ABSTRACT

DONG, PUXUAN. Design, Analysis and Real-Time Realization of Artificial Neural Network for Control and Classification. (Under the direction of Dr. Griff L. Bilbro.)

Artificial neural networks (ANNs) are parallel architectures for processing information even though they are usually realized on general-purpose digital computers. This research has been focused on the design, analysis and real-time realization of artificial neural networks using programmable analog hardware for control and classification.

We have investigated field programmable analog arrays (FPAAs) for realizing artificial neural networks (ANN). Our research results and products include a general theoretical limit on the number of neurons required by an ANN to classify a given number of data points, a design methodology for the efficient use of specific FPAA resources in ANN applications, several multi-chip FPAA implementations of ANNs for classification experiments, several single-chip FPAA implementations of analog PID controllers for an unmanned ground vehicle (UGV), experimental evaluation of FPAA PID controllers with a conventional digital PID controller on a UGV, and finally a single-chip FPAA implementation of a (non-linear) ANN controller for comparison with the previous FPAA PID controller on a UGV.

These results are collected as four papers formatted for publication and comprising chapters 3, 4, 5, and 6 of this thesis. The first paper develops our general bound for neural network complexity. The second presents a systematic approach based on the upper bound theory for implementing and simplifying neural network structures in FPAA technology. In the third paper, a FPAA based PID controller was designed and characterized in a path-tracking UGV; some of the results from this report are used as a baseline in the fourth paper. In the fourth paper, a FPAA based ANN controller is designed to control a path-tracking UGV and is investigated analytically and with simulation before its performance was experimentally compared to the previously designed FPAA PID controller regarding speed, stability and robustness.

In conclusion, this dissertation focuses on the design, analysis and real-time realization of artificial neural networks. The proposed upper bound for neural network complexity provides guidelines for reducing hardware requirements and applies to any layered ANN approach to classification. It is complemented by the neural network structure simplification method which exploits specific features available in the FPAA technology which we used in our experiments and which we believe possess great potential for future real-time control and classification applications.

DESIGN, ANALYSIS AND REAT-TIME REALIZATION OF ARTIFICIAL NEURAL NETWORK FOR CONTROL AND CLASSIFICATION

by

PUXUAN DONG

A dissertation submitted to the Graduate Faculty of

North Carolina State University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

ELECTRICAL ENGINEERING

Raleigh

2006

APPROVED BY:

_____	_____
Dr. Mo-Yuen Chow	Dr. Rhett Davis
_____	_____
Dr. Mohan Putcha	Dr. Wesley E. Snyder

_____

Dr. Griff L. Bilbro Chair of Advisory Committee

BIOGRAPHY

Puxuan Dong was born in Tangshan, China. He received a B.S. degree in Physics and a B.A. degree in English both in 1998 from Tianjin University. China. He continued his research until 2000 in the physics department.

In 2000, being awarded with fellowship, he went to Dartmouth College to study physics. He then transferred to the Department of Electrical and Computer Engineering at North Carolina State University in 2001. He obtained his M.S. degree in Electrical Engineering in 2003. During the spring and summer of 2005, he worked as co-op electrical engineer at ATI – Industrial Automation designing mixed-signal circuits for high-precision force/torque sensors. He joined the Advanced Diagnosis And Control (ADAC) lab directed by Dr. Mo-Yuen Chow in 2005. He is currently a Ph.D. candidate at North Carolina State University. His main interests are reconfigurable hardware design, computational intelligence and mechatronics systems. His current research topic is design, analysis and real-time realization of artificial neural networks using hardware for control and classification.

ACKNOWLEDGEMENTS

First of all, I cordially thank Dr. Griff Bilbro for his invaluable guidance, great encouragement and persistent support during the course of this research. It is he who led me into the world of analog circuit design, built up my knowledge and confidence, and shaped my philosophy in science. I would like to express my sincere appreciation to my other committee members: Dr. Mo-Yuen Chow, Dr. Wesley Snyder, Dr. Rhett Davis and Dr. Mohan Putcha for their discussions, valuable suggestions and encouragement.

I would also like to acknowledge the help from the members in Dr. Chow's ADAC lab: Rangsarit Vanijjirattikhan, Le Xu, Zheng Li, Tao Hong, Rachana Gupta and Manas Talukdar. I have benefited enormously from the inspiring academic atmosphere in the ADAC lab.

My special thanks go to electronic lab staff member Rudy Salas for his help in hardware facilities.

I wish to express my deepest thanks to my wife Shen, Fei, my parents, my brother, and my parents in law for their love and continuous support. I also wish to thank my daughter Elaine Fei Dong, who was born in Sept. 2005. She also became part of my mental support for my study although it may take time for her to understand it.

My thanks also go to all my friends at North Carolina State University which I could not mention all of them here.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I - Introduction

Artificial neural networks (ANNs) play an increasingly important role in areas such as robotics [1], process control [2-3], and motor fault detection [4-6]. Both software and hardware based approaches have been used for implementing ANNs. In general, software instructions executed serially cannot take advantage of the inherent parallelism of ANN architectures. Hardware implementations of neural networks promise higher speed operation when they can exploit this massive parallelism. Different hardware implements of neural network have been reported [7-25]. Other than the FPGA based approaches [10, 11, 18, 24], most hardware implementations provide no programmability even though real-time reconfigurability of the topology or the size of an ANN could presumably improve its performance in applications where its immediate environment changes or its immediate objective is updated.

The best choices for neural network implementations that achieve both high speed and rapid prototyping appear to be programmable hardware approaches like field programmable gate arrays (FPGAs) and field programmable analog arrays (FPAAs). Compared to digital hardware, FPAAs have the advantage of interacting directly with the real (analog) world because they receive, process, and transmit signals totally in the analog domain without need of A/D or D/A conversion. Their speed is also suited to real time applications. As reported in [26] on controlling a path-tracking unmanned ground

vehicle (UGV), the FPAA easily outperformed comparable digital hardware by processing the signal 8,000 times faster. Few Anadigm FPAA applications have been reported, except a voltage-to-frequency converter and a Hodgkin-Huxley neuron simulator [27-28].

Hardware requirements are important for economical implementations. Realization of ANN structures with minimal hardware is facilitated by an understanding of the general limits on the complexity required by an ANN structure for it to learn a set of examples of a certain size. This maximum complexity can be expressed as an upper bound on the neural network size. Designs of this size are guaranteed to be large enough for correct operation. Smaller ANNs might possibly operate correctly depending on the details of the problem, but any larger ANN are now guaranteed *a-priori* to waste neurons regardless of any particulars of the problem.

A few such bounds were known previously in certain cases. It was known that a single-hidden layer feed forward neural network with two hidden nodes can solve the nonlinearly separable XOR problem. XOR has only two input variables and only four data patterns, but is typical except for its size. For larger problems with more inputs and data patterns, general upper bounds on the complexity of neural networks have been unavailable until now. My general bound has been useful for increasing the size of ANNs that are feasible with the hardware available but in this research, ANN size was still constrained by hardware resources. For complicated tasks, further simplifications were developed to take advantage of some specific features of FPAA technology.

Following the introductory part of the dissertation, a literature survey of neural network hardware and the introduction to the FPAA technology is presented in Chapter II. Implementation examples of simple ANN structures are also present in the same chapter. Chapter III presents an upper bound on the complexity of feed forward neural networks - a single-hidden-layer network with at most (N/2)+1 hidden neurons is sufficient to classify N (N is even) data points of 2 classes with zero error. The upper bound reduces to (N-1)/2+1 when N is odd. The theory is then applied to design and train a neural classifier for the two-spiral problem, a benchmark problem in the neural network literature. Chapter IV applies the theory to solve another classification task and implements the ANN in the FPAA. A further neural network structure simplification technique for FPAA based ANNs is also proposed in Chapter IV. In Chapter V, an FPAA based PID controller is designed to control a path tracking unmanned ground vehicle for future performance comparison with FPAA based ANN controller. At last in Chapter VI, to demonstrate the application of FPAA based ANN in control systems, an ANN is designed in FPAA to control a path tracking unmanned ground vehicle. The performance of the FPAA based ANN controller is characterized in terms of speed, stability and robustness.

References

[1] F. L.Lewis, "Neural-network control of robot manipulators," *IEEE Expert*, pp. 64-75, June 1996.

[2] J. Teeter and M.-Y. Chow, "Application of Functional Link Neural Network to HVAC Thermal Dynamic System Identification," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 1, pp. 170-176, 1998.

[3] M.-Y. Chow and J. Teeter, “A Knowledge-Based Approach for Improved Neural Network Control of a Servomotor System with Nonlinear Friction Characteristics,” *Mechatronics*, vol. 5, no. 8, pp. 949-962, 1995.

[4] B. Ayhan, M.-Y. Chow, and M.-H. Song, “Monolith and Partition Schemes with LDA and Neural Networks as Detector Units for Induction Motor Broken Rotor Bar Fault Detection,” *KIEE International Transactions on Electrical Machinery and Energy Conversion Systems*, June 1, 2005 (invited).

[5] M.Y. Chow, “Methodologies of Using Artificial Neural Network and Fuzzy Logic Technologies for Motor Incipient Fault Detection,” *World Scientific Publishing Co. Pte. Ltd.*, 1998.

[6] M.-Y. Chow, G. Bilbro, and S. O. Yee, “Application of Learning Theory to a Single Phase Induction Motor Incipient Fault Detection Artificial Neural Network*,” International Journal of Neural Systems*, vol. 2, no. 1&2, pp. 91-100, 1991.

[7] M. Holler, S. Tam, H. Castro and R. Benson, “An electrically trainable artificial neural network (ETANN) with 10240 `floating gate' synapses ,” *Neural Networks, 1989. IJCNN, International Joint Conference on*, pp. 191 - 196 vol.2, 18-22 June 1989.

[8] S. Tam, B. Gupta, H. Castro and M. Holler, “Learning on an Analog VLSI Neural Network Chip,” *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*, 1990.

[9] Y. Maeda, H. Hirano and Y. Kanata, “AN Analog Neural Network Circuit with a Learning Rule via Simutaneous Perturbation,” *Proceedings of the IJCNN-93-Nagoya*, pp. 853-856, 1993.

[10] S. S. Kim [1]and S. Jung, "Hardware implementation of a real time neural network controller with a DSP and an FPGA," *presented at Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, 2004.

[11] S. S. Kim W. Qinruo, Y. Bo, X. Yun, and L. Bingru, "The hardware structure design of perceptron with FPGA implementation," *presented at Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 2003.

[12] S. B. Yun, Y. J. Kim, S. S. Dong, and C. H. Lee, "Hardware implementation of neural network with expansible and reconfigurable architecture," *presented at Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*, 2002.

