DYNAMICALLY SELF-RECONFIGURABLE SYSTEMS FOR MACHINE INTELLIGENCE

A dissertation presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Doctor of Philosophy

Haibo He

August 2006

This dissertation entitled

DYNAMICALLY SELF-RECONFIGURABLE SYSTEMS FOR

MACHINE INTELLIGENCE

by

HAIBO HE

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Janusz A. Starzyk

Professor of Electrical Engineering and Computer Science

Dennis Irwin

Dean, Russ College of Engineering and Technology

Abstract

HE, HAIBO, Ph.D., August 2006, Electrical Engineering and Computer Science

DYNAMICALLY SELF-RECONFIGURABLE SYSTEMS FOR MACHINE INTELLIGENCE (170 pp.)

Director of Dissertation: Janusz A. Starzyk

This dissertation is focused on the development of system level architectures and models of dynamically self-reconfigurable systems for machine intelligence. This research is significant for building brain-like intelligent systems. Although the development of deep submicron very large scale integration (VLSI) system, nanotechnology and bioinformatics facilitate building such intelligent systems, yet it is very challenging to study how these kinds of complex, reconfigurable systems can self-develop their connectivity structures, accumulate knowledge, make associations and predictions, dynamically interact with environment, and self-control to accomplish desired tasks.

A new framework of “learning-memory-prediction” for machine intelligence is proposed in this research, and it serves as the foundation for building intelligent systems through learning in dynamic value systems, memorizing in self-organizing networks, and predicting in hierarchical structures. These systems are characterized by on-line data driven learning, distributed structure of processing components with local and sparse interconnections, dynamic reconfigurability, self-organization, and active interaction with environment.

Learning is the fundamental element for biologically intelligent systems. The proposed online value system is able to learn and dynamically estimate the value of any multi-dimensional data set, and such value system can be used in reinforcement learning. Feedback mechanism is introduced in the self-organizing learning system to allow the machine to be able to memorize information in its distributed processing elements and make associations. After the information is learned and stored in the associative memory, a biologically-inspired anticipation-based temporal sequence learning architecture is proposed. All systems proposed in this research are hardware-oriented. A novel computing paradigm that can achieve low power consumption for designing large scale, high density intelligent systems is proposed, and a brief description of the system level hardware architecture for prototyping and testing of the proposed systems is also presented.

Intelligent systems have wide applications from military security systems to civilian daily life. In this research, different application problems, including pattern recognition, classification, image recovery, and sequence learning, are presented to show the capability of the proposed systems in learning, memory, and prediction.

Approved:

Janusz A. Starzyk

Professor of Electrical Engineering and Computer Science

Acknowledgements

I would like to take this chance to express my deepest gratitude to many people's support and help for my doctoral study in Athens.

Firstly, I would like to thank my advisor, Professor Janusz A. Starzyk, for all his supervision and advice for my research. He introduced this amazing research topic to me, and after all these years of study and research in this area, I find that nothing would be more interesting and rewarding than to devote my entire future career to this field. Throughout my doctoral research, he not only delivered the latest knowledge in this area to me and supervised my dissertation research, but also educated me about how to conduct research and how to be a creative, thoughtful, committed, and high integrity person. All of these have made a remarkable influence on my life, and I firmly believe that I will continue to benefit from these throughout my life. I will cherish forever all the happy time with him, all the meetings, conversations and discussions regarding research as well as other things, and all of these memories will become my lifetime fortune.

Secondly, I want to sincerely thank all my committee members for their suggestions and advice throughout my dissertation research: Professor Jeffrey Dill, Professor Robert Curtis, Professor Savas Kaya and Professor Xiaoping Shen. Their knowledge and comments are very important and helpful to me in developing the research presented in this dissertation.

During my four years of study in Athens, I enjoyed a happy time in our "family", the VLSI and intelligent systems research lab. I would like to thank all my officemates and teammates for their valuable comments on research, and their kind help in my daily life:

Yinyin, Lily, Zhen, Mingwei, and James. We are just like a family, and I will never forget the happy time together.

Last but not least, I would like to extend my deepest gratitude to my beloved parents and sister for all their support and encouragement for me throughout all these years. They are always my strongest support, and I would like to share every single step in my life together with them.

Table of Contents

List of Tables

Table ` Page

List of Figures

Figure Page

List of Abbreviations

AA:	Auto-Associative
AI:	Artificial Intelligence
BAM:	Bidirectional Associative Memories
BDNN:	Bayesian Decision-based Neural Network
BRU:	Bidirectional Routing Unit
CMOS:	Complementary Metal Oxide Semiconductor
CNN:	Cellular Neural Networks
CSU:	Configurable Switching Unit
DNN:	Dual-weight Neural Network
DPN:	Data Processing Network
DV:	Differential Voting
FPGA:	Field-Programmable Gate Array
HA:	Hetero-Associative
HANN:	Hetero-Associative Neural Networks
HSLM:	Hierarchical Sequence Learning Machine
IC:	Integrated Circuit
INOUA:	Input–Output Association
IOA:	Input Only Association
IPN:	Information Processing Network
IR:	Input Register
LFN:	Learning Flag Neuron
LN:	Learning Neuron
LSE:	Least Square Error
LTM:	Long Term Memories
MANN:	Multi-Associative Neural Network
MRA:	Multiresolution Analysis
MUX:	Multiplexer

MWDN:	Multiple Winner Detection Neuron
OOA:	Output Only Association
PCN:	Prediction Checking Neuron
PE:	Processing Elements
PMN:	Prediction Matching Neuron
PN:	Prediction Neuron
P-NET:	Projective Networks
PQ:	Power Quality
RAM:	Random Access Memory
RBF:	Radial Basis Function
RL:	Reinforcement Learning
SCU:	Switch Control Unit
SLC:	Switched Inductor-Capacitor
SNR:	Signal to Noise Ration
SOAM:	Self-Organizing Associative Memory
SOLAR:	Self-Organizing Learning Array
SOM:	Self-Organizing Map
SS:	Signal Strength
STM:	Short-Term Memory
SVM:	Support Vector Machine
SWTA:	Stiff Winner Take All
TCF:	Three Curve Fitting
TP:	Time Pointer
VLSI:	Very Large Scale Integration
WTA:	Winner Take All

Chapter 1

Introduction

1.1 Motivation

With the development of deep submicron very large scale integration (VLSI) systems, nanotechnology and bioinformatics, it is possible to build complex dynamically self-reconfigurable systems for machine intelligence. However, there is still no clear picture about how to design the truly brain-like intelligent machines. The biggest challenge comes from how to develop the system level models and architectures that are able to learn, process information, make associations and predictions to accomplish desired tasks, which are the critical elements for any biological intelligent systems.

Traditional structures for artificial intelligence, for instance, neural networks, were introduced with the hope to model functions of the human brain. However, very little progress was made towards implementation of structures and distributed learning algorithms, which would yield essential features of neural processing observable in a living brain. People agreed that human brains and computers use fundamentally different ways in handling various problems [Haw 04]. That is why a three-year old baby can easily watch, listen, learn, and remember various external environment information and adjust her behavior, while the most sophisticated computer can not. In such sense, modern computers are just computational machines without intelligence. The question is: how does one develop the system level models and architectures that are able to mimic the brain intelligence?

Recently, a new theoretical framework titled “memory-prediction” was proposed for designing real intelligent machines by J. Hawkins [Haw 04]. This theoretical framework provides a new solution for how to understand the memory and prediction, which are widely considered to be critical elements of human intelligence. In this dissertation, “learning” is added to this framework to present a complete picture for designing brain-like machine intelligence. While [Haw04] focused on understanding the intelligent framework based on the neocortex, the “learning-memory-prediction” framework proposed in this dissertation research is focused on developing system level models and architectures for dynamically self-reconfigurable systems that are able to mimic the brain-intelligence. The major advantage of such systems over the traditional hardware with embedded software systems is that they can dynamically self-reconfigure their structures in interaction with the external environment to learn, remember and predict information, therefore guiding the machine to achieve its goals. These kinds of systems have wide applications including dynamically adaptive control, intelligent robotics, automatic target recognition and tracking, image and signal processing, biological research, data mining, and decision-support systems.

Another motivation for designing brain-like intelligent systems comes from the development of the reconfigurable computing and VLSI technologies. With the capability of fabricating millions of transistors in a tiny silicon area, it is technologically possible for engineers to implement an integrated “silicon brain”. Therefore, it is critical to integrate knowledge from neuroscience, bioinformatics and engineering design to develop a truly intelligent machine. To this end, we need to develop the system level models and

architectures that not only can learn, remember and anticipate, but are also hardware oriented, fault tolerant and scalable to large systems.

1.2 Related work: learning, memory and prediction

As one of the most sophisticated, challenging, rewarding and significant interdisciplinary research areas, designing of intelligent systems has attracted extensive attention both in science and engineering. In this section, I will give a brief review of the state-of-the-art research achievements for understanding, modeling and designing of intelligent systems in learning, memory and prediction.

The ability of self-organization is critical for intelligent systems to learn and process information from the external environment. A self-organizing learning array (SOLAR) system was proposed by J. A. Starzyk [Sta 05a]. As a parallel learning architecture, SOLAR provides a general learning mechanism with sparsely, adaptive interconnections, data-driven learning, and multilayer structures. Neurons in SOLAR exchange information during learning, and have the ability to self-organize by adapting their functionality and interconnection weights in response to the information received.

The SOLAR structure proposed in [Sta 05a] is a supervised learning mechanism and shows solid performance for classification problems. However, the intelligent systems should also be able to learn without supervision, which means that there is no instructor telling the machine what it should or should not do. The intelligent machine should learn through its experience interacting with the external environment in order to maximize the reward for its

actions and thus achieve its goal. Reinforcement learning (RL) is this type of un-supervised learning that learns by optimizing the expected reward it receives when interacting with the environment [Sut 98]. Value system is an important element for the RL. The machine should have an internal biologically plausible organization that streamlines reward and punishment signals from environment to build its internal value system, is able to evaluate signals coming from environment according to its value system, and plans its actions to optimize the expected rewards. Although it is very important to estimate the value signal accurately and dynamically, it is very difficult to do so in complex, uncertain environment for the following reasons:

- Limited availability of information;
- Time variation of the information included in data;
- High information redundancy;
- High dimensionality of the data set;

Many results for using value systems in reinforcement learning have been reported in recent literature. For instance, [Bab 03] proposed a fuzzy-based navigation system for two mobile robots using distributed value function reinforcement learning. This approach enables the robots to learn a value function, which estimates the future rewards for both robots. In this way, cooperation between the two robots is maintained and each robot learns to execute the actions that are good for both of them. In [Hua 02b], a robotic value system was proposed by integrating novelty and reinforcement learning. This system includes sensory inputs, a cognitive mapping module, a value system, and an action network. In this dissertation research, a value system for machine learning is investigated. The proposed value system is

an online dynamic network, which can dynamically estimate the value of multidimensional data set.

The SOLAR structure implemented in [Sta 05a] utilizes a feedforward mechanism for the machine to learn the information from environment. This concept is further advanced in this dissertation research to include both feedforward and feedback mechanisms in hierarchical organization to learn and memorize information: self-organizing associative memory (SOAM). This introduced the second element of the proposed framework: associative memory.

Associative memory is of critical importance for human intelligence. There are two types of associative memories: hetero-associative (HA) and auto-associative (AA) memory. HA memory makes associations between paired patterns, such as words and pictures, while AA memory associates a pattern with itself, recalling stored patterns from fractional parts of the pattern as in image recovery. Due to the importance of the associative memory for machine intelligence, such systems have been extensively studied. Among HA studies, Salih et al. proposed a new approach for bidirectional associative memories (BAM) using feedback neural networks [Sal 00]. The perceptron training algorithm was used to solve a set of linear inequalities for the BAM neural network design. In [Cha 03], Chang et al. proposed adaptive local training rules for second-order asymmetric bidirectional associative memory. In [Wan 99], Wang presented a multi-associative neural network (MANN) and showed its application to learning and retrieving complex spatio-temporal sequences. Simulation results show that this system is characterized by fast and accurate learning, and has the ability to store and retrieve a large number of complex sequences of spatial patterns. Hopfield's paper [Hop 82]

is a classic reference among AA studies. Since that paper, many research results have been reported. Vogel presented an algorithm for auto-associative memory in sparsely connected networks [Vog 98]. The resulting networks have large information storage capacities relative to the number of synapses per neuron. Vogel et al. derived a lower bound on the storage capacities of two-layer projective networks (P-nets) with binary Hebbian synapses [Vog 97]. It is reported that given a 1% tolerance for activation of spurious neurons, the P-net with 1000 synapses per neuron may store more than 1.5×10^6 training vectors with 20 active neurons per vector. In [Wu 00], an efficient learning algorithm for feedforward associative memory is proposed. This memory uses a winner take all (WTA) mechanism and involves a two-layer feedforward neural network. Recently, Wang et al. proposed an enhanced fuzzy morphological auto-associative memory based on the empirical kernel map [Wan 05].

It is believed that the human brain employs both hetero-associative and auto-associative memory for learning, action planning, and anticipation [Riz 01] [Bro 95] [Mur 97]. The memory formed in the human brain is self-organized and data driven. Self-organization is responsible for formation of hierarchically organized interconnected structures not only in the human brain but also in the nervous systems of lower vertebrates [Mal 03]. In this research, a self-organizing associative memory capable of both hetero-associative and auto-associative learning is designed and analyzed. The SOAM is a hierarchically organized associative memory, with sparse and local connections, self-organizing processing elements (PE), and probabilistic synaptic transmissions. Initially, all PEs in the network are identical and record their input data distribution using probability estimators. Using associations observed in the input data, the network self-organizes defining

transfer functions of its PEs. Both feedforward and feedback mechanisms are used to transform signals and make associations.

After the information is stored in the memory, the prediction mechanism is introduced for the intelligent systems to be able to learn and predict temporal sequences. Temporal sequence learning is presumably among the most important components of human intelligence, as most human intelligence behaviors are in the sequential format. For instance, the natural language processing, speech recognition, reasoning and planning are all based on sequential memory. Over the past decade, models and mechanisms for temporal sequence learning have attracted considerable attention. R. Sun and C. L. Giles offer a useful review of the characteristics, problems, and challenges for sequence learning from recognition and prediction to sequential decision making [Sun 01].

D. Wang and M. A. Arbib proposed a complex temporal sequence learning model based on short-term memory (STM) [Wan 90a]. In their paper, two special types of neurons are proposed for complex temporal sequence learning. The first one is the dual neuron, which is used to store a signal for a short period of time. Unlike the traditional binary signal values used in many neural network models, the output of a dual neuron is a graded signal. The second neuron is the sequence detecting neuron. After learning, this sequence detecting neuron fires in response to the previous sequence of patterns, not just the previous pattern. It thereby overcomes a limitation of networks that cannot reliably recall sequences that share common patterns. The same authors presented a framework of learning, recognition and reproduction of complex temporal sequences in [Wan 93]. In this model, sequences are acquired by the attention learning rule, which combines Hebbian learning and a normalization

rule with sequential system activation. Time intervals between sequence components do not affect recognition. A global inhibitor is proposed to enable the model to learn context lengths required to disambiguate associations in complex sequence reproduction. In order to overcome the capacity limitation of STM, a hierarchical sequence recognition model based on the chunking notion is proposed in [Wan 93]. For instance, in a letter-word-sentence hierarchical structure, a unit for a given word is activated at the end of the presentation of that word, and the model learns the sequence of letters of that word based on the letter units active at that time. Once the word structure has been learned, the same mechanism can be applied to train a higher hierarchical level on the sequence of words. One more issue addressed in [Wan 93] is interval maintenance. This is achieved by coding intervals by connection weights from the detector layer to the input layer.

In [Wan 95], a neural network model capable of learning and generating complex temporal patterns by self-organization was proposed. This model actively regenerates the next component in a sequence and compares the anticipated component with the next input. A mismatch between what the model anticipates and the actual input triggers one-shot learning. Although the anticipation mechanism improves the learning efficiency of this model, it needs several training sweeps to learn a sequence. As the number of training sweeps required depends on the degree of complexity of the sequences, it is not very efficient for highly complex sequences.

Another paper [Wan 96] focused on the learning of the multiple temporal sequences and “catastrophic interference.” It is showed that the anticipation model proposed in [Wan 95] is capable of incremental learning with retroactive interference but without catastrophic

interference. In addition, a chunking mechanism was included in this model to detect repeated subsequences between and within sequences, thereby substantially reducing the amount of retraining in sequential training.

L. Wang proposed the use of associative memory to learn and retrieve spatio-temporal sequences in [Wan 98] and [Wan 99]. In [Wan 98], the associative memory model includes three major components: a voting network, a parallel array of hetero-associative neural networks (HANN), and delayed feedback lines from the output of the system to the associative neural network layers. The delayed sequence of hetero-associators “votes” on the next output at each time step. After learning, the system can retrieve the entire sequence from a small cue sequence. Since the model in [Wan 98] assumes that each HANN only learns hetero-associations between single spatial patterns and does not learn to associate multiple patterns in groups, this model was further extended to include associations of one pattern with multiple patterns in [Wan 99]. The proposed model in [Wan 99] has the advantages of short learning time and accurate retrievals, and the ability to store a large number of complex sequences. However, in both of the models the required number of networks (HANNs in [Wan 98] and MANNs in [Wan 99]) for learning and retrieving a sequence significantly increases with the complexity of the sequence.

In [Wan 01], a dual-weight neural network (DNN) scheme for fast learning, recognition, and reproduction of temporal sequences was developed. In a DNN, each neuron is linked to other neurons by long-term excitatory weights and short-term inhibitory weights. Fast learning is achieved by employing a two-pass training rule to encode the temporal distance between two arbitrary pattern occurrences. Based on this, the DNN was extended to

a more generalized model, DNN2, which incorporates a self-organizing algorithm. In this way, the DNN2 can achieve autonomous temporal sequence recognition and reproduction.

A hierarchical structure is important for sequence learning. For instance, in [Man 97], a linear-time algorithm called SEQUITUR was proposed for identifying hierarchical structure in sequences. The main idea of this algorithm is that phrases which appear more than once can be replaced by a grammatical rule that generates the phrase, and that this process can be continued recursively, producing a hierarchical representation of the original sequences. Although [Man 97] is not focused on sequence retrieval, it presents the idea of hierarchical representation of sequences, which is used in many research efforts that model sequence learning. For instance, D. George and J. Hawkins [Geo] discussed the problem of hierarchical structure for temporal sequence learning targets on invariant pattern recognition. It is concluded in [Geo] that the neocortex solves the invariance problem in a hierarchical structure. Each region in the hierarchy learns and recalls sequences of inputs, and temporal sequences at each level of the hierarchy become the spatial inputs to the next higher regions.

In this dissertation research, an anticipation based sequence learning neural structure is proposed. Hierarchical organization, prediction mechanism, and one-shot learning characterize this model. In the lowest level of the hierarchy, a modified Hebbian learning mechanism is used for pattern recognition. Prediction is an essential element of this temporal sequence learning model. By correct prediction, the machine indicates that it knows the current sequence and does not require additional learning. When the prediction is incorrect, one-shot learning is executed and the machine learns the new input sequence as soon as the sequence is completed.

1.3 Significance of Research

Significance of the research presented in this dissertation stems from the following needs and observations:

- It is important to understand how a brain-like complex system can be self-organized to develop its connectivity structure, build the mechanisms for association and anticipation, interact with external environment, accumulate knowledge through experience, formulate goals for its actions, and self-control to achieve its goals. To this end, the proposed “learning-memory-prediction” framework provides a foundation for designing the intelligent systems with such capabilities;
- It is critical to design system level models and architectures that are able to mimic such intelligent mechanisms. The proposed dynamic value learning system, self-organizing associative memory, and anticipation based sequence learning network provide a solution for designing necessary elements of such brain-like intelligent systems. In the proposed systems, the complexity of the network organization is self-decided by the system according to the information it receives, rather than by the arbitrary choice of a designer; as is the case of artificial neural networks. These characteristics make the proposed models in this research more like a “living-brain”; it can grow in complexity and ability to model external environment according to the information it receives, and it can accumulate such

knowledge. This is a fundamental difference in comparison to the fixed or pre-defined neural network structure;

- It is essential to develop new computing paradigm that can achieve extremely low power consumption for hardware implementation of such large scale integrated intelligent systems. Without careful consideration of the low power consumption, the designed intelligent system will not be able to function properly. Therefore, the proposed low power design scheme in this dissertation provides a possible solution for the future implementation of such large scale high density intelligent systems;

1.4 Research goals and objectives

As most of the state-of-the-art research in the effort of designing intelligent systems is focused on individual elements for learning, memory, and prediction, it is the goal of this dissertation research to develop mechanisms and organization for elements of intelligence and to provide a framework to integrate these individual elements together with the aim of designing brain-like intelligent systems.

To accomplish this research goal, I will focus on the designing of system level models and architectures for building dynamically self-reconfigurable systems for machine intelligence. The underlying foundation of building such brain-like intelligent systems is based on the proposed “learning-memory-prediction” framework. It is the objective of this research to justify such an integrated theoretical framework for designing real intelligent systems, and under this foundation, to develop the network models and architectures that:

- Are able to self-organize their connectivity structure and dynamically reconfigure to learn, process information, accumulate knowledge, and make associations and predictions;
- Are able to learn without a supervisor, learn through interaction with environment without software rules or a global learning algorithm, and learn through experiences to accomplish desired tasks;
- Are scaleable to very large systems, and are fault tolerant and modular for easy implementation in hardware, such as reconfigurable computing field-programmable gate array (FPGA) technology, dedicated analog or digital VLSI technology, or future system level technology (modular blocks represented by SOLAR systems or nanotechnology).

1.5 Dissertation Organization

Figure 1-1 illustrates the overall organization of this dissertation.

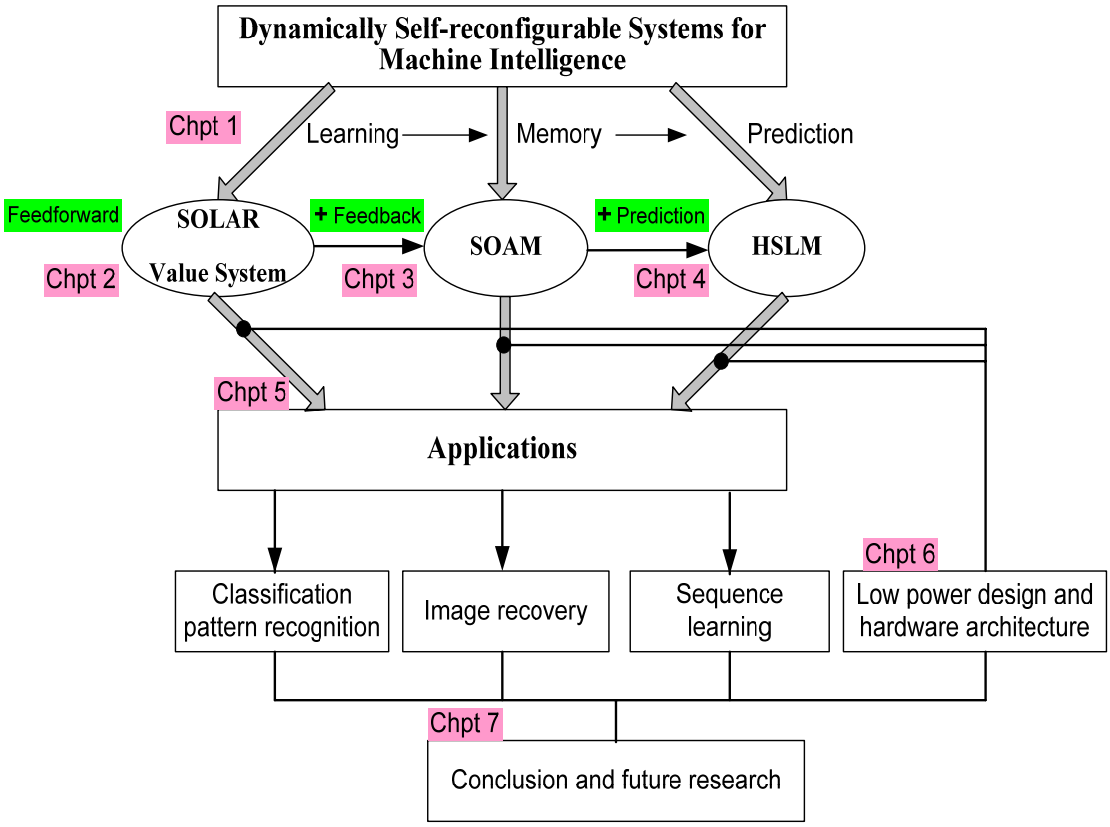


Figure 1-1 Dissertation organization

Chapter 1 presents the motivations, significance, and objective of this research. This chapter also provides a brief review of the state-of-the-art work in the area of learning, memory, and prediction. In addition, this chapter provides an overview and organization of the whole dissertation.

Chapter 2 focuses on the learning scheme. A brief review of the SOLAR and reinforcement learning is given in this chapter. It introduces one of the critical elements in the learning procedure: value learning system. A novel architecture of the dual-network online value system is proposed in this chapter, and this system can dynamically estimate the value of incoming multidimensional data sets from external environment. In the SOLAR learning system, information is only propagated forward, and this is not sufficient for the intelligent

systems to select, evaluate, and memorize useful information. Therefore, a feedback mechanism is introduced in the self-organizing network and this leads to the organization of SOAM in Chapter 3.

Chapter 3 discusses in detail the proposed self-organizing associative memory architecture, which is a hierarchically organized memory network with sparse and local interconnections. Both feedforward and feedback mechanisms are used to transform signals and make associations. This associative memory provides a mechanism for the intelligent system to be able to associatively recover a complete picture from partial information (auto-association), or associate one piece of information with another one (hetero-association). After the information is stored in the memory structure, the intelligent systems should be able to make predictions to learn and retrieve temporal sequences. This leads to the hierarchical sequence learning machine (HSLM) introduced in Chapter 4.

Chapter 4 focuses on the system level architecture of the proposed temporal sequence learning neural network. This network is hierarchically organized and is efficient in both learning effort and storage capacity. At each hierarchical level, a winner-take-all structure is used to select the firing neurons at this level, and the firing neurons in one level provide the input to the neurons on the next hierarchical level. A prediction mechanism is proposed in this model for efficient anticipation based learning. As long as all predictions are correct, no learning takes place. Otherwise, a learning signal is issued at the end of a sequence and a one-shot learning is executed to learn the new sequence.

Chapter 5 discusses different application problems for the proposed dynamically self-reconfigurable systems. The application areas studied in this dissertation include pattern

recognition and classification, image recovery, and temporal sequence learning. Through these application problems, it is clearly shown that the proposed models have the abilities to learn, process information, and make associations and predictions to solve practical problems.

Chapter 6 discusses the low power design of the large scale, high density intelligent systems. Power consumption will be a major design issue for such complex dynamically reconfigurable systems. In this chapter, a novel switched inductor-capacitor (SLC) architecture is proposed to reduce the dynamic power consumption. Finally, a brief description of the hardware architecture based on the reconfigurable FPGA technology is presented in this chapter. This FPGA system will provide a platform for prototyping and testing of the proposed models and architectures in this dissertation.

Chapter 7 summarizes the whole dissertation work, restates the novelty and original contributions of this dissertation work, and discusses future research directions.

Chapter 2

Value System for Machine Learning

2.1 Introduction

A new machine learning concept, the self-organizing learning array was recently proposed in [Sta 05a]. As a general learning mechanism, SOLAR has several fundamental advantages over the traditional neural network based learning scheme, including online data-driven learning, local and sparse interconnection, and entropy based learning. While the learning in Kohonen's self-organizing map (SOM) [Koh 95] is performed in the entire input data space, individual neurons in SOLAR learn information from their local input space. In addition, the interconnect structure in SOLAR can be dynamically reconfigured according to the information each neuron receives in the learning process. This provides SOLAR with more flexibility in the learning process compared to the fixed interconnection structure (defined by templates) in cellular neural networks (CNN) [Man 99].

The SOLAR architecture presented in [Sta 05a] uses supervised learning, which means a teacher (instructor) is needed to tell the machine what is good and what is bad. However, a supervisor is not always available in many learning environments [Sut 98]. For instance, when a one-year old baby stands alone in a baby-cart and plays with toys, turns around and looks at the fantasy world around him, there is no explicit teacher telling him about the information he is observing, processing and perceiving. However, learning does occur in this process. By the interaction with the external environment, the baby can process

information it receives, adjust its behavior and learn how to achieve his goals [Haw 04]. This kind of learning is a combination of reinforcement learning [Sut 98], anticipation, and novelty detection.

Learning from active interaction with the external uncertain environment is the foundation of reinforcement learning [Sut 98], which is also believed to be the foundation of learning for most of the biologically intelligent systems. People learn through interacting with the external environment, accumulate knowledge through experiences, and adjust behavior (action) to achieve their goals (maximize the rewards signal). A typical reinforcement learning system includes the external environment, a policy, a reward function, and a value function. Among all these elements, R. S. Sutton presented that value function is of critical importance and almost all reinforcement learning algorithms are based on estimating the state-action values [Sut 98]. Based on the information an intelligent system receives from the environment, it should be able to evaluate the signals according to the value system, and adjust its actions to maximize its rewards.

Although it is very important to estimate the value signal accurately and dynamically, it is very difficult to do so in practical applications. In this chapter, an online value system is proposed to dynamically evaluate the value signal for multi-dimensional data sets. Dynamic curve fitting is the core module for this value learning system. Based on this scheme, a three curve fitting (TCF) method is proposed to improve the fit to a statistically distributed high dimensional data set. The proposed value system model has a pipelined architecture and uses two networks: a data processing network (DPN) and an information processing network (IPN).

2.2 Online dynamic curve fitting

Consider dynamic adjustment of the fit function described by a linear combination of the selected base functions $\varphi_i, i = 1,2,...q$, where q is the number of base functions. The objective of the fit function is to fit values from the received data samples. Assume that each processing element will dynamically adjust its fit function to minimize the least square error (LSE) of the function approximating values of all training data x and y as follows:

$$Y = a_1 * \varphi_1 + a_2 * \varphi_2 + ... + a_q * \varphi_q \tag{2.1}$$

where $\varphi_i, i = 1,2,...q$, is the base function. The number of base functions can be adjusted according to the accuracy required and the data noise level.

Using the least square solution, the coefficients $a_1, a_2, ..., a_q$ can be determined by pseudo inversion. To do this dynamically, it is necessary to accumulate function values and their combinations for different input samples. Equation (2.1) can be represented as follows:

$$Y = \begin{bmatrix} \varphi_1 & \varphi_2 & ... & \varphi_q \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ ... \\ a_q \end{bmatrix} = \Phi * A \tag{2.2}$$

then the coefficients of the approximating function are obtained from

$$\begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_q \end{bmatrix} = \left(\Phi^T \Phi\right)^{-1} \Phi^T Y$$

$$= \begin{bmatrix} \sum_{i=1}^{n} \Phi_{1i}\Phi_{1i} & \sum_{i=1}^{n} \Phi_{1i}\Phi_{2i} & \dots & \sum_{i=1}^{n} \Phi_{1i}\Phi_{qi} \\ \sum_{i=1}^{n} \Phi_{1i}\Phi_{2i} & \sum_{i=1}^{n} \Phi_{2i}\Phi_{2i} & \dots & \sum_{i=1}^{n} \Phi_{2i}\Phi_{qi} \\ \dots & \dots & \dots & \dots \\ \sum_{i=1}^{n} \Phi_{1i}\Phi_{qi} & \sum_{i=1}^{n} \Phi_{2i}\Phi_{qi} & \dots & \sum_{i=1}^{n} \Phi_{qi}\Phi_{qi} \end{bmatrix}^{-1} * \begin{bmatrix} \sum_{i=1}^{n} \Phi_{1i} Y_i \\ \sum_{i=1}^{n} \Phi_{2i} Y_i \\ \dots \\ \sum_{i=1}^{n} \Phi_{qi} Y_i \end{bmatrix} \tag{2.3}$$

where n is the number of data points. For online implementation, this requires storage of $s = \frac{q(q+1)}{2} + q$ values of different combinations in equation (2.3)

$$\begin{cases} \sum_{i=1}^{n} \Phi_{ki}\Phi_{mi} \\ \sum_{i=1}^{n} \Phi_{ki} Y_i \end{cases} \quad where\ k, m = 1, \dots, q \tag{2.4}$$

As new samples arrive, these s values are updated, and equation (2.3) is solved for new coefficients $a_1, a_2, \dots, a_q$. In general, for q base functions one may need to store s combinations and invert $q \times q$ matrix $(\Phi^T \Phi)$ to update coefficients of the approximating equation.

2.3 Three curve fitting and voting scheme

Further investigation of the above online curve fitting scheme shows its limitations. This is illustrated in Figure 2-1, which gives a general idea of a single curve fit to noisy data

by individual PE. For noisy data values, the single curve fitting technique can not reflect the statistical distribution of the data values in areas A and B, which will cause poor value fitting in these areas. One can compute a standard deviation of the approximated data from the curve fit, but this only gives a uniform measure of statistical errors that does not reflect the different quality of approximation in different regions of the input space.

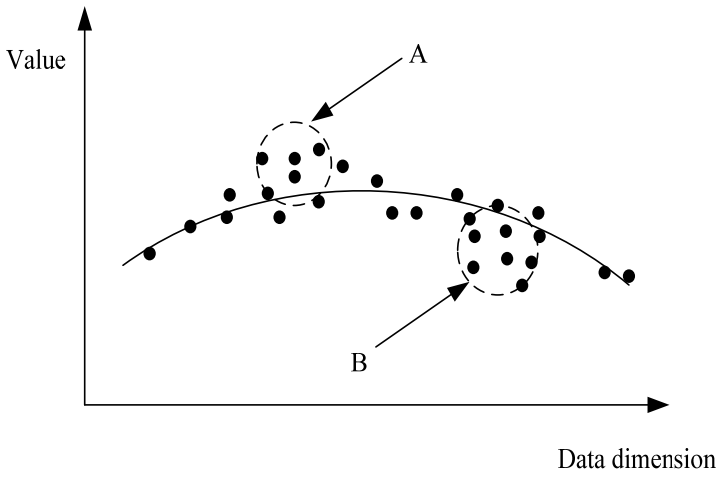


Figure 2-1 Single curve fitting

In order to overcome this limitation, a three curve fitting and differential voting (DV) scheme are proposed. Figure 2-2 shows the concept of the TCF scheme.

