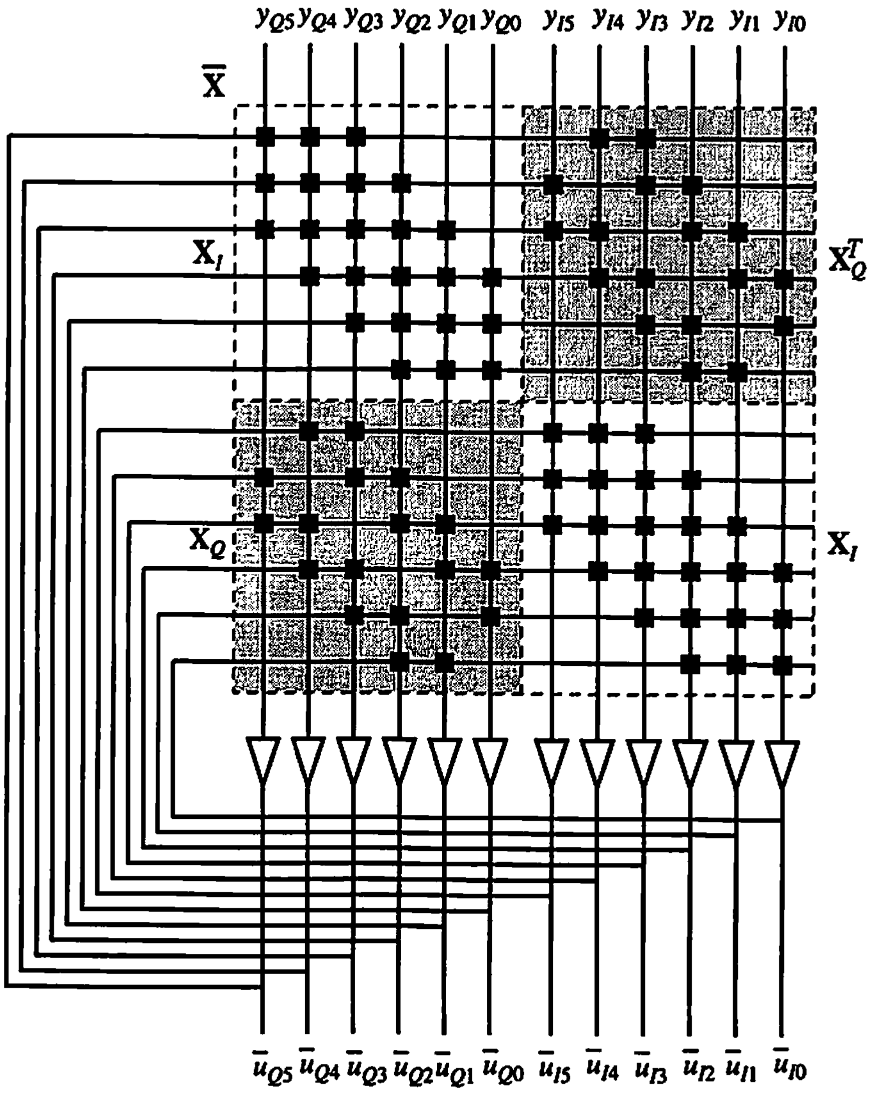


(a) Convex energy function: E=E_{cost}.



(b) Concave energy function: E=E_{cost}+E_{const}.

Figure 5-4: Addition of constraint energy for combinatorial optimization.



y_{Q5} y_{Q4} y_{Q3} y_{Q2} y_{Q1} y_{Q0} y_{I5} y_{I4} y_{I3} y_{I2} y_{I1} y_{I0}
$\overline{\mathbf{X}}$
$\mathbf{X}_I$
$\mathbf{X}_Q^T$
$\mathbf{X}_Q$
$\mathbf{X}_I$
$\bar{u}_{Q5}$ $\bar{u}_{Q4}$ $\bar{u}_{Q3}$ $\bar{u}_{Q2}$ $\bar{u}_{Q1}$ $\bar{u}_{Q0}$ $\bar{u}_{I5}$ $\bar{u}_{I4}$ $\bar{u}_{I3}$ $\bar{u}_{I2}$ $\bar{u}_{I1}$ $\bar{u}_{I0}$

Figure 5-5: Hopfield neural network for QPSK modulation (L=2, n=6).

$$\overline{\mathbf{X}} = \left[\begin{array}{c:c} \mathbf{X}_I & \mathbf{X}_Q^T \\ \hdashline \mathbf{X}_Q & \mathbf{X}_I \end{array}\right]. \tag{5.12}$$

directly corresponds to geometrical locations of submatrices and elements. Next, consider the antipodal binary signaling alphabet $\alpha = \{-1,+1\}$. In this case, M=2 and all the quantities in (5.5) are real. Thus, the cost function $\hat{J}_n$ and the corresponding synapse matrix $\overline{\mathbf{W}}$ are simplified as

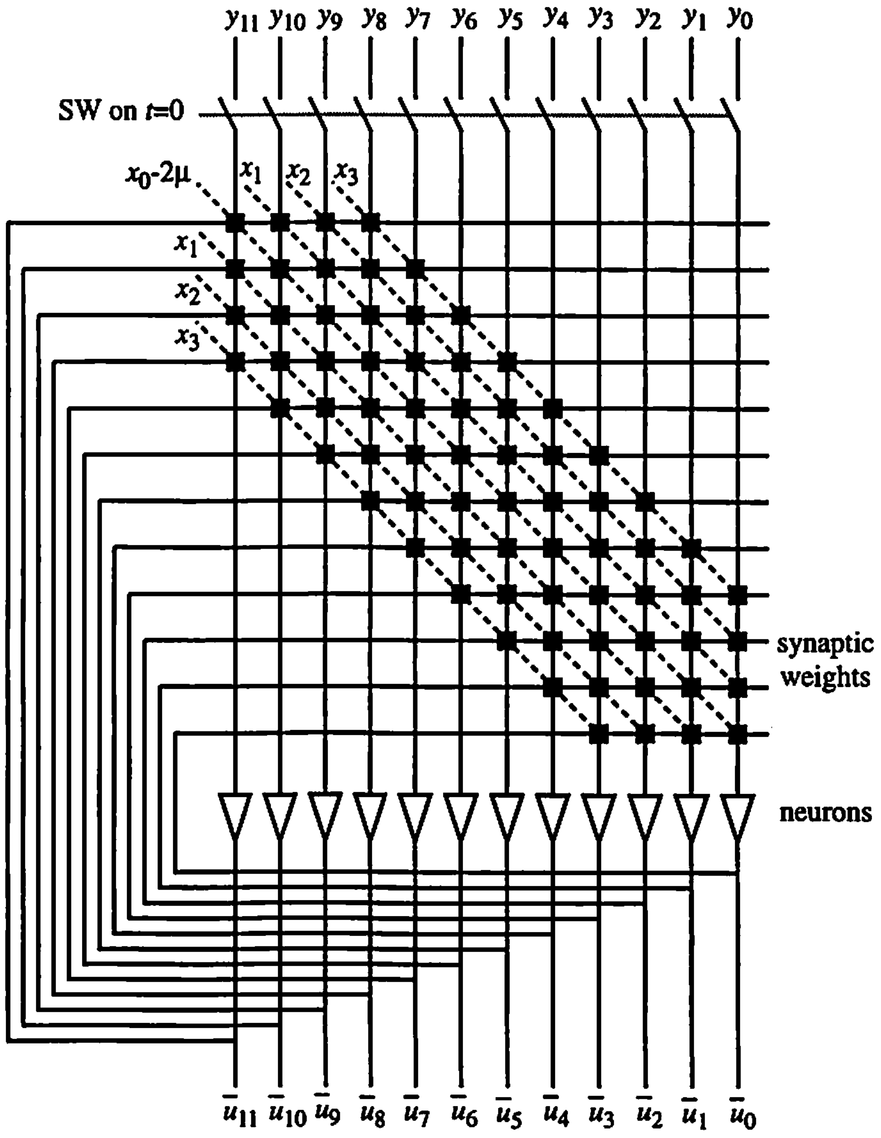
$$\hat{J}_n = \frac{1}{2}u^T \mathbf{X} u - u^T y \tag{5.13}$$
$$\overline{\mathbf{W}} = \overline{\mathbf{X}} - 2\mu\mathbf{I} = -\mathbf{X} - 2\mu\mathbf{I}.$$

The correlation matrix X is real symmetric and Toeplitz with $x_k = x_{-k}$, $k = 0,1,\cdots,L$, in the k-th diagonal above or below the main diagonal. In Figure 5-6, the block diagram of a neural network for the MLSE is shown (L=3, n=12). The network has n neurons and the $n \times n$ matrix $\overline{\mathbf{W}}$ has $n(2L+1) - L(L+1)$ nonzero elements which is approximately equal to $n(2L+1)$ if $n >> L$. Therefore, the complexity of the MLSE neural network is proportional to the sequence length n or channel memory L. The property of $\overline{\mathbf{W}}$ enables a further reduction in the number of synapses by $L(2n - L - 1)/2$. Furthermore, the MLSE neural network has a strong local connectivity

$$C_i \frac{du_i}{dt} = -T_i u_i + \sum_{j=0}^{n-1} \overline{W}_{ij} v_j + \overline{\theta}_i = -T_i u_i - \sum_{j=i-L}^{i+L} x_j v_j - 2\mu v_i + y_i, \tag{5.14}$$

Here, $-2\mu > 0$ is the transconductance for a self-feedback and $\mu < 0$ is a bias term. Note that the self-feedback is always positive because $-x_0 - 2\mu > 2\sum_{l=1}^{L} |x_l| \geq 0$.

$$E = -\frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \overline{W}_{ij} v_i v_j - \sum_{i=0}^{n-1} \overline{\theta}_i v_i$$
$$= \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=i-L}^{i+L} x_j v_i v_j + \mu \sum_{i=0}^{n-1} (v_i)^2 - \sum_{i=0}^{n-1} y_i v_i . \tag{5.15}$$



y_{11} y_{10} y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0
SW on t=0
x_0-2μ
x_1
x_2
x_3
x_1
x_2
x_3
synaptic weights
neurons
$\bar{u}_{11}$ $\bar{u}_{10}$ $\bar{u}_9$ $\bar{u}_8$ $\bar{u}_7$ $\bar{u}_6$ $\bar{u}_5$ $\bar{u}_4$ $\bar{u}_3$ $\bar{u}_2$ $\bar{u}_1$ $\bar{u}_0$

Figure 5-6: Hopfield neural network for binary modulation (L=3, n=12).

As can be seen from (5.14) and (5.15), regardless of the number n, only $2L+1$ adjacent or neighboring neurons are interconnected together. This local interconnection architecture is very desirable for efficient hardware implementation using VLSI technologies. Moreover, because $x_{-k} = x_k$, $1 \le k \le L$, for stationary channels, the number of interconnections can be reduced from $2L+1$ to $L+1$ if one input to the synapse circuit is given by $-(v_{i+k} + v_{i-k})$ through differential input terminals.

B) Piecewise-Linear Neuron and CNN Model:

Now consider the piecewise-linear neuron with the same architecture. Now consider the Lyapunov function of a CNN in vector and matrix forms,

$$E = -\frac{1}{2}v^T(\mathbf{A} - T_x\mathbf{I})v - v^T(\mathbf{B}u + I_b w) \equiv -\frac{1}{2}v^T\mathbf{M}v - v^T b, \tag{5.16}$$

where all parameters are defined in Chapter II. Here, u and $T_x = 1/R_x$ are the external input to the CNN and the input transconductance of a cell in the equivalent circuit diagram in Figure 2-4, respectively. The mapping can be done by letting $\mathbf{M} = -\overline{\mathbf{X}}$ and $b = \overline{y}$. Correspondingly, the transconductance matrix and input vector are given as $\mathbf{A} = -\overline{\mathbf{X}} + T_x\mathbf{I}$ and $\mathbf{b} = \overline{y}$, respectively. Here, the constant term $T_x\mathbf{I}$ represents positive unity feedback in each cell. As in the previous case, the matrix $-\mathbf{M} = \overline{\mathbf{X}}$ is positive semi-definite and accordingly E is a convex function of v. For the autocorrelation function, $x_0 \ge 0$ and all the diagonal elements of $\mathbf{M}$ are negative $A(i,j;i,j) - T_x = -x_0 \le 0$. Correspondingly, in each cell the amount of self-feedback is less than 1 and the saturated binary output is not guaranteed. The addition of the constraint energy $E_C = \mu v^T v$ as in the Hopfield network case gives us an equivalent concave energy function;

$$\begin{aligned}\hat{E} &= E\big|_{\mathbf{M}=\overline{\mathbf{X}},\mathbf{b}=\overline{y}} + \mu v^T v = -\frac{1}{2}v^T(\overline{\mathbf{X}} - 2\mu\mathbf{I})v - v^T\overline{y} \\ &\equiv -\frac{1}{2}v^T\overline{\mathbf{M}}v - v^T\overline{b} = -\frac{1}{2}v^T(\overline{\mathbf{A}} - T_x\mathbf{I})v - v^T\overline{b}.\end{aligned} \tag{5.17}$$

The parameters are now given by $\overline{\mathbf{A}} = -\overline{\mathbf{X}} + (T_x - 2\mu)\mathbf{I}$ and $\bar{b} = \bar{y}$, where a constant μ is chosen such that $\overline{\mathbf{M}} = -\overline{\mathbf{X}} - 2\mu\mathbf{I}$ is positive definite. The condition on the parameter μ can be somewhat alleviated by allowing an indefinite $\overline{\mathbf{M}}$ with all positive elements on the main diagonal, i.e., $\bar{A}(i,j;i,j) - T_x = -x_0 - 2\mu > 0$ [59]. From (4.1) and (5.17), the CNN has two rows and the feedback operator

$$\overline{\mathbf{A}} = \left[\begin{array}{c:c} \mathbf{A}_0 & \mathbf{A}_1^T \\ \hdashline \mathbf{A}_1 & \mathbf{A}_0 \end{array}\right] = \left[\begin{array}{c:c} -\mathbf{X}_I + (T_x - 2\mu)\mathbf{I} & -\mathbf{X}_Q^T \\ \hdashline -\mathbf{X}_Q & -\mathbf{X}_I + (T_x - 2\mu)\mathbf{I} \end{array}\right], \tag{5.18}$$

where $\mathbf{A}_0^T = \mathbf{A}_0$ and $\mathbf{A}_1^T = -\mathbf{A}_1$. The corresponding cloning templates are given as

$$\mathbf{T}_A = \left[\begin{array}{c|c|c|c|c|c|c|c|c} -x_{iL} & \cdots & -x_{i2} & -x_{i1} & 0 & x_{i1} & x_{i2} & \cdots & x_{iL} \\ \hline -x_{rL} & \cdots & -x_{r2} & -x_{r1} & -x_0 + T_x + 2\mu & -x_{r1} & -x_{r2} & \cdots & -x_{rL} \\ \hline x_{iL} & \cdots & x_{i2} & x_{i1} & 0 & -x_{i1} & -x_{i2} & \cdots & -x_{iL} \end{array}\right] \tag{5.19}$$

$$\mathbf{T}_B = [1.0].$$

For binary case, the feedback operator (5.19) is reduced to

$$\mathbf{T}_A = [-x_L | \cdots | -x_2 | -x_1 | -x_0 + T_x + 2\mu | -x_1 | -x_2 | \cdots | -x_L]. \tag{5.20}$$

The network diagram of (5.19) is shown in Figure 5-7. Note that the elements of $\mathbf{T}_A$ may be updated dynamically as the characteristics of transmission medium changes. For the cloning templates of the form (5.19), the feedback operator is symmetric in a row, but has an opposite sign in adjacent two rows. However, the matrix $\mathbf{M}$ is symmetric and the CNN always finds a stable equilibrium as shown in Chapter II.

5.4 Hardware Architecture

5.4.1 Neural Network

As noted above, the Hopfield neural network and CNN for the MLSE share the same interconnect architecture with possible different neuron models. The number of neurons

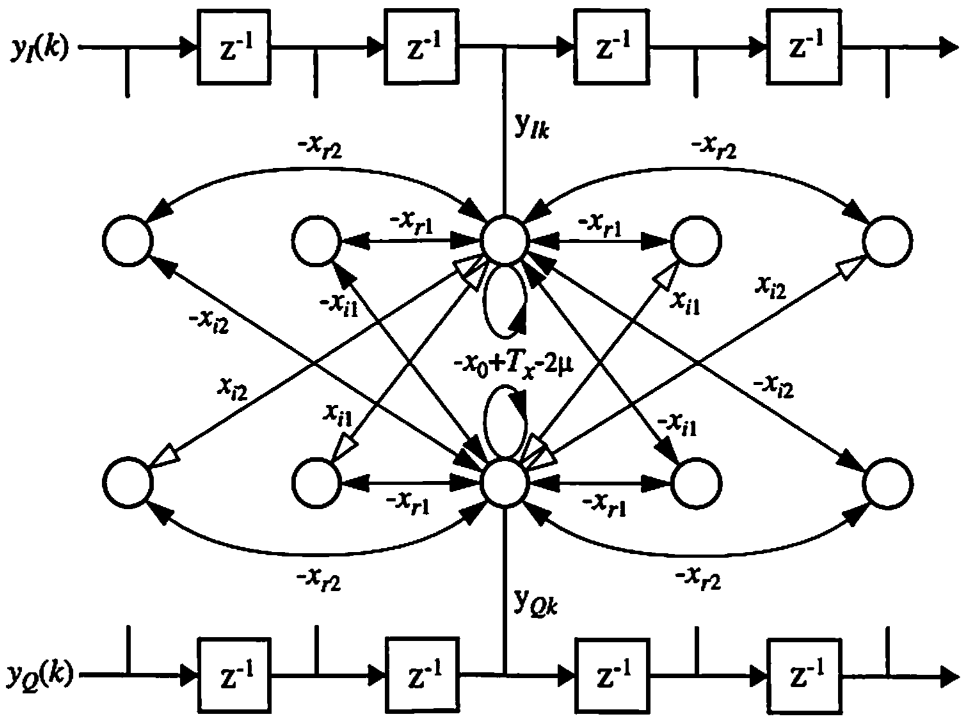


Figure 5-7: Cellular neural network for QPSK modulation (L=2).