[13] H. Withagen, “Implementing Backpropagation with Analog Hardware,” *Proceedings of the IEEE ICNN-94-Orlando Florida*, pp. 2015-2017, 1994.

[14] T. Szabo, L. Antoni, G. Horvath, and B. Feher, "A full-parallel digital implementation for pre-trained NNs," *presented at Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, 2000.

[15] S. Popescu, "Hardware implementation of fast neural networks using CPLD," *presented at Neural Network Applications in Electrical Engineering, Proceedings of the 5th Seminar on*, 2000.

[16] B. Girau, "Digital hardware implementation of 2D compatible neural networks," *presented at Neural Networks, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, 2000.

[17] H. Abdelbaki, E. Gelenbe, and S. E. EL-Khamy, "Analog hardware implementation of the random neural network model," *presented at Neural Networks, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, 2000.

[18] J. Zhu, G. J. Milne, and B. K. Gunther, "Towards an FPGA based reconfigurable computing environment for neural network implementations," *presented at Artificial Neural Networks, Ninth International Conference on*, 1999.

[19] J. Liu and M. Brooke, "A fully parallel learning neural network chip for real-time control," *presented at Neural Networks, International Joint Conference on*, 1999.

[20] J. Liu and M. Brooke, "Fully parallel on-chip learning hardware neural network for real-time control," *presented at Circuits and Systems, Proceedings of the IEEE International Symposium on*, 1999.

[21] E. J. Brauer, J. J. Abbas, B. Callaway, J. Colvin, and J. Farris, "Hardware implementation of a neural network pattern shaper algorithm," *presented at Neural Networks, International Joint Conference on*, 1999.

[22] P. M. Engel and R. F. Molz, "A new proposal for implementation of competitive neural networks in analog hardware," *presented at Neural Networks, Proceedings. 5th Brazilian Symposium on*, 1998.

[23] J. Tang, M. R. Varley, and M. S. Peak, "Hardware implementations of multi-layer feed forward neural networks and error backpropagation using 8-bit PIC microcontrollers," *presented at Neural and Fuzzy Systems: Design, Hardware and Applications, IEE Colloquium on*, 1997.

[24] D. S. Reay, T. C. Green, and B. W. Williams, "Field programmable gate array implementation of a neural network accelerator," *presented at Hardware Implementation of Neural Networks and Fuzzy Logic, IEE Colloquium on*, 1994.

[25] A. Achyuthan and M. I. Elmasry, "Mixed analog/digital hardware synthesis of artificial neural networks," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 13, pp. 1073-1087, 1994.

[26] P. Dong, G. Bilbro, and M.-Y. Chow, "Controlling a Path-tracking Unmanned Ground Vehicle with a Field-Programmable Analog Array," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Monterey, CA*, 24-28 July, 2005.

[27] P. I. Yakimov, E. D. Manolov, and M. H. Hristov, "Design and implementation of a V-f converter using FPAA," *presented at Electronics Technology: Meeting the Challenges of Electronics Technology Progress, 2004. 27th International Spring Seminar on*, 2004.

[28] M. Sekerli and R. J. Butera, "An implementation of a simple neuron model in field programmable analog arrays," *presented at Engineering in Medicine and Biology Society, 2004. EMBC 2004. Conference Proceedings. 26th Annual International Conference of the*, 2004.

CHAPTER II - Introduction to Field Programmable Analog Arrays and Survey of Artificial Neural Network Hardware

This chapter introduces FPAA technologies for ANN applications and surveys the literature of artificial neural network (ANN) hardware. Some ANN implementation examples using FPAAs are also presented.

I. The FPAA Technology

The field programmable analog array technology, which is the analog counterpart of the FPGA, appeared in 1980's [1-8]. The technology was made commercially available by Anadigm^{TM} in 2000. Anadigm's FPAA chips are mainly used as platforms for experiments along with our research.

The FPAA is an array of identical Configurable Analog Blocks (CABs). Which includes operational amplifiers, comparators and switched programmable capacitors . The FPAA allows designers to implement an extremely wide variety of signal processing functions using digital configuration data.

Anadigm's FPAAs are based on switched-capacitor technology. Switched capacitors take the place of resistors in switched-capacitor circuits. An effective resistance can be

defined for switched capacitors; its value depends on the capacitance but changes according to the sampling frequency.

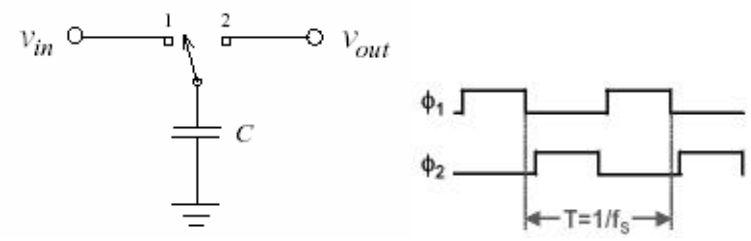


Figure 2.1. Switched capacitor and its sampling clocks

As shown in Figure 2.1, capacitor C is a switched capacitor; non-overlapping clocks control two switches respectively. V_{in} is sampled at the falling edge of ϕ_1, the sampling frequency is f_s. The charges going from V_{in} to V_{out} each sampling period is $Q = C(V_{in} - V_{out})$. So the current flowing from V_{in} to V_{out} is $I = f_s C(V_{in} - V_{out})$. Then we can get the equivalent resistor of resistance $R = \frac{1}{f_s C}$. The following two examples show that how the switched capacitor could take the place of the resistor in an inverting integrator and a non-inverting integrator.

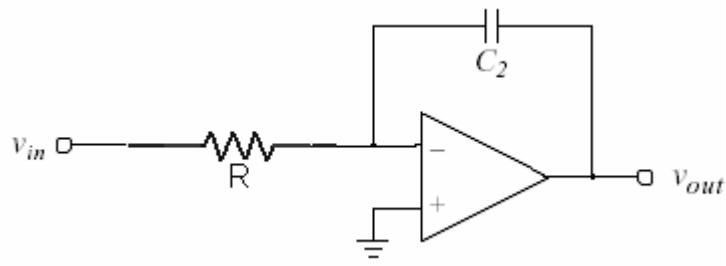


Figure 2.2. A normal inverting integrator

To replace the resistor with a switched capacitor, the first switched-capacitor design used following circuit:

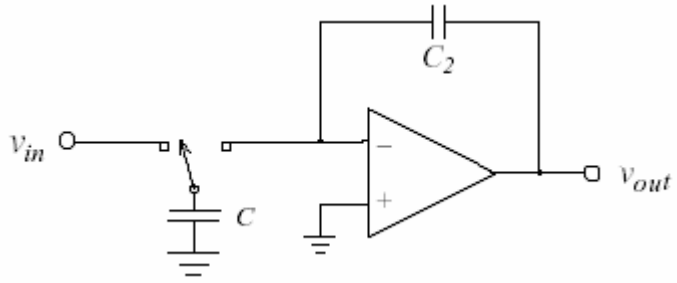


Figure 2.3. An inverting integrator with the resistor replaced by a switched capacitor

The advantage of the switched capacitors is that these are easy to build in an integrated circuit (IC) technology. It is difficult to make precise large-value resistors on silicon, but easier to make precise (and well-matched) small-value capacitors. When the switching noise is eliminated by filtering, a simple use of switched capacitor is as a resistor. For example, an effective resistance of 100 kilohms can be obtained at a switching frequency of 1 MHz by a capacitor with a capacitance of 10 pF. Switched-capacitor technology is key to the accuracy and flexibility of FPAA.

At present, Anadigm has shipped two generations of FPAA chips (AN10E40 and ANx2xE0x) and corresponding evaluation boards (shown in Figure 2.4 & 2.5). The frequencies of the master clock are 1Mhz and 4Mhz for the first-generation board and second-generation board respectively.

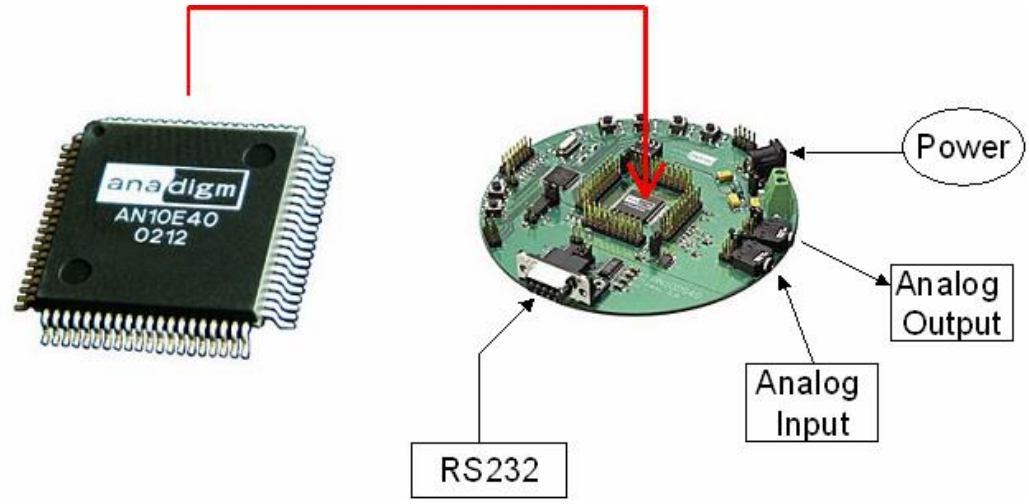


Figure 2.4. AN10E40 FPAA Chip and Evaluation Board

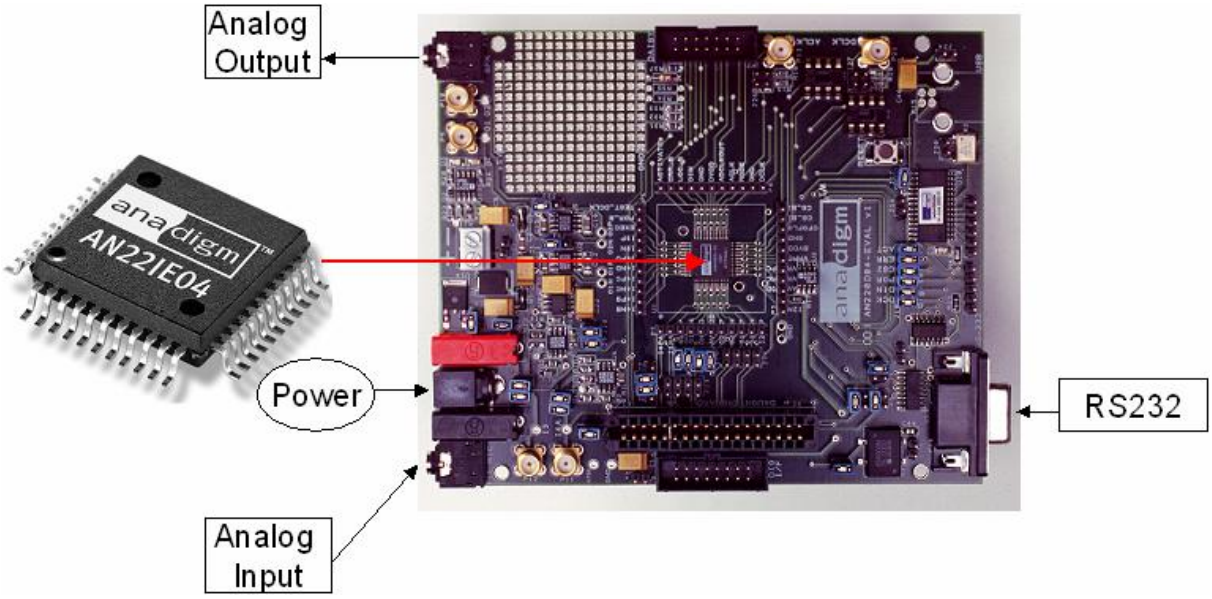


Figure 2.5. AN221E04 FPAA Chip and Evaluation Board

II. Introduction to Artificial Neural Networks

Artificial neural networks (ANNs), or more simply neural networks (NNs), are information processing systems that roughly replicate the behavior of a human brain by emulating the operations and connectivity of biological neurons. Under some