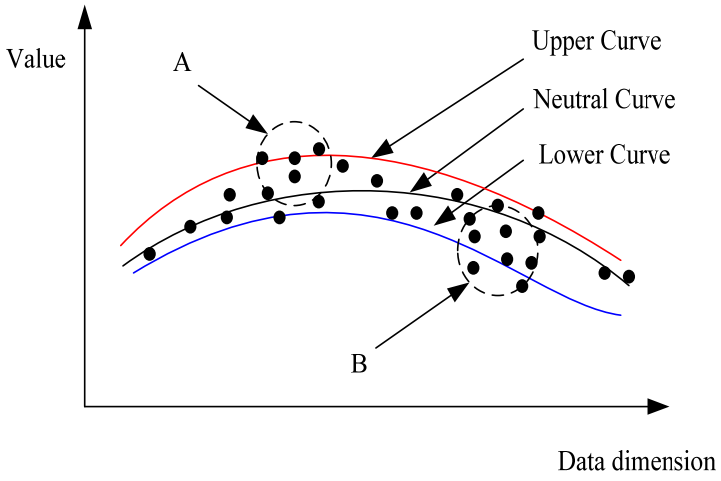


Figure 2-2 Three curve fitting (TCF) scheme

For the same data distribution as in Figure 2-1, there are three curves fitting to these data samples values in the following way:

Neutral Curve: it fits to all the data samples in the input space, and is the same as the curve in Figure 2-1.

Upper Curve: it only fits to the data points which are above the neutral curve.

Lower Curve: it only fits to the data points which are below the neutral curve.

As we can see from Figure 2-2, the neutral curve can provide a rough estimation of the fitted value, and the upper and lower curves can provide its statistical distribution information. Therefore, these upper and lower curves characterize a statistical deviation of the approximated data from the least square fit value. Next, it will be shown how to use this information to get the final estimated value by the differential voting.

Assume that there are total k processing elements in the network. During training, based on the incoming data samples, each processing element will update its own coefficients of the three curves. During testing, when one testing sample is input to the network, each

processing element will provide the estimated value for this sample based on the three fitting curves built during the training stage. Figure 2-3 shows the results of PE i when the incoming samples are fitted, where v_{ui} is the value estimated by the upper curve, v_{ni} is the value estimated by the neutral curve, and v_{li} is the value estimated by the lower curve. In order to reflect the statistical distribution characteristics of the data points, the standard deviation of the estimated value is defined in the following way.

$$d_{1i} = |v_{ni} - v_{ui}|$$

$$d_{2i} = |v_{ni} - v_{li}| \tag{2.5}$$

$$d_i = \frac{d_{1i} + d_{2i}}{2}$$

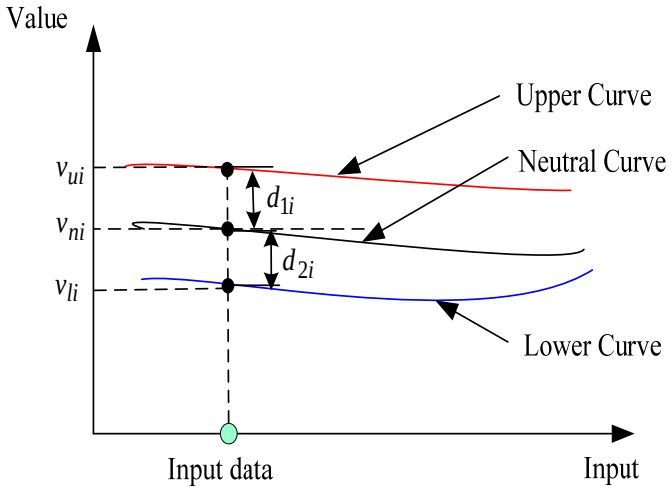


Figure 2-3 Three curve fitting results

Each d_i reflects how good the estimated value v_{ni} computed by its PE is compared to its true value. Small values of d_i mean the estimated value v_{ni} is obtained with greater confidence and should carry higher weight in the final voting scheme. On the other hand, when d_i is large, it means that the estimated value v_{ni} is not so good and

should contribute less to the final voting result. Based on this, the weights for each PE is derived by the following equation (2.6)

$$w_i = \frac{1}{d_i} \tag{2.6}$$

so the final value in differential voting is obtained by

$$v_{vote} = \frac{\sum_{i=1}^{k} (v_{ni} w_i)}{\sum_{i=1}^{k} w_i} \tag{2.7}$$

2.4 System architecture

Figure 2-4 shows the system architecture of the proposed dual network value system. This system has two network structures: a DPN and an IPN. The DPN is responsible for the incoming numerical data processing, including input data space transformation and online dynamic data fitting. The IPN is in charge of the final evaluation of the results provided by DPN. Each data processing element will conduct the three curve fitting scheme as discussed in Section 2.3, and will output the fitted values of v_{ni}, v_{ui} and v_{li} for information processing elements. These values will provide a rough estimation of the true value as well as its statistical distribution information. Based on this information, each information PE will vote on the final value based on the DV scheme as presented in equations (2.5) ~ (2.7).

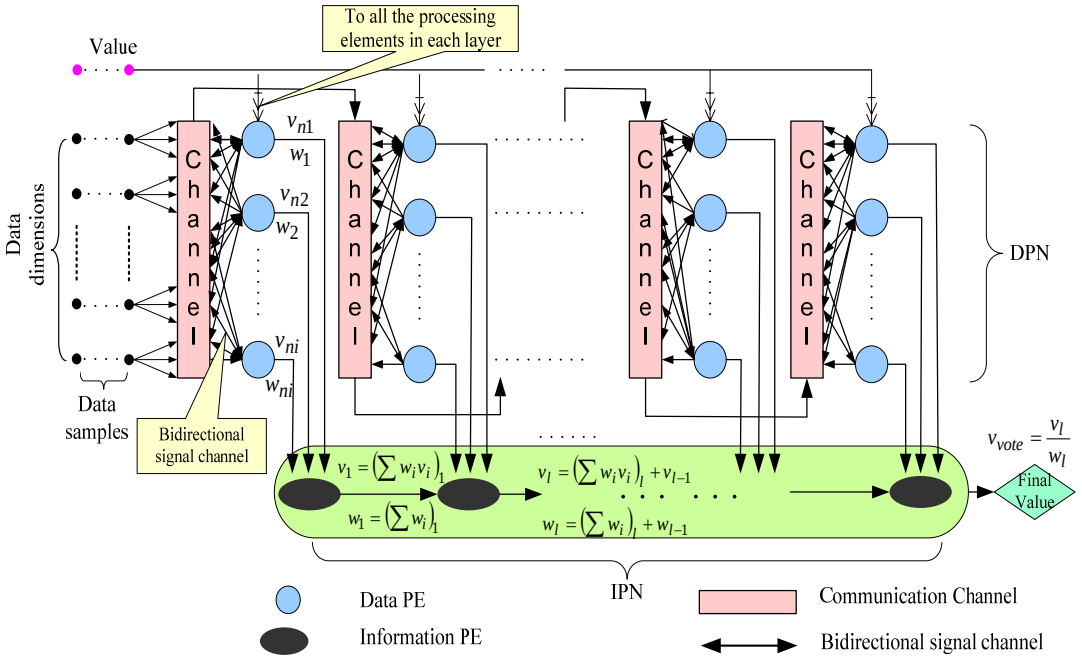


Figure 2-4 Pipelined value system architecture

A novel communication channel structure based on the shift-register is presented in Figure 2-4. Each data PE has a set of inputs, and they are pseudo randomly connected to their local channel. The PEs are more likely to be connected to their local neighbors, although some of them may still reach distant PEs.

This architecture processes the information in a way similar to the pipelined shift-register structure. In the first clock cycle, the data is available in the first layer channel, and the first layer data PEs will read this data as their inputs. After processing, PEs will output the transformed data into the same location in the input channel. At the same time, they will also output their estimated value v_{ni} and its corresponding weight w_i to the information PE in IPN network. The information PE will combine these local values and their weights according to the following equations and pass them to the next layer of information PE at the next clock cycle.

$$v_l = \left(\sum w_i v_i\right)_l + v_{l-1} \tag{2.8}$$

$$w_l = \left(\sum w_i\right)_l + w_{l-1} \tag{2.9}$$

here the subscript "l" means the information from channel layer l. Therefore, v_l and w_l are the combined value and weight information for layer l.

When the next clock cycle arrives, the transformed data (the output data of the PEs in the first layer channel) is shifted to the next layer channel as the input data to the data PEs in the second layer, while another set of input data samples can be sent to the first layer channel. Meanwhile, the information PEs in the second layer will combine the information from the second layer with that passed from the previous layer.

All processing elements in the system are active during all clock cycles making this architecture suitable for the dynamic online processing. Finally, when the data reaches the final layer, the final voted value is computed as

$$v_{vote} = \frac{v_l}{w_l} \tag{2.10}$$

Figure 2-5 shows the local organization and connectivity structure of one data processing element to illustrate how the individual data PE works in the system. During the training stage, the input multidimensional data samples and their corresponding true values are provided as inputs to the local channel. Each PE will use a set of input space transform functions to combine the information from different inputs. Based on the proposed curve fitting technique, each PE will modify its curve fitting coefficients every time a new sample is received. Each PE will also output the numerical value obtained by the input space transformation function. This output will be used as the input of the data processing element

in the next layer after shifting is performed in the routing channel. After training, each data PE has its TCF coefficients determined based on the characteristics of training information.

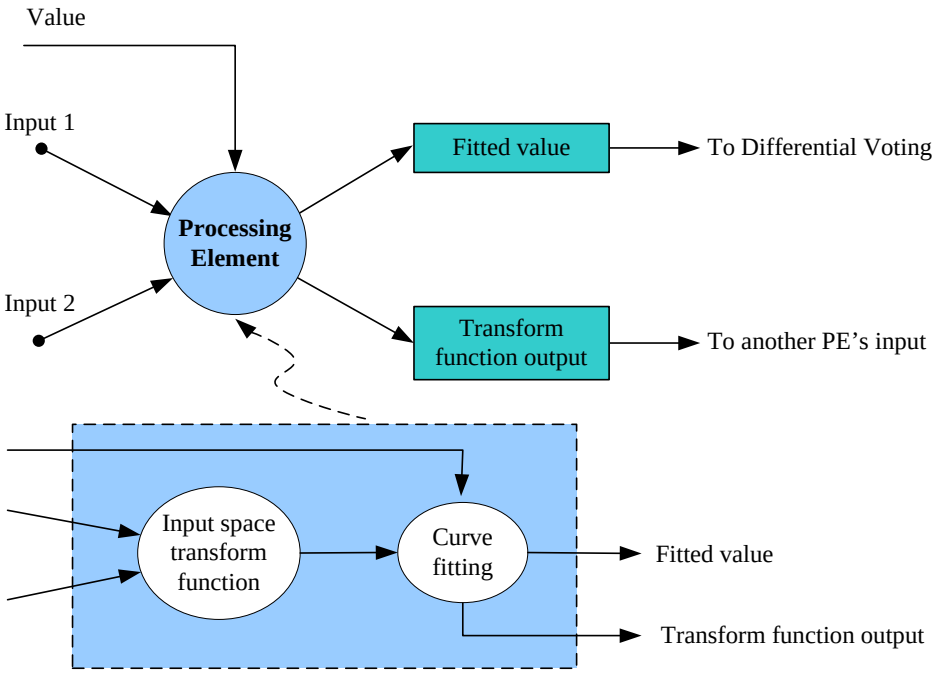


Figure 2-5 Detailed structure of data PE

In the testing stage, when a testing sample is applied to the inputs of the first layer processing elements of the value system, each data PE will output the three fitted values v_{ni}, v_{ui} and v_{li} according to its own fitting curves. All of this information will go to the information processing network for final voting based on the differential voting scheme.

2.5 Summary

This chapter presents a novel value system for machine learning. The ability to learn is the foundation of the biological intelligent systems, and forms the first element of the proposed "learning-memory-prediction" framework for machine intelligence. In a supervised

learning process, the machine learns the information from a knowledgeable teacher with an explicitly stated goal. However, supervisory learning is not always available, nor is it necessary for biological intelligent systems to learn and process information. Reinforcement learning does not have the requirements of supervised learning. In RL, the machine learns through interaction with the external environment, accumulates knowledge and plans its action to maximize the reward signal and to achieve the goal of its operation. Value system is one of the critical elements for RL. Based on the dynamic curve fitting technology, the proposed value system can accurately and dynamically estimate the value signal for multi-dimensional data set. This value system has two networks (DPN and IPN), and has an efficient pipelined architecture suitable for hardware implementation.

In the value system, signals only propagate forward for the machine to learn the information received from environment. However, this is not enough for building intelligent systems. Feedback is introduced in the learning mechanism for the machine to be able to remember the information in its distributed elements, which leads to the second element of the proposed framework: associative memory.

Chapter 3

Self-Organizing Associative Memory

3.1 Introduction

Biological intelligent systems use a different memory organization than that of digital computers. Their memory is characterized by associations and self-organization, is data driven, has hierarchical structure, and uses distributed information storage. This chapter focuses on the second element of the "learning-memory-prediction" framework: associative memory. In this chapter, the feedforward network presented in [Sta 05a] is further advanced to include a feedback mechanism to learn and remember information from the environment [Sta 06b]. This self-organizing associative memory has a hierarchical organization and can self-adjust its association depth according to the complexity of different application problems. In addition, this memory structure is capable of both auto-association and hetero-association, which co-exist in human brain [Riz 01] [Bro 95] [Mur 97].

3.2 Probability based associative learning

3.2.1 Structure of individual processing elements

The proposed self-organizing associative memory consists of a multilayer array of processing elements. Figure 3-1 gives the interface model of an individual PE, which consists of two inputs (I_1 and I_2) and one output (O). All the inputs and the output are bidirectional,

allowing signals to propagate forward and backward. Each PE stores observed probabilities p_{00}, p_{01}, p_{10} and p_{11} corresponding to four different combinations of inputs I_1 and I_2 ($\{I_1 I_2\} = \{00\},\{01\},\{10\},\{11\}$), respectively. These probabilities specify data distribution in each PE's input space, and are used to make associations.

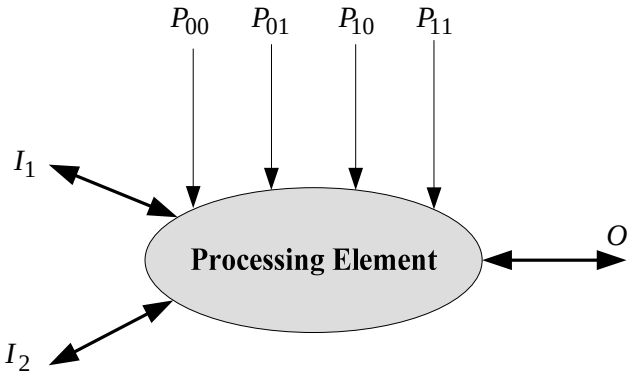


Figure 3-1 Individual PE interface model

Figure 3-2 gives an example of possible distribution of the observed input data points. Probabilities are estimated from $p_{00} = \frac{n_{00}}{n_{tot}}, p_{01} = \frac{n_{01}}{n_{tot}}, p_{10} = \frac{n_{10}}{n_{tot}}, p_{11} = \frac{n_{11}}{n_{tot}}$, where n_{00}, n_{01}, n_{10} and n_{11} are the number of data points located in $I_1 < 0.5 \,\&\, I_2 < 0.5$, $I_1 < 0.5 \,\&\, I_2 > 0.5$, $I_1 > 0.5 \,\&\, I_2 < 0.5$ and $I_1 > 0.5 \,\&\, I_2 > 0.5$, respectively. The value n_{tot} is the total number of data points defined as $n_{tot} = n_{00} + n_{01} + n_{10} + n_{11}$. An efficient algorithm for dynamic estimation of these probabilities without division on unlimited number of input data is proposed in [Sta 04]. Such an estimation is good for on-line learning and hardware implementation. This research takes advantage of the results presented in [Sta 04], and assumes that on-line probability estimation is inexpensive from a hardware point of view.

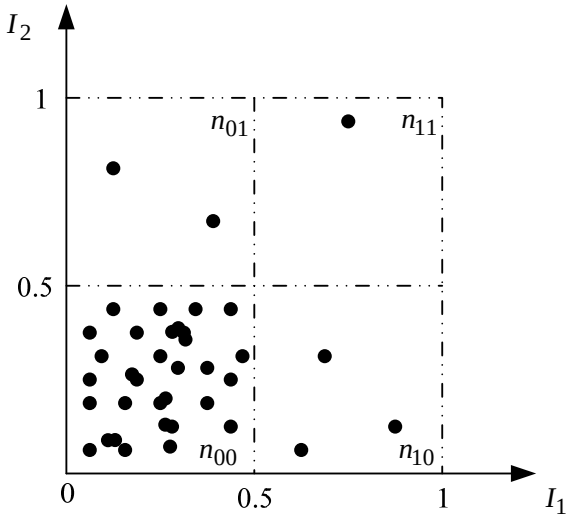


Figure 3-2 An example of input space distribution

3.2.2 Self-determination of the function value

Based on the observed probability distribution p_{00}, p_{01}, p_{10} and p_{11} of an individual PE as in Figure 3-2, each PE decides its output function value F by specifying its truth table as shown in Table 3-1.

Table 3-1 Self-determination of function value F

Probability	p_{00}	p_{01}	p_{10}	p_{11}
I_1	0	0	1	1
I_2	0	1	0	1
Function value	f_{00}	f_{01}	f_{10}	f_{11}

The output function values f_{00}, $f_{01,}$ f_{10} and f_{11} are decided as follows:

(1) The input, (I_1, I_2), that is associated with the largest probability, $p_{ij}, (i, j = 0,1)$, is assigned a corresponding output function F value of 0.

(2) If the largest probability is less than 0.5, then the input (I_1, I_2), that is associated with smallest probability $p_{ij}, (i, j = 0,1)$, is also assigned a corresponding F value of 0;

(3) If the sum of the largest and smallest probabilities is less than 0.5, then the input, (I_1, I_2), that is associated with the second-smallest probability $p_{ij}, (i, j = 0,1)$ is also assigned a corresponding F value of 0;

(4) All input combinations not assigned a corresponding F value of 0 by the above rules are assigned a corresponding F value of 1.

The probability that the neuron is active is smaller than 0.5. This type of assignment is motivated by the sparse activity of biological neurons [Tri 04]. In addition to biological motivation, lower activities are preferable for efficient power consumption. Table 3-2 shows two examples of this self-determination of the function value F.

Table 3-2 Two examples of setting F value

P_{00}	P_{01}	P_{10}	P_{11}	F			
0.4	0.2	0.3	0.1	0	1	1	0
0.4	0.05	0.3	0.25	0	0	1	0

3.2.3 Signal strength

In the proposed associative learning model, outside signals are presented to the network in a binary form. The internal signals have semi-logic values ranging from 0 to 1,

where 0 and 1 correspond to logic false and logic true, respectively. The signal strength is the absolute value of the distance between the signal level and a specified logic threshold ($Th = 0.5$ in the current model; other threshold values can be used as well, particularly if signals are biased towards 1 or 0):

$$\text{Signal strength } (SS) = |\text{Signal value} - \text{logic threshold } (Th)| \quad (3.1)$$

SS is in the range of [0, 0.5]. If $SS = 0.5$, the signal is either determinate high (logic true) or determinate low (logic false), corresponding to signal value 1 or 0, respectively. If signal value equals Th, it is undefined (inactive) and $SS = 0$. Signals in the range $0 < SS < 0.5$ are intermediate. An intermediate signal is weak low if its value is less than Th or weak high if it is higher than Th.

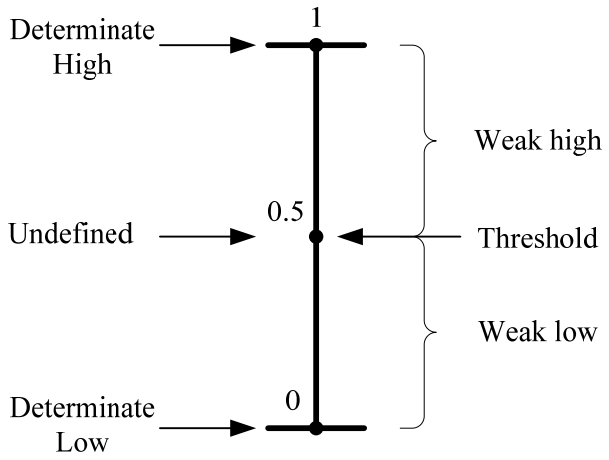


Figure 3-3 Signal strength and its semi-logic value

Figure 3-3 illustrates this definition of the signal strength. SS provides a coherent way of determining when to trigger an association, and helps to resolve the feedback signal value if more than one feedback signals are presented at the PE's output port.

3.2.4 Probability based associative learning algorithm

3.2.4.1 Example PE and its associations

During training, each PE counts its input data points in n_{00}, n_{01}, n_{10} and n_{11} and estimates their corresponding probabilities p_{00}, p_{01}, p_{10} and p_{11}. The objective of the training stage for each PE is to discover the potential relationship between its inputs. This relationship is remembered as the corresponding probabilities and is used to make associations during the testing stage. Consider the example in Figure 3-2, this particular PE finds that most of its input data points are distributed in the lower-left corner ($I_1 < 0.5 \,\&\, I_2 < 0.5$). Assume that the relevant probabilities are $P_{00} = 0.8$, $p_{01} = 0.07$, $p_{10} = 0.1$ and $p_{11} = 0.03$, resulting in $F = \{0, 1, 1, 1\}$. This function value is propagated through the network as input to other PE's and may be used to make associations.

Figure 3-4 illustrates three types of associations used in the testing stage to infer the undefined signal value.

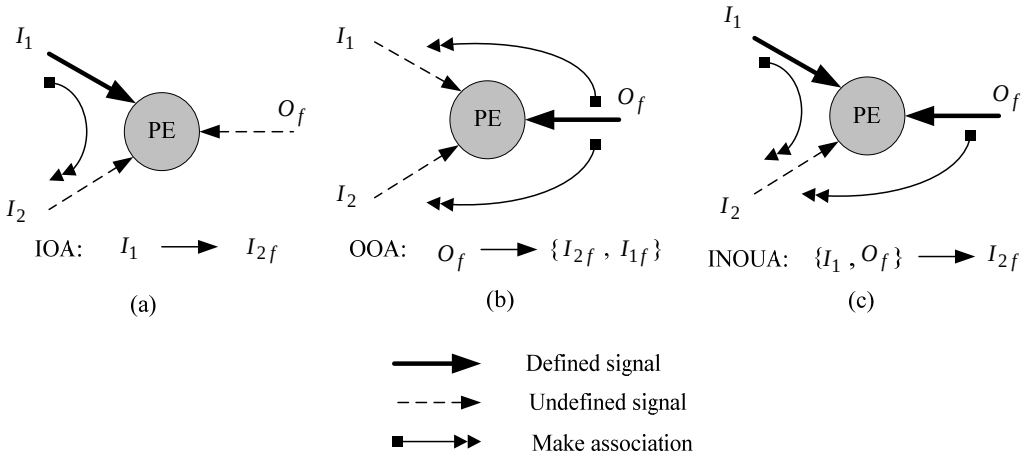


Figure 3-4 Three types of associations of processing element

(1) Input only association (IOA). If, in the testing stage, one input is defined while the other input and the received output feedback signal O_f from other PEs are undefined (for instance, if $I_1 = 0$, $I_2 = 0.5$ and $O_f = 0.5$ as in Figure 3-4 (a)), this PE will determine I_2 through association with I_1, driving I_2 towards logic 0.

(2) Output only association (OOA). If both inputs, I_1 and I_2, are undefined, a defined feedback signal, O_f, will determine both inputs (Figure 3-4 (b)). For instance, if $O_f = 0$, based on PE function F = {0, 1, 1, 1}, then this PE will set both inputs, I_{1f} and I_{2f} to 0. (Here I use I_{1f} and I_{2f} to denote the feedback signals of inputs 1 and 2 to distinguish them from the corresponding feedforward signals). On the other hand, if the received output feedback signal $O_f = 1$, the input feedback values, I_{1f} and I_{2f}, are intermediate and their values will be estimated according to data distribution probabilities.

(3) Input–output association (INOUA). If one input and the output feedback signal, O_f, are defined and the other input is undefined, the PE will set the other input signal according to its observed probabilities, as shown in Figure 3-4 (c).

3.2.4.2 Probability based associative learning algorithm

I will formulate the mathematical description of the probability based associative learning algorithm. There are four cases:

Case 1: Given the semi-logic values of both inputs $V(I_1)$ and $V(I_2)$, decide the output value $V(O)$

Assume one PE received input values $V(I_1) = m$ and $V(I_2) = n$, then

$$V(O) = \frac{p(I_1 = 1, I_2 = 1, F = 1)}{p(I_1 = 1, I_2 = 1)} \bullet V_{11} + \frac{p(I_1 = 0, I_2 = 1, F = 1)}{p(I_1 = 0, I_2 = 1)} \bullet V_{01}$$
$$+ \frac{p(I_1 = 1, I_2 = 0, F = 1)}{p(I_1 = 1, I_2 = 0)} \bullet V_{10} + \frac{p(I_1 = 0, I_2 = 0, F = 1)}{p(I_1 = 0, I_2 = 0)} \bullet V_{00} \tag{3.2}$$

where V_{11}, V_{01}, V_{10} and V_{00} are defined as

$$V_{11} = mn;$$
$$V_{01} = (1-m)n;$$
$$V_{10} = m(1-n); \tag{3.3}$$
$$V_{00} = (1-m)(1-n)$$

and $p(I_1 = 1, I_2 = 1, F = 1)$, $p(I_1 = 1, I_2 = 1)$ etc. are joint probabilities that can be obtained from Table 3-1 using probabilities p_{00}, p_{01}, p_{10} and p_{11}. For instance, if one PE has F = {0, 1, 1, 1}, then

$$p(I_1 = 1, I_2 = 1, F = 1) = p_{11};$$
$$p(I_1 = 0, I_2 = 0, F = 1) = 0; \tag{3.4}$$
$$p(I_1 = 0, I_2 = 1) = p_{01}$$

This case is required when a signal is propagated forward (during both training and testing stages). In the current model, this input and the output semi-logic values are typically 0, 1 or 0.5. However, these equations are still true if other semi-logic values are used.

Case 2: Given the values of one input, $(V(I_1)$ *or* $V(I_2))$*, and an undefined output* $V(O)$*, decide the value of the other input.*

This case corresponds to IOA as shown in Figure 3-4 (a). Consider the situation that given $V(I_1)$ to decide an unknown $V(I_2)$ as follows:

$$V(I_2) = \frac{p(I_1 = 1, I_2 = 1)}{p(I_1 = 1)} \bullet V(I_1) + \frac{p(I_1 = 0, I_2 = 1)}{p(I_1 = 0)} \bullet (1 - V(I_1)) \tag{3.5}$$

where

$$\begin{aligned} p(I_1 = 1) &= p_{10} + p_{11} \\ p(I_1 = 0) &= p_{00} + p_{01} \end{aligned} \tag{3.6}$$

In the case in which $V(I_2)$ is given and determines $V(I_1)$, I_1 and I_2 are switched in equation (3.5). This case is required when a signal is propagated backwards (in testing stage).

Case 3: Given the value of the output $V(O)$*, decide the value of both inputs* $V(I_1)$ *and* $V(I_2)$.

$$\begin{aligned} V(I_1) &= \frac{p(F = 1, I_1 = 1)}{p(F = 1)} \bullet V(O) + \frac{p(F = 0, I_1 = 1)}{p(F = 0)} \bullet (1 - V(O)) \\ V(I_2) &= \frac{p(F = 1, I_2 = 1)}{p(F = 1)} \bullet V(O) + \frac{p(F = 0, I_2 = 1)}{p(F = 0)} \bullet (1 - V(O)) \end{aligned} \tag{3.7}$$

This corresponds to OOA as shown in Figure 3-4 (b). $p(F = 1)$ and $p(F = 0)$ are determined by the probability in Table 3-1 in accordance with the F value. For instance, if

F = {0 1 0 1}, then $p(F=1) = p_{01} + p_{11}$ and $p(F=0) = p_{00} + p_{10}$. This case is required when a signal is propagated backwards.

Case 4: Given the values of one input, ($V(I_1)$ or $V(I_2)$, and the output, $V(O)$, decide the other input value, $V(I_2)$ or $V(I_1)$;

This case corresponds to the INOUA in Figure 3-4 (c). For example, consider the case that given $V(I_1)$ and $V(O)$ to decide $V(I_2)$ as follows:

$$V(I_2) = \frac{p(I_1 = 1, F = 1, I_2 = 1)}{p(I_1 = 1, F = 1)} \bullet \hat{V}_{11} + \frac{p(I_1 = 0, F = 1, I_2 = 1)}{p(I_1 = 0, F = 1)} \bullet \hat{V}_{01} + \frac{p(I_1 = 1, F = 0, I_2 = 1)}{p(I_1 = 1, F = 0)} \bullet \hat{V}_{10} + \frac{p(I_1 = 0, F = 0, I_2 = 1)}{p(I_1 = 0, F = 0)} \bullet \hat{V}_{00} \tag{3.8}$$

where $\hat{V}_{11}, \hat{V}_{01}, \hat{V}_{10}$ and $\hat{V}_{00}$ are determined in the following way:

$$\hat{V}_{11} = \begin{cases} V(I_1) \bullet V(O) & \begin{cases} X & X & 0 & 1 \\ X & X & 1 & 0 \end{cases} \\ 0 & X \quad X \quad 0 \quad 0 \\ V(I_1) & X \quad X \quad 1 \quad 1 \end{cases} \tag{3.9}$$

$$\hat{V}_{10} = \begin{cases} V(I_1) \bullet (1 - V(O)) & \begin{cases} X & X & 0 & 1 \\ X & X & 1 & 0 \end{cases} \\ V(I_1) & X \quad X \quad 0 \quad 0 \\ 0 & X \quad X \quad 1 \quad 1 \end{cases} \tag{3.10}$$

$$\hat{V}_{01} = \begin{cases} (1-V(I_1)) \bullet V(O) & \begin{cases} 0 & 1 & X & X \\ 1 & 0 & X & X \end{cases} \\ 0 & 0 \quad 0 \quad X \quad X \\ (1-V(I_1)) & 1 \quad 1 \quad X \quad X \end{cases} \tag{3.11}$$

$$\hat{V}_{00} = \begin{cases} (1-V(I_1)) \bullet (1-V(O)) & \begin{cases} 0 & 1 & X & X \\ 1 & 0 & X & X \end{cases} \\ (1-V(I_1)) & 0 \quad 0 \quad X \quad X \\ 0 & 1 \quad 1 \quad X \quad X \end{cases} \tag{3.12}$$

The conditions in equation (3.9) ~ (3.12) refer to the value of F in Table 3-1, where "X" is a do not care, which means its value can be either '0' or '1'. For example, if one PE received $V(I_1) = m$ and $V(O) = t$, and the function value of this PE is F = {0 1 1 1}, we will get the following results:

$$\begin{aligned} \hat{V}_{11} &= m; \\ \hat{V}_{10} &= 0; \\ \hat{V}_{01} &= (1-m) * t; \\ \hat{V}_{00} &= (1-m) * (1-t) \end{aligned} \tag{3.13}$$

When $V(I_2)$ and $V(O)$ are given one only needs to switch I_1 and I_2 in equations (3.8) ~ (3.12) to decide $V(I_1)$. This case is required when a signal is propagated backwards.

3.3 Self-organizing associative memory structure and operation

3.3.1 Network structure

The overall memory network is a hierarchy of sparsely connected self-organizing processing elements similar in its connectivity structure to the self-organizing learning array reported in [Sta 05a]. As in SOLAR, each PE in the array can self-organize by dynamically adapting its function in response to the input data. However, unlike SOLAR, the memory proposed in this research uses feedback as well as feedforward connections between PEs to produce an associative memory scheme. Since in this research I focus on the associative memory mechanism, I assume a fixed interconnection structure. This, however, is not a restriction of the proposed model. In a self-organizing structure, each PE can chose its connections based on probabilities of success as discussed in [Sta 05b].

All PEs in the memory network have two inputs, I_1 and I_2, and one output, O. Each port is also associated with a feedback signal for input 1, input 2, and the output denoted by I_{1f}, I_{2f} and O_f, respectively. All the PEs are identical and function according to their own probability distributions. In the hierarchical structure adopted in this research, each PE connects only to PEs in the next lower and higher layers. This differs from the previous work [Sta 05a] in which each PE can reach distant layers. The hierarchical connections are suitable for hardware implementation, time control, and correlate well to complexity of object representation. The further away a PE is from the sensory input, the more abstract and invariant the representation of objects or their features captured by the PEs. Each PE is more likely to connect to other PEs within a short Euclidean distance. This organization is observed

in biological memory where neurons tend to have mostly local connections. Thus, the lateral connection probability is a superposition of a Gaussian and a uniform distribution.

3.3.2 Network operation

3.3.2.1 Feedforward operation

Feedforward operation is necessary in both training and testing stages. Figure 3-5 shows a feedforward network structure for the associative memory. For simplicity, I will only illustrate four layers with six PEs per layer. The bold lines from PE 1 to PE 11 and from PE18 to PE21 are two examples of distant connections.