N_N is equal to the number of bits in the sequence, e.g., $N_N = n$ in binary case and $N_N = 2n$ in QPSK case. The number of synaptic weights for $n >> L$ is

$$N_W \approx \begin{cases} n(2L+1) & \text{: real (binary)} \\ 2n(4L+1) & \text{: complex (QPSK).} \end{cases} \tag{5.21}$$

If synapse circuits with differential input terminals are used, N_W can be reduced to $n(L+1)$ and $n(2L+1)$, respectively. To accommodate the channels with slowly time-varying characteristics, the synapse circuits need to be programmable. The transconductance multiplier using the double-MOS linear resistors [126,127] may be a hardware-efficient element for this purpose. It requires only four transistors of small or moderate device sizes and its input and control terminals are both differential. Figure 5-8 shows a possible implementation of the i-th neuron and associated synapses. When all transistors in the double-MOS linear resistors are the same device-size ratio W/L, the differential current flowing two virtual ground nodes of operational amplifier is given for $i = 0,1,\cdots,n-1$, by

$$I_1 - I_2 = -K_0\left((x_0 - 2\mu)v_i + \sum_{k=1}^{L} x_k(v_{i+k} + v_{i-k})\right) + K_0 V_{C2} y_i, \tag{5.22}$$

where $K_0 = \mu C_{OX}(W/L)$ and V_{C2} is a fixed voltage. Here, μ is the mobility of electrons for n-channel MOSFETs. Some of v_{i-k} or v_{i+k} are zero for the edge neurons $i = 0,1,\cdots,L-1$ and $i = n-L,\cdots,n-1$. The differential output voltage is thus

$$V_1 - V_2 \equiv -R_i(I_1 - I_2) = -\frac{1}{K_0 V_{C1}}(I_1 - I_2), \tag{5.23}$$

where V_{C1} is the control voltage of the feedback double-MOS linear resistor, which determines the overall neuron gain. The equivalent resistance R_i together with the capacitance C performs a lossy integration represented by the differential equation (5.14). At the beginning of each operation, the capacitors are set to zero initial values by closing

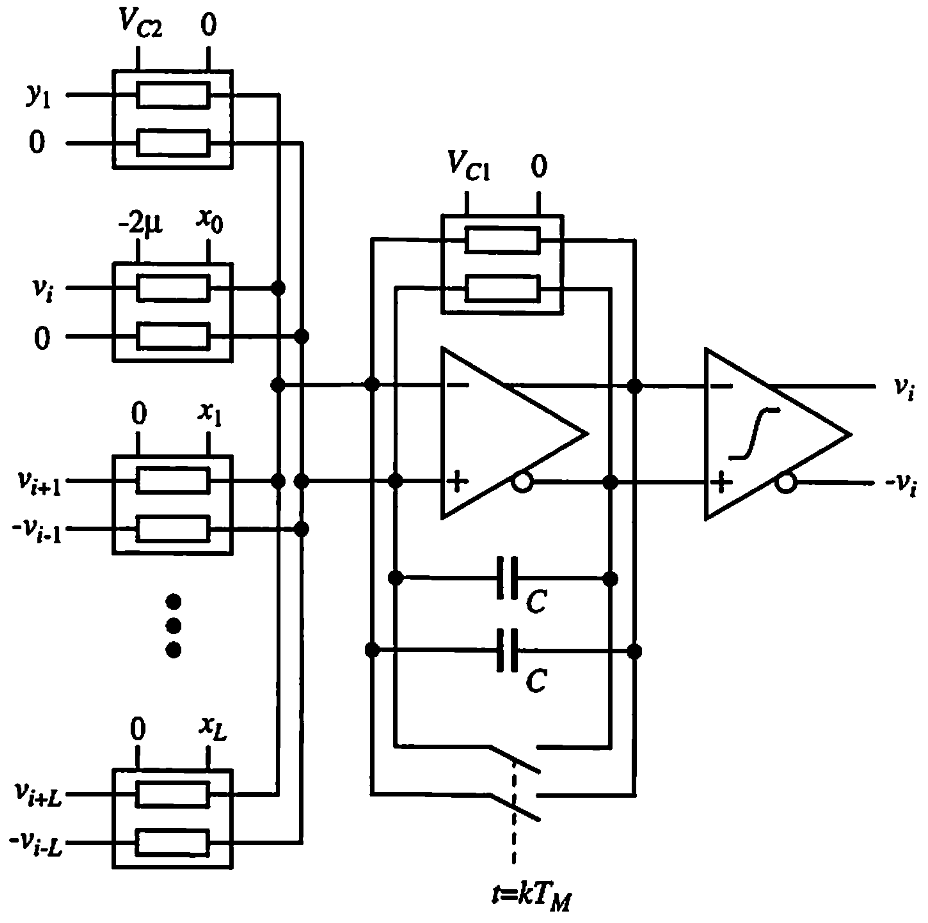


Figure 5-8: Implementation of *i*-th neuron and associated synapses using double-MOS linear resistors of Figure 6-10 (a).

switches connected in parallel. The nonlinear amplifier following the summing circuit has either the sigmoid or piecewise-linear transfer function. To generate both positive output v_i and negative output $-v_i$, a nonlinear amplifier with fully balanced outputs is required.

5.4.2 Tapped-delay Line

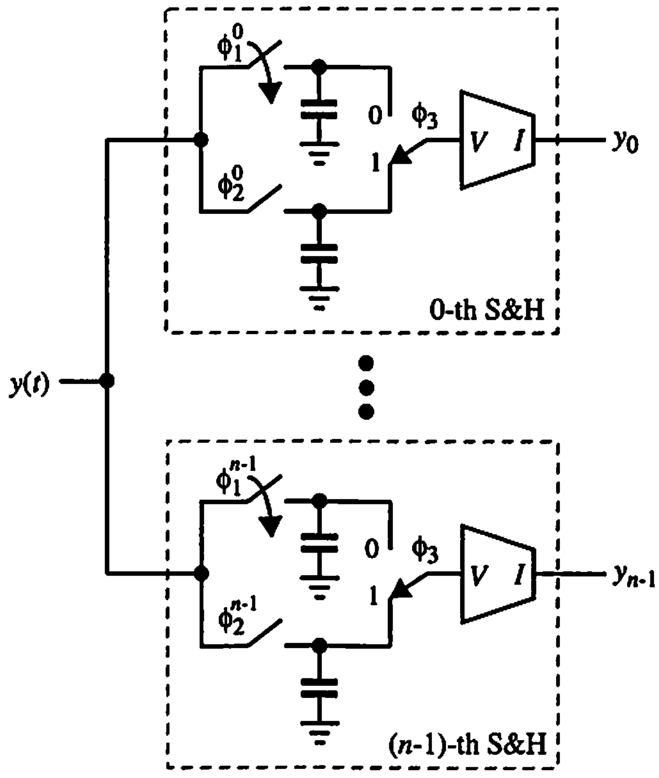
The analog tapped-delay line consisting of a cascade of sample-and-hold (S&H) circuits typically causes significant errors as discrete-time samples are shifted through, due to the accumulation of noise generated in a single S&H circuit. For continuous operations while the neural network is performing the MLSE, the delay lines must be pipelined. As shown in Figure 5-9 (a), n S&H circuits in parallel can perform the pipelined operation without the accumulation of noise. Each transconductor has two separate holding capacitors at the input, one of which is to accept new sample while the other holding the previous value as the input to the neural network in operation. The clock signals ϕ_1^k and ϕ_2^k, $k = 0,1,\cdots,n-1$, are used for sampling, and ϕ_3 is a control signal which is common to all S&H circuits for multiplexing two voltages on the holding capacitors. Note that either ϕ_1^k's or ϕ_2^k's are off for the period of T_M during which n shift operations occur. Figure 5-9 (b) shows the timing diagram of these clock signals.

5.4.3 Modular Architecture

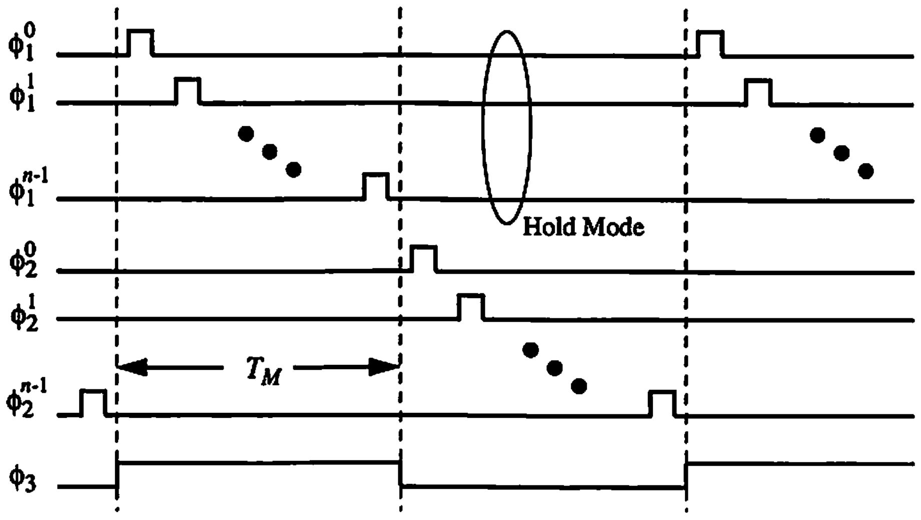
For the detection of especially long sequence, a multiple of MLSE neuroprocessors with modular architecture can be cascaded if the effects of interconnections between networks are negligible. If we define

$$\mathbf{W}_C^T = \left[\mathbf{W}_U^T \vdots \mathbf{W}^T \vdots \mathbf{W}_L^T\right] = \left[\mathbf{W}_U^T \vdots \mathbf{W} \vdots \mathbf{W}_L^T\right] \qquad (5.24)$$

where



(a) Schematic diagram.



(b) Timing diagram.

Figure 5-9: Pipelined analog tapped-delay line.

$$\mathbf{W}_U = -\begin{bmatrix} x_L & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ x_{L-1} & x_L & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ x_1 & x_2 & x_3 & \cdots & x_L & 0 & \cdots & 0 \end{bmatrix} \text{ and } \mathbf{W}_L = -\begin{bmatrix} 0 & \cdots & 0 & x_L & \cdots & x_3 & x_2 & x_1 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & x_L & x_{L-1} \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & x_L \end{bmatrix},$$

then the neural network with the transconductance matrix $\mathbf{W}_C$ can be divided into m subnetworks such that $\mathbf{W}_C = [\mathbf{W}_0 \mid \mathbf{W}_1 \mid \cdots \mid \mathbf{W}_{m-2} \mid \mathbf{W}_{m-1}]$, where $\mathbf{W}_k$, $k = 0,1,\cdots,m-1$, is $\mathbf{W}_0$ shifted or permutated down by k rows. Furthermore, if $\mathbf{\Omega}$ denotes $\mathbf{W}_0$ with all zero row(s) removed, i.e., $\mathbf{W}_0^T = [\mathbf{\Omega}^T \mid \underline{\mathbf{0}}]$, then the original network can be constructed by cascading m subnetworks, each having the transconductance matrix $\mathbf{\Omega}$. The uppermost L and bottommost L feedback inputs associated with $\mathbf{W}_U$ and $\mathbf{W}_L$ are set to zeros, which correspond to the leading and tailing zeros of the sequence to be estimated. In this way, the original network structure is preserved, i.e., $\mathbf{W}_U = \mathbf{W}_L = \underline{\mathbf{0}}$ and $\mathbf{W}_C = \mathbf{W}$. In QPSK case, the resulting network for the MLSE can be turned into a modular architecture by adding dummy rows to the transconductance matrix. In this case $\mathbf{W}_U$ and $\mathbf{W}_L$ defined above are required for every sub-matrix of W=X, as given in (5.5).

5.4.4 Hardware Annealing

Even with a correct mapping of the MLSE function onto a neural network, the desired optimal or near-optimal solutions are not guaranteed because a combinatorial optimization problem always involves a large number of local minima [1]. Therefore, in addition to the basic structure of the network, the annealing capability is provided to obtain the global minimum of the cost function (5.3) over all possible combinations of sequence. The hardware annealing [34,35,112,113] is a dynamic relaxation process for finding the optimum solutions in the recurrent associative neural networks such as Hopfield network and CNN. Near optimal solutions can be obtained by applying the hardware annealing technique for avoiding local minima problems which are inherent in combinatorial optimizations. Here, the optimum solution is defined as the one with the lowest value of the generalized energy function associated with the network.

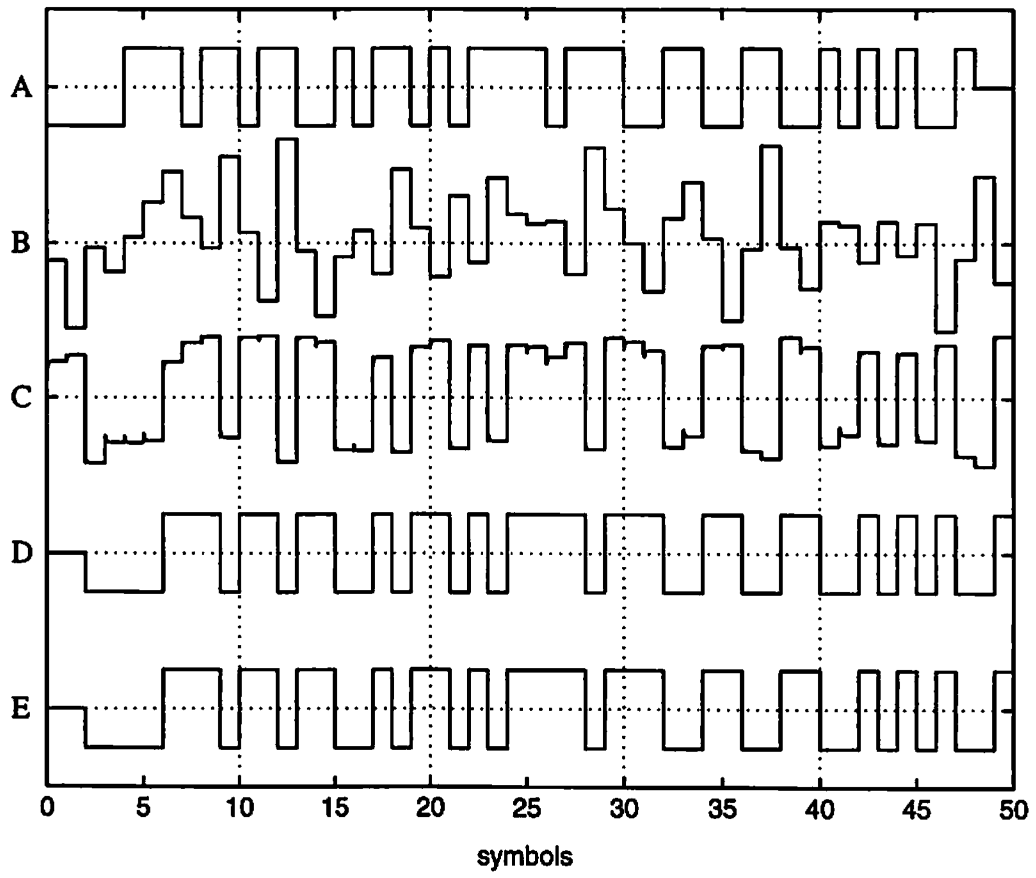
5.5 Simulation Results

The simulations of simple binary communication system with several ISI channels are performed by solving the differential equations (5.14). Random data sequence $u_n = \{u_k\}$, $k = 0,1,\cdots,n-1$, is generated and convolved with a channel response $h(k)$ which is assumed to be known exactly. Figure 5-10 shows various sample waveforms of the neural network MLSE (NN-MLSE). A 48-bit transmit sequence (A) is influenced by an ISI channel with $H(z) = 0.40823 + 0.8165z^{-1} - 0.40823z^{-2}$ and 6 dB of SNR. The received signal (B) is correlated with $h(k)$ and then used as input to a 50-neuron Hopfield neural network whose initial state values are zero, i.e., $u_i(0) = 0,\ \forall i$. Additional 2 neurons account for the leading or tailing effect of the ISI channel on the sequence. Waveform (C) is simply a concatenation of neuron states $u_i(t)$, $i = 0,1,\cdots,49$, during the network evolution. The steady-state outputs of unannealed and annealed NN's are shown in (D) and (E), respectively. The outputs of two leading neurons are discarded and set to zero values. As shown in the figure, both outputs are error-free.

To compare the proposed method with a conventional optimization technique, simulations are conducted on a binary communication system with ISI channel given by

$$H_m(z) = \frac{1}{\sqrt{1.25}}\left(1.0 + 0.5z^{-1}\right). \tag{5.25}$$

In this case, $x_0 = 1.0$, $x_1 = x_{-1} = 0.4$, $x_k = 0$ for $|k| \geq 2$, and the correlation matrix is given by



A: transmitted sequence, u(k)
B: received signal, r(k)
C: time evolution of neural network state values, ui(t), i=1,2,..,50
D: unannealed network output, vi, i=1,2,..,50
E: annealed network output, vi, i=1,2,..,50

Figure 5-10: Sample waveforms in neural network MLSE of 48-bit binary sequence (channel: h0=0.40825, h1=0.8165, h2=-0.40825, SNR=6dB).

$$\mathbf{X}=\begin{bmatrix} 1 & 0.4 & & & & & \\ 0.4 & 1 & 0.4 & & & \mathbf{0} & \\ & 0.4 & 1 & \ddots & & & \\ & & 0.4 & \ddots & 0.4 & & \\ & & & \ddots & 1 & 0.4 & \\ & \mathbf{0} & & & 0.4 & 1 & 0.4 \\ & & & & & 0.4 & 1 \end{bmatrix}, \qquad (n\text{-by-}n). \tag{5.26}$$

For both cases of $n = 20$ and $n = 100$, the minimum and maximum eigenvalues of X are approximately $\lambda_{\min} = 0.2$ and $\lambda_{\max} = 1.8$, respectively. Therefore, the coefficient μ can be chosen such that the additional value of synaptic weights in the main diagonal satisfies $T_x - 2\mu > T_x + 1.8 = 2.8$ for $T_x = 1$.

Table 5.1 shows the comparison of the NN-MLSE and conventional simplex method for finding 20-bit estimates. Since the simplex method is not efficient for concave optimization problems, the original cost function (5.4) is used and the signs of the final results are taken as the estimate. For each method, estimates of 1,000 symbols through 50 independent experiments are conducted. The results shows that the NN-MLSE has slightly better performance at low SNR values and is much faster than the simplex method in simulation execution time. By considering actual steady-state outputs at fractions (typically 0.2) of the average execution time 2.15 sec. per run, the speed advantage over the simplex method amounts to several hundreds in the simulations.

In Figure 5-11, error rates are plotted for the MLSE of unannealed and annealed NN's. In addition, two different neuron models, i.e., the sigmoid and piecewise-linear functions, are used to compare their effects on combinatorial optimization problem. Here, the channel model is the same as (5.25) and 100 simulations are performed independently on the sequences of length 100 ($n = 100$) for each signal-to-noise ratio (SNR) value. From the figure, it can be seen that the neural networks with

- piecewise-linear function, and

Table 5.1: Comparison of two optimization methods in MLSE of 1000 binary symbols for channel with inter-symbol interference and additive noise.

SNR (dB)	Errors/1000		Average Execution Time per run	
	Neural Network	Simplex Method	Neural Network	Simplex Method
0	198	238	2.15 sec.	286 sec.
3	111	164		
6	46	63		
9	17	16		
12	3	3		
15	0	0		
18	0	0		
21	0	0		
24	0	0		
27	0	0		

Test Conditions:

1. Channel characteristics: $H(z)=0.89443+0.4472\ z^{-1}$.
2. $n=20$, $L=2$.
3. - Neural network: 20-neuron Hopfield model.
   - Simplex method (part of commercial tool): 20-variable search for minimum of cost function J.
4. 50 independent runs for each method.

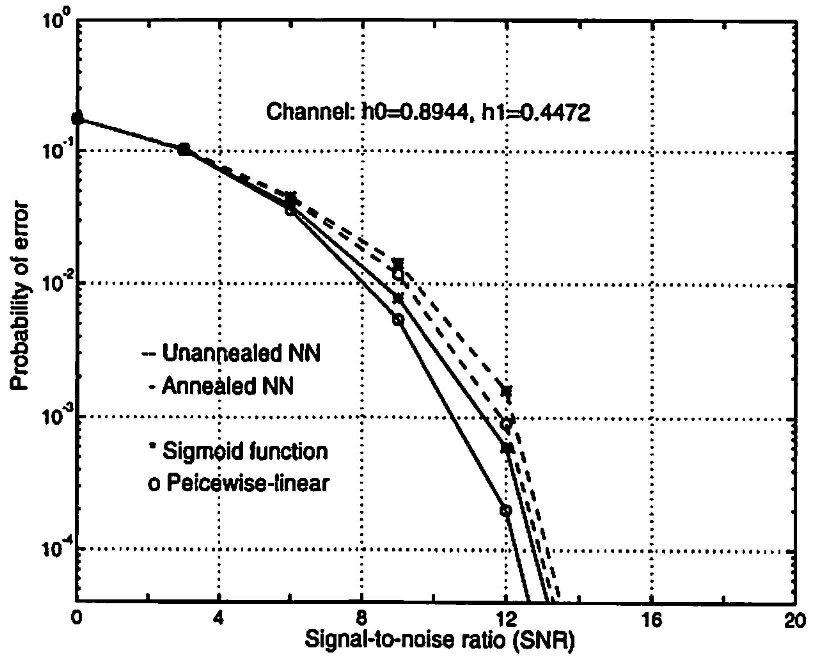


Figure 5-11: Effects of neuron models and hardware-annealing on MLSE performance (L=1).

