USC–SIPI REPORT #269

Toward Microelectronics-Based Intelligent Systems Including Neural Network Understanding

by

Oscal Tzyh-Chiang Chen

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,

Szu-Hsien Chen

and

Pei-Pei Chang,

for their love and encouragement.

Acknowledgements

I would like to express my deepest thanks to my research advisor Professor Bing J. Sheu for his guidance and support during my Ph.D. research. I wish to extend my sincere appreciation to Professor Ted Berger and Professor Zhen Zhang for serving as my dissertation committee members. I would also like to thank them along with Professor Irving Reed, Professor Richard Leahy and Professor C.-C. Jay Kuo for serving on my qualifying examination committee.

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 Alvin Despain, Professor Alice C. Parker, Ms. Ramona Gordon, Ms. Anna Fong, Ms. Gloria Halfacre and Ms. Susan Moore in the Electrical Engineering Program, for providing me the great research environment for my Ph.D. study at the University of Southern California (USC). This research work was conducted through connections with several research organizations at USC including Center for Neural Engineering (CNE), Signal and Image Processing Institute (SIPI), Center for Photonic Technology (CPT), and MOSIS Service of Information Science Institute (ISI). Very valuable suggestions from Professor Jerry M. Mendel, Professor Michael A. Arbib, Professor Christoph von der Malsburg were highly appreciated.

Discussions with graduated doctoral colleagues from VLSI Signal Processing Laboratory were truly valuable. They include Dr. Bang W. Lee and Dr. Joongho Choi on VLSI neural chips design, Dr. Ji-Chien Lee on digital data bus design, Dr. Wen-Jay Hsu and Dr. Sudhir Gowda on circuit simulation, and Dr. Wai-Chi Fang on image processing. Many thanks to Sa H. Bang and Josephine C.-F. Chang for helping to obtain some simulation

results. I also thank Robert C.-H. Chang, Tony H.-Y. Wu, and Vincent W.-J. Wang for managing the computing facility. Interactions with Hiroto Okada and Barton Sano were also very useful.

Family support helped me to finish this dissertation smoothly. I would like to thank my parents, Szu-Hsien Chen and Pei-Pei Chang for their love and understanding, and my grandparents Chung-Hwa Chen and Kei-Jen Pan for their encouragement.

Contents

List Of Tables

List Of Figures

Abstract

With the rapid progress of microelectronic technologies, an intelligent machine possessing skills of sensing, information processing, moving and thinking has been explored for many commercial, medical, and scientific applications. The high-speed and low-power features in VLSI design are pursued for many high-performance machines and portable devices. The medical prosthetic devices for cochlear implants and pacemakers for heart diseases are excellent examples. The various information processing schemes for medical images are also presented. In an artificial neural network, the backpropagation learning method, quasi-Newton method, non-derivative quasi-Newton method, Gauss-Newton method, secant method and simulated Cauchy annealing method to obtain an optimized solution have been investigated. By using the quasi-Newton method, a three-layered feed-forward network can successfully learn no-crossover trajectories. In a biologically-inspired neural network area, the nonlinear model of the functional properties of the hippocampal formation has been developed. The architecture of the proposed hardware implementation has a topology highly similar to the anatomical structure of the hippocampus, and the dynamical properties of its components are based on experimental characterization of individual hippocampal neurons. Recently, multimedia systems have received a lot of attention from the industry for consumer markets, publishing, education and entertainment. Data compression is an important scheme for multimedia systems to reduce data storage and transmission costs. A self-organization neural network architecture is used to implement vector quantization for image compression. A modified self-organization algorithm, which is based on the frequency-sensitive cost function and the centroid learning rule, is utilized to construct the codebooks. This self-organization method is quite efficient and can achieve near-optimal results. A new adaptive vector quantization method based

on the Gold-Washing method is also presented. The algorithm is shown to reach rate distortion function for memoryless sources. The computation power of the move-to-front vector quantizer can reach 40 billion operations per second at a system clock 100 MHz by using a 0.8 μm CMOS technology. The algorithm and architecture for the Gold-Washing method can lead to the development of a high-speed image compressor with great local adaptivity, minimized complexity, and fairly good compression ratio. The work described in this dissertation has paved a way for further study of intelligent microelectronic systems.

Chapter 1

Introduction

This dissertation contributes to understand multimedia systems, artificial and biologically-inspired neural networks for different engineering and medical applications. The objective is to develop microelectronics intelligent systems based on the understanding of human brain [1.1]. The intelligent machine is an integrated information processing system which can communicate with the real world through different sensors such as audio and video channels, together with a large knowledge database. High-speed image processing, vision understanding, and 3-dimensional motion graphics provide the systems with visual capabilities. Speech recognition and synthesis techniques provide the systems with audio processing capabilities. The system is also equipped with microsensor and controller units to accomplish physical actions. Such a powerful multi-media data-fusion machine can be used in many places, and will help people in their business, education, and daily lives.

Artificial neural networks can be thought of as an architectural solution to common engineering problems such as optimization, pattern recognition, and control systems [1.2]. In electronic implementation, there are currently a variety of problems yet to be solved. Theory remains ahead of hardware implementations. The design choices are digital, or analog, or a combination of both. The implementation of neuron models can be oriented toward emulation of biological systems or simulation of neural network paradigms. In many real-world applications, thousands of neurons are required which can be effectively

implemented in multiple VLSI chips at present and in wafer-scale integration in the future. Therefore, development of an effective VLSI architecture which allows an array of neuroprocessors to function together is critical.

Data compression is essential in reducing the data transmission or storage costs for broad areas of applications such as high-definition television, teleconferencing, remote sensing, radar, sonar, computer communication, facsimile transmission, and multimedia system [1.3]. A computer with a multimedia system will play voice and music in high-fidelity digital-audio stereo, show movie-quality image, easily access or interact with industry, and consumer markets, publishing, education and entertainment systems. All of these applications require efficiently to process, store and communicate information. Traditionally, there has been a tradeoff between the benefits of employing data compression versus the computational costs incurred to perform the encoding and subsequent decoding. However, with the advent of cheap microprocessors and custom chips, data compression is rapidly becoming a standard component of communications and data storage. A data encoding/decoding chips can be placed at the ends of a communication channel with no computational overhead incurred by the communication processes.

The rest of this dissertation is organized as follows. A general introduction for microelectronic technologies is addressed in Chapter 2. Based on the advanced VLSI technologies, the commercial, medical, and military applications have been presented. In Chapter 3, the backpropagation learning method, quasi-Newton method, non-derivative quasi-Newton method, Gauss-Newton method, secant method and simulated Cauchy annealing method have been investigated. By using the quasi-Newton method, a three-layered feedforward network can successfully learn trajectories without crossovers. Chapter 4 presents the nonlinear model of the functional properties of the hippocampal formation. The architecture of the proposed hardware implementation has a topology highly similar to the anatomical structure of the hippocampus. VLSI design techniques, optical interconnection, and medical applications are well described in Chapter 5. In Chapter 6, a self-organization

neural network architecture is used to implement vector quantization for image compression. Chapter 7 describes the impact and conclusion of dissertation work. In Appendix A, the adaptive vector quantization method based on the Gold-Washing method is presented. The proposed algorithm and architecture can lead to the development of a high-speed image compressor.

References

[1.1] T. Kaminuma, G. Matsumoto, *Biocomputers,* Chapman and Hall, New York, 1991.

[1.2] A. D. Kulkarni, *Artificial Neural Networks for Image Understanding,* Van Nostrand Reinhold, New York, 1994.

[1.3] G. Held, T. R. Marshall, *Data Compression: techniques and applications: hardware and software considerations,* John Wiley & Sons, New York, 1991.

Chapter 2

Microelectronics-Based Intelligent Systems

What is the intelligence? The intelligence can be defined as a process which is comprised of learning, reasoning, and the ability to provide correct decision [2.1]. In order to develop a similar understanding of the mechanisms of intelligence, the research of machine intelligence helps us to understand natural intelligence. Since the high-performance smart machine requires a lot of computation power, the innovative VLSI designs at the architectural and circuit levels with the advanced manufacturing technologies can provide the high-speed information processing.

2.1 Commercial Applications

Technological advances are revolutionizing computers and electronic devices to support digital multimedia [2.2], which stimulate the development of a wide spectrum of applications, such as tele-conferencing and video entertainment. These applications coupled with virtual reality will result in a new generation of systems enabling effective tele-personal interactions between individuals via their multimedia home workstations. The integration of virtual reality techniques with multimedia teleconferencing leads to the development of tele-virtual conferencing systems, that synthesize panoramic, life-like three-dimensional video images and stereophonic audio. Some interesting applications of the multimedia systems are listed as follows.

(1) two-way interactive broadcast media,

(2) interactive television, video games, music, and movie,

(3) virtual-reality entertainment and education systems,

(4) interactive magazines and books,

(5) interview, working, shopping, and museum tour at home,

(6) health diagnosis at home.

The exciting technological and business trends are development of home and office electronics with digital computer and communication technologies including the built-in intelligence and multimedia systems [2.3]. Reliable person identification, using pattern-recognition techniques applied to visual and speech patterns, replace locks and keys in many instances. Office automation products such as powerful pen- and voice-based personal digital assistants and voice translators are very useful. Home and office electronics will have interactive speech control. An intelligent robot will act like a family servant or a personal secretary. The smart vehicles will improve the safety of commutation and reduce the road traffic in the future.

2.2 Medical Applications

While human diagnosticians will continue for many years to examine images from X-ray machines, CAT scanners, nuclear-magnetic-resonance scanners, and supersonic scanners, a high degree of confidence will ultimately be placed in the ability of computerized systems to detect and diagnose problems automatically. Lifetime patient records and histories will be maintained in nationally or internationally coordinated data banks in place of today's disorganized system of partial, fragmented, and often illegible records. Intelligent software will be available to enable this extensive data bank to be analyzed and accessed quickly by both human and machine experts. Expert systems will influence virtually all diagnostic and treatment decision. Surgical operations will make extensive use of robotic assistants [2.4]. In types of surgery requiring very precise performance, e.g., in eye surgery, the actual operation will be carried out by the robot, with human doctors overseeing the operations [2.1].

Through the applications of computer technologies, handicaps associated with the major sensory and physical disabilities can be overcome during the next decade or two [2.1]. For the blind, reading machine will be pocket-sized devices that can instantly scan not only pages of text but also signs and symbols found in the real-world. They will also be able to describe pictures and graphics, translate from one language to another, and provide access to on-line knowledge bases and libraries through wireless networks. Blind persons will carry computered navigational aids that will perform the functions of seeing-eye dogs. The deaf will have hearing machines that can display what people are saying. Eventually we may find suitable channels of communication directly into the brain to provide truly artificial sight and hearing. The physically handicapped will have their ability to walk and climb stairs, abilities that will overcome the severe access limitations wheel chairs impose. One that has shown promising in experiments at a number of research institutes, is direct electrical stimulation of limb muscles. This technique effectively reconnects the control link that was broken by spinal cord damage.

2.3 Military Applications

A profound change in military strategy arrives. Virtual reality provides real-time interactive simulation for military training. Advanced frequency control and timing devices will improve the stability and accuracy which impact the performance of military communication, navigation, surveillance, electronic warfare, missile guidance, and identification-friend-or-foe systems [2.5]. Intelligent satellite systems will be developed to precisely determine the time, position and speed of vehicles or objects. The more developed nations increasingly rely on "smart weapon," which incorporate electronic copilots, pattern recognition techniques, and advanced technologies for tracking, identification, and destruction. A small and light-weight military land navigation device and smart combat system can be carried by the human soldier. In the future, most combat work will be done by robots.

References

[2.1] R. Kurzweil, *Intelligent Machines,* The MIT Press, Cambridge, 1990.

[2.2] J. A. Adam, "Interactive multi-media," *IEEE Spectrum,* vol. 30, no. 3, pp. 22-23, March 1993.

[2.3] H. Mizuno, "The fusion of home electronics with computer technologies," *Tech. Digest IEEE Inter. Solid-State Circuits Conf.,* pp. 20-23, San Francisco, CA, Feb. 1994.

[2.4] W. S. Ng, B. L. Davies, R. D. Hibberd, A. G. Timoney, "Robotic Surgery," *IEEE Engineering in Medicine and Biology,* pp. 120-125, Mar. 1993.

[2.5] J. R. Vig, "Military applications of high-accuracy frequency standards and clocks," *IEEE Trans. on Ultrasonics Ferroelectrics and Frequency Control,* vol. 40, pp. 522-527, Sep. 1993.

Chapter 3

Neural Network Learning on Space Trajectories

Neural networks are parameterized by a set of synaptic weights. The task of an optimization scheme for a neural network is to find a set of synaptic weights that make the network perform the desired function. The backpropagation learning method, quasi-Newton method, non-derivative quasi-Newton method, Gauss-Newton method, secant method and simulated Cauchy annealing method have been investigated. According to the computation time, convergence speed, and mean-squared error between the network outputs and desired results, the comparison of these six methods for learning a sine function has been presented. By using the quasi-Newton method, a three-layered feedforward network can successfully learn a circle. After the learning process, the network is connected into a recurrent network. Any point of the circular trajectory can be used as the starting point. With different starting points which can be inside or outside of the circular trajectory, the trained network also generates a circular trajectory. It indicates that the recurrent network has very good curvature attraction. In another example, one half of symbol-eight trajectory can also be reconstructed by the proposed learning procedure. Therefore, a feedforward network without the time-delay elements has the necessary capability to learn trajectories without crossovers.

3.1 Introduction

For engineering applications, neural networks can be thought of as an important architectural solution to common engineering problems such as image analysis, plant control,

optimization, and pattern recognition [3.1]. By far, the most popular neural networks today are the multilayer perceptrons [3.2], Kohonen's self-organizing maps [3.3], adaptive resonance theory (ART) networks [3.4], Hopfield nets [3.5], and cellular neural networks [3.6]. Many of the learning theories of engineering neural networks can be traced back to Rosenblatt's Perceptron and Bernard Widrow's Adaline. Another significant breakthrough was the discovery of the back-propagation training algorithm by Werbos [3.7]. In a neural network, the relationship between the input data and the desired results can be described by using nonlinear mathematical equations. Based on the equations, the quasi-Newton method, non-derivative quasi-Newton method, Gauss-Newton method, and secant method [3.8] can be applied. The quasi-Newton method and Gauss-Newton method are only used for solving the derivative functions. The non-derivative quasi-Newton method and secant method are extensions from the quasi-Newton method and Gauss-Newton method, irrespectively. They are suitable for solving the non-derivative functions.

Neural networks map the input vectors to the output vectors. This mapping might classify some input data, produce a control action, predict the next state of a system, and so on. In general, there are two categories of training methods for engineering neural networks: the deterministic and statistical categories [3.9]. The deterministic methods include the backpropagation learning method, quasi-Newton method, Gauss-Newton method. The statistical methods, such as simulated Cauchy annealing method [3.10,3.11] and Boltzmann machine [3.12], search for the globally optimal solution but they converge very slowly.

In order to learn the space trajectories such as circular and symbol-eight trajectories, recurrent neural networks [3.13,3.14] and time-delay neural networks [3.15-3.17] have been proposed by many researchers. In the biological study, recurrent networks are very commonly found in the brain. The recurrent neural networks are capable of holding memories in the network loops. The learning methods [3.13,3.14,3.18] for recurrent networks are required to integrate the information in the previous iterative stages. This learning operation is more complicated than that of feedforward network.

Time-delay feedforward networks have been successfully applied to speech recognition [3.19,3.20]. Based on the back-propagation learning method, the adaptive time-delay neural network in the continuous-time mode has been reported in [3.21]. To learn spatiotemporal topology by using adaptive time-delay neural networks was proposed by Dayhoff et al. [3.15-3.17]. They used a gradient decent method for training a feedforward network on the discrete-time mode. The training strategy is to map the current input data to the next incoming data. After the training process, the network is connected into a recurrent network so that the number of input neurons are the same as that of the output neurons. Before a trajectory is reconstructed from a time-delay neural network, a segment of the seed data are required to fill in the delay buffers. According to their simulation results [3.15], a quarter of a complete circle was used as the network input in order to generate the complete circle. Possible applications of time-delay neural networks include time-series analysis and prediction.

By using the quasi-Newton method, the multilayer feed-forward networks for learning space trajectories have been presented. The network without any feedback connection is used to learn the mapping from the input vector to the next incoming vector. After the training process, the network is connected into a recurrent network and only one starting point is provided to reconstruct the complete trajectory. The most common feed-forward networks are static, having no internal delays and responding to a particular input by immediately generating a specific output. This type of network is suitable for learning trajectories without crossovers.

3.2 Learning Methods

3.2.1 Backpropagation Learning Method

The backpropagation learning method has been very popular in the field of neural networks during the past decade. The synaptic weights are adjusted by using the backpropagation

technique in the negative gradient direction. During the training, the network performance is improved by minimizing the cost function,

$$E(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^{N} (\mathbf{d}_i - \mathbf{y}_i)^2, \tag{3.1}$$

where $\mathbf{W}$ is a synaptic weight matrix, N is the total number of input vectors, $\mathbf{d}_i$ and $\mathbf{y}_i$ are the desired and actual output vector, respectively, corresponding to the i^{th} input vector $\mathbf{x}_i$. The backpropagation learning rule in the batch-mode operation [3.22] can be described as follows,

(a) the hidden-to-output connection:

$$\begin{aligned} \Delta w_{pq} &= -\eta \frac{\partial E}{\partial w_{pq}} = \eta \sum_{i=1}^{N} \delta_{i,q} F(S_{i,p}) \ , \quad and \\ \delta_{i,q} &= (d_{i,q} - y_{i,q})\, y'_{i,q} \ , \end{aligned} \tag{3.2}$$

where w_{pq} is a synaptic weight that connects neuron p in the present layer to neuron q in the next layer, η is a learning rate constant, and F is the neuron transfer function. Here, $S_{i,q}$ is the weighted sum to neuron q, $y_{i,q}$ is the output of the q^{th} neuron, and $d_{i,q}$ is the desired output of the q^{th} neuron corresponding to the i^{th} input vector $\mathbf{x}_i$.

(b) the hidden-to-hidden or input-to-hidden connection:

$$\begin{aligned} \Delta w_{pq} &= -\eta \frac{\partial E}{\partial w_{pq}} = \eta \sum_{i=1}^{N} \delta_{i,q} F(S_{i,p}) \ , \quad and \\ \delta_{i,q} &= F'(S_{i,q}) \sum_j w_{qj} \delta_{i,j} \ . \end{aligned} \tag{3.3}$$

For the synaptic weight between the input neuron and hidden neuron, $F(S_{i,p})$ in Eq. (3.3) is equal to the input signal $x_{i,p}$.

3.2.2 Quasi-Newton Method and Non-derivative Quasi-Newton Method

The learning process of a feedforward network, in which the network connection strengths are modified systematically so that the response of the network progressively approximates the desired response, can be structured as a nonlinear optimization problem. The network performance is improved by minimizing the cost function in Eq. (3.1). In addition to using the first derivative of cost function like the backpropagation learning method, the second

derivative terms are also used in the Newton's method. By applying the truncated Taylor series expansion, $E(\mathbf{W})$ around $\mathbf{W}^{(k)}$ can be described as [3.8]

$$E(\mathbf{W}^{(k)} + \delta) \approx Q^{(k)}(\delta) = E(\mathbf{W}^{(k)}) + \mathbf{g}^{(k)^T}\delta + \frac{1}{2}\delta^T\mathbf{G}^{(k)}\delta, \tag{3.4}$$

where $\mathbf{g}$ is the first derivative term of $E(\mathbf{W})$, $\mathbf{G}$ is the second derivative term of $E(\mathbf{W})$, δ is equal to $\mathbf{W} - \mathbf{W}^{(k)}$, and $Q^{(k)}(\delta)$ is the resulting quadratic approximation at the k^{th} iteration. The $\mathbf{W}^{(k+1)}$ in the Newton's method is simply $\mathbf{W}^{(k)} + \delta^{(k)}$, where the correction $\delta^{(k)}$ is used to minimize the value of $Q^{(k)}(\delta)$. The method requires zero, first and second derivatives of E to be available at any point. The k^{th} iteration of the Newton's method can be performed in the following procedures,

$$\begin{aligned} &(a)\ \textit{solve}\ \mathbf{G}^{(k)}\delta = -\mathbf{g}^{(k)}\ \textit{for}\ \delta = \delta^{(k)},\ \textit{and} \\ &(b)\ \textit{set}\ \mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \delta^{(k)}. \end{aligned} \tag{3.5}$$

The second derivative of the cost function with respect to the synaptic weights provides information about the curvature of the error surface. By using the first and second derivatives information, the network can be trained to reach the minimum of the cost function. Since the computation complexity of the second derivative is very high, the quasi-Newton method can be applied by using an iterative approximation scheme for the inverse second derivative term. This approach avoids direct computation of second derivative, and computational complexity is reduced by a factor of O(N) [3.8]. The basic quasi-Newton algorithm consists of the following steps:

$$\begin{aligned} &(a)\ \textit{set a search direction}\ \mathbf{s}^{(k)} = -\mathbf{H}^{(k)} \cdot \mathbf{g}^{(k)}, \\ &(b)\ \textit{let}\ \mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \eta\mathbf{s}^{(k)},\ \textit{and} \\ &(c)\ \textit{update}\ \mathbf{H}^{(k)}\ \textit{to}\ \mathbf{H}^{(k+1)}, \end{aligned} \tag{3.6}$$

where $\mathbf{H}$ is the approximate inverse second derivative matrix, $\mathbf{s}$ contains weight-change information, η is the learning rate constant, and k is the iteration index. The key feature of the algorithm is the updating strategy for the approximate inverse second derivative

matrix. The Broyden, Fletcher, Goldfarb and Shanno (BFGS) technique [3.8] can be applied,

$$\begin{aligned}
\mathbf{H}^{(k+1)} &= \mathbf{H}^{(k)} + (1 + \frac{\gamma^{(k)T}\mathbf{H}^{(k)}\gamma^{(k)}}{\delta^{(k)T}\gamma^{(k)}})\frac{\delta^{(k)}\delta^{(k)T}}{\delta^{(k)T}\gamma^{(k)}} - \frac{\delta^{(k)}\gamma^{(k)T}\mathbf{H}^{(k)} + \mathbf{H}^{(k)}\gamma^{(k)}\delta^{(k)T}}{\delta^{(k)T}\gamma^{(k)}}, \\
\gamma^{(k)} &= \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}, \text{ and} \\
\delta^{(k)} &= \eta\mathbf{s}^{(k)} = \mathbf{W}^{(k+1)} - \mathbf{W}^{(k)}.
\end{aligned} \tag{3.7}$$

The initial matrix $\mathbf{H}^{(0)}$ is usually selected to be a unity matrix. In the quasi-Newton method, the BFGS formula is suitable because it can provide a better performance than the other formulas such as the rank-one formula and Davidon-Fletcher-Powell (DFP) formula [3.8]. For the non-derivative quasi-Newton method, the first derivative function $\mathbf{g}$ is replaced by the finite-difference function,

$$\mathbf{g}^{(k+1)} = \frac{E(\mathbf{W}^{(k)} + \mathbf{\Delta}) - E(\mathbf{W}^{(k)})}{\mathbf{\Delta}}, \tag{3.8}$$

where $\mathbf{\Delta}$ contains small values.

3.2.3 Gauss-Newton Method and Secant Method

The pertinent mathematical equations can be constructed according to the relationship between the input vectors and desired output vectors. The synaptic weights are the unknown variables in the equations. If there are n input vectors and m synaptic weights, the n equations with m unknown variables are to be solved. Usually, the number of input vectors for the neural network is larger than that of synaptic weights. These equations contribute to an over-determined system so that there are no exact solutions. On the other hand, if the neural network is mapped into a well-determined system or an under-determined system, appropriate solutions for synaptic weights can be easily found. In the over-determined system, the least squared-error solutions can be obtained by minimizing the cost function in Eq. (3.1). The mathematical equations of the neural network can be represented as

$$\mathbf{a}_i(\mathbf{W}) = \mathbf{d}_i - \mathbf{y}_i, \tag{3.9}$$

where $\mathbf{d}_i$ and $\mathbf{y}_i$ are the desired and the actual output vectors, respectively, corresponding to the i^{th} input vector $\mathbf{x}_i$. The m x n Jacobian matrix $\mathbf{J}$ can be obtained through the differentials of Eq. (3.9). The iteration process of the Gauss-Newton method is

$$\begin{aligned} &(a)\ solve\ \mathbf{J}^{(k)}\mathbf{J}^{(k)^T}\mathbf{s} = -\mathbf{J}^{(k)}\mathbf{a}^{(k)}\ \ for\ \mathbf{s} = \mathbf{s}^{(k)}, \\ &(b)\ set\ \mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \eta\mathbf{s}^{(k)}. \end{aligned} \tag{3.10}$$

For the non-derivative scheme, the secant method which is modified from the Gauss-Newton method can be applied. Here, the generalized secant method based on Broyden's approach [3.8] is used. The Jacobian matrix $\mathbf{J}$ in the Gauss-Newton method is replaced by the following equations,

$$\begin{aligned} &\mathbf{z}^{(k)} = \mathbf{a}^{(k+1)} - \mathbf{a}^{(k)},\ \ and \\ &\mathbf{J}^{(k+1)} = \mathbf{J}^{(k)} + \frac{\mathbf{s}^{(k)}(\mathbf{z}^{(k)} - \mathbf{J}^{(k)^T}\mathbf{s}^{(k)})^T}{\mathbf{s}^{(k)^T}\mathbf{s}^{(k)}}. \end{aligned} \tag{3.11}$$

The above formula is used to produce an approximate Jacobian matrix for Eq. (3.9). The updating formula in the generalized secant method is the same as in Eq. (3.10) for the Gauss-Newton method.

3.2.4 Simulated Cauchy Annealing Method

The simulated Cauchy annealing method proposed by Szu [3.10,3.11] is an useful statistical method for network training. The method has three important features: (i) States are generated with a probability density that has a Gaussian-like peak and lorentzian wings that imply occasional long jumps among local sampling. (ii) The canonical ensemble for a state acceptance probability allows occasional hill-climbing among descents. (iii) An artificial cooling temperature enters both (i) and (ii) as a control parameter of noise.

The cost function of the simulated Cauchy annealing method is the same as in Eq. (3.1). The annealing temperature is

$$T(k) = \frac{T_c}{1 + k}, \tag{3.12}$$

where T_c is a initial temperature, and k is the number of iterations. The random displacement for generating a new vector $\mathbf{W}^{(k+1)}$ can be $T(k) * \tan(\theta)$, where θ is a random

value normalized between $-\pi/2$ and $\pi/2$. The acceptance criterion is based on the following equation,

$$P_T(\Delta E) = \frac{1}{1 + e^{(\Delta E/T(k))}}, \tag{3.13}$$

where ΔE is the cost function difference between the neighboring input stages. If ΔE is smaller than zero, the displacement for generating the new vector $\mathbf{W}^{(k+1)}$ is applied. Otherwise, a random search is performed to determine the updating of vector $\mathbf{W}^{(k)}$. If the random number generated between 0 and 0.5 is less than the acceptance criterion, then the displacement is used to generate a new vector $\mathbf{W}^{(k+1)}$.

3.3 Computer Analysis Results

A three-layered feedforward network was configured with 1 input unit, 2 hidden units, and 1 output unit. The input data range is from 0 to 1.5π with 128 sampling points. The two hidden units have the same transfer function,

$$F(S) = \frac{1}{1 + e^{-S}}. \tag{3.14}$$

The output neuron has a linear transfer function with a scaling factor r. Figure 3.1 shows the neural network configuration. The network function can be described as follows,

$$y(x) = r \cdot [w_5 + w_6 \cdot F(w_1 + w_2 \cdot x) + w_7 \cdot F(w_3 + w_4 \cdot x)], \tag{3.15}$$

where x is an input signal, y is the network output, and w_1, w_2, ..., and w_7 are the synaptic weights. The backpropagation learning method, quasi-Newton method, nonderivative quasi-Newton method, Gauss-Newton method, secant method, and simulated Cauchy annealing method were used to train this network for the sine function. Simulation results are listed in Table 3.1.