conditions, they can be trained directly from data, and sometimes complicated or imprecise data. NNs can be used to extract patterns and detect trends thus it be applied to data classification and nonlinear functional mapping. Specific application examples include process modeling, control, machine diagnosis, and real-time recognition. The history of the ANNs stems from the 1940s, the decade of the first electronic computer. An important step toward artificial neural networks occurred in 1943 when Warren McCulloch and Walter Pitts wrote a paper on the working mechanism of the neurons. They modeled a simple neural network with electrical circuits [9]. However, the first significant application only took place in 1958 when Rosenblatt introduced the first concrete neural model - the perceptron [10]. In 1959, Bernard Widrow and Marcian Hoff of Stanford University developed models they called ADALINE and MADALINE [11-12]. These models were named for their use of Multiple ADAptive LINear Elements. MADALINE was the first neural network to be applied to a real world problem. It is an adaptive filter that eliminates echoes on phone lines. This neural network is still in commercial use. The perceptron model Rosenblatt proposed contained only one layer. And after that, a multi-layered model was derived by Fukushima in 1975 [13]. The multi-layer perceptron network didn't show its potential in various applications until Rummelhart and Mclelland introduced a general backpropagation algorithm for a multi-layered perceptron in 1986 [14-15]. In 1982, Hopfield proposed a type of neural network called "Hopefield network" in which there is only one layer whose neurons are fully connected with each other [16]. In 1982, Kohonen introduced a unique kind of network model called "Self-Organizing Map (SOM)". SOM is a certain kind of topological map which organizes itself based on the input patterns that it is trained with. The SOM

originated from the LVQ (Learning Vector Quantization) network underlying the idea of which was also Kohonen's in 1972 [17].

A. Application Areas of ANNs

Neural networks have been successfully applied to various data-intensive applications in industry, business and science [18]. Those applications include bankruptcy prediction [19-24], handwriting recognition [25-29], speech recognition [30], product inspection [31-32], fault detection [33-34], medical diagnosis [35-37], and bond rating [38-40]. A number of performance comparisons between neural and conventional classifiers have been made by many studies [41-43]. In addition, several computer experimental evaluations of neural networks for classification problems have been conducted under a variety of conditions [44-45].

One example of application to motor fault detection was proposed by Li, Chow, Tipsuwan and Hung [46]. They present an approach for motor rolling bearing fault diagnosis using neural networks and time/frequency-domain bearing vibration domain analysis. The results of motor bearing faults can be effectively diagnosed using neural network if the signal of motor bearing vibration is appropriately measured and translated.

B. Artificial Neuron Model and Neural Network Structures

An artificial neuron (process element) is often called a node or unit. The model is shown in Figure 2.6. It receives input from some other units, or perhaps from an external source. Each input has an associated weight *w*, which can be modified so as to model synaptic learning. The unit computes some function *f* of the weighted sum of its inputs: y = f (x). Its output, in turn, can serve as input to other units. The function *f* is the unit's

activation function. The activation function could be linear function, step function, sigmoid function $f(x)=\frac{1}{1+e^{-x}}$, hyperbolic tangent function $f(x)=\frac{1-e^{-2x}}{1+e^{-2x}}$ and radial basis function $f(x)=e^{-x^2}$.

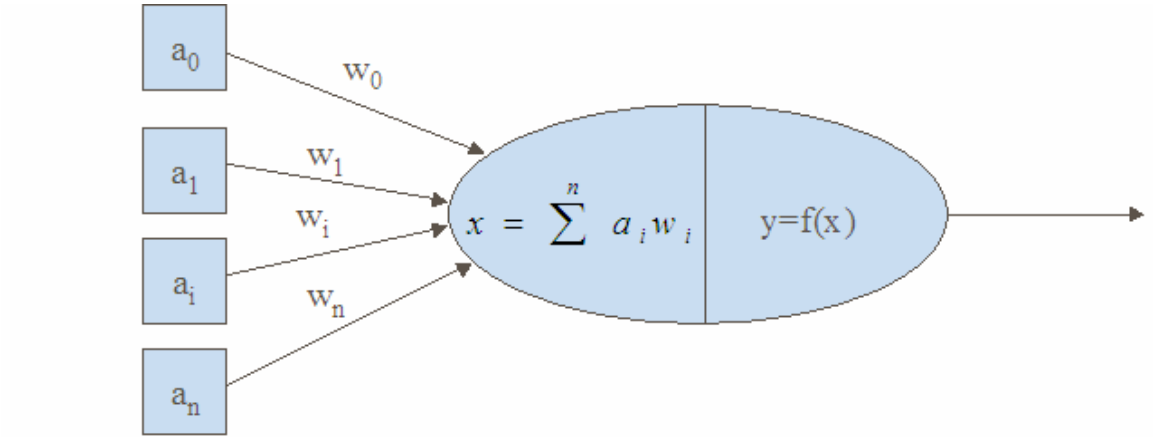


Figure 2.6. Artificial neuron model

A neural network normally consists of many artificial neurons and a large number of interconnections among them as shown in Figure 2.7. Based on the structure of the connections, the neural networks can be classified into two categories: first, the feed forward neural network in which the neurons in one layer get input from the previous layer and the output is connected to the next layer; and second, the recurrent neural network where connections to the neurons are to the same layer to the previous layers. The Hopfield neural network [16] is a widely used example of recurrent neural networks.

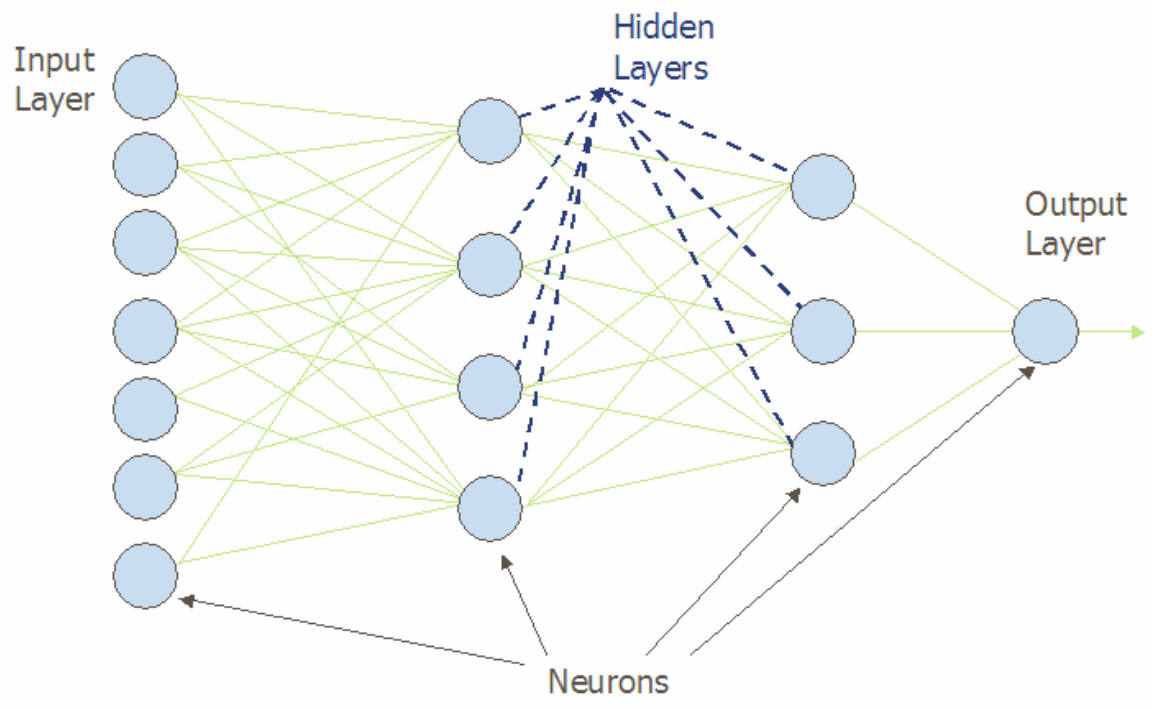


Figure 2.7. Example of feed forward neural network.

C. Overview of ANN Hardware Implementation

Neural network can be implemented with either software or hardware. Software implementation means programming the neural network on a PC or workstation. Nowadays the performance of von-Newman processors like the Intel Pentium series are advancing steadily. However, since they are designed to process instructions more-or-less one after the other instead of concurrently in parallel, alternative ANN simulators might supplement general purpose digital processors. Software implementation is preferred when the neural network is of low neuron density or the specific problem doesn't need high speed processing. When the speedup is of the interest, special purpose hardware implementations are a more attractive choice. For example, some dedicated devices such as those for hand-written character recognition [25-29] and speech recognition [30] are implemented in special purpose hardware. Most hardware with

multiple inputs implementing neural networks has a speed advantage over software since it can take the advantage of the inherent parallelism of the ANN.

In the literature treating the hardware, implementation of neural networks, specifications include the technology used (analog, digital, or hybrid), the precision (in equivalent numbers of bits) of the input/outputs, of the weights, and of the accumulators, etc. There are two traditional criteria for the performance of neural network hardware: the first one is the MCPS or Millions of Connections Per Second, which is defined as the rate of multiply and/or accumulate operations. The other one is MCUPS or Millions of Connection Update Per Second value that denotes the rate of weight changes during network learning.