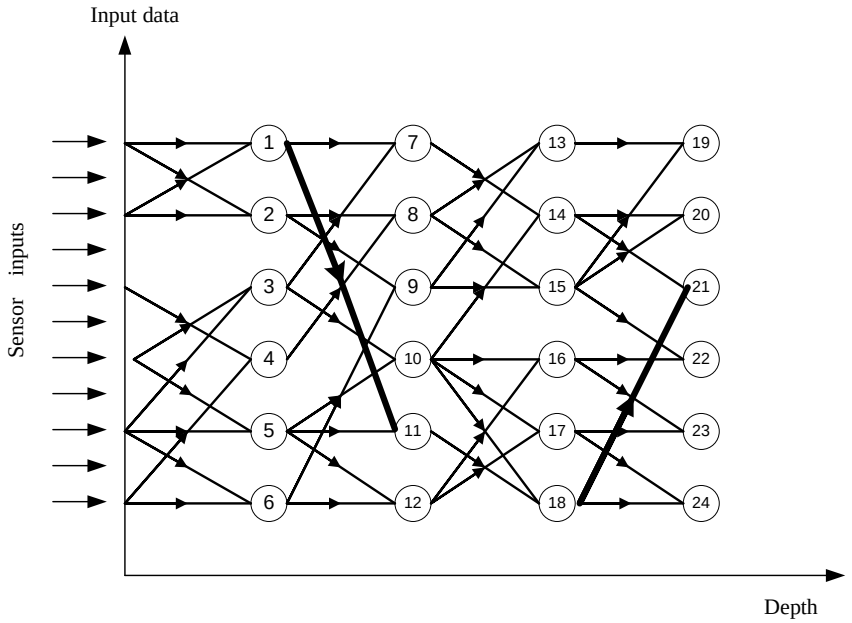


Figure 3-5 An example of feedforward operation network

During training, all external sensor input data is presented to the network. Each PE counts activities on its inputs to estimate the corresponding probabilities, $p_{ij}, (i, j = 0,1)$, and decide its output function as in case 1 of Section 3.2.4.2. During testing, some input values are undefined (the signal value is set to 0.5). Whenever there is an undefined input signal, the output of the PE is undefined; otherwise, it will be decided as in case 1 of Section 3.2.4.2 using the probabilities established during training. In fact, the distinction between training and testing is artificial as the network always learns, updating input probabilities of all PEs that receive determinate inputs.

3.3.2.2 Feedback operation

Feedback operation is essential for the network to make correct associations and to recover the missing parts of the input data.

Figure 3-6 shows a feedback structure in the testing stage. For consistency, the network connection is the same as shown in Figure 3-5, although some feedforward connections are not shown in order to show feedback signals clearly. In testing stage, some information is undefined as would be the case in a classification application (all the class ID code inputs are undefined and only the feature input values are presented to the associative memory network) or an image recovery application (part of the image could be blocked or undefined). In both cases, the network will use associations to determine the undefined signal values.

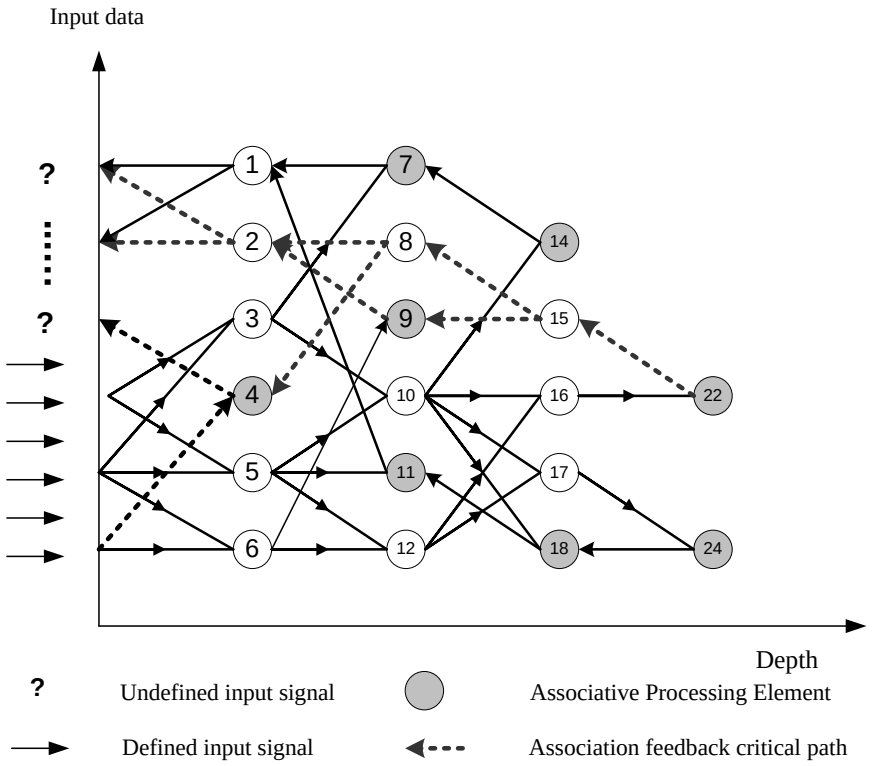


Figure 3-6 Example of feedback structure in testing stage

In Figure 3-6, the shaded PEs are associative and will use associations to recover the undefined values. All three types of association presented in section 3.2.4 are used. As illustrated in Figure 3-6, the proposed model has the ability to self-determine the depth of the association used in the feedback structure based on the complexity of the input data. This kind of self-organization provides great flexibility in applications of different complexity.

To illustrate the feedback association mechanism, let us consider part of an associative memory shown in Figure 3-7.

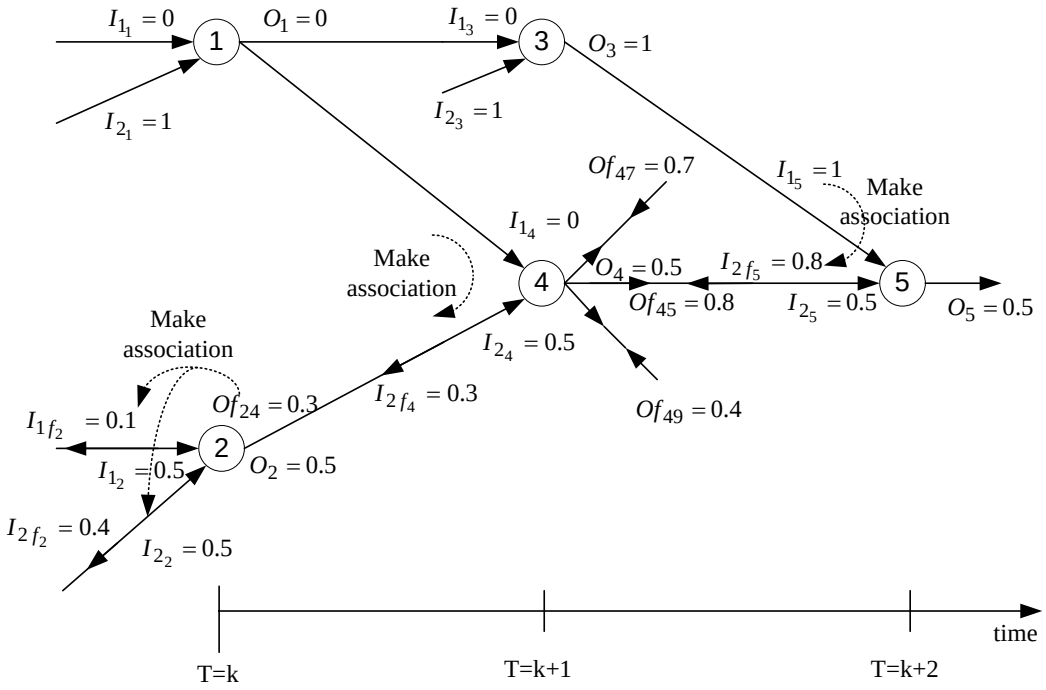


Figure 3-7 Associative memory feedback mechanism

At time T=k:

PE1 has two determinate inputs (I_{1_1} is determinate low and I_{2_1} is determinate high, sub-subscripts denote PE numbers). In this case, PE1 decides its output value according to the probability learning algorithm (case 1) in Section 3.2.4.2. Assume that one get output $O_1 = 0$. PE2 has two undefined inputs. Accordingly, PE2 outputs an undefined value $O_2 = 0.5$. No associations or feedback values are used.

At time T=k+1:

Since both inputs of PE3 are defined, it decides its output value according to case 1 in Section 3.2.4.2. Since $I_{2_4} = 0.5$ and output of PE1 is defined ($O_1 = 0$), PE 4 makes an IOA to determine the feedback signal I_{2f_4}. Assume that the calculated feedback signal of I_{2f_4} is

0.3. This feedback signal, I_{2f_4}, becomes the output feedback signal of PE2 ($O_{f_{24}} = 0.3$). Here the first and second subscripts indicate the target and source of the feedback (4 onto 2). Since both inputs of PE2 are undefined, $O_{f_{24}}$ triggers the association from output to both inputs (OOA), which corresponds to case 3 in Section 3.2.4.2. Assume that calculation from the training information leads to $I_{1f_2} = 0.1$ and $I_{2f_2} = 0.4$ for PE2. These input feedback signals become the output feedback signals to the targets of PE2 in lower level layers, and may trigger other associations.

At time T=k+2:

Since I_{1_5} is defined ($I_{1_5} = 1$) and I_{2_5} is undefined, the IOA association will decide the input feedback signal of PE5 according to case 2 in Section 3.2.4.2 (assume that the calculated $I_{2f_5} = 0.8$). At the same time, let us assume that PE4 receives three feedback signals from three different PEs (PE 7, PE5 and PE9) with $O_{f_{47}} = 0.7$, $O_{f_{45}} = 0.8$ and $O_{f_{49}} = 0.4$. In this case, the output feedback signal of PE4 is selected as the one with the largest signal strength, which means:

$$O_f = \max\left(SS\left(O_{f_{4i}}\right)\right) \tag{3.14}$$

where i represents the PEs from which PE4 received feedback signals. In this case, we have:

$$\begin{aligned} SS\left(O_{f_{47}}\right) &= |0.7 - 0.5| = 0.2 \\ SS\left(O_{f_{45}}\right) &= |0.8 - 0.5| = 0.3 \\ SS\left(O_{f_{49}}\right) &= |0.4 - 0.5| = 0.1 \end{aligned} \tag{3.15}$$

The feedback signal of PE4 becomes $O_{f_4} = 0.8$. This signal triggers associations in PE4. For PE4, I_{1_4} is defined ($I_{1_4} = 0$), O_{f_4} is defined ($O_{f_4} = 0.8$), and I_{2_4} is undefined. Accordingly, the feedback signal, I_{2f_4}, is based on the input-output association. Assume that, from case 4 in Section 3.2.4.2, one get $I_{2f_4} = 0.2$. In the previous step, the signal on I_{2f_4} is 0.3. Since there can be only one value in a particular signal line, these two signals need to be resolved. This is achieved by choosing the signal with the largest signal strength. In this case, 0.2 ($SS = 0.3$) is stronger than 0.3 ($SS = 0.2$), so I_{2f_4} is updated to 0.2. Now, updated I_{2f_4} becomes the output feedback signal for PE2, which in turn triggers the associations of PE 2 based on case 3 in Section 3.2.4.2.

In summary, the proposed memory makes the necessary associations to trace a signal to previous layers, ultimately deciding the undefined signal values. It should be noted that the updated input feedback signals (I_{1f} and I_{2f}) and the output feedback signal (O_f) must not propagate forward to higher hierarchical layers, since they may cause instability and start oscillations in the network.

3.4 Summary

This chapter discussed the self-organization associative memory for machine intelligence. Self-organization and association are desirable features of learning structures and evidently play a critical role in development of the brain [Hau 83] [Erw 95] [Swi 96]. Self-organizing structures require no explicit supervision and selectively affect different areas of learning networks, depending on how new data is related to the information stored in the

network and how the emergent network behavior is useful to the machine. Self-organization in human memory builds natural associations between different sensors and internal representations of the external world. As in the real brain, in the described associative memory network there is no pre-designated group of PEs that represent a specific concept or make a specific association. These emerge spontaneously from interactions between sensors, PEs and the external world. In particular, the signal propagation depth in this network is not predefined. It emerges from building associations and is a function of structural complexity of the input data. It can be observed that, in more complex cases, the depth of association increases, and more PEs are actively involved in the decision-making processes.

Chapter 4

Anticipation-Based Temporal Sequences Learning in Hierarchical Structure

4.1 Introduction

Temporal sequence learning is one of the most critical components for human intelligence. I presented the learning and memory structures in Chapter 2 and Chapter 3, respectively. After the information is learned and remembered in the distributed elements (neurons), the machine should be able to anticipate the future inputs. As most human behaviors and related information processing are sequential, a prediction mechanism is introduced in the learning and memory structure for the machine to be able to learn, memorize, and predict any multiple complex sequences [Sta 06c]. This contributes to the third element of the proposed "learning-memory-prediction" framework: prediction.

4.2 System architecture

In this dissertation, I focus on the learning and retrieval of multiple complex temporal sequences in the network of neurons. I have adopted the terminology introduced by D. Wang ad M. A. Arbib in [Wan 90a] and [Wan 93]. A temporal sequence is defined as

$$S: \ S_1 - S_2 ... - S_n \tag{4.1}$$

Where S_i, $i = 1,...,n$ is a component of sequence S and the length of the

sequence S is n. If a sequence includes repetitions of the same subsequence in different contexts, it is called a complex sequence; otherwise it is a simple sequence. Figure 4-1 shows the hierarchical organization of three levels of letter-word-sentence. It is believed that such a hierarchical structure of a sequence is efficient [Wan 93] [Man 97] and [Geo].

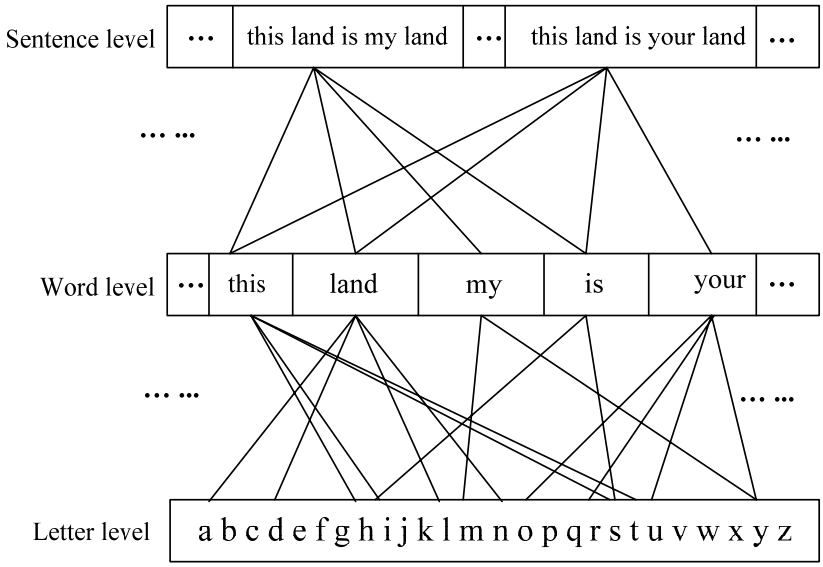


Figure 4-1 Hierarchical organization of sequences

The overall system architecture of the proposed model is shown in Figure 4-2. It contains a hierarchical layer structure. The output of one hierarchical level is the input to the next level. At each level, a winner-takes-all algorithm is used to select active neurons.

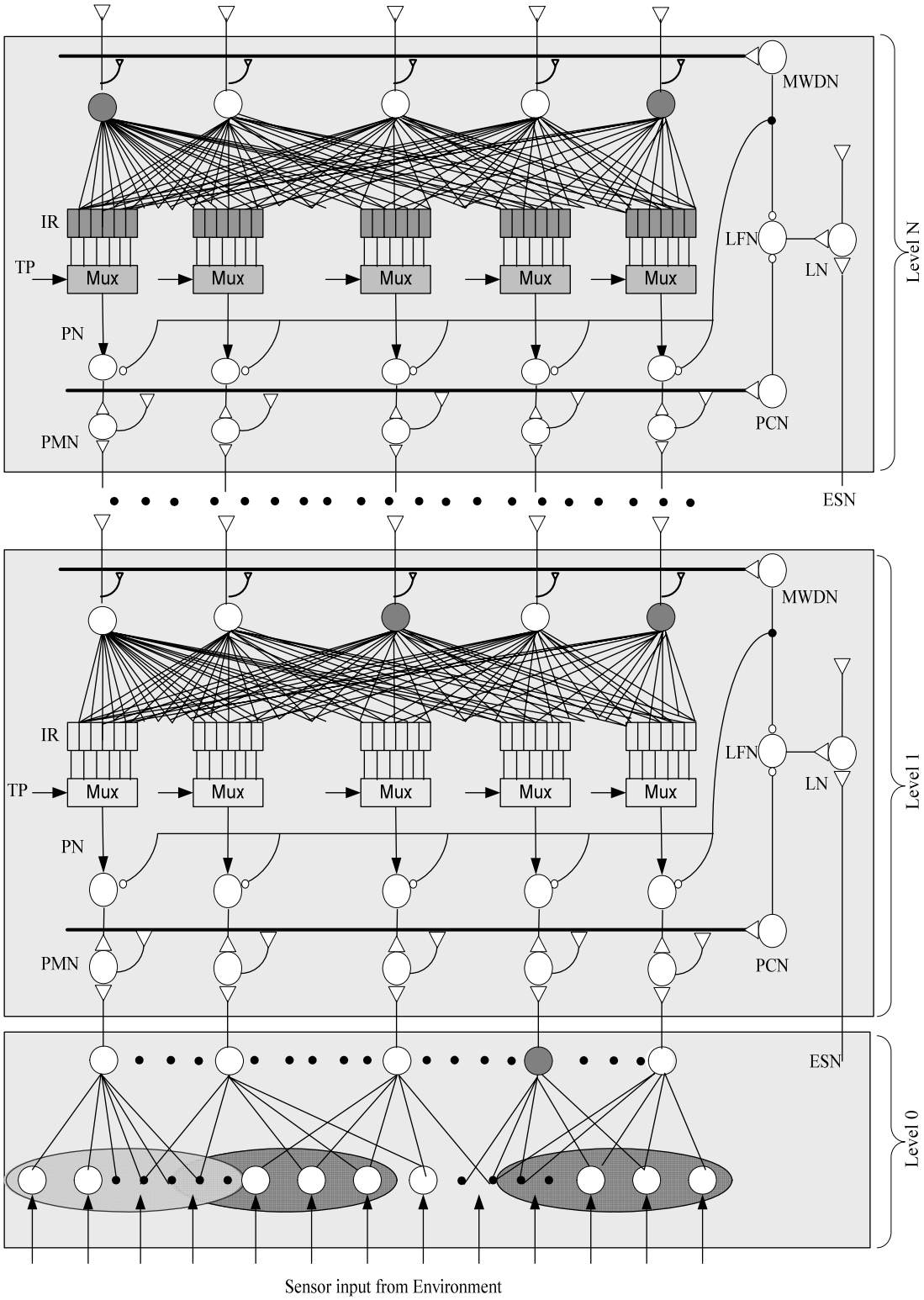


Figure 4-2 System architecture of anticipation-based hierarchical temporal sequence learning

Winning neurons are indicated in grey. In the first layer (level 0), a modified Hebbian learning mechanism is utilized for pattern recognition. Level 1 to level N are identical structures for temporal memory. The key components of each hierarchical level are: input register (IR), multiplexer (MUX), prediction neuron (PN), prediction checking neuron (PCN), prediction matching neuron (PMN), learning flag neuron (LFN), multiple winner detection neuron (MWDN) and learning neuron (LN). The IR spatially encodes a temporal sequence of outputs from the next lower level. The sequence may be recently entered or recalled from the sequential memory. The MUX calls the contents of the IR sequentially for comparison with a sequence being output by the next lower level. I will show that this network is efficient for complex temporal sequence learning and retrieval.

4.3 Level 0: A modified Hebbian learning mechanism

A modified Hebbian learning mechanism is used in the first hierarchical layer of the proposed model (level 0 on Figure 4-2). Since biological neurons either fire or do not, it is assumed that each sensory input from the environment is either 0 or 1. In this way, Figure 4-3 illustrates the idea of a both active-0 and active-1 representation of the sensory information. When the sensory input is 1, the left neuron will fire. When the sensory input is 0, this input value is passed through an “inverter” and drives the right neuron to fire. In this way, different neurons firing represent different sensory input values for the binary coded input.

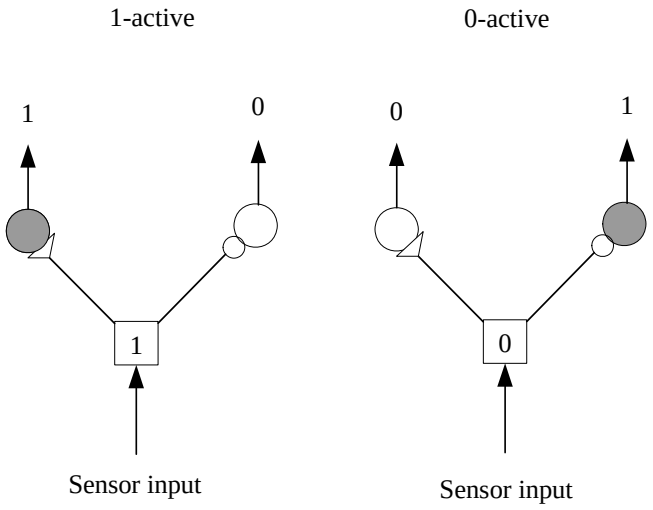


Figure 4-3 Two-active area neuron firing mechanism

Figure 4-4 shows the detailed structure of the modified Hebbian learning mechanism used in this dissertation research. For simplification, I show a three level hierarchical structure with unsupervised learning. The neurons in the second layer are grouped into several groups. Each neuron of a second layer group is sparsely connected, at random, to the same subset of neurons in the first layer. There is some overlap of the subsets of neurons in the first layer to which second layer groups project.

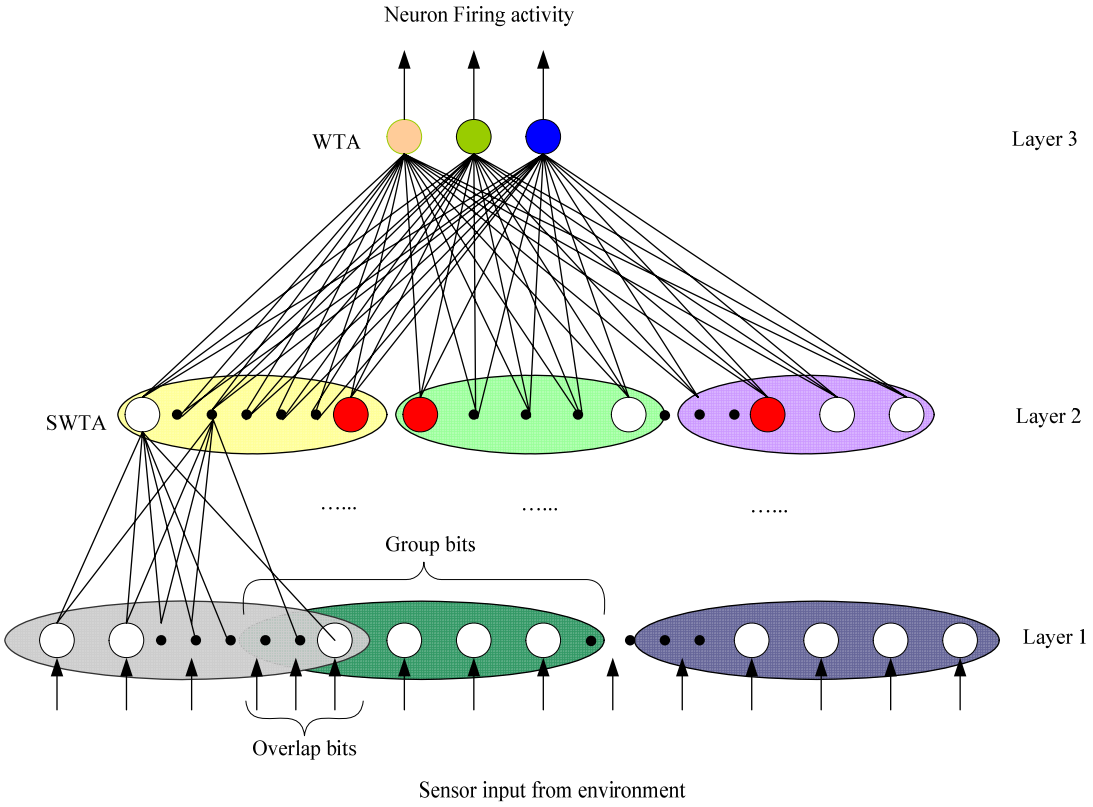


Figure 4-4 Hierarchical structure of the modified Hebbian learning

Two similar winner-take-all mechanisms are used in this research to improve the learning efficiency of the model as well as to reduce its learning complexity. The first one is a stiff WTA (SWTA), which can be achieved by a simple counter. SWTA is used in the second layer. Since the sensory inputs from the environment are either 0 or 1, SWTA simply counts the number of 1's each neuron receives, and selects, as winners, the one neuron from each group that receives the largest number of 1's. In particular, there is no weight adjustment in SWTA.

The second WTA mechanism is used in the output layer (layer 3 in Figure 4-4). Initially, all the weights, w_i, for neurons in the output layer are randomly set with the following conditions:

$$w_{ni} = \pm 1$$
$$\sum_i w_{ni} = 0 \tag{4.2}$$

Where n indexes the neurons, and $i = 1,2...k$ indexes connections onto neuron n from all the neurons of the previous layer. The winner is given by

$$\text{Winner (w)} = \max_n \left(\sum_i w_{ni} I_{ni} \right) \tag{4.3}$$

Where I_{ni} is the activity (0 or 1) of the neuron from which projection w_{ni} arises. In each time instance, after the winner is selected, the weights of the winner (w_{wi}, where the subscript w denotes a winner) are adjusted as follows:

For the connections that received input $I_{li} = 1$,

$$w_{wi}(t+1) = w_{wi}(t) + \Delta * M(0) \tag{4.4}$$

For the connections that received input $I_{li} = 0$:

$$w_{wi}(t+1) = w_{wi}(t) - \Delta * M(1) \tag{4.5}$$

Where Δ is a small adjustment, and $M(1)$ and $M(0)$ are the number of 1s and 0s the neuron received. This adjustment guarantees that the sum of the weights for the winner after adjustment still equals 0. After all the weights of the winners are adjusted, these weights are linearly scaled to be in the range [-1,1]. Unsupervised learning is used in this model, meaning that each output neuron updates its activity whenever a new training sample is presented.

4.4 Level 1 to level N: Sequence learning and prediction

4.4.1 Storing input sequences

As indicated in Section 4.3, the first hierarchical level provides recognition of component S_i, $i = 1...n$ in the sequence S. From now on, I focus on how the model can store and predict complex sequences.

The output of one level is stored in the input registers of the next higher level by means of connections not shown in Figure 4-2. Neurons of the input registers project onto the output neurons of the same level through trainable connections. Initially all output neurons are fully connected to all IR's through untrained links with electrical (resistive) synapses. In the current model, the initial weights for all the output neurons are set to small positive numbers $0.001 < w_i < 0.01$.

Once an input sequence is stored in the input registers, all the output neurons compete and weights of the winning neuron are adjusted. All the weights of projections of active neurons onto a winning neuron in IR are set to 1 (excitatory) and all other weights onto a winning neuron are set to -100 (inhibitory). I employ this strong inhibition to guarantee that once an output neuron is trained to store a sequence, it is excluded from further learning. For instance, consider the IR states as shown in Figure 4-5, where the links with a triangle represent excitatory projections and links with a circle represent inhibitory projections. The locations of the various IR's establish that this neuron stores the letter sequence “miss” (i.e., above the letter “m” there is a trained connection from the section of the IR representing the first time step). Once an output neuron’s links are trained, the neuron responds only to one

specific input sequence.

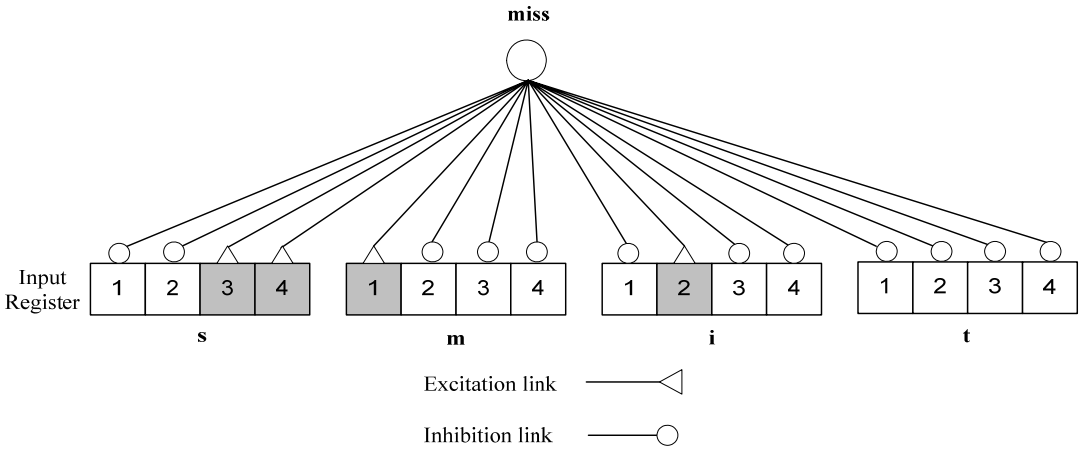


Figure 4-5 Storage of sequence in input register

Before discussing the prediction mechanism, I describe the structure of the input register. In the proposed model, the IR stores the input data selected by WTA in the location indicated by a time pointer, each pattern in a sequence is directed to a different location. The structure of the IR is shown in Figure 4-6, which contains pointer neurons and IR neurons with inhibitory and excitatory links. A unit delay Δ in activation of the links between neurons is considered in this research.

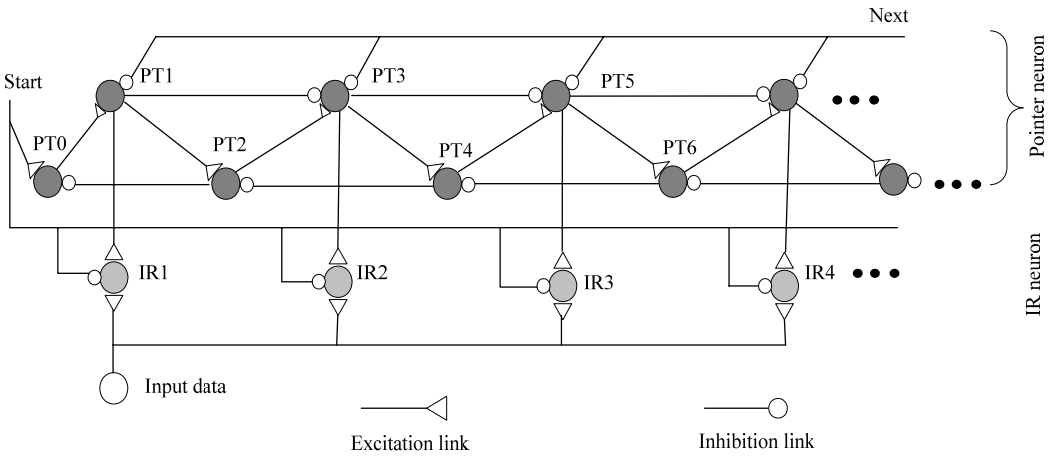


Figure 4-6 Structure of input register

Figure 4-7 illustrates the storage of input data based on the input register structure in Figure 4-6 timed by neurons activity. Suppose the incoming data is "*AB*". At the beginning of the new sequence a *Start* signal is presented (*start* signal is an impulse signal at the beginning of the new sequence). The *Start* signal clears all information in the IR through inhibitory links. At the same time, $PT0$ fires. When the first data, "*A*", is received, the *Next* signal is generated. When the *Next* signal goes low, the inhibition of the upper pointer neurons is removed. Therefore, after the delay time Δ, $PT1$ will fire. After one more delay time, the lower pointer neuron $PT2$ fires since it has an excitatory link from $PT1$. At the same time, $PT1$ provides the inhibition to $PT3$. As illustrated in Figure 4-6, since the *Start* signal is low, and both the $PT1$ and the input data neuron are active, input register neuron $IR1$ fires to store the first data in the $IR1$. Assume that after some time, another data "*B*" is presented, and a pulse signal *Next* is generated. This signal inhibits all the upper pointer neurons $PT1$, $PT3$, $PT5$ and so on. However, $PT2$ continues firing. When the *Next* signal goes low, after the delay time Δ, $PT3$ fires because it is excited by $PT2$. As before, $PT4$ fires after $PT3$ with delay Δ. At the same time, $IR2$ fires because it is excited by $PT3$ and the input data neuron. This will store the data "*B*" in the second input register, $IR2$. The entire time diagram is illustrated in Figure 4-7.