For the back-propagation learning method, the scaling factor r of the output neuron was chosen to be 0.5 which is a constant value during the training process. If the initial synaptic weights are zero, the synaptic weights at the same layer after training have the same results. In such condition, the network cannot be effectively trained. In order to

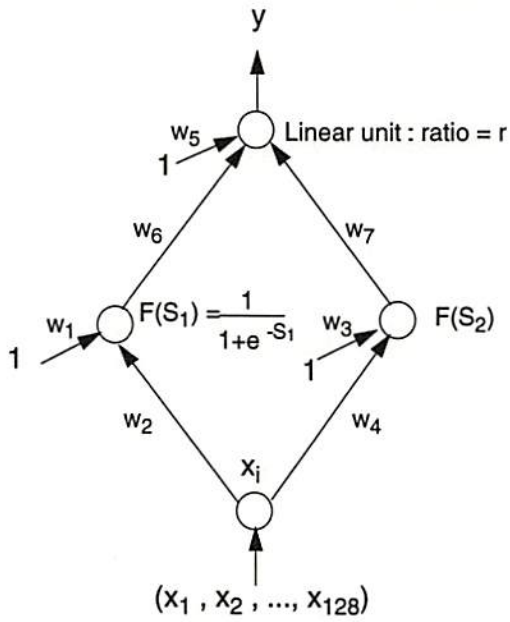


Figure 3.1: A three layered feedforward network.

achieve a good performance, initial synaptic weights are required to have different values. The learning rate constant is chosen to be 0.01. If a larger learning rate constant is used, the training process might not converge. After 1,050 iterations, the mean-squared error between the reconstructed and original sine curve is 0.000271. Figure 3.2(a) shows the plot of the cost function against the iteration index. The reconstructed curve and original sine curve are shown in Figure 3.2(b). For the quasi-Newton method, the network training is performed by optimizing the following equation,

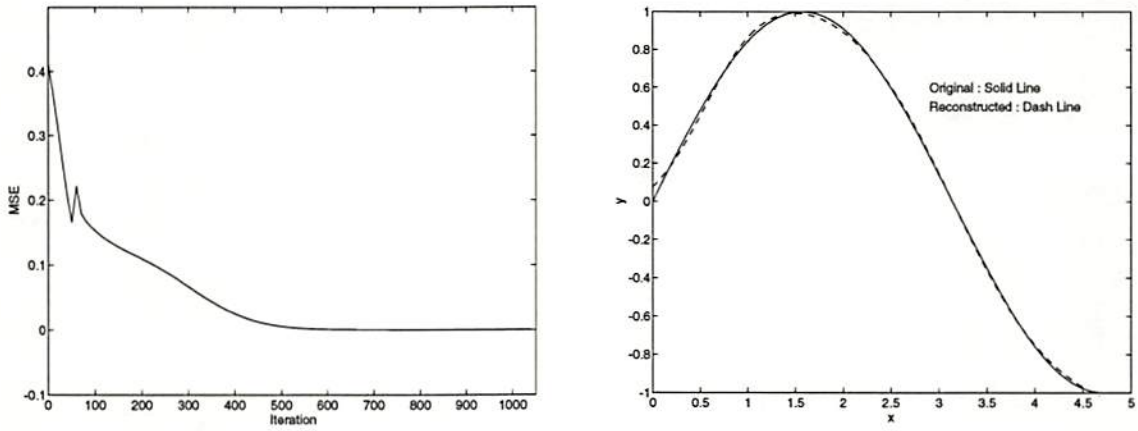
$$E(\mathbf{W}) = \sum_{i=1}^{128} [\sin(x_i) - y(x_i)]^2 , \tag{3.16}$$

where sin() is the sine function. The vector $\mathbf{W}$ consists of 8 components which are r, w_1, w_2, ..., and w_7. If an initial weight vector $\mathbf{W}$ is selected, the cost function E will be updated based on the information of its first and second derivatives. If a good initial vector is used, a very good performance can be achieved. The plot of the cost function at each iteration is shown in Fig. 3.3(a). After 4,650 iterations, a mean-squared error of 0.000009 can be achieved by using the quasi-Newton method. Figure 3.3(b) shows the plots of the reconstructed curve and original sine curve. They are almost perfectly matched. For the

non-derivative quasi-Newton method, the derivative function in the quasi-Newton method is replaced by the finite-difference function. If the initial vector for the quasi-Newton method in Table 3.1. was used for the non-derivative quasi-Newton method, a mean-squared error of 0.045978 could be achieved. It indicates that the finite-difference function is not a perfect approximation of the derivative function and the final result converged to a different local minimum. Therefore, a different initial vector should be used for the non-derivative quasi-Newton method. After 5,000 iterations, a good performance can be achieved with a mean-squared error of 0.000058. Figure 3.4(a) shows this plot of the cost function at each iteration for non-derivative quasi-Newton method. The reconstructed curve and original sine curve are shown in Fig. 3.4(b).

Table 3.1: Performance comparison of six different learning methds. (MSE: mean-squared error)

Methods / Para-meters	Backpropagation		Quasi-Newton		Non-derivative Quasi-Newton		Gauss-Newton		Secant		Simulated Cauchy annealing	
	Initial	After learning	Initial	After learning	Initial	After learning	Initial	After learning	Initial	After learning	Initial	After learning
r	0.5	0.5	-0.25	8.13	0.25	-2.83	-0.5	22.37	-1.25	-4.07	0.5	0.60
w_1	0.1	-1.76	0.1	1.20	0.25	4.87	0.5	0.55	-0.5	-0.54	1.0	-4.96
w2	1.0	3.10	1.5	-0.45	1.0	-1.60	-1.5	-1.85	0.5	1.85	1.0	1.62
w_3	0.2	-5.61	0.1	2.45	-0.5	0.54	0.5	-4.90	-0.5	4.90	1.0	0.57
w_4	-1.0	1.79	-1.5	-0.81	-1.0	-1.84	1.5	1.60	-2.5	-1.60	1.0	-1.89
w_5	0.3	-0.21	0.1	0.38	0.5	0.43	0.5	0.06	-0.5	0.83	1.0	2.31
w_6	1.5	2.57	1.0	-3.59	1.5	-0.93	-1.0	-0.10	-0.5	-0.54	1.0	-4.34
w_7	-1.5	-4.66	-1.0	2.58	-1.5	0.77	1.0	-0.12	-1.5	-0.64	1.0	-3.54
MSE	0.000271		0.000009		0.000058		0.000058		0.000059		0.000059	
Iterations	1,050		4,650		5,000		16,000		17,000		161,589	



(a) (b)

Figure 3.2: Network training by using backpropagation learning method. (a) Cost function ver training iterations. (b) Reconstruction and original Sine curves

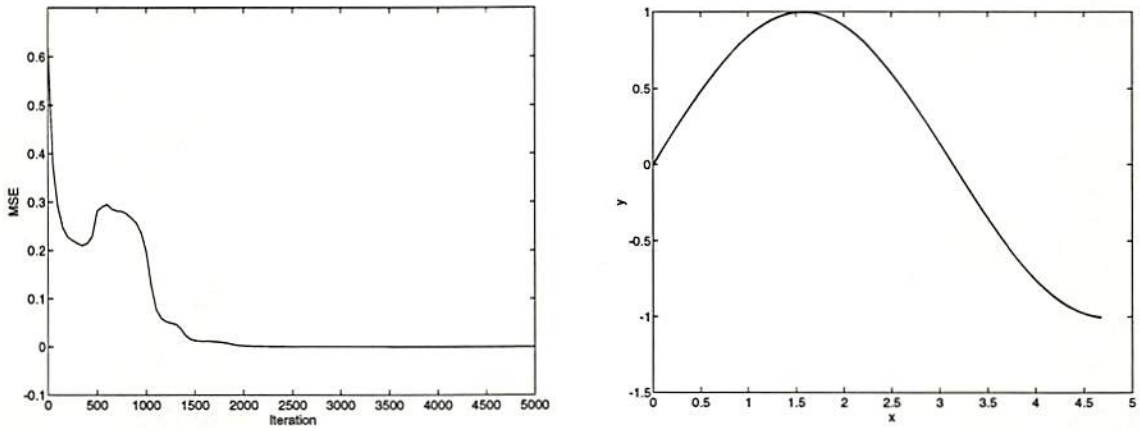
In the proposed neural network, the 128 sampling data $\mathbf{x}$ are used to describe the sine function. They can be contained in 128 equations which become an over-determined system. Each equation is described as follow,

$$a_i(\mathbf{W}) = \sin(x_i) - y(x_i) , \quad for\ i = 1, 2, ..., 128. \tag{3.17}$$

The 8x128 Jacobian matrix $\mathbf{J}$ can be obtained by taking the differentials of Eq. (3.17). This Jacobian matrix is

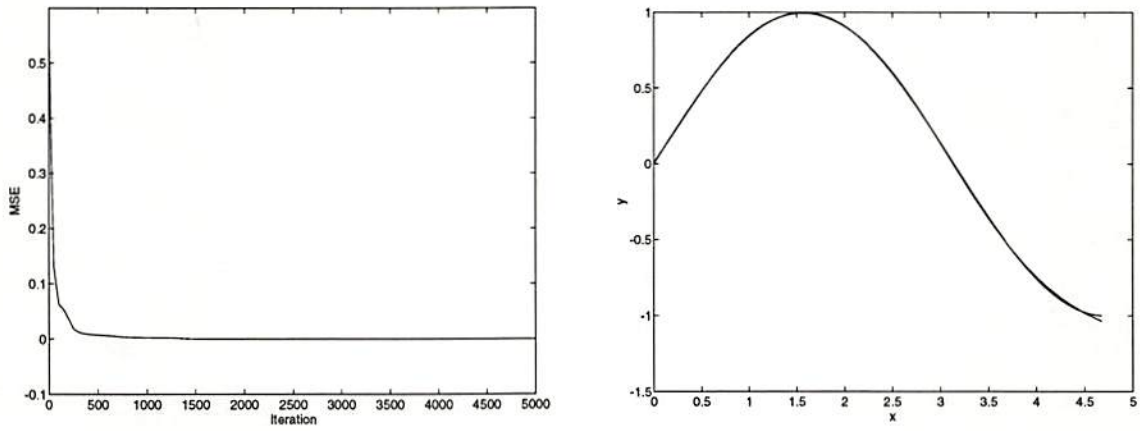
$$\mathbf{J} = [\frac{\partial \mathbf{a}}{\partial r} \frac{\partial \mathbf{a}}{\partial w_1} \frac{\partial \mathbf{a}}{\partial w_2} \frac{\partial \mathbf{a}}{\partial w_3} \frac{\partial \mathbf{a}}{\partial w_4} \frac{\partial \mathbf{a}}{\partial w_5} \frac{\partial \mathbf{a}}{\partial w_6} \frac{\partial \mathbf{a}}{\partial w_7}] . \tag{3.18}$$

According to Eqs. (3.10) and (3.11), the Gauss-Newton method and the secant method can be used. Since the operations of matrix inverse are required in the iterations of these two methods, the good initial vector $\mathbf{W}$ and learning rate constant η are very crucial in order to achieve converged results. If a poor initial weight vector or the learning rate constant value is used, singular results of matrix inverse could occur. The cost functions of the Gauss-Newton method and the secant method at each iteration are shown in Figs. 3.5(a) and 3.6(a), respectively. The reconstructed curves also are shown in Figs. 3.5(b) and 3.6(b).



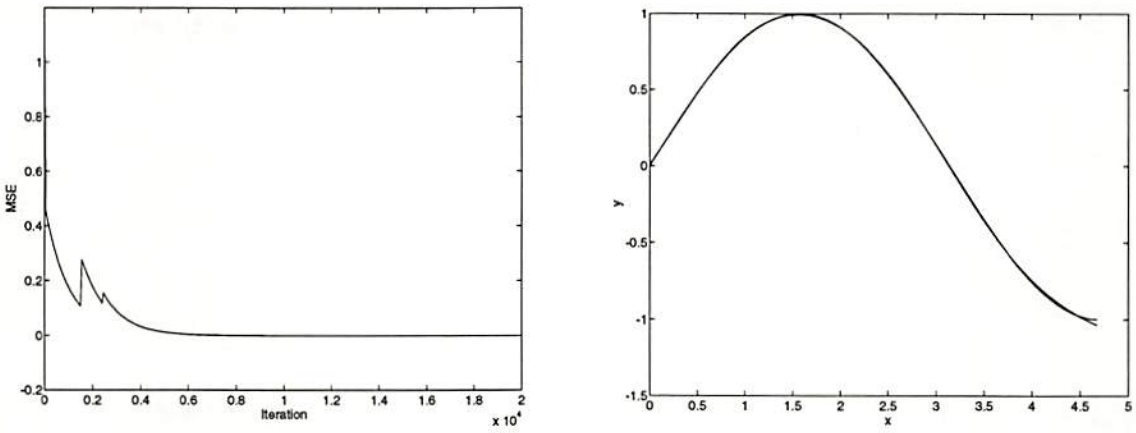
(a) (b)

Figure 3.3: Network training by using quasi-Newton method. (a) Cost function versus training iterations. (b) Reconstruction and original Sine curves



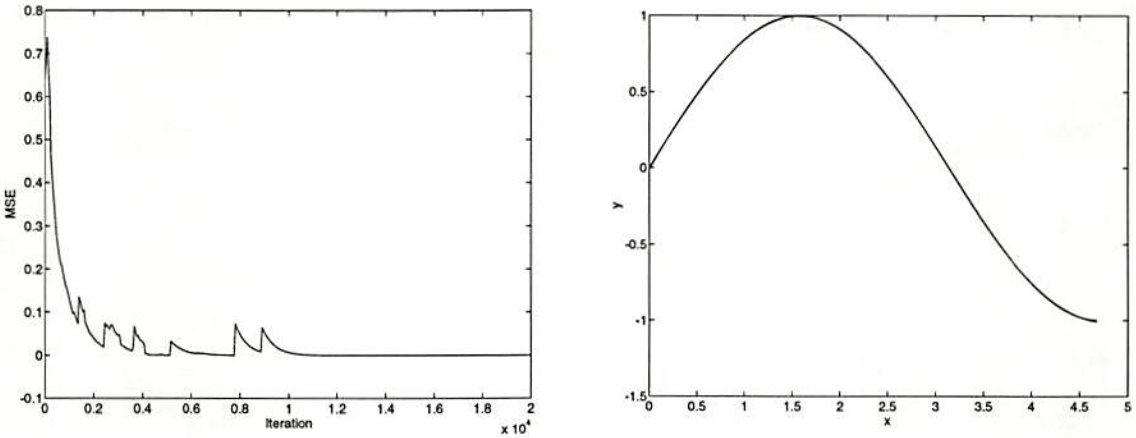
(a) (b)

Figure 3.4: Network training by using non-derivative quasi-Newton method. (a) Cost function versus training iterations. (b) Reconstruction and original Sine curves



(a) (b)

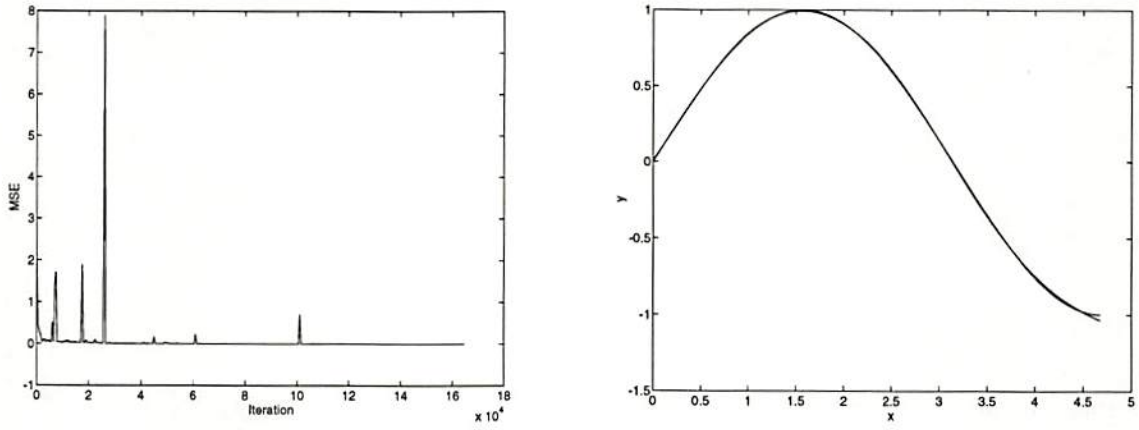
Figure 3.5: Network training by using Gauss-Newton method. (a) Cost function versus training iterations. (b) Reconstruction and original Sine curves



(a) (b)

Figure 3.6: Network training by using secant method. (a) Cost function versus training iterations. (b) Reconstruction and original Sine curves

The values of cost functions for the quasi-Newton method and Gauss-Newton method usually smoothly decayed without large disturbance during the learning process. In the simulated Cauchy annealing method, the steep hill-climbing occurs which strongly affects the plot of the cost function in Fig. 3.7(a). The converged result is not sensitive to the initial condition but sensitive to the annealing procedure. As compared with other methods, the required number of iterations for good convergence is quite large but the computational time for each iteration is very small. Fig. 3.7(b) shows the plots of the original sine curve and the reconstructed curve after 151,589 iterations of training.



(a) (b)

Figure 3.7: Network training by using simulated Cauchy annealing method. (a) Cost function versus training iterations. (b) Reconstruction and original Sine curves

The computation time and mean-squared errors of these six methods are listed in Table 3.2. The backpropagation learning method shows the behavior of quick convergence but with a very suboptimal solution. The best performance is obtained by the quasi-Newton method. The total computation time for the Gauss-Newton method and the secant method are quite large because they require heavy computation in matrix multiplication and inverse. The mean-squared errors of network training by the non-derivative quasi-Newton method, Gauss-Newton method, secant method, and simulated Cauchy annealing method are almost the same at their converged stages, but their final synaptic weights

and scaling factor values are quite different. It indicates that the network has a lot of local minima in the cost function. In view of the performance criteria of computation time, convergence speed, and mean-squared error, the quasi-Newton method stands out well and the Gauss-Newton method is also quite attractive. In real-world applications, most systems are more suitable for non-derivative operations so that the non-derivative quasi-Newton method and secant method can be very practical.

Table 3.2: Computation time and mean-squared errors of six different learning methods after 100, 200, 450 iterations, and the final convergence.

Iterations \ Methods		Back-propagation	Quasi-Newton	Non-derivative Quasi-Newton	Gauss-Newton	Secant	Simulated Cauchy annealing
100	Time (sec)	0.7	1.5	3.4	3.6	3.6	0.3
	MSE	0.175603	0.289970	0.062914	0.431622	0.738700	0.488601
200	Time (sec)	1.4	3.0	6.8	7.1	7.0	0.6
	MSE	0.140996	0.228342	0.036978	0.390267	0.597239	0.463296
450	Time (sec)	3.2	6.7	15.1	16.1	16.2	1.3
	MSE	0.014565	0.228855	0.008166	0.303796	0.297951	0.451719
Conver-gence	Time (sec)	7.4	68.6	170.9	577.6	610.6	467.9
	MSE	0.000271	0.000009	0.000058	0.000058	0.000059	0.000059

(Note: Computation time is obtained from the SUN-4 SPARC station-2.)

3.4 Neural Networks for Learning Space Trajectories

A three-layered feedforward network for learning a circular trajectory was configured with 2 input units, 6 hidden units, and 2 output units. The hidden and output neurons have

the same transfer function in Eq. (3.14). The desired circular trajectory is sampled at 128 reference points which become the training data. These points can be defined as

$$\begin{aligned} x_{i,1} &= \tfrac{1}{2}\,(\sin(\tfrac{\pi}{64}i) + 1)\,, \quad and \\ x_{i,2} &= \tfrac{1}{2}\,(\cos(\tfrac{\pi}{64}i) + 1)\,, \quad for\ 0 \leq i \leq 127. \end{aligned} \tag{3.19}$$

According to the computer analysis results in section 3, the quasi-Newton method can yield the best performance for learning a sine function. Here, only the quasi-Newton method based on the BFGS scheme is used for the feedforward network learning. If the input vector is $(x_{i,1},\ x_{i,2})$, then the desired output vector is $(x_{i+1,1},\ x_{i+1,2})$. The network learning is performed by optimizing Eq. (3.1). If an initial synaptic weight matrix is selected, the cost function E will be updated based on the information of its first and second derivatives. After 15,000 iterations, a mean-squared error of 0.000010 can be achieved. This trained network is connected into a recurrent network. Only one initial input vector is required to be the network input. Figure 3.8 shows the plots of the desired and reconstructed trajectories with the starting point (0.5, 1.0). The mean-squared error is increased to 0.000261 because the distortion is accumulated in the reconstruction process of the recurrent network. Any point of the desired circular trajectory can serve as a network input datum. Figure 3.9 shows the reconstruction performance versus the different input data. The worst case is the mean-squared error of 0.000859, and the plot of its reconstructed circle is shown in Fig. 3.10. This reconstructed circle still agrees well with the desired circular trajectory. Figures 3.11 shows the plots of two reconstructed trajectories for the input vectors, (1.0, 1.0) and (0.5, 0.5), which are not on the desired circle. It indicates that the recurrent network memorizes the topology of a circular trajectory and is not sensitive to the noise added to the input signals.

In a separate experiment, one half of the symbol-eight trajectory is used for the network learning. The same three-layered feedforward network is used. The desired trajectory is sampled at 64 reference points which can be defined as

$$\begin{aligned} x_{i,1} &= \tfrac{1}{2}\,(\sin(\tfrac{\pi}{32}i) + 1)\,, \quad and \\ x_{i,2} &= \tfrac{1}{2}\,(\sin(\tfrac{\pi}{64}i) + 1)\,, \quad for\ 0 \leq i \leq 63. \end{aligned} \tag{3.20}$$

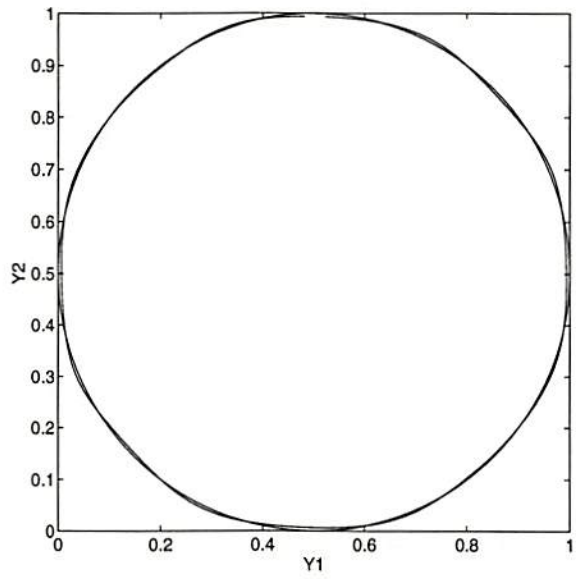


Figure 3.8: Desired circle and reconstructed circle at the initial vector (0.5, 1.0).

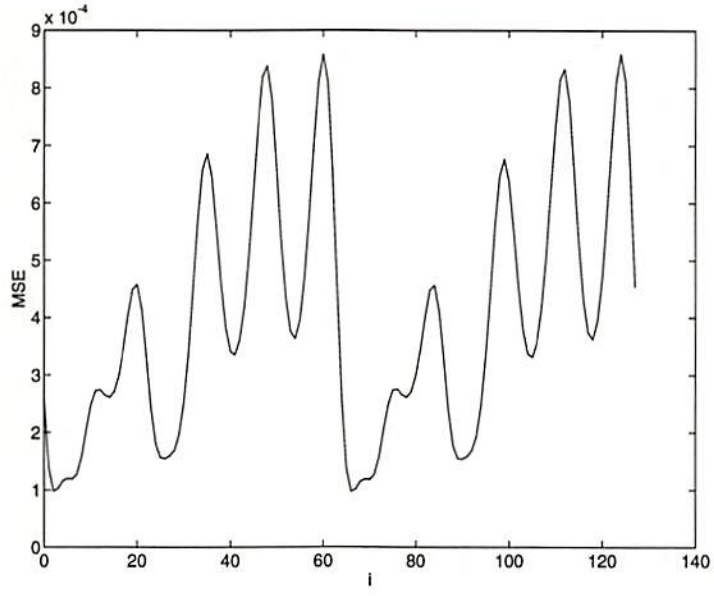


Figure 3.9: Reconstruction performance versus the different initial input vectors.

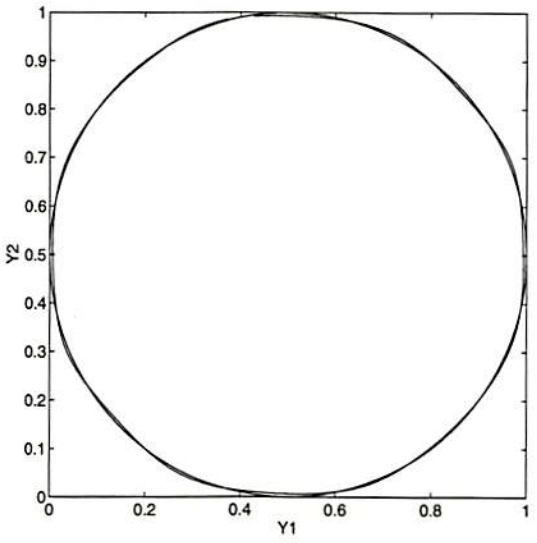
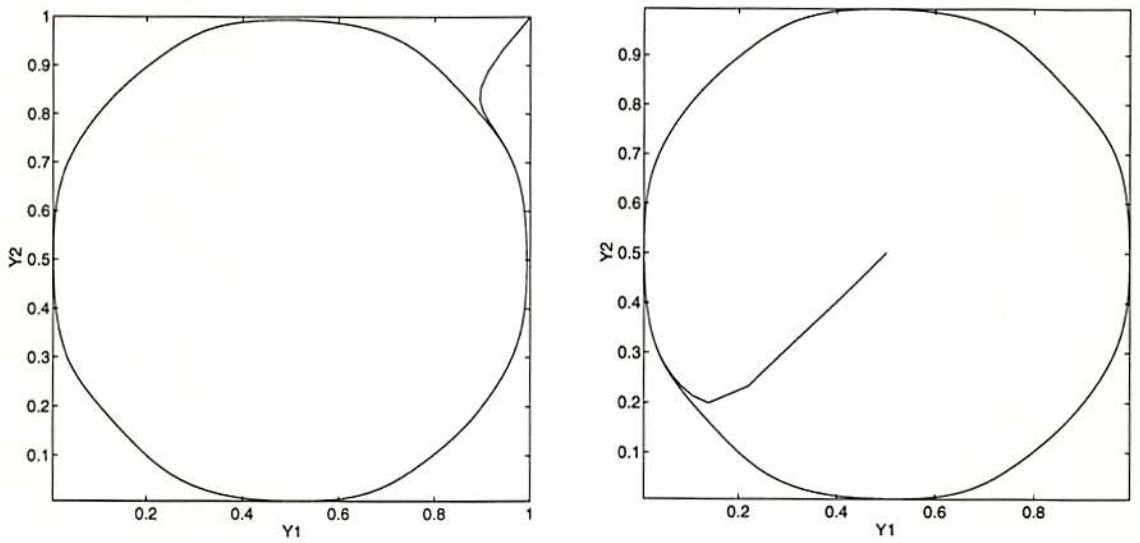


Figure 3.10: Desired circle and reconstructed circle at the initial vector (0.4025, 0.9904).



(a) (b)

Figure 3.11: Reconstructed trajectories from noisy input data. (a) Initial input vector (1.0, 1.0). (b) Initial input vector (0.5, 0.5).

The network was trained by using the quasi-Newton method. The mean-squared error between the desired and reconstructed trajectories is 0.000038 at the 10,000 iterations for the feedforward network. After the training process, the network is connected into a recurrent network. Figure 3.12 shows the plots of the reconstructed and desired trajectories for the initial input vector (0.5, 0.5). The mean-squared error is increased to 0.000501. Figure 3.13 shows the plots of two reconstructed trajectories for the input vectors, (1.0, 0.5) and (0.5, 0.75), which are not on the desired trajectory. In such a case, a very good reconstruction performance can also be achieved.

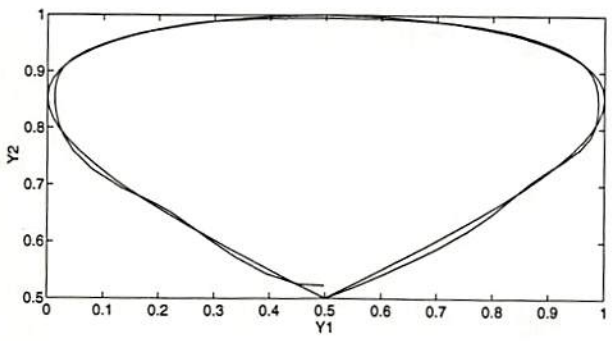
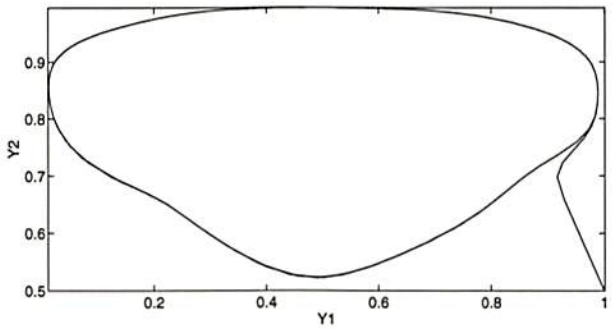
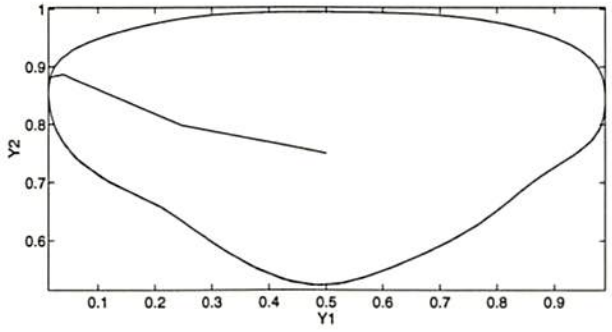


Figure 3.12: Desired trajectory and reconstructed trajectory at the initial vector (0.5, 0.5).

A three-layered feedforward network for learning a symbol-eight trajectory was configured with 2 input units, 12 hidden units, and 2 output units. After 20,000 iterations, the mean-squared error between the desired and reconstructed trajectories is 0.000059 for the feedforward network. If the trained network is connected to a recurrent network, the mean-squared error is increased to be 0.275 for the input vector (0.5, 0.5). Figure 3.14 shows the plots of the desired and reconstructed trajectories. In the intersection point (0.5, 0.5) of the symbol-eight trajectory, there are four possible moving directions. Here, the recurrent network only remembers one direction in the intersection point so that one half of symbol-eight trajectory is reconstructed. In order to determine the correct direction, the network is required to keep the information of the previous curvature positions as in the time-delay neural networks.



(a)



(b)

Figure 3.13: Reconstructed trajectories from noisy input data. (a) Initial input vector (1.0, 0.5). (b) Initial input vector (0.5, 0.75).

References

[3.1] R. Hecht-Nielsen, *Neurocomputing,* Addison-Wesley: Reading, MA, 1989.

[3.2] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine,* pp. 4-22, April 1987.

[3.3] T. Kohonen, *Self-Organization and Associative Memory,* 2nd Ed., Springer-Verlag: New York, 1988.

[3.4] S. Grossberg, "Competitive learning: from interactive action to adaptive resonance," *Cognitive Science 11,* pp. 23-63, 1987.

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

[3.6] L. O. Chua, L. Yang, "Cellular neural networks: applications," *IEEE Trans. on Circuits and Systems,* vol. 35, no. 10, pp. 1273-1290, Oct. 1988.