Efforts have been made to develop a more detailed classification of the neural network hardware from aspects such as system architecture, degree of parallelism or whether general-purpose or special-purpose devices are employed [48].

As for the chips, hardware for neural network implementation can be classified into two main categories: general-purpose hardware that can be reconfigured for different tasks (such as Adaptive Solutions CNAPS [47] and the algorithm specific hardware to implement one specific task very efficiently such as handwriting recognition. The reported general-purpose hardware includes PC accelerator boards and neurocomputers. The reported algorithm specific hardware has three sub-categories: digital, analog and hybrid. The resulting classification is shown in Figure 2.8 describes the hardware reported in references [49-65], [68] and will be used to discuss them in the following.

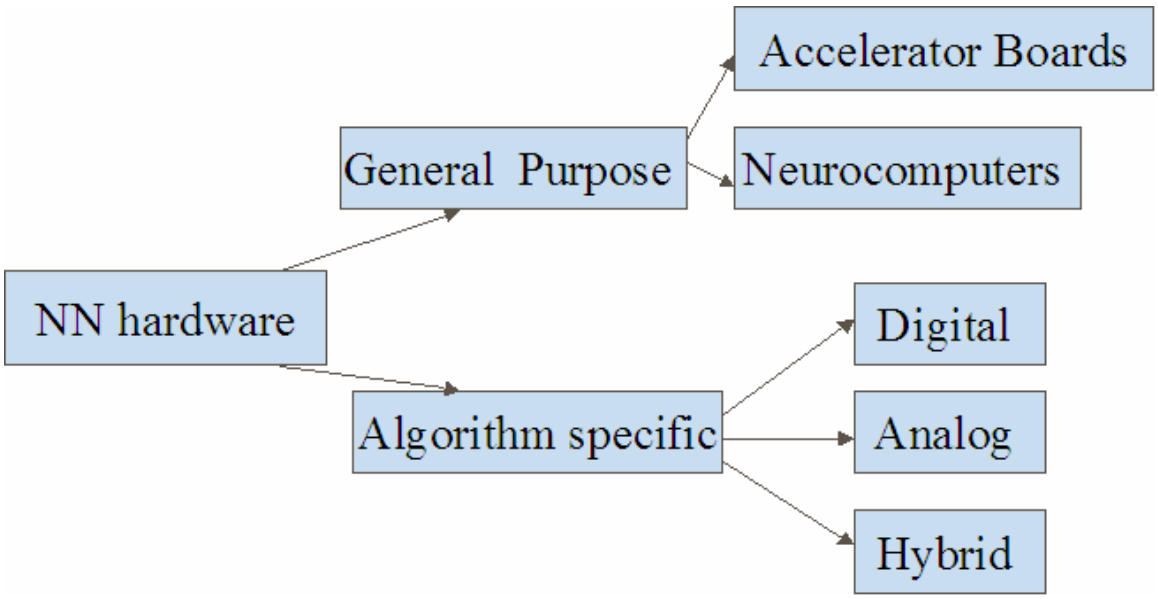


Figure 2.8. Classification of neural network hardware

Accelerator boards are a kind of hardware that can work in conjunction with PC to speed up the neural network implementations. They normally reside in the expansion slot. Accelerator boards are usually based on neural network chips but some just use fast digital signal processors that do multiple-accumulate operations. One example of the accelerator boards is the IBM ZISC ISA and PCI cards [49]. The ZISC is a digital chip with 64 8-bit inputs and 36 radial basis function neurons. Multiple chips can be cascaded together to create networks of arbitrary size. The ISA card holds up to 16 ZISCX036 chips providing 576 neurons and the PCI cards can hold up to 19 chips providing 684 neurons. Other accelerator cards include SAIC SIGMA-1 [50], Neuro Turbo [51] and HNC [52].

Neurocomputers are stand-alone systems that are intended for large scale processing applications such as high throughput optical characters recognition. They are built from general-purpose processors to provide high programmability for implementation of

different neural networks. Neurocomputers are complex and expensive. Examples include BSP400 [53], COKOS [54] and RAP (Ring Array Processor) [55]. RAP was developed at the ICSI (International Computer Science Institute, Berkeley, CA) and has been used as an essential component in the development of connectionist algorithms for speech recognition since 1990. Implementations consist of 4 to 40 Texas Instruments TITMS320C30 floating point DSPs containing 256 Kbytes of fast static RAM and 4 Mbytes of dynamic RAM each. These chips are connected via a ring of FPGAs, each implementing a simple two-register data pipeline. Additionally each board has a VME bus interface logic, which allows it to connect to a host computer.

The algorithm specific neural network hardware (neural network VLSI) can be divided into three broad categories: digital, analog, and hybrids.

The digital neural network category itself includes slice architectures [56-58], single instruction multiple data (SIMD) [59-60] and systolic array devices [61]. For SIMD design, each processor executes the same instruction in parallel but on different data. In systolic arrays, a processor does one step of a calculation (always the same step) before passing its result on to the next processor in a pipelined manner. A systolic array system can be built with the Siemens MA-16 [62]. Digital hardware has the advantages of mature fabrication techniques, weight storage in RAM, and arithmetic operations exact within the number of bits of the operands and accumulators. However, digital operations are usually slower than in analog systems, especially in some computationally expensive part of the neural calculation such as the hyperbolic tangent transfer function, and analog-to-digital and digital-to-analog processing take extra time compared to analog chips.

Compared to digital neural network hardware, analog hardware networks have the advantages of high speed since there is no A/D and D/A conversion needed. The first commercially available analog neural network chip is Intel's analog ETANN chip [63]. Later more analog chips appeared such as [64-65]. Besides the advantages of high speed and high densities, since analog hardware interact directly with the real world and process signals totally in analog domain which is very fast, it is more suitable for real-time applications such as controlling unmanned vehicles compared to digital hardware.

The following section presents a new platform for ANN implementation: field programmable analog arrays (FPAAs), which can be classified into either algorithm specific or general-purpose hardware category, which are mentioned above.

III. Sample Design of ANN Using FPAA

We can conclude from the above survey that to obtain high speed and flexible neural network simulator, programmable analog hardware is desired. This paper proposes the new neural network hardware to be built using FPAA technology, which could be either algorithm specific slice analog chip or large-scale neurocomputer after integrate multiple FPAAs together.

The FPAA shows considerable potential as a neural network simulator. It processes signals totally in the analog domain. It is digitally programmable. Multiple chips (or multiple evaluation boards in the present case, since the FPAA are provided as evaluation boards in my implementations) can be integrated easily to realize artificial neural networks of arbitrary sizes as needed. Moreover, the transfer function of the neurons in

the neural network can be arbitrarily tabulated in a look up table in each FPAA chip. Figure 2.9 is an example circuit implementing a hyperbolic tangent transfer function. The look up table as shown in Figure 2.10 was loaded from an Excel file. Figure 2.11 is the simulation result.

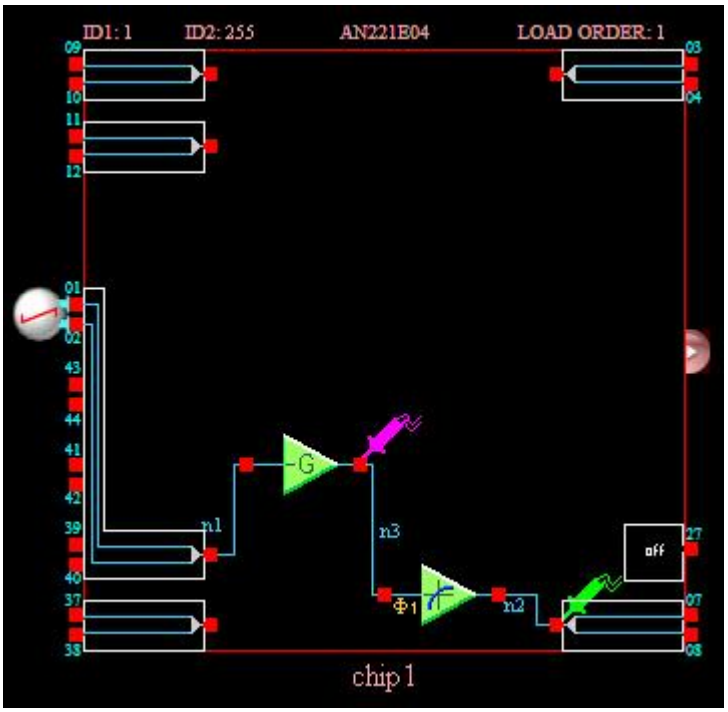


Figure 2.9. FPAA circuit that implements the hyperbolic tangent function using look up table.

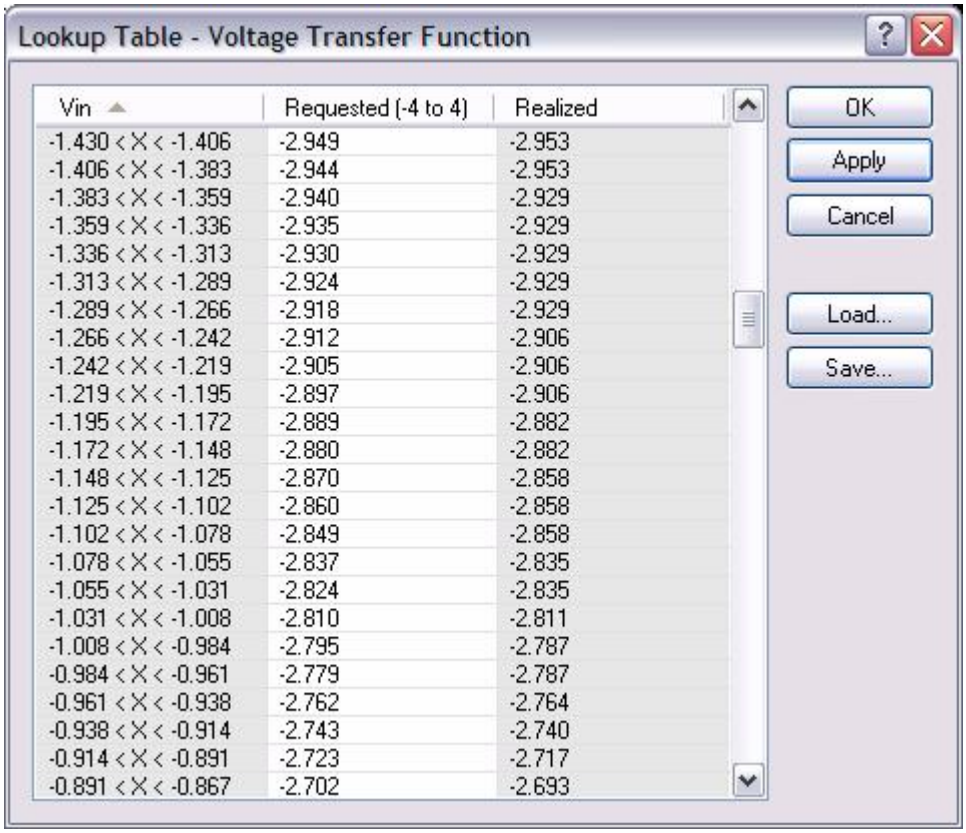


Figure 2.10. Look up table for the hyperbolic tangent transfer function.