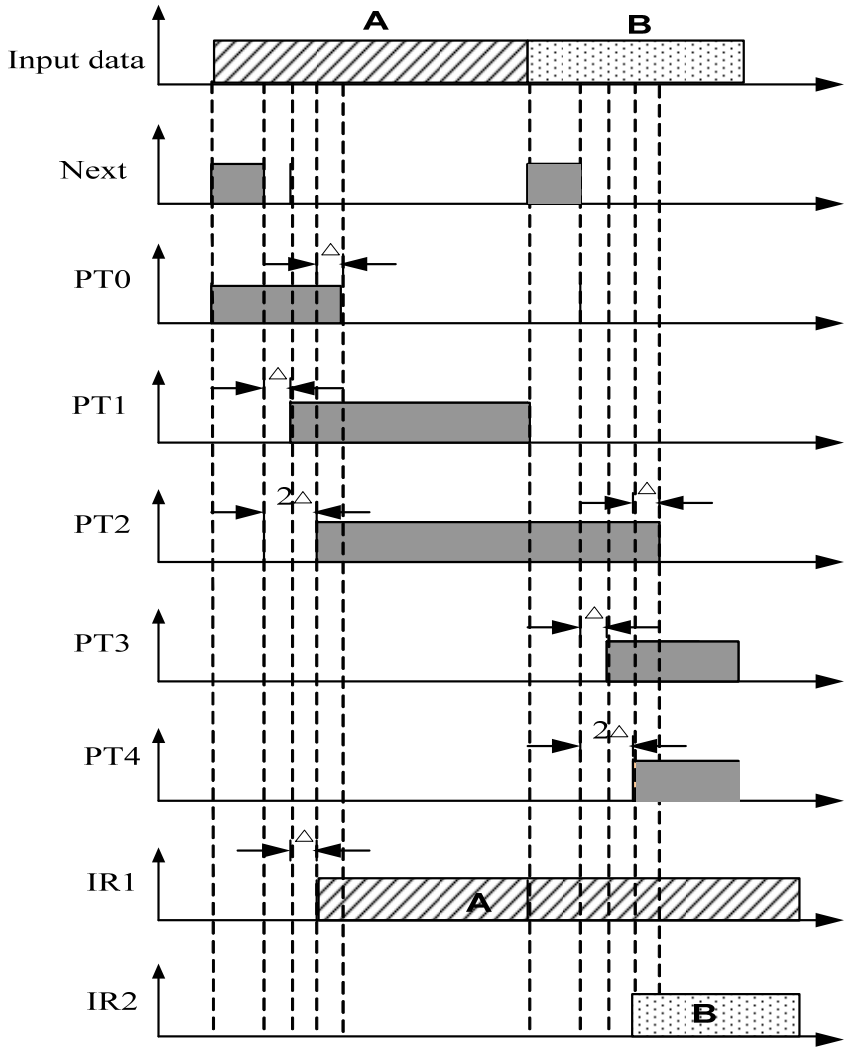


Figure 4-7 Timing diagram of neurons firing in the IR

The process continues until all the input data of the sequence is presented. In this scheme, the lower level pointer neurons provide inhibitory feedback to remove excitation from prior pointer neurons and to excite the next pointer neuron. This approach forms long term memories (LTM) of the learned sequences. One may make these memories modifiable by allowing training to decay after some specified period of time.

4.4.2 Predicting the input sequences

4.4.2.1 Prediction mechanism

Predicting an input sequence is an essential part of sequence learning in this model. By correctly predicting a sequence, the machine indicates it knows the sequence and does not require additional learning. If there is a single error, the long term memory is modified to learn the new sequence. A simple one-shot learning mechanism is employed.

The first stage of prediction is a competition stage. Several sequences stored in LTM compete to determine a unique sequence that may correspond to the input sequence. Once such a sequence is determined, it predicts the consecutive inputs. (Notice, that the same mechanism is used if LTM is simply playing back a stored sequence after a high level node of LTM is triggered by an internal process). A multiple winner detection neuron is used to detect whether there are multiple winners with trained links. This is achieved by setting the threshold of the MWDN equal to 2. Therefore, it will fire when there are 2 or more winners in the output layer (This occurs when the sum of the weights of all the winners are the same). The output of the MWDN is connected to the learning flag neuron as well as all the prediction neurons through inhibitory links. This provides a mechanism for setting the proper learning flag signal. The output of the LFN is connected to the learning neuron through an excitatory link. Together with the excitatory links from the end of input sequence neuron, the whole system provides a proper mechanism for setting the learning signal. It should be noted that each hierarchy level requires its ESN to indicate the end of input sequence for this level. In addition, if an ESN neuron fires in a higher hierarchical level, it automatically generates an

ESN signal in all the lower hierarchical levels. Consider an example in which the first level of hierarchy in LTM stores letters, the second level is stores words, and the third level stores sentences. The last letter of every word will trigger the activation of the ESN neuron for the second level. The last letter of the last word in every sentence will be followed by the activation of the ESN for the third level, and at the same time, the ESN signal will also be sent to the lower levels (level 2 in this case).

There are three cases of all possible outcomes for the competition stage of the prediction mechanism.

Case 1: There is a single winner with trained links

In this case, MWDN does not fire because it has a threshold of 2. Therefore, PN and LFN are not inhibited. The set of PN's includes one PN corresponding to each output neuron of the next lower level. By a mechanism described in Section 4.4.2.3, LTM acts through the MUX at each time step to activate an excitatory projection onto a PN that represents the prediction of the network for the time step. The time pointer (TP) is incremented with each new symbol (pattern) presented to LTM. As may be seen in Figure 4-2, each PN receives an inhibitory projection from the MWDN, as well as the excitatory projection from the MUX. Since, in this case, the PN is not inhibited by the MWDN, LTM (acting through the MUX) activates the particular PN corresponding to the predicted symbol for the input sequence. Each prediction neuron and its corresponding output neuron from the next lower level form two inputs to a prediction matching neuron. Firing of a PMN verifies that the predicted symbol corresponds to the input symbol, as shown in Figure 4-8.

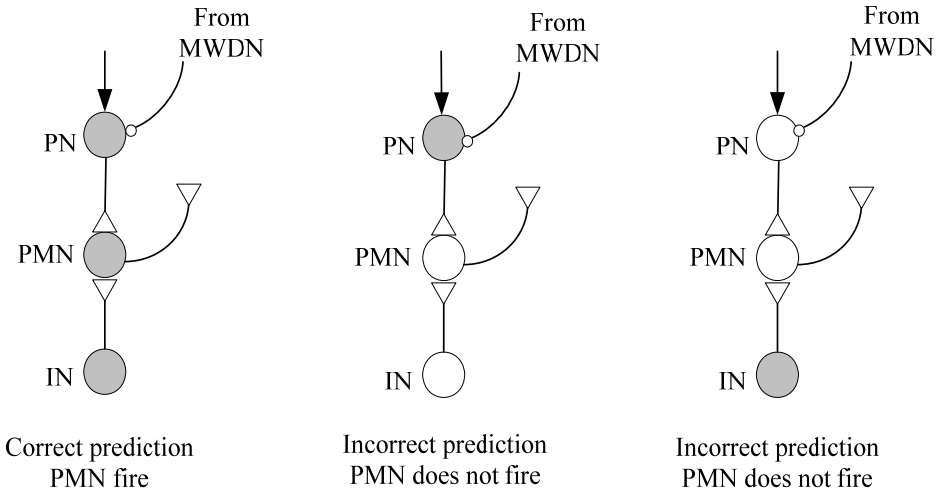


Figure 4-8 Firing mechanism of PMN

All prediction matching neurons have their outputs connected to the prediction checking neuron as shown in Figure 4-9. This neuron fires to indicate there is a correct prediction.

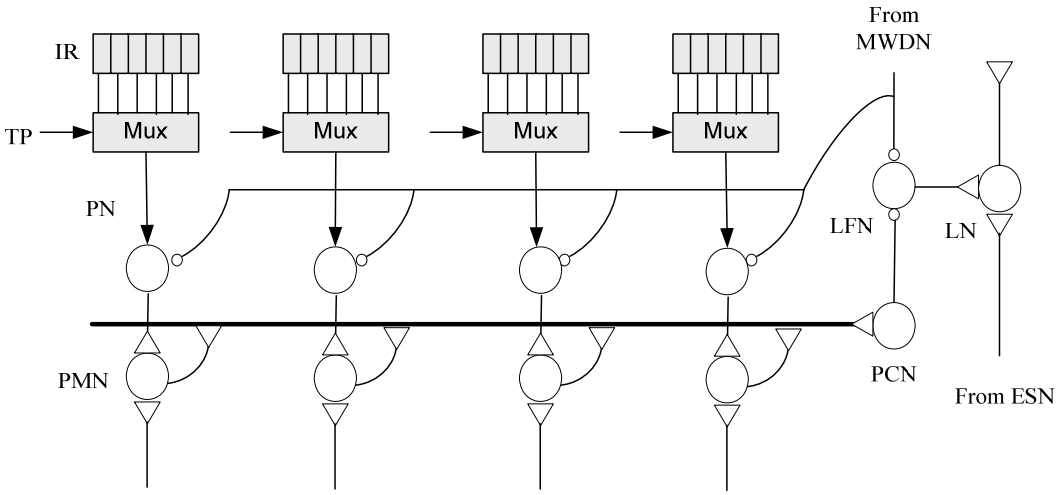


Figure 4-9 Prediction mechanism

If there is no match, the learning flag neuron is set automatically (no inhibition from PCN or MWDN). LFN remains on, and the sequence continues until ESN fires. Thus, the

firing of both LFN and ESN triggers the learning neuron, as shown in Figure 4-10 (a). If there is a match on the output of the PMN, PCN will fire. Therefore LFN is inhibited. Figure 4-10 (b) and (c) shows the remaining two conditions for the LN. Figure 4-10 (b) indicates that if only the LFN fires (meaning there is no correct prediction), the LN neuron will not fire because the ESN does not fire in this situation. Figure 4-10 (c) indicates that if only the ESN fires, the LN will not fire because the LFN does not fire (meaning there is a correct prediction).

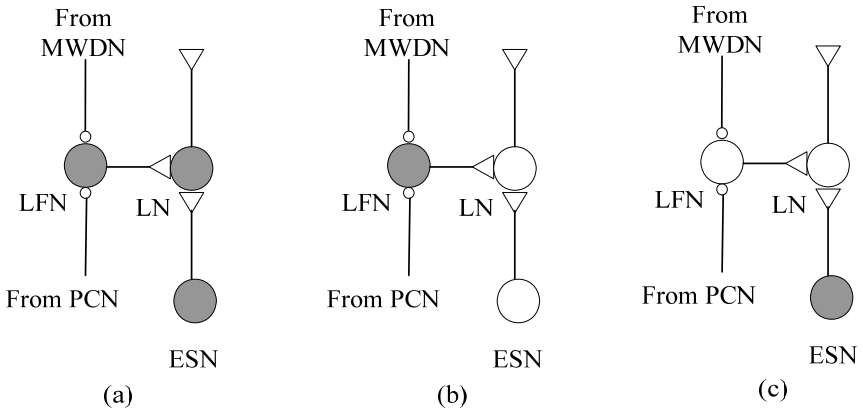


Figure 4-10 Firing mechanism of learning neuron

Case 2: There are multiple winners with trained links

Consider the case shown in Figure 4-11. Assume two words "*miss*" and "*mom*" are already stored in LTM (only excitation links are shown in Figure 4-11) and the new sequence is "*mit*". When the first symbol of this sequence "*m*" is input, both neuron $n1$ and $n3$ win with trained links (weight equal to 1). MWDN then reaches its threshold of 2 and fires. As indicated in Figure 4-11, MWDN inhibits both PN and LFN. Since both neurons are the winners with trained links, it is premature for the network to attempt a prediction.

(Inhibition of all the PN's also reduces energy consumption). When the second symbol "*i*" is presented, neuron $n1$ wins because it receives two excitatory projections from IR, while neuron $n3$ receives an inhibitory projection. In this situation, there is a single winner with trained links (i.e., case 1). MWDN does not fire, removing the inhibition on PN and LFN. With the control of the TP signal through the MUX, $n1$ will predict the next symbol as "*s*", which is not correct in this case. Therefore, PMN will not fire (please refer to Figure 4-8), and PCN does not fire. In this way, LFN will fire because there is no inhibition form PCN. At this time, LN does not fire because ESN does not fire. When the third symbol "*t*" is presented, both $n1$ and $n3$ receive inhibition from IR and neither wins. Assume $n2$ is the winner without lost of generality, this will lead to case 3 in which there is a single winner with untrained links.

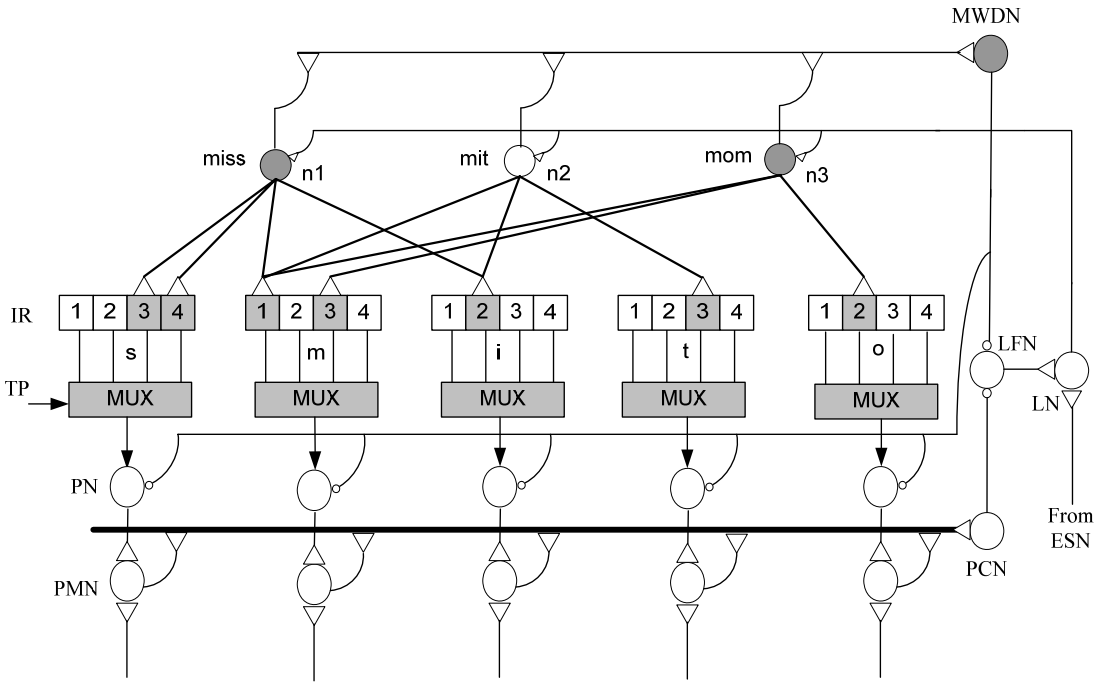


Figure 4-11 LTM and multiple winners

Case 3: There is a single winner with untrained links.

In this situation, MWDN does not fire, and since the winner has untrained links, PN does not fire. Therefore PCN does not fire, allowing LFN to fire. LFN remains active until ESN fires. The combination of LFN and ESN then causes LN to fire, and a learning signal activates one-shot learning, and adjusts weights according to the previously described rules to learn the sequence "*mit*". Figure 4-11 illustrates the strengthened connections after learning.

4.4.2.2 Activation of a prediction neuron

To perform sequence prediction each IR neuron is associated with a dual IR neuron. WTA neurons responsible for storing the sequence are linked to dual IR neurons through untrained links. IR neurons connect to their dual neurons through trained links. Thus firing an IR neuron automatically activates its dual neuron. When a sequence is stored in a WTA neuron, connections from the WTA neuron to dual IR neurons corresponding to active IR neurons in the sequence are trained (the bold line in Figure 4-12). When a previously stored sequence is input again, a partially matched sequence may activate the WTA neuron of this sequence. This will activate all dual IR neurons that compose the entire sequence. The structure is shown in Figure 4-12. This structure combined with the time-controlled multiplexer provides the mechanism for the prediction scheme.

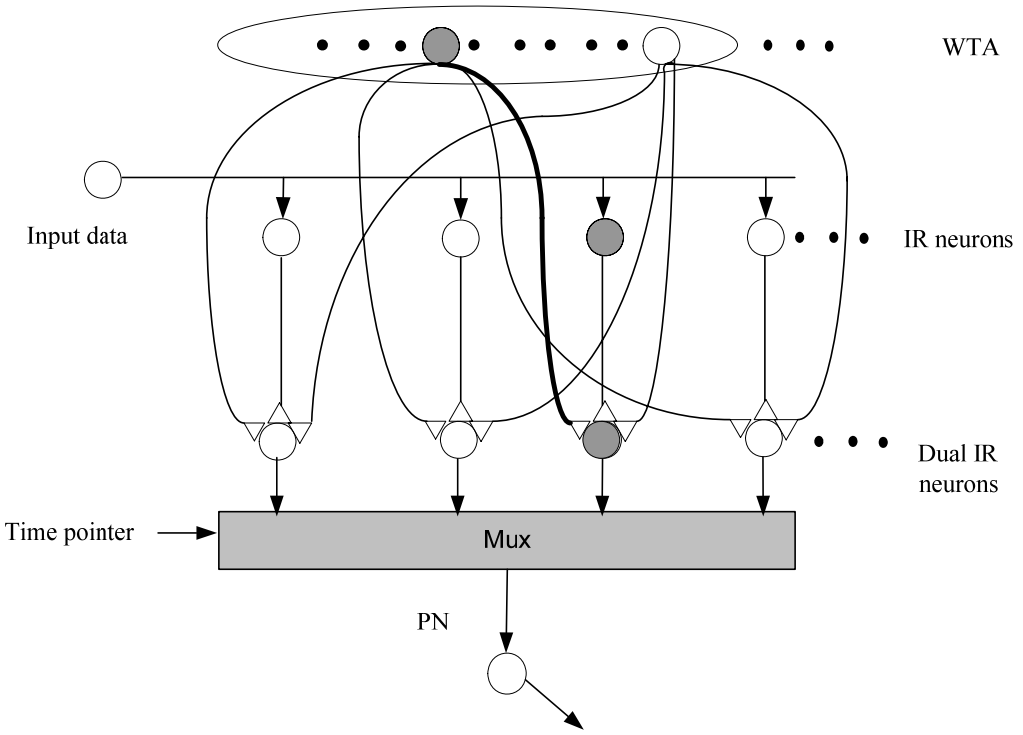


Figure 4-12 Activation of prediction neuron

4.4.2.3 Time controlled multiplexer

The neural network structure of the time-controlled multiplexer is shown in Figure 4-13. The output from WTA activates the dual IR neurons that represent predictions for each time step as discussed in Section 4.4.2.2. At a given time step indicated by an active pointer neuron, this dual IR neuron actives the corresponding IR output neuron, and subsequently the prediction neuron for the next element of the stored sequence. This predicted data is connected to the PMN, which compares the PN with the actual data and fires if the prediction is correct.

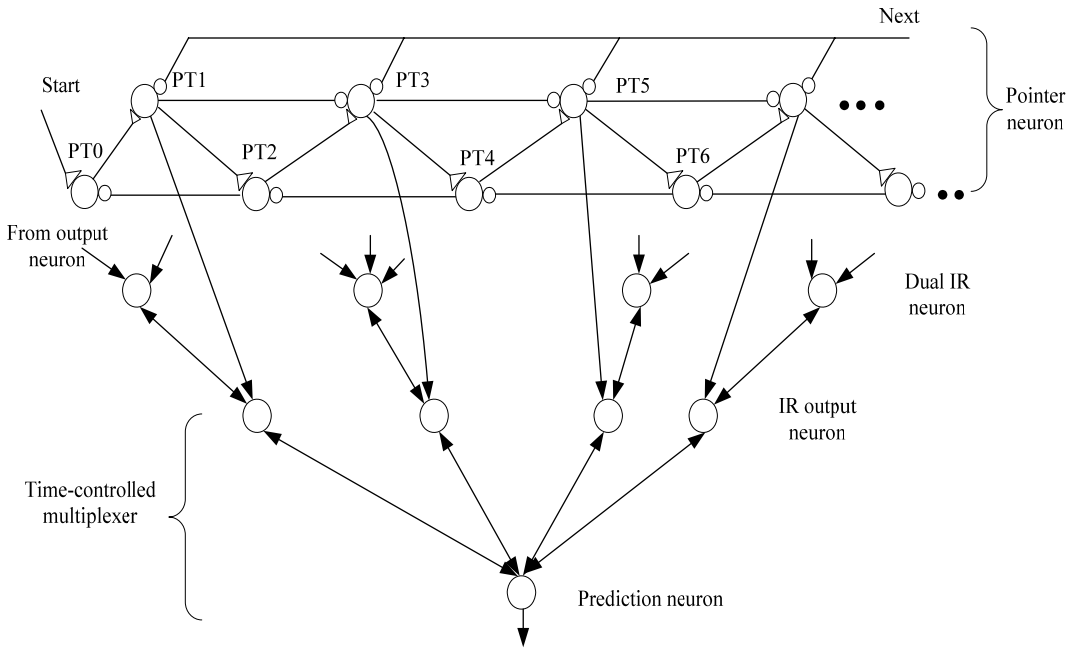


Figure 4-13 Time-controlled multiplexer

4.5 Retrieval of the stored sequences

Stored spatio-temporal sequences can be retrieved by activating neurons at any hierarchical level through internal processes or by association with a first level (sensory) input cue. An important element in the process of retrieving the stored sequences is the duration of each element of the sequence. By scaling this time, different speeds of presentation may be achieved. [Wan 93] describes a mechanism for storing the presentation time intervals in which timing is related to timing of the input sequences. While it is necessary for applications to reproduce the stored sequence at approximately the pace of the original sequence, it is not necessary in general.

If a person is given a complex task that requires completion of sequence of operations, timing of the sequence must be determined by the time needed to complete each task, and this

timing may be either unknown ahead of time or may depend on the context in which the task is executed. Biological organisms commonly rely on sensory-motor coordination to provide proper timing for execution of the stored sequences. The speed of retrieval depends on the natural delay and feedback received through sensory inputs which verifies that a particular element of the sequence was completed. This completion of the sequence element induces presentation of the next element of the stored sequence. Thus the process of retrieval of stored temporal sequences is self-timed and does not depend on any internal clock, but rather on interaction with the environment. This may be a desired feature for many real life applications where spatio-temporal memories are needed.

4.6 Example of anticipation based sequence learning

So far, I have shown how the proposed model can correctly and efficiently implement the learning and prediction of a complex sequence. I will now give an example to summarize how this mechanism works.

Example:

In order to focus on the sequence learning mechanism, I assume that each sensory input from the environment causes a corresponding winning neuron to fire at the output of the first level. Therefore, I focus on the hierarchical level 1 in Figure 4-2. As in [Wan 95], let "#" be the end-of-sequence marker for this hierarchical level. Assume one need to store and retrieve the multiple sequences: "*mis#mit#miss#mit#*", as shown in Figure 4-14. Without loss

of generality, let this hierarchical level have 3 output neurons and 27 input neurons (representing the complete alphabet plus the end marker “#”). Each of the output neurons is fully connected to all of the input neurons of the input registers and their initial weights of the synapses are set to $0.001 < w_i < 0.01$.

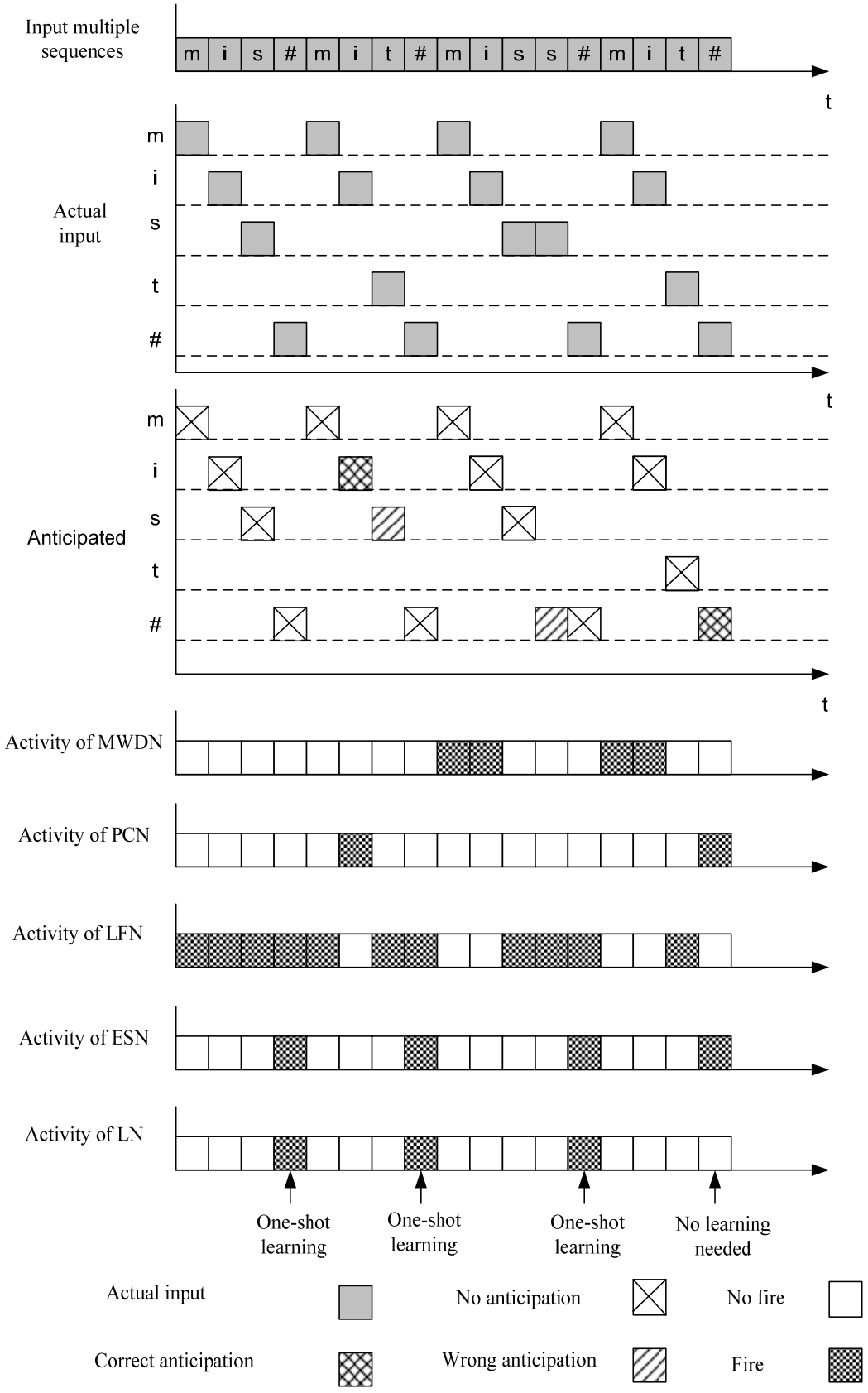


Figure 4-14 Learning and anticipation of multiple sequences

When the first symbol of the first sequence "*m*" is activated, the time pointer is set to 1. Since all the weights are initially randomly set, assume neuron 1 ($N1$) is the winner without loss of generality. Since there is no previous training, there is no prediction at this time. Therefore PCN is not fired, which causes LFN to fire. LFN continues to fire until ESN fires at the end of the first sequence. The combination of LFN and ESN fires LN, which sets the learning signal. A one-shot learning is triggered and the weights of the winner are adjusted as discussed in Section 4.4.1 (i.e., the excitatory weights are set to 1 and inhibitory weights are set to -100).

When the first symbol of the second sequence "*m*" is activated, TP is set to 1. The previously fired neuron ($N1$) becomes the single winner since it receives the all the excitatory projections from the first location of the input register. $N1$ predicts the next symbol is "*i*" through the multiplex controlled by the TP signal. In this case the prediction is correct and the corresponding PMN fires, which fires PCN and inhibits LFN. When the second symbol "*i*" is presented to the model, TP is incremented to 2. $N1$ is still the only winner since it has two excitatory links and no inhibitory links from the first two locations of the corresponding input registers. The corresponding PN neuron predicts "*s*" as the next symbol. As this prediction is not true, no PMN fires, and PCN does not fire. Accordingly, LFN is not inhibited and fires. When the third symbol "*t*" is activated, TP is increased to 3. $N1$ is not the winner because it has an inhibitory projection from the input register. Without lost of generality, let us assume $N2$ is the winner. When the end of sequence marker "#" is activated, ESN fires. When both LFN and ESN fire, LN fires, which sets the learning signal. A one-shot learning is triggered and the appropriate weights of the winner ($N2$) are adjusted.

When the first symbol of the third word "*m*" is presented to the model, there will be 2 winners with trained connections ($N1$ and $N2$). Therefore, MWDN will fire. The firing of MWDN will inhibit all PN and LFN. The MWDN fires, again, when the second symbol, "*i*", is presented. When the third symbol "*s*" is activated, $N1$ is the single winner. MWDN does not fire, and the inhibition of PN and LFN is removed. $N1$ predicts the next symbol is "#", which is not correct. When the fourth symbol "*s*" is activated, both $N1$ and $N2$ are inhibited. Assume $N3$ is the winner without lost of generality. Because $N3$ has no trained link, there is no prediction. Therefore, PCN does not fire and LFN does fire. The procedure continues until the "#" is activated and ESN fires. The combined firing of LFN and ESN sends the learning signal, and a one-shot learning is executed, adjusting weights on $N3$.

When the first symbol of the forth sequence, "*m*", is activated, there are three winners ($N1$, N_2 and N_3). MWDN fires and inhibits PN and LFN. When the second symbol "*i*" is activated, these three neurons still are winners. MWDN fires and inhibits PN and LFN. When the third symbol "*t*" is activated, $N2$ is the single winner with trained links. MWDN does not fire, and $N2$ correctly predicts the next symbol, "#". When the last symbol of this sequence, "#", is activated, ESN fires. Since LFN does not fire when the prediction is correct, LN does not fire. Therefore, no learning is needed for this last sequence.

Figure 4-14 shows the firing activities of neurons over the course of the above example. The model stored the four sequences in three output neurons (the second and the last sequence being stored in the same neuron: $N2$). One may noticed that this anticipation based model does not separate the learning and retrieval processes, which make this model more efficient for sequence learning and retrieval.

4.7 Discussion

The results reported in this chapter suggest that the presented temporal sequence learning model can be an essential element of an intelligent machine, and may be of interest for the construction of engineered devices. The use, in this model, of WTA and single neuron representations of symbols (so called “grandmother cells”) flies in the face of the widely held view that real brains use distributed representations, perhaps consisting of a few hundred neurons. The use of grandmother cells prevents the model from showing the “graceful degradation” of memory seen with loss of neurons from real brains. The model also seems too highly engineered to be an evolved device, and there is little in the real environment to serve as *start* or *end-of-sequence* signals. However, this model is only a simplification of the biologically plausible process in which the grandmother cells are replaced by groups of neurons firing synchronously, as it is suggested in [Wan 90b]. Groups of neurons organized in minicolumns are able to perform feature selection and classification in structures that resemble WTA operation of grandmother cells as demonstrated in [Luc 04]. In addition, sufficient computational resources should make it possible to render the present model in large associative networks that use sparsely coded, distributive representations. For example, as in the model presented here, R-nets [Vog 01] [Vog 05] implement a mechanism of associative recall that depends on the absence of inhibitory projections. The biological plausibility of the mechanism is argued in [Vog 05]. Nearly identical R-nets have already been assembled in a modular fashion to create the sort of “and”, “or”, and “if not” gates needed in the present model. As the function of an R-net depends chiefly on its position in the

network, the use of such networks would reduce the appearance of *ad hoc* engineering.

The model proposed in this dissertation research bears some resemblance to the incremental learning of sequences presented by Wang and Yuwono [Wan 95] [Wan 96]. However, there are several significant differences between the proposed model in this research compared to those in [Wan 95] and [Wan 96].

First, the proposed model uses a hierarchical approach that extends the concept of learning by chunking. I believe that the proposed sequence learning model is a more natural approach to sequence learning and allows a natural grouping of sequences within the context of learned events.

A second difference relates to the efficiency of learning. In [Wan 95] and [Wan 96], training requires several presentations of a complex sequence that includes repetitions of the same subsequence in different contexts. (For simple sequences in which no subsequence is repeated, only 1 training sweep is necessary.) The training phase is completed when there is no mismatch occurring during the last training sweep. In the proposed model, a complex sequence needs to be presented only once. The number of retraining sweeps required in [Wan 95] and [Wan 96] is a function of the sequence degree. The upper bound of the sweep number is $k \times (k+1)/2$, where k is the sequence degree defined as the shortest prior subsequence that uniquely defines the following sequence component. This upper bound is reduced to one in the proposed model without sacrificing computational efficiency or increasing the hardware cost.

The third difference is the hardware implementation structure. The proposed model use a memory-like storage of temporal events rather than the shift register structure proposed

in [Wan 95] and [Wan 96]. Instead of shifting all inputs as each new element of the sequence is presented to the system, each new element is stored at a specified location in IR in the proposed model. Accordingly, the hardware cost in Wang's work depends on the degree of complexity of the sequence while in the proposed method it depends on the sequence length. One can argue that the sequence degree is smaller than its length. However, the opposite is true for complex sequences hierarchically represented as sequences of elements representing chunked sequences. Take, for example, a sequence in which the same word is repeated more than once. The sequence degree is then at least equal to the length of this word, and if several words are repeated, to the combined length of these words. Chunking may help this to some degree but it introduces a requirement for additional training to accommodate chunks. However, this is not a natural approach adopted in biological neural systems. People are not capable of memorizing long sequences even if their order is low. Instead of learning a sentence as a long sequence of letters in which individual words are just part of the sequence, people learn sentences as sequences of words, learn stanzas as sequences of sentences, learn songs as sequences of stanzas, and learn stories as sequences of scenes.

What I have tried to preserve from Wang and Yuwono's approach [Wan 96] is its incremental learning of sentences that allows learning based on acquired knowledge rather than learning of all sequences at once. Incremental learning is natural for biological systems, where knowledge is built upon prior experience, modifies behavior, and influences further learning. Not all methods for learning temporal sequences preserve this important feature. For instance, the approach taken in [Wan 01] claims many advantages over the method presented in [Wan 96]. However, it is not an incremental learning model as it requires two-pass training.

In the first pass all temporal weights are computed and they affect the global threshold established in the second pass. Thus, whenever there is a change in learning sequence, the entire training process must be repeated.