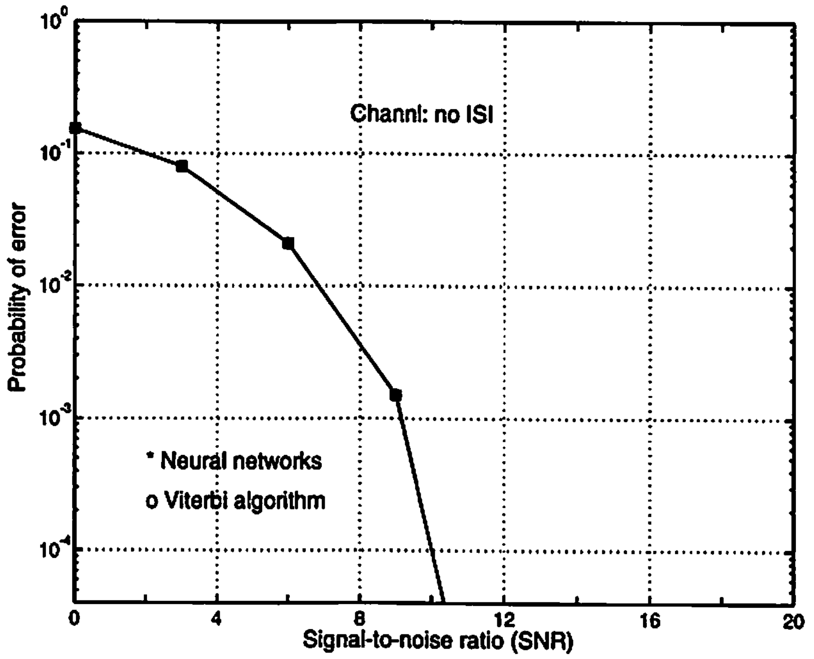


Figure 5-12: Error rate performances of neural network MLSE and Viterbi algorithm (L=0).

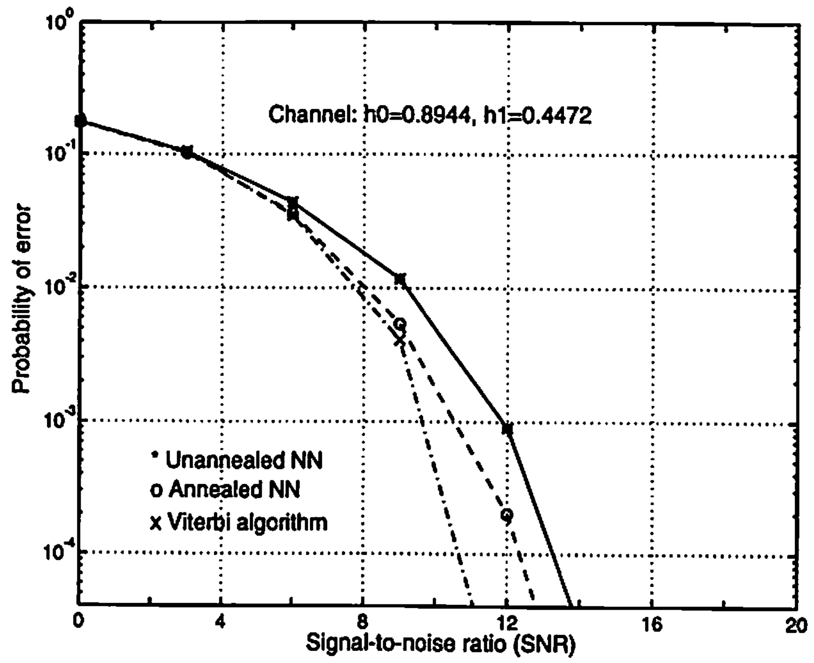
• annealed operation

have $1 \sim 1.5\,dB$ better performance over those with others. The sigmoid nonlinearity not only reduces the number of local minima but also provides chances of smoothing down the desired global minimum.

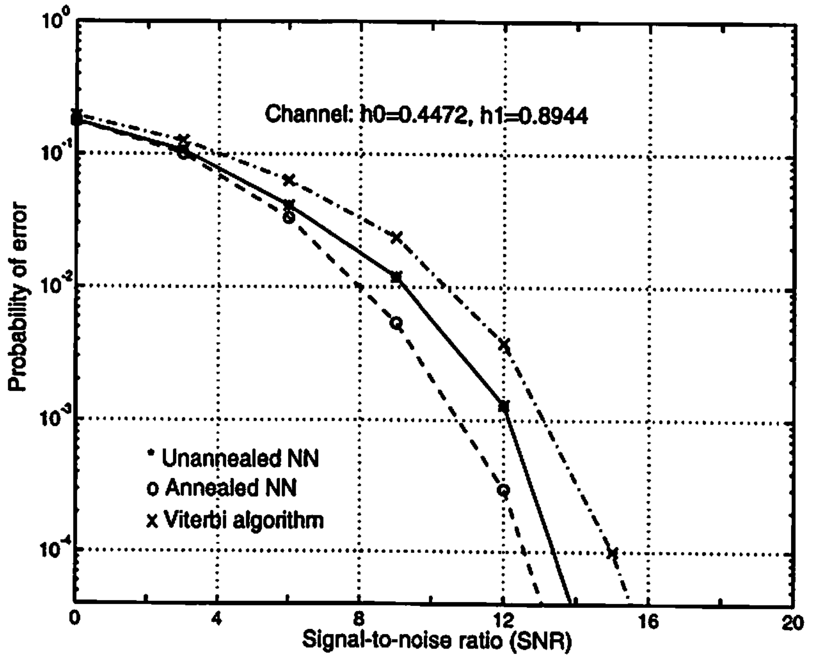
For the case when no inter-symbol interference is present, the error rate performances of the proposed method and Viterbi algorithm (VA) are shown in Figure 5-12. All CNN models and VA have the same error rates in this case. This might be the consequence of independence between received samples. However, since the correlator in Figure 5-3 just bypasses the input $(\bar{h}(k) = \delta(k))$, the performances are somewhat poor than theoretical values [119] $P_e = 0.5\,\mathrm{erfc}\left(\sqrt{\gamma_b}\right)$ which is obtained through an ideal continuous-time matched-filtering of received signal prior to the sampling. Here, $\mathrm{erfc}(\cdot)$ is the complementary error function and γ_b is received SNR per information bit.

Next, the error rates of MLSE by unannealed and annealed NN's are shown in Figure 5-13 (a) for two-ray minimum-phase channel (5.25) and in Figure 5-13 (b) for two-ray nonminimum-phase channel $H_n(z) = \left(0.5 + z^{-1}\right)/\sqrt{1.25}$, respectively. For comparison, the results of VA are also shown in the figures. In the simulations for the second channel $H_n(z)$, it is assumed that the decisions are made in reference to direct received samples which are half the magnitude of delayed versions. As expected, an annealed NN has better performance than an unannealed NN in both cases. It might be worthwhile mentioning that the NN-MLSE for a minimum-phase channel $H_m(z)$ is less efficient than the VA at the moderate values of SNR, but does not suffer much from the nonminimum-phase characteristics of the channel.

In addition to the network and neuron models, there are three ways to provide the input y to a CNN, i.e.,



(a) Two-ray minimum-phase channel.



(b) Two-ray nonminimum-phase channel.

Figure 5-13: Error rate performances of neural network MLSE and Viterbi algorithm (L=1, n=100, 10,000 binary symbols).

(a) weighted feedforward input $\boldsymbol{v}_u$,

(b) initial state $\boldsymbol{v}_x(0)$, and

(c) combination of these two.

The hardware annealing also can be incorporated with the network in two different ways;

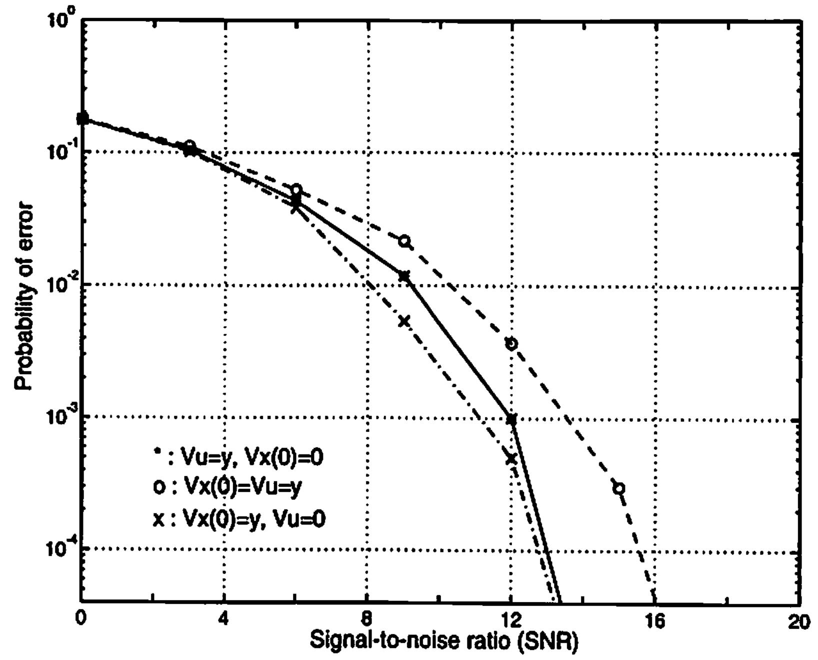
(1) annealed operation alone, and

(2) annealed operation following unannealed operation.

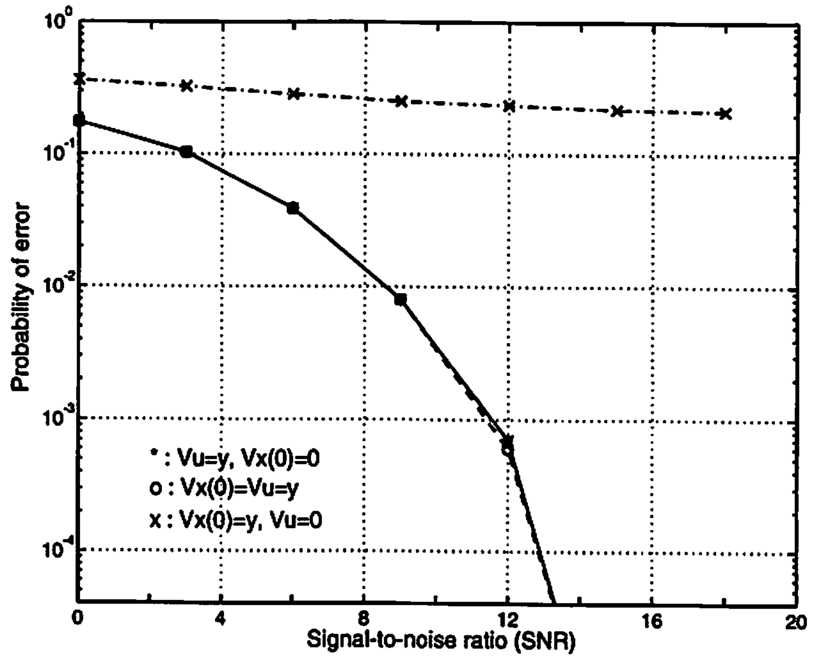
In the case (2), the final state values become the initial states of annealed operation. The results of Figure 5-13 are done by the case (a) and annealing method (2). With the annealing method (1), the difference of three methods are shown in Figure 5-14 (a) and (b) for unannealed and annealed conditions, respectively. For the method (a), the initial state values are zero and an unannealed NN has less chances of getting stuck in local minima as noted in Chapter IV. Thus, the improvement by hardware annealing is small in this case. The method (b) is the winner of all cases in unannealed operations. However, because the annealing in this case results in the optimal solutions of a cost function $\hat{J}_n = 0.5\boldsymbol{u}^T\mathbf{X}\boldsymbol{u}$ instead of (5.13), the results are completely unsatisfactory as shown in the figure. See Figure 4-11 (d). For the case (c), the error rate is somewhat higher than that of the case (a) in unannealed NN-MLSE, but is almost the same in annealed operations. From these simulation results, we may have the following preferences;

- method (b) ($\boldsymbol{v}_u = \mathbf{0}$ and $\boldsymbol{v}_x(0) = \boldsymbol{y}$) for unannealed NN-MLSE and
- methods (a) ($\boldsymbol{v}_u = \boldsymbol{y}$ and $\boldsymbol{v}_x(0) = \mathbf{0}$) and (2) for annealed NN-MLSE.

Similar results can be obtained for the model of two-ray nonminimum-phase channel as shown in Figure 5-15.

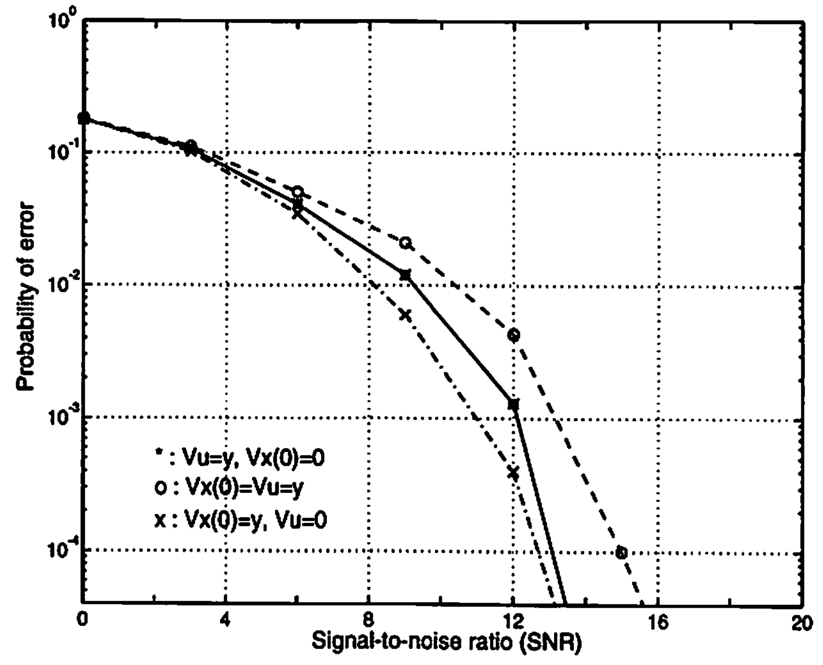


(a) Unannealed neural network.

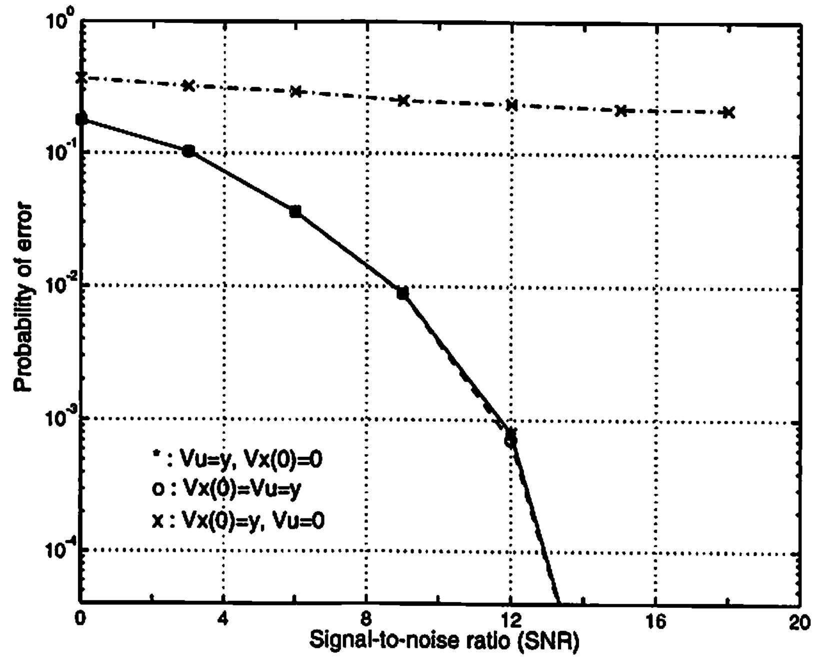


(b) Annealed neural network.

Figure 5-14: Comparison of methods for providing input y (minimum-phase 2-ray channel model, L=1, n=100, 10,000 binary symbols).



(a) Unannealed neural network.



(b) Annealed neural network.

Figure 5-15: Comparison of methods for providing input y (nonminimum-phase 2-ray channel model, L=1, n=100, 10,000 binary symbols).

5.6 Discussions

The maximum-likelihood estimation of signals in interference and white Gaussian noise environments can be efficiently performed by utilizing the collective computational behaviors of artificial neural network. Both the Hopfield neural network and CNN models can be used as an approximate and exact realization of the MLSE, respectively. Several issues such as network models, neuron models, and methods of supplying input to the network are investigated in terms of the probability of error. It is demonstrated that artificial neural network is an efficient way of direct realization of the MLSE. In addition, the performance of the NN-MLSE can be improved by paralleled hardware annealing technique which is developed for high-speed, real-time neural networks.

The hardware architecture of proposed method is discussed regarding circuits of neural network and analog tapped-delay line, modular architecture for large-scale network, and hardware annealing. It has the following important properties;

1. *massively paralleled operations,*
2. *circuit complexity proportional to length L,*
3. *space-independent local interconnections*, and
4. *simple modularity.*

These are very desirable in VLSI implementations of the network. The MLSE of a sequence is done in a single evolution interval of the neural network, which can be orders of microseconds or fractions of a microsecond, the speed of MLSE is typically limited by supporting circuitry such as analog tapped-delay line which runs at a symbol rate. Moreover, the operating speed of recurrent associative neural networks are almost independent of network size.

Chapter VI

CMOS Design of CNN with Programmable Synapses

6.1 Introduction

A cellular neural network (CNN) is a locally connected, massively paralleled computing machine. Under the mild conditions [59], a CNN autonomously finds a stable solution for which the Lyapnov function of the network is locally minimized. The CNN's can be used in many computation-intensive applications such as image and signal processing. Moreover, the quadratic nature of the Lyapnov function allows us to map it into optimization problems [4,59]. Several structural variations of the continuous-time, shift-invariant, rectangular-grided network which was introduced by Chua and Yang [59,60] in 1988, have been reported. Among them are discrete-time CNN [64], CNN with nonlinear and delay-type templates [65], etc.

Local interconnection and simple synaptic operators are the most attractive features of the CNN for VLSI implementation in high-speed, real-time applications. Several hardware implementations of the CNN have been reported in the literatures [89-99]. In this second part of the chapter, the CMOS VLSI design of a continuous-time, shift-invariant CNN with digitally-programmable operators is considered. In addition, the circuits for hardware annealing [34,35,112,113] is included to provide the flexibility of the network in a variety of applications. Since its systematic introduction, many issues have been addressed and remarkable results have been achieved. Under the mild conditions, a CNN is always stable and finds a locally optimum output in the steady state. However, the output so obtained may not be a globally optimum solution in terms of the generalized network energy, because there may exist multiple minima at which the energy is minimized locally. This might, in turn, cause sub-optimal network operations as long as the objective is to map the input signal in one space to the output in another space by minimizing the cost function involved.

6.2 Hardware architecture

The differential equations (2.15) governing a CNN is rewritten here for convenience

$$C\frac{dx}{dt} = -T_x x + \mathbf{A}y + \mathbf{B}u + I_b w, \tag{6.1}$$

where $y \in D^N = \{y \in R^N : -1 \le y_k \le 1; k = 1,2,\cdots,N\}$, and **A** is an N-by-N real symmetric matrix defined as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & \mathbf{0} & \\ \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{A}_1 & & \\ & \mathbf{A}_1 & \mathbf{A}_0 & \ddots & \\ & & \ddots & \ddots & \mathbf{A}_1 \\ & \mathbf{0} & & \mathbf{A}_1 & \mathbf{A}_0 \end{bmatrix}. \tag{6.2}$$

Here $w = [1\ 1\ \cdots\ 1]^T$ is an N-by-1 constant vector. In (6.2), $\mathbf{A}_0$ and $\mathbf{A}_1$ are two $m \times m$ Toeplitz matrices with elements determined by a given cloning template. The synapse weights of the shift-invariant CNN can be described by the feedback and feedforward cloning templates

$$\mathbf{T}_A = \begin{bmatrix} a_2 & a_1 & a_2 \\ a_1 & a_0 & a_1 \\ a_2 & a_1 & a_2 \end{bmatrix} \text{ and } \mathbf{T}_B = \begin{bmatrix} b_2 & b_1 & b_2 \\ b_1 & b_0 & b_1 \\ b_2 & b_1 & b_2 \end{bmatrix}, \tag{6.3}$$

respectively, where all elements represent the normalized numbers to $T_x = 1/R_x$, and a_0 = $A(i,j;i,j)/T_x > 1$. Then $\mathrm{A}_0 = toeplitz(a_0,a_1)$ and $\mathrm{A}_1 = toeplitz(a_1,a_2)$, where $toeplitz(a,b)$ is defined as the Toeplitz matrix with all a's in the main diagonal, and all b's in above and below the main diagonal. The input vector $\mathbf{v}_u$ and the matrix **B** can be defined in a similar way. Note that the matrix **A** or **B** is not Toeplitz, but always real symmetric as long as the cloning templates are shift-invariant. Let u, x and y be the input, state and output voltages normalized to $v_{yij}(t = +\infty)$. Then, the maximum value of x in the steady state is the sum of absolute values of all inputs from the neighborhood cells,

$$\begin{aligned} |x_{\max}| &= \sum_{i,j=1}^{3} |\mathrm{T}_A(i,j)| + \sum_{i,j=1}^{3} |\mathrm{T}_B(i,j)| + |x_0| \\ &= a_0 + 4(|a_1| + |a_2|) + b_0 + 4(|b_1| + |b_2|) + |x_0|, \end{aligned} \tag{6.4}$$

where $|u| \le 1$, $|y| \le 1$, and $x_0 = I/T_x$. For example, if $a_0 = 2$, $a_1 = a_2 = b_0 = 1$, and $b_1 = b_2 = 0$, then $|x_{\max}| = 11$. The neuron cell should be able to handle the state voltage of the range $|x| \le |x_{\max}|$.