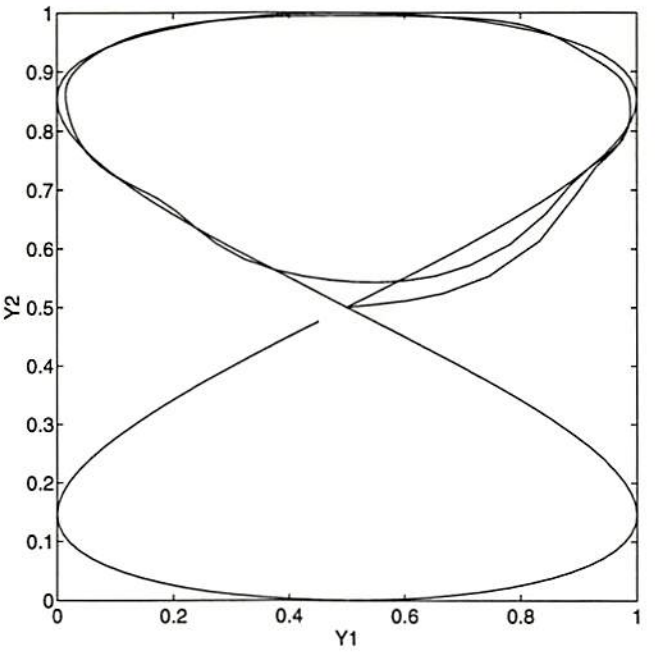


Figure 3.14: Desired trajectory and reconstructed trajectory at the initial vector (0.5, 0.5).

[3.7] P. J. Werbos, *Beyond Regression: new Tools for Prediction and Analysis in the Behavioral Sciences,* PhD thesis, Harvard University, 1974.

[3.8] R. Fletcher, *Practical Methods of Optimization* John Wiley & Sons: New York, 1991.

[3.9] J. Hertz, A. Krogh, R. G. Palmer, *Introduction to the Theory of Neural Computation,* Addison-Wesley: New York, 1993.

[3.10] H. Szu, R. Hartley, "Fast simulated annealing," *Physics Letters A* vol. 122, pp. 157-162, June. 1987.

[3.11] H. Szu, "Automatic Fault Recognition by Image Correlation Neural Network Techniques," *IEEE Transactions on Industrial Electronics,* vol. 40, np. 2, pp. 197-208, Apr. 1993.

[3.12] D. H. Ackley, G. E. Hinton, T. J. Sejnowski, "A learning algorithm for Boltzmann machine," *Cognitive Science,* vol. 9, pp. 147-169, 1985.

[3.13] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," *Physical Review Letters,* vol. 59, no. 19, pp. 2229-2232, Nov. 1987.

[3.14] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Proc. of IEEE Inter. Joint Conf. on Neural Networks,* vol. 2, pp. 365-372, Washington, 1989.

[3.15] D.-T. Lin, J. E. Dayhoff, P. A. Ligomenides, "Learning with the adaptive time-delay neural network," *Technical Report,* Systems Research Center, University of Maryland, College Park, Aug. 20, 1993.

[3.16] D.-T. Lin, P. A. Ligomenides, J. E. Dayhoff, "Learning spatiotemporal topology using an adaptive time-delay neural network," *Proc. of World Congress on Neural Networks,* vol. 1, pp. 291-294, Portland, July 1993.

[3.17] D.-T. Lin, J. E. Dayhoff, P. A. Ligomenides, "Trajectory recognition with a time-delay neural network," *Proc. of IEEE Inter. Joint Conf. on Neural Networks,* vol. 3, pp. 197-202, Baltimore, 1992.

[3.18] K. Doya, S. Yoshizawa, "Adaptive neural oscillator using continuous-time back-propagation learning," *Neural Networks,* vol. 2, pp. 375-385, 1989.

[3.19] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang, "Phoneme recognition: neural networks versus hidden markov models," *Proc. of IEEE Inter. Conf. Acoustics, Speech, and Signal Processing,* vol. 1, pp. 107-110, April 1988.

[3.20] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. 37, no. 3, pp. 328-339, March 1989.

[3.21] S. P. Day, M. R. Davenport, "Continuous-time temporal back-propagation with adaptive time delays," *IEEE Transactions on Neural Networks,* vol. 4, no. 2, pp. 348-354, March 1993.

[3.22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation," in *Parallel Distributed Processing, vol. 1,* edited by D. E. Rumelhart and J. L. McClelland, The MIT Press: Cambridge, MA, 1986.

Chapter 4

Algorithms and Architecture of the Biological Hippocampus

The hippocampal formation is a brain system which performs the cognitive functions of learning and memory. The architecture of the proposed hardware implementation has a topology highly similar to the anatomical structure of the hippocampus, and the dynamical properties of its components are based on experimental characterization of individual hippocampal neurons. A mixed analog/digital processor has been effectively implemented for the hippocampal dentate gyrus. The local data computation is executed by analog circuitry to achieve full parallelism and to minimize power consumption. Interneuron communication is carried out in the digital format to achieve network scalability. The proposed VLSI design of a nonlinear model of the hippocampus will not only facilitate for efficient emulation of biological systems or simulation of neural network paradigms but also produce useful knowledge and results for scientific and engineering applications.

4.1 Introduction

Research on the neurobiological substrates of learning and memory has progressed substantially during the past two decades, with a convergence of evidence identifying several brain systems critical for memory function [4.1] and several cellular and biochemical mechanisms of learning-induced synaptic plasticity [4.2]. Because of the complexity of the mammalian brain, however, further progress in understanding the relationship between neuronal processes and mnemonic processes will require the use of mathematical models and computer

simulations of the dynamic properties expressed by networks of neurons [4.3]. With regard to this goal, one of the most important unresolved issues is how to ensure that there is a sufficient degree of identity between mathematical models of neural networks and functional properties of real brain that the focus of neural network research remains biologically meaningful, and thus, likely to identify the basis for the unique information processing and storage capabilities of the brain.

Algorithms based on neural network paradigms have been demonstrated to be useful in signal processing and pattern recognition tasks. In order to effectively address complex real world problems, the neural networks must be scaled up, or modularized, and then must be efficiently implemented in hardware. In general, a neural network module consists of a large collection of simple processing elements. These simple processing elements execute mathematical algorithms to collectively carry out information processing through their responses to stimuli. There are technological constraints to the scale size and capacity of neural network hardware. In contrast, biological networks which incorporate features of real neurons and the connectivity of real neural networks have been shown to exhibit theoretical advantages in dimensional scaling and processing time. In order to develop a hardware implementation of the proposed model [4.1], the role of cellular and circuitry characteristics in the computational basis of hippocampal memory function have been studied.

Rapid advances in silicon fabrication and design technologies, especially the advent of very large-scale integration (VLSI) circuits, have made possible the implementation of engineering and biological neural networks. There has been much research in the area of analog neural network hardware implementations for various applications of adaptive signal processing. Lyon and Mead [4.4] described an analog electronic cochlea for speech recognition. Koch et al. [4.5] reported a real-time chip for rudimentary computer vision and robotics. Moore et al. [4.6] presented the VLSI implementation of an engineering neural system for color constancy. Van der Spiegel et al. [4.7] presented a simple analog neural computer for speech processing. Sackinger at al. [4.8] developed the analog neural

processor for high-speed character recognition. Mead et al. [4.9] developed an analog VLSI chip for binaural hearing. Sheu et al. implemented a motion sensor chip [4.10] and a neuroprocessor for self-organization mapping [4.11,4.12]. Many cellular neural network (CNN) implementations have been reported [4.13,4.14]. In addition to CMOS technology, various design and fabrication technologies such as BiCMOS [4.15], field-programmable gate array (FPGA) [4.16], and charge-coupled device (CCD) [4.17] also have been used for efficient construction of neurocomputing systems.

The custom VLSI hardware for neuron network applications can be constructed by the digital or analog design approaches. In the digital approach, a higher resolution and less noise-sensitive can be achieved. However, the silicon area and power consumption is higher than those of the analog design. On the other hand, an analog design can have many properties in common with real neural tissue. Analog computation can allow many neurons to collectively perform complicated functions in real time. Due to the regular and local connections among neural cells, the architecture of analog CNN circuitry is well-structured and the operation speed is independent of the network size. Usually, current saturation problems can occur in large-dimensional, global-connected neural networks. This effect can be alleviated by using the local-connected networks, such as CNN, or the networks with a small volume of interconnections.

4.2 Nonlinear Systems Model of the Hippocampus

The dynamic properties of individual hippocampal neurons were characterized experimentally using a nonlinear systems analytic approach [4.18]. Experiments were conducted using *in vitro* slice preparations of the hippocampus of New Zealand white rabbits. The primary afferents to the hippocampus, perforant path axons of the entorhinal cortex, were stimulated with a random interval train of electrical impulses: a series of 4064 impulses with a Poisson distribution inter-impulse intervals. The mean inter-event interval (Δ) was 500 ms, with a range of 1-5000 ms. Throughout random train delivery, electrophysiological

activity was recorded intracellularly from single granule cells of the dentate gyrus, which receive excitatory input from perforant path axons.

The nonlinear input/output properties of granule cells were defined as the kernels of a functional power series expansion:

$$y(t) = G_0 + G_1[h_1, x(t)] + G_2[h_2, x(t)] + G_3[h_3, x(t)] + ..., \quad (4.1)$$

where y(t) is the output of dentate granule cells, (g_i) is a set of mutually orthogonal functions, and (h_i) is a set of kernels which characterize the relationship between the input and output:

$$\begin{aligned} G_0(t) &= 0, \quad (4.2) \\ G_1(t) &= \int h_1(\tau)x(t-\tau)d\tau, \\ G_2(t) &= 2\iint h_2(\tau,\tau+\Delta)x(t-\tau)x(t-\Delta-\tau)d\Delta d\tau, \ \textit{and} \\ G_3(t) &= 6\iiint h_3(\tau,\tau+\Delta_1,\tau+\Delta_2)x(t-\tau)x(t-\tau-\Delta_1) \\ &\quad x(t-\tau-\Delta_1-\Delta_2)d\Delta_1 d\Delta_2 d\tau. \end{aligned}$$

The train of discrete input events defined by x(t) is a set of δ-functions. The first, second and third order kernels of the series are obtained by the process of orthogonalization using cross-correlation techniques applied to point process events [4.19].

The first order kernel, $h_1(\tau)$, is the average of all evoked granule cell responses occurring during train stimulation as shown in Fig. 4.1(a). The second order kernel, $h_2(\tau, \Delta)$ as shown in Fig. 4.1(b), represents the modulatory effect of a preceding stimulus occurring Δ ms earlier on the number or probability of granule cell activation by the most current stimulation impulse, where τ is the cell activation latency. The third order kernel, $h_3(\tau, \Delta_1, \Delta_2)$ as shown in Fig. 4.1(c) represents the modulatory effect of any two preceding stimuli occurring Δ_1 ms and Δ_2 ms earlier on the number or probability of granule cells activation by the most current stimulation impulse. In total, the kernel functions represent a complete characterization of the functional properties resulting from the interaction

among whatever system of neural elements is studied, and provide a basis for predicting the activity of those elements in response to any arbitrarily selected stimulus condition.

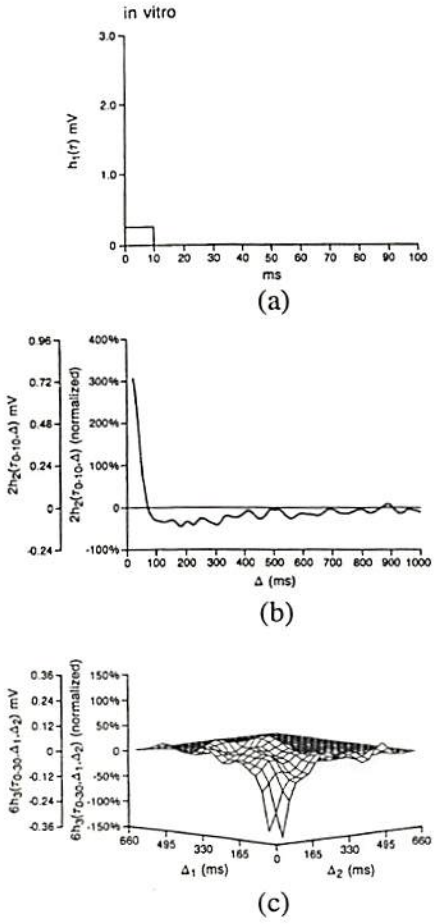


Figure 4.1: The kernels for dentate granule cells *in vitro*. (a) First order kernel. (b) Second order kernel. (c) Third order Kernel.

4.3 Engineering Neural Networks for the Hippocampus

In the area of neural network hardware design, there are currently a variety of problems yet to be solved. Theory remains ahead of hardware implementations. In electronic implementation, the choices are digital, or analog, or a combination of both. The analog design

approach can be divided into continuous-time and discrete-time electronics. A critical content in the VLSI implementation of learning techniques is the accuracy and convenience of using analog memory. A recent important improvement of VLSI hardware is in cellular neural networks due to the simplicity of architecture and the potential for many applications [4.13,4.14]. Figure 4.2 shows the relationship between engineering neural network and biological neural network for the hippocampus. The different hardware implementation schemes such as analog computing, analog subthreshold and pulse stream methods have been proposed by many researchers.

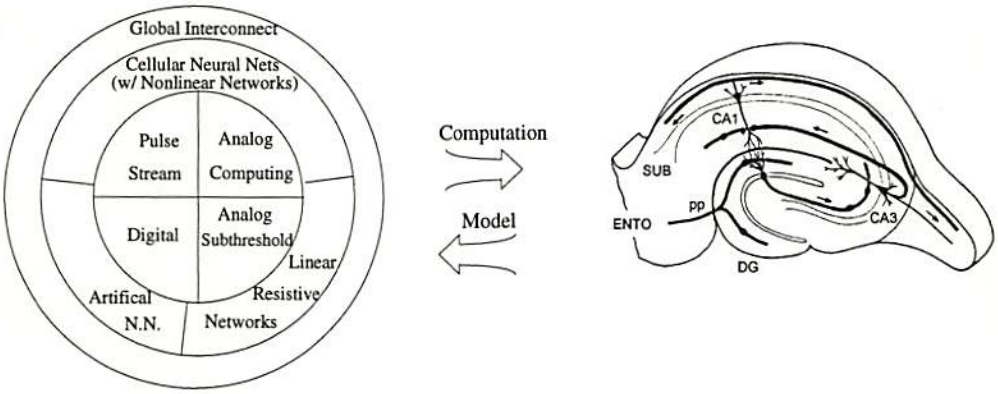


Figure 4.2: The relationship between engineering neural network and biological neural network.

The proposed neural network is based on the biological neural input/output and interconnection models. All neural operations are performed asynchronously according to biological neural functions. The neural network is a dynamic system which will be trained according to the stimuli. Potential applications are in the areas of data storage, classification, and understanding. In additional to the local neural interconnections, the proposed biological neural network has the interconnections between the neighboring neural layers such as CA1, CA3 and dentate [4.1,4.18]. The analog CNN design scheme is extensively studied for the proposed neural memory system.

The implementation of neuron models can be oriented toward emulation of biological systems or simulation of neural network paradigms. Progress in hardware implementation will contribute to a better understanding of paradigms and biological systems. In many real-world applications, thousands of neurons are required which can be effectively implemented in multiple VLSI chips at present and in wafer-scale integration in the future. Therefore, development of an effective analog VLSI architecture which allows an array of neuroprocessors to function together is critical. A mixed analog/digital approach is one of the good choices in the implementation of electronic neural systems. The local data computation is executed by analog circuitry to achieve full parallelism and to minimize power consumption. Interprocessor communication is carried out in the digital format to preserve signal strength across the chip boundary and to achieve network scalability by using an array of neural chips.

Due to the limitations by production yield and fabrication cost of semiconductor technologies, the high-performance analog neurocomputing IC chip will have a finite physical size. In real-world applications, there will be cases when the size of the data set is larger than the number of neurons on one chip. Effective partition of the data set with appropriate handling of the boundary problems is the key to map the input data to multiple chips for collective computation. The proposed analog VLSI design of a nonlinear model of the hippocampus will not only facilitate for efficient emulation of biological systems or simulation of neural network paradigms but also produce useful knowledge and results for scientific and engineering applications.

4.4 VLSI Design

The response properties of the granule cells have been analyzed by using the time-series neural model. The output of a neural cell can be described with the estimated kernel functions. The hardware implementation is used to realize the Eq. (4.1) which can be described by the functional block diagram as shown in Fig. 4.3. A VLSI architecture design of a neural cell, which is a hybrid analog/digital design, is shown in Fig. 4.4. Before

the operation of neuron function being performed, the signal RESET is issued to set the contents of input latch and shift registers to the V_{ss}. The input signal X is pipelined into the input latch. The previous input sequence is stored in the shift registers. The signals CLK1 and CLK2 are used to control the data shifting among the input latch and shift registers. The timing relationship in the neural model is mapped into the relationship of space connection in the shift registers. The signal READ is used as a enable signal for memory access. It controls the period of memory access which is determined by the response time of the analog current-sum circuitry. The AND gate functions are used to generate the memory access signal. From timing diagram shown in Fig. 4.5, the processing of the neural cell can be divided by three different operations: (a) shift the previous data and latch the incoming data, (b) generate memory access signals, fetch the memory data, and generate the output result, and (c) refresh the memory cell.

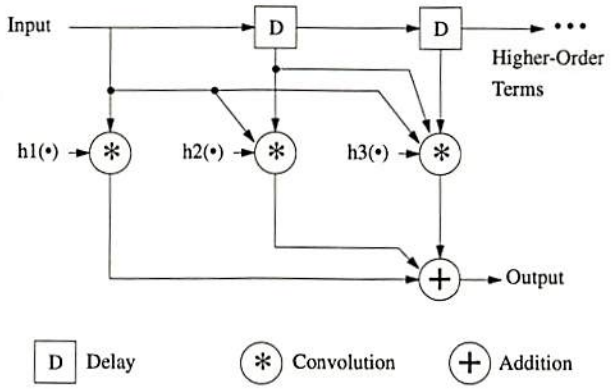


Figure 4.3: Functional block diagram of the nonlinear input/output property of a granule cell.

The h_1, h_2, and h_3 are stored dynamically in the analog memory cells which can be implemented by using the modified Gilbert multiplier. The circuit schematic of the compact and wide-range Gilbert multiplier for the neuron memory cell is shown in Fig. 4.6(a). The measured dc characteristics of the synaptic weight memory is shown in Fig. 4.6(b). Since the memory for synaptic weights is implemented by using the gate capacitances of MOS transistors, periodical refresh is required to prevent the leakage current through the

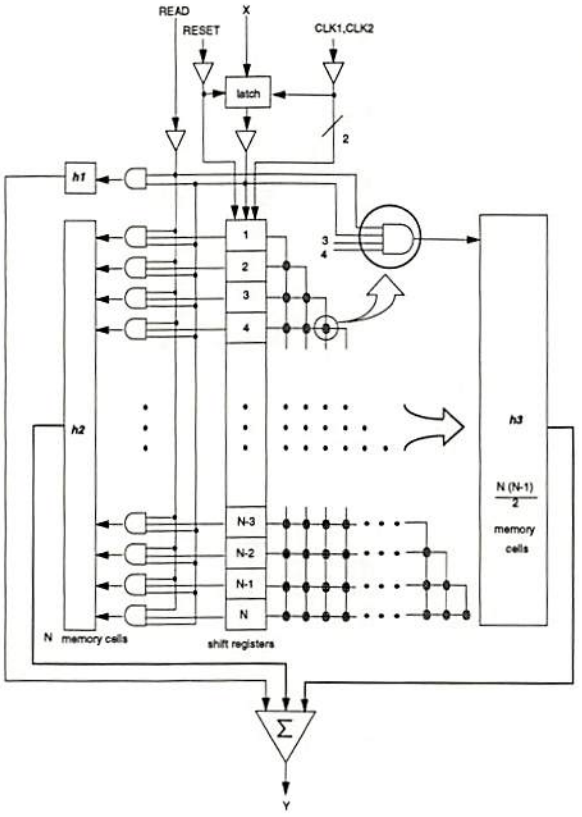


Figure 4.4: The VLSI architecture of a neural cell.

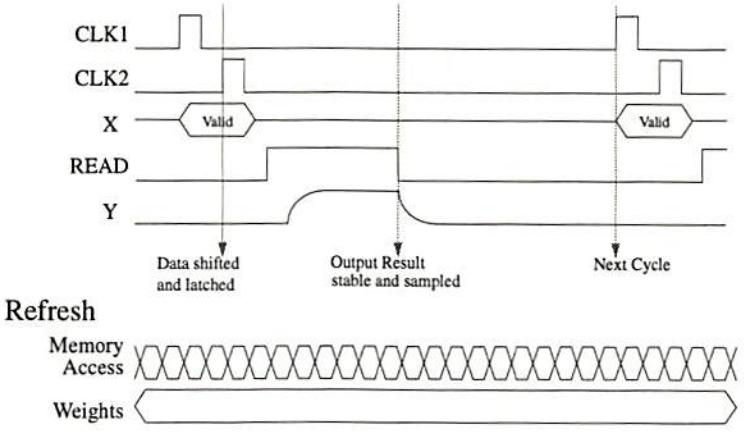
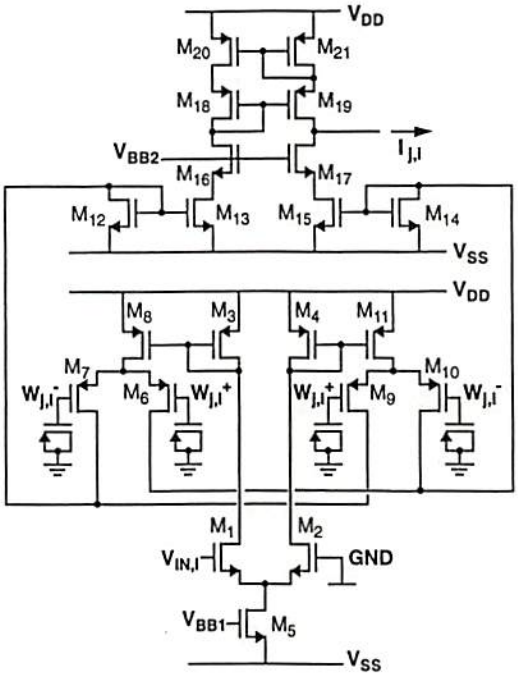


Figure 4.5: Timing diagram.

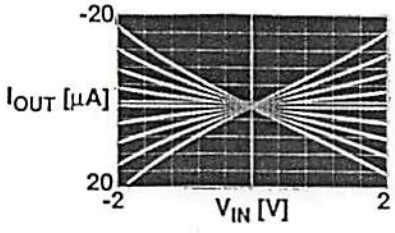
diffusion-to-substrate junction from changing the stored value. From the measurement on charge retention characteristics, a refresh cycle of around 0.1 sec is sufficient to retain the 8-bit accuracy [4.20]. Each updating or refreshing the data is performed while the correct memory address is provided. A decoder is required to decode the memory address in order to generate the control signals for the pass transistors. The input data will be stored in the synaptic weight memory through pass transistors. The refresh operations can be overlapped with the operations of the neuron function.

The currents from the synaptic weight memory will sink to the I-to-V converter which is an operational amplifier with a linear floating active resistor feedback. Figure 4.7(a) shows the circuit schematic of I-to-V converter for the operation of the current summation. The measured characteristics of this I-to-V converter is also shown in Fig. 4.7(b). Due to the operation range of I-to-V converter, the total current from the synaptic weight memory is limited. According to Fig. 4.7(b), the 500 μA is the maximum total current. In the Fig. 4.6(b), the linear current range of memory cell is adjustable by using different voltages, V_{IN}. Empirically, the contribution of higher order kernels can result in output values that range from -100% to +400% of the magnitude of the h_1 value. Since the currents are proportional to the values of h_1, h_2, and h_3, the total current will be within four times the current from the h_1 memory cell. Therefore, the N can be a large number, if the physical layout routing and propagation delay issues can be efficiently implemented. If the 500 μA is the maximum operation current in the I-to-V converter, the h_1 memory cell can provide the 125 μA with the maximum data resolution. The total current from the h_2 and h_3 will be within -250 μA to 375 μA so that the circuitry of I-to-V converter still performs in the linear operation range.

A set of 9 neurons is arranged in a two-dimensional 3x3 mesh array. Each neuron connects with its four neighboring neurons and has a physical size 1,000 λ x 1,200 λ with the number of stored time units equaling four (N=4). Figure 4.8 shows a layout of a complete neuron with a decoder for refreshing address. According to the SPICE-3 circuit simulation, the functionality of a neuron with 11 synaptic weights can be correctly

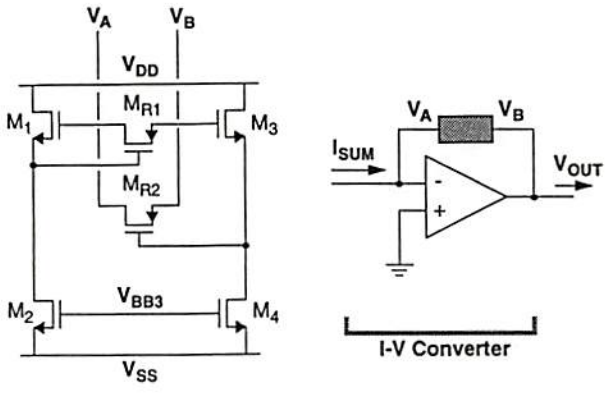


(a)

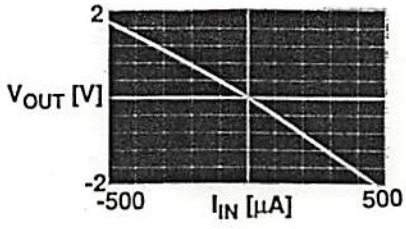


(b)

Figure 4.6: Synaptic weight memory. (a) Circuit schematic of the compact and wide-range Gilbert multiplier. (b) Measured dc characteristics.



(a)



(b)

Figure 4.7: Output function generator. (a) Circuit schematic of I-to-V converter with linear floating active resistor. (b) Measured characteristics of I-to-V converter.

addressed. The total response time is approximately 1 μs. The 3x3 neuron array with some testing modules, as shown in Fig. 4.9, was fabricated in a 2-μm double-polysilicon CMOS technology through the MOSIS Service of USC/Information Sciences Institute at Maria del Rey, CA.

References

[4.1] T. W. Berger, J. L. Bassett, "System properties of the hippocampus," *Learning and Memory: The Biological Substrates,* I. Gormezano, E. A. Wasserman (Eds.), Hillsdale, New Jersey: Lawrence Erlbaum, pp. 275-320, 1992.

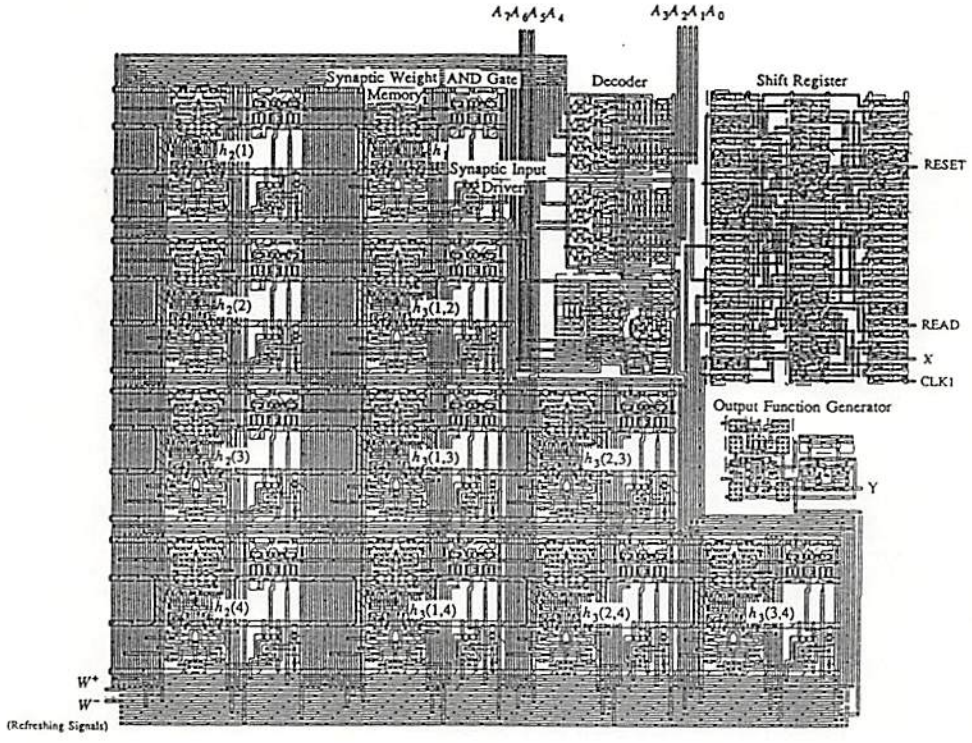


Figure 4.8: Layout of a complete neuron with N equaling four.