The importance of incremental learning and the difficulty involved in obtaining such learning has been know at least since Grossberg [Gro 87]. Subsequently, McCloskey and Cohen identified the problem as the "catastrophic interference" [Mcc 89]. Sharkey characterized the catastrophic interference as one of forgetting previously learned information as new information is added [Sha 94]. There have been many attempts to solve the problem by minimizing the overlap between synaptic weights storing various sequences [Kor 90] [Fre 91] [Slo 92] with little success demonstrated for relatively small sequences. Catastrophic interference in incremental learning is chiefly the result of modification of weights representing all previously stored patterns. While some interference may be found in human memory [Bow 94], the catastrophic interference is not. In the proposed model in this dissertation, ideally there is no interference between stored patterns, as a learning increment involves only those neurons not previously involved in information storage. Only when the number of presented sequences exceeds the storage capacity of the proposed associative memory does partial interference appear. The catastrophic interference never happens in the proposed model in this dissertation.

Another important feature of Wang and Yuwono's approach [Wan 95] preserved in the proposed model is anticipation based learning in which the model actively anticipates the next input. As long as the anticipated inputs are correctly verified, no learning is required on any level of the hierarchy. When a mismatch is detected on a given level of the hierarchy, a

new sequence on this and all higher levels must be learned.

In addition, the model presented in this research uses one-shot, normalized, Hebbian learning such that the entire sequence is learned using only a single presentation of the input data. Any new sequence at any level of the hierarchy is learned by self-organization that allocates the required resources. The proposed approach can store arbitrarily complex sequences so long as the number of distinct subsequences is smaller than the storage capacity expressed by the total number of neurons on various hierarchy levels.

Hierarchical sequence recognition is also considered in [Doy 90] [Jen 90] where complex sequences are separated into simple sequences combined at higher hierarchical levels. In [Wan 95] hierarchical learning is used and requires a number of presentations of the training sequence before a higher level sequence can be learned. In the proposed approach in this dissertation, learning on all levels proceeds without repeated input presentations, and sequence reproduction does not require special handling. In particular, there is no need for the interval maintenance required in [Wan 95].

4.8 Summary

In this chapter, a novel temporal sequence learning model is presented. This model is characterized by hierarchical organization, anticipation mechanism and incremental learning. A modified Hebbian learning mechanism is proposed for input pattern recognition at the lowest level of the model. In each hierarchical level, a winner-take-all mechanism is used to select neurons that serve as the input for the next higher hierarchical level. The resulting

model can efficiently handle large scale, multiple, complex sequence learning and retrieval. Prediction is an essential component of the temporal sequence learning in the proposed model. By correct prediction, the machine indicates it knows the current sequence and does not require additional learning, otherwise, one-shot learning is executed. This chapter together with the Chapter 3 (learning) and Chapter 4 (memory) provide a complete framework for designing brain-like intelligent systems that are able to learn, memorize and predict information. From the next chapter, I will present different applications of the proposed systems as well as their hardware design architecture.

Chapter 5

Application Research

5.1 Introduction

This chapter explores different application problems for the proposed dynamically self-reconfigurable systems for machine intelligence. Based on the characteristics of the learning, memory and prediction capability of the proposed reconfigurable systems, they have a wide range of applications from classification and pattern recognition [He 06], data mining [Zhu 06] [Sta 03b], image recovery [Sta 06b], to sequence learning and prediction [Sta 06c].

5.2 Self-organizing learning array for power quality classification

The first application is the power quality (PQ) classification problem [He 06]. PQ is recently becoming prevalent and of critical importance for power industry. According to the results of the investigating by the Electrical Power Research Institute, the US economy is losing between $104 billion and $164 billion a year to power outages, and another $15 billion to $24 billion to PQ phenomena [Pri 01]. Artificial intelligence (AI) and machine learning is one of the powerful tools used to deal with the power quality classification problems. Ibrahim et al. presented a survey of the advanced AI techniques for PQ applications [Ibr 02]. The most interesting AI tools for PQ problems include expert systems [San 00a], fuzzy logic [Hua 02a] [Elm 00] [Elm 01] [Far 02], artificial neural networks [Elm 01] [San 00b] [San 00c] [Wij 02]

and genetic algorithms [Kun 98]. As a novel learning scheme, SOLAR has the potential to be a more powerful tool for the PQ classification application. In this research, I first use wavelet transform to extract feature vectors for various PQ disturbances based on the multiresolution analysis (MRA), then this data sets are sent to the SOLAR system for learning and classification.

5.2.1 Wavelet based feature extraction

The mathematics of the wavelet transform have been extensively studied and can be referenced in [Dau 92] and [Bur 98]. Multiresolution analysis was introduced by Mallat and a detailed study about MRA can be found in [Mal 89]. Briefly speaking, in multiresolution analysis, any time series can be completely decomposed in terms of the approximations provided by scaling functions, and the details provided by the wavelets function. The decomposition procedure starts with passing the signal through filters banks. The approximations are the low-frequency components of the time series and the details are the high-frequency components. In this way, the decomposition of a signal $x(t)$ can be illustrated as in Figure 5-1, where $A_i(t)$ is called the approximation at level i, and $D_i(t)$ is called the detail at level i, and f_s is the sample frequency.

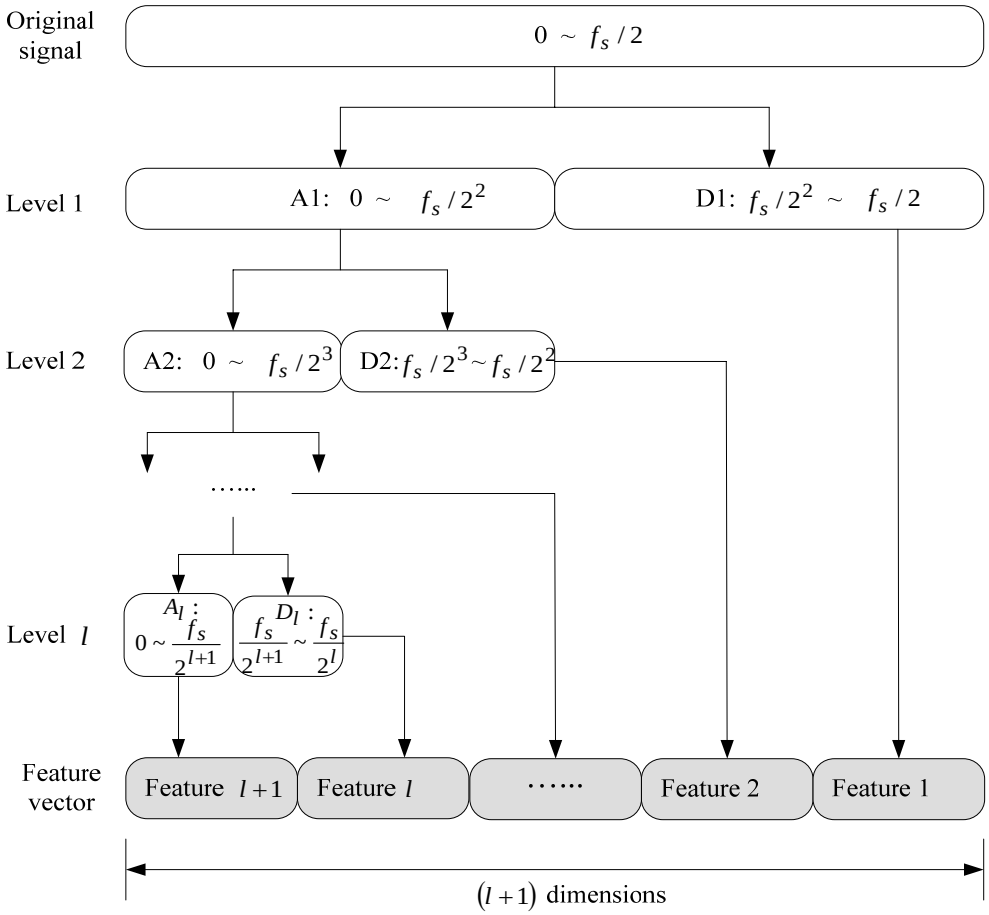


Figure 5-1 MRA analysis and feature extraction

In order to reduce the feature dimensions, signal energy at each decomposition level is used as a new input feature for SOLAR classification. The energy at each decomposition level is calculated using the following equations:

$$ED_i = \sum_{j=1}^{N} |D_{ij}|^2, i = 1...l \tag{5.1}$$

$$EA_l = \sum_{j=1}^{N} |A_{lj}|^2 \tag{5.2}$$

where $i = 1...l$ is the wavelet decomposition level from level 1 to level l. N is the number of the coefficients of detail or approximation at each decomposition level. ED_i is

the energy of the detail at decomposition level i and EA_l is the energy of the approximation at decomposition level l. In this way, for a l level wavelet decomposition, a $(l+1)$ dimensional feature vector is created for SOLAR analysis. Figure 5-2 shows the data flow in the proposed wavelet feature extraction.

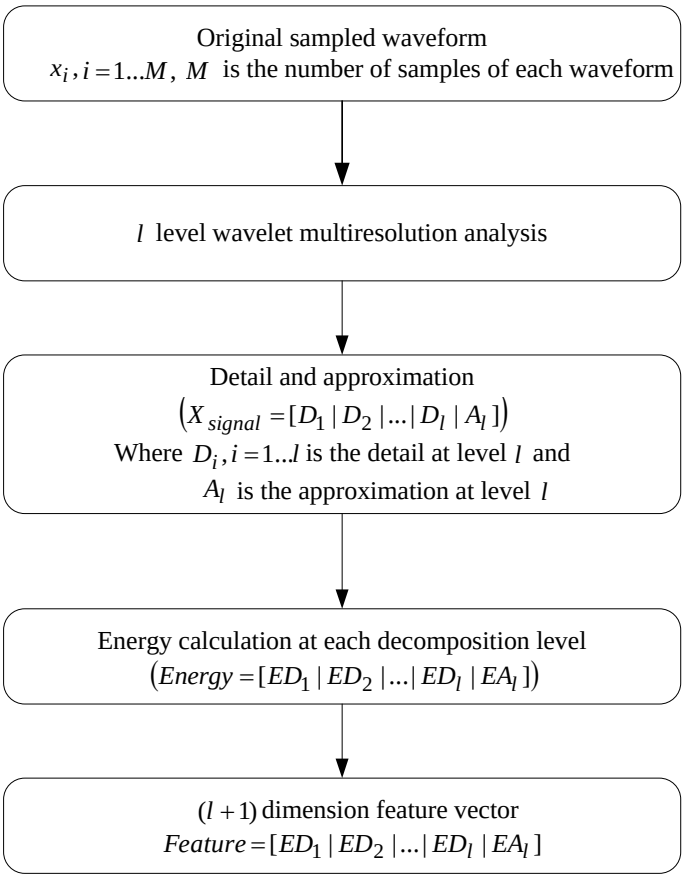


Figure 5-2 Wavelet based feature extraction

5.2.2 Simulation and analysis

5.2.2.1 Data generation

The simulation data is generated based on the model in paper [Gal 04]. Seven classes ($C1 \sim C7$) of different PQ disturbances are considered: undisturbed sinusoid (normal), swell, sag, harmonics, outage, sag with harmonic, and swell with harmonic. Table 5-1 gives the signal generation model and their control parameters. Two hundred cases of each class with different parameters are generated for training and another 200 cases are generated for testing. Both the training and testing signals are sampled at 256 points/cycle (same as in reference [Gal 04] for results comparison) and the normal frequency is $50Hz$. Ten power frequency cycles which contain the disturbance is used for a total of 2560 points.

Table 5-1 Power quality distance model [Gal 04]

PQ disturbances	Class Symbol	Model	Parameters
normal	C1	$x(t) = \sin(\omega t)$	N/A
Swell	C2	$x(t) = A(1+\alpha(u(t-t_1)-u(t-t_2)))\sin(\omega t)$ $t_1 < t_2, u(t) = \begin{cases} 1, t \geq 0 \\ 0, t < 0 \end{cases}$	$0.1 \leq \alpha \leq 0.8$ $T \leq t_2 - t_1 \leq 9T$
Sag	C3	$x(t) = A(1-\alpha(u(t-t_1)-u(t-t_2)))\sin(\omega t)$	$0.1 \leq \alpha \leq 0.9; T \leq t_2 - t_1 \leq 9T$
Harmonic	C4	$x(t) = A(\alpha_1 \sin(\omega t) + \alpha_3 \sin(3\omega t) + \alpha_5 \sin(5\omega t) + \alpha_7 \sin(7\omega t))$	$0.05 \leq \alpha_3 \leq 0.15, 0.05 \leq \alpha_5 \leq 0.15$ $0.05 \leq \alpha_7 \leq 0.15, \sum {\alpha_i}^2 = 1$
Outage	C5	$x(t) = A(1-\alpha(u(t-t_1)-u(t-t_2)))\sin(\omega t)$	$0.9 \leq \alpha \leq 1; T \leq t_2 - t_1 \leq 9T$
Sag with Harmonic	C6	$x(t) = A(1-\alpha(u(t-t_1)-u(t-t_2)))$ $(\alpha_1 \sin(\omega t) + \alpha_3 \sin(3\omega t) + \alpha_5 \sin(5\omega t))$	$0.1 \leq \alpha \leq 0.9, T \leq t_2 - t_1 \leq 9T$ $0.05 \leq \alpha_3 \leq 0.15, 0.05 \leq \alpha_5 \leq 0.15; \sum {\alpha_i}^2 = 1$
Swell with Harmonic	C7	$x(t) = A(1+\alpha(u(t-t_1)-u(t-t_2)))$ $(\alpha_1 \sin(\omega t) + \alpha_3 \sin(3\omega t) + \alpha_5 \sin(5\omega t))$	$0.1 \leq \alpha \leq 0.9, T \leq t_2 - t_1 \leq 9T$ $0.05 \leq \alpha_3 \leq 0.15, 0.05 \leq \alpha_5 \leq 0.15; \sum {\alpha_i}^2 = 1$

5.2.2.2 Simulation results

Daubechies 4 (Db4) wavelets with 10 levels of decomposition are used for analysis ($l = 10$). Based on the feature extraction shown in Figure 5-2, 11-dimensional feature sets for training and testing data are created. The dimensions describe different features resulting from the wavelet transform, that is to say, the total size of the training data or testing data set is 1400×11, where 1400 comes from 200 cases per class multiplied by 7 classes, and 11 is the feature size dimension of each case. All data sets are scaled to the range of (1~255) before applied to SOLAR for training and testing.

Figure 5-3 and Figure 5-4 shows two 2-dimensional projections of the training set. As we can see from these two figures, in some dimensions, such as dimension 8 and dimension 10 shown in Figure 5-3, the data sets are mixed up, while in some other dimensions, such as dimension 8 and dimension 6 shown in Figure 5-4, the data sets are better separated. Since SOLAR will dynamically choose its functionality and connection structure in its learning stage, it can easily handle this kind of data classification problem.

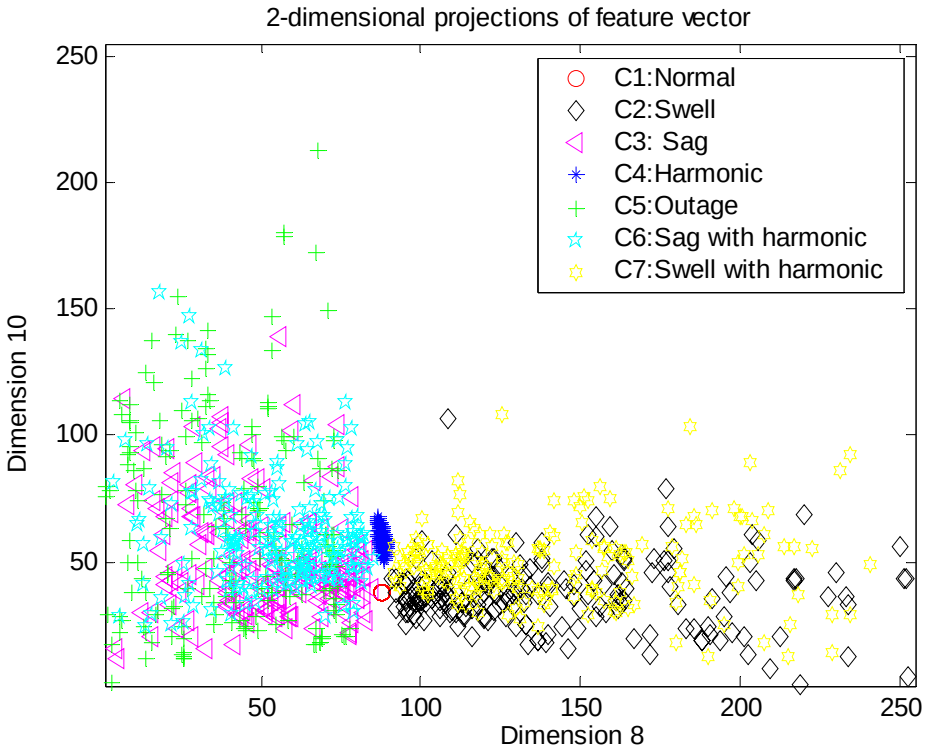


Figure 5-3 Two dimensional projections of the feature vector: dimensional 8 and 10

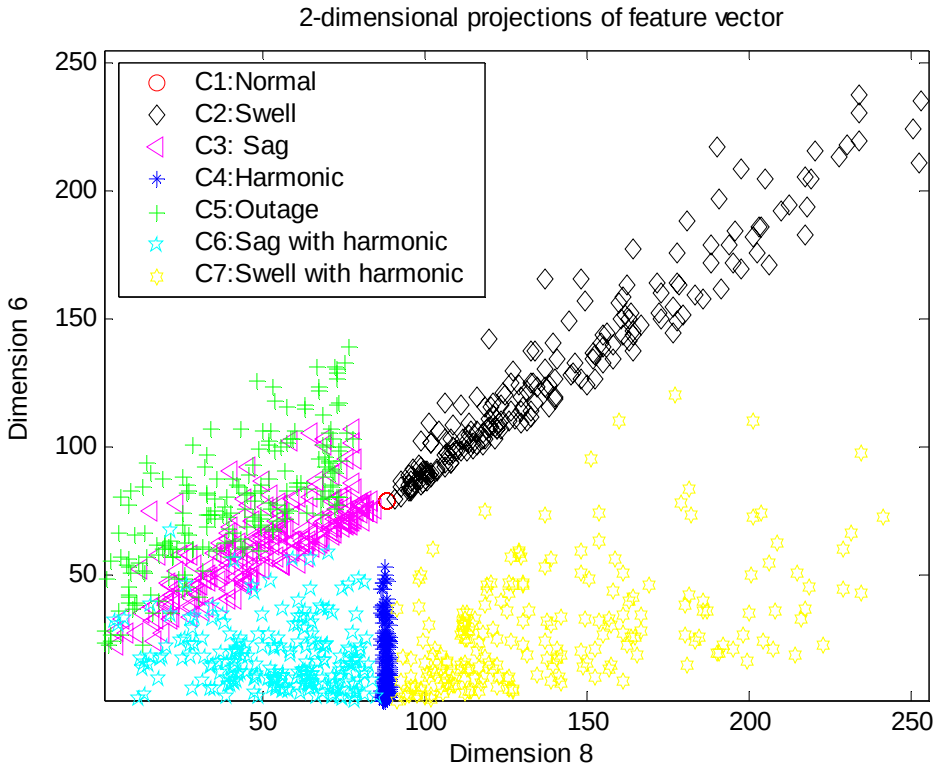


Figure 5-4 Two dimensional projections of the feature vector: dimensional 8 and 6

In order to evaluate the performance of the proposed method, three classification results are compared: inductive interface approach [Gal 04], support vector machine (SVM) and SOLAR. Simulation of the SVM for classification is based on the modification of the Ohio State University SVM Classifier Matlab Toolbox [Ohi]. Two types of SVM are implemented in this research: C -support vector classification and ν - support vector classification. For each type of SVM, the following four kernel functions $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$, where x_i and x_j are the feature vectors in the input space (i and j denotes the index of the features) and ϕ is the mapping function, are taken into account:

Linear:

$$K(x_i, x_j) = x_i^T x_j \tag{5.3}$$

Polynomial:

$$K(x_i, x_j) = \left(\gamma x_i^T x_j + r\right)^d, \gamma > 0 \tag{5.4}$$

Radial Basis Function (RBF):

$$K(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \gamma > 0 \tag{5.5}$$

Sigmoid:

$$K(x_i, x_j) = \tanh\left(\gamma x_i^T x_j + r\right) \tag{5.6}$$

where γ, r and d are kernel parameters. The detailed discussion about these types of SVMs and the kernel functions can be referred in [Sch 00] [Vap 98] [Cor 95].

Table 5-2, Table 5-3 and Table 5-4 give the simulation results for this 7-class PQ disturbance classification problem based on SOLAR, inductive inference approach [Gal 04], and SVM classification, respectively.

Table 5-2 Classification results for SOLAR based on wavelet feature extraction

	C1	C2	C3	C4	C5	C6	C7
C1	200	0	0	0	0	0	0
C2	0	200	0	0	0	0	0
C3	1	0	174	0	24	1	0
C4	0	0	0	200	0	0	0
C5	15	0	16	0	161	8	0
C6	0	0	2	1	2	194	1
C7	0	0	0	0	0	0	200
Overall accuracy				***94.93%***			

Table 5-3 Classification results reported in [Gal 04]: Inductive interface approach

	C1	C2	C3	C4	C5	C6	C7
C1	200	0	0	0	0	0	0
C2	0	194	0	0	0	0	6
C3	0	0	153	0	11	36	0
C4	0	0	0	200	0	0	0
C5	0	0	1	0	180	19	0
C6	0	0	42	0	15	143	0
C7	0	4	0	0	0	0	196
Overall accuracy				***90.4%***			

Table 5-4 Classification results based on SVM

	C-SVM								ν-SVM							
Linear kernel function		C1	C2	C3	C4	C5	C6	C7		C1	C2	C3	C4	C5	C6	C7
	C1	200	0	0	0	0	0	0	C1	200	0	0	0	0	0	0
	C2	32	168	0	0	0	0	0	C2	43	157	0	0	0	0	0
	C3	36	0	132	0	31	1	0	C3	55	0	113	0	31	1	0
	C4	0	0	0	200	0	0	0	C4	0	0	0	177	0	10	13
	C5	0	0	11	0	189	0	0	C5	3	0	13	0	184	0	0
	C6	0	0	1	0	10	189	0	C6	0	0	1	0	15	184	0
	C7	0	0	0	0	0	0	200	C7	0	0	0	0	0	0	200
	Overall accuracy	**91.29%**							Overall accuracy	**86.79%**						
Polynomial Kernel function		C1	C2	C3	C4	C5	C6	C7		C1	C2	C3	C4	C5	C6	C7
	C1	200	0	0	0	0	0	0	C1	200	0	0	0	0	0	0
	C2	31	169	0	0	0	0	0	C2	44	156	0	0	0	0	0
	C3	36	0	140	0	23	1	0	C3	54	0	117	0	28	1	0
	C4	0	0	0	196	0	4	0	C4	0	0	0	155	0	45	0
	C5	0	0	21	0	179	0	0	C5	3	0	18	0	179	0	0
	C6	0	0	2	0	7	191	0	C6	0	0	1	0	14	185	0
	C7	0	0	0	0	0	0	200	C7	0	0	0	0	0	0	200
	Overall accuracy	**91.07%**							Overall accuracy	**86.14%**						
RBF kernel function		C1	C2	C3	C4	C5	C6	C7		C1	C2	C3	C4	C5	C6	C7
	C1	200	0	0	0	0	0	0	C1	200	0	0	0	0	0	0
	C2	12	188	0	0	0	0	0	C2	38	162	0	0	0	0	0
	C3	17	0	158	0	23	2	0	C3	52	0	121	0	26	1	0
	C4	0	0	0	200	0	0	0	C4	0	0	0	181	0	0	19
	C5	0	0	7	0	193	0	0	C5	0	0	9	0	191	0	0
	C6	0	0	3	0	11	186	0	C6	0	0	1	0	24	175	0
	C7	0	0	0	0	0	0	200	C7	0	0	0	0	0	0	200
	Overall accuracy	**94.64%**							Overall accuracy	**87.86%**						
Sigmoid kernel function		C1	C2	C3	C4	C5	C6	C7		C1	C2	C3	C4	C5	C6	C7
	C1	200	0	0	0	0	0	0	C1	200	0	0	0	0	0	0
	C2	0	200	0	0	0	0	0	C2	39	161	0	0	0	0	0
	C3	0	0	171	0	28	1	0	C3	55	0	112	0	32	1	0
	C4	0	0	0	200	0	0	0	C4	0	0	0	179	0	0	21
	C5	0	0	33	0	164	3	0	C5	3	0	10	0	187	0	0
	C6	0	0	1	0	6	193	0	C6	0	0	2	0	15	183	0
	C7	0	0	0	0	0	0	200	C7	0	0	0	0	0	0	200
	Overall accuracy	**94.86%**							Overall accuracy	**87.29%**						

For the results in Table 5-2, Table 5-3 and Table 5-4, a 7×7 confusion matrix C is constructed for each method to show the classification performance. The diagonal elements represent the correctly classified PQ types. The off-diagonal elements represent the misclassification. As illustrated in these three tables, the SOLAR learning mechanism can effectively classify different kinds of PQ disturbances. One thing that should be noted here is that, it is recognized that an optimum selection of the SVM type, kernel function and parameter setting may reach better results than those listed in Table 5-4. However, it is not easy to choose the optimum SVM kernel function and parameters in advance. SOLAR can dynamically configure its connectivity structure and select its function according to the information that individual neurons receive, therefore it can automatically provide statistically stable results. Further discussion about the selection of SVM kernel function and parameters can be found in papers [Sch 00] [Vap 98] [Cor 95].

5.2.2.3 Discussion

5.2.2.3.1 Classification performance and wavelet decomposition levels

As discussed in Section 5.2.1, l levels wavelet decomposition will create a $(l+1)$ dimensional feature vector for SOLAR system. Obviously, more levels of decomposition will increase the computational cost. So, how to choose a reasonable number of decomposition levels?

In this simulation, the Db4 wavelet is selected and its decomposition levels are scanned from 1 to 10. For each type of PQ class (C1 to C7 as in Table 5-1), 200 cases are generated with different parameters for training and another 200 cases are generated for testing. Sample frequency is 256 points/cycle. Figure 5-5 gives the simulation results.

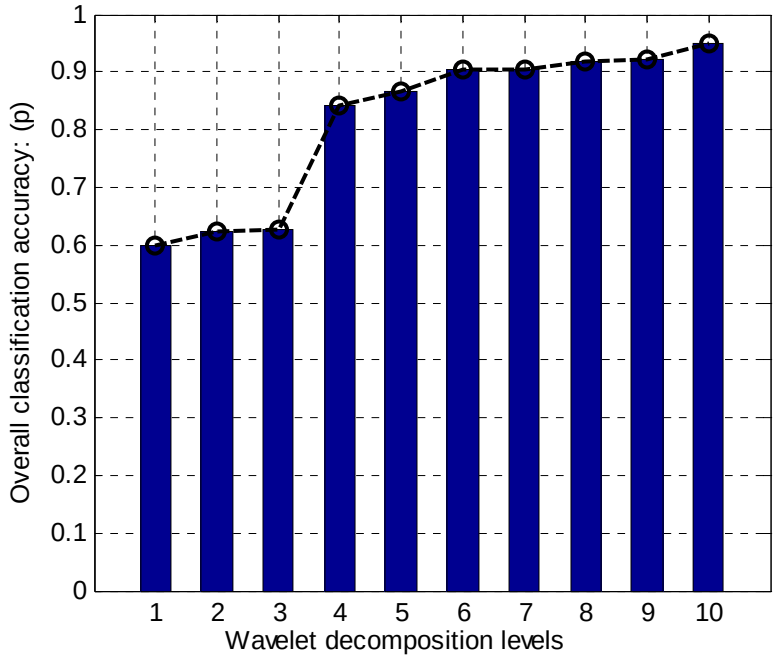


Figure 5-5 Relationship between the wavelet decomposition levels and classification accuracy

As we can see from Figure 5-5, when the wavelet decomposition levels are relatively small, such as $l \le 3$, the overall classification accuracy is about 60%. When the decomposition level is larger then 6, the classification accuracy can reach about 90%. Further investigation shows that when $l \ge 6$, the increase of the classification accuracy is small. Figure 5-5 provides a reference for the trade off between the decomposition levels and classification performance.

5.2.2.3.2 Choosing a suitable wavelet

Another import issue is the choice of a suitable wavelet. Obviously, the longer the wavelet filter length, the higher the computational cost. In this part, the influence of different kinds of wavelets on the classification accuracy is investigated.

Conjecture 1: No single wavelet transform has a statistically significant advantage over other wavelets on the performance of the proposed method for PQ classification.

To verify conjecture 1, four commonly used wavelets, the Haar wavelet, Daubechies wavelets, Symlets, and Coiflets wavelet are taken into account. Table 5-5 shows their corresponding characteristics. The detailed discussion about these wavelet characteristics can be found in [Dau 92] and [Bur 98].

Table 5-5 Wavelet characteristics

Wavelet Name	Orthogonal	Compact support	Support Width	Filters length	Symmetry
Haar	Yes	Yes	1	2	Yes
Daubechies	Yes	Yes	2N-1	2N	Far from
Coiflets	Yes	Yes	6N-1	6N	Near from
Symlets	Yes	Yes	2N-1	2N	Near from

In order to evaluate the performance of different kind of wavelets, the number of decomposition levels are fixed at 6. The reason that 6 levels of decomposition is selected here is based on the results in Figure 5-5, in which we can see that 6 levels decomposition can provide nice classification accuracy (about 90%) as well as relatively small computational cost compared to higher levels of decomposition, such as 9 levels or 10 levels. Although these

additional decomposition levels can provide slightly better classification accuracy, they come with extra computing cost. The trade off between the decomposition levels and classification performance is also discussed in Section 5.2.2.3.1. For each choice of wavelet, 10 simulation runs is conducted to show the statistical performance. In each run, 200 cases of each class (7 classes total) of PQ disturbances based on the model in Table 5-1 are generated for training and another 200 cases of each class are generated for testing. The average accuracy over the 10 runs for each choice of wavelet are presented in Table 5-6.

Table 5-6 Averaged classification results for different wavelets

Wavelet name	Haar	Db2	Db3	Db4
Average accuracy	0.8700	0.8557	0.8746	0.8908
Wavelet name	Db5	Db6	Db7	Db8
Average accuracy	0.8909	0.8786	0.8784	0.8776
Wavelet name	Db9	Db10	Coif1	Coif2
Average accuracy	0.8996	0.8882	0.8903	0.8921
Wavelet name	Coif3	Coif4	Coif5	Sym2
Average accuracy	0.8819	0.8694	0.8883	0.8646
Wavelet name	Sym3	Sym4	Sym5	Sym6
Average accuracy	0.8856	0.8692	0.8644	0.8755
Wavelet name	Sym7	Sym8		
Average accuracy	0.8894	0.8650		

To test if there is any significant difference among different wavelet families, or among different wavelets within one wavelet family, hypothesis testing of the means [Mil 65]

is used. Table 5-7 gives the mean and standard deviation results of each family of wavelet as presented in Table 5-6. The mean, μ, and the standard deviation, σ, of the population are calculated using the following equations:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} p_i \tag{5.7}$$

$$\sigma = \sqrt{\frac{n\sum_{i=1}^{n} p_i^2 - \left(\sum_{i=1}^{n} p_i\right)^2}{n(n-1)}} \tag{5.8}$$

where p_i is the classification accuracy of different wavelets in each wavelet family in Table 5-6 and n is the number of wavelets in each family.

Table 5-7 Performance of different wavelets

Wavelet	p	Wavelet	p	Wavelet	p
Haar (Db1)	0.8700	Coif1	0.8903	Sym2	0.8646
Db2	0.8557	Coif2	0.8921	Sym3	0.8856
Db3	0.8746	Coif3	0.8819	Sym4	0.8692
Db4	0.8908	Coif4	0.8694	Sym5	0.8644
Db5	0.8909	Coif5	0.8883	Sym6	0.8755
Db6	0.8786			Sym7	0.8894
Db7	0.8784			Sym8	0.8650
Db8	0.8776				
Db9	0.8996				
Db10	0.8882				
Mean(μ)	**0.8804**		**0.8844**		**0.8734**
Standard deviation (σ)	**0.0125**		**0.0092**		**0.0105**

Null hypothesis:

$$H_0 : \mu_1 = \mu_2 \tag{5.9}$$

Alternative hypothesis:

$$H_1 : \mu_1 \neq \mu_2 \tag{5.10}$$

The test statistic is calculated as follows:

$$Z = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{5.11}$$

For a two-tailed test, H_0 will be rejected if $|Z| > 1.96$. (1.96 is for a two-tailed test where the results are significant at a level of 0.05). Table 5-8 shows the hypothesis testing result.

Table 5-8 Wavelet family hypothesis test

Wavelet name	Wavelet name	$\|Z\|$	Accept or reject H_0
Daubechies	Coiflets	0.7011	Accept
Daubechies	Symlet	1.2497	Accept
Coiflets	Symlet	1.9243	Accept

From the analysis results in Table 5-8, it can be seen that all $|Z|$ values are less than 1.96, thus, the null hypothesis H_0 will be accepted. This means that there is no statistically significant difference in the classification performance when different wavelet families are chosen. The same test can be performed for different wavelets within one wavelet family and the same results can be concluded. Figure 5-6 shows the analysis results for different wavelets within one wavelet family, namely Daubechies wavelets and their corresponding wavelet filter length.