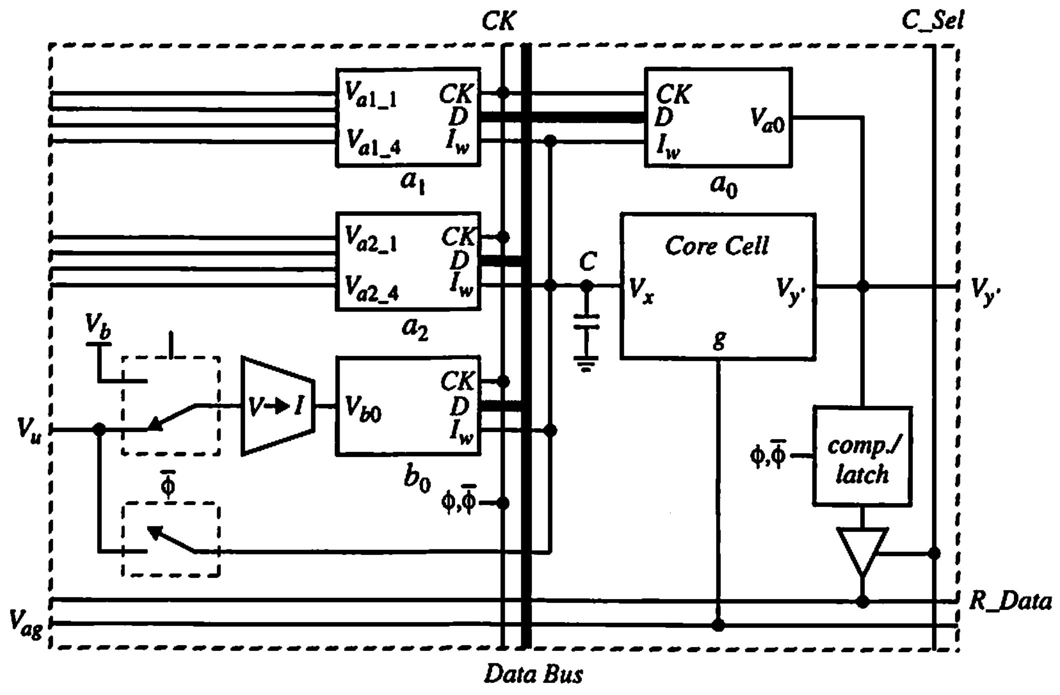
The shift-invariant architecture of the CNN is the most advantageous feature in realizing the network in electronic circuits using the VLSI technologies. The basic cell consists of a summing circuit for the right hand side of (2.12) and nonlinear function generation circuit. The neuron cells and operator weights are the basic elements of the CNN hardware. To accommodate a large-size neural network in a single VLSI chip, it is especially important to design the low-complexity and high-accuracy circuits for the efficient realization of neurons and weights. The shift-invariant property of the operator weights in the CNN implies that the number of distinct weights are much less than that of neighboring cells and the network shares the same unit consisting of a neuron cell and synaptic weights to and from the neighboring cells. If the symmetry of the weights is assumed except in the edges, the number of the r-neighborhood cells $C(k,l) \in N_r(i,j)$ of the cell $C(i,j)$, $\forall i,j$, is

$$N_C(r) = 1 + 4r(r+1), \tag{6.5}$$

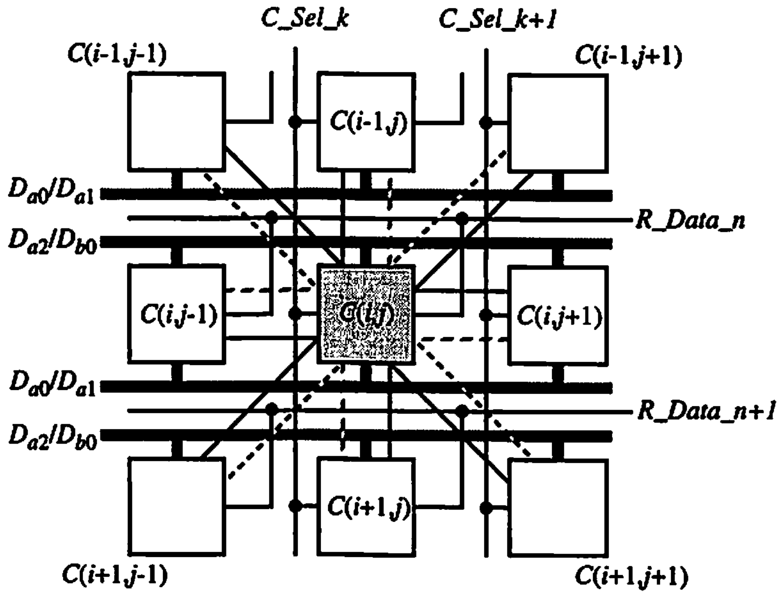
while the number of distinct weights is

$$N_W(r) = \frac{1}{2}(r+1)(r+2). \tag{6.6}$$

The network considered in this design is a continuous-time, rectangular-type CNN with $r = 1$. A CNN unit consists of a core neuron cell, synaptic weights, input/output circuits, and digital interfaces, as shown in Figure 6-1 (a). To construct a complete CNN, a multiple of the units can be arranged in an n-by-m rectangular grid with appropriate interconnections with the neighborhood cells as shown in Figure 6-1 (b). The digital interfaces provided are control data buses and read data lines. Four control buses for weights a_0, a_1, a_2, and b_0 are 5-bit wide each. On the other hand, a read data line is common to all the cells in a row, and a column select line is activated at a time to read cell outputs in a specific column. The data for synaptic weights are written into operator registers and the network outputs are read out of the cell output latches. This method reduces the number of output signals and, because the read operations can take place during the next network operation through the direct memory access (DMA), there is no speed penalty in a moderate-size network. Figure 6-2 shows a detailed block diagram of synaptic weights and input circuits. There are three kinds of feedback synaptic weights, i.e., a_0, a_1, and a_2 in (6.3), each of which is independently programmable via control bus lines. There is only one feedforward synapse in each cell for the direct input b_0 of (6.3). In many applications, scalar feedforward templates, i.e., $b_0 \neq 0$, $b_1 = b_2 = 0$, are used. If necessary, non-scalar $\mathbf{T}_B$ may be implemented externally with $b_0 = 1$. The operator weight for b_0 can be configured as an independent programmable source with the input u



(a) Block diagram of single unit consisting of core cell, synapses,and input/output.



(b) Communication with neighborhood cells.

Figure 6-1: Architecture of CNN.

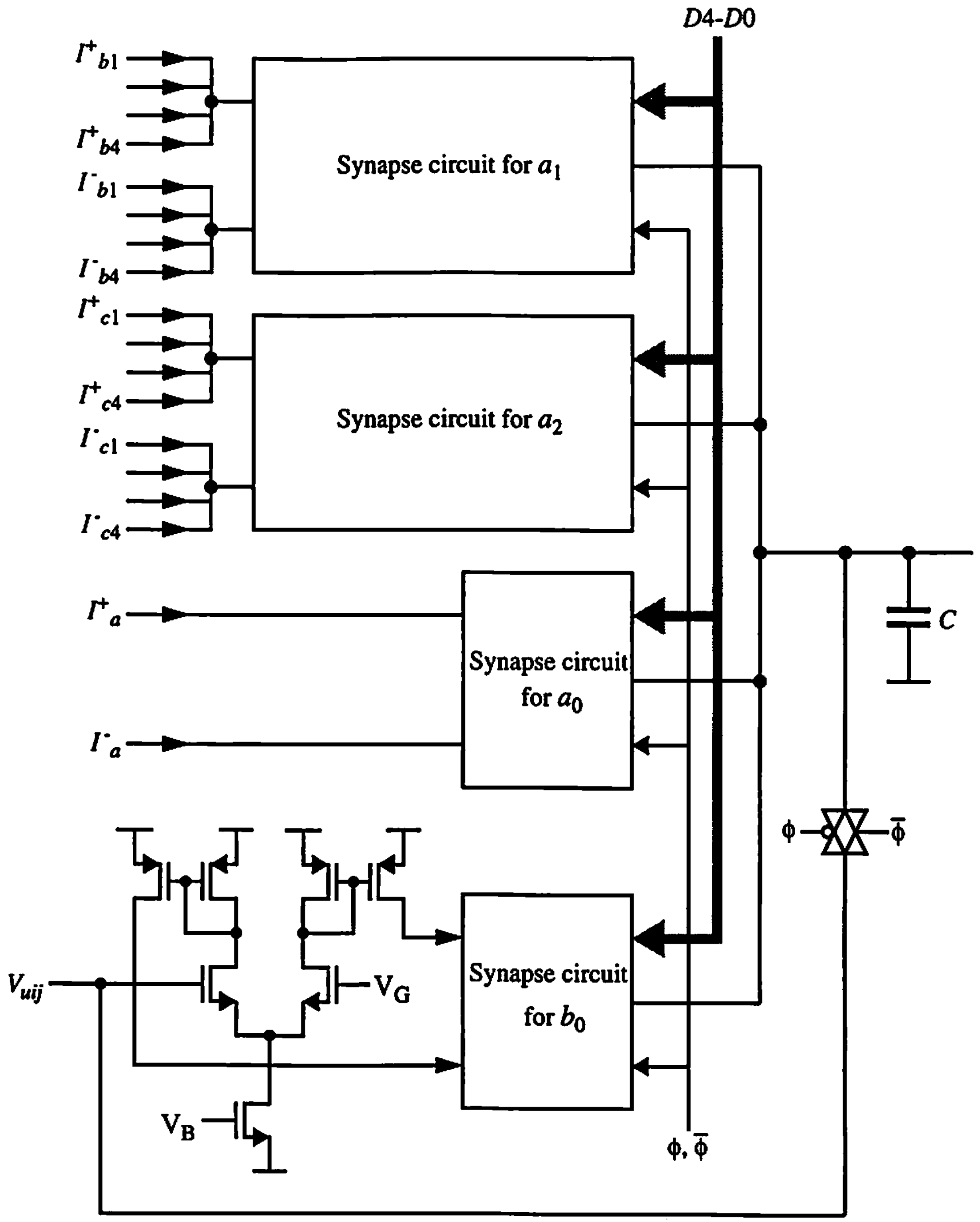


Figure 6-2: Schematic of input and synapse weights.

connected to a fixed bias voltage. In other words, the input u can be written into the network through the digital interface or a simple multilayer CNN can be realized in a time-multiplexed fashion. In each operation, the cell needs to be initialized to a value $-1 \leq x(0) \leq +1$ where, depending on the applications, $x(0)$ can be a direct or weighted input $x(0) = v_u$, or zero. One end of the capacitor C is switched to the voltage $x(0)$ during the initialization operation. At the same time, the outputs of synapses go into high-impedance state by control signals ϕ and $\overline{\phi}$, and the state node is connected to V_{GND} for avoiding possible spurious operation caused by the closed loop with the parasitic capacitance at the state node.

In addition to the basic structure of the network, the annealing capability is provided to accommodate the applications in which the optimal solutions of energy function are required. The hardware annealing is a dynamic relaxation process for finding the optimum solutions in the recurrent associative neural networks such as Hopfield network and CNN. Here, the optimum solution is defined as the one with the lowest value of the generalized energy function associated with the network. In addition, it has been also shown that it provides a enough stimulation to the network such that the output can cross over energy barriers that might exist among the local minima. The hardware annealing is performed by controlling the gain of the neuron, which is assumed to be the same for all neurons throughout the network. After the state is initialized $x = x(0)$, the initial gain at time $t = 0$ can be set to an arbitrarily small, positive value such that $0 \leq g(0) << 1$. It then increases continuously for $0 < t \leq T_A$ to the nominal gain of 1. The maximum gain $g_{max} = 1$ is maintained for $T_A < t \leq T$, during which the network is stabilized. When the hardware annealing is applied to a CNN by increasing the neuron gain $g(t)$, the transfer function can be described by $v_{yij}(t) = f\left(g(t)v_{xij}(t)\right)$ or simply $y = f(gx)$. Note that the saturation level is still $y = +1$ or -1 and only the slope of $f(x)$ around $x = 0$ varies. By using the normalized variables in a vector and matrix notations, (2.12) can be re-written as

$$C\frac{dx}{dt} = \mathbf{A}y - T_x x + b, \tag{6.7}$$

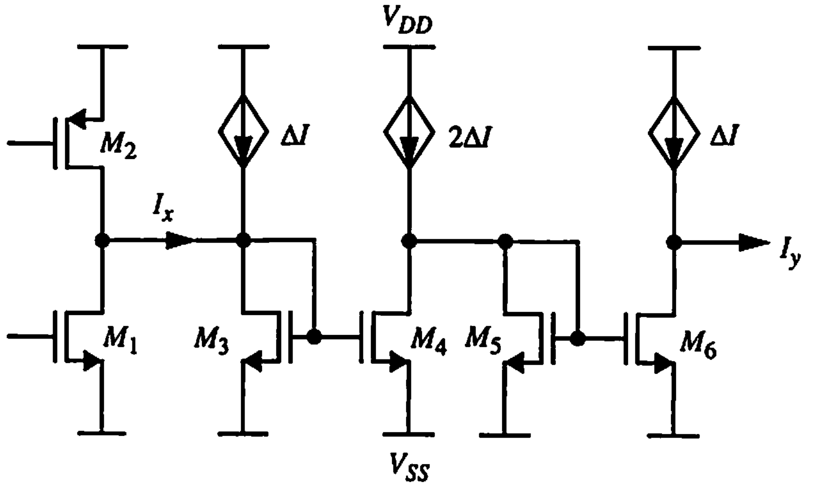
where $y = f(gx)$ and $b = \mathbf{B}u + I_b w$ for a constant vector $w = [1\ 1\ \cdots\ 1]^T$. In (6.7), **A** and **B** are two real N-by-N matrices determined by given cloning templates $\mathbf{T}_A$ and $\mathbf{T}_B$, respectively. For the shift-invariant CNNs, they are real symmetric. Initially, the gain g is small so that the network can be linearized. For the piecewise linear function, the assumption is exact until some of neurons are saturated. In this case, $y = gx$ and (6.7) becomes

$$C\frac{dx}{dt} = \mathbf{M}_g x + b, \tag{6.8}$$

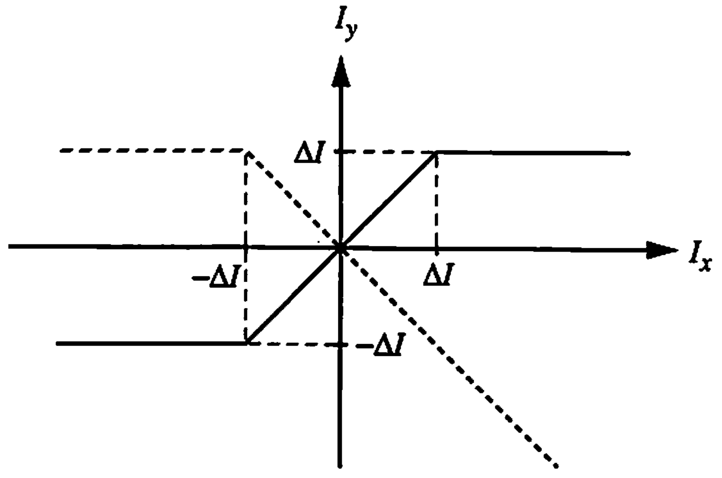
where $\mathbf{M}_g = g\mathbf{A} - T_x\mathbf{I}$ for an N-by-N identity matrix **I**. The process of finding the optimal solutions takes place during the change of $\mathbf{M}_g$ from negative definite to indefinite matrix, as the annealing gain g increases.

6.3 Circuit Design

The first circuit designed is an unannealed CNN with fixed synaptic weights. To simplify the design, the current-mode circuits [92-94] are used. The summation of weighted currents is simply done by a wired-ORing of all signals and the fixed weights can be accomplished by appropriate transistor sizing. A cell circuit consists of a constant input resistance negative current conveyer followed by the piecewise-linear circuit. The piecewise-linear function is achieved by cascading two current limiters as shown in Figure 6-3. The limiting operation of the input current denoted by I_x first takes place at a negative value $I_x = -\Delta I$ and positive value $I_x = \Delta I$. For $I_x \le -\Delta I$, no currents flow



(a) Circuit schematic of piecewise linear function.



(b) Transfer characteristics of two current limiters in cascade.

Figure 6-3: Piecewise linear function.

through the transistors M_3 and M_4. Therefore, $I_{DS5} = I_{DS6} = 2\Delta I$ and $I_y = -\Delta I$, where I_{DS5} represents the drain-to-source current of M_5, and so on. For $I_x > -\Delta I$, $I_{DS3} = I_{DS4} = \Delta I + I_x$ and $I_{DS5} = I_{DS6} = \Delta I - I_x$ producing the output current $I_y = \Delta I - I_{DS6} = I_x$. However, if $I_x \geq \Delta I$ then $I_{DS5} = I_{DS6} = 0$ and $I_y = \Delta I$. The negative current conveyer [128] circuit shown in Figure 6, provides a constant input resistance whose value is independent of the input current. The transistors M_1 and M_2 operate in the saturation region and have the same geometry and W/L value. The linear $I-V$ conversion is performed by M_1 and M_2 and the equivalent input resistance is given by

$$R_x = R_{eq} = \frac{1}{K(W/L)(V_2 - 2V_T)}. \tag{6.9}$$

If we denote the currents through M_1 and M_2 by I_1 and I_2, respectively, then $I_1 = I_2 + I_{in}$. The range of input I_{in} for proper current conveying operation is given by $|I_{in}| < K(W/2L)(V_2 - 2V_T)^2$, where V_2 is the gate voltage of M_2 for biasing. When I_1 and I_2 are mirrored by transistors $M_3 - M_5$ and subtracted at the output node, the output current is given by

$$I_{out} = I_2 - I_1 = -I_{in}. \tag{6.10}$$

Therefore, the circuit of Figure 6-4 is a negative, bidirectional current conveyer with constant input resistance R_{eq}. The equivalent input resistance R_{eq} corresponds to R_x in (2.12) and determines the time constant with which the network evolves. Therefore, it is desirable for a neuron cell to have a controllable resistance R_x. Figure 6-5 shows a simple current comparator using a cascade of two inverters. The input current I_y and reference set by V_b are compared and a logic voltage V_y^L is obtained. Figure 6-6 shows a complete schematic diagram of a neuron cell. The synaptic weighting is carried out by $M_{11} - M_{14}$ for a_0 and $M_{15} - M_{16}$ pair for a_1. Note that four copies of a current mirror are used to provide the weight a_1 for four neighborhood cells. All the current including the

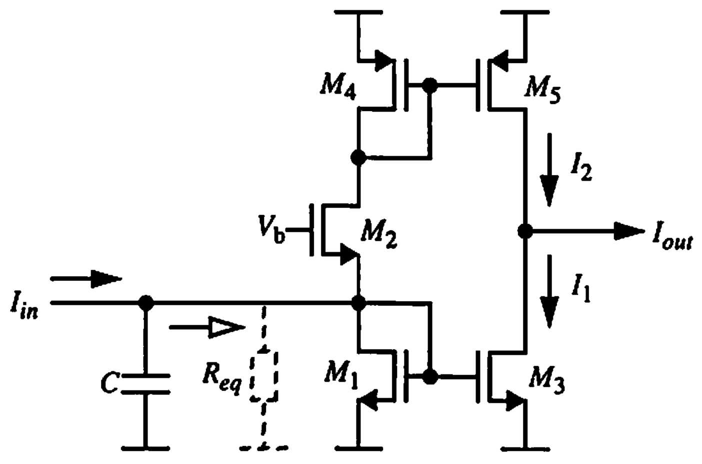


Figure 6-4: Constant input impedance negative current conveyer.