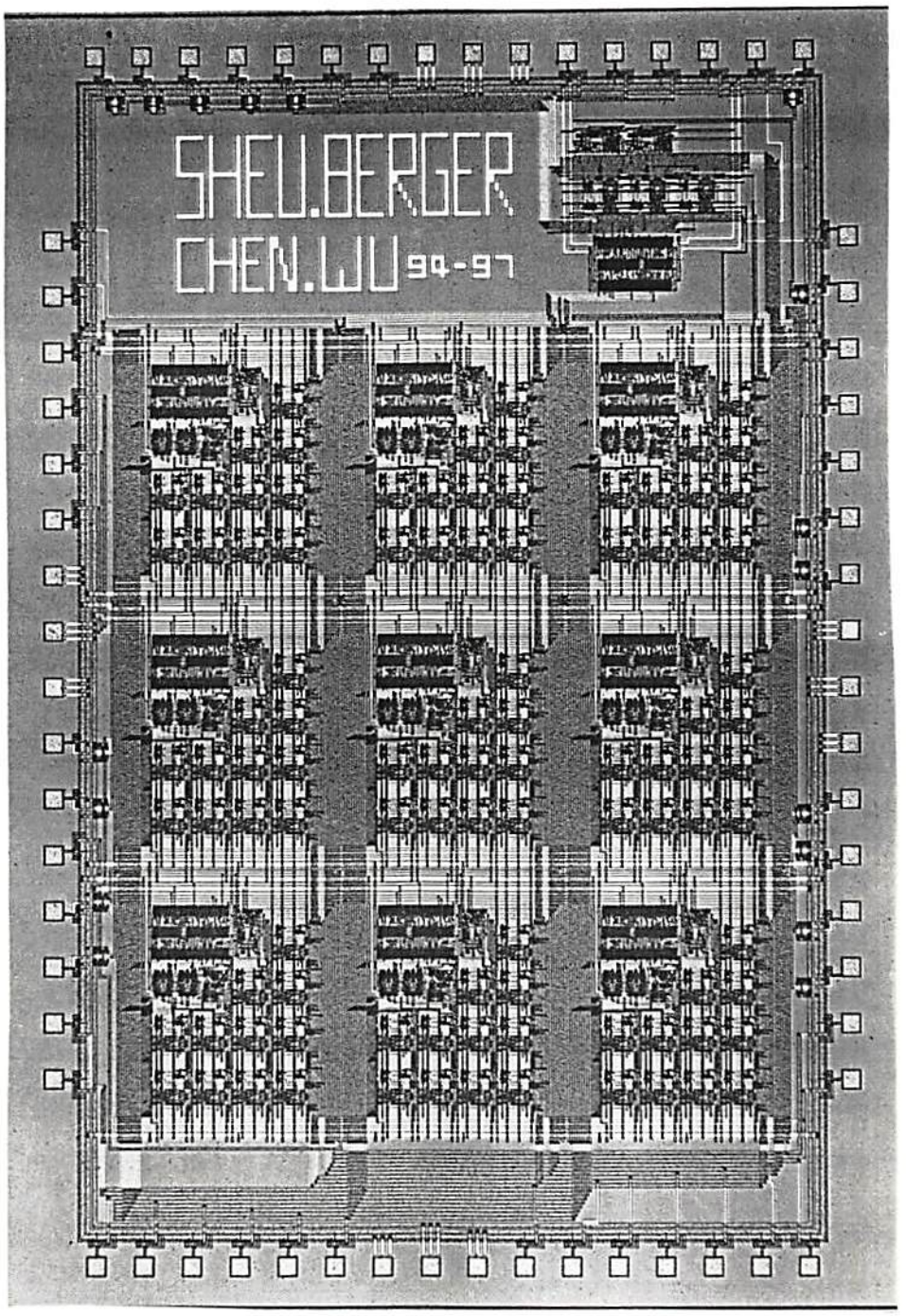


Figure 4.9: Prototype 3x3 neuron array.

[4.2] J. H. Byrne, "Cellular analysis of associative learning," *Physiological Reviews,* vol. 67, pp. 329-439, 1987.

[4.3] J. H. Byrne, W. O. Berry (Eds.), *Neural Models of Plasticity: Experimental and Theoretical Approaches,* New York: Academic Press, 1989.

[4.4] R. F. Lyon, C. A. Mead, "An analog electronic cochlea," *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. 26, no. 7, pp. 1119-1134, July 1988.

[4.5] C. Koch, W. Bair, J. G. Harris, T. Horiuchi, A. Hsu, J. Luo, "Real-time computer vision and robotics using analog VLSI circuits," *Advances in Neural Information Processing Systems 2,* Editor: D. Touretzky, pp. 750-757, Morgan Kaufmann: San Matao, CA, 1990.

[4.6] A. Moore, J. Allman, R. M. Goodman, "A real-time neural system for color constancy," *IEEE Trans. Neural Networks,* vol. 2, no. 2, pp. 237-247, Mar. 1991.

[4.7] J. Van der Spiegel, P. Mueller, D. Blackman, P. Chance, C. Donham, R. Etienne-Cummings, P. Kinget, "An analog neural computer with modular architecture for real-time dynamic computations," *IEEE J. Solid-State Circuits,* vol. 27, no. 1, pp. 82-92, Jan. 1992.

[4.8] E. Sačkinger, B. E. Boser, J. Bromley, Y. LeCun, L. D. Jackel, "Application of the ANNA neural network chip to high-speed character recognition," *IEEE Trans. Neural Networks,* vol. 3, no. 3, pp. 498-505, May 1992.

[4.9] C. A. Mead, X. Arreguit, J. Lazzaro, "Analog VLSI model of binaural hearing," *IEEE Trans. on Neural Networks,* vol. 2. no. 2, pp. 230-236, Mar. 1991.

[4.10] J.-C. Lee, B. J. Sheu, W.-C. Fang, R. Chellappa, "VLSI neuronprocessors for video motion detection," *IEEE Trans. on Neural Networks,* vol. 4, no. 2, pp. 178-191, Mar. 1993.

[4.11] B. J. Sheu and J. Choi and C.-F. Chang, "An analog neural network processor for self-organizing mapping", *Tech. Digest IEEE Inter. Solid-State Circuits Conf.,* pp. 136-137, San Francisco, CA, Feb. 1992.

[4.12] W.-C. Fang, B. J. Sheu, O. T.-C. Chen, J. Choi, "A VLSI neural processor for image data compression using self-organizing networks," *IEEE Trans. on Neural Networks,* vol. 3, no. 3, pp. 506-518, May 1992.

[4.13] L. O. Chua, L. Yang, "Cellular Neural Networks: Applications," *IEEE Trans. on Circuits and Systems,* vol. 35, no. 10, pp. 1273-1290, Oct. 1988.

[4.14] T. Roska, L. O. Chua, "The CNN universal machine: an analogic array computer," *IEEE Trans. on Circuits and Systems - II: Analog and Digital Signal Processing,* vol. 40. no. 3, pp. 163-173, Mar. 1993.

[4.15] T. Morishita and Y. Tamura and T. Otsuki, "A BiCMOS analog neural network with dynamically updated weights," *Tech. Digest IEEE Inter. Solid-State Circuits Conf.,* pp. 142-143, San Francisco, CA, Feb. 1990.

[4.16] E. K. F. Lee and P. G. Gulak, "A CMOS field-programmable analog array," *IEEE Jour. Solid-State Circuits,* vol. 26, no. 3, pp. 1860-1867, Dec. 1991.

[4.17] A. M. Chiang and M. L. Chuang, "A CCD programmable image processor and its neural network applications," *IEEE Jour. Solid-State Circuits,* vol. 26, no. 12, pp. 1894-1901, Dec. 1991.

[4.18] T. W. Berger, G. Barrionuevo, S. P. Levitan, D. N. Krieger, R. J. Sclabassi, "Nonlinear systems analysis of network properties of the hippocampal formation," *Neurocomputation and Learning: Foundations of Adaptive Networks,* J. W. Moore, M. Gabriel (Eds.), Cambridge, MA: MIT Press, pp. 283-352, 1991.

[4.19] H. Krausz, "Identification of nonlinear systems using random impulse train inputs," *Biological Cybernetics,* vol. 19, pp. 217-230, 1975.

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

Chapter 5

Smart Machines and Medical Applications

5.1 Powerful and Portable Machines

A powerful machine with interactive user interfaces, multimedia applications, and build-in intelligence is driving up computational complexity. The high-speed processing elements, high-throughput architecture and efficient networks are required to meet the future computation power. Recently, a 500-MHz processor [5.1] from NEC Corp. is fabricated in a 0.4-μm CMOS technology to provide a high-speed computation core. It uses an 8-stage pipelined data path. This extremely high clock speeds unearthed a number of design issues such as the clock distribution scheme, a programmable phase-lock loop for generating the internal 500-MHz clock, and the design of a stable power supply.

Consumers have driven to demand for portable and high-functionality products. The power consumption has to be reduced in order to maintain longer operation period. Low-voltage, very-low-current integrated circuits techniques have progressively found numerous applications in small portable battery-operated instruments such as medical devices, short-range low-frequency radio communication devices and palm computers. In these systems, low power consumption is the first constraint, for which speed and/or dynamic range of circuitry have to be sacrificed [5.2].

For both analog and digital circuits, there are limits to how much power can be reduced. In analog circuits, a desired signal-to-noise ratio must be maintained, while for

digital IC power, the lower limit is set by cycle time, operating voltage, and circuit capacitance. A smaller supply voltage is not only way to reduce power for digital circuits. More radical approaches include minimizing the number of device transitions needed to perform a given function, local suppression of clocks and reduction of clock frequencies, and even elimination of system clocks in favor of self-timed modules. Parallelism can be used to compensate the performance lost by slower clocks. For analog circuits, reducing signal-to-noise ratios to the minimum value can cut power.

5.2 Optical Sensors and Interconnection

High-speed computer communication and information processing are now approaching a bottleneck limited by electronic interconnection bandwidth, which can be alleviated by the optical interconnection. Optical interconnection and optoelectronic information processing systems can take the combined advantages of the optics and electronics. Due to the monolithic integration of many optoelectronic integrated circuits (OEICs) in the same substrate, optical information processing can be realized in a very compact hardware [5.3]. As optical interconnection and optoelectronic information processing become more popular, high-performance integrated optical receiving modules and the laser diodes are in strong demand.

In order to achieve the transmission data rate of higher than 1 GHz, specialized fabrication technologies based on compound semiconductors such as GaAs, InGaAs/InP, AlInAs/InGaAs have been utilized. The corresponding feature sizes have been continuously reduced to the sub-micron level [5.4]. The key engineering design challenge is due to the fact that the front-end unit of the receiver needs to operate at a very high rate and at a very low noise level. The following information processing units require the high-speed operation which could generate a large switching noise. The GaAs MESFET technology becomes a suitable candidate for this type of integration because various circuits can be integrated onto a single chip as fabrication technologies become more mature for large-volume production [5.5].

5.3 Medical Devices

The advances in implantable biomedical devices have saved lives and improved the quality of life for hundreds of thousands of patients suffering from various medical conditions. Nowadays, implantable cardioverter/defibrillators, drug delivery systems, neurological stimulators, bone growth stimulators, and other implantable devices make possible the treatment of a variety of diseases [5.6]. The future appears even more exciting, with the prospect of implantable devices and prosthese aiding in the recovery of body functions and the improvement in quality of life, as well as the saving of lives.

Each year, a lot of people die suddenly from heart disease. The prevention or termination of heart disease with medications has not proven to be either reliably effective or safe because of the variations in physiology and disease states from one individual to another. In contrast, electrical impulse therapy promises to be more consistent in its behavior and more adaptable as an individual's disease progresses. An artificial cardiac pacemakers incorporate sophisticated algorithms for stimulating the heart to produce an adequate cardiac output for daily life. It has two major components, a pulse generator and one or more leads [5.7]. The pulse generator is an electronic device that emits an electric pulse capable of stimulating the heart muscle to contract. The pulse generator requires a lead to transmit stimuli to the heart and cardiac signals back to the generator's sense amplifier. Today, most pulse generators can also sense natural electrical activity in the heart and supply a stimulus only as needed.

The perception of sound is actually a multifaceted task; frequency discrimination, loudness judgment, sound localization, and speech perception place different demands on the auditory system. A fairly recent development in treatment for the profoundly deaf is the cochlear implant. A cochlear implant is an electronic device that partially restores hearing via electrical stimulation of the inner ear. Candidacy requirements for cochlear implantation include a bilateral profound sensorineural hearing loss, little or no benefit from hearing aids, and no medical or radiological contraindications to surgery[5.8]. The Nucleus 22 Nultiple Channel Cochlear Implant [5.9] consists of both internal and external

components. The internal component consists of an electrode array made up of 22 pure platinum bands supported on a flexible silastic carrier. Each electrode band is separately connected to the receiver-stimulator via a platinum-iridium wire. The electrode array is surgically implanted into the scala tympani. Insertion depth of array varies depending on the condition of the cochlea. The maximum insertion depth for the array is 25 mm [5.9]

5.4 Biomedical Imaging

"Seeing is believing" is a powerful perception that has pervaded medicine from its outset. The natural inclination of health professionals to directly observe tissues and organs has prompted the development of various optical instruments such as microscopes, ophthalmoscopes, gastroscopes, endoscopes, and diverse devices for examining every accessible tissue or organ in the body [5.10]. The diversity of modern imaging methods challenges the imagination. The various probes (i.e. x-rays, nuclides, magnetic resonance) to elicit information regarding structure, chemical composition, physiological function and metabolism of internal organs. Such techniques can provide images displaying not only size, shape and location of internal structures, but also the spatial distribution of certain specific constituents.

Medical imaging has became a vital part of patient diagnostic systems. Almost all physical processes are performed by using image human anatomy, X-ray computed tomography (CT), angiography, positron emission tomography (PET), and magnetic resonance imaging (MRI). These medical imaging modalities have introduced significant changes in routine patient care, adding diagnostic evidence and expanding therapeutic efficiency. Nevertheless, the mechanisms of medical image understanding are complex. They rely particularly upon the knowledge of organ topologies and the relationships they share. Efficient image segmentation and restoration, motion detection, 3-D reconstruction [5.11], image registration [5.12] and database are very important techniques for medical image understanding. Based on these techniques, a medical expert system can provide initial diagnosis or suggestion for the physicians and patients.

5.4.1 Image Segmentation and Restoration

Segmentation subdivides an image into its constituent parts or objects. The level to which this subdivision is carried depends on the problem to be solved. Segmentation should stop when the objects of interest in an application have been isolated. Segmentation algorithms for monochrome images generally are based on one of two basic properties of gray-level values: discontinuity and similarity [5.13]. In the first category, the approach is to partition an image based on abrupt changes in gray level. The principal areas of interest within this category are detection of isolated points and detection of lines and edges in an image. The principal approaches in the second category are based on thresholding and region growing. Discontinuity or similarity of gray-level values of its pixels is applicable to both static and dynamic images. The motion information can often be used as a powerful cue to improve the performance of segmentation algorithms. Tissue classification of magnetic resonance images is a process in which image elements representing the same tissue type are grouped together and referred as a class. In addition to discontinuity and similarity approaches, statistical approach [5.14] is also very useful for segmentation of magnetic resonance images, especially textured images.

The ultimate goal of restoration techniques is to improve an image in some sense. The restoration techniques are oriented toward modeling the degradation and applying the inverse formulating a criterion of goodness. These techniques basically are heuristic procedures designed to manipulate an image in order to take advantage of the psychophysical aspects of the human visual system [5.13]. For example, contrast stretching is considered an enhancement technique because it is based primarily on the pleasing aspects it might present to the viewer, whereas removal of image blur by applying a deblurring function is considered a restoration technique.

5.4.2 Motion Analysis and Deformable Models

Motion can be characterized by observing the apparent motion of discrete set of features or brightness patterns in the images. Two major distinct approaches [5.15] have been developed for the computation of motion from image sequences. The first of these is based on extracting a set of relatively sparse two-dimensional features in the images corresponding to three-dimensional object features in the scene, such as corners, occluding boundaries of surfaces, and boundaries demarcating changes in surface reflectivity. Such points, lines or curves are extracted from each image. Inter-frame correspondence is then established between these features. The other approach is based on computing the optical flow or the two-dimensional field of instantaneous velocities of brightness values in the image plane.

A technique for modeling shape changes in a time series of biological images of arbitrary dimension is described in [5.16]. The technique consists of first segmenting the image to locate the specimen, and then parameterizing the specimen in the initial image with an orthogonal material coordinate system. The deformation of the material coordinate system caused by the changing shape of the specimen is then solved for by minimizing an energy function. The energy function is determined by the factors of brightness continuity and shape changing.

5.4.3 Data Visualization and Image Database

Image processing offers a powerful tool for medical diagnosis by using visual inspection. A computer vision approach can aid understanding of a series of sequential images. For example, the computation is addressed of local descriptors of the heart pumping function from ventricular contours. Physical constraints are exploited such as spatial smoothness of the displacement field and shape correspondence between ventricular boundaries during the beats [5.17]. Effective visualization schemes based on clinical information can be used to obtain the detailed operations of heart pumping.

Information management and communications systems for medical applications have been undergoing design and development for more than ten year now [5.18]. The aim

of these activities has been the development of distributed computer systems providing storage, processing, and communication services required by the medical community. One of the main critical issues of such systems is the handling of information (i.e., text, images, graphics, and voice) in a uniform way and the fast access through the network or storage media. The efficient compression schemes [5.19] can highly improve the performance of information management and communication.

References

[5.1] K. Suzuki et. al., "A 500 MHz 32b 0.4μm CMOS RISC Processor LSI," *Tech. Digest IEEE Inter. Solid-State Circuits Conf.*, pp. 142-143, San Francisco, CA, Feb. 1994.

[5.2] E. A. Vittoz, "Low-power design: ways to approach the limits," *Tech. Digest IEEE Inter. Solid-State Circuits Conf.*, pp. 14-18, San Francisco, CA, Feb. 1994.

[5.3] J. W. Parker, "Optical interconnection for advanced processor systems: A review of the ESPRIT II OLIVES programs," *IEEE/OSA Jour. Lightwave Technology,* vol. 9, no. 12, pp. 1764-1773, Dec. 1991.

[5.4] M. Soda, T. Suzaki, T. Morikawa, H. Tezuka, C. Ogawa, S. Fujita, H. Tekeura, and T. Tashiro, "A Si bipolar chip set for 10 Gb/s optical receiver," *Tech. Digest IEEE Inter. Solid-State Circuits Conf.*, pp. 100-101, San Francisco, CA, Feb. 1992.

[5.5] J. F. Ewen, K. P. Jackson, R. J. S. Bates, and E. B. Flint, "GaAs fiber-optic modules for optical data processing networks," *IEEE/OSA Jour. Lightwave Technology,* vol. 9, no. 12, pp. 1755-1763, Dec. 1991.

[5.6] W. Greatbatch, C. F. Holmes, "History of implantable devices," *IEEE Engineering in Medicine and Biology,* pp. 38-49, Sep. 1991.

[5.7] K. Stokes, "Implantable pacing lead technology," *IEEE Engineering in Medicine and Biology,* pp. 43-49, Jun. 1990.

[5.8] T. A. Zwolan, P. R. Kileny, "Cochlear implants for the profoundly deaf," *IEEE 6th Symposium on Computer-Based Medical Systems,* pp. 241-246, Ann Arbor, MI, Jun. 1993.

[5.9] *Nucleus 22 Channel Cochlear Implant System: Audiologist's Handbook* Englewood, CO, Cochlear Corporation.

[5.10] R. F. Rushmer, "Technologies and health care in the 21st century," *IEEE Engineering in Medicine and Biology,* pp. 50-52, June 1990.

[5.11] C. Barillot, "Surface and volume rendering techniques to display 3-D data," *IEEE Engineering in Medicine and Biology,* pp. 111-119, Mar. 1993.

[5.12] J. Y. Chiang, B. J. Sullivan, "Coincident bit counting - a new criterion for image registration," *IEEE Transactions on Medical Imaging,* pp. 30-38, Mar. 1993.

[5.13] R. C. Gonzalez, R. E. Woods, *Digital Image Processing,* Addison-Wesley: New York, 1992.

[5.14] Z. Liang, "Tissue classification and segmentation of MR images," *IEEE Engineering in Medicine and Biology,* pp. 81-85, Mar. 1993.

[5.15] J. K. Aggarwal, N. Nandhakumar, "On the computation of motion from sequences of images - a review," *Proceedings of the IEEE,* vol. 76, no. 8, pp. 917-935, Aug. 1988.

[5.16] K. Bartels, A. Bovik, S. J. Aggarwal, K. R. Diller, "The analysis of biological shape changes from multidimensional dynamic images," *Computerized Medical Imaging and Graphics,* vol. 17, pp. 89-99, 1993.

[5.17] G. Coppini, M. Demi, R. Calamai, G. Valli, "Artificial vision approach to the understanding of heart motion," *Journal of Biomedical Engineering,* vol. 14, pp. 321-328, Jul. 1992.

[5.18] G. Sudhakar, A. Karmouch, N. D. Georganas, "Design and performance evaluation considerations of a multimedia medical database," *IEEE Trans. on Knowledge and Data Engineering,* vol. 5, pp. 888-894, Oct. 1993.

[5.19] P. C. Cosman, R. M. Gray, R. A. Olshen, "Evaluating quality of compressed medical images - SNR, subjective rating, and diagnostic-accuracy," *Proceedings of the IEEE,* vol. 82, pp. 919-932, June 1994.

Chapter 6

Image Compression Using Self-Organization Networks

A self-organization neural network architecture is used to implement vector quantization for image compression. A modified self-organization algorithm, which is based on the frequency-sensitive cost function and centroid learning rule, is utilized to construct the codebooks. Performances of this frequency-sensitive self-organization network and a conventional algorithm for vector quantization are compared. The proposed method is quite efficient and can achieve near-optimal results. Good adaptivity for different statistics of source data can also be achieved.

6.1 Introduction

Image compression can be applied to the motion-picture industry and consumer electronics for high-definition TVs, advanced multi-media computer systems, remote sensing systems via satellite, aircraft, radar, sonar, teleconferencing or movie-on-a-chip systems. Efficient compression of data would significantly decrease both the communication and archival costs [6.1,6.2].

According to Shannon's rate-distortion theory, a better performance is always achievable by coding a block of signals instead of coding each signal individually [6.3,6.4]. A vector is a k-dimensional ordered set of real numbers. The components of a vector represent signal samples or numerical values of certain parameters or features that have been extracted from an image. In the most direct application of vector quantization to image compression, a group of contiguous signal samples is blocked into a vector so that

each vector simply describes a small portion of the original image. This leads to efficient exploitation of the correlation between samples within an individual vector. Vector quantization (VQ) is popular in image processing, speech processing and facsimile transmission [6.2,6.5-6.7]. It is capable of producing good-quality reconstructed images. The efficiency of a data compression scheme is measured by its compression ability, the resulting distortion, and the implementational complexity.

Artificial neural network approaches appear to be very promising for intelligent information processing [6.8-6.15] due to their massively paralleled computing structures and the use of learning to adapt the network parameters. Here, a self-organization neural network architecture is used to implement the vector quantizer. An improved self-organization algorithm, which is based on the frequency-sensitive cost function and centroid learning rule, is utilized to construct the VQ codebooks. This algorithm yields near-optimal results with very few iteration paths. A vector quantizer is adaptive if the codebook or the encoding rule is changed in time in order to match the local statistics of the input sequence. The proposed method of using one iteration path can provide a fairly good local vector quantizer with a minimized computational complexity. The training source data can be a subset of an image frame, an individual image frame, or multiple image frames. The proposed adaptive vector quantizer based on the self-organization network is a forward adaptation method [6.16]. Before encoding the different statistics of the source data, the codebook is updated and transmitted to the decoder so that it can correctly reproduce the current vector.

6.2 The Algorithm

6.2.1 Self-Organization Learning

Artificial neural network approaches provide an effective alternative to solving complex information processing problems. The principle of constructing artificial neural networks

comes from the understanding of operation of the neurons in the biological brains. Neurons are placed in an orderly fashion and reflect some physical characteristics of the external stimulus. Although much of the low-level organization in the brain is genetically predetermined, it is likely that some of high-level organization is created during learning which promotes self-organization [6.11]. A self-organization network consists of the input layer and the output layer, which is also called the competitive layer. The basic theory and operation of self-organizing neural networks were described by Grossberg [6.8-6.10], Kohonen [6.11], and other researchers [6.17-6.19]. One major challenge of using the basic self-organization network is that some of the neural units may be under-utilized. Various modifications have been proposed to solve this problem [6.8,6.17,6.20]. The proposed frequency-sensitive self-organization method applies the variable-threshold model from Grossberg [6.8-6.10] to overcome the under-utilization problem.

6.2.2 Frequency-Sensitive Self-Organization (FSO) Method

The neural network architecture based on frequency-sensitive self-organization for image compression is shown in Fig. 6.1. The FSO network consists of two layers: an input layer, and the competitive layer. The input layer serves as a data buffer. It also distributes the input data X_p to the competitive layer. In the competitive layer, each node computes the distortion between its codevector W_i and the input vector. The competitive process is performed throughout the whole layer by the winner-take-all operation. The winning neural unit is determined according to the minimum distortion criterion. The synaptic weights are then updated according to the FSO learning rule.

The FSO method systematically distributes the codevectors in the vector space R^n to approximate the unknown probability density function $P(X)$ of the training vectors. It overcomes the under-utilization problem by including the frequency-sensitive cost function to the learning rule. The neural units are modified with an approximately equal frequency. The 1-path FSO scheme for adaptive vector quantization is described as follows:

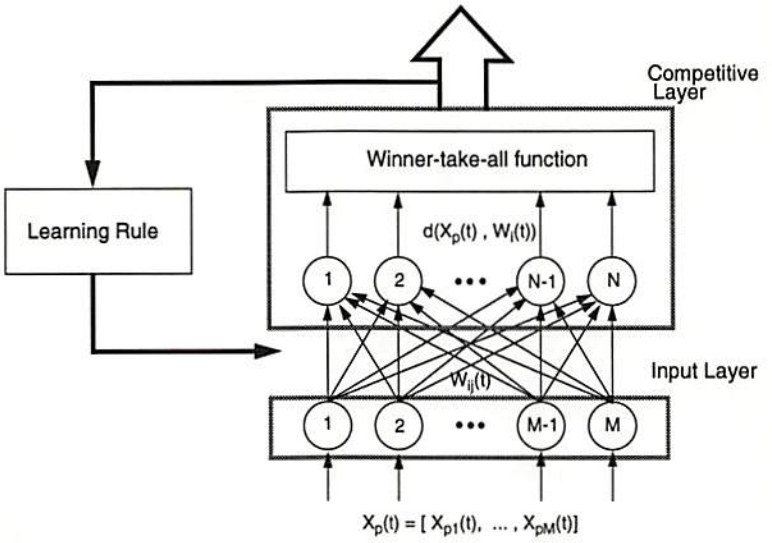


Figure 6.1: The structure of the FSO neural network.

(1) Initialize codevectors (synaptic vectors): $W_i = X_i$ or $Rand(i), i = 1, \ldots, M$ with $Rand$ being a random number generation function and $W_i = [W_{i1}, W_{i2}, \ldots, W_{ik}]$.

(2) For input vector X_p, find the frequency-sensitive distortion $FD_i = d(X_p, W_i)$ for all output neural units:

$$FD_i = (1 + \frac{F_i}{F_{thd}}) \cdot \sum_{j=1}^{k} (X_{pj} - W_{ij})^2, \tag{6.1}$$

where F_i is the frequency count of the codevector W_i and F_{thd} is the frequency threshold.

(3) Select the output unit N_i with the smallest frequency-sensitive distortion and label it as the winner and increase its winning frequency count F_i by one.

(4) Update the winning weight vector $W_{i^*}(t)$ with a frequency-sensitive learning rule:

$$W_{i^*}(t+1) = W_{i^*}(t) + S(t)\,[\,X(t) - W_{i^*}(t)\,], \tag{6.2}$$

where t is the iteration index; and $0 \leq S(t) \leq 1$. Notice that the learning rule moves the winning weight vector toward the input vector by some fractional amount, $S(t) = \frac{1}{F_i}$, which is a centroid learning ratio value.

(5) Repeat steps (2) through (4) for all training vectors.

Use of the frequency-sensitive cost function can avoid under-utilization of some codevectors during the learning process for an inadequately chosen initial codebook. The $\frac{F_i}{F_{thd}}$ term in Eq. (6.1) represents the frequency sensitivity of the cost function. In order to pursue a better performance, the selection of the frequency threshold F_{thd} is very important. If the F_{thd} value is equal to one, the distortion calculation in the 1-path FSO method is similar to that of the frequency-sensitive competitive learning method proposed by Krishnamurthy et al. [6.13]. If the value of F_{thd} is larger than the total number of source vectors, the cost function is not sensitive to the frequency. Figures 6.2 and 6.3 shows the relationship between the reconstructed performance and the frequency threshold F_{thd} for the 512x512-pixel Couple and Creek images, respectively. The good frequency threshold is close to the average training frequency. In an example with 10,402 source vectors and 6-bit codebook, the average training frequency for each codevector will be $\frac{10,402}{64} = 163$. The selection of F_{thd} value is related to the grouping of source vectors. The index histograms of codevectors for the different frequency threshold values are shown in Fig. 6.4, where the initial 6-bit codebook is sampled from the source data. The standard deviations of the codevector index frequencies for the F_{thd} being equal to 1, 163, and the infinity in the Couple image are 89, 152, and 235, respectively. In general, if the value of F_{thd} is smaller than the average training frequency, the index histogram of codevectors is close to the uniform distribution and the codevector grouping is very sensitive to the frequencies of the codevectors. Some dissimilar source vectors might be inappropriately assigned to one codevector. The reconstruction performance is the worst while the F_{thd} value is equal to one. On the other hand, if F_{thd} is much larger than the average training frequency, a diverse distribution will occur and the frequencies of some codevectors could be very small. While the F_{thd} is equal to the infinity, the reconstruction performance is also quite poor. Therefore, the choice of frequency threshold F_{thd} value being the average training frequency can yield a good performance according to the computer analysis.