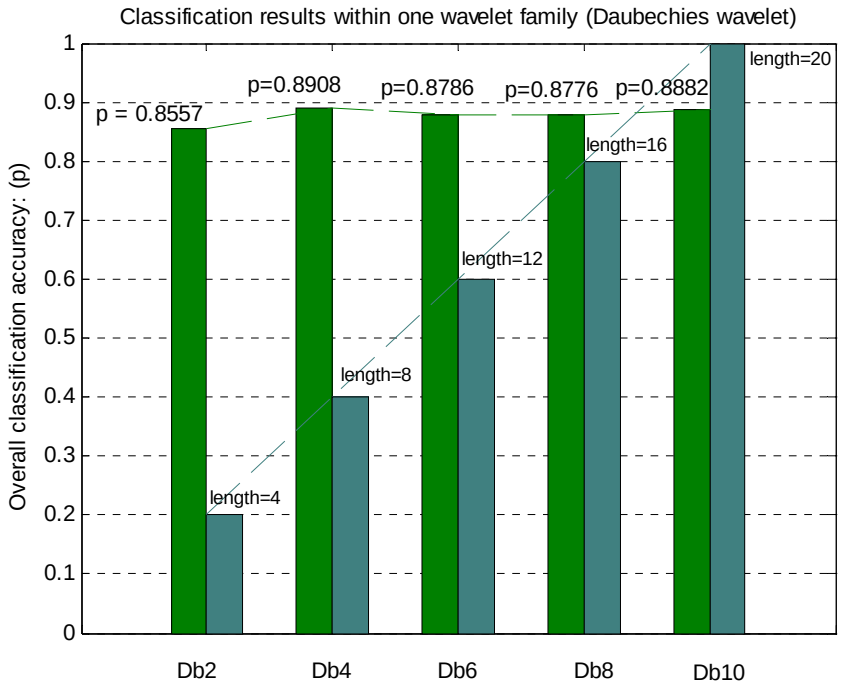


Figure 5-6 Classification results and their corresponding wavelet filter length

So far, it is tested that there is no statistically significant difference in performance of the proposed method for PQ classification when different wavelets are chosen. Since the longer the wavelet filter length, the larger the computational cost, one can choose the wavelet with a short wavelet filter length, such as the Haar wavelet or Db2 wavelet to achieve both good classification results as well as small computational cost.

5.2.3 Noise analysis

Since noise is omnipresent in an electrical power distribution network, in this section, I now analyze whether the proposed method is still effective in a noisy environment.

Gaussian white noise is widely considered in the research of power quality issues [Yan 01] [She 02]. To test the proposed method's performance in different noise environment,

different levels of noise with the signal to noise ratio (SNR) values ranging from 10 to 50 dB are considered here. The value of SNR is defined as following

$$SNR = 10\log(P_s / P_n)\ \text{dB} \tag{5.12}$$

where P_s is the power of the signal and P_n is that of the noise. The Monte-Carlo method is used to generate the simulation data set with different parameters as shown in Table 5-1. The Db4 wavelet with 6 levels of decomposition and 10 levels of decomposition is used in this simulation. Figure 5-7 shows the simulation results. Even in very low SNR conditions, the proposed method still can achieve high overall classification accuracy. Ten levels of decomposition classification result is slightly better than that of 6 levels of decomposition, but the improvement is not very great. This is consistent with the previous results as shown in Figure 5-5.

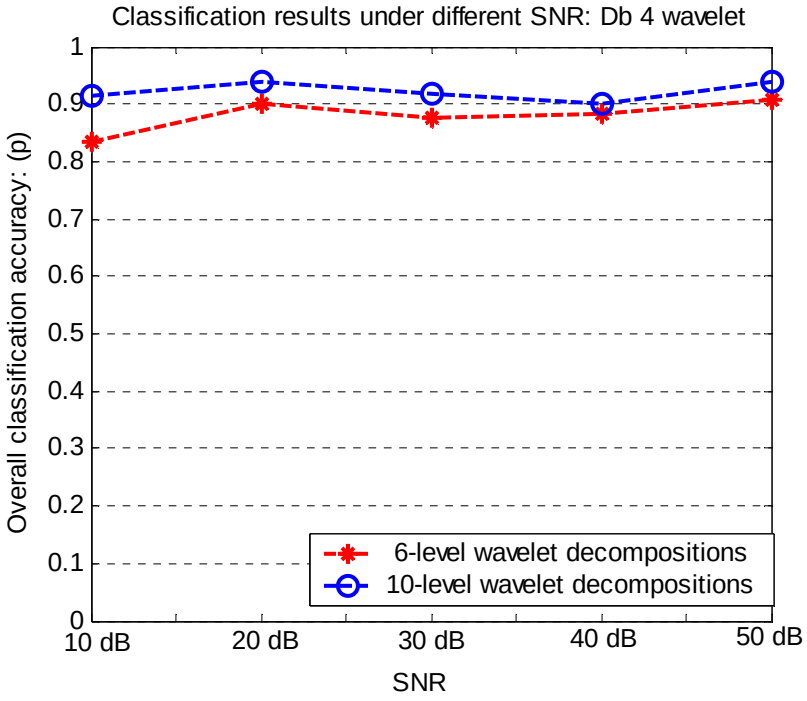


Figure 5-7 Classification results under different SNR condition

Figure 5-8 shows the overall classification performance for different wavelets in noisy environments. Ten levels of wavelet decomposition is used here. As illustrated in Figure 5-8, the SOLAR based method has robust anti-noise performance and it still can achieve high classification accuracy in a noisy environment. In addition, although Db4 wavelets show slightly better classification results among the chosen wavelets, there is no statistically significant difference in performance of the proposed method with different wavelets. This is also consistent with the previous results in Section 5.2.2.3.2.

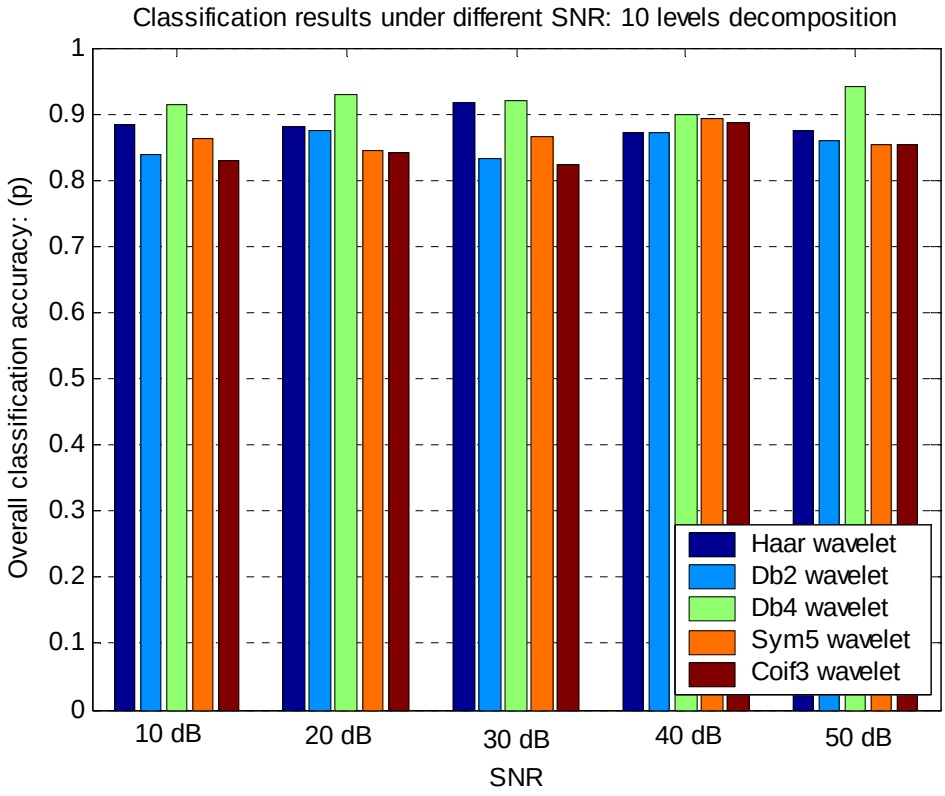


Figure 5-8 Different wavelet classification results under different SNR conditions

5.3 Self-organizing associative memory for pattern recognition and image recovery

As discussed in Chapter 3, the proposed self-organizing associative memory is a general memory network capable of both hetero-association and auto-association. In this section, both classification and image recovery applications are utilized to show the capability of the proposed associative memory applied to practical problems [Sta 06b].

5.3.1 Hetero-associative memory: Iris database classification

The Iris database [Fis 36] is used to test the classification performance of the proposed associative memory. This database has 3 classes (Iris Setosa, Iris Versicolour and Iris Virginica) and 4 numeric features (sepal length, sepal width, petal length, and petal width).

The N-bits sliding bar coding mechanism presented in [Sta 06a] is used in this research to code the input data. Assume that the maximum and minimum values to be coded are $V_{\max}$ and $V_{\min}$, respectively. The length of the sliding bar is defined by $N - L = V_{\max} - V_{\min}$. Assume that the value of the scaled feature to be coded is V. In the coded input, bits numbered from $(V - V_{\min}) + 1$ to $(V - V_{\min}) + L$ are set to 1s, while the remaining bits are set to 0s. This scheme is illustrated in Figure 5-9.

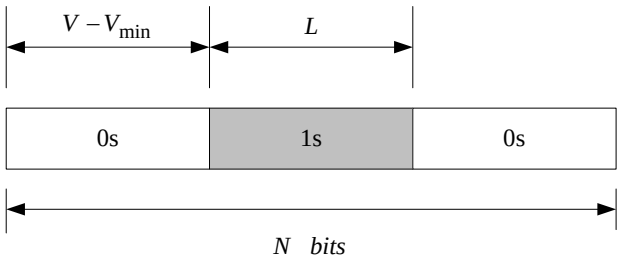


Figure 5-9 N input bits sliding bar coding mechanism [Sta 06a]

The class ID is coded in a similar way by using M bit code redundancy. Since there are 3 classes in this database, total number of $M \times 3$ bits are used to code the class ID, maximizing their Hamming distance. This is achieved by filling the M bits from position $(C_i - 1) \times M$ to $C_i \times M$ with 1's, while filling the remaining $M \times 2$ bits with 0's. Here $C_i = 1,2$ and 3 for this 3 classes Iris database. In this simulation, N is set to 80, L is set to 20 and M is set to 30. Such coding is compatible with a binary neural network in which neurons either fire or do not, and all activation signals are binary. While it may be more numerically efficient to represent signals as continuous values, this is not biologically plausible.

In the training stage, both the feature code and class ID code are presented to the associative memory. This information is used to discover the potential relationship in PEs' input spaces. In testing, only the feature code is presented to the input layer, and the class ID code is filled with undefined values. Through the feedback mechanism, the network makes associations and decides the class ID code values. After the class ID code values are decided, the system votes on the class ID of each testing instance according to the minimum Hamming distance to code values of all the possible classes. The classification accuracy is then

calculated by the ratio of correctly classified instances over the total number of testing instances.

Since there are only 150 instances in the Iris database, the N-fold cross validation method is used to handle this small sample dataset as used in [Hon 00] [Das 80] [Qui 87] [Lee 01] [Cha 04a]. In N-fold cross validation, all instances are randomly divided into N subsets of as nearly equal size as possible. In each trial, one of the N subsets is used as the testing set and the other $(N-1)$ subsets are used as training sets. Therefore, totally N trials are necessary in order for each instance to be tested once. The final classification accuracy is calculated by averaging results across all N trials. In this simulation, N is set to 10 (10-fold cross validation). Based on the description in Chapter 3, the proposed memory network self-organizes its structure, decide the necessary depth of the feedback association. Figure 5-10 (a) shows the associative PEs and their connection structure, and Figure 5-10 (b) shows the associative PE firing activity for part of the network. The Y-axis represents the input bits, and the X-axis represents the distance from the input (association depth). The associative PEs are represented by circles and their backward propagation paths are marked. The large dots at the input layer represent correctly recognized class ID code bits. It may be noted that only six layers are needed for the network to learn the associations in the Iris database.

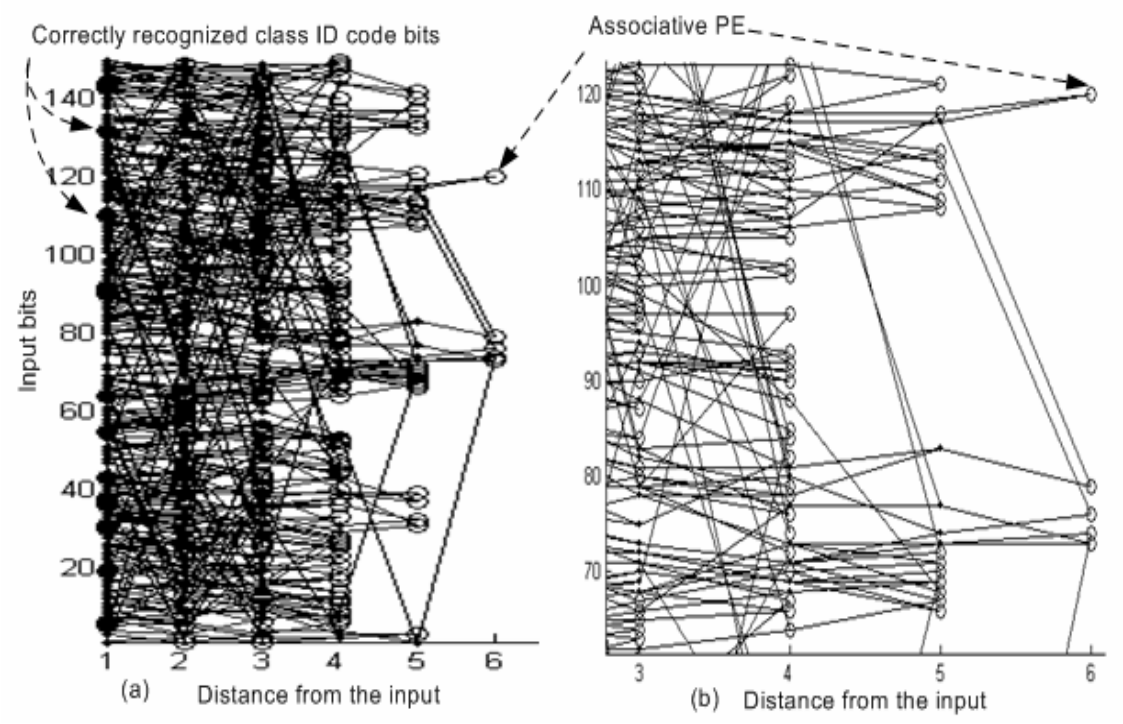


Figure 5-10 Associative PEs and their inter connection structure

Table 5-9 shows the classification performance of the proposed associative memory compared with some other classification performances reported in literature using the same database. These results indicate that the self-organizing associative memory presented in this dissertation shows satisfactory performance in a classification problem, satisfying necessary elements of learning through associations. Learning through associations is useful if one wants to associate specific behavior in a reinforcement learning value system.

Table 5-9 Comparison of the classification performance on Iris database

Method	Average classification accuracy
Merging-membership-functions-first [Hon 00]	97.33%
C4 method [Hon 00] [Qui 87]	93.87%
Influential Rule Search Schemes [Cha 04a]	96.00%
Dasarathy's pattern-recognition approach [Hon 00] [Das 80]	94.67%
Fuzzy entropy-based fuzzy classifier [Lee 01]	96.7%
Self-organizing associative memory	96.00%

5.3.2 Auto-associative memory: image recovery

Image recovery problems can be used to test the effectiveness of the proposed memory for auto-associative applications. This is necessary for applications where only partial images are available without specifying class identities. The proposed SOAM network can learn features of the training data using unsupervised learning, self-determine the feedback depth, and make correct associations to recover the original images.

5.3.2.1 Panda image recovery

The 64×64 binary panda image used in [Dju 02] is used in this research. The panda image is represented by a vector $p_i = (x_1 \quad x_2 \quad ... \quad x_n), n = 4096$, with $x_i = 1$ for a black pixel and $x_i = 0$ for a white pixel. In testing, $r\%$ percentage ($r = 10,20$ and 30) of the

panda image is randomly blocked by setting $n \times (r\%)$ pixels to the undefined value (0.5). The original panda image and samples of its blocked image are shown in Figure 5-11 (a) and (b), respectively. Figure 5-11 (c) shows images recovered through the proposed associative memory. As in [Dju 02], the image recovery performance is evaluated by computing the ratios of the number of incorrectly recovered pixels (both erroneous pixels and pixels remaining undefined after recovery) over the total number of pixels. As one can see in Table 5-10, the SOAM model can provide results comparable to those in [Dju 02].

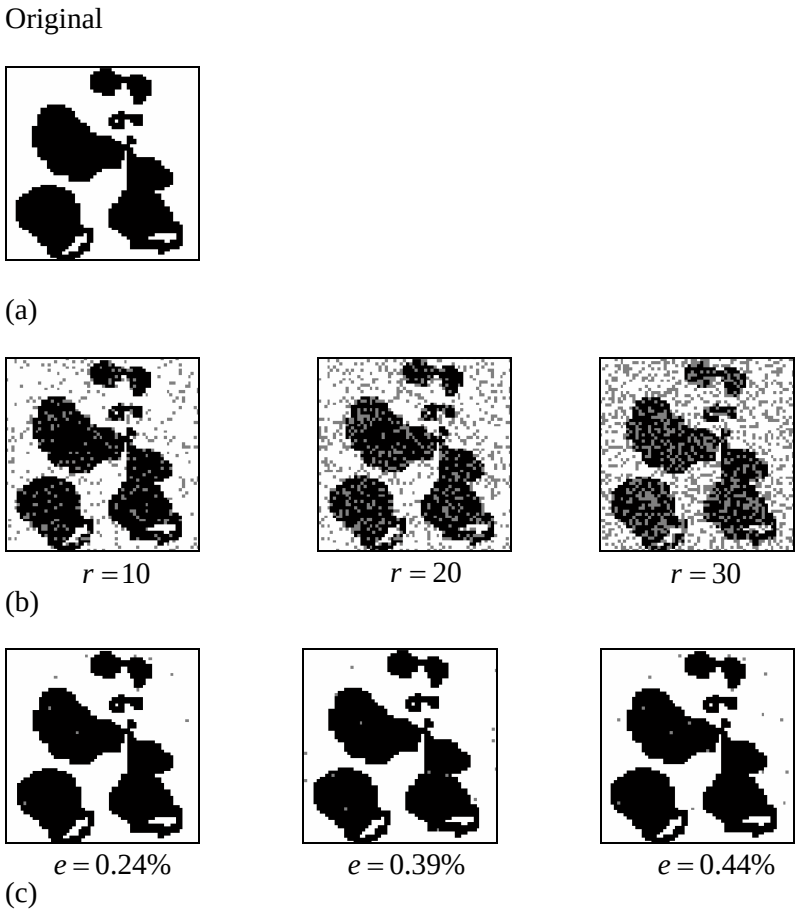


Figure 5-11 The 64x64 binary panda image
(a) The original training image;
(b) Blocked image with r% of undefined values (r = 10, 20 and 30 respectively);
(c) Recovered image and the recovery error.

Table 5-10 Image recovery error information

Noise/block level	10%	20%	30%
Reference [Dju 02]	2.95%	4.83%	6.57%
SOAM	0.24%	0.39%	0.44%

While Figure 5-11 tested the capability of the self-organizing associative memory in recovering randomly blocked images, Figure 5-12 shows the recovery performance under the condition of the entire lower half of the panda image being blocked. In this case, the recovery error bits are at the level of 2.42% of the total image.

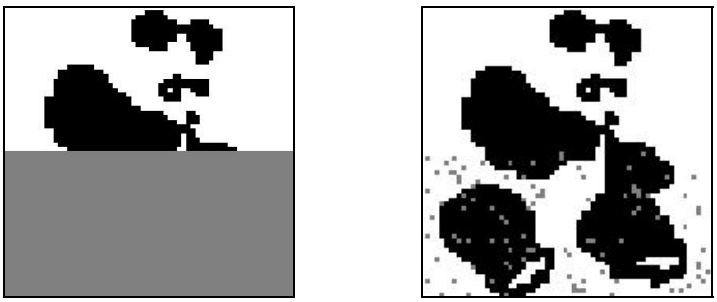


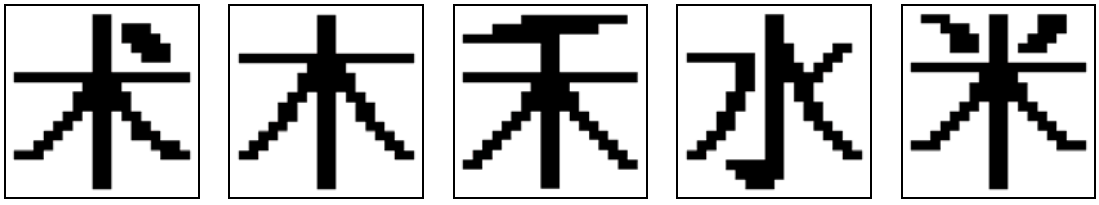
Figure 5-12 Testing image (block half) and recovered image

5.3.2.2 Chinese character recognition and recovery

Chinese character recognition is considered as a very challenging problem due to a couple of reasons discussed in [Won 98]. First, the Chinese vocabulary is extremely large. Second, many Chinese characters look very similar to each other. And third, the Chinese characters themselves are very complex. In [Wu 00] and [Wu 01], a local identical index

associative memory algorithm is proposed for Chinese character recognition. On a 16 Chinese prototype patterns data set, the proposed method achieved about 97.3% [Wu 00] and 100% [Wu 01] recognition accuracy. In [Fu 98], H. C. Fu et al. presented a Bayesian decision-based neural network (BDNN) for multilinguistic handwritten character recognition. BDNN is a self-growing probabilistic decision-based neural network and adopts a hierarchical network structure with nonlinear basis functions and a competitive credit-assignment scheme. Simulation results on three different Chinese character databases show the recognition rate around $86 \sim 94\%$. Most of the references in literature solely handle the Chinese character recognition problem. Since the proposed self-organizing associative memory is capable of both classification and image recovery, the Chinese character recognition and recovery is illustrated in this section.

Figure 5-13 shows the 5 black-and-white Chinese characters considered in this research. As we can see these 5 characters are very similar to each other. Each of these patterns is scanned (20×20 pixel images) and represented in the vector format. Similar to Section 5.3.2.1, each pattern is represented by a vector $p_i = (x_1 \quad x_2 \quad ... \quad x_n), n = 400$, where $x_i = 1$ if it is a black pixel and $x_i = 0$ if it is a white pixel. In testing, each character is randomly blocked 50%, which means 200 randomly selected pixels are set to the undefined value (0.5). These testing patterns are sent to the associative memory for recognition and recovery. Figure 5-14 shows the input testing patters, corresponding to the training patterns in Figure 5-13.



Pattern 1 Pattern 2 Pattern 3 Pattern 4 Pattern 5

Figure 5-13 Training patterns: Five black-white Chinese characters

Pattern 1 Pattern 2 Pattern 3 Pattern 4 Pattern 5

Figure 5-14 Testing patterns with 50% pixels (200 pixels) blocked

The performance of the associative memory is evaluated in two ways. The first one is the correct recognition rate, which is similar to what is defined in Section 5.3.1. Ten-runs simulation is performed to get the statistical performance, and SOAM achieved an average of 100% correct recognition rate for these very similar Chinese characters. The second evaluation step is to observe what the recovered character looks like, and what the ratios of the error bits and missing bits in the recovered pattern are, similar to Section 5.3.2.1.

Figure 5-15 shows the recovered patterns corresponding to Figure 5-14, and Table 5-11 is the error bits and missing bits information for each pattern. The grey part in Figure 5-15 shows that there are still some missing pixel values after recovery. From Figure 5-15 one can see that the associative memory can correctly recover the original image even if only half of the pattern pixels are presented. Table 5-11 illustrates that the average error bits percentage and missing bits percentage are about 6.2% and 2.4%, respectively.

Pattern 1 Pattern 2 Pattern 3 Pattern 4 Pattern 5

Figure 5-15 Recovered Chinese testing pattern

Table 5-11 Testing error bits and missing bits information

Testing pattern	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Mean
Error bits	12	9	14	13	14	12.4
Error bits percentage	6%	4.5%	7%	6.5%	7%	6.2%
Missing bits	6	5	7	2	4	4.8
Missing bits percentage	3%	2.5%	3.5%	1%	2%	2.4%

5.3.3 Discussion

The results presented in this section demonstrate adequate performance of the proposed self-organizing associative memory on both HA and AA applications. I did not expect these results to be the best among the networks used for comparison, which are optimized for a specific problems (e.g., classification with fixed number of classes). The proposed SOAM network is more robust in terms of the variety of problems it can solve without modification of the network structure. The aim of the proposed associative memory is to search for network structures that can be extended towards building real intelligent

machines, perhaps toward structures resembling the macrocolumns and minicolumns that have been the objects of recent biological studies [Mou 97] [Jon 00]. Flexibility and robustness, rather than optimality, is the trademark of human brain, and I wish to contribute progress that takes neural networks from implementing specialized filters that solve classification problems in an optimum way towards flexible structures capable of accumulating knowledge and using them to reach higher levels of self-organization and problem solving.

5.4 Temporal sequence learning

As presented in Chapter 4, sequence learning and prediction is one of the most important elements of human intelligence. In this section, a four level hierarchical structure with letters, words, sentences and strophes of the popular song “This land is your land” is simulated to illustrate the proposed hierarchical sequence learning machine [Sta 06c].

The original sensory input data received from the environment contains the scan of 20×20 pixel images for the entire alphabet plus three special characters: space, dot, and semicolon for the end of input sequence for word level, sentence level and strophes level, respectively. There is no distinction between learning and playing back, which means that with each incoming sequence, the model will either correctly predict this sequence, or conduct a one-shot learning at the end of the sequence at a particular level of sequence hierarchy. Figure 5-16 shows the simulation results for this model.

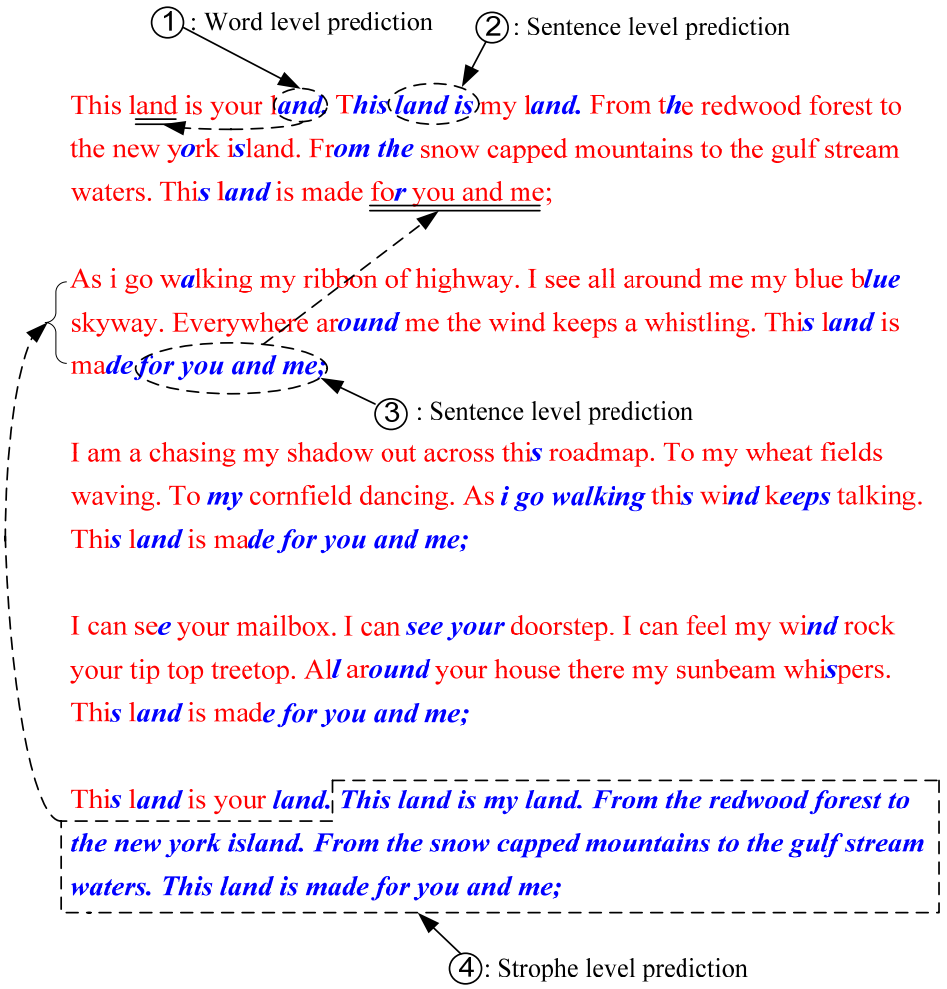


Figure 5-16 Simulation results for the temporal sequence learning model

The light text (in red) is the input sequence and the bold italic text (in blue) stands for the correctly predicted elements of the sequence. ① indicates the correct prediction at the word level. For instance, when the first letter "*l*" of the second "*land*" is activated, the memory correctly predicted the next symbol, "*a*", because it had already learned the sequence "*land*" when it is presented the first time. ② and ③ in Figure 5-16 indicate correct predictions at the sentence level. "*This land is made for you and me*" is repeated at the end of the second strophe, therefore the words "*for you and me*" is correctly predicted when "*This*

land is made" is presented. The reason "*land is made for you and me*" cannot be predicted after "*This*" is that there are other sentences, such as "*This land is your land*", "*This land is my land*" with the same first three words. Accordingly, the MWDN neuron fired after "*This*" is presented, inhibiting firing of the prediction neuron. ④ in Figure 5-16 indicates a correct prediction at the strophe level. The last strophe is the repetition of the first one. Therefore, after the first sentence of the strophe, the model correctly predicted the rest of the strophe.

After the model has stored this song, it should have the ability to predict the song from an input hint, or to learn a new sequence without destroying the stored sequence. For example, if a hint "*e*" is given to the system, there is a unique neuron in the word level that stores the word "*everywhere*". So, the model will play the word "*everywhere*". After this neuron in the word level fires, it will trigger a neuron in the sentence level to fire. In this case, a unique neuron in the sentence level will be the winner, and that neuron will play back its stored sequence from the sentence level, to the word level, and finally generate the original sequence. Figure 5-17 shows the simulation result after the character "*e*" is activated as a hint to the model.

Input hint: e

Prediction output: ***everywhere around me the wind keeps a whistling.***

Figure 5-17 Prediction result based on hint

In addition, if a new word, sentence or strophe is presented to this model, it can further learn the new sequence without destroying the previously stored sequence, and this learned sequence can be used for associative predictions.

5.5 Summary

In this chapter, different applications of the proposed dynamically self-reconfigurable systems for machine intelligence are investigated. Such applications include classification, pattern recognition, image recovery and sequence learning. These applications show that the proposed models are capable of learning, memory and prediction based on active interacting with the external environment. It is confirmed that such system level models and architectures are essential elements for building real brain-like intelligent systems, and will have wide applications in different areas.

Chapter 6

Low Power Design and Hardware Architecture

6.1 Introduction

So far, I have presented the “learning-memory-prediction” framework aimed at designing brain-like intelligent systems, and under this framework, system level models and architectures of dynamically self-reconfigurable systems for machine intelligence are proposed. One of the major advantages of such proposed systems is that they are hardware-oriented, fault tolerant, and modular for easy implementation in hardware.

Power consumption is one of the critical design issues in hardware implementation (either using dedicated VLSI technology or FPGA technology) of such large scale dynamically intelligent systems. Due to the high density of interconnections (although each processing element has sparsely and locally dominated interconnections, there still are many interconnections in the whole system, as in the association network in Figure 5-10), high data flow, and high volume of switching activities, the designed systems will not be able to function properly without careful consideration of low power design issues.

Power dissipation in a complementary metal oxide semiconductor (CMOS) based logic circuit is dominated by the switching energy [Cha 04b] [Rab 03]. This is the energy that must be dissipated in order to change the internal state of the devices. Other sources of energy loss in CMOS circuits are related to leakage current [Rab 03] [Han 04] and to short circuit current [Rab 03] [Umi 95]. While the other two types of energy loss can be reduced to

arbitrarily low levels (at least in theory) by regulating the voltage levels (both power supply and thresholds), there seems to be no solution to eliminate dynamic power dissipation. In this dissertation research, a novel computing paradigm, that in the ideal case, may lead to logic operations without dynamic power dissipation is proposed [Sta 06d]. The solution proposed is a theoretical one, since so far no practical devices have been built to support this kind of circuits. However, lowering the path resistance by several orders of magnitude is certainly within the reach of superconducting devices, thus the proposed paradigm may become a valuable solution for future large scale, high density intelligent systems design.

Finally, I give a brief description of the system level hardware architecture of the proposed reconfigurable systems. The current design takes advantage of reconfigurable computing FPGA technology, and is under prototyping and testing [SOL 06].

6.2 Low power design

6.2.1 Low power switched inductor-capacitor (SLC) architecture

The primary components of power dissipation in digital CMOS design can be estimated using the following equations [Rab 03].

Dynamic power:

$$P_{dyn} = \alpha C_L V_{DD}^2 f \tag{6.1}$$

Short circuit power:

$$P_{sc} = t_{sc} V_{DD} I_{peak} f \tag{6.2}$$

Leakage power:

$$P_{leakage} = V_{DD} I_{leakage} \tag{6.3}$$

where α is the switching activity factor, C_L is the load capacitance, V_{DD} is the supply voltage, f is the clock frequency, t_{sc} represents the time for short-path conducting from V_{DD} to ground and $I_{leakage}$ is the leakage current.

Dynamic power is the dominant component among these three sources. Assume a logic gate is driving a load capacitor C_L. Each time there is a logic 1 at the output, C_L is charged through the pull-up circuit from the supply voltage V_{DD}. This energy is stored in the capacitor until the output logic value changes to 0 and the load capacitor C_L is discharged. In this case, the stored energy will be dissipated through the pull down circuit to the ground. This energy is changed to thermal energy and needs to be removed by a cooling process. The amount of energy dissipated through switching the logic output value does not depend on switch resistance. Even if superconducting switches were used it will remain on the same level. Not only does this drain the power, it also creates a requirement for an efficient heat removal and increases the packaging cost.