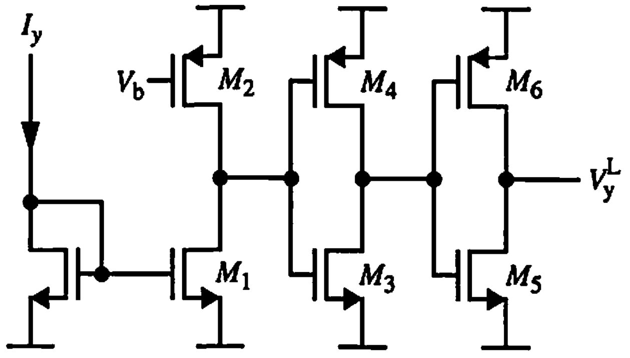


Figure 6-5: Current comparator for logic output of a cell.

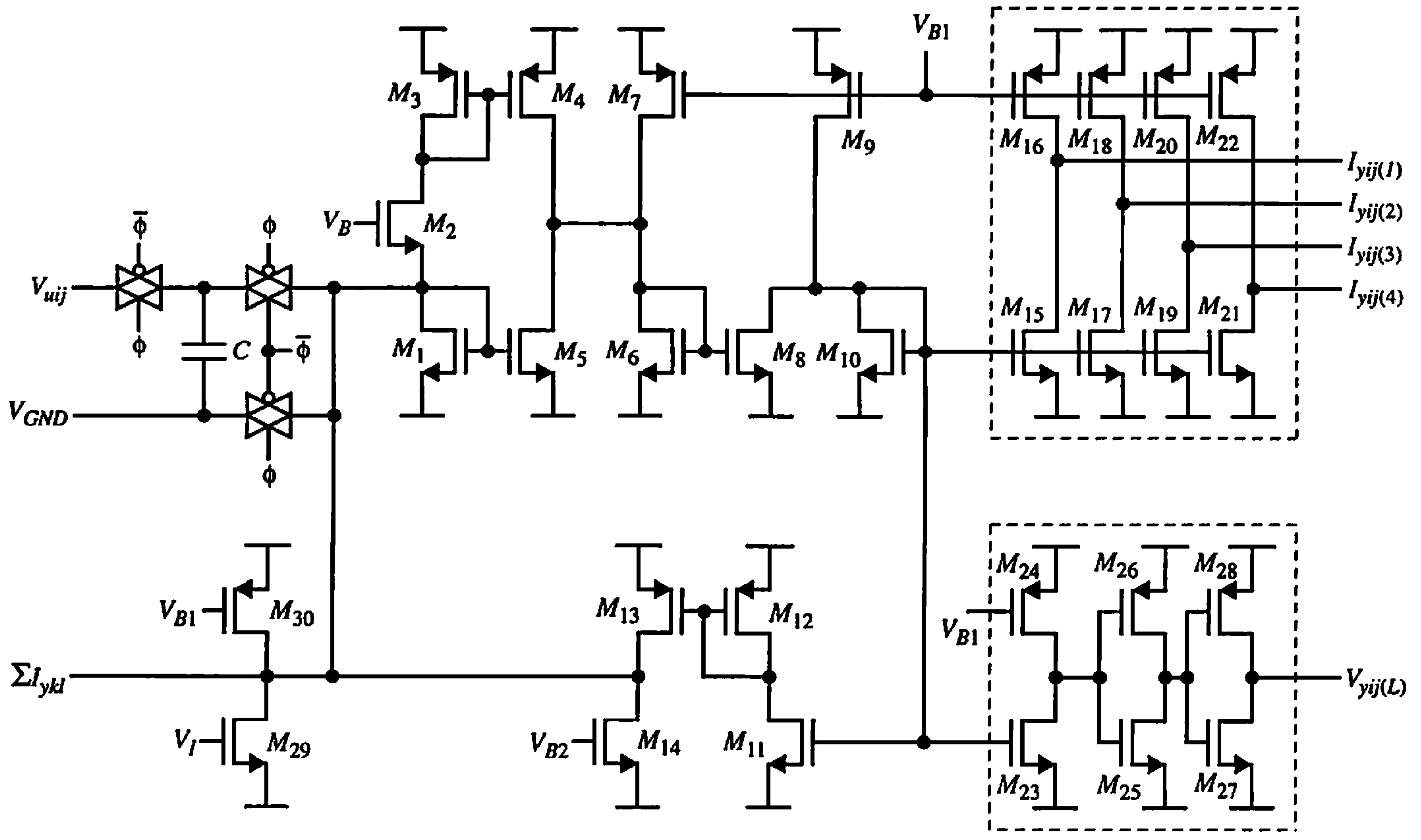


Figure 6-6: Neuron schematic of fixed-weights CNN.

feedback current, bias current, and those from the neighborhood cells are summed together at the input node, i.e., the drain of M_1. Transistors $M_{29} - M_{30}$ pair provides a bias current which is set by the bias voltage V_I. A simple current comparator circuit consisting of transistors $M_{23} - M_{28}$ produces the logic output $V_{yij(L)}$ which represents the sign of the neuron output. The SPICE simulation results of the piecewise-linear function together with the synaptic weights are shown in Figure 6-7.

The core cell of an annealed CNN consists of a summing circuit, analog multiplier, and nonlinear function circuit as shown in Figure 6-8. The synaptic weights are digitally programmable through binary-weighted current switches. When the weights need to be programmable, the current-mode circuits also provide the simplicity over voltage-mode circuits because $N_W(r) << N_C(r)$ for $r \geq 1$ as indicated by (6.5) and (6.6).

A. Transresistance multiplier: The hardware annealing is performed by the pre-multiplication of the state v_{xij} by the gain control function g before the nonlinear function $f(\cdot)$ takes place. Thus an analog multiplier is placed between the summing circuit and nonlinearity as shown in Figure 6-8 (b). Since g is positive, the two-quadrant multiplier is employed. The analog multiplier [124,125] is a basic building block in many analog signal processing applications including artificial neural networks, and can perform both multiplication and nonlinear function. However, a care must be taken in choosing the circuit, because some of the analog multipliers often result in the post-multiplication $g \cdot f(v_{xij})$ as well as the desired function $f(g \cdot v_{xij})$. In other words, as the gain decreases, the linear operation region and saturation levels are also reduced. In this case, the minimum gain $g_{\min}$ can not be made small arbitrarily.

The basic element of the proposed circuit is the double-MOS differential resistor [126,127] operating in triode region. As shown in Figure 6-9, the differential MOS linear resistor has two input terminals driven by the voltages V_1 and V_2, and two output

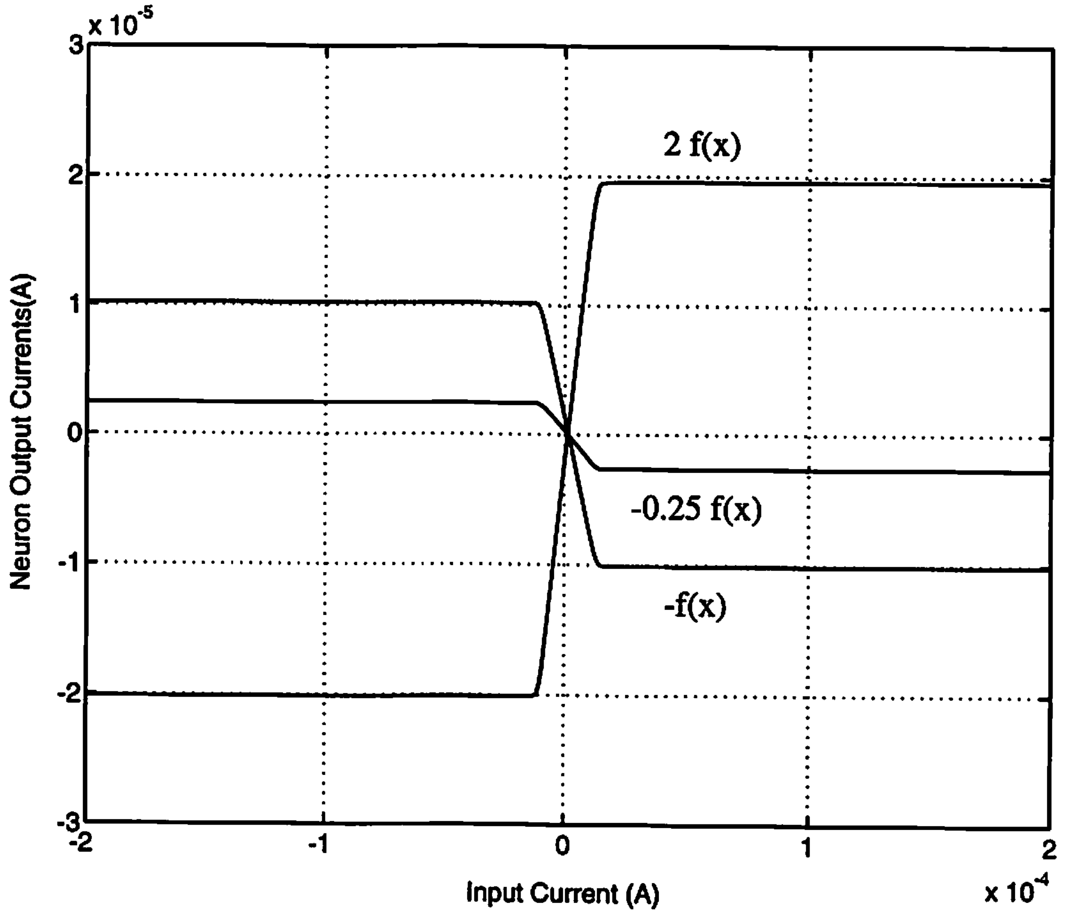
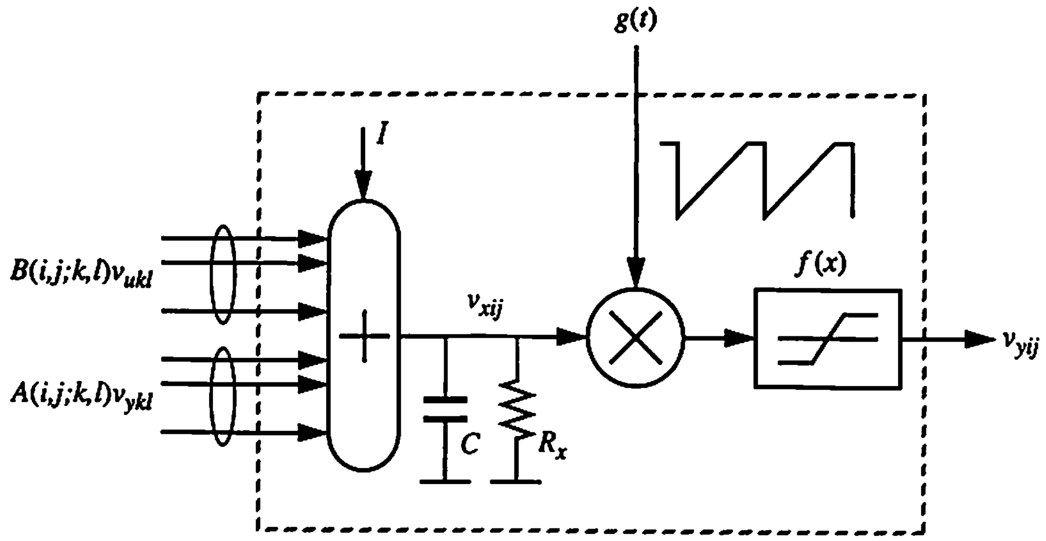
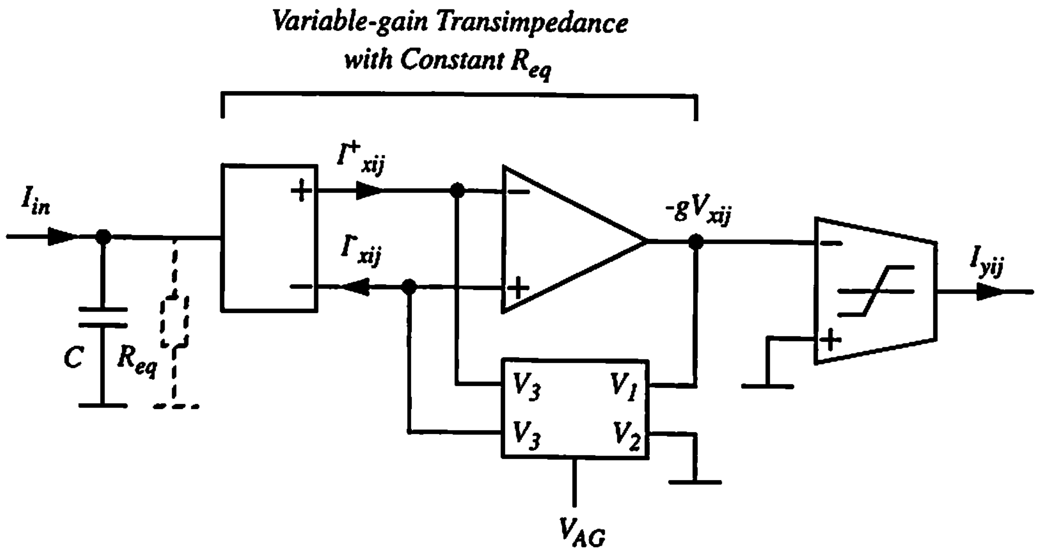


Figure 6-7: SPICE-simulated transfer characteristics of combined neuron and synapses.

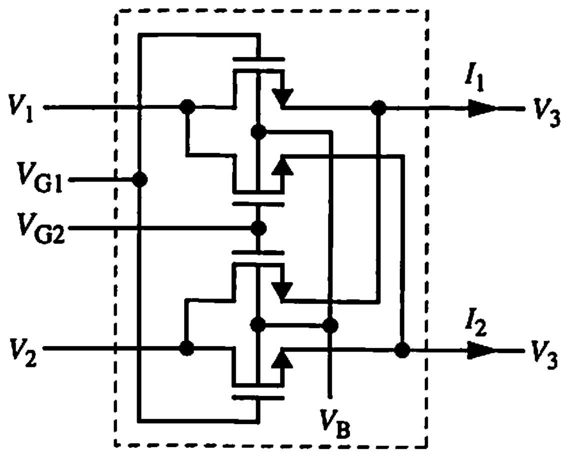


(a) Block diagram.

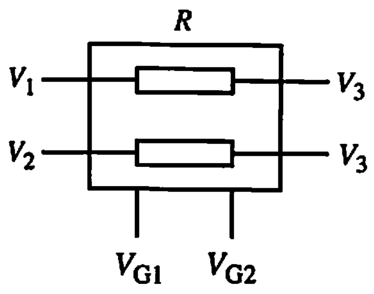


(b) Transimpedance multiplier using single-ended op-amp.

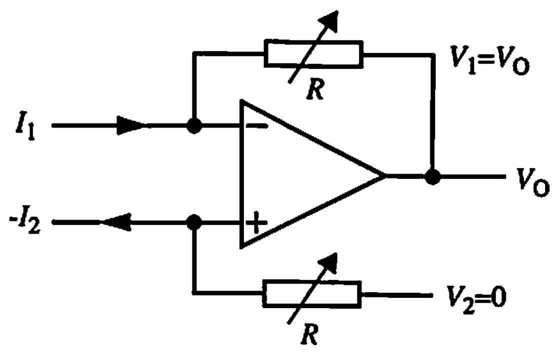
Figure 6-8: Variable-gain neuron cell for hardware annealing.



(a) Double-MOS linear resistor.



(b) Equivalent circuit of (a).



(c) Variavle transimpedance using single-ended op-amp.

Figure 6-9: Transimpedance analog multiplier.

terminals driving the currents I_1 and I_2 at the same virtual voltage V. All transistors are matched through identical geometry and operate in the triode region. When the nonlinearity cancellation condition ($V_{C1} = V_{C4}$ and $V_{C2} = V_{C3}$) is met, the currents I_1 and I_2 which flow out of the output terminals are given by

$$I_1 = K\frac{W}{L}\left[(V_{C1} - V_3 - V_T - \frac{1}{2}V_{13})V_{13} + (V_{C2} - V_3 - V_T - \frac{1}{2}V_{23})V_{23}\right] \tag{6.11}$$

and

$$I_2 = K\frac{W}{L}\left[(V_{C2} - V_3 - V_T - \frac{1}{2}V_{13})V_{13} + (V_{C1} - V_3 - V_T - \frac{1}{2}V_{23})V_{23}\right],$$

where $V_{13} = V_1 - V_3$ and $V_{23} = V_2 - V_3$, and their difference is simply

$$I_{12} = I_1 - I_2 = 2K\frac{W}{L}(V_{C1} - V_{C2})(V_1 - V_2) \tag{6.12}$$

where $K = \mu C_{OX}/2$. To ensure triode region operation, the voltages V_1, V_2 and V_3 must satisfy

$$V_1, V_2, V_3 \le \min\left[V_{C1} - V_T, V_{C2} - V_T\right] \tag{6.13}$$

where V_T is the threshold voltage of MOS device. When the double-MOS differential resistor is configured as the feedback resistor as shown in Figure 6-9 (c), the following relation holds;

$$V_O = -\frac{I_{12}}{2K(W/L)(V_{C1} - V_{C2})}. \tag{6.14}$$

For the negative feedback of operational amplifier, the gain control voltage $V_C = V_{C1} - V_{C2}$ needs to be positive. Despite its good linearity over a wide range of input, the drawback of this circuits is the limited output voltage as is given by $V_O < V_{C2} - V_T$ from (6.13). For example, if $V_{DD} = V_{C1} = +5\text{ V}$, $V_T = 1\text{ V}$, then $V_O <\sim 3.5\text{ V}$ for a useful range of V_C. However, when the gain is low the corresponding output is also small around the bias voltage V_{BIAS}. Thus it does not cause serious problem in most cases, provided that V_{BIAS} is properly chosen.