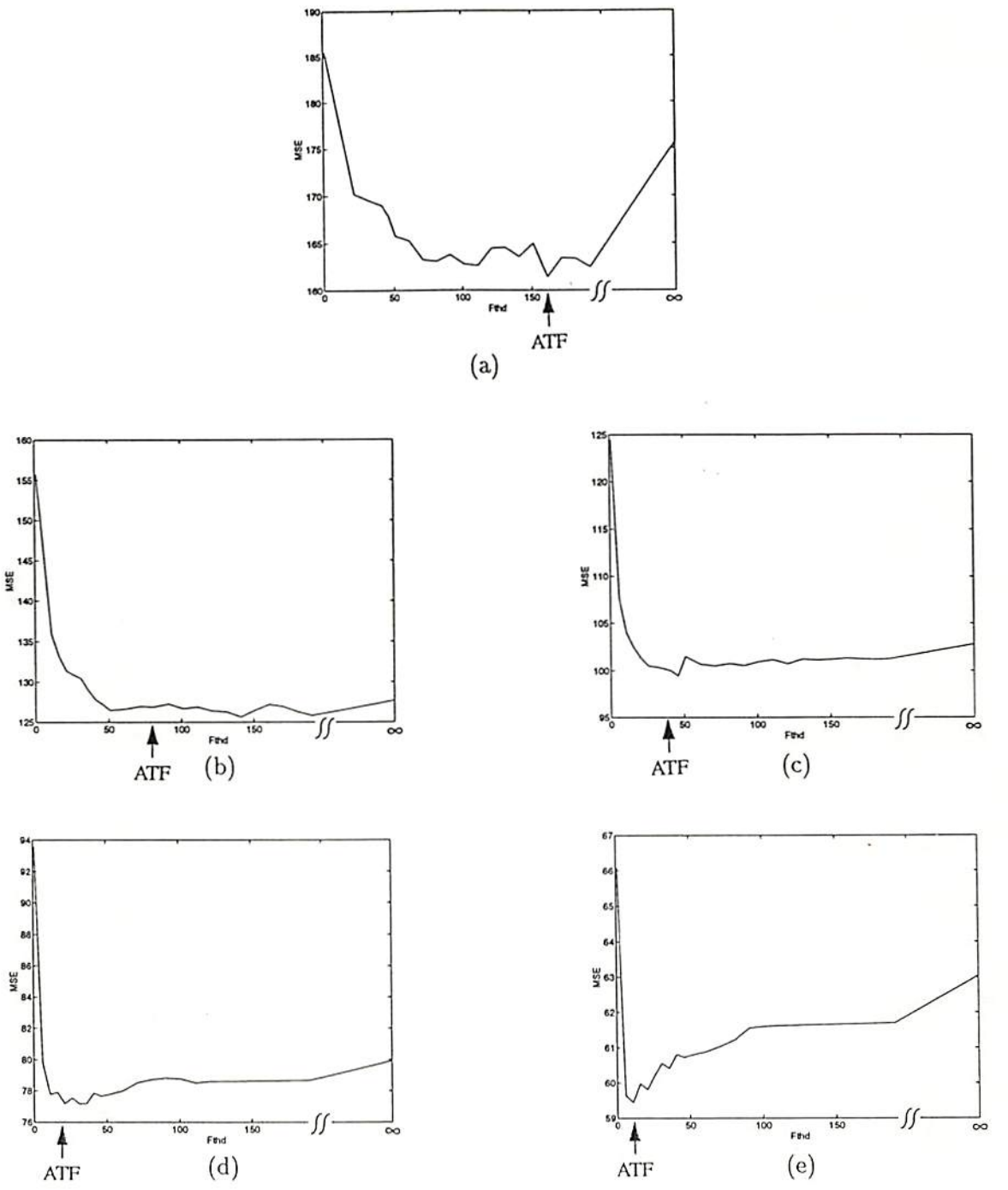


Figure 6.2: Plots of the mean-squared error against the frequency threshold for 512x512-pixel Couple image using the 1-path FSO method on 5x5-pixel subimage blocks. (a) 6-bit codebook; ATF=163. (b) 7-bit codebook; ATF=82. (c) 8-bit codebook; ATF=41. (d) 9-bit codebook; ATF=21. (e) 10-bit codebook; ATF=11. (MSE: Mean-squared error; ATF: Average training frequency)

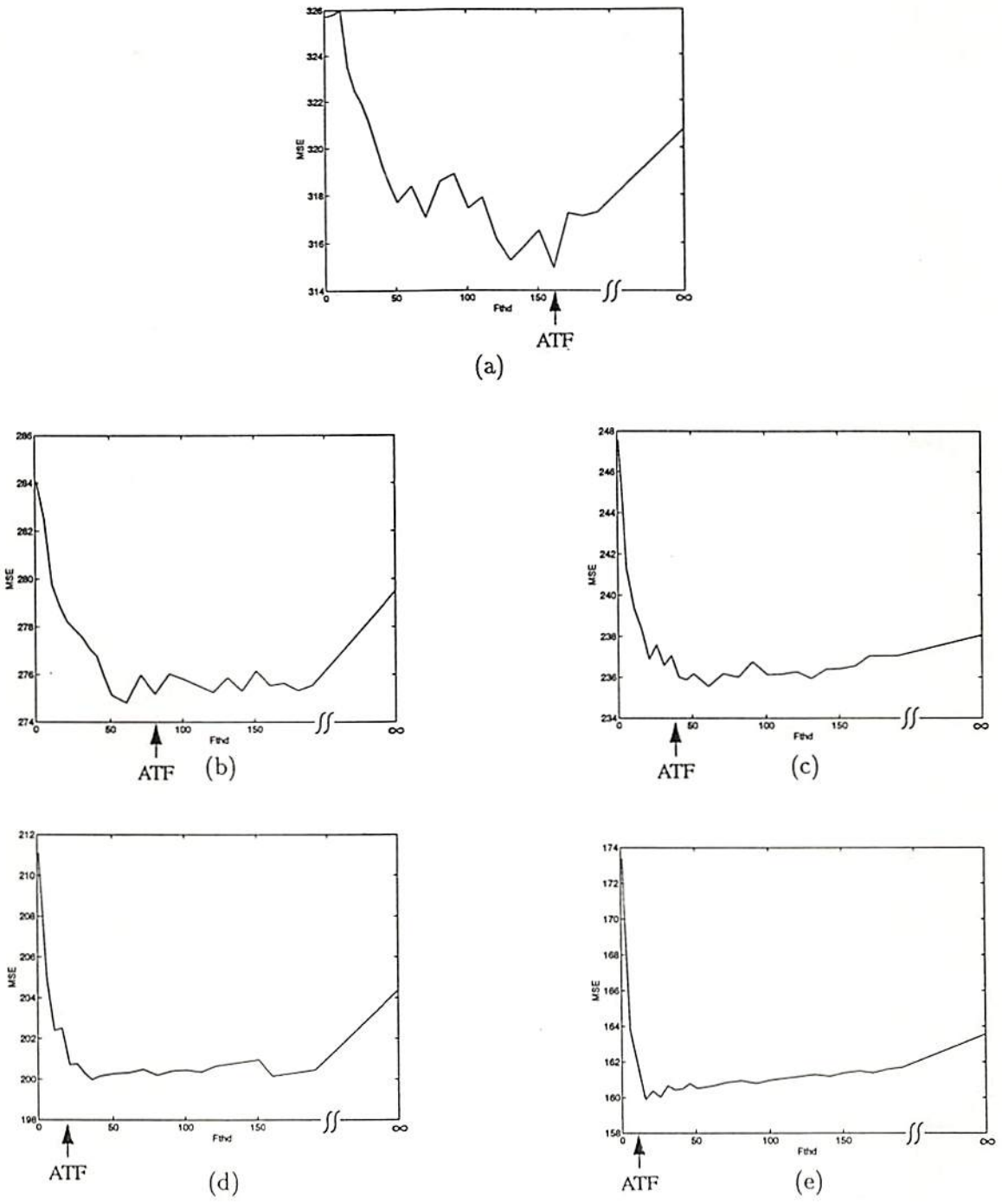


Figure 6.3: Plots of the mean-squared error against the frequency threshold for 512x512-pixel Creek image using the 1-path FSO method on 5x5-pixel subimage blocks. (a) 6-bit codebook; ATF=163. (b) 7-bit codebook; ATF=82. (c) 8-bit codebook; ATF=41. (d) 9-bit codebook; ATF=21. (e) 10-bit codebook; ATF=11. (MSE: Mean-squared error; ATF: Average training frequency)

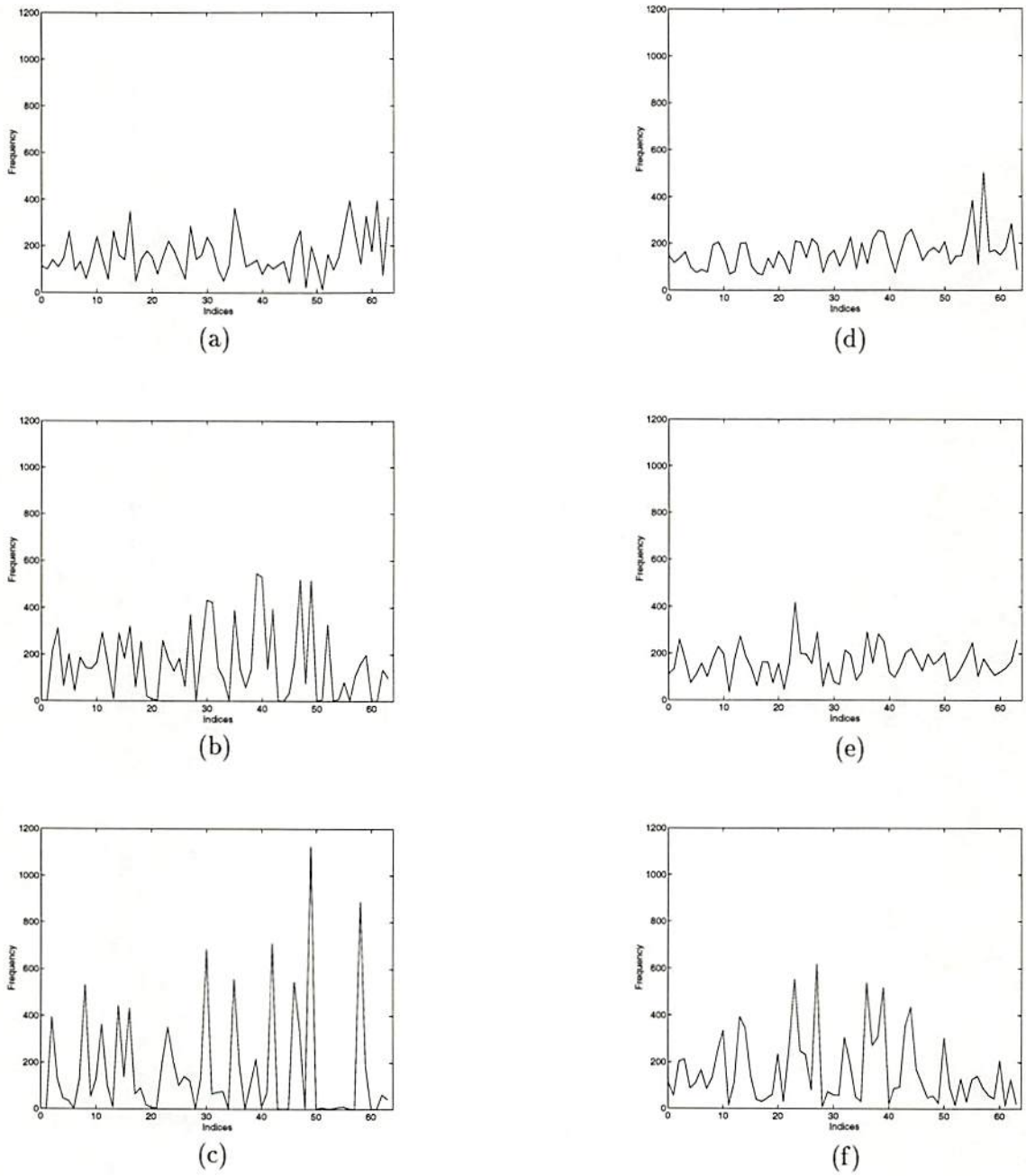


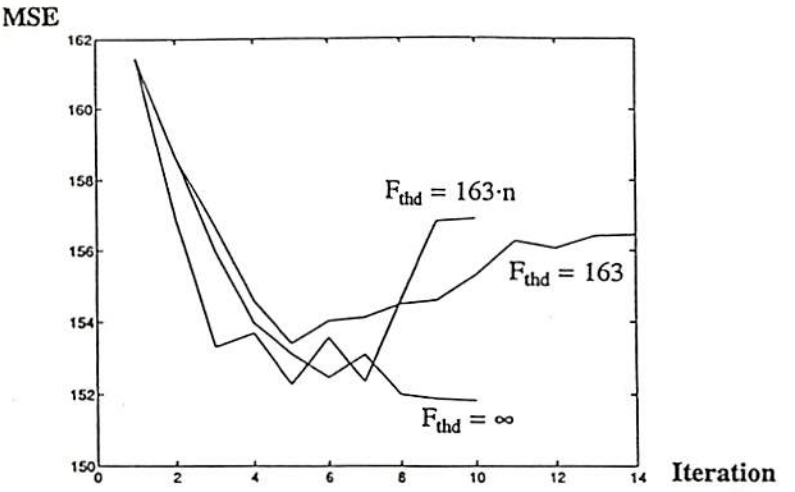
Figure 6.4: Index histograms of 64 codevectors using the 1-path FSO method on 5x5-pixel subimage blocks. (a) 512x512-pixel Couple image with F_{thd} equal to 1; standard deviation = 89. (b) 512x512-pixel Couple image with F_{thd} equal to 163; standard deviation = 152. (c) 512x512-pixel Couple image with F_{thd} equal to ∞; standard deviation = 235. (d) 512x512-pixel Creek image with F_{thd} equal to 1; standard deviation = 75. (e) 512x512-pixel Creek image with F_{thd} equal to 163; standard deviation = 69. (f) 512x512-pixel Creek image with F_{thd} equal to ∞; standard deviation = 147.

In order to further improve the performance, a second path is used to adjust codevectors into better cluster centroids. The codebook produced by the 1-path FSO method is used as a reference codebook. In the Fig. 6.4(b), the frequencies of few codevectors are very small or zero while the average training frequency is used for the frequency threshold. During the second path, the unused or least frequently used codevectors in the 1-path FSO method are deleted. Each highly used codevector is split into the two codevectors. According to the number of the unused or least frequently used codevectors, the corresponding number of the highly used codevectors is used to generate the new codevectors. Here, the splitting scheme in the LBG method [6.6] is used. The highly used codevector W_i is split into $W_i + \delta$ and $W_i - \delta$, where δ is a small constant vector. The training process of the 2-path FSO method is the same as that of the 1-path FSO method.

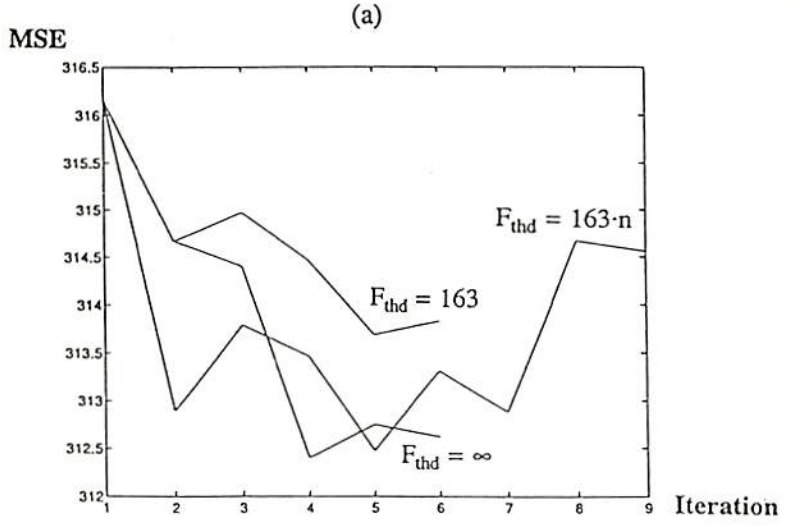
In the LBG method, the initial codebook could come from the splitting algorithm [6.7]. First, these codevectors are used to partition source data into subgroups. Then new codevectors are produced by calculating centroids of these subgroups. The iteration of grouping and calculating centroids in the LBG method is similar to that of incrementally updating the closest codevector for each incoming data through the centroid technique in the FSO method. If the frequency-sensitive term $\frac{F_i}{F_{thd}}$ is not used in the cost function, and the scheme for generating the new codevectors from the highly used codevectors is also not performed, then the result from the iterative FSO method asymptotically approximates that from the LBG method. The learning process in the FSO method is repeated with the same termination criterion in the LBG method. The result of the n-path FSO method appears very close to that of the LBG method.

The frequency-sensitive term is used to avoid under-utilization of some codevectors. After the first and second iterations, a good resultant codebook has been generated. If the frequency-sensitive term is used in the n-path FSO method, it may generate little disturbance in the grouping operation. The total distortion in each iteration is not smoothly decayed. The proposed method may need more iterations to achieve the convergence. Figure 6.5 shows the distortion in each iteration for three different frequency thresholds, 163,

$163 \cdot n$, and infinity, with the same convergence criterion when a 6-bit codebook is used. The best performance can be achieved when the frequency threshold is equal to infinity. It indicates that frequency-sensitive term is not required in the n-path FSO method in order to optimize the computational complexity and reconstruction performance. Therefore, the n-path FSO scheme for vector quantization can be described as follows:



(a)



(b)

Figure 6.5: Comparison for three different frequency thresholds, 163, $163 \cdot n$, and ∞, with the same convergence criterion. (a) 512x512-pixel Couple image using a 6-bit codebook. (b) 512x512-pixel Creek image using a 6-bit codebook.

(1) The codevectors after training in the 2-path FSO method are used as the initial codevectors. The initial value of "n" is set to 3.

Codebook Training:

(2) According to the previous index histogram of codevectors, the unused codevectors or the least frequently used codevectors are deleted. By using the splitting method, the new codevectors are generated from the highly used codevectors.

(3) The frequencies of all codevectors are reset to zero.

(4) For input vector X_p, find the distortion $D_i = d(X_p, W_i)$ for all output neural units:

$$D_i = \sum_{j=1}^{k} (X_{pj} - W_{ij})^2. \tag{6.3}$$

(5) Select the output unit N_i with the smallest distortion and label it as the winner and increase its winning frequency count F_i by one.

(6) Update the winning weight vector $W_{i^*}(t)$ with a frequency-sensitive learning rule:

$$W_{i^*}(t+1) = W_{i^*}(t) + S(t)\,[\,X(t) - W_{i^*}(t)\,], \tag{6.4}$$

where t is the iteration index; and $0 \leq S(t) \leq 1$. Notice that the learning rule moves the winning weight vector toward the input vector by some fractional amount, $S(t) = \frac{1}{F_i}$, which is a centroid learning ratio value.

(7) Repeat steps (4) through (6) for all training vectors.

Estimation of Convergence:

(8) The total distortion TD^n is reset to zero.

(9) For input vector X_p, find the distortion $D_i = d(X_p, W_i)$ for all output neural units.

(10) Select the output unit N_i with the smallest distortion and calculate the total distortion,

$$TD^n = TD^n + D_i\,. \tag{6.5}$$

(11) Repeat steps (9) through (10) for all source vectors.

(12) The convergence of the n-path FSO method is determined by the following equation,

$$\frac{|\,TD^n - TD^{n-1}\,|}{TD^n} \leq \epsilon\,, \tag{6.6}$$

where ϵ is a convergence parameter. If the convergence criterion is matched, the n-path FSO method is finished. Otherwise, "n" is increased by one and steps (2) through (12) are performed again.

Table 6.1 lists the performance of the Couple and Creek images at each path step before reaching the termination criterion of the n-path FSO method. In the 10-bit codebook size, the values of n are 14 and 7 for the Couple and Creek images, respectively. Here, the ϵ is chosen to be 0.0005 for the cases listed in the Table 6.1.

The large dynamic range of images requires that the effective compression algorithm be adaptive to the local image frame statistics. For the vector quantization approach, edge degradation is very severe if no adaptation is allowed for different scene characteristics. In order to simplify computational complexity, the 1-path FSO method can be used as an adaptive method. The adaptivity of the 1-path FSO method is a forward adaptation [6.16] in which the current block information is extracted from the future of the vector sequence. A new codebook can be trained from a next subset of an image frame, an individual image frame, or multiple image frames. The trained codebook completely replaces the old one and is required to be transmitted to the decoder before reproducing the source data with different statistics. While an image sequence is encoded, the codebook can be updated periodically or updated by a criterion which is determined according to the reconstruction performance.

Table 6.1: Mean-squared error between original and reconstructed results for the Couple and Creek images by using the n-path FSO method.

Image / Path number n	Couple					Creek				
	Codebook size					Codebook size				
	10-bit	9-bit	8-bit	7-bit	6-bit	10-bit	9-bit	8-bit	7-bit	6-bit
	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE	MSE
1	59.35	77.11	99.93	126.68	161.44	161.76	200.67	235.97	275.46	316.16
2	51.59	69.75	91.07	119.20	158.64	156.62	196.15	232.81	270.72	314.67
3	48.83	66.29	86.94	115.16	156.00	152.22	192.13	229.78	267.44	314.41
4	47.91	65.27	85.55	114.20	153.99	151.27	191.32	228.88	265.97	312.40
5	47.43	64.77	85.25	113.67	153.13	150.87	191.08	229.13	265.19	312.75
6	47.21	64.68	85.17	113.44	152.47	150.68	190.76	228.38	264.88	312.62
7	47.07	64.57	85.09	113.18	153.11	150.73	190.70	228.33	265.49	
8	47.00	64.41	85.03	113.08	152.00				265.43	
9	46.96	64.35	84.97	113.09	151.89					
10	46.89	64.30	84.95		151.84					
11	46.86	64.31								
12	46.82									
13	46.79									
14	46.79									

6.3 System Simulation

In the computer analysis, the original and reconstructed Couple images using the 1-path, 2-path and n-path FSO methods for the 10-bit codebook are shown in Fig. 6.6. The mean-squared error (MSE) measure is used to evaluate the reconstructed image quality,

$$MSE = \sum_{x=0}^{N_1-1} \sum_{y=0}^{N_2-1} \frac{[\, I(x\ ,\ y) - I'(x\ ,\ y)\,]^2}{N_1\ N_2}\ , \tag{6.7}$$

where I is the original image of size $N_1 \cdot N_2$ and I' is the reconstructed image. Table 6.2 lists the mean-squared error and computational time of images using the 1-path, 2-path

and n-path FSO methods, as well as the LBG method. Similar simulation results for the Creek image are shown in Fig. 6.7 and listed in Table 6.3. In the n-path FSO method, the performance of the Couple and Creek images at each path step is illustrated in the Table 6.1. The results from the n-path FSO method are very close to those from the LBG method with the same convergence criterion. In some selective cases, the performances of the reconstructed images from the n-path FSO method can be better than those from the LBG method. The 7-bit and 8-bit reconstructed Couple images, and the 9-bit and 10-bit reconstructed Creek images are such cases. The reconstructed images using the proposed method on 5x5-pixel subimage blocks are reasonably good.

Table 6.2: Reconstruction performance of image compression using the FSO method and the LBG method on 5x5-pixel subimage blocks of a 512x512-pixel Couple image.

Methods / Codebook Size		FSO										LBG			
	F_{thd}	1-path			2-path			n-path							
		MSE	SNR	CT (min)	MSE	SNR	CT (min)	MSE	SNR	n	CT (min)	MSE	SNR	n	CT (min)
10-bit	11	59.35	24.51	16.9	51.59	25.12	33.8	46.79	25.54	14	482.0	46.68	25.56	13	235.9
9-bit	21	77.11	23.37	8.4	69.75	23.81	16.6	64.31	24.16	11	189.6	63.92	24.19	16	140.3
8-bit	41	99.93	22.25	4.2	91.07	22.65	8.3	84.95	22.95	10	86.2	86.83	22.86	21	92.4
7-bit	82	126.67	21.22	2.1	119.20	21.48	4.3	113.09	21.71	9	39.5	116.27	21.59	27	57.3
6-bit	163	161.44	20.16	1.0	158.64	20.24	2.1	151.84	20.43	10	21.4	150.33	20.48	18	19.5

Note: n: number of iterations; CT: computational time from a SPARCstation-2; min: minutes; SNR: signal-to-noise ratio; MSE: mean-squared error; F_{thd}: frequency threshold.

In order to maintain the fidelity, the codebook is to be updated according to the image frame statistics. If the 10-bit codebook generated from the 1-path FSO method for the Couple image is used to encode and decode the Creek image without any modification, the mean-squared error is up to 258, as shown in Fig. 6.8. The bridge in this reconstructed Creek image is quite blurred. After adapting this poor codebook by using the 1-path FSO method, a better reconstructed image as shown in Fig. 6.7(b) can be achieved with the mean-squared error of 162. This result illustrates that the codebook needs to be adjusted

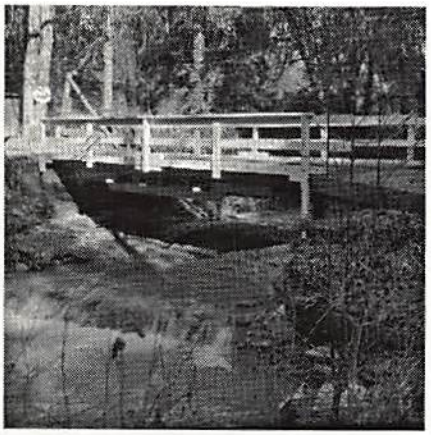
(a)

(b)

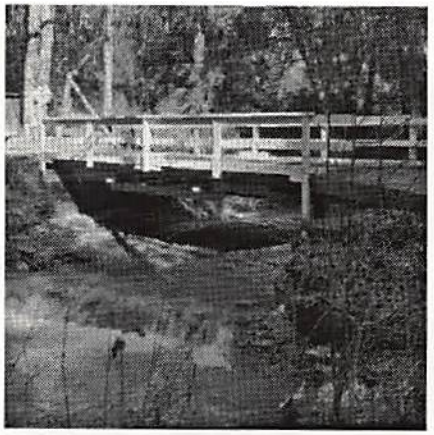
(c)

(d)

Figure 6.6: Image compression using the FSO method on 5x5-pixel subimage blocks. (a) Original Couple image; 512x512 pixels. (b) Reconstructed image using 10-bit 1-path FSO codebook; MSE=59.35. (c) Reconstructed image using 10-bit 2-path FSO codebook; MSE=51.89. (d) Reconstructed image using 10-bit n-path FSO codebook; MSE=46.79.



(a)



(b)

(c)

(d)

Figure 6.7: Image compression using the FSO method on 5x5-pixel subimage blocks. (a) Original Creek image; 512x512 pixels. (b) Reconstructed image using 10-bit 1-path FSO codebook; MSE=161.76. (c) Reconstructed image using 10-bit 2-path FSO codebook; MSE=156.62. (d) Reconstructed image using 10-bit n-path FSO codebook; MSE=150.73.

Table 6.3: Reconstruction performance of image compression using the FSO method and the LBG method on 5x5-pixel subimage blocks of a 512x512-pixel Creek image.

Methods / Codebook Size	F_{thd}	FSO										LBG			
		1-path			2-path			n-path							
		MSE	SNR	CT (min)	MSE	SNR	CT (min)	MSE	SNR	n	CT (min)	MSE	SNR	n	CT (min)
10-bit	11	161.76	19.96	16.8	156.62	20.10	33.5	150.73	20.26	7	250.8	159.67	20.00	13	231.8
9-bit	21	200.67	19.02	8.3	196.15	19.12	16.6	190.70	19.24	7	124.1	191.08	19.21	17	148.3
8-bit	41	235.97	18.32	4.2	232.81	18.38	8.3	228.33	18.46	7	61.4	222.23	18.56	20	86.7
7-bit	82	275.46	17.65	2.1	270.72	17.72	4.2	265.44	17.81	8	34.6	256.22	17.94	22	48.4
6-bit	163	316.16	17.05	1.0	314.67	17.07	2.1	312.62	17.10	6	13.3	296.28	17.31	19	21.0

Note: n: number of iterations; CT: computational time from a SPARCstation-2; min: minutes; SNR: signal-to-noise ratio; MSE: mean-squared error; F_{thd}: frequency threshold.

according to the statistic change of the source data. In the 1-path FSO method, the codebook is trained by the source data with one iterative path. Each source vector is used to update the corresponding codevector only one time. In the LBG method, a lot of iterative paths for codebook training are required. According to the results of Tables 6.2 and 6.3, the computational complexity of the 1-path FSO method is close to that of one iterative path in the LBG method. Therefore, the proposed 1-path FSO method can achieve a very good adaptivity.

References

[6.1] A. Netravali, F. Mounts, "Ordering techniques for facsimile coding: a review," *Proc. IEEE,* vol. 68, pp. 796-807, July 1980.

[6.2] N. M. Nasrabadi, R. A. King, "Image coding using vector quantization: a review," *IEEE Trans. on Communications,* vol. 36, no. 8, pp. 957-971, Aug. 1988.

[6.3] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. Journal,* vol. 27, pp. 379-423, and 623-656, 1948.

[6.4] R. M. Gray, *Source Coding Theory,* Kluwer Academic Publishers: Boston, MA, 1990.

[6.5] A. Gersho, "On the structure of vector quantizers," *IEEE Trans. on Inform. Theory,* vol. 28, no. 2, pp. 157-162, March 1982.

Figure 6.8: Reconstructed Creek image using a 10-bit FSO codebook from Couple image; MSE=258.

[6.6] R. M. Gray, "Vector quantization," *IEEE ASSP Magazine,* pp. 4-29, April 1984.

[6.7] Y. Linde, A. Buzo, R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. on Communication,* vol. COM-28, no. 1, pp. 84-95, Jan. 1980.

[6.8] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Sci.,* vol. 11, pp. 23-63, 1987.

[6.9] S. Grossberg, "Adaptive pattern classification and universal recording: I. Parallel development and coding of neural feature detectors," *Biological Cybernetics,* 23, pp. 121-134, 1976.

[6.10] S. Grossberg, "Adaptive pattern classification and universal recording: II. Feedback, expectation, olfaction, illusions," *Biological Cybernetics,* 23, pp. 187-202, 1976.

[6.11] T. Kohonen, *Self-Organization and Associative Memory,* 2nd Ed., Springer-Verlag: New York, NY, 1988.