An alternative approach would store this energy and re-use it whenever it is needed. Based on this idea, I propose a novel SLC logic architecture capable of operating without energy loss in the ideal case. In this architecture, the energy is stored alternatively in an inductor whenever the load capacitor C_L is discharged, and in the capacitor C_L whenever the load capacitor needs to be charged again. In this way, ideally, no energy is wasted and the load capacitor only needs to be charged once. In non-ideal devices, stored energy is reduced through leakage current or dissipated on path resistance (defined as a combination of switch

resistance, serial resistance of inductor, capacitor or wires) and needs to be restored. The proposed architecture is especially suitable in the situation where large load capacitor needs to be charged and discharged. Buses and clock distribution trees are two examples of such loads. For instance, it is reported that the bus dissipated about $15\% \sim 30\%$ of the total power in Alpha 21064 and Intel 80386 [Liu 94]. It is also reported that a clock distribution tree dissipated about $20 \sim 50\%$ of the total power [Far 01] [Pan 02]. For example, in the second generation of the Alpha microprocessor, the Alpha 21164, the clock distribution tree consumes 20W, which is 40% of the total power dissipation of the processor [Bow 95]. A simple SLC circuit may save most of this energy. The major difficulty in implementing the proposed logic design scheme will arise from the fabrication process and implementation technology. A brief discussion of these issues is addressed in section 6.2.2.4.

Figure 6-1 shows a basic logic gate structure for the proposed SLC architecture. It consists of a switch control unit (SCU) with two control signals $CS1$ and $CS2$ and two energy storage elements – inductor (L) and load capacitor (C). Inside the SCU, there are four switches controlled by two-phase control signals $CS1$ and $CS2$, which control switches $s1$ and $s2$ respectively. If the control signal is equal to 1, the corresponding switch is on, otherwise it is off.

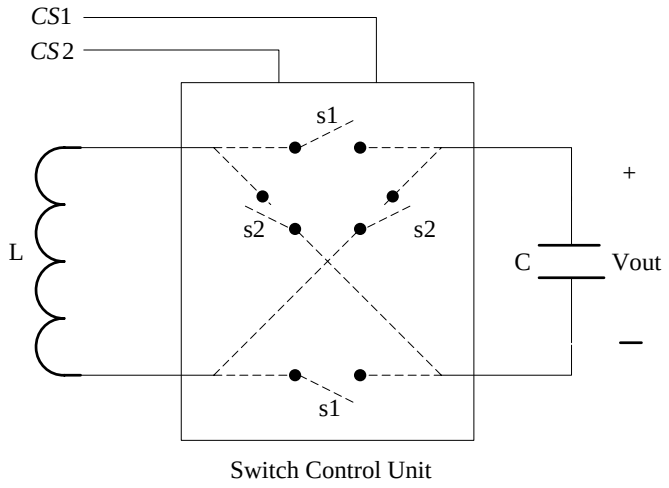


Figure 6-1 Proposed SLC architecture for low power design

The circuit shown in Figure 6-1 inverts the control input value, thus implementing an inverter. Unlike CMOS gates that implement inverting logic, this design style can produce both inverting and noninverting logic in the same circuit. I now give a detailed description of this circuit focusing on inverting logic operation first.

Figure 6-2 shows the waveform of the control signals $CS1$ and $CS2$, where T is the period related to the LC resonant frequency: $T = 2\pi\sqrt{LC}$.

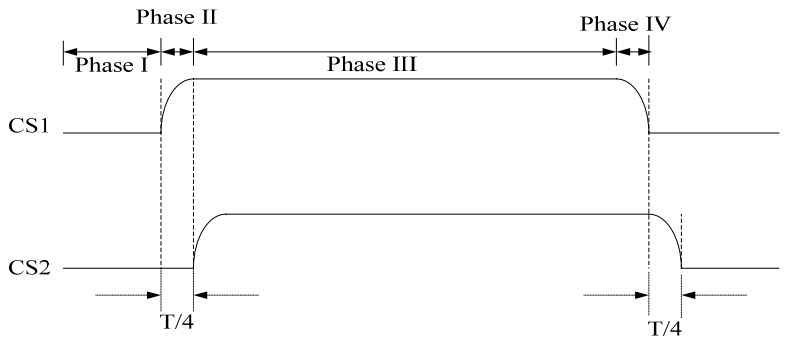


Figure 6-2 Control signals of $CS1$ and $CS2$

Based on the value of control signals $CS1$ and $CS2$, this circuit will have four phases of operation as illustrated in Figure 6-3.

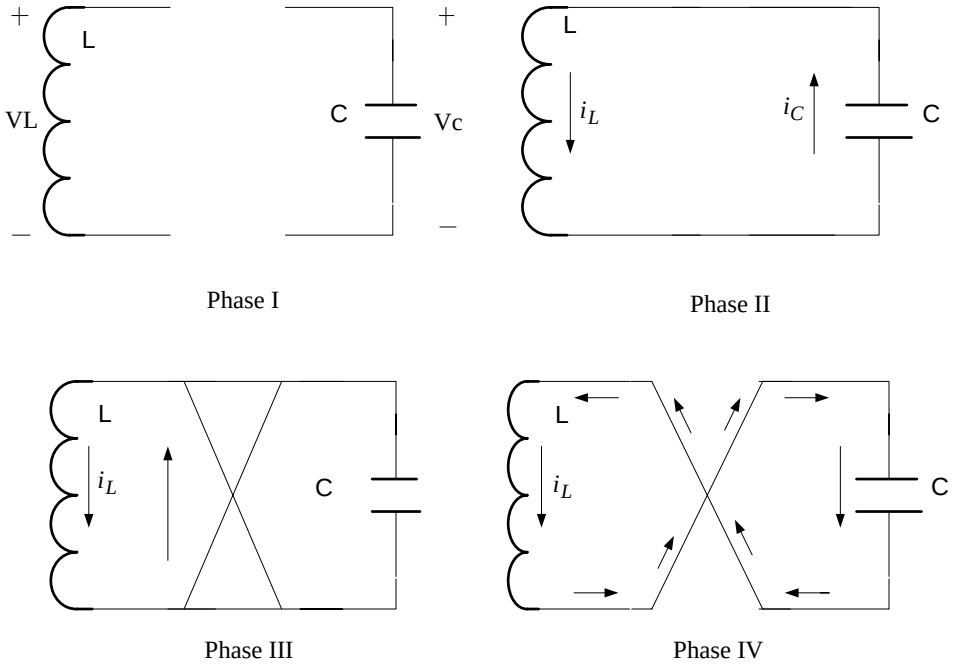


Figure 6-3 Operation phases of the proposed circuit

Phase I: $CS1 = 0, CS2 = 0$

In this phase, switches $s1$ and $s2$ are off. Energy is stored in the load capacitor C. This corresponds to a logic 1 in the output load.

Phase II: $CS1 = 1, CS2 = 0$

In this phase, the $s1$ switches are on and the $s2$ switches are off. The stored energy in the capacitor C is transformed to the inductor L. The capacitor voltage and the inductor current are:

$$v_c(t) = v_0 - \frac{1}{C}\int_0^t i(t)dt$$

$$i_l(t) = \frac{1}{L}\int_0^t v(t)dt \tag{6.4}$$

After $T/4$, the capacitor voltage becomes 0 and the inductor current reaches the maximum point. At this moment, switches $s2$ are closed (on) according to the control signal. This leads to the phase III.

Phase III: $CS1=1, CS2=1$

In this phase, all switches are on. The current is cycled in the inductor and the short circuit paths. This will keep the capacitor voltage at 0, which corresponds to a logic 0 at the output load. The load energy is not lost in this phase as it is stored in the magnetic field of the inductor.

Phase IV: $CS1=0, CS2=1$

Whenever there is a need to re-charge the load capacitor again to switch the load output from 0 to 1, the $s2$ switches are set on and $s1$ switches are off. This will re-charge the load capacitor to the full voltage. After $T/4$, the capacitor voltage reaches the maximum point and energy is restored in the electrical field of the capacitor. The inductor current is 0 at this point. Switches $s2$ then are opened again (off), which leads to the initial phase I.

From the above four operation phases, the energy is stored in the inductor when there is a need to discharging the load capacitor. Whenever next time the capacitor needs to be re-charged, the energy stored in the inductor can be reused to charge the capacitor. Ideally, no energy is lost in this architecture. The reason for the cross connection of the switches in the switch control unit is to avoid a negative charge on the load capacitor.

6.2.2 Simulation results

In this section, SPICE simulation for Figure 6-1 to build inverting and noninverting logics is presented. Meanwhile, the comparison of the energy loss of the proposed circuit with those of the traditional integrated circuit (IC) design is also discussed in this section.

6.2.2.1 Inverting logic construction

Various logic gates can be built using the described energy exchange and storage mechanism. To implement a general compound gate, the same approach as in NMOS design can be used. To accomplish a pull down operation (logic 0), one needs to compose a conducting path of switches controlled by various input signals. A series connection of switches implements logic AND, while a parallel connection implements logic OR. By properly combining switches controlled by the input signals, any logic function F in the pull down path can be implemented. Thus a switch controlled path implements a logic function F, and the resulting SLC gate implements this logic function inverse $\overline{F}$.

For instance, to implement a 2-input NAND gate $F = \overline{A \bullet B}$, each switch $s1$ in Figure 6-1 is replaced by a series connection of two switches controlled by signals A and B respectively. Each switch $s2$ is replaced by a series connection of two switches controlled by signals A and B delayed by $T/4$. The $T/4$ delay can be achieved by passing the input signal through a transmission delay line. Figure 6-4 shows the simulation

result. The logic function is correctly implemented and the four operation phases presented in Section 6.2.1 are clearly marked.

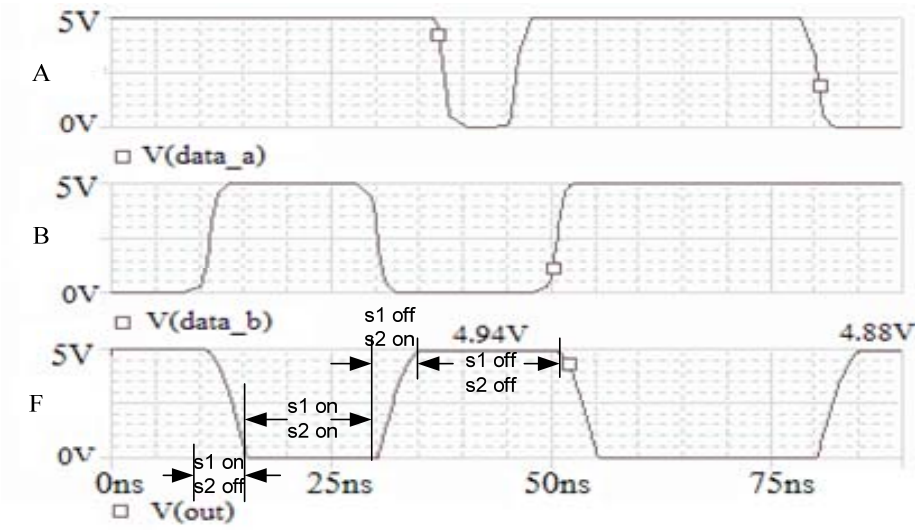


Figure 6-4 $F = \overline{A \bullet B}$ output waveform

Figure 6-4 also shows there is some energy loss when the load capacitor is re-charged from the inductor (the output voltage drops from 5V at the beginning of simulation to 4.94V and 4.88V in the first and second re-charge period, respectively). This voltage drop is due to the path resistance. This means it is necessary to periodically re-charge the output capacitor through the pull-up path when the stored energy is below a set threshold. The energy losses of the SLC scheme are discussed in detail in Section 6.2.2.3.

6.2.2.2 Non-inverting logic

Unlike CMOS circuits that implement inverting logic only, the proposed SLC structure can be used to build non-inverting logic. This can be accomplished by using

complementary switches in the switch control unit. The complementary switch is on when the control signal is 0 and is off, when the control signal is 1.

In this case, the conducting path must stay open to maintain the high output value, so it is necessary to express the complemented function value through a logic combination of complemented input signals. For instance to obtain a 2-input AND operation, one need to build a conducting path of complementary switches connected in parallel (Figure 6-5) that implements $\overline{F} = \overline{A} + \overline{B}$ (equivalent to $F = A \bullet B$). Notice that, if the parallel connection of regular switches is used, the same SLC circuit would implement a NOR gate

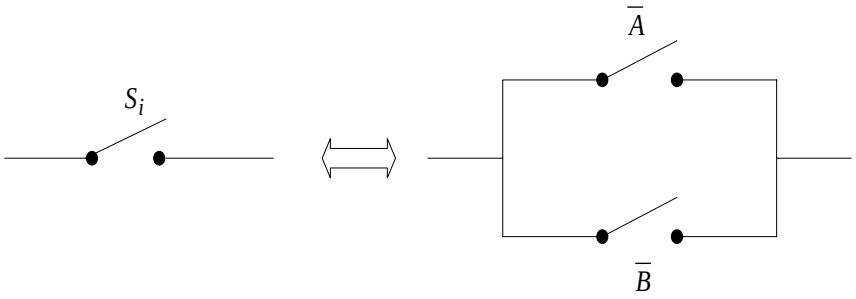


Figure 6-5 Switch S_i replaced by path of complementary switches

6.2.2.3 Energy loss and timing analysis

Figure 6-6 shows an equivalent circuit to charge the load capacitor from 0 to V_{dd} in the traditional IC design [Rab 03].

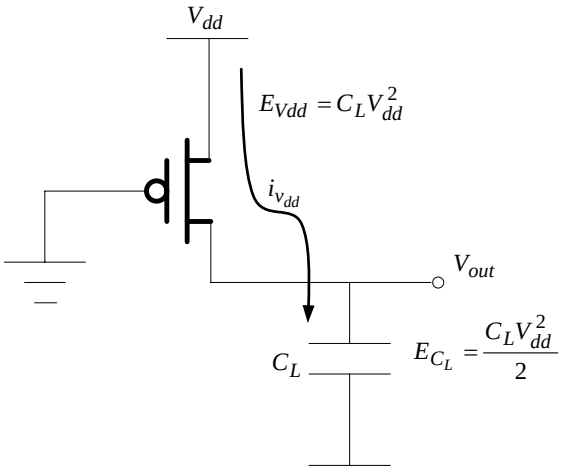


Figure 6-6 Equivalent circuit in the charging load capacitor phase [Rab 03]

It is well known that the energy taken from the supply during the charging transition $E_{V_{dd}}$ and the energy stored in the load capacitor at the end of the transition E_C are [Rab 03]:

$$E_{V_{dd}} = \int_0^\infty i_{V_{dd}}(t) V_{dd}\, dt = V_{dd} \int_0^\infty C_L \frac{d_{v_{out}}}{dt} dt = C_L V_{dd} \int_0^{V_{dd}} dv_{out} = C_L V_{dd}^2 \tag{6.5}$$

$$E_c = \int_0^\infty i_{v_{dd}}(t) v_{out}\, dt = \int_0^\infty C_L \frac{dv_{out}}{dt} v_{out}\, dt = C_L \int_0^{V_{dd}} v_{out}\, dv_{out} = \frac{C_L V_{dd}^2}{2} \tag{6.6}$$

As we can see from Equation (6.5) and (6.6), half of the energy is dissipated in the transition regardless of the size of the PMOS device and its effective resistance. During the discharging phase, the stored energy $C_L V_{dd}^2 / 2$ will be dissipated through the NMOS device to the ground. Thus with each change of the logic value this device dissipates E_C energy.

However, this is different in the proposed SLC circuit. Figure 6-7 shows the relationship between the percentages of stored energy losses per cycle with respect to the path

resistance for different L and C values. The x-axis is the value of the path resistance (in ohms) and y-axis is the percentage of energy losses. Figure 6-7 indicates that the energy loss increases with the increase of the path resistance. This means it is necessary to periodically recharge the output node through the pull-up path when the output voltage is lower than some threshold. Assume this threshold is set to $V_{dd}/2$, which corresponds to the lose of 75% of the energy stored in the output capacitor. This 75% threshold line determines the maximum value of the path resistance. Figure 6-7 also shows that to get the optimal energy saving performance, the fabrication of very low resistance switches (below 10 ohms for this circuits parameters) is required.

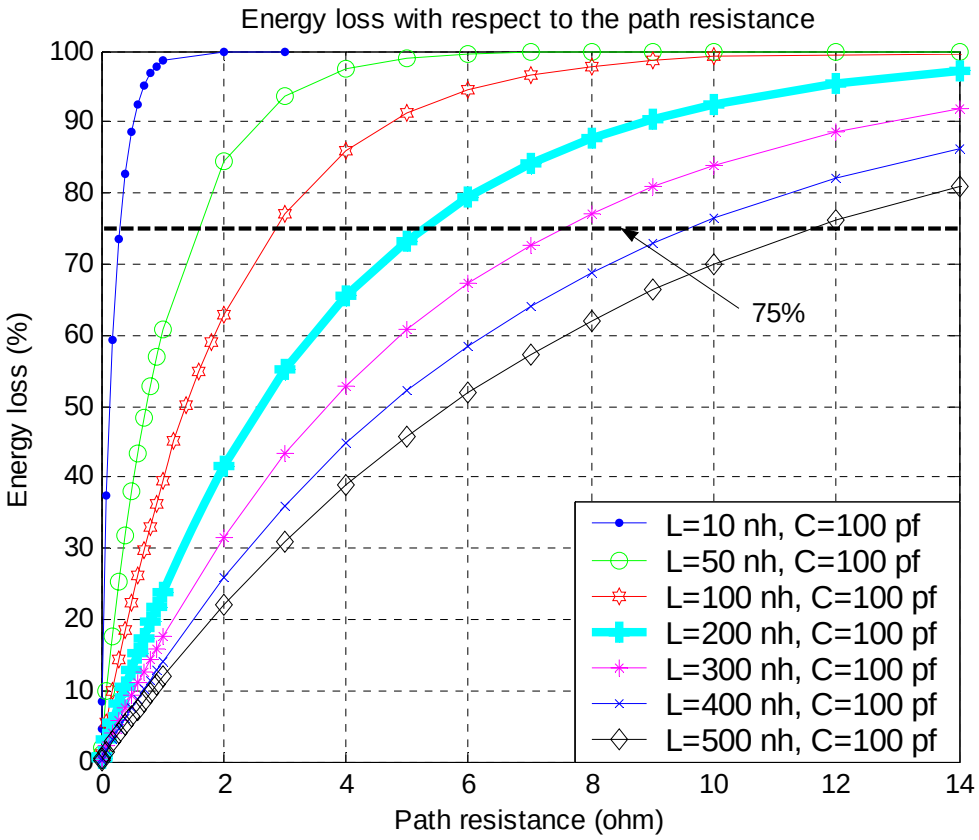


Figure 6-7 Energy loss with respect to the path resistance

Clock jitter will also cost energy loss in the proposed SLC circuit. As illustrated in Figure 6-2, control signal $CS2$ should be delayed exactly by $T/4$ compared to $CS1$. However, this delay time is hard to maintain in a real circuit due to the clock phase jitter. Figure 6-8 shows the jitter effect. For simplification, let us assume that the control signal $CS1$ has fixed phase and signal $CS2$ either leads (positive jitter) or lags (negative jitter) $CS1$ signal delayed by $T/4$. The jitter in this dissertation research is defined as percentage of clock time $(\delta / T * 100)\%$.

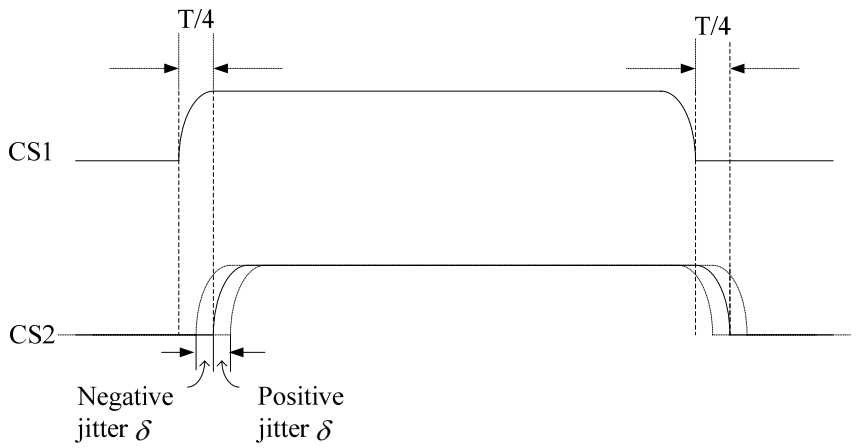


Figure 6-8 Control signal jitter effect

Considering a SLC design with the parameters of $L = 200nh$ and $C = 100pf$ from Figure 6-7, Figure 6-9 illustrates the energy loss for different path resistance due to the jitter effect. As one can see from Figure 6-9, if there is no jitter, the corresponding energy loss is consistent with the results illustrated in Figure 6-7. With the increase of the jitter, more energy is lost in this switching scheme. For instance, if the jitter is larger than 15%, majority of the energy (more than 90%) will be lost due to the jitter effect. This means that for the proposed

SLC circuit to work properly, it is very critical to control the proper timing of the control signals $CS1$ and $CS2$.

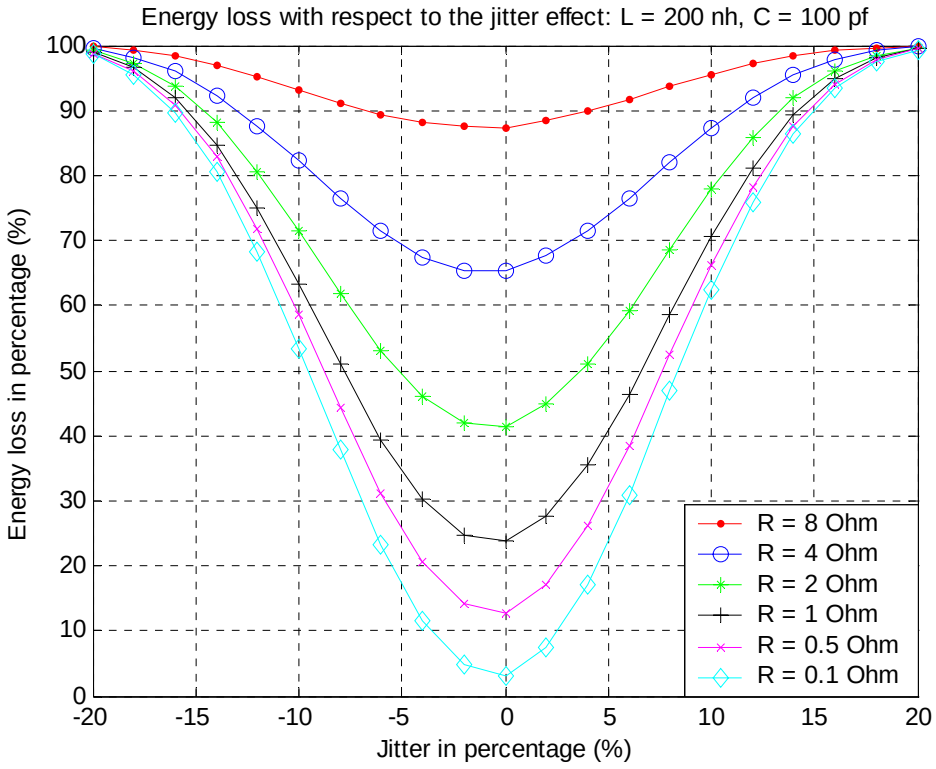


Figure 6-9 Energy loss with respect to jitter effect

Another issue related to the SLC circuit is the operating frequency. As illustrated in Figure 6-7, path resistance affects not only the energy loss but also the frequency of operation of the SLC circuit. The larger the path resistance, the larger the inductor value that corresponds to the same level of energy loss. This in turn means that the larger the resistance, the smaller the operating frequency. Figure 6-10 shows the LC resonance frequency with respect to the path resistance for different energy loss levels.

There are two time periods in the SLC circuit: Period T is decided by the resonance frequency of the LC parameters and sets $T/4$ delay of $CS2$ signal with respect to the $CS1$

signal (Figure 6-2). The other period T_f is related to the operating frequency of the logic function. T_f must be significantly larger than T to guarantee that the logic function is correctly implemented. For instance, consider the 40% energy loss line in Figure 6-10. If the path resistance is about $100m\Omega$, then the resonance frequency is about $1GHz$. If the logic operating frequency is set to be 10 times lower than this resonance frequency, then the maximum logic operating frequency is $100MHz$. With further reduction of the path resistance, the logic operating frequency will increase. Though fabrication of low resistance switches is not practical in the current technology, lowering the path resistance to single milliohms could be within reach of superconducting devices in the future. Therefore, the speed of the SLC circuits should satisfy the requirements of modern digital circuits.

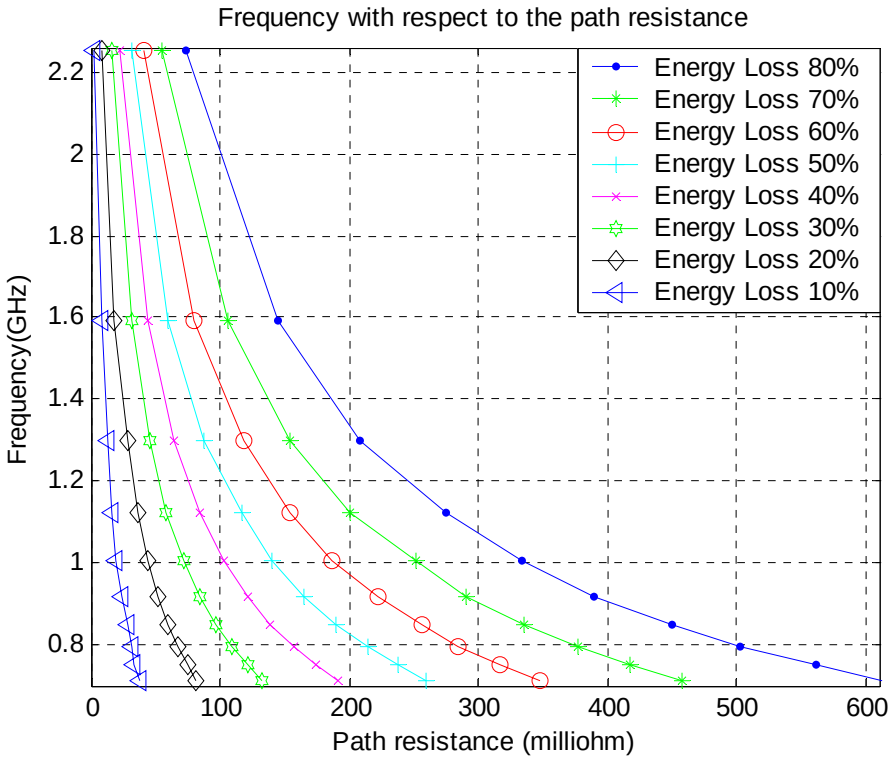


Figure 6-10 Frequency relationship with the path resistance

6.2.2.4 Discussion

A novel idea of low power digital design based on the proposed SLC architecture is presented and simulation results show correct behavior of the proposed logic. However, for this methodology to be used in practical sub-micron IC design, there are several issues that need to be carefully evaluated and require further research.

(1) Area overhead. Generally speaking, the proposed SLC gate needs 2 times the number of transistors plus an inductor than the equivalent CMOS gate. The proposed scheme is especially suitable for circuits that need to drive large capacitive loads, such as buses and clock distribution trees, or in the situations where power consumption is the most critical concern with no strict limitation on design area.

(2) Energy loss. Since the path resistance will consume some energy, it is necessary to periodically recharge the load capacitor through the pull-up path when its voltage is lower than some pre-set threshold. This would require extra circuitry. In addition, to meet the speed requirements of the modern digital circuit, the switch-on resistor should be on the level of milliohms or less. This is out-of reach of current MOS technology.

(3) High quality on-chip inductors. This new circuit needs the fabrication of accurate high quality (low resistance) on-chip inductors. This presents a serious challenge for fabrication technology. For reference, please review the practical design considerations for on-chip inductors presented in [Yue 99]. The impact of interconnect scaling, copper metallization and low-K dielectric on the achievable inductor quality factor is studied there.

(4) Timing of the control signals. The correct operation of this circuit depends on proper timing of the control signals. Control signal $CS2$ is a control signal $CS1$ delayed by a quarter of the LC oscillation period. How to provide this accurate delay time is also a design challenge. One way to alleviate this problem is to use clocked gate operations, in which all switched paths will contain a clock control switch for synchronization of the control signals.

(5) Antenna and radiation effect. In the proposed SLC circuit, there are two types of energy losses: one is the energy loss due to the path resistance, while the other one is due to the antenna effect. The amount of this radiation energy loss is related to the operating frequency (influence the wavelength of the radiation), size of the SLC circuit (influence the antenna length) and the path resistance (influence the current magnitude in SLC circuit) [Sad 01]. Without degrading the operating frequency of the SLC circuit, reduction of the radiation energy loss can be achieved by carefully designing the size of the on-chip inductor and routing the length of wires in SLC (reduce antenna length and avoid cross coupling between different SLC circuits).

Explorations of superconductor properties to digital computing are the subject of intensive research. Perhaps the most promising technologies today explore superconducting quantum computing circuits [Orl 02] and the superconducting properties of nanowires [Rog 03]. None of these techniques, however, considers energy transfer from magnetic to electric fields, which is a basis of the computing model developed in this research.

6.3 Programmable hardware for intelligent systems

So far, I have presented system level architectures and models of the dynamically self-reconfigurable systems as well as a concept of low power organization of such system. In this section, I will briefly describe the hardware architecture for designing such systems using reconfigurable FPGA technology.

Intelligent system models can be simulated in a sequential computer, mapped into a programmable hardware, or built in VLSI. Software implementation is the easiest one but it has its inherent limitations. With the existing state of personal computer technology, it is not practical to develop a network with more than 10000 neurons. This limitation comes from the limited computer speed and their dynamic memory. While such networks can be used to test some machine learning ideas, they are not sufficient to build human-like intelligence. Assuming continuous progress in the computer industry and extrapolating computer efficiency growth into near future, we can expect computers to be 10 times faster and have 20 times larger random access memory (RAM) in the next 7 years [Int 05] [Mor]. This would allow the increase in size of simulated networks to 100000 neurons (with twice as many connections per neuron as in today models). Yet, as the industry is expected to produce 60 times as many transistors as it produces today [Mor], the system level capacity is expected to grow 60 times in the same time period. Thus, the expected capacity of programmable hardware will grow significantly faster than the software speed. In addition, this hardware will operate at increased speed (the same growth rate as the growth rate of computer speed), thus in the next 7 years, the computing capacity of parallel hardware will grow 600 times

comparing to 10 times growth in the software speed. This discrepancy between software capacity to simulate networks of neurons for machine intelligence and hardware capacity to implement them will grow even more in the future. Thus, we need to develop methods to build concurrent hardware prototypes for networks of neurons, program them, and test their performance in arrays of processors. This is a motivation behind the SOLAR program [SOL 06]. This program intends to build support for system level implementation of networks of neurons in a parallel array of processing elements implemented in FPGA technology.

Figure 6-11 illustrates the system level hardware architecture of the proposed dynamically reconfigurable system [Sta 03a]. Every processing element (neural unit) will receive input data and feed output data after processing to the routing channel through the bidirectional routing unit (BRU). The configurable switching unit (CSU) permutes the incoming data to any location of the output address, which provides a mechanism for dynamically defined connections between different neurons. In addition, such a connection structure can be dynamically reconfigured based on the embedded configuration units in the routing path. We fabricated the hardware FPGA boards with each board hosting 4 Xilinx XCV 1000 chips, and these boards are under prototyping and testing. A more detailed description of the design progress in this research can be found in [SOL 06].

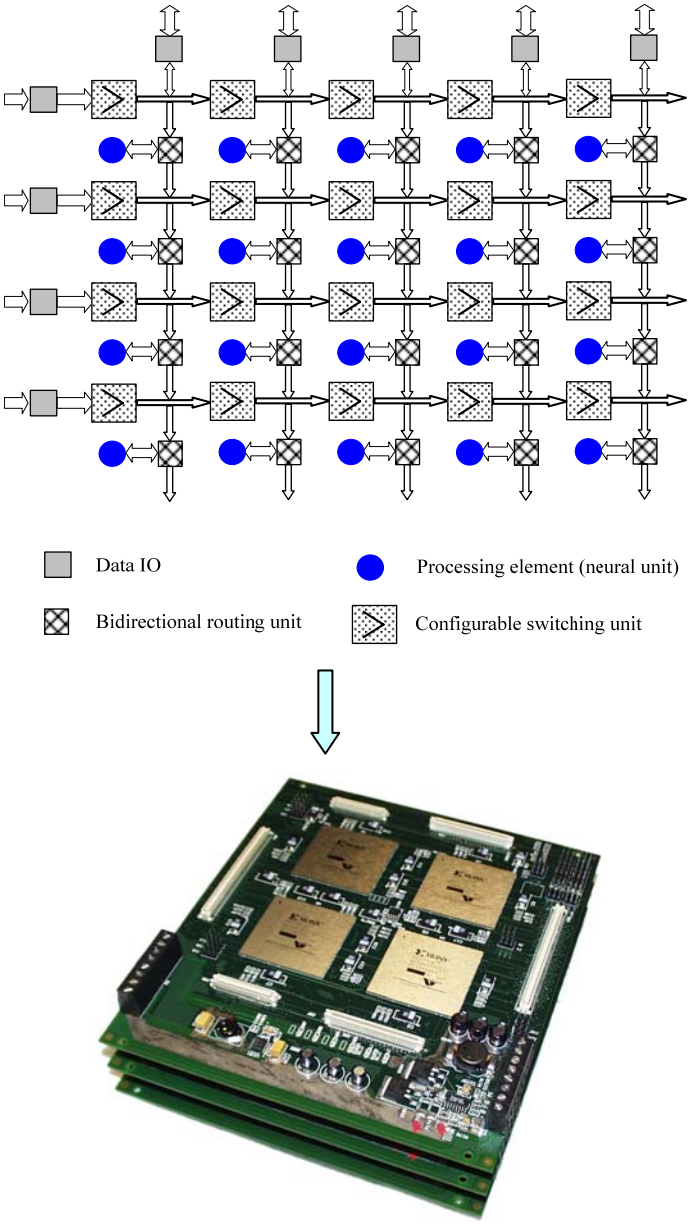


Figure 6-11 System level hardware architecture [Sta 03a] & [Sol 06]

6.4 Looking ahead

Power dissipation is the most critical issue for the hardware implementation of intelligence. Today, a simple processor running at $3\,GHz$ speed dissipates over 100 Watts of power. Making a significant array of such processors (several millions of them) will be too

costly to operate. For comparison, a human brain with 10^{11} neurons dissipates less than 10Watts of power. If we consider a neuron to be the elementary processor of intelligent systems, this translates to 10^{-10} Watts per neuron. Comparing this to 100Watts dissipated by today's computers, each neuron dissipates on average 10^{12} times less energy. Thus power dissipation is a key issue.