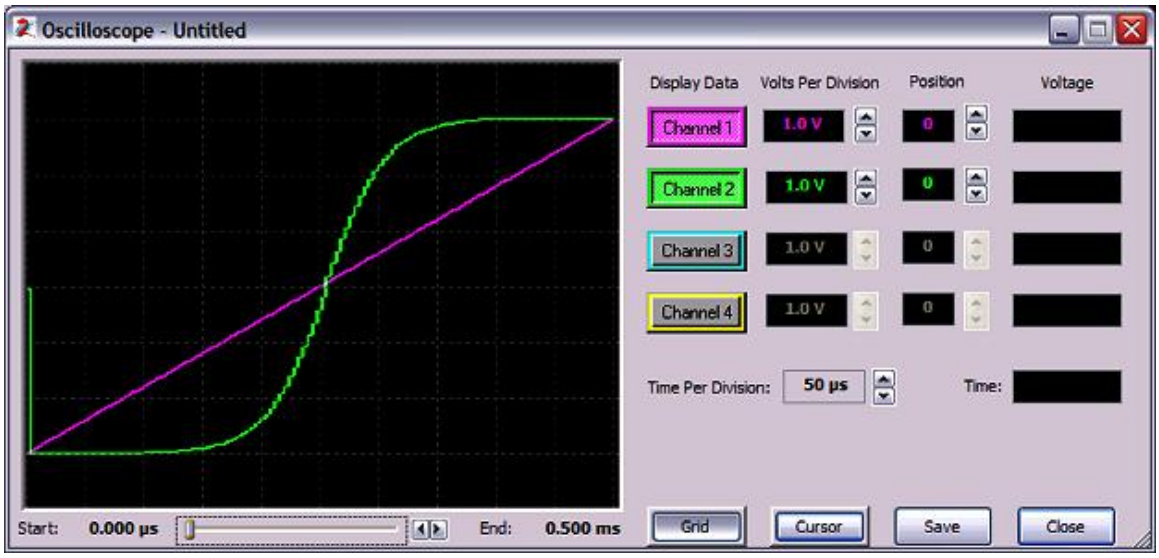


Figure 2.11. Simulation result of hyperbolic tangent transfer function.

IV. References

[1] D. Anderson, C. Marcjan, D. Bersch, H. Anderson, P. Hu, O. Palusinki, D. Gettman, I. Macbeth, A. Bratt, "A Field Programmable Analog Array and its Application," *1997 Custom Integrated Circuits Conference Proceedings*, May, 5-8, 1997, Santa Clara, California, USA.

[2] E.K.F. Lee and W.L. Hui, "A novel switched-capacitor based field-programmable analog array architecture," *Kluwer Analog Integrated Circuits and Signal Processing - Special Issue on Field Programmable Analog Arrays*, Vol. 17, No. 1-2, pp. 35-50, September 1998.

[3] S.H.K. Embabi, X. Quan, N. Oki, A. Manjrekar, and E. Sanchez-Sinencio, "A current-mode based field-programmable analog array for signal processing applications," *Kluwer Analog Integrated Circuits and Signal Processing - Special Issue on Field Programmable Analog Arrays*, Vol. 17, No. 1-2, pp. 125-142, September 1998.

[4] E. Pierzchala and M.A. Perkowski, "A high-frequency field-programmable analog array (FPAA) part I: design," *Kluwer Analog Integrated Circuits and Signal Processing - Special Issue on Field Programmable Analog Arrays*, Vol. 17, No. 1-2, pp. 143-156, September 1998.

[5] E. Lee and G. Gulak, "A CMOS field-programmable analog array," *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 12, pp. 1860-1867, December 1991.

[6] S.T. Chang, B.R. Hayes-Gill, and C.J. Paul, "Multi-function Block for a Switched Current Field Programmable Analog Array", *1996 Midwest Symposium on Circuits and Systems*, August 1996.

[7] K.F.E. Lee and P.G. Gulak, "A Transconductor-Based Field-Programmable Analog Array", *ISSCC Digest of Technical Papers*, pages 198-199, Feb. 1995.

[8] K.F.E. Lee and P.G. Gulak, "A CMOS Field-Programmable Analog Array", *ISSCC Digest of Technical Papers*, pages 186-188, Feb. 1991.

[9] McCulloch, W.S. & Pitts, W.H. "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics,* 5: pp. 115-137, 1943.

[10] Rosenblatt, F. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review,* 65: pp. 386-408, 1958.

[11] B. Widrow, "Generalization and information storage in networks of adaline 'neurons'," *Self-Organizing Systems*, Ed. M. C. Yovits, Washington, DC: Spartan, pp. 435-461, 1962.

[12] B. Widrow and M.E. Hoff, "Adaptive Switching Circuits," *IRE WESCON Convention Record*, New York: IRE pp. 96-104, 1960.

[13] K. Fukushima, "Cognitron: A Self-organizing Multilayered Neural Network," *Biological Cybernetics,* 20: pp. 121-136, 1975.

[14] D.B. Parker, "A Comparison of Algorithms for Neuron-Like Cells," In J.S. Denker (Ed.), *Neural Networks for Computing, New York: American Institute of Physics,* pp. 327-332, 1986.

[15] D.D. Rumelhart, G.E. Hinton and R.J. Williams, Ronald J. "Learning Representations by Back-Propagating Errors," *Nature* 323: pp. 533-536, 1986.

[16] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences,* 79: pp. 2554-2558, 1982.

[17] T. Kohonen, "Correlation Matrix Memories," *IEEE Transaction on Computers,* C-21: pp. 353-359, 1972.

[18] B. Widrow, D. E. Rumelhard, and M. A. Lehr, "Neural networks: Applications in industry, business and science," *Communications. AC*M, vol. 37, pp. 93–105, 1994.

[19] E. I. Altman, G. Marco, and F. Varetto, "Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (theItalian experience)," *Jouranl of Bank and Financ*e, vol. 18, pp. 505–529, 1994.

[20] R. C. Lacher, P. K. Coats, S. C. Sharma, and L. F. Fant, "A neural net-work for classifying the financial health of a firm," *European Journal of Operations Research.*, vol. 85, pp. 53–65, 1995.

[21] M. Leshno and Y. Spector, "Neural network prediction analysis: Thebankruptcy case," *Journal of Neurocomputing*, vol. 10, pp. 125–147, 1996.

[22] K. Y. Tam and M. Y. Kiang, "Managerial application of neural networks:The case of bank failure predictions," *Management Science*., vol. 38, no. 7, pp. 926–947, 1992.

[23] R. L. Wilson and R. Sharda, "Bankruptcy prediction using neural net-works," *Decision Support System*, vol. 11, pp. 545–557, 1994.

[24] G. Zhang, M. Y. Hu, E. B. Patuwo, and D. Indro, "Artificial neural networks in bankruptcy prediction: General framework and cross-valida-tion analysis," *European Journal of Operations Research*, vol. 116, pp. 16–32, 1999.

[25] I. Guyon, "Applications of neural networks to character recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 5, pp. 353–382, 1991.

[26] S. Knerr, L. Personnaz, and G. Dreyfus, "Handwritten digit recognition by neural networks with single-layer training," *IEEE Transactions on Neural Network*s, vol. 3, pp. 962–968, 1992.

[27] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubberd, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," *Advanced Neural Information Processing System*, vol. 2, pp. 396–404, 1990.

[28] D. S. Lee, S. N. Sriharia, and R. Gaborski, "Bayesian and neural-net-work pattern recognition: A theoretical connection and empirical results with handwritten characters," in *Artificial Neural Networks and Statistical Pattern Recognitio*n, I. K. Sethi and A. K. Jain, Eds. New York: Elsevier, pp. 89–108, 1991.

[29] G. L. Martin and G. L. Pitman, "Recognizing hand-printed letter and digits using backpropagation learning," *Neural Computing*, vol. 3, pp. 258–267, 1991.

[30] H. Bourlard and N. Morgan, "Continuous speech recognition by connectionist statistical methods," *IEEE Transactions on Neural Network*s, vol. 4, pp. 893–909, 1993.

[31] J. Lampinen, S. Smolander, and M. Korhonen, "Wood surface inspection system based on generic visual features," in *Industrial Applications of Neural Network*s, F. F. Soulie and P. Gallinari, Eds, Singapore: World Scientific, pp. 35–42, 1998.

[32] T. Petsche, A. Marcantonio, C. Darken, S. J. Hanson, G. M. Huhn, and I. Santoso, "An autoassociator for on-line motor monitoring," in *Industrial Applications of Neural Network*s, F. F. Soulie and P. Gallinari, Eds, Singapore: World Scientific, pp. 91–97, 1998.

[33] E. B. Barlett and R. E. Uhrig, "Nuclear power plant status diagnostics using artificial neural networks," *Nuclear Technology*, vol. 97, pp. 272–281, 1992.

[34] J. C. Hoskins, K. M. Kaliyur, and D. M. Himmelblau, "Incipient fault detection and diagnosis using artificial neural networks," in *International Joint Conference on Neural Network*s, pp. 81–86, 1990.

[35] W. G. Baxt, "Use of an artificial neural network for data analysis in clinical decision-making: The diagnosis of acute coronary occlusion," *Neural Computing*, vol. 2, pp. 480–489, 1990.

[36] H. B. Burke, “Artificial neural networks for cancer research: Outcome prediction,” *Seminars in Surgical Oncology*, vol. 10, pp. 73–79, 1994.

[37] H. B. Burke, P. H. Goodman, D. B. Rosen, D. E. Henson, J. N. Wein-stein, F. E. Harrell, J. R. Marks, D. P. Winchester, and D. G. Bostwick, “Artificial neural networks improve the accuracy of cancer survival pre-diction,” *Cance*r, vol. 79, pp. 857–862, 1997.

[38] S. Dutta and S. Shekhar, “Bond rating: A nonconservative application of neural networks,” in *Proc. IEEE International Conference on Neural Network*s, vol. 2, San Diego, CA, pp. 443–450, 1988.