[6.12] N. M. Nasrabadi, Y. Feng, "Vector quantization of images based upon the Kohonen self-organizing feature maps," *Proc. 1988 Int. Joint Conf. on Neural Networks,* vol. I, pp. 101-108, San Diego, CA, June 1988.

[6.13] A. K. Krishnamurthy, S. C. Ahalt, D. E. Melton, P. Chen, "Neural networks for vector quantization of speech and images," *IEEE Journal on Select Areas in Communication,* vol. 8, no. 8, pp 1449-1457, Oct. 1990.

[6.14] C. Lu, Y. Shin, "A neural network based image compression system," *IEEE Trans. on Consumer Electronics,* vol. 38, no. 1, pp. 25-29, Feb. 1992.

[6.15] O. T.-C. Chen, B. J. Sheu, W.-C. Fang, "Adaptive vector quantizer for image compression using self-organization approach," *IEEE Inter. Conf. on Acoustics, Speech and Signal Processing,* vol. 2, pp. 385-388, Mar. 1992.

[6.16] A. Gersho, R. M. Gray, *Vector Quantization and Signal Compression,* Kluwer Academic Publishers: Boston, MA 1992.

[6.17] R. Hecht-Nielsen, "Application of counterpropagation networks," *Neural Networks, 1(2),* pp. 131-141, 1988.

[6.18] B. Kosko, "Stochastic competitive learning," *Proc. 1990 International Joint Conference on Neural Networks,* vol. II, pp. 215-226, San Diego, CA, June 1990.

[6.19] D. Rumelhart, D. Zipser, "Feature discovery by competitive learning," *Cognitive Sci.,* vol. 9, pp. 75-112, 1985.

[6.20] D. DeSieno, "Adding a conscience to competitive learning," *Proc. of IEEE International Joint Conference on Neural Networks,* vol. I, pp. 117-124, San Diego, CA, June 1988.

Chapter 7

Impact of Dissertation Work

Algorithms based on neural network paradigms have been demonstrated to be useful in optimization and classification tasks. Different learning methods such as the back-propagation learning method, Gauss-Newton method, and Quasi-Newton method can be thought of as quadratic optimization techniques. A three-layered neural network without time-delayed elements can successfully learn space trajectories without crossovers. The proposed work provides the concept for using a simple neural network to learn the space trajectories which were investigated by the other researchers using time-delayed neural networks. It indicates that some applications with short memory source for information processing can be efficiently implemented by a compact neural network.

The frequency-sensitive self-organization (FSO) method for image compression can achieve high adaptivity, good reconstruction performance and a fairly good compression ratio. Since the proposed FSO method is based on the vector quantization scheme, the other applications for adaptive classification or identification can also be implemented by the FSO method. In Appendix A, a new adaptive vector quantization scheme based on the Gold-Washing method are also presented. These two compression methods can be combined with the industrial standards for high-performance compression. The next significant improvement of existing compression algorithms for images, video, speech and audio can be achieved by using perceptual modeling. In addition to the source coding, the complexity and reliability of channel coding must be considered together.

In order to effectively address complex real-world problems, the neural networks must be scaled up, or modularized, and then must be efficiently implemented in hardware. In general, a neural network module consists of a large collection of simple processing elements. These simple processing elements execute mathematical algorithms to collectively carry out information processing through their responses to stimuli. There are technological constraints to the scale size and capacity of neural network hardware. In contrast, biological networks which incorporate features of real neurons and the connectivity of real neural networks have been shown to exhibit theoretical advantages in dimensional scaling and processing time.

Hippocampus is a very good research topic for understanding the design of high-performance intelligent machines. It plays an important role in the functions of learning and memory. However, hippocampus remains to be clarified: how the memory system is performed and how the hippocampal circuitry is constructed. In hardware implementation, a mixed-signal design technique is one of the good choices. The local data computation is executed by analog circuitry to achieve full parallelism and to minimize power consumption. Interneuron communication is carried out in the digital format to achieve network scalability by using an array of neural chips. The proposed VLSI implementation of a non-linear model of the hippocampus will not only facilitate for efficient emulation of biological systems or simulation of neural network paradigms but also produce useful knowledge and results for scientific and engineering applications.

In future research, further investigation and understanding of brain functions will be conducted. A new massively parallel computing system from the study of biological neural behavior and topology can be created for the intelligent living machines. In addition to investigating visual and auditory perception, the sensors for measuring humidity, atmospheric pressure, temperature, tastes and smells can be explored for the multi-channel computers. New hardware design techniques also need to be developed further to meet high computation power in the large-scale real-world applications. Optical computing

and interconnection through optical fiber or optoelectronic components provides another scheme for intelligent system design.

Appendix A

An Adaptive Vector Quantizer Based on the Gold-Washing Method

The VLSI architecture for an adaptive vector quantization is presented. The adaptive vector quantization method does not require *a-priori* knowledge of the source statistics and the pre-trained codebook. The codebook is generated on the fly and is constantly updated to capture local textual features of data. The source data are directly compressed without requiring the generation of codebook in a separate pass. The adaptive method is based on backward adaption without any side information. The speed of data compression by using the proposed adaptive method is much faster than those by using the conventional vector quantization methods. The algorithm is shown to reach rate distortion function for memoryless sources. In image processing, most smooth regions are matched by the codevectors and most edge data are preserved by using the block-data interpolation scheme. The VLSI architecture consists of two move-to-front vector quantizers and an index generator. It explores parallelism in the direction of codebook size and pipelining in the direction of vector dimension. According to the circuit simulations using the popular SPICE program, the computation power of the move-to-front vector quantizer can reach 40 billion operations per second at a system clock 100 MHz by using a 0.8 μm CMOS technology. It can provide a computing capability of 50M pixels per second for high-speed image compression. The proposed algorithm and architecture can lead to the development of a high-speed image compressor with great local adaptivity, minimized complexity, and fairly good compression ratio.

A.1 Introduction

Data compression is essential in many applications such as digital laser-discs, electronic cameras, video-phones, video-conferencing systems, image and interactive video tools on personal computers and workstations, and high-definition televisions. Efficient compression of information data would significantly decrease both the communication and archival costs [A.1,A.2]. A fundamental result of Shannon's rate-distortion theory [A.3] states that better performance can be achieved by coding vectors instead of scalars. Vector quantization [A.1] is a process in which data are divided into small vectors, which are then individually encoded in sequence. The objective is to identify a set of possible vectors which are representative of the information to be encoded. For each source vector, the vector quantization encoder selects the closest matching vector from the codebook.

Vector quantization (VQ) is a very effective technique for speech waveform coding and image data compression [A.4-A.8]. An optimal codebook should completely reflect the statistics of the input vectors. A codebook generated by using the LBG algorithm [A.5] is locally optimized for the particular training data. Generally there are two types of codebook generation methods: the universal method and the adaptive method. In an universal method, the codebook generated from a large set of images is fixed at both the transmitter and the receiver. Therefore, there is no overhead required for transmitting the codebook. However, new images which are not included in the training sequence may not be well represented, and a large codebook is needed in order to maintain a good fidelity for the various kinds of data. In an adaptive method for image compression, Gersho et al. [A.9] proposed a progressive codevector replacement method. It adaptively replenishes codevectors in order to keep the partial distortion constant while the source statistics change. In Goldberg's method [A.10], a new codebook is generated for the individual images or subparts of the image. The advantage of this adaptive method is that the codebook represents the individual image better, and a smaller codebook can reach a distortion requirement. On the other hand, a larger codebook is used in Zeger's method [A.11]. A new partial VQ codebook is periodically generated from the source data and

then replaces the nearest codevectors from the currently used codebook. In our proposed adaptive method, the codebook is generated on the fly as the data flow in. There is no need to have the codebook training in a separate pass. The codebook is constantly updated during the encoding and decoding operations in order to capture the local textual features of the data. This adaptive method has several advantages. First, the separate codebook transmission is not required. Second, a smaller codebook can meet the distortion criterion. Finally, the computational cost for generating new codevectors is also very low.

Usually, the reconstructed images from the vector quantization approach reveal several types of annoying distortion such as sawtooth edges, visible blocking structure, smeared out details and contouring in the smooth area. In the proposed adaptive vector quantization (AVQ), these drawbacks are reduced by using an interpolation method which is called the block-data interpolation method. The detailed explanation of the block-data interpolation method is in Section A.2.2. If a source vector cannot be matched by the codevectors according to the performance requirement, a new codevector is generated by using the block-data interpolation method. The computational complexity of the block-data interpolation method is very low, as compared with those of the conventional VQ methods for generating new codevectors. Therefore, the compression speed for the AVQ method is very fast.

Traditionally, the search operation is executed sequentially. Each source vector is compared with a codevector at a time. For k input vectors of dimension n and a codebook of size 2^m, the computation time is $O(kn2^m)$. This step requires a huge computational power. If parallelism in the direction of the codebook size and pipelining in the direction of vector dimension are explored, all codevectors can be evaluated in parallel for a given source vector. The computation time is reduced to $O(kn)$. In the proposed VLSI architecture, key functional modules include an array controller, two move-to-front vector quantizers (MFVQs), and an index generator. The source data are pipelined into the two vector quantizers. The index generator calculates the index value from the output results of the two MFVQs. If there exists a codevector which can meet the distortion requirement,

the Huffman code for the index value of the codevector and an identification code form the compressed data. Otherwise, the host machine uses the new codevector generated by the block-data interpolation technique to represent the source data, and then update the codebook data in the MFVQ.

A.2 The Algorithm

The proposed vector-quantization-based lossy data compression was briefly described in [A.12]. The AVQ method has a feature derived from lossless move-to-front algorithms [A.14,A.15]. If the current source word is used as the new codevector to update the codebook, the rate distortion function can not be reached. Ornstein and Shields [A.16] used a greedy algorithm to form the codebook. They demonstrated that their method can reach the rate distortion function in the almost-sure sense. However, a fixed codebook was used. When the statistics of the source data changes from time to time, the performance of their algorithm would degrade. Zhang and Wei [A.13] have developed the mathematical foundation of the "Gold-Washing" technique, and show that for memoryless sources and Markov sources, the rate distortion function can be achieved.

Since the source statistics are unknown, the optimal codevector distribution is not available to the encoder. Therefore the key issue in the design of this type of adaptive lossy data compression algorithms is how to choose the new codevectors. The basic idea for the Gold-Washing technique is to use a finite buffer, and to keep on replacing unused or infrequently used codevectors. Notice that there is a survivability associated with each codevector. A properly generated codevector has a much better chance to survive than an inadequately created codevector. The Gold-Washing technique uses this property to retain useful codevectors in the codebook and to discard useless codevectors.

A.2.1 Gold-Washing Method

There are two buffers (buffer-1 and buffer-2) in the proposed codebook as shown in Fig. A.1. In the buffer-2, a frequency table for each codevector is used. For each incoming data

$\mathbf{x}_i$, the best matched codevector is searched by calculating distortion measure $\mu(\mathbf{x}_i, \mathbf{y}_j)$. If the distortion measure is less than a threshold, then the match has been achieved. Here

$$\mu(\mathbf{x}_i, \mathbf{y}_i) = \sum_{k=1}^{n} (x_{ik} - y_{ik})^2 . \tag{A.1}$$

When a best-matched codevector $\mathbf{y}_j$ exists, two possible cases are considered:

(1) If codevector $\mathbf{y}_j$ is in buffer-1, move it to the top of buffer-1 and push down the first j-1 entries of buffer-1 by one notch.

(2) Otherwise, codevector $\mathbf{y}_j$ is in buffer-2 and the corresponding frequency counter entry is to be increased by 1. When the match is not achieved, a new codevector is created by using the block-data interpolation method. This new codevector is put on the top of buffer-2 and its initial frequency counter is set to zero. All previous entries of buffer-2 are pushed down by one notch. If the content of the frequency counter for the codevector in the bottom of buffer-2 is below a frequency threshold value T_f, this entry is deleted. Otherwise, this entry is copied to the top of buffer-1 and the whole entries of buffer-1 are pushed down by one notch. In such a case, the previous last entry in buffer-1 is discarded.

In image processing, buffer-1 functions as a frame adaptive buffer and buffer-2 functions as a block adaptive buffer. The sizes of buffer-1 and buffer-2 can be adjusted according to data statistics. For example, if buffer-1 is not used completely after encoding one frame image data, it means that either the size of buffer-1 is too big or the size of buffer-2 is too small, which can be caused by an inadequate frequency threshold value.

A.2.2 Block-data Interpolation

Selective data from the block source information are quantized. The quantized data are used to find the neighboring data values by the interpolation method. The coefficient codes for the neighboring data are also constructed. The quantized level and jumping step for the coefficient table are selected according to performance requirements. For example, 4 bits are used to quantize image pixel data such that the maximum distortion for each pixel is 8 gray-levels. Based on these quantized data, linear interpolation is used to estimate other pixel results. Here, the jumping step is chosen to be 32. If the difference between the

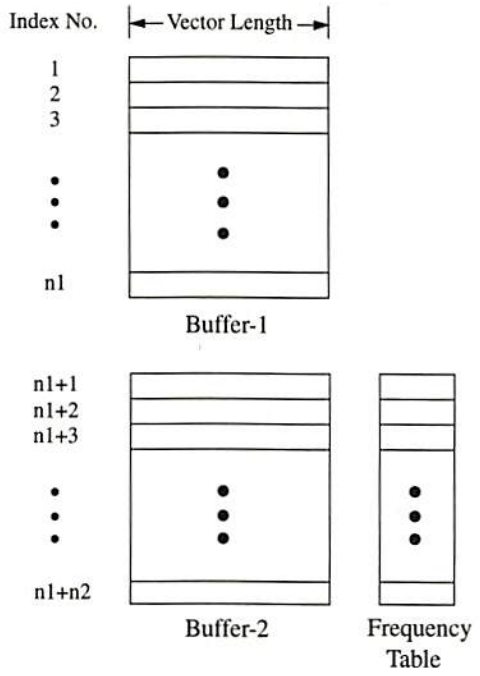


Figure A.1: Buffer configurations in Gold-Washing algorithm.

original pixel value and interpolated value is within (-16,16), then "0" is used to encode this pixel and the linearly interpolated value represents this pixel. If the difference is larger than 16, then "10" code is used, and the interpolated value is increased by 32 to represent the pixel. Otherwise, it is encoded by "11" and the interpolated value is decremented by 32 to represent the pixel. Without a large gray-level difference among neighboring pixels, the maximum distortion can be limited to 15 gray-levels for each pixel.

If the 4x4 subimage block is used, the 4 corner pixels are quantized and the others are estimated by interpolation using these 4 quantized values. After quantization and interpolation, the estimated subimage block is

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} . \tag{A.2}$$

Here, the a_{00}, a_{03}, a_{30}, and a_{33} are individually quantized from the original gray-level values of the 4 corner pixels. The linearly interpolated formula for the other pixels is described as follows,

$$a_{ij} = b_j a_{i0} + b_{3-j} a_{i3} + b_i a_{0j} + b_{3-i} a_{3j} , \qquad \text{(A.3)}$$

$$for\ (i,j) \in \{ (0,1), (0,2), (1,0), (2,0), (1,3), (2,3), (3,1), (3,2) \} ,$$

where $b_0 = 0$, $b_1 = \frac{2}{3}$, $b_2 = \frac{1}{3}$, and $b_3 = 0$. According to the values of a_{01}, a_{02}, a_{10}, a_{20}, a_{13}, a_{23}, a_{31}, and a_{32}, the values of a_{11}, a_{21}, a_{12}, and a_{22} are estimated by the following formula,

$$a_{ij} = \tilde{b}_i a_{0j} + \tilde{b}_{3-i} a_{3j} + \tilde{b}_j a_{i0} + \tilde{b}_{3-j} a_{i3} , \quad for\ 1 \le i,j \le 2 , \qquad \text{(A.4)}$$

where $\tilde{b}_1 = \frac{1}{3}$ and $\tilde{b}_2 = \frac{1}{6}$. The difference between the original and interpolated values can form a coefficient table. Figure A.2 illustrates one representative example. If a 5x5 subimage block is used, the center pixel of the block can be used as a sampled datum as well as the four corner pixels. In other words, if a larger block size is used, more sampled data are required in order to maintain a good quality for reconstruction.

The selection of a new codevector is very important. If the source data are used as a new codevector, the rate distortion function cannot be achieved, and the compression ratio is decreased due to transmitting or storing the entire source data. Therefore, the block-data interpolation method tries to introduce a small change to the source data in order that the new codevector can represent a possible grouping of source data. This process can increase the possibility of matching codevectors. The code for source data is also reduced by using the block-data interpolation method.

As compared with the bits of codevector index, more bits are used to encode a subimage block by applying the block-data interpolation method. The high-entropy data can be preserved very well. According to the Gold-Washing method, most codevectors in buffer-1 and buffer-2 represent the smooth regions. The complex edge data are usually encoded by using the block-data interpolation method so that the edges can be reconstructed very clearly.

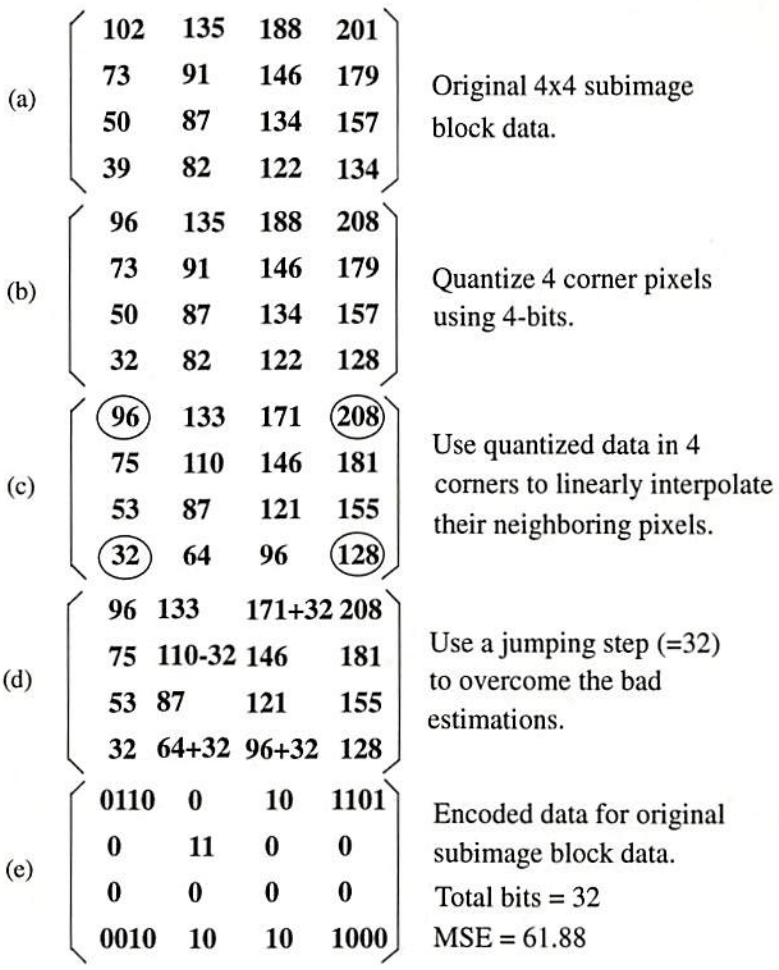


Figure A.2: One example of block-data interpolation.

A.2.3 Adaptive Vector Quantization

The procedure for encoding is:

1. Create the initial codebook from a random generator or the previous codebook data.

2. Encode the incoming data $\mathbf{x}_i$ by codevector matching. If a best-matched codevector $\mathbf{y}_j$ exists such that $\mu(\mathbf{x}_i, \mathbf{y}_j)$ is smaller than the threshold, then an identification bit of type 1 and the Huffman code for the j index are used to encode $\mathbf{x}_i$. The codebook is updated by the Gold-Washing method.

3. If no codevectors can meet the minimum distortion requirement, the block-data interpolation is used to generate a new codevector. This new codevector is added to the codebook according to the Gold-Washing method. An identification bit of type 2, quantized sample data, and coefficient table are used to encode this $\mathbf{x}_i$.

In the decoding process, the following procedure is used. Upon receiving data, the decoder checks the identification bit first. If it is of type 1, the received code reconstructs the codevector index by using the inverse Huffman table. The codevector index is used to retrieve the data by using table lookup and to update the codebook according to the Gold-Washing method. Otherwise, the data is produced from quantized sample data and a coefficient table. The codebook also needs to be updated by using the Gold-Washing method.

A variable-length coding scheme can be implemented for the indices of codevectors in buffer-1 since the codebook updating of buffer-1 is based on the move-to-front feature. The frequently used codevectors are usually in the front entries of buffer-1. Here, the dynamic Huffman coding scheme [A.17] is used to reduce the entropy of indices due to the unknown source statistics. On the other hand, the codevectors of buffer-2 have similar probabilities of being matched because the frequently used codevectors are moved to buffer-1 and the infrequently used codevectors are discarded. The indices of codevectors in buffer-2 can use a fixed length code.

An efficient matching strategy has been developed. If more than one entry can make the μ values smaller than the threshold, then $\mathbf{x}_i$ can be encoded by either the first-matched entry or the entry with the smallest distortion. The smallest-distortion scheme is also called the best-matching scheme. Table A.1 illustrates the performance comparison between the best-matching scheme and the first-matching scheme. Due to Huffman codes for the indices of buffer-1, the first-matching scheme can yield a higher compression ratio for a given distortion threshold but the quality of the reconstructed image is worse than that by using the best-matching scheme. By comparing the similar reconstruction performance in the two schemes, the compression ratio using best-matching scheme is higher than

the other one. In the AVQ method, the new codevector generated from the block-data interpolation method can be quite different from the other codevectors. Therefore, no similar codevectors appear in the codebook at the same time so that the best-matching yields the best performance.

Table A.1: Mean-squared errors and compression ratios versus distortion thresholds for Girl and Pepper images by using the best-matching and first-matching schemes.

Images	Girl				Pepper			
Performance / Distortion Thrd.	Best-Matching		First-Matching		Best-Matching		First-Matching	
	MSE	CR	MSE	CR	MSE	CR	MSE	CR
800	50.61	7.57	53.59	7.62	61.89	8.41	66.00	8.66
1000	48.95	9.13	53.78	9.32	62.92	8.99	69.26	9.40
1200	49.44	10.24	56.46	10.72	66.39	9.45	72.50	9.95
1400	52.65	10.98	61.80	11.70	69.13	9.99	76.56	10.50
1600	56.22	11.60	67.08	12.40	71.09	10.53	79.53	11.07
1800	61.87	11.95	73.16	12.79	73.95	10.81	84.84	11.44
2000	66.25	12.56	80.12	13.48	78.11	11.18	93.24	11.85

Note: The same distortion threshold is used in buffer-1 and buffer-2.
MSE : mean-squared error, CR : compression ratio.

In the Gold-Washing method, the values for the codebook size (buffer-1 and buffer-2), the distortion threshold, and the frequency threshold should be carefully chosen. Initially, a typical codevector size of a 4x4 window can be selected. If a larger codevector size is used, then the codebook sizes for buffer-1 and buffer-2 are increased in order to maintain a good performance. According to the same reconstruction quality for the AVQ and the LBG methods, the codebook size for the LBG method can be referred as the codebook size of buffer-1. The codebook size of buffer-2 is larger than or equal to that of buffer-1. The choice for the distortion threshold can affect the survival probability of codevectors.

In some cases, the compression ratio can be increased by choosing a larger distortion threshold. Relationship between the compression ratio and distortion threshold is also illustrated in Table A.1. Usually, the distortion threshold is determined by the required quality of the reconstructed image. The frequency threshold is chosen to make a good use of the codebook size. If swapping of codevectors between buffer-1 and buffer-2 occurs too frequently, the frequency threshold is too small and should be increased. In the codebook design, buffer-1 functions as a frame adaptive buffer and buffer-2 functions as a block adaptive buffer. An efficient codebook implementation for encoding an image frame can be described as that the number of codevectors swapping between buffer-1 and buffer-2 is close to the codebook size of buffer-1. If block-data interpolation occurs frequently, then it could be the main cause of an inefficient encoder operation. In such a situation, it is better to increase the codebook size. The quantized level and jumping step for the block-data interpolation can be varied in order to achieve a good estimation of the source data. Therefore, values of the codebook size, the distortion threshold, frequency threshold, quantized level and jumping step should be judiciously chosen in order to ensure that block-data interpolation occurs infrequently, and a high performance is maintained.

In order to increase compression efficiency, the source data can be classified or transformed first [A.18], and then each classified data can be compressed by using the AVQ method individually. In video processing [A.19], intraframe compression and interframe compression can be performed. Since the most bits are used for the intraframe compression, the proposed adaptive vector quantization is used to enhance the performance of intraframe (I-frame) compression. According to the MPEG standard [A.20], the motion-detection scheme is used for the predicted frame (P-frame) and bi-directionally predicted frame (B-frame). In the P-frame and B-frame, the reconstruction performance can be improved by combining the motion-detection scheme with the AVQ method. Each 8x8 subimage block in the P-frame or B-frame is used to find the best matched data from the I-planes or the interpolated data from these best matched data. If the best matched data or interpolated data cannot meet performance requirement, this 8x8 subimage block is

partitioned into four 4x4 subimage blocks. Each 4x4 subimage block can be compressed by using the AVQ method in order to meet the reconstruction performance. Therefore, the AVQ method can be suitably used for video applications.

A.3 Computer Analysis Results

In our computer analysis experiments, the sum of the buffer-1 size and the buffer-2 size is set to 512. The individual sizes of buffer-1 and buffer-2 are selected according to image statistics. The frequency threshold is set to 2 for a moderate buffer-2 size. It makes the codevectors in buffer-2 easy to survive and migrate to buffer-1. The initial codebook is created from a random number generator. The distortion threshold is chosen such that the performance of reconstruction can be similar to the results from the LBG method [A.4] for an 8-bit codebook. Here, the distortion threshold of buffer-1 is the same as that of buffer-2. The simulation results are listed in Table A.1. The original Girl image and its reconstructed images using the AVQ and the LBG methods are shown in Fig. A.3. In the LBG method, the original image is used to generate the codebook, and then this codebook is used to encode and decode the original image. Figure A.4 shows the difference images between the original Girl image and its reconstructed images for the AVQ and the LBG methods. The error contour of the difference image using the LBG method is clearer than that by using the AVQ method. In other words, edge information is well preserved by using the proposed AVQ method since most edges are estimated by the block-data interpolation method. Similar simulation results for the Pepper image are shown in Figures A.5 and A.6. If similar source vectors occur in sequence, the encoded data for each source vector can be the same. The redundancy of encoded data by using the AVQ method can be further compressed by an entropy coding. The UNIX command "compress" (LZW) [A.21,A.22] was performed. The mean-squared errors and compression ratios for the images in Figures A.3 and A.5 are listed in Table A.2.

In order to reduce the block-data interpolation frequency and maintain a good image fidelity, the distortion threshold of buffer-2 can be assigned to the average distortion value

(a)

(b)

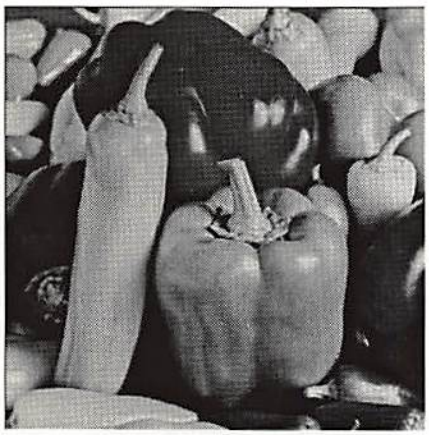
(c)

Figure A.3: Image compression on the 4×4 subimage block. (a) Original Girl image; 512×512 pixels. (b) Reproduced image using the LBG method. (c) Reproduced image using the AVQ method.

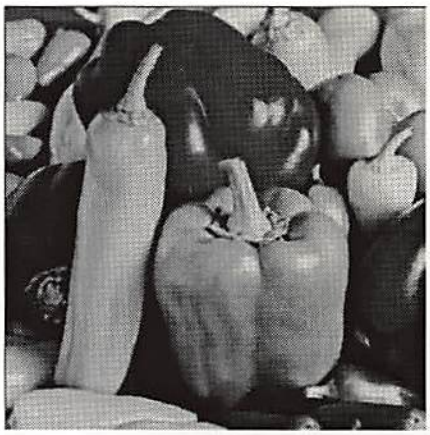
(a)

(b)

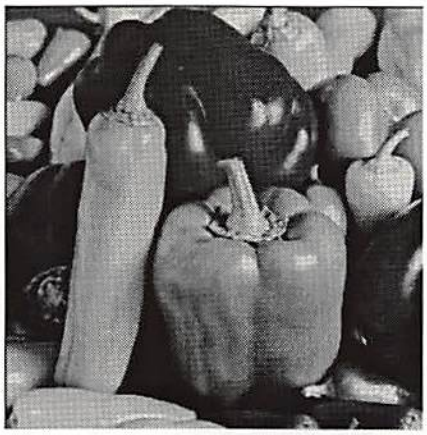
Figure A.4: Difference image between the original Girl image and the reconstructed image. (a) Difference image with inverse illuminance for the LBG method. (b) Difference image with inverse illuminance for the AVQ method.



(a)



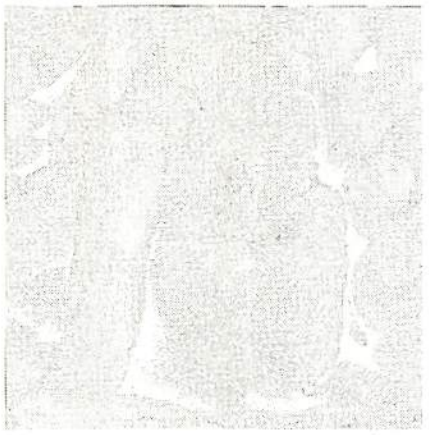
(b)



(c)

Figure A.5: Image compression on the 4×4 subimage block. (a) Original Pepper image; 512×512 pixels. (b) Reproduced image using the LBG method. (c) Reproduced image using the AVQ method.