B. Summing circuit: If the summation of the weighted currents from the neighborhood cells is carried out directly in the transimpedance multiplier, the value of resistance R_x varies as the annealing gain changes because

$$R_x = \frac{1}{2K(W/L)V_C}, \quad V_C > 0. \tag{6.15}$$

The equation (6.15) indicates that the value of the resistance R_x changes with the inverse proportionality as the gain control voltage V_C does. This is not desirable for hardware annealing since the eigenvalues of $\mathbf{M}_g$ in (6.8) remain unchanged. In order to accommodate a constant R_x, the same current inverter shown in Figure 6-4 can be used at the input stage of the multiplier. If a capacitance C is connected between the input and V_{GND}, then the circuit performs a lossy integration

$$C\frac{dV_0}{dt} + K\left(\frac{W}{L}\right)(V_2 - 2V_T) = I_{in}. \tag{6.16}$$

where $V_0 = V_x$ is the voltage at the input node. By using the Laplace transform, the transfer function of the summing circuit corresponding to (6.16) is given by

$$H(s) = \frac{V_0(s)}{I_{in}(s)} = \frac{1}{K(W/L)(V_2 - 2V_T)} \cdot \frac{1}{1 + sC/\left(K(W/L)(V_2 - 2V_T)\right)}. \tag{6.17}$$

C. Nonlinear function: The circuit for the nonlinear function $y = f(x)$ is accomplished by a simple transconductor consisting of a differential amplifier. Its large signal transfer function is a smooth, sigmoid-like characteristics. However, because the output swing of the transimpedance multiplier is limited by (6.13), the value of differential transconductance $g_m = (KI_0(W/L))^{1/2}$ is relatively small in accommodating the normalized linear operating region $-1 \le x \le 1$ for the input range of $|x| \le |x_{max}| \sim 10$. Here, I_0 is the bias current and W/L is the device ratio of the differential-pair transistors. Notice that the situation remains unchanged for increased I_0 because it also increases the saturated output levels. Therefore, as shown in Figure 6-10, a weak positive feedback is applied to increase the transconductance value without increasing the W/L ratio of M_1 and M_2. The transistors $M_4 - M_7$ in saturation determine the feedback factor α, $0 \le \alpha < 1$,

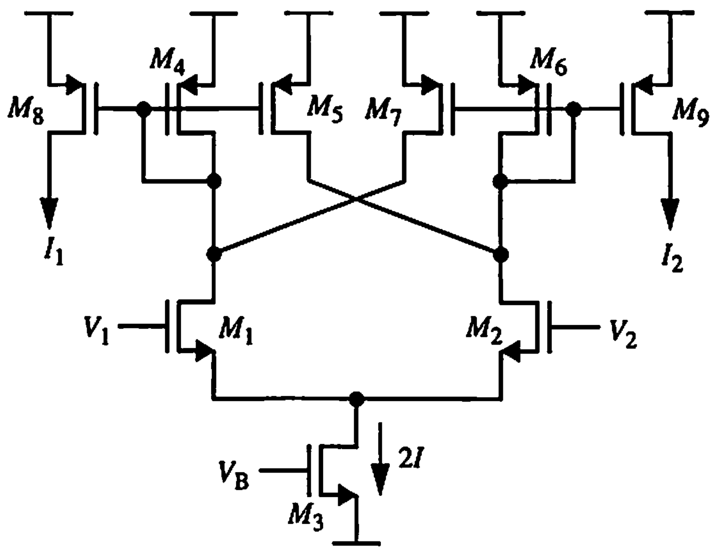
$$\alpha = \frac{(W/L)_{M5}}{(W/L)_{M4}} = \frac{(W/L)_{M7}}{(W/L)_{M6}}. \tag{6.18}$$

Then, the transconductance at $V_d = V_1 - V_2 = 0$ is shown to be

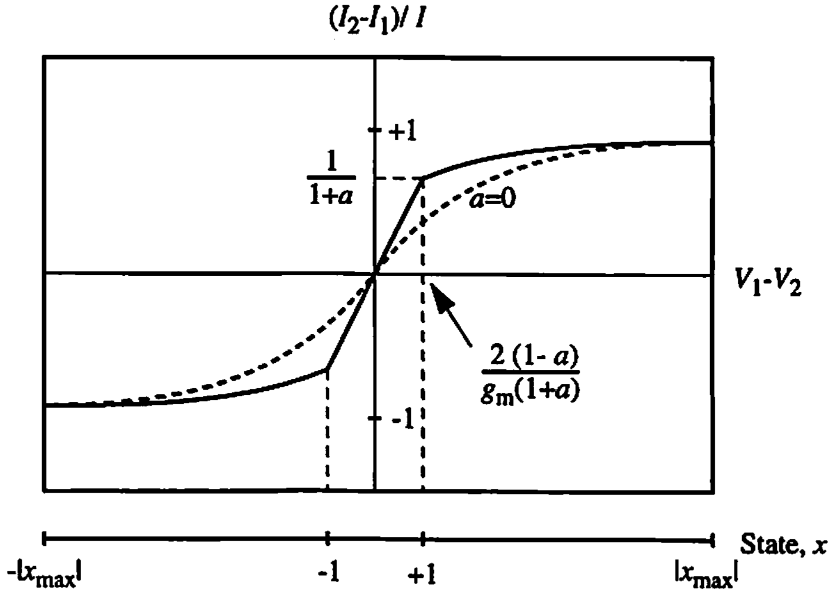
$$g_m = \frac{g_{m(\alpha=0)}}{1-\alpha}. \tag{6.19}$$

The complete schematic diagram of a neuron cell for an annealed CNN is shown in Figure 6-11. If the piecewise-linear function is needed, the circuit shown in Figure 6-12 may be used instead of the circuit described above.

D. Programmable synapses: The circuit shown in Figure 6-13 is a binary-weighted current source array with the capability of four-quadrant multiplication. The magnitude of the multiplication is done by $n-1$ LSB bits and the polarity of the output I_W is controlled by swapping the inputs I_1 and I_2 through the MSB bit. In this design, $n=5$ and the sizes of transistors for weighting are chosen such that $|I_W| \le (2-2^{-3})|I_d|$, where $I_d = I_1 - I_2$, in a step of $0.125 I_d$. Because the current mirrors are used several times, it is important to match them as closely as possible through a careful layout design. The synapse weight for the self-feedback $A(i,j;i,j)$ must be a positive number greater than one. Thus, only four control bits are used for this synapse and, with the addition of constant factor of one, the range of output current is given by $I_d \le I_W \le (5-2^{-2})I_d$ in a step of $0.25 I_d$.



(a) Schematic diagram.



(b) Transfer curves with ($0<a<1$) and without ($a=0$) positive feedback.

Figure 6-10: Differential transconductance with positive feedback.

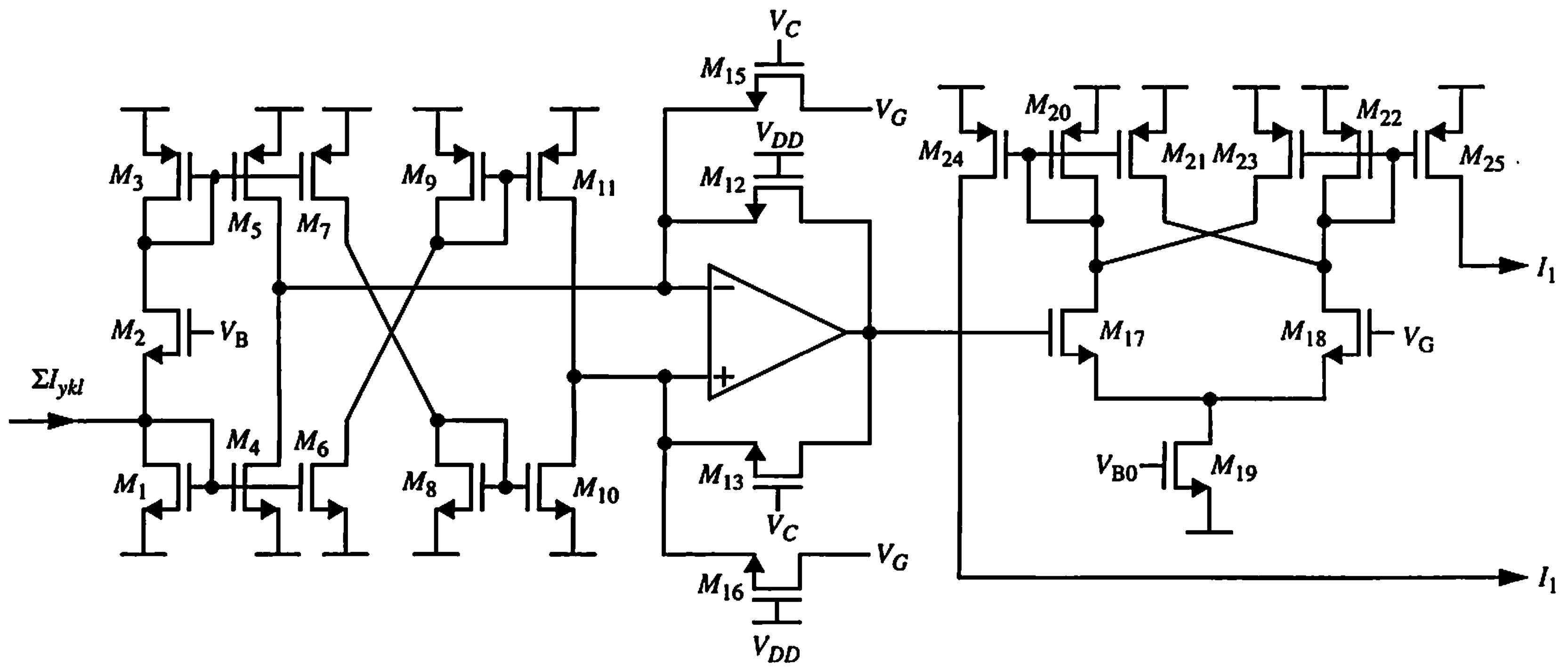


Figure 6-11: Complete schematic diagram of variable-gain neuron cell.

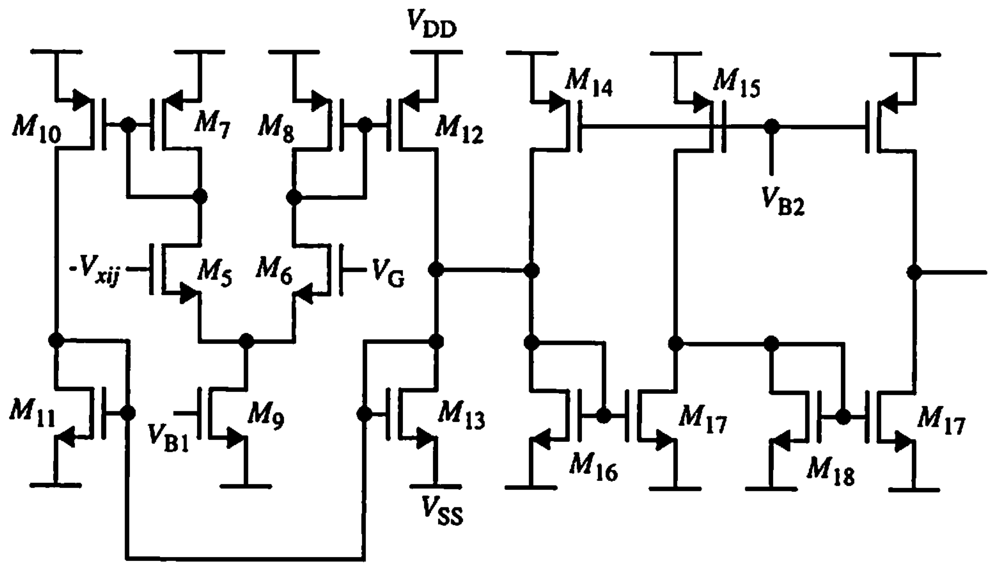


Figure 6-12: Circuit schematic of piecewise-linear function.

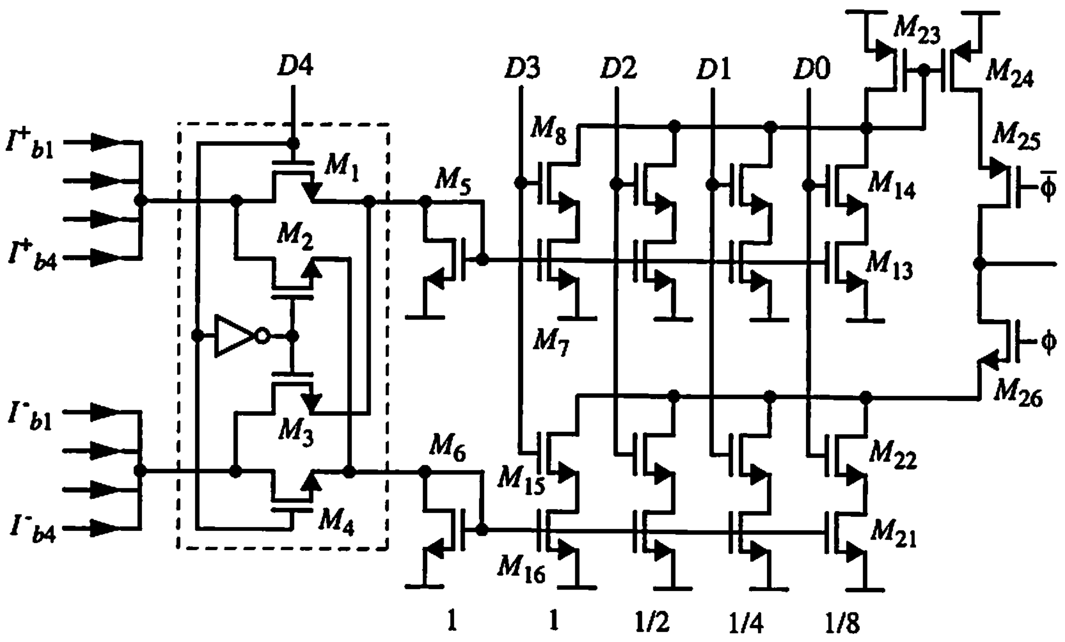


Figure 6-13: Circuit schematic of digitally-programmable synaptic weigth.

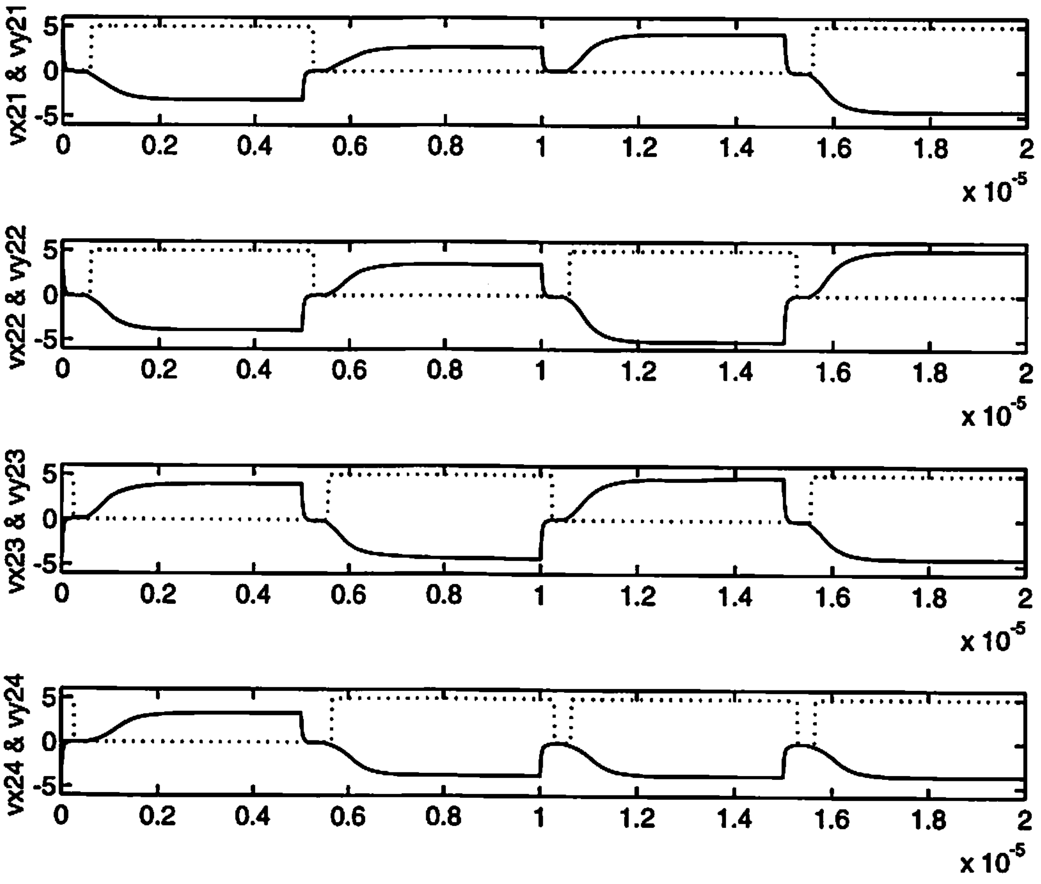
6.4 Simulation results

Figure 6-14 shows the SPICE simulation results of state and logic output voltages in a 4-by-4 CNN during four operation intervals. The cloning templates used are

$$\mathbf{T}_A = \begin{bmatrix} 0 & -0.5 & 0 \\ -0.5 & 2 & -0.5 \\ 0 & -0.5 & 0 \end{bmatrix} \text{ and } \mathbf{T}_B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \qquad (6.20)$$

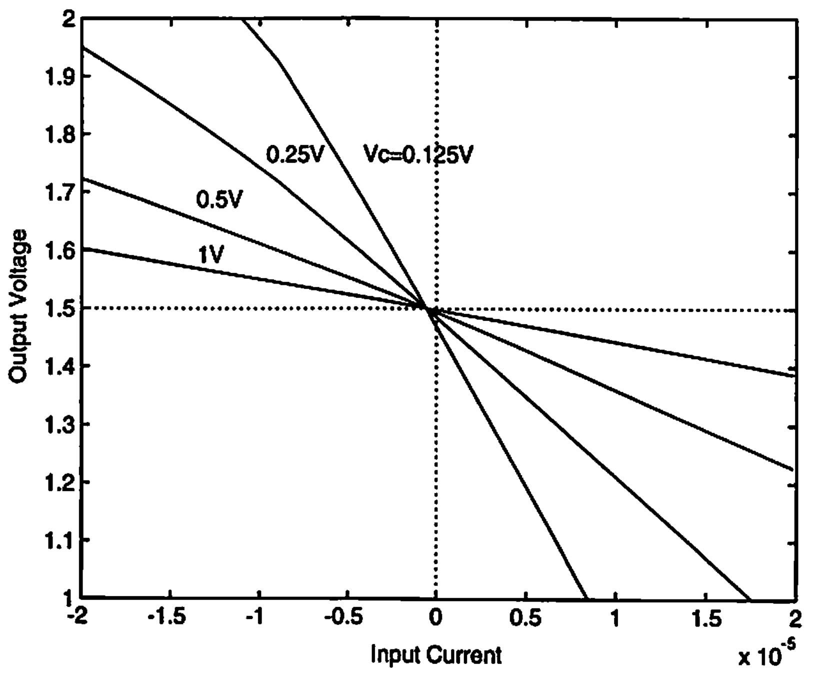
In each operation, the hardware annealing is applied for $0 \leq t \leq 4.5\,uS$ with an initialization period $0.5\,\mu S$ and $g_{\min} = 0.05$. Figure 6-15 shows simulated transfer characteristics of the programmable synaptic circuit shown in Figure 6-9, and combined synaptic weight and piecewise-linear circuit of Figure 6-12. Figures 6-15 (a), (b), and (c) show the simulated transfer characteristics of the transimpedance multiplier, differential-pair transconductor without feedback, and the piecewise-linear circuit, respectively. A composite result is shown in Figure 6-15 (d) for several gain control voltages. If the piecewise-linear circuit is not used, the circuit of Figure 6-10 must be used in order to accommodate relatively small dynamic range of transimpedance multiplier. When $V_{C2} = V_{DD}$, the minimum gain obtainable is approximately 0.02, which is small enough to stimulate the network during the annealing operation. Figure 6-16 (a) shows the transfer characteristics of digitally programmable synapse circuit shown in Figure 6-13 together with the piecewise-linear function. Because the role of transistor M_5 in Figure 6-3 (a) is done by either M_5 or M_6 in Figure 6-13, discontinuous transitions between linear and saturation operating regions are produced through the polarity control circuit ($M_1 - M_4$ in Figure 6-13). However, they can be avoided by using complete piecewise-linear and synapse circuits. In Figure 6-17, experimental circuits of variable-gain neuron and fixed synaptic weights are shown.

A CMOS design of continuous-time CNN with programmable synaptic weights is presented. For the maximum flexibility over a variety of applications, CNN circuits with hardware annealing capability and digitally programmable synaptic weights are designed and verified via circuit simulations. The proper network operation is confirmed for arbitrary and known cloning templates.