[39] A. J. Surkan and J. C. Singleton, “Neural networks for bond rating improved by multiple hidden layers,” in *Proc. IEEE International Joint Conference on Neural Network*s, vol. 2, San Diego, CA, pp. 157–162, 1990.

[40] J. Utans and J. Moody, “Selecting neural network architecture via the prediction risk: Application to corporate bond rating prediction,” in *International Conference on Artificial Intelligence Applications Wall Stree*t, pp. 35–41, 1991.

[41] S. P. Curram and J. Mingers, “Neural networks, decision tree induction and discriminant analysis: An empirical comparison,” *Journal of Operational Research Society*, vol. 45, no. 4, pp. 440–450, 1994.

[42] W. Y. Huang and R. P. Lippmann, “Comparisons between neural net and conventional classifiers,” in *IEEE International Conference on Neural Network*s, San Diego, CA, pp. 485–493, 1987.

[43] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, Eds., *Machine Learning, Neural, and Statistical Classificatio*n, London, U.K.: Ellis Horwood, 1994.

[44] E. Patwo, M. Y. Hu, and M. S. Hung, “Two-group classification using neural networks,” *Decision Science*, vol. 24, no. 4, pp. 825–845, 1993.

[45] V. Subramanian, M. S. Hung, and M. Y. Hu, “An experimental evaluation of neural networks for classification,” *Computers and Operations Research*., vol. 20, pp. 769–782, 1993.

[46] B. Li, M.-Y. Chow, Y. Tipsuwan, J. C. Hung, "Neural Network Based Motor Vibration Signal Analysis, Fault Detection, and Diagnosis," *IEEE Transactions on Industrial Electronics Special Issue on Motor Fault Detection and Diagnosis*, October, 2000.

[47] H. McCartor, "A Highly Parallel Digital Architecture for Neural Network Emulation," *In Delgado-Frias, J. G. and Moore, W. R. (eds.), VLSI for Artificial Intelligence and Neural Networks*, Plenum Press, New York, pp 357-366, 1991.

[48] T. Schoenauer, A. Jahnke, U. Roth and H. Klar, "Digital Neurohardware: Principles and Perspectives," *Proceedings of Neuronal Networks in Applications (NN'98)*, Magdeburg, 1998

[49] C. S. Lindsey, Th. Lindblad, G. Sekniaidze, M. Minerskjold, S. Szekely and A. Eide, "Experience with the IBM ZISC Neural Network Chip," *Proceedings of 3rd Int. Workshop on Software Engineering, Artificial Intelligence, and Expert Systems, for High Energy and Nuclear Physics*, Pisa, Italy, April 3-8, 1995.

[50] P.C. Treleaven, "Neurocomputers," *International Journal of Neurocomputing*, 1, 4-31, 1989.

[51] A. F. Arif, S. Kuno, A. Iwata and Y. Yoshita, "A Neural Network Accelerator Using Matrix Memory with Broadcast Bus," *Proceedings of the IJCNN-93-Nagoya*, pp. 3050-3053, 1993.

[52] "HNC, High-Performance Parallel Computing," *SIMD Numerical Array Processor, DataSheet*, San Diego.

[53] J.N. H. Heemskerk, J. Hoekstra, J.M.J., Murre, L.H.J.K. Kemna and P.T.W. Hudson, "The BSP400: A Modular Neurocomputer," *Microprocessors and Microsystems*, 18, 2, pp. 67-78, 1994.

[54] H. Speckman, P. Thole and W. Rosentiel, "COKOS: A Coprocessor for Kohonen's Selforganizing Map," *Proceedings of the ICANN-93-Amsterdam,* London: Springer-Verlag, pp. 1040-1045, 1993.

[55] N. Morgan, J. Beck, P. Kohn, J. Bilmes, .E Allman and J. Beer, "The Ring Array Processor: A Multiprocessing Peripheral for Connectionist Applications," *Journal of Parallel and Distributed Computing*, 14, pp. 248-259, 1992.

[56] MD1220 *Data Sheet*, Micro Devices, 30 Skyline Dr., Lake Mary, Fl. 32746, March 1990.

[57] NLX420 *Data Sheet*, Neurologix, Inc., 800 Charcot Av., Suite 112, San Jose, CA, June 1992.

[58] N. Mauduit, M. Duranton and J. Gobert, "Lneuro 1.0: A Piece of Hardware LEGO for Building Neural Network Systems," *IEEE Transactions on Neural Networks*, 3, pp. 414-422, May 1992.

[59] A. Jahnke, U. Roth, and H. Klar, "A SIMD/Dataflow Architecture for a Neurocomputer for Spike-Processing Neural Networks (NESPINN)," *Proceedings of the 6th International Conference on Microelectronics for Neural Networks (Micro Neuro)*, pp. 232-237, 1996.

[60] M. Glover and W. T. Miller, "A massively-parallel SIMD processor for neural network and machine vision applications," *Proceedings NIPS-6*, ed. J.D. Cowan et al., Morgan Kaufmann Pub, pp. 843-849, 1993.

[61] H. Amin, K. M. Curtis and B. R. Hayes Gill, "Two-ring systolic array network for artificial neural networks," *Circuits, Devices and Systems, IEE Proceedings,* vol. 146, iss. 5, pp. 225-230, Oct. 1999.

[62] U. Ramacher, "SYNAPSE - A Neurocomputer that Synthesizes Neural Algorithms on a Parallel Systolic Engine," *Journal of Parallel and Distributed Computing*, 14, pp. 306-318, 1992.

[63] S. Tam, B. Gupta, H. Castro and M. Holler, "Learning on an Analog VLSI Neural Network Chip," *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*, 1990.

[64] Y. Maeda, H. Hirano and Y. Kanata, "AN Analog Neural Network Circuit with a Learning Rule via Simutaneous Perturbation," *Proceedings of the IJCNN-93-Nagoya*, pp. 853-856, 1993.

[65] H. Withagen, "Implementing Backpropagation with Analog Hardware," *Proceedings of the IEEE ICNN-94-Orlando* Florida, pp. 2015-2017, 1994.

[66] Synaptics, 2860 Zanker Rd., Suite 206, San Jose, Ca. 95134, USA.

[67] M. Jabri and B. Flower, "Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feed forward and Recurrent Multi-Layer Networks," *Neural Computation*, vol. 3, pp. 546-565, 1991.

[68] V. F. Koosh and R. M. Goodman, "VLSI neural network with digital weights and analog multipliers," *Proc. IEEE International Symposium on Circuits and Systems*, vol. III, pp. 233-236, May 2001.

[69] S. Churcher, D. J. Baxter, A. Hamilton, A. F. Murray and H. M. Reekie, "Generic Analog Neural Computation-the Epsilon Chip," *Advances in Neural Information Processing Systems: Proceedings of the 1992 Conference*, Denver, Colorado.

[70] Y. Hirai, "A 1,000-Neuron System with One Million 7-bit Physical Interconnections," *In Jordan, M.I., Kearns, M.J. and Solla, S.A. eds. Advances in Neural Information Processing Systems 10, A Bradford Book*, The MIT Press, Cambridge, Massachusetts, pp.705-711, 1998.

[71] B. Kim, C. Kim, S. Han, S. Kim and H. Park, "1.2-μM non-epi CMOS smart power IC with four H-bridge motor drivers for portable applications," *Circuits and Systems, ISCAS '96., 'Connecting the World'., 1996 IEEE International Symposium on* , Vol.1 pp: 633 – 636, 12-15 May 1996

[72] Datasheet of L293D by Texas Instrument Inc.

[73] Product GH12-1828Y of Jameco Electronics Inc.

[74] N.N. Bengiamin and M. Jacobsen, "Pulse-width modulated drive for high performance DC motors," *Industry Applications Society Annual Meeting, 1988., Conference Record of the 1988 IEEE*, Vol.1, pp:543 – 550, 2-7 Oct. 1988.

[75] www.handyboard.com

[76] M.-Y. Chow, G. Bilbro, and S. O. Yee, "Application of Learning Theory to a Single Phase Induction Motor Incipient Fault Detection Artificial Neural Network," *International Journal of Neural Systems*, vol. 2, no. 1&2, pp. 91-100, 1991.

[77] J. Van der Spiegel, P. Mueller, V. Agami, P. Aziz, D. Blackman, P. Chance, A. Choudhury, C. Donham, R. Etienne-Cummings, L. Jones, J. Kim, P. Kinget, M. Massa, W. von Koch, J. Xin, "A multichip analog neural network," *Proceedings of Technical Papers, International Symposium on VLSI Technology, Systems, and Applications*, pp. 64 – 68, 22-24 May 1991

[78] S. Tam, M. Holler, J. Brauch, A. Pine, A. Peterson, S. Anderson, S. Deiss," A reconfigurable multi-chip analog neural network: recognition and back-propagation training," *International Joint Conference on Neural Networks*, 1992., Vol. 2, pp. 625-630, 7-11 June 1992

[79] P. Hasler, L. Akers, "Implementation of analog neural networks," *Tenth Annual International Phoenix Conference on Computers and Communications*, pp. 32-38, 27-30 March 1991

[80] M. Ponca, G. Scarbata, "Implementation of Artificial Neurons Using Programmable Hardware," *Synopsys User Group Conference - SNUG Europe*, Munich, Germany, 12-13 March, 2001

[81] Y. Liao, "Neural Network Hardware: A Survey," *University of California – Davis*, 1995

CHAPTER III - Upper Bound on the Size of Neural Networks for Data Classification

Puxuan Dong[1]
IEEE Student Member
pdong@ncsu.edu

Griff Bilbro[1*]
IEEE Senior Member
glb@ncsu.edu

[1] Department of Electrical and Computer Engineering,
North Carolina State University, Raleigh NC 27695 USA

* Corresponding Author

This work has been submitted to IEEE Transactions on Neural Networks

Abstract

This paper proposes an upper bound on the complexity of feed forward neural networks for classifying data. We show that N data points drawn from two classes can be correctly labeled by a network with a single hidden layer network containing at most N/2+1 hidden neurons (when N is even or (N-1)/2+1 hidden neurons when N is odd) with hyperbolic tangent activation functions.

Keywords: upper bound, classification, feed forward neural network

I. Introduction

Proper size of a neural network (NN) allows it to learn a task efficiently and to predict future output appropriately [1] and has been considered by several researchers. Lecun proposed optimal brain damage [2]; Hassibi presented optimal brain surgeon [3]; Mozer "trimmed" the excessive neuron and weights from the neural network via relevance assessment, which he called "Skeletonization" [4]. The initial maximum necessary network size is desired for pruning techniques since it makes the techniques efficient. Besides the pruning techniques mentioned above, there are also constructive methods such as cascade-correlation (CC) introduced by Falman [5]. Thivierge, Rivest and Shultz even found a way to prune constructive neural networks which is a "dual phase" technique [6].