(a)



(b)

Figure A.6: Difference image between the original Pepper image and the reconstructed image. (a) Difference image with inverse illuminance for the LBG method. (b) Difference image with inverse illuminance for the AVQ method.

Table A.2: Mean-squared errors and compression ratios for Fig. A.3 and Fig. A.5.

Algorithms / Images	LBG			AVQ			
	MSE	SNR	CR^1	MSE	SNR	CR^2	CR^3
Fig. A.3	45.04	26.63	16	49.44	26.22	10.24	15.75
Fig. A.5	64.83	24.67	16	62.92	24.80	8.99	13.25

Note: Vector size: 4x4 window.
CR^1 : compression ratio without codebook data.
CR^2 : compression ratio with codebook data.
CR^3 : compression ratio with codebook data and entropy reduction.

Table A.3: Mean-squared errors and compression ratios versus buffer-2 distortion thresholds for Girl and Pepper images.

Images / Performance / Thrd. 1	Thrd. 2	Girl			Pepper		
		MSE	CR	$\frac{CR}{MSE}$	MSE	CR	$\frac{CR}{MSE}$
1000	1000	48.95	9.13	0.187	62.92	8.99	0.143
1000	1200	49.08	9.29	0.189	63.77	9.13	0.143
1000	1400	49.74	9.41	0.189	64.61	9.19	0.142
1000	1600	50.79	9.58	0.189	65.21	9.32	0.143
1000	1800	52.22	9.63	0.184	64.84	9.43	0.145
1000	2000	53.28	9.81	0.184	65.57	9.52	0.145
1000	2200	54.67	9.99	0.183	66.11	9.55	0.144
1000	2400	56.60	10.18	0.180	66.41	9.64	0.145
1000	2600	59.24	10.37	0.175	67.90	9.67	0.142
1000	2800	61.21	10.43	0.170	68.63	9.72	0.142

Note: MSE : mean-squared error.
CR : compression ratio.
Thrd. 1 : distortion threshold of buffer-1.
Thrd. 2 : distortion threshold of buffer-2.

of the reconstructed data by using the block-data interpolation method. The distortion threshold of buffer-1 can be different from the distortion threshold of buffer-2. The distortion threshold of buffer-1 is chosen according to the performance requirement. Table A.3 lists the compression performances for different distortion thresholds of buffer-2. The average distortion values by using block-data interpolation are 1,350 and 2,460 for the Girl image and the Pepper image, respectively. Usually, a higher compression ratio, faster compression speed, and lower distortion between the original image and the reconstructed image are pursued in the data compression schemes. The good distortion thresholds of buffer-2 can be around 1,400, and 2,400 for the Girl image and the Pepper image, respectively, after optimizing against the compression ratios and mean-squared errors. Here, the criterion for selecting the good performance is proportional to $\frac{CR}{MSE}$, where CR is the compression ratio, and MSE is the mean-squared error. After encoding a image frame or subparts of the image, the distortion threshold of buffer-2 can be periodically updated by the average distortion value.

The codebook search of VQ method is an intense computation operation. If the squared error is used for the distortion measure, each codevector search consists of 31 additions and 16 multiplications for a 4x4 window. On the other hand, the distortion measure is performed by using the absolute error scheme. Since there is no multiplication for a codevector search, the computation speed can be improved and the complexity of VLSI hardware design can also be reduced. The absolute error scheme for distortion measure is defined as

$$\mu'(\mathbf{x}_i, \mathbf{y}_i) = \sum_{k=1}^{n} \mid x_{ik} - y_{ik} \mid . \tag{A.5}$$

Table A.4 illustrates the simulation results by using the AVQ method based on the absolute error scheme. The reconstructed Girl and Pepper images using the AVQ method with the distortion threshold of 100 are shown in Fig. A.7. The proposed AVQ method still can provide the good performance for the distortion measure using absolute error scheme. From the above computer analysis, the all simulation results show that the AVQ method

has several great outstanding features such as local adaptivity, less complexity, and fairly good compression ratio.

Table A.4: Mean-absolute errors and compression ratios versus distortion thresholds for Girl and Pepper images by using the absolute error scheme for distortion measure.

Images / Performance / Distortion Thrd. (MAE)	Girl				Pepper			
	CR	MAE	MSE	SNR	CR	MAE	MSE	SNR
90	7.43	5.35	51.02	26.09	8.39	4.89	62.42	24.83
100	8.76	5.31	49.48	26.22	8.97	4.96	63.28	24.77
110	10.04	5.40	50.32	26.15	9.52	5.26	67.82	24.47
120	11.05	5.54	52.41	25.97	10.07	5.36	69.78	24.35
130	11.72	5.81	58.03	25.53	10.74	5.54	73.05	24.15
140	12.22	6.14	65.81	24.98	11.13	5.92	81.74	23.66
150	12.64	6.52	74.27	24.45	11.51	6.26	87.83	23.35

Note: The same distortion threshold is used in buffer-1 and buffer-2.
CR : compression ratio, MAE : mean-absolute error.
MSE : mean-squared error, SNR : signal-to-noise ratio.

A.4 The VLSI Architecture

The high speed and high throughput of digital systems are pursued in the VLSI design methodology. Nowadays, the technologies can possess huge computing capabilities which make possible powerful personal workstations, sophisticated computer graphics, and multimedia system such as real-time digital video and speech recognition [A.23]. The proposed VLSI architecture for the AVQ method is shown in Fig. A.8. Major functional blocks are the array controller, move-to-front vector quantizers, and index generator. The array controller interprets control instructions from the host computer to set up the adaptive vector

(a)

(b)

Figure A.7: Image compression using the absolute error scheme for distortion measure on the 4×4 subimage block. (a) Reproduced Girl image using the AVQ method. (b) Reproduced Pepper image using the AVQ method.

quantizers and generates timing and control signals. The array controller can be implemented on standard DSP chips such as the TMS320C30/40 chip from Texas Instruments [A.24] or the DSP-56000/96000 chip from Motorola Inc. [A.25]. The two move-to-front vector quantizers function as the buffer-1 and buffer-2 of the Gold-Washing method. The indices, differential distortions, and signal-valid bits from the two MFVQs are applied to the index generator. The signal-valid bit S_{valid} is used to show the status of distortion measure for the best-matched codevector in the MFVQ. If the signal-valid bit is equal to Logic-1, the best-matched codevector in the MFVQ meets the distortion criterion. Otherwise, no codevector can match the distortion threshold requirement. The OR-gate function for the two signal-valid bits from the buffer-1 and buffer-2 are used to generate a decision signal I_{valid} back to the host machine to show the output index I_{out} valid or invalid. The output index I_{out} is generated according to the magnitudes of the differential distortions. If the differential distortion from buffer-1 is larger than that from buffer-2, the buffer-1 index with an extra bit of logic-1 forms the output I_{out} of the index generator. If the differential distortion from buffer-1 is smaller than or equal to that from buffer-2, then the buffer-2 index with an extra bit of logic-0 forms the output I_{out} of the index generator. The signal $Load_{p1}$ is generated for updating the priority value of each PE in buffer-1.

The proposed move-to-front vector quantizer consists of two major functions: vector quantization and codevector index updating. The codebook search and encoding procedure of the vector quantization can be expressed in a general matrix-vector multiplication form, where the multiplication operator represents the evaluation of scalar distortion and the addition operator is the sum of the scalar distortions. Therefore, systolic architectures can be implemented for this matrix-vector operation [A.26,A.27]. A dependence graph of vector quantization is shown in Fig. A.9 to illustrate the dependence of the computations. The linear systolic architectures of vector quantization can be classified into the distortion-stay systolic array and the distortion-move systolic array. In the distortion-stay systolic array, the distortion value u_i is associated with the i^{th} processing element (PE) in which the distortion is computed. In the distortion-move systolic array, the distortion value moves

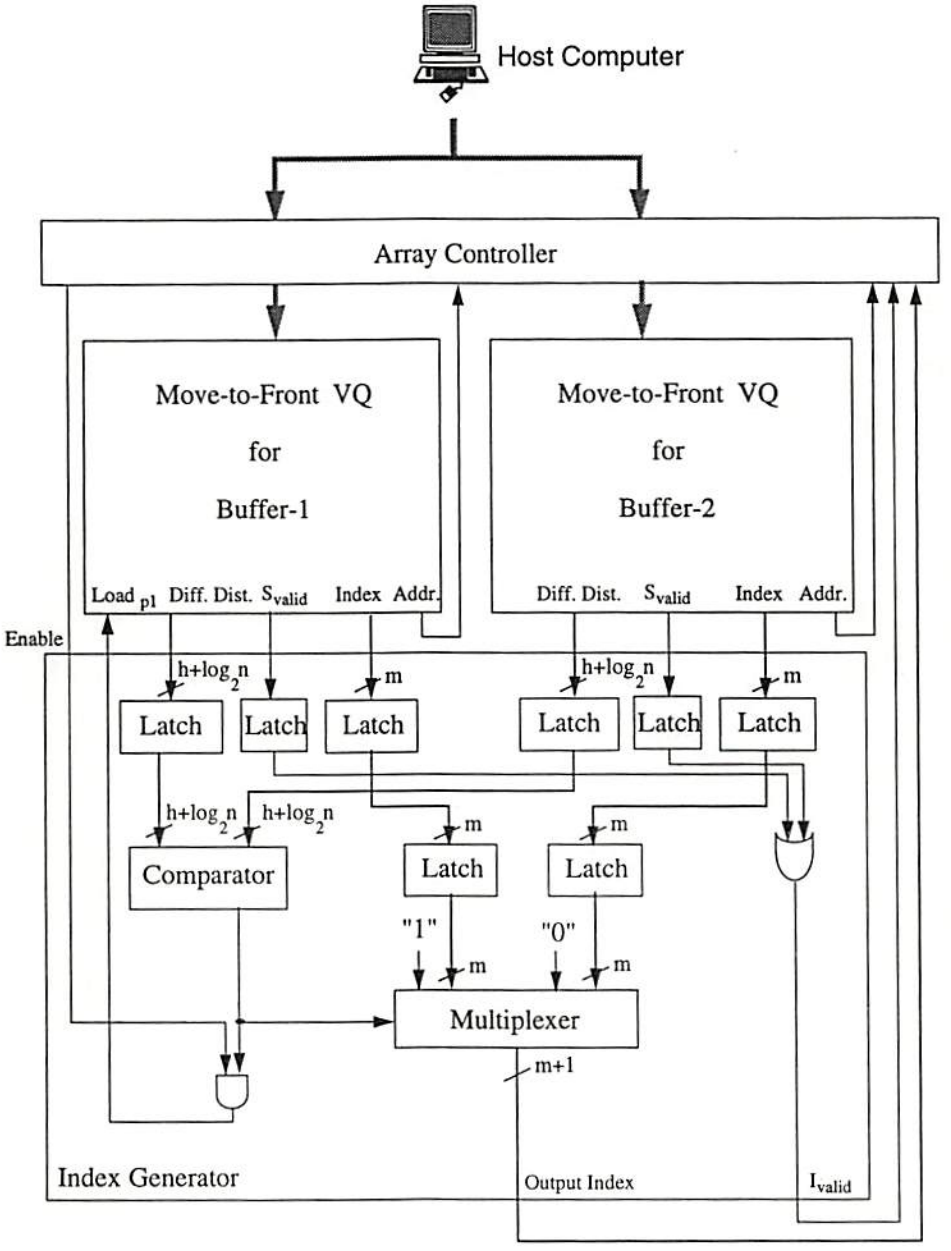


Figure A.8: Block diagram of the VLSI architecture for proposed AVQ method.

while the source vector component stays in the processing element. The throughput rate of the distortion-move VQ designs [A.28-A.31] are slower than the optimized throughput rate of 1 pixel/clock since codebook size is usually much larger than the codevector size. To achieve the optimized throughput rate, the distortion-stay VQ design [A.12] requires 2^m processors, where 2^m is the codebook size. The distortion-stay VQ can be further classified into the source-pipeline architecture and the source-parallel architecture. In the distortion-stay source-pipeline VQ shown in Fig. A.10 (a), the source vector is pipelined into a linear PE array. The distortion comparison is also performed in a pipelined architecture. In the distortion-stay source-parallel VQ shown in Fig. A.10 (b), the source vector is parallelly distributed into all processing elements for distortion calculation. The distortion comparison is performed in a parallel architecture. The numbers of the PEs and comparison (CP) units for the source-pipeline and the source-parallel architectures are the same. The throughput rates of these two architectures are also the same. The source-pipeline architecture reduces the connections between source vector and PEs, and increases the dependence between the PEs. If one codevector needs to be updated during encoding, the source-parallel architecture will lose only one source vector search but the source-pipeline architecture will lose many source vector searches depending on the location of the new codevector. Therefore, the distortion-stay source-parallel architecture is preferred in the proposed AVQ method.

Since the MFVQ explores parallelism in the direction of codebook size and pipelining in the direction of vector dimension, it is the distortion-stay source-parallel architecture. Block diagram of the MFVQ is shown in Fig. A.11. The major functional modules are SRAM [A.32] for codebook data, two counters for the codebook memory address, a source vector pipeline buffer, distortion-computation processing elements, a winner calculation module, an index calculation module, and an priority updating module [A.33]. The input data sequence continuously flows through the MFVQ. The source vector pipeline buffer is h-bit wide and n-word deep, where h is the bit number for representing the intensity value of a pixel, and n is a vector length. It serves as a first-in-first-out buffer for every

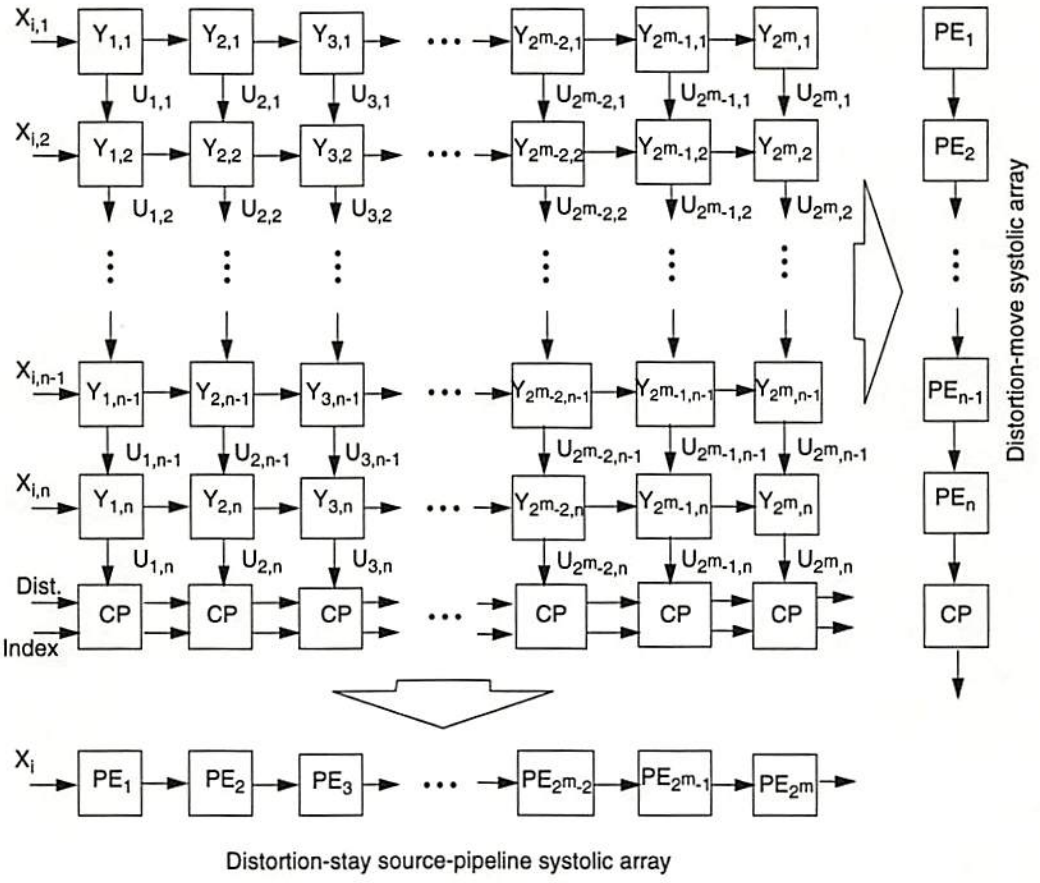


Figure A.9: Dependence graph of vector quantization.

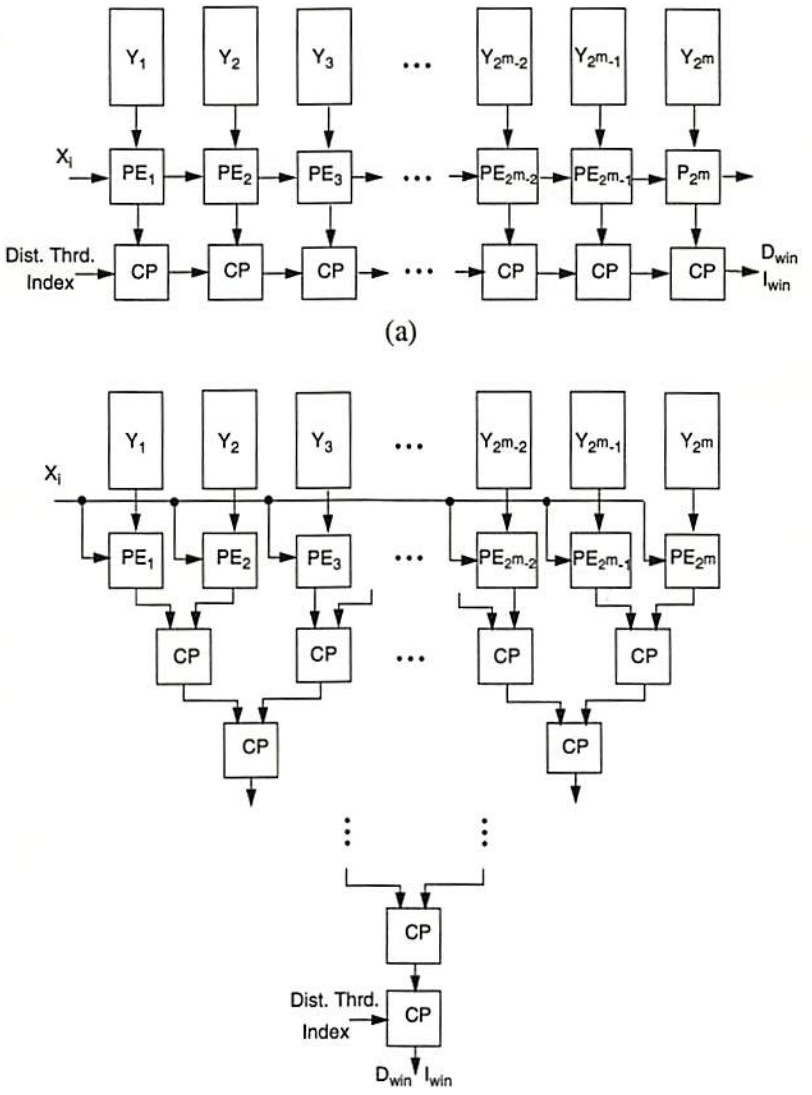


Figure A.10: Distortion-stay systolic architecture for vector quantization. (a) Distortion-stay source-pipeline architecture. (b) Distortion-stay source-parallel architecture.

input vector which consists of n elements. The first latch of this pipeline buffer distributes the component of source vector into each PE during encoding. The pipeline buffer also distributes the source vector into the SRAM during codebook updating. In order to set up the initial priority entry for each codevector, the $Reset_p$ signal is issued before starting the encoding operation. The two counters generate the codebook memory addresses for the two memory bands. The host computer will reset the counters before the PE starts to calculate the distortion error between source data and each codevector.

Figure A.12 shows the major functional blocks of each processing element (PE), which consists of data latches, distortion-computing data path, and a comparator. According to Davidson's method [A.29], the mean-squared error calculation can be reduced to the inner product calculation. However, the multiplication still is required for the distortion measure. In our method, the good performance listed in Table A.4 can also be achieved by using the absolute error scheme. Here, distortion computation in the PE is the absolute error calculation in terms of mean-squared error calculation. The advantages are that a multiplier is not required and the intermediate data dimensions are reduced in the PE design. During encoding, the codevector components are addressed by the counter. An accumulator collects the intermediate result from the absolute differential value between an element of codevector and an element of source vector. After n clock cycles, The accumulator will consecutively contain the total absolute differential value between the source vector $\mathbf{x}_i$ and the codevector $\mathbf{y}_j$. In order to reduce the VLSI implementation complexity of the MFVQ, each PE can calculate two or more distortion values for each source vector so that the total number of PEs in the MFVQ can be decreased with the tolerable reduction of throughput rate. Here, each PE matches the source vector with the two codevectors sequentially. The control instructions ($Select_1$ and $Select_2$) are used to load the total absolute differential values into the latches, respectively. The comparator selects the minimum distortion from the two absolute differential values. The results in each PE are stored in the sign-bit register and distortion register while $Load_s$ instruction is active.

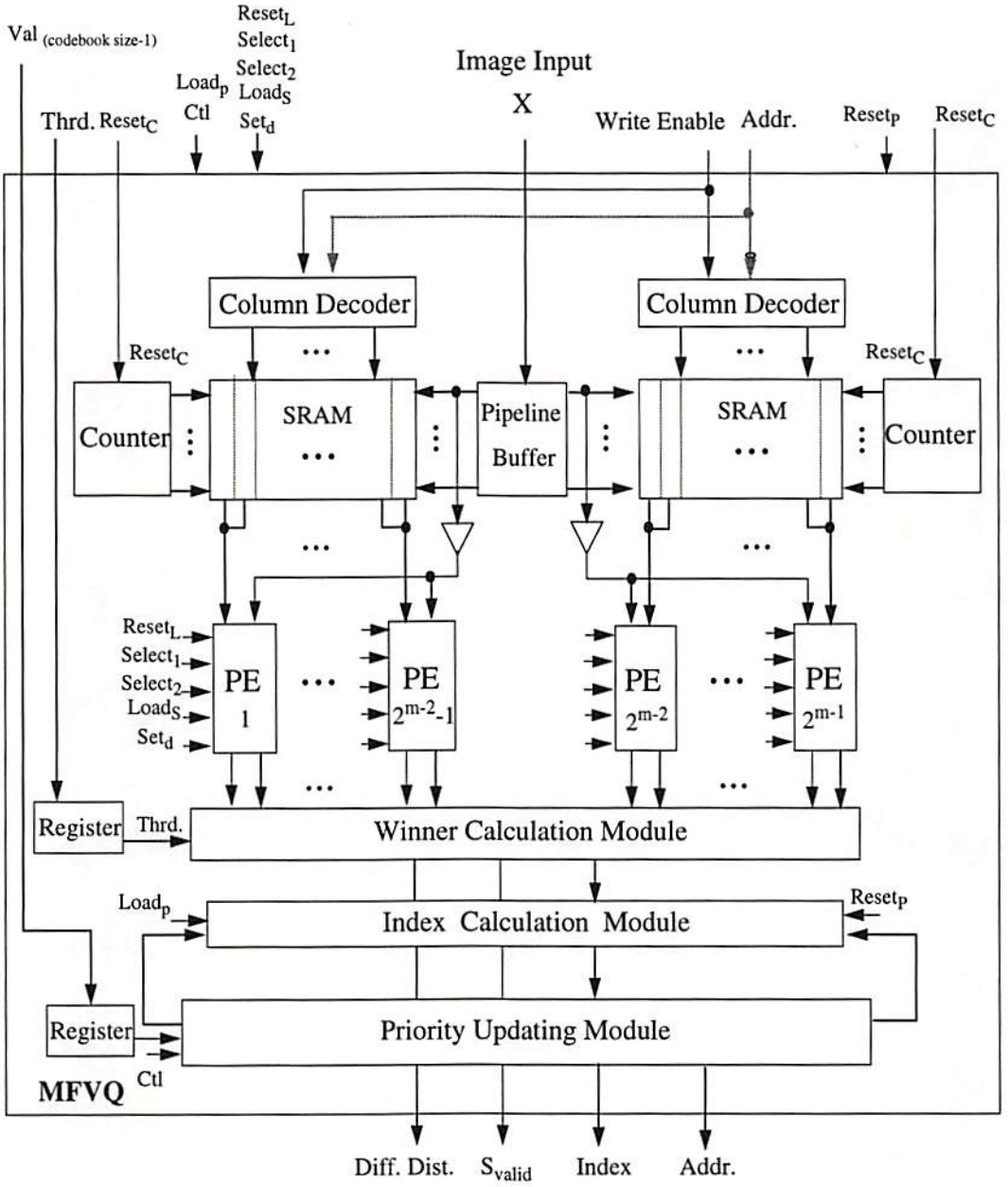


Figure A.11: The Functional block diagram of the MFVQ.

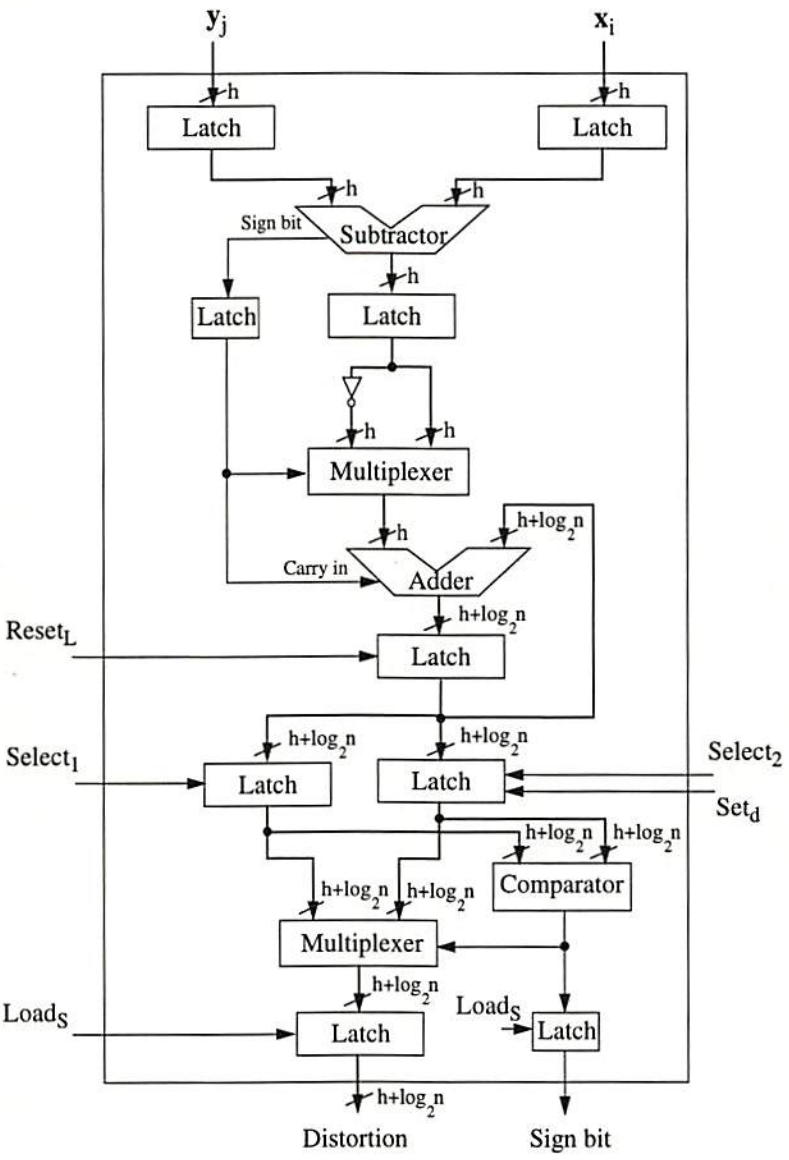


Figure A.12: Detailed block diagram of the PE.

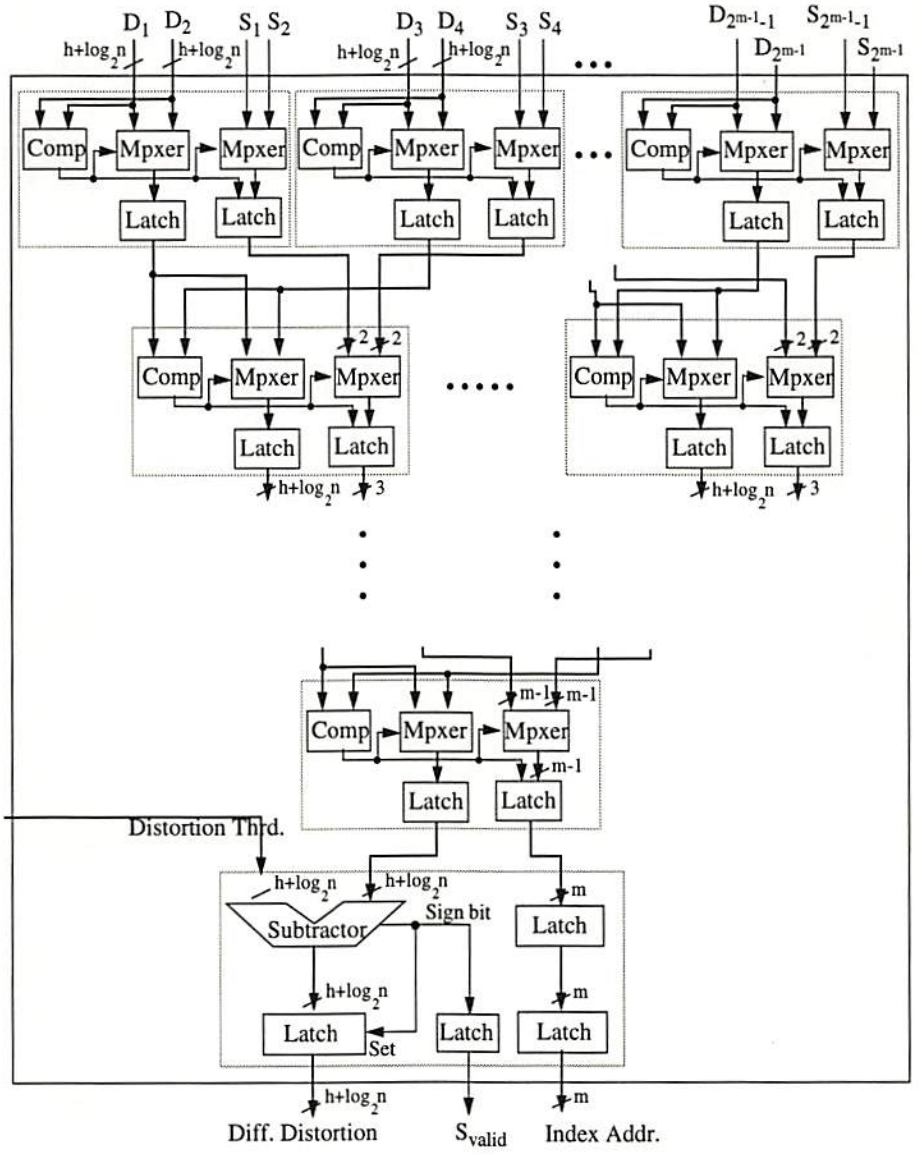


Figure A.13: Functional block diagram of the winner calculation module. (Note: Mpxer: multiplexer; Comp: comparator.)