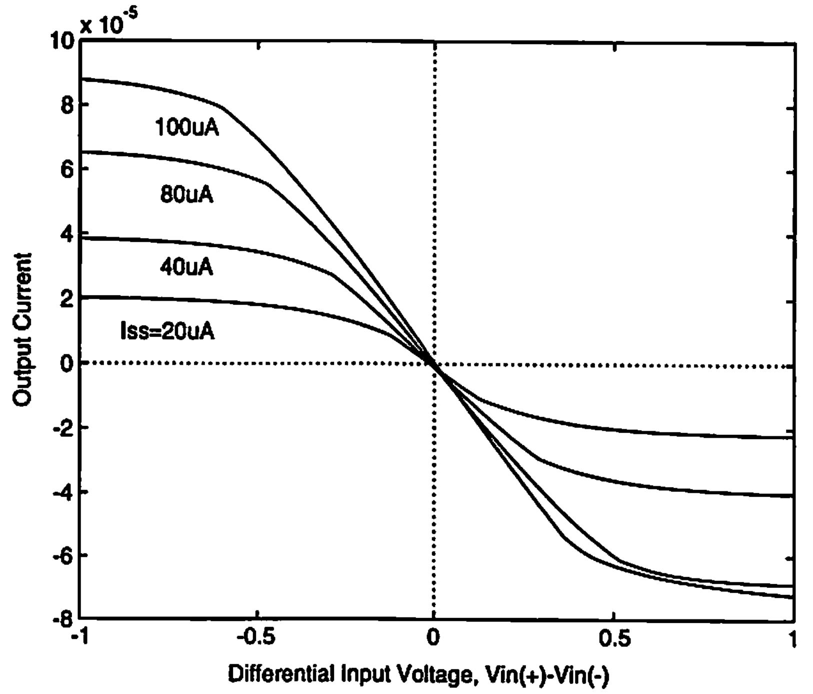


initialization: 0.5 usec., operation: 4.5 usec.

Figure 6-14: Simulated state and logic output voltages of 4 cells in 2nd row. (solid lines: state voltages, dotted lines: logic output voltages)

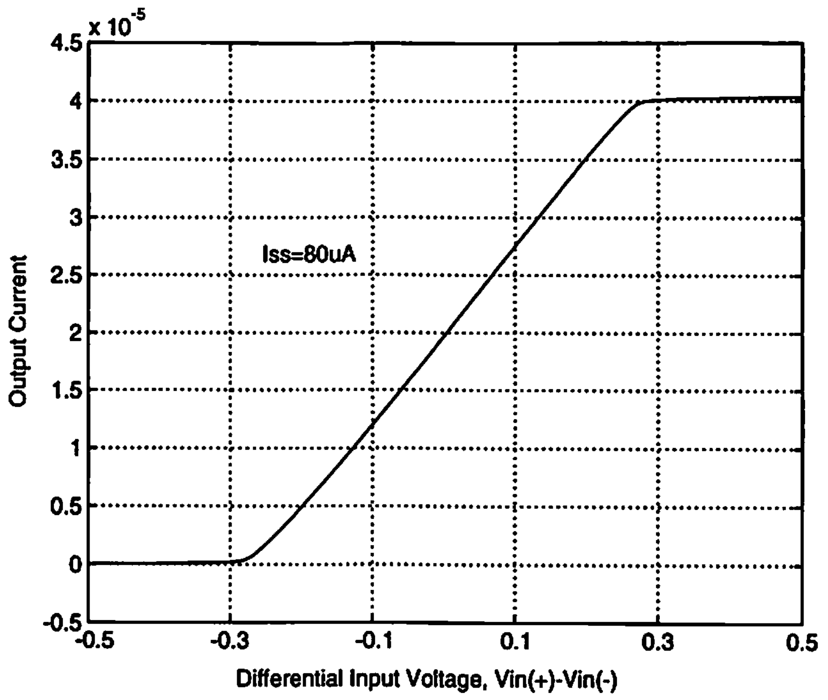


(a) Transfer curves of transimpedance multiplier for several control voltages Vc.

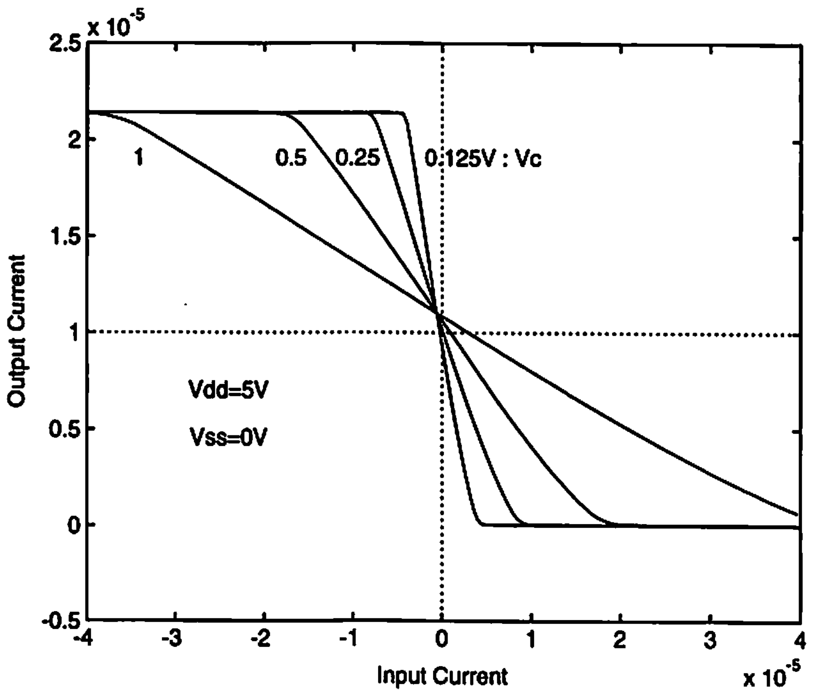


(b) Transfer curves of transconductor for various bias current Iss.

Figure 6-15 (cont.): SPICE simulation results of variable-gain neuron.

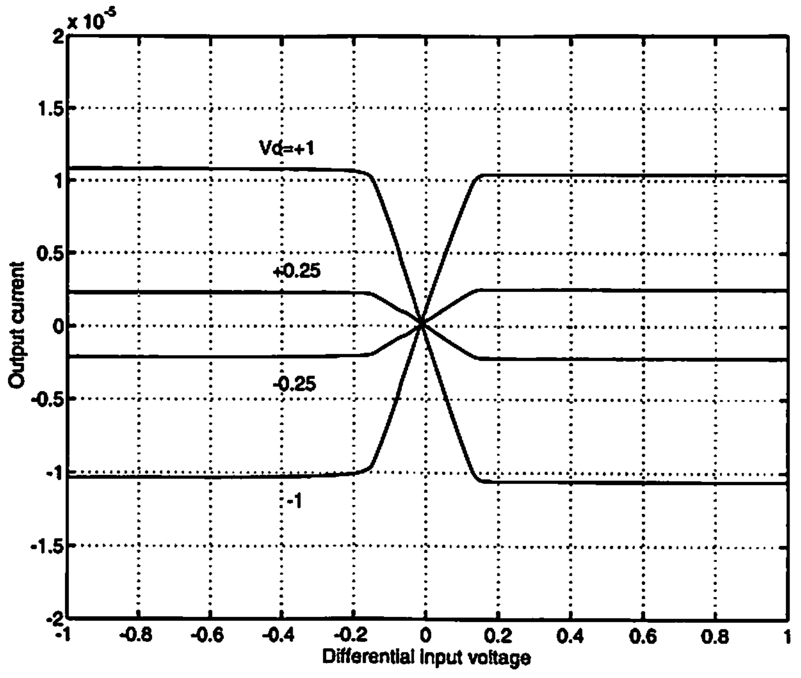


(c) Transfer curve of piecewise-linear neuron (constant gain).

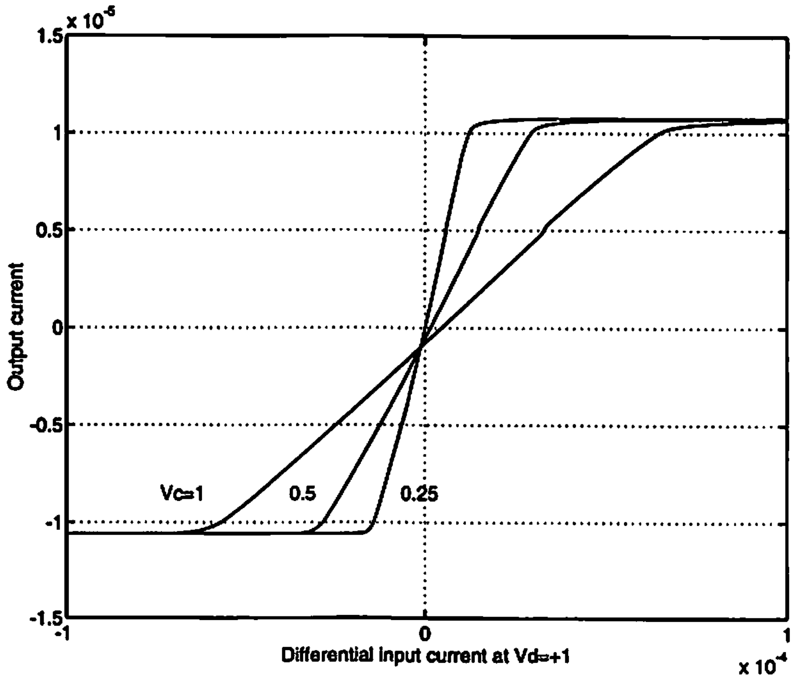


(d) Transfer curves of neuron cell for several annealing gain values.

Figure 6-15: SPICE simulation results of variable-gain neuron.



(a) For several weight values of programmable synapse.



(b) For several annealing gain values.

Figure 6-16: Simulated characteristics of variable-gain neuron with digitally programmable synapse circuit.

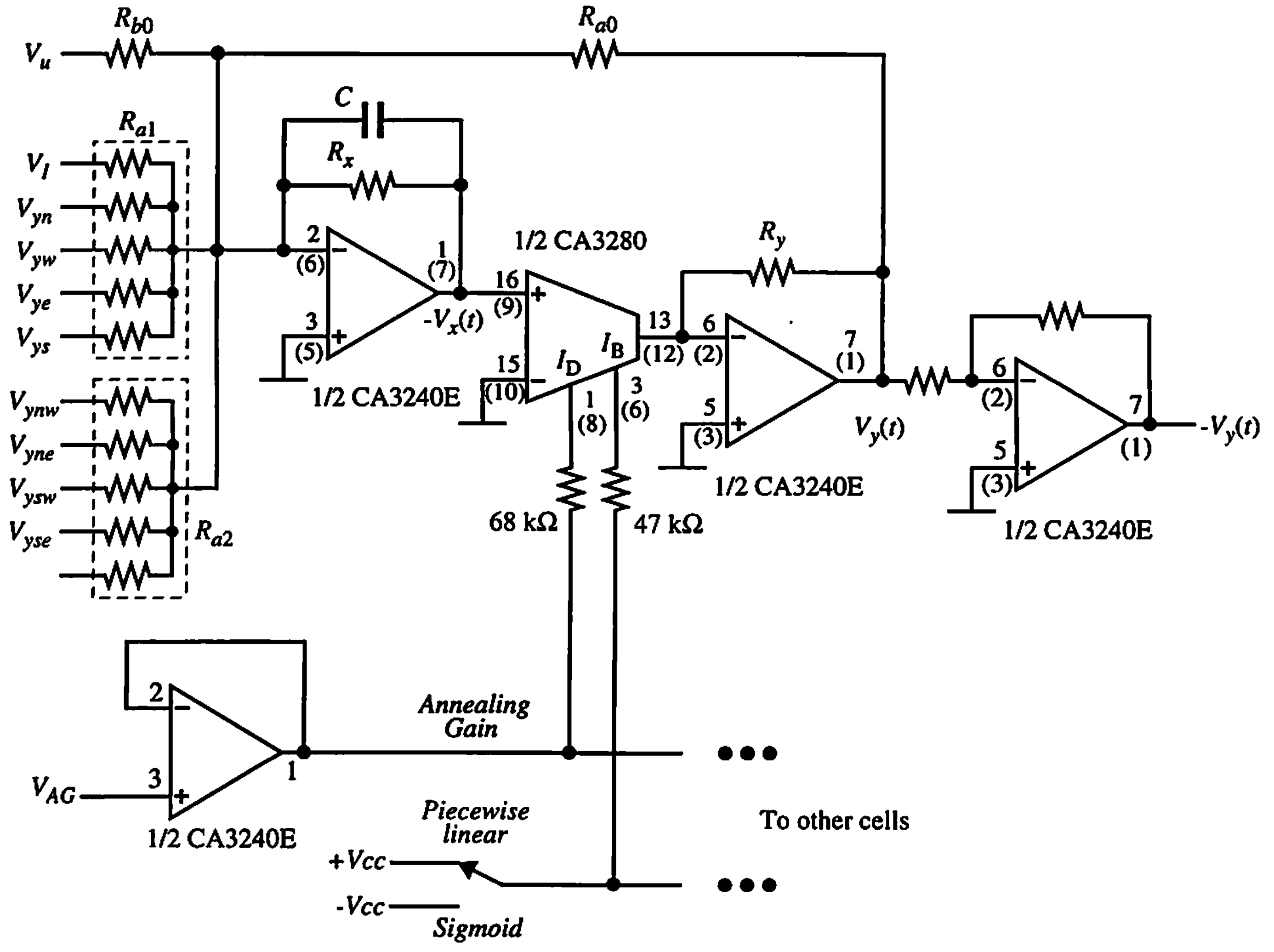


Figure 6-17: Circuit schematic of neuron cell and synapses for annealed CNN.

Conclusions and Further Works

A paralleled annealing method for the recurrent associative neural networks is presented. By using the basic, prototype, general, and application models of neural networks, the method is described and analyzed in terms of the generalized energy which corresponds to the cost function in optimization problems. In contrast to iterative neural network algorithms such as Boltzmann machine and mean-field annealing, it is directly incorporated with a network for solving combinatorial optimization problems and achieves as much parallelism as the network does. In addition, the network stability and parameters in annealed neural networks are also covered as a guideline toward applications.

The generalized energy function of a network which corresponds to the cost function to be minimized or maximized in optimization problems, is first increased by reducing the voltage gain of neurons. Then, the hardware annealing searches for the globally minimum energy state by continuously increasing the gain of neurons. In this sense, the hardware annealing shall be a paralleled, hardware-based realization of the mean-field annealing. However, the process of global optimization can be described by the eigenvalue problem of a time-varying dynamic system, instead of stochastic reasoning. As a consequence, the hardware annealing;

- achieves fast operation via true parallelism and
- is suitable for hardware realization.

The speed of convergence can be faster than those of the stochastic methods in several orders of magnitude, and is comparable to those of typical analog VLSI neuroprocessors.

In a field of applications where real-time operation for the optimal solutions is required, conventional iterative algorithms are limited in accommodating with high-speed, low-power electronic system. With the annealing capability, a VLSI neuroprocessor can easily incorporate with a variety of combinatorial optimization problems in high speed. The applications of hardware annealing may include, but are not limited to

- general-purpose hardware accelerator,
- special-purpose engineering optimization tool, e.g., VLSI design and network router,
- signal detection and error correction in communications, and
- image processing and feature extraction, etc.

A fundamental architecture of hardware annealing method is covered in this thesis. To establish the hardware annealing method as a general approach to neural network optimizations, further researches on the following areas may be required;

1. qualitative analysis of annealed neural network,
2. applications,
3. efficient hardware architecture for large-scale problems,
4. VLSI implementation,

in general aspect, and

1. choice of annealing schedule for better performance,
2. relaxation time for guaranteed optimal or near-optimal solutions,
3. efficiency of hardware annealing in practical applications of CNN,
4. extensive performance evaluation of NN-MLSE,

in relation to the topics covered in this thesis.

References

[1] P.M. Pardalos, J.B. Rosen, *Constrained Global Optimization: Algorithms and Applications*, Berlin; Germany, Springer-Verlag, 1987.

[2] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 81, pp. 3088-3092, 1984.

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

[4] 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 Systs.*, vol. 33, no. 5, pp. 533-541, May 1986.

[5] R.E. Kalman and J.E. Bertram, "Control system analysis and design via the 'second method' of Lyapunov Part I: Continuous-time systems," *Trans. ASME*, pp. 371-393, June 1960.

[6] R.E. Kalman and J.E. Bertram, "Control system analysis and design via the 'second method' of Lyapunov Part II: Discrete-time systems," *Trans. ASME*, pp. 394-400, June 1960.

[7] G.A. Tagliarini, J.F. Christ, E.W. Page, "Optimization using neural networks," *IEEE Trans. Comput.*, vol. 40, pp. 1347-1358, Dec. 1991.

[8] C.-Y. Maa, M.A. Shanblatt, "A two-phase optimization neural network," *IEEE Trans. Neural Networks*, vol. 3, pp. 1003-1009, Nov. 1992.

[9] B. Bhaumik, "Optimization with neural networks: a recipe for improving convergence and solution quality," *Biol. Cybern.*, vol. 67, pp. 445-449, 1992.

[10] S. Abe, J. Kawakami, K. Hirasawa, "Solving inequality constrained combinatorial optimization problems by the Hopfield neural networks," *Neural Networks*, vol. 5, pp. 663-670, 1992.

[11] C. Peterson, "Parallel distributed approaches to combinatorial optimization: Benchmark studies on traveling salesman problem," *Neural Computation*, vol. 2, pp. 261-269, 1990.

[12] G.V. Wilson, G.S. Pawley, "On the stability of the travelling salesman problem algorithm of Hopfield and Tank," *Biol. Cybern.*, vol. 58, pp. 63-70, 1988.

[13] E. Wacholder, J. Han, R.C. Mann, "A neural network algorithm for the multiple traveling salesmen problem," *Biol. Cybern.*, vol. 61, pp. 11-19, 1989.

[14] R. Cuykendall, R. Reese, "Scaling the neural TSP algorithm," *Biol. Cybern.*, vol. 60, pp. 365-371, 1989.

[15] S.V.B. Aiyer, M. Niranjan, F. Fallside, "A theoretical investigation into the performance of the Hopfield model," *IEEE Trans. Neural Networks*, vol. 1, pp. 204-215, June 1990.

[16] J.-S. Yih, P. Mazumder, "A neural network design for circuit partitioning," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 1265-11271, Dec. 1990.

[17] D.E. Van den Bout, T.K. Miller, "Graph partitioning using annealed neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 191-203, June 1990.

[18] P.-H. Shih, W.-S. Feng, "An application of neural networks on channel routing problem," *Parallel Computing*, vol. 17, pp. 229-240, 1991.

[19] A. Ushida, L.O. Chua, "A global optimization algorithm based on circuit partitioning technique," *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 475-478, 1992.

[20] S.T. Chakradhar, et al, "Toward massively parallel automatic test generation," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 981-994, Sept. 1990.

[21] D.M. Wells, "Solving degenerate optimization problems using networks neural oscillators," *Neural Networks*, vol. 5, pp. 949-959, 1992.

[22] M. Forti, S. Manetti, M. Marini, "Neural networks for optimization of non-quadratic cost functions with application to adaptive signal processing," *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 2909-2912, 1992.

[23] J.H.M. Korst, E.H.L. Aarts, "Combinatorial optimization on a Boltzmann machine," *J. Parallel and Distributed Comput.*, vol. 6, pp. 331-357, 1989.

[24] P. Peretto, *An Introduction to the Modeling of Neural Networks*, Cambridge; Great Britain, Cambridge University Press, 1992.

[25] A. Cichocki, R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, West Sussex; England, John Wiley & Sons, 1993.

[26] R. Hechht-Nielsen, *Neurocomputing*, Reading, MA: Addison-Wesley Publishing Co., Inc., 1990

[27] M.P. Kennedy, L.O. Chua, "Unifying the Tank and Hopfield linear programming circuit and the canonical nonlinear programming circuit of Chua and Lin," *IEEE Trans. Circuits Syst.*, vol. 34, pp. 210-214, Feb. 1987.

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

[29] C.-Y. Maa, M.A. Shanblatt, "Linear and nonlinear programming neural network analysis," *IEEE Trans. Neural Networks*, vol. 3, pp. 580-594, July 1992.

[30] C. Chiu, C.-Y. Maa, M.A. Shanblatt, "Energy function analysis of dynamic programming neural networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 418-426, July 1991.

[31] A. Rodriguez-Vazequez, et al., "Nonlinear switched-capacitor "neural" networks for optimization problem," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 384-398, Mar. 1990.

[32] D. Kunz, "Suboptimum solutions obtained by the Hopfield-Tank neural network algorithm," *Biol. Cybern.*, vol. 65, pp. 129-133, 1991.

[33] B.W. Lee, B.J. Sheu, "An investigation of local minima of Hopfield network for optimization circuits," in *IEEE Int. Conf. Neural Networks*, San Diego, CA, vol. I, pp. 45-51, July 1988.