Apart from these growing and pruning methods, the approximation capabilities of neural networks have been intensively analyzed [7-14] as another kind of effort to find the ideal topology of neural networks for certain problems. Analysis of approximation

capability is also important since it helps to choose the size of neural networks. Excessively large neural networks are computationally demanding to simulate and may not generalize appropriately; and on the other hand, neural networks of inadequate complexity may fail to learn significant features of a data set. A lower bound on the size of the neural network was recently reported by Gao and Ji [15].

In this paper we present an upper bound for the size of the neural network solving classification problems. We show that a conventional three-layer feed forward neural network with hyperbolic tangent activating functions containing K+1 neurons in its hidden layer can learn any arbitrary assignment of 2K vectors to two classes.

We apply the proposed theory to the two spirals benchmark problem [17]. We find that a neural network with the proposed number of hidden neurons learns the two-spiral problem perfectly within 500 training epochs. This is about 70% less training effort than for the cascade-correlation algorithm proposed by Fahlman and Lebiere [5]. We also find that networks with 15% fewer neurons than the proposed upper bound cannot learn the data without error.

The paper is organized as follows: section II derives the upper bound; section III applies the theory to a benchmark classification problem - the two-spiral problem and section IV offers some concluding remarks.

II. Theory

Gao and Ji's research result shows that hidden neuron number required in a single-hidden-layer neural network to solve the n-class problem is independent of the dimension

of the input data vectors [15]. We therefore restrict our attention to the one-dimensional case to analyze the upper bound on the number of hidden neurons of a single-hidden-layer feed forward neural network solve two-class classification problems of N data points.

In one dimension, the alternate label problem [16] requires the most decision points since when elements from different classes of data alternate with each other, each pair of adjacent points needs one decision point between them. The N-point alternate label problem needs N-1 decision points and provides an upper bound on the number of hidden neurons required for classification.

For simplicity, we choose $\{+1,-1\}$ as the label set for the alternate label problem. This is equivalent to using the algebraic sign of the neural network output as the label that it computes for a particular input value. This can also be described as choosing a threshold value of zero for the real-valued function computed by the neural network.

We assume that the number $N = 2K+1$ of data points is odd, that they are indexed by size so that $x_1 < x_2 < x_3 \cdots < x_{2K+1}$, that the odd points are labeled $+1$, and that the even points are labeled -1. We define $H \equiv \frac{1}{2}\max_{n=1}^{N-1}\left(x_{n+1} - x_n\right)$ to be half the maximum interval between adjacent data points and $h \equiv \frac{1}{2}\min_{n=1}^{N-1}\left(x_{n+1} - x_n\right)$ to be half the minimum interval between adjacent data points. Consider the unit sigmoid

$u(x) = \frac{1}{2}\left(1 + \tanh(\beta x)\right)$ shown in Figure 3.1 where β is chosen to solve $\tanh(\beta x) = 1 - 2\delta$ for $\delta = \frac{h}{4KH}$ so that $u \geq 1 - \delta$ for $x \geq h$ and $u \leq \delta$ for $x \leq -h$.

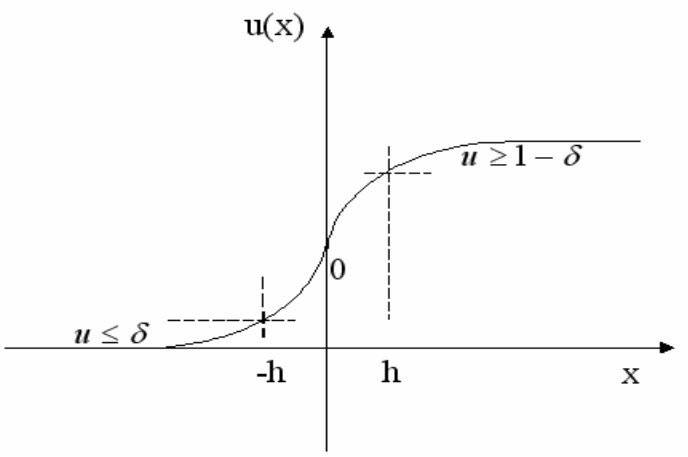


Figure 3.1. The unit sigmoid function

It is convenient to define a second set of N points as $y_n \equiv x_n - h$ for $1 \leq k \leq N$ as indicated in Figure 3.2. It follows from these definitions for appropriate n that $x_n - y_n = h$, and $h \leq y_{n+1} - x_n \leq H$, and $2h \leq x_{n+1} - x_n = y_{n+1} - y_n \leq 2H$.

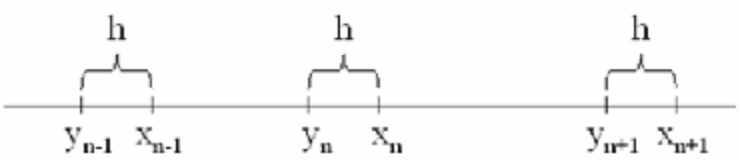


Figure 3.2. Example of data points

We define the function $u_0(x) = \frac{1}{\alpha}\tanh(\alpha x)$ to be approximately linear in the interval $y_1 \leq x \leq x_N$ by choosing α to satisfy $x_N - y_1 - \frac{1}{\alpha}\tanh\left(\alpha(x_N - y_1)\right) = 4H\delta = \frac{h}{K}$. The

relationship between $(x_N - y_1)$ and $u_0(x) = \frac{1}{\alpha}\tanh(\alpha x)$ is shown in Figure 3.3. It follows that $x - y_1 - 4H\delta \le u_0(x - y_1) \le x - y_1$.

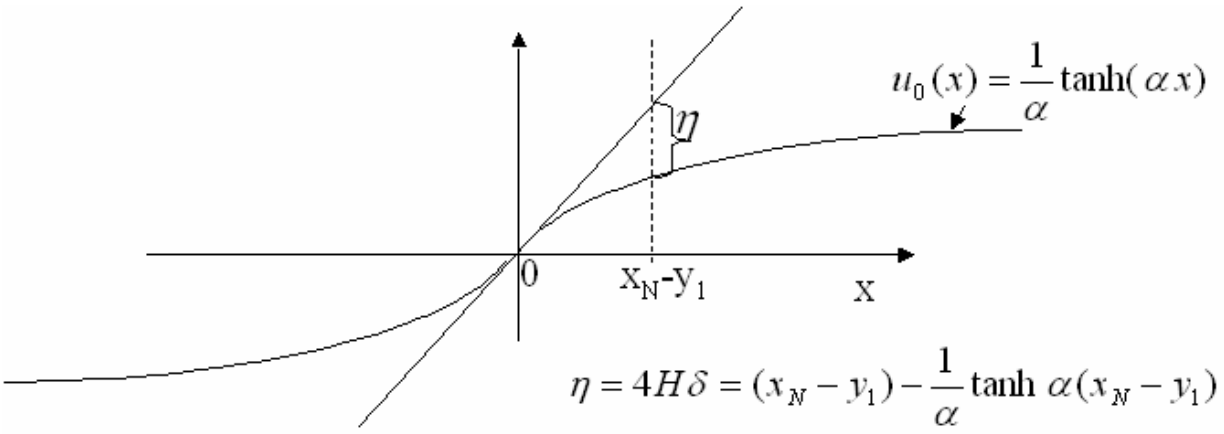


Figure 3.3: $u_0(x) = \frac{1}{\alpha}\tanh(\alpha x)$ and the corresponding straight line showing the error at $(x_N - y_1)$ as $4H\delta$

We now show that the function $f(x) = u_0(x - y_1) - \sum_{k=1}^{K}\left(y_{2k+1} - y_{2k-1}\right)u\left(x - y_{2k}\right)$ correctly labels any point x_k as the algebraic sign of $f(x_k)$. There are two cases. First we show that $f(x_{2n}) \le 0$ so that the evenly indexed points are labeled -1. Second we will show that $f(x_{2n+1}) \ge 0$ correctly labels every odd data point as $+1$. An illustrative cartoon of f for 5 points using 3 hyperbolic tangents are shown in Figure 3.4.

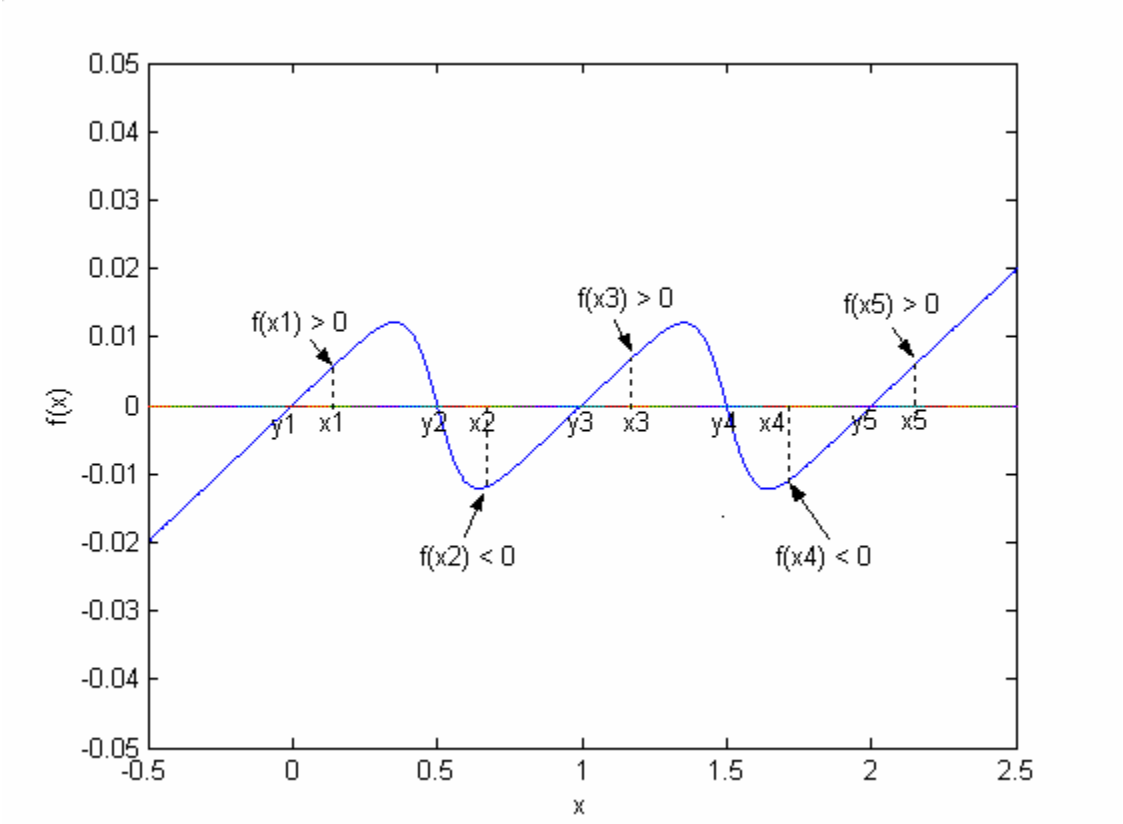


Figure 3.4. Three hyperbolic tangent functions classifying five data points

In the expression $f(x_{2n}) = u_0(x_{2n} - y_1) - \sum_{k=1}^{K} (y_{2k+1} - y_{2k-1})\, u(x_{2n} - y_{2k})$, we decompose the sum into two parts. In the first part over $1 \le k \le n$ we have $x_{2n} - y_{2k} \ge h$ so that $u(x_{2n} - y_{2k}) \ge 1 - \delta$. In the second part over $n+1 \le k \le K$ we have $x_{2n} - y_{2k} \le -h$ and we assert only that $u > 0$, so that $f(x_{2n}) \le x_{2n} - y_1 - \sum_{k=1}^{n} (y_{2k+1} - y_{2k-1}) + \sum_{k=1}^{n} (y_{2k+1} - y_{2k-1})\delta$. The first sum telescopes to $y_{2n+1} - y_1$ and combines with the first two terms to yield $x_{2n} - y_{2n+1} \le -h$. The second term is less than $4n\delta H$, so that $f(x_{2n}) \le -h + 4nH\delta \le -h + 4KH\delta = -h + h$ which implies that $f(x_{2n}) \le 0$ as desired.