The winner calculation module shown in Fig. A.13 uses the sign bits, the distortion values, and distortion threshold to determine the winner. The minimum distortion is obtained by using the hierarchical comparison from the distortion results of the all PEs and the corresponding index address is also produced from cascading the sign bits of the comparators in each layer. The subtractor performs the comparison between the minimum distortion and the distortion threshold. If the minimum distortion is larger than the threshold, the register for storing the differential distortion is set to zero. Otherwise, the differential distortion and its index address are stored in the registers. The sign bit of the subtractor is the signal-valid bit S_{valid} of the MFVQ. In the index calculation module shown in Fig. A.14, the index address from the winner calculation module are decoded by a m-bit decoder. The output signals of the m-bit decoder are used to control the pass transistor gates. Only the winner priority value goes through the pass transistor gates to the output index register. In order to achieve the scalability of the codebook size, the total bits of priority value can be larger than that of the index address by m^*. The additional m^* bits are used for the multichip design. The maximum 2^{m^*} MFVQ chips can be performed in parallel for buffer-1 or buffer-2. Functional block diagram of the priority updating module is shown in Fig. A.15. The priority updating module compares the output index I value from the index calculation module with values P from the priority table. If the difference is zero, the priority value is set to the value of codebook size minus one, which is used as a high priority. If I is smaller than P, the old priority value is decreased by one. Otherwise, there is no change in the priority value. The priority updating is performed only while the $Load_p$ instruction is active. In the encoding operation, the $Load_p$ signal for buffer-1 is active while the new priority value is ready. According to the Gold-Washing method, the $Load_p$ signal is not active for buffer-2 during the encoding operation. On the other hand, the $Load_p$ signal is active for both buffer-1 and buffer-2 during the codebook-updating operation. In this case, the Ctl signal selects the zero value as the input of the comparator. All the priority values which are larger than zero are decreased by one. The priority value, which was equal to zero, is assigned to the high priority and the

corresponding codevector will be replaced by a new value. The C_{eq} signals from the all comparators of the priority updating module are encoded by the 2^m-to-m encoder which is used to generate the address of the new codevector for the host machine.

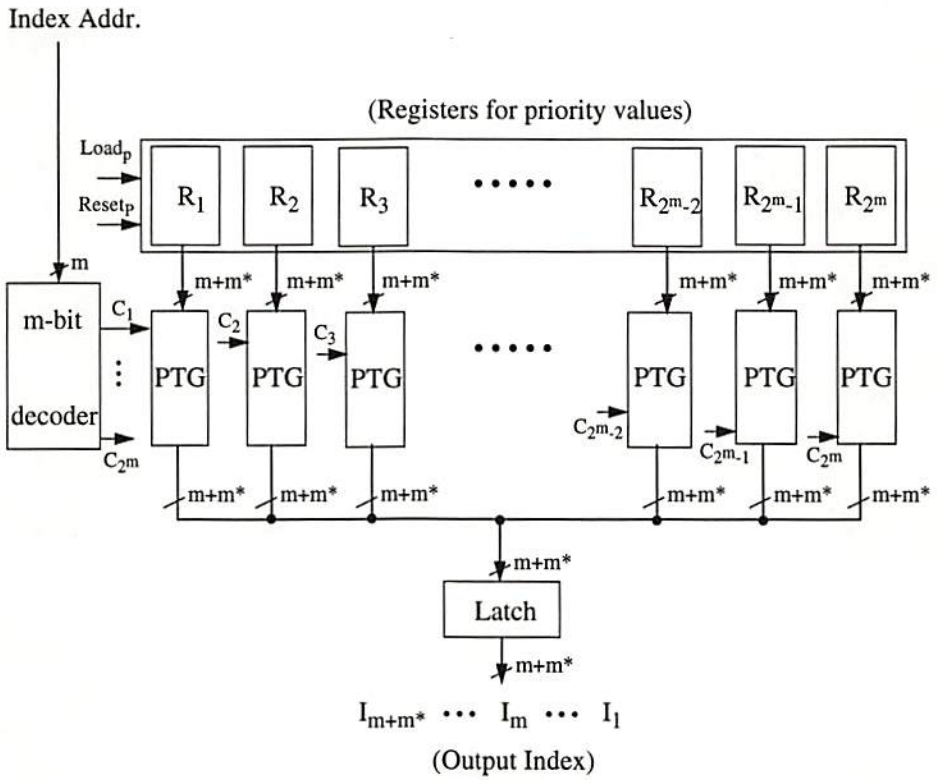


Figure A.14: Index calculation module. (Note: PTG: pass transistor gates.)

Since the computation complexity of the block-data interpolation method is very low, it can be implemented by the custom VLSI or the standard DSP chips. The source vector is not only pipelined into the MFVQs but also into the block-data interpolation module simultaneously. If the signal-valid bit I_{valid} from the index generator is equal to logic-1, then the Huffman code for this index value and an identification code form the encoded data. If the signal-valid bit I_{valid} from the index generator is equal to logic-0, then a new codevector and the encoded result are generated from the block-data interpolation module. In this situation, the host machine will issue the *Ctl* instructions to the priority updating modules of the MFVQs for updating indices and generating codebook memory addresses.

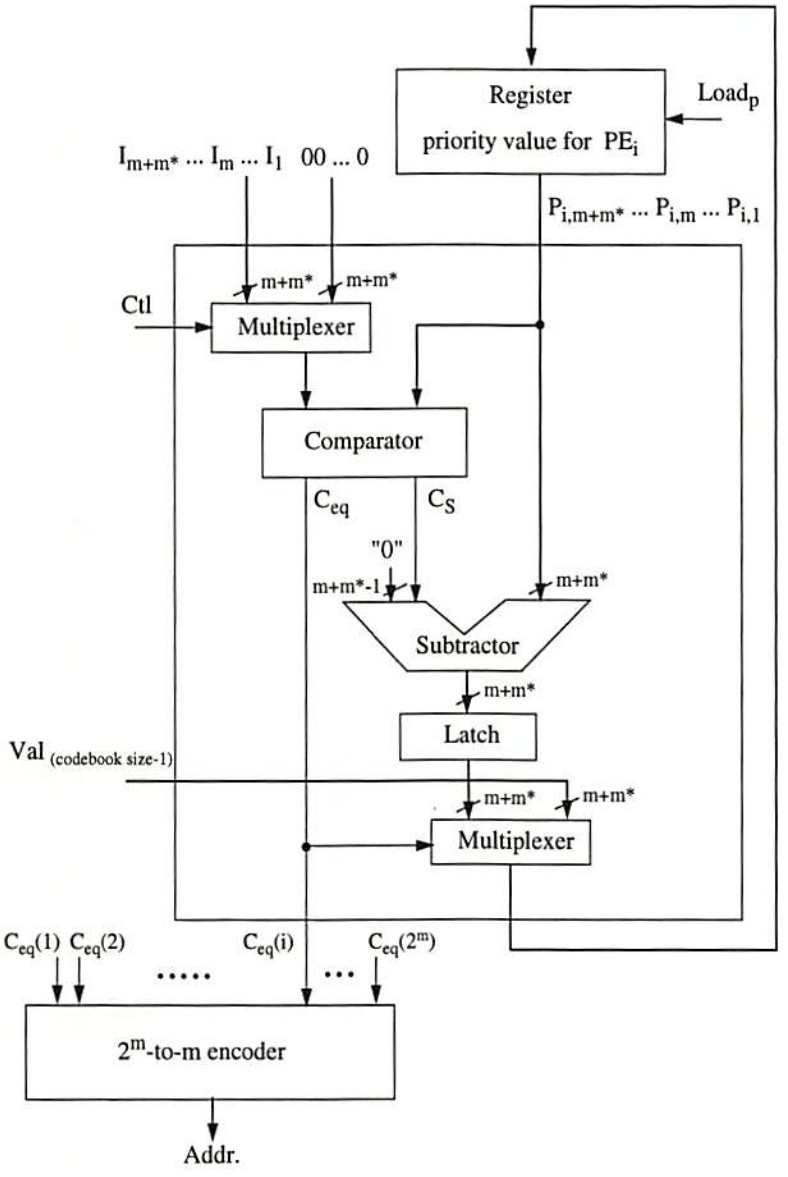


Figure A.15: The priority updating module for one codevector.

By using these codebook memory addresses, the codebooks of buffer-1 and buffer-2 will be updated by the host machine according to the Gold-Washing method.

In the MFVQ, the number of processing units is 2^{m-1}. If each PE addresses two codevectors, the pipeline latency for encoding one codevector equals $2n$ clock cycles. The input data rate is 0.5 pixel per clock cycle, e.g., $\frac{h}{2}$ bits per clock cycle. In the computer simulations for the AVQ method, the number of codebook updating is around 10% to 20% of the number of the source vectors, where the codebook size is 256 for buffer-1 and buffer-2, and the vector dimension is 16. The host machine will be noticed by the index-valid bit I_{valid} of the index generator after 15 clock cycles for processing next source vector. If there is no codevector to match the distortion criterion, the new codevector data are pipelined into the MFVQs. It requires 16 clock cycles to fill the source vector pipeline buffer. At the same time, the host machine issues the *Ctl* signal to the MFVQ of buffer-2 to get the codebook memory address. The frequency of the last entry of buffer-2 is also used by the host machine to determine the codebook updating of buffer-1. If the codebook updating is required, the *Ctl* signal is also issued for buffer-1. The generations of codebook memory addresses are overlapped with the operations of codevectors pipelined into the MFVQs. The Write-Enable signal for the SRAMs is performed to update the codebook data at one clock cycle. After the codebook updating, the MFVQ needs additional 4 clock cycles to fill the pipelining latency for encoding next source vector. The total processing time for codebook updating is 36 clock cycles. Therefore, the throughput rate is reduced to 0.45 pixel/clock for 10% codebook updating or 0.41 pixel/clock for 20% codebook updating. In the applications of high-definition television, the proposed AVQ method can be used to compress a 1408x960-pixel color video source according to the Digicipher system [A.34]. Empirically, the 20% codebook updating is a worse case in the intraframe compression. Since the neighboring frames in the video source are similar, the frequency of codebook updating will be reduced. If the 20% codebook updating is assumed for all video frames and the three quantizers using the AVQ method are performed for the three different colors individually, the processing speed can reach 30.3 frames/sec at a system clock 100 MHz.

In the proposed architecture, the codebook size for one MFVQ can be scaled down by half. In this situation, each PE only addresses one codevector. The counter for the codebook memory address will be reset to zero after its value is equal to $n-1$. In the PE, the latch for storing the second total absolute differential value is set to a high distortion value, and the $Select_2$ signal is never active. Therefore, the MFVQ can operate at 1 pixel/clock in the encoding operation. Due to additional m^* bits used for the priority value, the codebook size can be scaled to 2^{m+m^*}. If each MFVQ performs with the codebook size of 2^m, the maximum 2^{m^*} MFVQs can be arranged in a parallel architecture for high-speed processing. Such an architecture for the MFVQ has the advantages of modularity, regular data flow, simple interconnection, simple global control, pipeline processing, and parallel processing [A.26,A.27]. Hence it is well suited for VLSI implementation.

A.5 Design of Modules

For signal processing applications, one appropriate measure would be the numbers of operation per second, where an operation is defined as data access, store, add, shift or multiply. When both buffer-1 and buffer-2 have 256 codevectors, the required indices are 8-bit data. If the intensity of each pixel is represented by an 8-bit datum and the 4x4 subimage block is used, the total bits of the absolute differential error between the source vector and codevector will be 12 bits. Hence, the latches for the codevector index and differential distortion are 8-bit and 12-bit data latches, respectively. In the MFVQ, the pipeline buffer is used to store all the components of a codevector. The 7-bit column decoders are used for the two memory bands which consist of 128 codevectors individually. Since the vector dimension is 16 and each PE addresses two codevectors, the counter in the MFVQ is a 5-bit synchronous counter. The SRAM for storing the codebook data is implemented by the two-port memory cells with one read port and one write port. The high-speed current-mode sense amplifier is implemented in the SRAM design [A.35]. According to the SPICE simulation results [A.36], the 5-ns memory access time and 10-ns cycle time for 16 words x 16 bits can be achieved by using a 0.8 μm CMOS technology from

Hewlett-Packard Co. through the MOSIS Service of USC/Information Sciences Institute at Marina del Rey, CA [A.37,A.38]. The total memory size requires 4.08 K bytes.

Major functional units of the PE are the data latches, an 8-bit subtractor, 8-bit and 12-bit multiplexers, a 12-bit adder and a 12-bit comparator. The accumulating operations are performed in a single processor cycle that is overlapped with the codebook memory read cycle. Each functional unit is simulated by the SPICE circuit simulator and its corresponding physical layout is generated. The delay time for the 12-bit carry lookahead adder is around 4.5 ns. The encoding rate is constrained by the codebook memory access. The silicon area for the custom PE design is about 1.8 x 2.4 mm^2 in a 0.8-μm CMOS technology. It can provide a computing capability of 50M pixels per second at a system clock 100 MHz. Each pipelining operation in the PE consists of memory fetch, subtraction, and accumulation. The computation power can reach 300 million operations per second (MOPS) The total transistor count is about 5K. The performance estimation of the PE unit is illustrated in the Table A.5.

Table A.5: Performance estimation of the PE unit.

Design Rule	HP 0.8-µm CMOS, N-well, 3-metal
Chip Size	1.8 mm x 2.4 mm
Operating Frequency	100 MHz (Simulated)
Transistor Count	5 K
Supply Voltage	5 V
Power Dissipation	30 mW@ 5 V
Pin Count	40
Signal Representation	digital
Processing Speed	300 MOPS
Desgin Style	full custom design

The winner calculation module consists of a 12-bit subtractor, 12-bit comparators, multiplexers, and data latches. The 128 comparisons are performed in 7 hierarchical

Table A.6: Computation power estimation.

Processing Unit	Processing Speed
128 PEs	38,400 MOPS
winner calculation module	400 MOPS
index calculation module	3 MOPS
priority updating module	1,600 MOPS

layers. The numbers of comparisons in each layer are 64, 32, 16, 8, 4, 2, and 1. The cascaded sign bits from the comparators are stored in the data latches of each layer. The functional units of the index calculation module is an 8-bit decoder and the pass transistor gates. If the value of the additional m^* bits is equal to zero, the registers for storing the priority values are 8-bit data registers. In the priority updating module, the major function units are 8-bit multiplexers, 8-bit comparators, 8-bit subtractors, 8-bit data latches, and a 256-to-8 encoder. The computation speed for each module in the MFVQ is listed in Table A.6. The processing power of the MFVQ at a system clock 100 MHz can reach 40 billion operations per second. The total transistor count in the MFVQ can be about 1.8 million and the silicon area can be about 40 x 45 mm^2. The power consumption is around 7 W. Such a system with a very large transistor count can be achieved by using either MCM (multi-chip module) technology [A.39] or by ULSI technologies with built-in fault tolerance [A.40]. This VLSI design meets light-weight, small-volume, high-speed, and user-transparent requirements and can play a crucial role in the high-performance data processing systems.

References

[A.1] A. Gersho, R. M. Gray, *Vector Quantization and Signal Compression,* Kluwer Academic Publishers: Boston, MA 1992.

[A.2] N. M. Nasrabadi, R. A. King, "Image coding using vector quantization: a review," *IEEE Trans. on Communications,* vol. 36, no. 8, pp. 957-971, Aug. 1988.

[A.3] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. Journal,* vol. 27, pp. 379-423, and 623-656, 1948.

[A.4] R. M. Gray "Vector quantization," *IEEE ASSP Magazine,* pp. 4-29, Apr. 1984.

[A.5] Y. Linde, A. Buzo, R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. on Communications,* vol. COM-28, no. 1, pp. 84-95, Jan. 1980.

[A.6] A. Gersho, B. Ramamurthi, "Image coding using vector quantization," *Proc. of International Conf. on Acoustics, Speech, and Signal Processing,* vol. 1, pp. 428-431, May 1982.

[A.7] A. Buzo, A. H. Gray, R. M. Gray, J. D. Markel, "Speech coding based upon vector quantization ," *IEEE Trans. on Acoustics Speech and Signal Processing,* vol. ASSP-28, no. 5, pp. 562-574, Oct. 1980.

[A.8] A. Gersho, V. Cuperman, "Vector quantization: a pattern-matching technique for speech coding," *IEEE Communications Magazine,* vol. 21, pp. 15-21, Dec. 1983.

[A.9] A. Gersho, M. Yano, "Adaptive vector quantization by progressive codevector replacement," *Proc. of International Conf. on Acoustics, Speech, and Signal Processing,* vol. 1, pp. 133-136, Mar. 1985.

[A.10] M. Goldberg, P. R. Boucher, S. Shlien, "Image compression using adaptive vector quantization," *IEEE Trans. on Communications,* vol. COM-34, pp. 180-187, Feb. 1986.

[A.11] K. Zeger, A. Bist "Universal adaptive vector quantization using codebook quantization with application to image compression," *Proc. of International Conf. on Acoustics, Speech, and Signal Processing,* vol. 3, pp. 381-384, Mar. 1992.

[A.12] O. T.-C. Chen, Z. Zhang, B. J. Sheu, "An adaptive high-speed lossy data compression," J. A. Storer, M. Cohn (Eds.), *Proc. of Data Compression Conference,* IEEE Computer Society Press: Los Alamitos, CA, pp. 349-358, Mar. 1992.

[A.13] Z. Zhang, V. K. Wei, "A universal lossy data compression algorithm by continuous codebook sieving," *was submitted to IEEE Trans. on Information Theory,*

[A.14] J. Bentley, D. Sleator, R. Tarjan, V. K. Wei, "A locally adaptive data compression scheme," Communications of the ACM, vol. 29, pp. 320-330, 1986.

[A.15] P. Elias, "Interval and recency rank source coding: two on-line adaptive variable length schemes," *IEEE Trans. on Information Theory,* vol. IT-33, no. 1, pp. 1-15, Jan. 1987.

[A.16] D. S. Ornstein, P. Shields, "Universal almost sure data compression," *The Annual of Probability,* vol. 18, no. 2, pp. 441-452, 1990.

[A.17] D. Knuth, "Dynamic Huffman Coding," *Journal of Algorithms,* pp. 163-180, Jun. 1985.

[A.18] J. W. Woods, *Subband Image Coding,* Kluwer Academic Publishers: Boston, MA 1990.

[A.19] P. H. Ang, P. A. Ruetz, D. Auld, "Video compression makes big gains," *IEEE Spectrum,* pp. 16-19, Oct. 1991.

[A.20] D. L. Gall "MPEG: a video compression standard for multimedia applications," *Communications of the ACM,* vol. 34, no. 4, pp. 46-58, Apr. 1991.

[A.21] T. A. Welch, "A technique for high performance data compression," *IEEE Computer Magazine,* vol. 17, no. 6, pp. 8-19, June 1984.

[A.22] A. Lempel, J. Ziv, "On the complexity of finite sequences," *IEEE Trans. on Information Theory,* vol. IT-22, pp. 75-81, Jan. 1976.

[A.23] R. Brodersen, A. Chandrakasan, S. Sheng, "Low-power signal processing systems," *VLSI Signal Processing V,* K. Yao, R. Jain, W. Przytula, J. Rabaey (Eds.), pp. 3-13, IEEE Press: New York, Oct. 1992.

[A.24] *TMS320C4x User's Guide,* Texas Instruments Inc., Dallas, TX, 1991.

[A.25] *DSP56000 Digital Signal Processor User's Manual,* Motorola Inc., 1986.

[A.26] H. T. Kung, "Why systolic architectures ?," *IEEE Computer Magazine,* vol. 15, no. 1, pp. 37-46, Jan. 1982.

[A.27] S. Y. Kung, *VLSI Array Processors,* Prentice Hall: Englewood Cliffs, NJ, 1988.

[A.28] P. R. Cappello, G. A. Davidson, A. Gersho, C. Koc, V. Somayazulu, "A systolic vector quantization processor for real-time image coding," *Proc. of International Conf. on Acoustics, Speech, and Signal Processing,* vol. 3, pp. 2143-2146, Tokyo, Japan, 1986.

[A.29] G. A. Davidson, P. A. Cappello, A. Gersho, "Systolic architectures for vector quantization," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. 36, no. 10, pp. 1651-1664, Oct. 1988.

[A.30] R. Dianysian, R. L. Baker, "A VLSI chip set for real-time vector quantization of image sequences," *Proc. of Int. Symp. on Circuits and Systems,* vol. 1, pp. 221-224, May 1987.

[A.31] P. A. Ramamoorthy, B. Potu, T. Tran, "Bit-serial VLSI implementation of vector quantizer for real-time image coding," *IEEE Trans. on Circuit and Systems,* vol. 36, no. 10, pp. 1281-1290, Oct. 1989.

[A.32] H. Okuyama, T. Nakano, S. Nishida, E. Aono, H. Satoh, S. Arita "A 7.5-ns 32K x 8 CMOS SRAM," *IEEE Jour. of Solid-State Circuits,* vol. 23, no. 5, pp. 1054-1059, Oct. 1988.

[A.33] N. Weste, K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective,* Addison Wesley: Reading, MA, 1985.

[A.34] A. Harris, "Putting the right numbers into HDTV," *Electronics World + Wireless World,* pp. 481-485, June 1992.

[A.35] K. Sasaki, K. Ishibashi, K. Ueda, K. Komiyaji, T. Yamanaka, N. Hashimoto, H. Toyoshima, F. Kojima, A. Shimizu, "A 7ns 140mW 1Mb CMOS SRAM with current sense amplifier," *IEEE ISSCC Dig. Tech. Papers,* pp. 148-149, San Francisco, CA, Feb. 1992.

[A.36] B. Johnson, T. Quarles, A. R. Newton, D. O. Pederson, A. Sangiovanni-Vincentelli, *SPICE-3E1 User's Guide,* Dept. of Electrical Engineering and Computer Sciences, Univ. of California, Berkeley, Apr. 1991.

[A.37] C. Tomovich, "MOSIS - A gateway to silicon," *IEEE Circuits and Devices Magazine,* vol. 4, no. 2, pp. 22-23, Mar. 1988.

[A.38] G. Lewicki, "Foresight: A fast turn-around and low cost ASIC prototyping alternative," *Proc. of IEEE ASIC Seminar and Exhibit,* pp. 6-8.1/8.2, Rochester, NY. Sept. 1990.

[A.39] W. M. Siu "MCM and monolithic VLSI perspectives on dependencies, integration, performance and economics," *Proc. of IEEE Multi-Chip Module Conference,* pp. 4-7, March 1992.

[A.40] M. Griffin, G. Tahara, K. Knorpp, R. Pinkham, B. Riley, "An 11-million transistor neural network execution engine," *IEEE ISSCC Dig. Tech. Papers,* pp. 180-181, San Francisco, CA, Feb. 1991.

Appendix B

Publications Out of the Dissertation Work

Journal Papers

- W.-C. Fang, B. J. Sheu, **O. T.-C. Chen**, J. Choi, "A VLSI neural processor for image data compression using self-organization networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 506-518, May 1992.
- **O. T.-C. Chen**, B. J. Sheu, W.-C. Fang, "Image compression on a VLSI neural-based vector quantizer," *Journal of Information Processing and Management*, vol. 28, no. 6, pp. 687-706, Pergamon Press Ltd.: New York, NY, 1992.
- W.-C. Fang, C. Chang, B. J. Sheu, **O. T.-C. Chen**, J.-C. Curlander, "VLSI systolic binary tree-searched vector quantizer for image compression," *IEEE Transactions on VLSI Systems*, vol. 2, no. 1, pp. 33-44, Mar. 1994.
- J. Choi, B. J. Sheu, O. **T.-C. Chen**, "A monolithic GaAs receiver for optical interconnect systems," *IEEE Journal of Solid-State Circuits*, vol. 29, no.3, pp. 328-331, Mar. 1994.
- **O. T.-C. Chen**, Z. Zhang, B. J. Sheu, "An adaptive vector quantizer based on the gold-washing method," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no.2, pp. 143-157, April 1994.
- **O. T.-C. Chen**, B. J. Sheu, W.-C. Fang, "Image compression using self-organization networks," will appear to *IEEE Transactions on Circuits and Systems for Video Technology*, Dec 1994.
- S. H. Bang, **O. T.-C. Chen**, J. C.-F. Chang, B. J. Sheu, "Paralleled hardware annealing in multi-level hopfield neural networks for optimal solutions" will appear to *IEEE Transactions on Circuits and Systems: Part II*, vol. 41, no. 12, Dec. 1994.
- **O. T.-C. Chen**, B. J. Sheu, "Neural network learning on space trajectories without time-delay elements," was submitted to *IEEE Transactions on Circuits and Systems: Part II*,1994.

Conference Papers

- W.-C. Fang, B. J. Sheu, **O. T.-C. Chen**, "A neural network based VLSI vector quantizer for real-time image compression," *Proc. of IEEE & NASA Data Compression Conference*, J. A. Storer, J. H. Reif (Ed.), IEEE Computer Society Press: Los Alamitos, CA, pp. 342-251, April 1991.

- **O. T.-C. Chen**, B. J. Sheu, W.-C. Fang, "Adaptive codebook construction and real-time hardware implementation for binary tree-searched vector quantization," *Proc. of IEEE Workshop on Visual Signal Processing and Communication*, Hinchu, Taiwan, pp. 35-38, June 1991.

- B. J. Sheu, C.-F. Chang, T.-H. Chen, **O. T.-C. Chen**, "Neural-based analog trainable vector quantizer and digital systolic processors," *Proc. of IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 1380-1383, Singapore, June 1991.

- W.-C. Fang, B. J. Sheu, **O. T.-C. Chen**, "A real-time VLSI neuroprocessor for adaptive image compression based upon frequency-sensitive competitive learning," *Proc. of IEEE/INNS International Joint Conference on Neural Networks*, vol. I, pp. 429-436, Seattle, WA, July 1991.

- **O. T.-C. Chen**, B. J. Sheu, W.-C. Fang, "Adaptive vector quantizer for image compression using self-organization approach," *Proc. of IEEE International Conf. on Acoustic, Speech and Signal Processing*, vol. II, pp. 385-388, San Francisco, CA, Mar. 1992.

- **O. T.-C. Chen**, Z. Zhang, B. J. Sheu, "An adaptive high-speed lossy data compression," *Proc. of IEEE & NASA Data Compression Conference*, J. Storer, M. Cohn (Ed.), IEEE Computer Society Press: Los Alamitos, CA, pp. 349-358, Mar. 1992.

- **O. T.-C. Chen**, Z. Zhang, B. J. Sheu "An adaptive high-speed vector quantization," *VLSI Signal Processing V*, Editors: K. Yao, R. Jain, I. Rabaey, IEEE Press: New York, pp. 205-214, Oct. 1992.

- **O. T.-C. Chen**, T. Berger, B. J. Sheu, "VLSI implementation of the hippocampal dentate gyrus," INNS World Congress on Neural Networks, vol. II, pp. 647-652, San Diego, CA, June 1994.

- **O. T.-C. Chen**, B. J. Sheu, "Neural networks for learning space trajectories based on quasi-Newton method," *INNS World Congress on Neural Networks*, vol. IV, pp. 354-359, San Diego, CA, June 1994.

- **O. T.-C. Chen**, B. J. Sheu, "Optimization schemes for neural network training," *IEEE World Congress on Computational Intelligence*, vol. II, pp. 817-822, Orlando, FL, June 1994.

- **O. T.-C. Chen**, T. Berger, B. J. Sheu, "VLSI implementation of the hippocampal on nonlinear system model," *IEEE World Congress on Computational Intelligence*, vol. IV, pp. 2009-2014, Orlando, FL, June 1994.

Technical Reports

- B. J. Sheu, **O. T.-C. Chen**, Digital Image Processing on VLSI, USC-SIPI Report #186, August 1991.
- **O. T.-C. Chen**, B. J. Sheu, "VLSI architecture for wavelet transform on digital images," *Proc. of Annual Research Review*, Signal and Image Processing Institute, Dept. of Electrical Engineering, University of Southern California, April 1992.
- W.-C. Fang, M. Chen, B. J. Sheu, **O. T.-C. Chen**, "Survey of image compression methods and VLSI hardware," *Proc. of Annual Research Review*, Signal and Image Processing Institute, Dept. of Electrical Engineering, University of Southern California, April 1992.
- **O. T.-C. Chen**, B. J. Sheu, "Fractal-based image synthesis and compression," *Proc. of Annual Research Review*, Signal and Image Processing Institute, Dept. of Electrical Engineering, University of Southern California, Feb. 1993.