If we look at the dynamic power dissipated by N digital processors working concurrently, we can estimate it to be in proportion to $NCV^2 f$, where C is the total actively switched capacitance of a unit processor, V is the power supply voltage and f is the frequency of operation. Let us assume that the effective switching capacitance of a unit processor is $10nF$ (this number corresponds to currently built processors). This capacitance can go down linearly as the feature size will be reduced in future. Since the feature sizes are reduced at the rate of 2.5 times every 7 years (about 30% reduction every three years [Int 05] [Mal 96]), there is not much reduction expected due to this factor. The voltage level is around 1V and will not be significantly reduced in the next 7 years. Thus, if we wish to design a system with 10^{11} processors (not withstanding its cost), its total dynamic power would be:

$$P = 10^{11} \times \left(10 \times 10^{-9}\right) \times 1^2 \times f \tag{6.7}$$

Such a system would have to operate at a frequency of $f = 10^{-2} Hz$ to dissipate the same power as the human brain. Since neuron response time is estimated at 5 ms, such a system would be 20000 times slower than the human brain while operating at the same power budget. Even if we assume that only 5% of neurons are active at any given time, this would still require the system to operate 1000 times slower than the brain. Thus the only viable implementation for a brain-level system in modern VLSI technology is in analog VLSI

implementation. Analog VLSI circuits for neural network implementation were pioneered by E. A. Vittoz [Vit 85] [Vit 90a] [Vit 90b] [Vit 96] [Vit 98] and C. Mead [Mea 89], and many useful circuits were built since their pioneering work. Analog processors are 2 orders of magnitude smaller [Bay 96] [Bay 97] (and therefore require to load 100 times smaller capacitance), they can also operate at lower voltages. Lowering voltage level to $40mV$ in information transformed between neurons (as it is in real neural networks) would lower power requirements over 600 times. This combined with a smaller total capacitance to load would reduce power consumption to levels comparable or lower than those used by living brain making hardware implementation of human level intelligence feasible from an energy point of view.

Handling the power dissipation problem does not guarantee that a system with human level capacity will be built in near future. Existing FPGA chips can integrate about 400 picoblaze controllers, this is by no means is close to what one would need to design a system with 10^{11} processors. Even if an array of FPGA chips is designed with 10000 chips, it would allow to emulate 4×10^{6} processors. Hardware progress in the next 7 years could increase this number to 10^{8} processors in a 10000 chips system (assuming current growth of the number of transistors per chip doubles every 18 month). Even with this capacity, fully parallel implementation will not be comparable to the capacity of the human brain. Thus, a hybrid solution that combines parallel array implementation with software simulation needs to be considered. For instance, if each processor would simulate a cluster of 1000 neurons we would be able to implement a system of human level complexity within next 7 years.

However, 10000 chips on 2500 boards would carry a high price tag. Assuming a chip

cost of 1000 dollars 7 years from now, such a system would cost more than 10 million dollars to build and use around 1 MW of power to operate. However, with the current rate of transistor price drop of 1.5 times per year [Mor], such a system would only cost 3000 dollars after 20 years. Power dissipation of such system would still be on the level of 20 KW, so unless the energy price will drop it will be too expensive to operate. Only analog VLSI implementation will be both affordable to buy and operate. Thus in 25 years, people could afford to have their own personal intelligent system. This of course will only be true if we learn how to build such intelligent systems.

6.5 Summary

This chapter presents a low power design scheme for large scale, high density intelligent systems. A novel SLC architecture for reducing the dynamic power consumption is proposed in this chapter. The basic idea of this low power design scheme is to store the energy in an inductor instead of dissipating it to the ground, and re-use it to charge the load capacitor whenever it is needed. This scheme provides a possible solution for future extremely low power design for large scale, high density integrated intelligent systems. The current platform selected for the prototyping and testing of the proposed intelligent systems in this research is the reconfigurable FPGA technology. A brief description of the system level hardware architecture, and a perspective of the future hardware development of human level intelligent systems is also presented in this chapter.

Chapter 7

Conclusions and Future Research

7.1 Conclusions

Development of deep submicron VLSI technology and neuroscience research facilitate building complex dynamically self-reconfigurable systems for machine intelligence. Over the past decades, the traditional AI approach has shown little progress in designing brain-like intelligent systems. Therefore, it is critical to develop a new framework for the designing of such systems. This is the major motivation of the research presented in this dissertation.

In this dissertation, a novel framework titled “learning-memory-prediction” is proposed. This framework is an extension of the “memory-prediction” theory as presented recently by J. Hawkins in [Haw 04]. As the human brain uses a fundamentally different way compared to today’s computers in handling information, the proposed “learning-memory-prediction” framework provides a solution for designing intelligent systems. Under this framework, three major dynamically self-reconfigurable systems that are able to learn, process information, and make associations and predictions for intelligent systems are proposed.

Learning is the fundamental property of biological intelligent systems. Therefore, learning mechanism is studied in Chapter 2. While both supervised learning and unsupervised learning are important for machine intelligence, I focus on the unsupervised reinforcement

learning. The reason for this is that in a practical learning environment, the supervisor (instructor) is not always available. In most of the cases, an intelligent system should learn through active interaction with the external uncertain environment, accumulate knowledge through experiences, define goals for its actions, and adjust its actions to achieve these goals. Many of these objectives can be accomplished through reinforcement learning. Since a value system is considered as one of the most critical components for reinforcement learning, Chapter 2 proposes a novel online value system, which can dynamically evaluate the signal value for any multi-dimensional data sets. This value system can be an important element for reinforcement learning to help the machine to adjust its actions.

Chapter 3 focuses on the development of a self-organizing associative memory network. As an intelligent system learns the information from its environment, it should also be able to memorize such information in its distributed elements, and be able to make associations to recover a complete picture based on fractional information. The proposed memory network is a hierarchical network with sparse and local inter-connections. In addition, this memory is capable of self-adjusting its association depth according to the complexity of the incoming information. This is fundamentally different than the fixed structure of a pre-set memory network.

After an intelligent system is able to learn and memorize information, it should be able to anticipate certain responses from the environment that is a result of its actions and thus predict the incoming information. It is well known that the ability of prediction is the most important element for human-intelligence. Therefore, a prediction mechanism is introduced in Chapter 4 to allow the machine to be able to learn, remember and anticipate any complex

sequences. This sequence learning machine is a biologically plausible system, in which every neuron only has two states: fire or do not. I believe that this sequence learning mechanism will be a critical element for the development of integrated intelligent systems.

Chapters 2, 3, and 4 provide a complete picture of the proposed “learning-memory-prediction” framework. In order to show that the proposed system level models and architectures have the intelligence to handle different tasks, several application examples are studied in Chapter 5, which include pattern recognition and classification, image recovery and temporal sequence learning.

As the long-term objective of the intelligent systems design is to implement them in hardware, a novel SLC architecture that is able to significantly reduce the dynamic power consumption of traditional CMOS designs is proposed in Chapter 6. As a novel computing paradigm, I hope that this scheme can provide a possible solution for low power design for future large scale, high-density intelligent systems in silicon. In addition, a brief description of the system level hardware architecture for prototyping and testing the proposed models is illustrated.

7.2 Original Contributions

One of the major contributions of this dissertation research is the proposed “learning-memory-prediction” framework for designing real-time, large, distributed intelligent systems. Under this framework, different system level models and architectures of dynamically self-reconfigurable systems for machine intelligence, including dynamic value

system, self-organizing associative memory, and temporal sequence learning are proposed in this research. Such systems are characterized by on-line data driven learning, dynamic reconfiguration, distributed information processing, hierarchical structure, local and sparse interconnections, and self-organization. By integrating such systems together, this dissertation provides the overall perspective for designing a brain-like intelligent system that is able to learn, process information, make predictions and associations, and dynamically adjust its organization and actions to accomplish the desired tasks and achieve its goals.

Another contribution of this dissertation is that the proposed systems are hardware oriented, fault tolerant, and scalable. Therefore, this research contributes to the effort of designing large scale intelligent electronic systems. If Moore's Law continues to hold through the quick development of modern electronic technology, it will be possible to design large scale, high density, and low power intelligent systems. As J. Hawkins predicted in his book "On Intelligence" [Haw 04], building brain-like intelligent systems will be the "last great terrestrial frontier of science", and will trigger the renovation of the whole next generation of computer technology. This dissertation research contributes to the research in such an effort.

7.3 Future research

As studying intelligence and discovering mechanisms for intelligent behavior are one of the most exciting research areas in science and engineering, there is still a long way to go before we can design the truly intelligent systems. Along the research direction presented in this dissertation, the following issues are some of the challenges for future research:

(1) Selective observation and representation of the external environment for intelligent systems. In other words, how does the intelligent machine sense the external environment to its own advantage? For instance, when we look around, how can we focus our attention to the interesting information and distinguish it from the environment background noise? How can we successfully build the invariant object representations by integrating continuous observation, understanding and saccade movements? The research in [Li 06] provided an extensive study of active vision through invariant representations and saccade movements. I believe that the results presented in [Li 06] and in this dissertation research can be integrated together to build a complete system for machine intelligence from external environment representation, information sampling to the internal "brain" processing for learning, memory and prediction.

(2) The proposed value system should be integrated into a real reinforcement learning system, and develop mechanisms under which such a system will modify a network's organization and influence its learning, adjust its actions, maximize the reward signals, and achieve its goals.

(3) So far, most of the proposed dynamically self-reconfigurable systems are tested in software. For real time testing in hardware design, either using reconfigurable FPGA technology or dedicated VLSI technology, there are many issues that need to be addressed. For instance, how to make full use of the hardware resources, how to dynamically reconfigure the connectivity structure, and how to process information and provide communication between different processing elements in real-time.

Design of brain-like intelligent systems is one of the most ambitious research topics nowadays. With the development of VLSI systems, nanotechnology, neuroscience and bioinformatics research, I believe that “brain-on-silicon” will not just be a dream and scientific fiction in the future. I also believe that such technology will benefit science and economy alike, and will trigger the renovation of our life in future. However, there are many difficult issues that need to be carefully investigated and studied. I hope that the research presented in this dissertation will contribute to this exciting, interesting and rewarding field.

Bibliography

[Bab 03] S. Babyey, O. Momtahan, and M. R. Meybodi, “Multi mobile robot navigation using distributed value function reinforcement learning,” *Proceedings of IEEE International Conference on Robotics and Automation (ICRA’03)*, pp. 14-19, 2003.

[Bay 96] I. Bayraktaroglu, S. Balkir, and G. Dundar, “ANNSiS: A circuit level simulator for analog neural networks,” *Turkish Symposium on Artificial Intelligence and Neural Networks*, pp. 305-310, June 1996.

[Bay 97] I. Bayraktaroglu, S. Ogrenci, G. Dundar, S. Balkir, and E. Alpaydin, “ANNSyS: An analog neural network synthesis system,” *International Conference on Neural Networks*, pp.910-915, June 1997.

[Bow 94] G. H. Bower, S. Thompson-Schill, and E. Tulving, “Reducing retroactive interference: an interference analysis,” *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 20, pp. 51-66, 1994.

[Bow 95] W. J. Bowhill, R. L. Allmon, S. L. Bell, E. M. Cooper, D. R. Donchin, J. H. Edmondson, T. C. Fischer, P. E. Gronowski, A. K. Jain, P. L. Kroesen, B. J. Loughlin, R. P. Preston, P. I. Rubinfeld, M. J. Smith, S. C. Thierauf, and G. M. Wolrich, “A 300 MHz 64-b quad-Issue CMOS RISC microprocessor,” *International Solid State Circuits Conference (ISSCC’95)*, San Francisco, pp. 182-183, February 1995.

[Bro 95] G. D. A. Brown, P. Dalloz, and C. Hulme, “Mathematical and connectionist models of human memory: a comparison,” *Memory*, vol. 3, no. 2, pp. 113-145, June 1995.

[Bur 98] C. S. Burrus, R. A. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*, Prentice Hall, 1998.

[Cha 03] J. Y. Chang and C. W. Cho, “Second-order asymmetric BAM design with a maximal basin of attraction,” *IEEE Transactions on System, Man, and Cybernetics, part A: Systems and Humans*, vol. 33, pp. 421-428, July 2003.

[Cha 04a] A. Chatterjee and A. Rakshit, “Influential rule search scheme (IRSS) – a new fuzzy pattern classifier, ” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 881-893, August 2004.

[Cha 04b] N. Chabini and W. Wolf, “Reducing dynamic power consumption in synchronous sequential digital designs using retiming and supply voltage scaling, ” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 12, pp. 573-589, June 2004.

[Cor 95] C. Cortes and V. Vapnik, “Support-vector network,” *Machine Learning*, vol. 20, pp. 273–297, September 1995.

[Das 80] B. V. Dasarathy, “Nosing around the neighborhood: a new system structure and classi_cation rule for recognition in partially exposed environments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 1, pp. 67-71, 1980.

[Dau 92] I. Daubechies, *Ten lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. 1992.

[Dju 02] P. M. Djuric, Y. Huang, and E. Ghirmai, "Perfect sampling: a review and applications to signal processing," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 345 – 356, February 2002.

[Doy 90] K. Doya and S. Yoshizawa, "Memorizing hierarchical temporal patterns in analog neuron networks," *Proceedings of Intenational Joint Conference on Neural Networks (IJCNN'90)*, vol. 3, pp. 299-304, San Diego, CA, 1990.

[Elm 00] A. Elmitwally, S. Abdelkader, and M. Elkateb, "Universal power quality manager with a new control scheme," *IEE Proceedings – Generation, Transmission and Distribution*, vol. 147, pp. 183 – 189, May 2000.

[Elm 01] A. Elmitwally, S. Farghal, M. Kandil, S. Abdelkader, and M. Elkateb, "Proposed wavelet-neurofuzzy combined system for power quality violations detection and diagnosis," *IEE Proceedings – Generation, Transmission and Distribution*, vol. 148, pp. 15-20, Janurary, 2001.

[Erw 95] E. Erwin, K. Obermayer, and K. Schulten, "Models of orientation and ocular dominance columns in the visual-cortex: a critical comparison," *Neural Computation*, vol. 7, pp. 425-468, 1995.

[Far 01] A. H. Farrahi, C. Chen, A. Srivastava, G. Téllez, and M. Sarrafzadeh, "Activity-driven clock design," *IEEE Transactions on Computer-Aided Design*, vol. 20, pp. 705-714, June 2001.

[Far 02] S. A. Farghal, M. S. Kandil, and A. Elmitwally, “Quantifying electric power quality via fuzzy modelling and analytic hierarchy processing,” *IEE Proceedings – Generation, Transmission and Distribution*, vol. 149, pp. 44-49, Janurary 2002.

[Fis 36] R. A. Fisher, “The use of multiple measurements in taxonomic problem,” *Annals of Eugenics*, vol. 7, pp. 179-188, 1936.

[Fre 91] R. M. French, “Using semidistributed representations to overcome catastrophic forgetting in connectionist networks,” *Proceedings of the Thirteenth Annual Cognitive Science Society*, pp. 173-178, 1991.

[Fu 98] H. C. Fu and Y. Y. Xu, “Multilinguistic handwritten character recognition by Bayesian decision-based neural networks,” *IEEE Transactions on Signal Processing*, vol. 46, no. 10, pp. 2781-2789, October 1998,

[Gal 04] T. K. A. Galil, M. Kamel, A. M. Youssef, E. F. E. Saadany, and M. M. A. Salama, “Power Quality Disturbance Classification Using the Inductive Inference Approach,” *IEEE Transactions on Power Delivery*, vol. 19, pp. 1812-1818, October 2004.

[Geo] D. George and J. Hawkins, “Invariant pattern recognition using Bayesian inference on hierarchical sequences,” [Online], Available: http://www.stanford.edu/~dil/RNI/DilJeffTechReport.pdf

[Gro 87] S. Grossberg, “Competitive learning: from interactive activation to adaptive resonance,” *Cognitive Science*, vol. 11, pp. 23-63, 1987.

[Han 04] N. Hanchate and N. Ranganathan, “LECTOR: A technique for leakage reduction in CMOS circuits,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol 12, pp. 196-205, February 2004.

[Hau 83] A. F. Haussler, and C. V. D. Malsburg, C. “Development of retinotopic projections: An analytical treatment,” *Journal of Theoretical Neurobiology*, vol. 2, pp. 47–73, 1983.

[Haw 04] J. Hawkins and S. Blakeslee, *On Intelligence*, Henry Holt and Company, New York, NY, 2004.

[He 06] H. He and J. A. Starzyk, “A Self Organizing Learning Array System for Power Quality Classification based on Wavelet Transform,” *IEEE Transactions on Power Delivery*, vol. 21, pp. 286-295, January 2006.

[Hon 00] T. P. Hong and J. B. Chen, “Processing individual fuzzy attributes for fuzzy rule induction,” *Fuzzy sets and systems*, vol. 112, pp. 127-140, 2000

[Hop 82] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the National Academy of Sciences of the USA*, vol. 79, pp. 2554-2558, April 1982.

[Hua 02a] J. Huang, M. Negnevitsky, and D. T. Nguyen, “A neural-fuzzy classifier for recognition of power quality disturbances,” *IEEE Transactions on Power Delivery*, vol. 17, pp. 609-616, April 2002.

[Hua 02b] X. Huang and J. Weng, “Novelty and reinforcement learning in the value system of developmental robots,” *Proceedings of Second International Workshop on*

Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, pp. 47-55, 2002.

[Ibr 02] W. R. A. Ibrahim and M. M. Morcos, "Artificial intelligence and advanced mathematical tools for power quality applications: A survey," *IEEE Transactions on Power Delivery*, vol. 17, pp. 668-673, April 2002.

[Int 05] International roadmap committee, *International Technology Roadmap for Semiconductors, Executive summary,* 2005 Edition.

[Jen 90] P. J. Jennings and S. W. Keele, "A computational model of attentional requirements in sequence learning," *Proceedings of the Twelth Annual Cognitive Science Society,* Hillsdale, NJ, pp. 876-883, 1990.

[Jon 00] E. G. Jones, "Microcolumns in the cerebral cortex," *Proceedings of the National Academy of Sciences of the USA*, vol. 97, issue 10, pp. 5019 – 5021, May 2000.

[Koh 95] T. Kohonen, *Self-organizing maps*. NewYork: Springer-Verlag, 1995.

[Kor 90] C. A. Kortge, "Episodic memory in connectionist networks," *Proceedings of the Twelth Annual Cognitive Science Society,* pp. 764-771, 1990.

[Kun 98] C. K. Kung, M. J. Devaney, C. M. Huang, and C. M. Kung, "Fuzzy-based adaptive digital power metering using a genetic algorithm," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, pp. 183-188, January 1998.

[Lee 01] H. M. Lee, C. M. Chen, J. M. Chen, and Y. L. Jou, "An efficient fuzzy classifier with feature selection based on fuzzy entropy," *IEEE Transactions on Systems,*

Man and Cybernetics, part B: Cybernetics, vol. 31, no. 3, pp. 426 – 432, June 2001.

[Li 06] Y. Li, *Active vision through invariant representations and saccade movements*, [Master thesis], School of Electrical Engineering and Computer Science, Ohio University, 2006.

[Liu 94] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 663-670, June 1994.

[Luc 04] J. Lücke and C. V. Malsburg, "Rapid processing and unsupervised learning in a model of the cortical macrocolumn," *Neural Computation*, vol. 16, no 3, pp. 501 – 33, 2004.

[Mal 89] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-693, July 1989.

[Mal 96] W. Maly, "The future of IC design, testing, and manufacturing," *IEEE Design and Test of Computers*, vol. 13, no. 4, pp. XIX-XXII, 1996.

[Mal 03] C. V. Malsburg, "Self-organization and the brain." *The handbook of brain theory and neural networks*, MIT Press, 2003.

[Man 97] C. G. N. Manning and I. H. Witten, "Identifying hierarchical structure in sequences: a linear-time algorithm," *Journal of Artificial Intelligence Research*, vol. 7, pp. 67-82, 1997.

[Man 99] G. Manganaro, P. Arena, and L. Fortuna, *Cellular Neural Networks: Chaos, Complexity and VLSI Processing*, Springer, 1999.

[Mcc 89] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: the sequential learning problem,” *Physical Learning and Motivation*, vol. 24, pp. 109-165, 1989

[Mea 89] C. Mead, *Analog VLSI and neural systems*, Addison Wesley Publishing Company, 1989.

[Mil 65] I. Miller and J. E. Fruend, *Probability and statistics for engineers*, Englewood, NJ: Prentice-Hall, 1965.

[Mor] G. E. Moore, “Our Revolution,” [Online], Available: http://www.sia-online.org/downloads/Moore.pdf

[Mou 97] V. B. Mountcastle, “The columnar organization of the neocortex,” *Brain*, vol. 120, pp. 701 – 722, 1997.

[Mur 97] B. B. Murdock, “Context and mediators in a theory of distributed associative memory (TODAM2),” *Psychological Review*, vol. 104, pp. 839-862, 1997.

[Ohi] Ohio State University SVM Classifier Matlab Toolbox, [Online], Available: http://www.ece.osu.edu/~maj/osu_svm/ or http://sourceforge.net/projects/svm

[Orl 02] T. P. Orlando, S. Lloyd, L. S. Levitov, K. K. Berggren, M. J. Feldman, M. F. Bocko, J. E. Mooij, C. J. P. Harmans, and C. H. Wal, “Flux-based superconducting qubits for quantum computation,” *Physica C: Superconductivity and its applications*, vol. 372, pp. 194-200, August 2002.

[Pan 02] J. Pangjun and S. S. Sapatnekar, “Low-Power Clock Distribution Using Multiple Voltages and Reduced Swings,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, pp. 309-318, June 2002.

[Pri 01] Primen, "The Cost of Power disturbance to Industrial & Digital Economy Companies," *Consortium for Electrical Infrastructure to Support a Digital Society,* An Initiative by EPRI and the Electrical Innovation Institute, June 2001.

[Qui 87] J. R. Quinlan, P. J. Compton, K. A. Horn, and L. Lazarus, "Inductive knowledge acquisition: A case study," *Processings of the Second Australian Conference on Applications of Expert Systems*, pp. 137-156, 1987.

[Rab 03] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital integrated circuits- a design perspective*, 2nd edition, Prentice Hall, 2003

[Riz 01] D. S. Rizzuto and M. J. Kahana, "An autoassociative neural network model of paired-associate learning," *Neural Computation*, vol. 13, pp. 2075-2092, 2001.

[Rog 03] A. Rogachev and A. Bezryadin, "Superconducting properties of polycrystalline Nb nanowires templated by carbon nanotubes," *Applied Physics Letters*, vol. 83, no. 3, pp. 512-514, 2003.

[Sad 01] M. N. O. Sadiku, *Elements of electromagnetics*, 3rd edition, NewYork, Oxford University Press, 2001.

[Sal 00] I. Salih, S. H. Smith, and D. Liu, "Synthesis approach for bidirectional associative memories based on the perceptron training algorithm," *Neurocomputing*, vol. 35, pp. 137-148, 2000.

[San 00a] S. Santoso, J. Lamoree, W. M. Grady, E. J. Powers, and S. C. Bhatt, "A scalable PQ event identification system," *IEEE Transactions on Power Delivery*, vol. 15, pp. 738-743, April 2000.

[San 00b] S. Santoso, E. J. Powers, W. M. Grady, and A. C. Parsons, “Power quality disturbance waveform recognition using wavelet-based neural classifier-Part I: Theoretical foundation, ” *IEEE Transactions on Power Delivery*, vol. 15, pp. 222-228, January 2000.

[San 00c] S. Santoso, E. J. Powers, W. M. Grady, and A. C. Parsons, “Power quality disturbance waveform recognition using wavelet-based neural classifier-Part II: Application,” *IEEE Transactions on Power Delivery*, vol. 15, pp. 229-235, January 2000.

[Sch 96] A. Schaik, E. Fragnière, and E. A. Vittoz, “A silicon model of amplitude modulation detection in the auditory brainstem,” *Neural Information Processing Systems*, pp. 741-750, 1996.

[Sch 00] B. Scholkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. “New support vector algorithms,” *Neural Computation*, vol. 12, pp. 1207-1245, May 2000.

[Sha 94] N. E. Sharkey and A. J. C. Sharkey, “Understanding catastrophic interference in neural nets,” *Technical Report CS-94-4*, Department of Computer Science, University of Sheffield, UK, 1994.

[She 02] P. O’shea, “A high-resolution spectral analysis algorithm for power-system disturbance monitoring,” *IEEE Transactions on Power System*, vol. 17, pp. 676-680, August 2002.

[Slo 92] S. A. Sloman and D. E. Rumelhart, “Reducing interference in distributed memories through episodic gating,” *Journal of Neurology, Neurosurgery, and Psychiatry*, vol. 20, pp. 11-21, 1992.

[SOL 06] SOLAR research project, School of Electrical Engineering and Computer Science (EECS), Ohio University, 2006, [Online], Available: http://www.ent.ohiou.edu/~webcad/Current_Projects/solar/index.html

[Sta 03a] J. A. Starzyk, Y. Guo, and Z. Zhu, "SOLAR and its hardware development," *Proceedings of Computational Intelligence and Natural Computing (CINC'03)*, North Carolina, USA, September 26-30, 2003.

[Sta 03b] J. A. Starzyk, Z. Zhu, H. He, and Z. Zhu, "Self-organizing learning array and its application to economic and financial problems," *Proceedings of Joint Conference on Information Systems*, North Carolina, USA, September 26-30, 2003.

[Sta 04] J. A. Starzyk, and F. Wang, "Dynamic probability estimator for machine learning," *IEEE Transactions on Neural Networks*, vol. 15, pp. 298-308, March 2004.

[Sta 05a] J. Starzyk, Z. Zhu, and T. H Liu, "Self-organizing learning array," *IEEE Transactions on Neural Network*, vol. 16, no. 2, pp. 355-363, March 2005.

[Sta 05b] J. A. Starzyk, M. Ding, and H. He, "Optimized interconnections in probabilistic self-organizing learning," *Proceedings of International Conference on Artificial Intelligence and Applications*, Innsbruck, Austria, February 14-16, 2005.

[Sta 06a] J. A. Starzyk, Y. Li, and Z. Zhu, "Associative Learning in Hierarchical Self Organizing Learning Arrays," 2006. (Under review)

[Sta 06b] J. A. Starzyk, H. He, and Y. Li, "Probability based self-organizing associative memory," 2006. (Under review).

[Sta 06c] J. A. Starzyk and H. He, “Anticipation based temporal sequence learning in hierarchical structure,” 2006. (Under review).

[Sta 06d] J. A. Starzyk and H. He, “A novel low power logic circuit design scheme,” *IEEE Transactions on Circuits and Systems, Part II: Express Briefs*, 2006 (to appear).

[Sun 01] R. Sun, C. L. Giles, “Sequence learning: from recognition and prediction to sequential decision making,” *IEEE Intelligent Systems*, pp. 2-5, July/August 2001.

[Sut 98] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[Swi 96] N. V. Swindale, “The development of topography in the visual cortex: a review of models,” *Network: Computation in Neural Systems*, vol. 7, pp. 161-247, 1996.

[Tri 04] J. Triesch, “Synergies between intrinsic and synaptic plasticity in individual model neurons,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 17, 2004.

[Umi 95] K. Uming and P. T. Balsara, “Short-circuit power driven gate sizing technique for reducing power dissipation,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol 3, pp. 450-455, September 1995.

[Vap 98] V. Vapnik, *Statistical Learning Theory*, Wiley, New York, NY, 1998.

[Vit 85] E. A. Vittoz, “The Design of High-Performance Analog Circuits on Digital CMOS Chips,” *IEEE Journal of Solid-State Circuits*, vol. 20, issue 3, pp. 657-665, June 1985.

[Vit 90 a] E. A. Vittoz, “Analog VLSI implementation of neural networks,” *IEEE International Symposium on Circuit and Systems*, vol.4, pp. 2524-2527, May 1990.

[Vit 90 b] E. A. Vittoz, “Future of analog in the VLSI environment,” *IEEE International Symposium on Circuit and Systems*, vol.2, pp. 1372-1375, May 1990.

[Vit 96] E. A. Vittoz, “Biology inspired circuits,” *IEEE Micro*, vol. 16, issue 5, pp. 10, October 1996

[Vit 98] E. A. Vittoz, “Analog VLSI for collective computation,” *IEEE International Conference on Electronics, Circuits and Systems*, vol. 2, pp. 3–6, September 1998.

[Vog 97] D. Vogel and W. Boos, “Sparsely connected, Hebbian networks with strikingly large storage capacities,” *Neural Networks*, vol. 4, no. 10, pp. 671-682, 1997.

[Vog 98] D. Vogel, “Auto-associative memory produced by disinhibition in a sparsely connected network,” *Neural Networks*, vol. 5, no. 11, pp. 897-908, 1998.

[Vog 01] D. Vogel, “A biologically plausible model of associative memory which uses disinhibition rather than long term potentiation,” *Brain and Cognition*, vol. 45, no. 2, pp. 212-228, 2001.

[Vog 05] D. Vogel, “A neural network model of memory and higher cognitive functions,” *International Journal of Psychophysiology*, vol. 55, pp. 3-21, 2005.

[Wan 90a] D. Wang and M. A. Arbib, “Complex temporal sequence learning based on short-term memory,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1536-1543, September 1990

[Wan 90b] D. Wang, J. Buhman, and C. Malsburg, “Pattern segmentation in associative memory,” *Neural Computation*, vol. 2, pp. 94-106, 1990.

[Wan 93] D. Wang and M. A. Arbib, “Timing and chunking in processing temporal order,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no.4, pp. 993-1009, July/August 1993

[Wan 95] D. Wang and B. Yuwono, “Anticipation-based temporal pattern generation,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 4, pp. 615-628, April 1995.

[Wan 96] D. Wang and B. Yuwono, “Incremental learning of complex temporal patterns,” *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1465-1481, November 1996

[Wan 98] L. Wang, “Learning and retrieving spatio-temporal sequences with any static associative neural network,” *IEEE Transactions on Circuit and Systems-II: analog and digital signal processing*, vol. 45, no. 6, pp. 729-738, June 1998

[Wan 99] L. Wang, “Multi-associative neural networks and their applications to learning and retrieving complex spatio-temporal sequences,” *IEEE Transactions System, Man, and Cybernetics, part B, Cybernetics*, vol. 29, pp. 73-82, February 1999.

[Wan 01] J. H. Wang, M. C. Tsai, and W. S. Su, “Learning temporal sequences using dual-weight neurons,” *Journal of Chinese Institute Engineers*, vol.24, no. 3, pp.

329-344, 2001.

[Wan 05] M. Wang and S. Chen, “Enhanced EMAM based on empirical kernel map,” *IEEE Transactions on Neural Network*, vol. 16, pp. 557-563, May 2005.

[Wij 02] J. V. Wijayakulasooriya, G. A. Putrus, and P. D. Minns, “Electric power quality disturbance classification using self-adapting artificial neural networks,” *IEE Proceedings – Generation, Transmission and Distribution*, vol. 149, pp. 98 – 101, January 2002.

[Won 98] P. K. Wong and C. Chan, “Off-line handwritten Chinese character recognition as a compound bayes decision problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no.9, pp. 1016-1023, September 1998.

[Wu 00] Y. Wu and S. N. Batalama, “An efficient learning algorithm for associative memories,” *IEEE Transactions on Neural Networks*, vol. 11, pp. 1058-1066, September 2000.

[Wu 01] Y. Wu and S. N. Batalama, “Improved one-shot learning for feedforward associative memories with application to composite pattern association,” *IEEE Transactions on System, Man, and Cybernetics, part B, Cybernetics*, vol. 31, pp. 119-125, February, 2001.

[Yan 01] H. T. Yang and C. C. Liao, “A De-noising scheme for enhancing wavelet-based power quality monitoring system,” *IEEE Transactions on Power Delivery*, vol. 16, pp. 353-360, July, 2001.

[Yue 99] C. P. Yue and S. S. Wong, “Design strategy of on-chip inductors for highly integrated RF systems,” *Design Automation Conference (DAC’99)*, pp. 982-987, New Orleans, Louisiana.

[Zhu 06] Z. Zhu, H. He, J. A. Starzyk, and C. Tseng, “Self-organizing learning array and its Application to Economic and Financial Problems,” *Information Science*, 2006 (to appear)

Appendix A: Source Code and Data Sets Used in Dissertation

The source code and data set packages used in this dissertation are available at the following address.

http://www.ent.ohiou.edu/~webcad/dissertation/haibohe.html

You can also email me to ask for a copy of these documents.

Email: haibohe@bobcat.ent.ohiou.edu

Software packages:

Value_System:	package for the dynamic value system
Associative_Memory:	package for the self-organizing associative memory
Sequence_Learning:	package for the hierarchical sequence learning machine
SOLAR_SVM:	package for the SOLAR and SVM power quality classification