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

[35] B.W. Lee, B.J. Sheu, "Paralleled hardware annealing for optimal solutions on electronic neural networks," *IEEE Trans. Neural Networks*, vol. 4, pp. 588-598, July 1993.

[36] N. Metropolis, et al., "Equations of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, pp. 1087-1091, 1953.

[37] S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680, May 1983.

[38] W. Jeffrey, R. Rosner, "Optimization algorithms: simulated annealing and neural network processing," *Astrophysical J.*, vol. 310, pp. 473-481, Nov. 1986.

[39] S. Geman, D. Geman, "Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, pp. 721-741, Nov. 1984.

[40] E.H.L. Aarts, J.H.M. Korst, *Simulated Annealing and Boltzmann Machine*, Wiley, Chichster, 1988.

[41] S.S. Wilson, "Teaching network connectivity using simulated annealing on a massively parallel processor," *Proc. IEEE*, vol. 79, pp. 559-566, Apr. 1991.

[42] C.R. Nassar, M.R. Soleymani, "Codebook design for trellis quantization using simulated annealing," *IEEE Trans. Speech Audio Proc.*, vol. 1, pp. 400-404, Oct. 1993.

[43] C. Peterson, J.R. Anderson, "A mean field theory learning algorithm for neural networks," *Complex Systems*, vol. 1, no. 5, pp. 995-1019, 1987.

[44] C. Peterson, E. Hartman, "Explorations of the mean field theory learning algorithm," *Neural Networks*, vol. 2, pp. 475-494, 1989.

[45] C. Peterson, "Mean field theory neural networks for feature recognition, content addressable memory and optimization," *Connection Science*, vol. 3, pp. 3-33, 1991.

[46] G. Bilbro, et al., "Optimization by mean field annealing," in D.S. Touretzky [Ed.], *Advances in Neural Information Processing Systems, Vol. I*, Morgan Kaufmann, New York, 1990.

[47] J. Van der Spiegel, et al, "An analog neural computer with modular architecture for real-time dynamic computations," *IEEE J. Solid-State Circuits*, vol. 27, pp. 82-92, Jan. 1992.

[48] B.E. Boser, et al, "An analog neural network processor with programmable topology," *IEEE J. Solid-State Circuits*, vol. 26, pp. 2017-2025, Dec. 1991.

[49] E. Sackinger, et al, "Application of the ANNA neural network chip to high-speed character recognition," *IEEE Trans. Neural Networks*, vol. 3, pp. 498-505, May 1992.

[50] J. Choi, "Analog-digital VLSI neuroprocessors for signal processing and communication," USC-SIPI Report #246, Dept. of Electrical Engineering, University of Southern California, Dec. 1993.

[51] W.-C. Fang, B.J. Sheu, O.T.-C. Chen, and J. Choi, "A VLSI neural processor for image data compression using self-organizing networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 506-518, May 1992.

[52] R. Bijjani, P. Das, "An M-ary neural network model," *Neural Computation*, vol. 2, pp. 536-551, 1990.

[53] K. Sivakumar, U.B. Desaai, "Image restoration using a multilayer perceptron with a multilevel sigmoid function," *Proc. IEEE Int. Symp. Circuits Systs.*, pp. 2917-2920, 1992.

[54] J. Si, A.N. Michel, "Analysis and synthesis of discrete-time neural networks with multilevel threshold functions," *Proc. IEEE Int. Symp. Circuits Systs.*, pp. 1461-1464, 1991.

[55] J.-D. Yuh, R.W. Newcomb, "Circuits for multilevel nonlinearities," *Proc. Int. Conf. Neural Networks*, vol. II, pp. 27-32, 1992.

[56] J.-D. Yuh, R.W. Newcomb, "A multilevel neural network for A/D conversion," *IEEE Trans. Neural Networks*, vol. 4, pp. 470483, May 1993.

[57] S.H. Bang, B.J. Sheu, J.C.-F. Chang, "Search of optimal solutions in multi-level neural networks," *Proc. IEEE Int. Symp. Circuits Systs.*, vol. 6, pp. 423-426, 1994.

[58] S.H. Bang, B.J. Sheu, J.C.-F. Chang, "Multi-level neural networks with optimal solutions," scheduled for *IEEE Trans. Circuits Syst.*, Dec. 1994.

[59] L.O. Chua, L. Yang, "Cellular neural network: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257-1272, Oct. 1988.

[60] L.O. Chua, L. Yang, "Cellular neural network: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1273-1290, Oct. 1988.

[61] L.O. Chua, T. Roska, "The CNN paradigm," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 147-156, Mar. 1993.

[62] T. Roska, L.O. Chua, "The CNN universal machine: An analogic array computer," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 163-173, Mar. 1993.

[63] T. Roska, J. Vandewalle, Eds., *Cellular Neural Networks*, West Sussex; England, John Wiley & Sons, 1993.

[64] H. Harrer, J. A. Nossek, "Discrete-time cellular neural networks," T. Roska, J. Vandewalle, Eds., *Cellular Neural Networks*, West Sussex; England, John Wiley & Sons, 1993.

[65] T. Roska, L.O. Chua, "Cellular neural networks with non-linear and delay-type template elements and non-uniform grids," *Int J. Circuit Theory and Applications*, vol. 20, pp. 469-481, 1992, also in [6].

[66] H. Harrer, "Multiple layer discrete-time cellular neural networks using time-variant templates," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 191-199, Mar. 1993.

[67] A. Bouzerdoum, R.B. Pinter, "Shunting inhibitory cellular neural networks: Derivation and stability analysis," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 215-221, Mar. 1993.

[68] L.O. Chua, T. Roska, "Stability of a class of nonrecprocal cellular neural networks," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1520-1527, Dec. 1990.

[69] T. Roska, C.W. Wu, M. Balsi, L.O. Chua, "Stability and dynamics of delay-type general and cellular neural networks," *IEEE Trans. Circuits Syst. Part I*, vol. 39, pp. 487-490, June 1992.

[70] P.P. Civalleri, M. Gilli, L. Pandolfi, "On stability of cellular neural networks with delay," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 157-165, Mar. 1993.

[71] M.P. Joy, V. Tavsanoglu, "A new parameter range for the stability of opposite-sign cellular neural networks," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 204-207, Mar. 1993.

[72] P. Thiran, "Influence of boundary conditions on the behavior of cellular neural networks," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 207-212, Mar. 1993.

[73] F.A. Savaci, J. Vandewalle, "On the stability analysis of cellular neural networks," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 213-215, Mar. 1993.

[74] T. Roska, C. Wah, L.O. Chua, "Stability of cellular neural networks with dominant nonlinear and delay-type templates," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 270-272, April 1993.

[75] T. Matsumoto, L.O. Chua, H. Suziki, "CNN cloning template: Connected component detector," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 633-635, May 1990.

[76] T. Matsumoto, L.O. Chua, R. Furukawa, "CNN cloning template: Hole-filler," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 635-638, May 1990.

[77] T. Matsumoto, L.O. Chua, T. Yokohama, "Image thinning with a cellular neural network," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 638-640, May 1990.

[78] T. Matsumoto, L.O. Chua, H. Suziki, "CNN cloning template: Shadow detector," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1070-1073, Aug. 1990.

[79] B.E. Shi L.O. Chua, "Resistive grid image filtering: Input/output analysis via the CNN framework," *IEEE Trans. Circuits Syst. Part I*, vol. 39, pp. 531-548, July 1992.

[80] K.R. Crounse, T. Roska, L.O. Chua, "Image halftoning with cellular neural networks," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 267-283, June 1993.

[81] B.E. Shi, T. Roska, L.O. Chua, "Design of linear cellular neural networks for motion sensitive filtering," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 267-283, May 1993.

[82] V. Cimagalli, M. Bobbi, M. Balsi, "MODA: Moving object detecting architecture," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 174-183, Mar. 1993.

[83] T. Sziranyi, J. Csicsvari, "High-speed character recognition using a dual cellular neural network architecture (CNND)," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 223-231, Mar. 1993.

[84] V. Perez-Munuzuri, V. Perez-Villar, L.O. Chua, "Autowaves for image processing on a two-dimensional CNN array of excitable nonlinear circuits: Flat and wrinkled labyrinths," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 174-181, Mar. 1993.

[85] T. Roska, et al., "The use of CNN models in the subcortical visual pathway," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 182-195, Mar. 1993.

[86] S. Paul, K. Huper, J. Nossek, L.O. Chua, "Mapping nonlinear lattice equations onto cellular neural networks," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 196-203, Mar. 1993.

[87] Z. Galias, "Designing cellular neural networks for the evaluation of local Boolean function," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 219-223, Mar. 1993.

[88] P. Szolgay, G. Voros, Gy. Eross, "On the applications of the cellular neural network paradigm in mechanical vibrating systems," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 222-227, Mar. 1993.

[89] L. Yang, L.O. Chua, K.R. Krieg, "VLSI implementation of cellular neural networks," *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 2425-2427, 1990.

[90] H. Halonen, V. Porra, T. Roska, L.O. Chua, "Programmable analog VLSI CNN chip with local digital logic," *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1291-1294, 1991.

[91] J.M. Cruz, L.O. Chua, "A CNN chip for connected component detection," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 812-817, July 1991.

[92] S. Espejo, et al., "Switched-current techniques for image processing cellular neural networks in MOS VLSI," *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1537-1540, 1992.

[93] A. Rodriguez-Vazquez, et al., "Current-mode techniques for the implementation of continuous- and discrete-time cellular neural networks," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 132-146, Mar. 1993.

[94] J.E. Varrientos, E. Sanchez-Sinencio, J. Ramirez-Angulo, "A current-mode cellular neural network implementation," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 147-155, Mar. 1993.

[95] H. Harrer, J.A. Nossek, R. Stelzl, "An analog implementation of discrete-time cellular neural networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 466-476, May 1992.

[96] I.A. Baktir, M.A. Tan, "Analog CMOS implementation of cellular neural networks," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 200-206, Mar. 1993.

[97] G.F.D. Betta, S. Graffi, Zs.M. Kovacs, G. Masetti, "CMOS implementation of an analogically programmable cellular neural network," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 206-215, Mar. 1993.

[98] M. Anguita, F.J. Pelayo, A. Prieto, J. Ortega, "Analog CMOS implementation of a discrete time CNN with programmable cloning templates," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 215-218, Mar. 1993.

[99] G.C. Cardarilli, F. Sargeni, "Very efficient VLSI implementation of CNN with discrete templates," *Electronics Letters*, vol. 29, pp. 1286-1287, July 1993.

[100] N. Fruehauf, E. Lueder, G. Bader, "Fourier optical realization of cellular neural networks," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 156-162, Mar. 1993.

[101] T. Kozek, T. Roska, L.O. Chua, "Genetic algorithm for CNN template learning," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 392-402, June 1993.

[102] L.O. Chua, P. Thiran, "An analytic method for designing simple cellular neural networks," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 1332-1341, Nov. 1991.

[103] G. Seiler, A.J. Schuler, J.A. Nossek, "Design of robust cellular neural networks," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 358-364, May 1993.

[104] G. Seiler, J.A. Nossek, "Winner-take-all cellular neural networks," *IEEE Trans. Circuits Syst. Part II*, vol. 40, pp. 184-190, Mar. 1993.

[105] F. Zou, J.A. Nossek, "A chaotic attractor with cellular neural networks," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 811-812, July 1991.

[106] F. Zou, J.A. Nossek, "Bifurcation and chaos in cellular neural networks," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 166-173, Mar. 1993.

[107] F. Zou, A. Katerle, J.A. Nossek, "Homoclinic and hetroclinic orbits of the three-cell cellular neural networks," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 843-848, Nov. 1993.

[108] M. Gilli, "Strange attractors in delayed cellular neural networks," *IEEE Trans. Circuits Syst. Part I*, vol. 40, pp. 849-853, Nov. 1993.

[109] B.W. Lee, B.J. Sheu, *Hardware Annealing in Analog VLSI Neurocomputing*, Norwell, MA: Kluwer Academic Publishers, 1991.

[110] D. Quinney, *An introduction to the Numerical Solution of Differential Equations*, Research Studies Press, England, 1985.

[111] C.-T. Chen, *Linear System Theory and Design*, Orlando, FL; Holt, Rinehart and Winston, 1984.

[112] S. Bang, B.J. Sheu, "A quick search of optimal solutions for cellular neural networks," submitted for journal publication.

[113] S. Bang, B.J. Sheu, "Optimal solutions for cellular neural networks by paralleled hardware annealing," admitted for *IEEE Trans. Neural Networks.*

[114] G. Ungerboeck, "Adaptive maximum-likelihood receiver for carrier-modulated data transmission systems," *IEEE Trans. Communications*, vol. 22, pp. 624-636, May 1974.

[115] P.G. Gulak and E. Shewdyk, "VLSI structures for Viterbi receivers: Part I - General theory and applications," *IEEE Trans. Selected Areas in Communications*, vol. 4, pp. 142-154, Jan. 1986.

[116] P.G. Gulak and E. Shewdyk, "VLSI structures for Viterbi receivers: Part II - Encoded MSK modulation," *IEEE Trans. Selected Areas in Communications*, vol. 4, pp. 155-159, Jan. 1986.

[117] P.G. Gulak, T. Kailath, "Locally connected VLSI architectures for the Viterbi algorithm," *IEEE Trans. Selected Areas in Communications*, vol. 6, pp. 527-537, Apr. 1988.

[118] N.J.P. Frenette, et al, "Implementation of a Viterbi processor for a digital communications systems with a time-dispersive channel," *IEEE Trans. Selected Areas in Communications*, vol. 4, no. 1, pp. 160-167, Jan. 1986.

[119] J. G. Proakis, *Digital Communications*, McGraw Hill, New York, 1983.

[120] G. Zeng, D. Hush, and N. Ahmed, "An application of neural net in decoding error-correcting codes," *Proc. IEEE Int. Symp. Circuits Systs.*, pp. 782-785, 1989.

[121] J. Bruck and M. Blaum, "Neural networks, error-correcting codes, and polynomials over the binary n-cube," *IEEE Trans. Information Theory*, vol. 35, no. 5, pp. 976-987, Sept. 1989.

[122] A. Hiramatsu, "Integration of ATM call admission control and link capacity control by distributed neural networks," *IEEE J. Selected Areas in Communications*, vol. 9, pp. 1131-1138, Sept. 1991

[123] J. Provence, "Neural network implementation for an adaptive maximum-likelihood receiver," *Proc. IEEE Int. Symp. Circuits Systs.*, pp. 2381-2385, 1988.

[124] S. Haykin, *Adaptive Filter Theory*, 2nd Ed., Prentice Hall, NJ, 1991.

[125] J.N. Babanezhad and G.C. Temes, "A 20-V four-quadrant CMOS analog multiplier," *IEEE J. Solid-State Circuits*, vol. 20, pp. 1158-1168, Dec. 1985.

[126] K. Bult and H. Wallinga, "A CMOS four-quadrant analog multiplier," *IEEE J. Solid-State Circuits*, vol. 21, pp. 430-435, June 1986.

[127] M. Ismail, S.V. Smith, R.G. Beale, "A new MOSFET-C universal filter structure for VLSI," *IEEE J. Solid-State Circuits*, vol. 23, pp. 183-194, Feb. 1988.

[128] Z. Czarnul, "Novel MOS resistive circuit for synthesis of fully integrated continuous-time filters," *IEEE Trans. Circuits Syst.*, vol. 33, pp. 718-721, July 1986.

[129] K. Bult, H. Wallinga, "A class of analog CMOS circuits based on the square-law characteristic of an MOS transistor in saturation," *IEEE J. Solid-State Circuits*, vol. 22, pp. 357-365, June 1987.

[130] S.H. Bang, B.J. Sheu, "A neural-based digital communication receiver for intersymbol interference and white Gaussian noise channels," *Proc. IEEE Int. Symp. Circuits Systs.*, pp.2933-2936, May 1992.

[131] S.H. Bang and B.J. Sheu, "Neural network communication receiver based on the nonlinear filtering," *Proc. IEEE Int. Jnt. Conf. Neural Networks*, June 1992.

[132] J. Choi, S.H. Bang, B.J. Sheu, "A programmable analog VLSI neural network processor for communication receivers," *IEEE Trans. Neural Networks*, vol. 4, no. 3, pp. 484-495, May 1993.

[133] S. Bang, B.J. Sheu, J. Choi, "Programmable VLSI neural network processors for equalization of digital communication channels," *Proc. Int. Workshop on Applications of Neural Networks to Telecommunications*, Princeton, NJ, 1993.

[134] J. Choi, S.H. Bang, B.J. Sheu, "A programmable VLSI neural network processor for digital communication," *Proc. IEEE Custom Integrated Circuits Conf.*, May 1993.

[135] S. H. Bang, J. Choi, and B. J. Sheu, "A 3V data transceiver chip for dual-mode cellular communication systems," *IEEE Symp. on VLSI Circuits*, pp. 71-72, Koyto, Japan, May 1993.

APPENDIX

Journal Papers:

[1] J. Choi, S.H. Bang, and B.J. Sheu, "A programmable analog VLSI neural network processor for communication receivers," *IEEE Trans. Neural Networks*, vol. 4, May 1993.

[2] S.H. Bang, B.J. Sheu, J.C.-F. Chang, "Multi-level neural networks with optimal solutions," scheduled for *IEEE Trans. Circuits Systs.*, Dec. 1994.

[3] S.H. Bang, B.J. Sheu, "Optimal solutions for cellular neural networks by paralleled hardware annealing," admitted for *IEEE Trans. Neural Networks.*

[4] S.H. Bang, B.J. Sheu, "Cellular neural networks for MLSE detection of signals in digital communications," submitted for journal publication.

Conference Papers:

[1] S.H. Bang and B.J. Sheu, "A neural-based digital communication receiver for inter-symbol interference and white Gaussian noise channels," *Proc. IEEE Int. Symp. Circuits Systs.*, May 1992.

[2] S.H. Bang and B.J. Sheu, "Neural network communication receiver based on the nonlinear filtering," *Proc. IEEE Int. Jnt. Conf. Neural Networks*, June 1992.

[3] S.H. Bang and B.J. Sheu, "A multi-chip module for hand-held digital cellular mobile telephone," *Proc. IEEE Multi-chip Module Conf.*, 1992.

[4] S.H. Bang, J. Choi, and B.J. Sheu, "A 3V data transceiver chip for dual-mode cellular communication systems," *1993 Symp. on VLSI Circuits*, Kyoto, Japan, May 1993.

[5] S.H. Bang, B.J. Sheu, J. Choi, "Programmable VLSI neural network processors for equalization of digital communication channels," *Proc. Int. Workshop on Applications of Neural Networks to Telecommunications*, Princeton, NJ, 1993.

[6] J. Choi, S.H. Bang, B.J. Sheu, "A programmable VLSI neural network processor for digital communication," *Proc. IEEE Custom Integrated Circuits Conf.*, May 1993.

[7] S.H. Bang, B.J. Sheu, J.C.-F. Chang, "Search of optimal solutions in multi-level neural networks," *Proc. IEEE Int. Symp. Circuits Systs.*, vol. 6, pp. 423-426, May/June 1994.

[8] S.H. Bang, B.J.Sheu, J.C.-F. Chang, "Paralleled hardware annealing for optimal solutions in multi-level recursive neural networks," *Proc. World Cong. Neural Networks*, pp. IV104-109, June 1994.

[9] S.H. Bang, B.J. Sheu, "A quick search of optimal solutions for cellular neural networks," *Proc. World Cong. Neural Networks*, pp. II549-554, June 1994.

[10] S.H. Bang, B.J. Sheu, R.C.-H. Chang, "Maximum-likelihood sequence estimation of communication signals by a Hopfield neural network," *Proc. IEEE Int. Conf. Neural Networks*, pp. 3369-3374, June 1994.

[11] S.H. Bang, B.J. Sheu, T.H.-Y. Wu, "Paralleled hardware annealing of cellular neural networks for optimal solutions," *Proc. IEEE Int. Conf. Neural Networks*, pp. 2046-2051, June 1994.

[12] S.H. Bang, B.J. Sheu, "Dense CMOS design of cellular neural networks with programmable synaptic weights (Invited)," *Proc. IEEE Int. Conf. Neural Networks*, pp. 1923-1928, June 1994.