For the remaining data points, we have $f(x_{2n+1}) = u_0(x_{2n+1} - y_1) - \sum_{k=1}^{K} (y_{2k+1} - y_{2k-1})\, u(x_{2n+1} - y_{2k})$. Again we decompose the sum into two parts. For $n+1 \le k \le K$, we have $x_{2n+1} - y_{2k} \le -h$ so that $u(x_{2n+1} - y_{2k}) \le \delta$ and we have $u < 1$ for all k including $1 \le k \le n$. In addition $u_0(x_{2n+1} - y_1) \ge x_{2n+1} - y_1 - 4H\delta$ by construction. Consequently we can write $f(x_{2n+1}) \ge x_{2n+1} - y_1 - 4H\delta - \sum_{k=1}^{n} (y_{2k+1} - y_{2k-1}) - \sum_{k=n+1}^{K} (y_{2k+1} - y_{2k-1})\delta$, where the first sum telescopes as before and combines with the first two terms. The second sum is bounded above by $\sum_{k=n+1}^{K} 4H\delta$, so that $f(x_{2n+1}) \ge x_{2n+1} - y_{2n+1} - 4H\delta - 4H(K-n-1)\delta = h - 4H(K-n)\delta$. But $K - n \le K$ so that this implies that $f(x_{2n+1}) \ge h - 4HK\delta = h - h$ and we can conclude that $f(x_{2n+1}) \ge 0$, as desired.

Therefore, 2*K*+1 points can be classified correctly by *K*+1 neurons in the hidden layer of a feed forward neural network. In other words, $(N-1)/2+1$ hidden neurons are sufficient to classify *N* data points of two classes.

The case of an even number $N = 2K$ of data points, the same function can be used and the same proof is valid if only a fictitious point is introduced at $x_{2K+1} = 2h + x_{2K}$ with an appropriate label. Consequently the proposed network output function *f* can correctly treat the alternate label problem regardless of the parity of N. Therefore, 2*K* points can be classified correctly by *K*+1 neurons in the hidden layer of a feed forward neural

network. In other words, $N/2+1$ hidden neurons are sufficient to classify N data points of two classes.

Since f is contained in the set of functions naturally realized by conventional three-layer feed forward neural networks, we have shown that a single-hidden-layer network with at most (N/2)+1 hidden neurons is sufficient to classify N (N is even) data points of 2 classes with zero error. The upper bound reduces to (N-1)/2+1 when N is odd. Since this problem is the worst case of any one-dimensional problem of size N and since the number of hidden layer neurons does not depend on the dimension of the inputs, we have shown that any 2-labeling problem of size N can be learned without error by a conventional feed forward neural network with at most $N/2+1$ hidden neurons.

III. Experimental Results

The application of the theory is demonstrated with the classification benchmark problem – the two spirals problem [17]. The two spirals problem has been used to test all kinds of neural classifiers [18-20]. The two spirals problem consists of 194 X-Y training points forming two interlocking classes of which one class has the output of 1 and the other has the output of -1. Each class has 97 points circling around the origin for three and half times as shown in Figure 3.5.

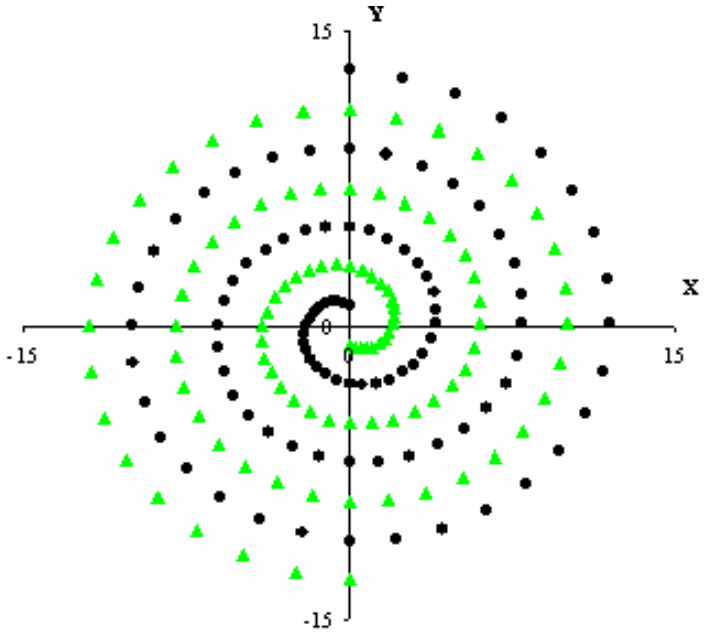


Figure 3.5. Data mapping of the two spirals problem.

To solve the two spiral problem, Yu and Tang proposed to use adaptive activation functions in the neural network [21]; Xiong, et al. present the branch control network as a supplement for the exiting neural network to solve the problem [22]; Wilamowski and Jaeger recommend to use input pattern transformation method to obtain a solution [23]; Jia and Chua present the method of input data representation [18] and Waterhouse and Robinson adopt hierarchical mixture of experts as another alternative [24]. The classification results of all above mentioned method are listed in Table I in terms of network structure, training epochs, training MSE error, number of training data points and number of testing data points. It appears that only the adaptive activation function and branch control network method classify the 194 data points with 100 percent accuracy as shown in Figure 3.6. However, none of these two methods use traditional feed forward neural network – adaptive activation function method indicates the activation functions of the neurons need to be changed during training which increases

the computational complexity and branch control network add additional structure to the existing neural network which adds complexity to training as well.

Table 3.1. Comparison of different methods classifying the two-spiral problem.

	Network Structure	Training Epoch	Training Set Data Points	Testing Set Data Points
Cascade-Correlation, S.E.Falman, C.Lebiere	>11 hidden layers	1700	194	Not specified
Adaptive Activation Function C.C. Yu, Y.C. Tang, and B.D. Liu.	2-12-1	Not specified	200	Not specified
Branch Control Network Q.Xiong, K. Hirasawa, J, Hu and J. Murata	2-100-1 with extra branch network	>1,000,000	194	2601
Input Pattern Transformation B.M. Wilamowski and R.C. Jaeger	2-8-1 with extra transformation layer	Not specified	Not specified	Not specified
Input Data Representation (Weighted Binary) J. Jia and H.C. Chua	18-40-2	<2,000	Not specified	Not specified
Hierarchical Mixture of Experts S.R. Waterhouse and A.J. Robinson	10 layer tree	315	Not specified	Not specified
The Upper Bound Theory P. Dong and G. Bilbro	2-98-1	500	194	14,400

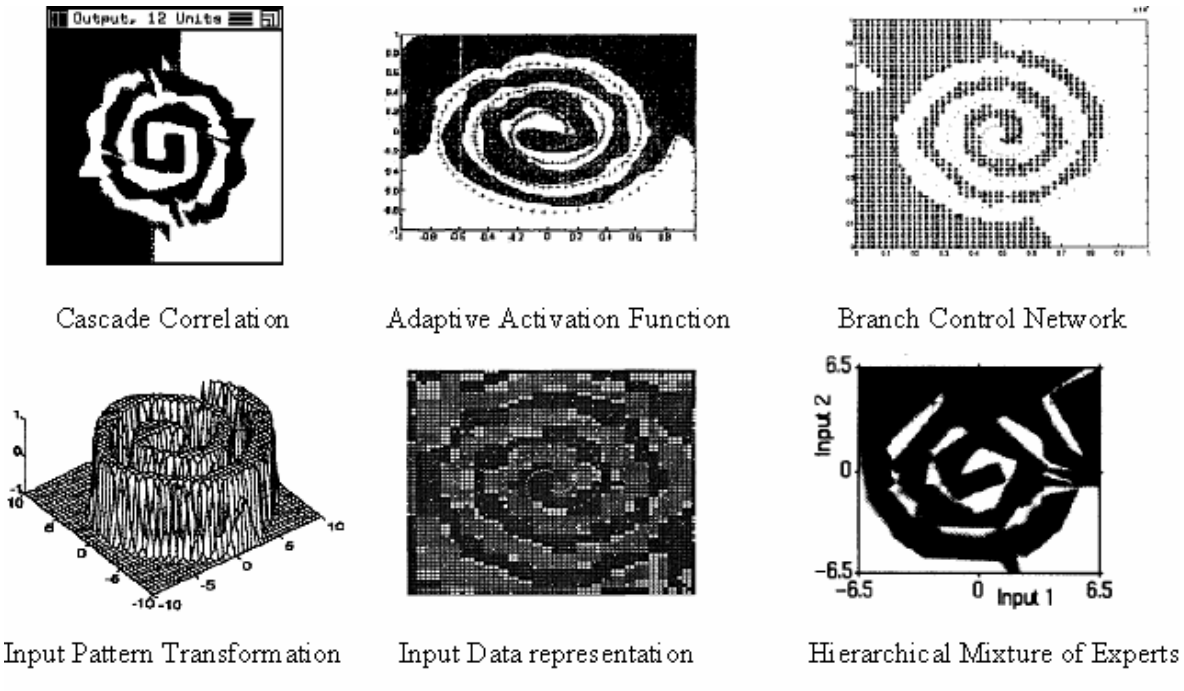
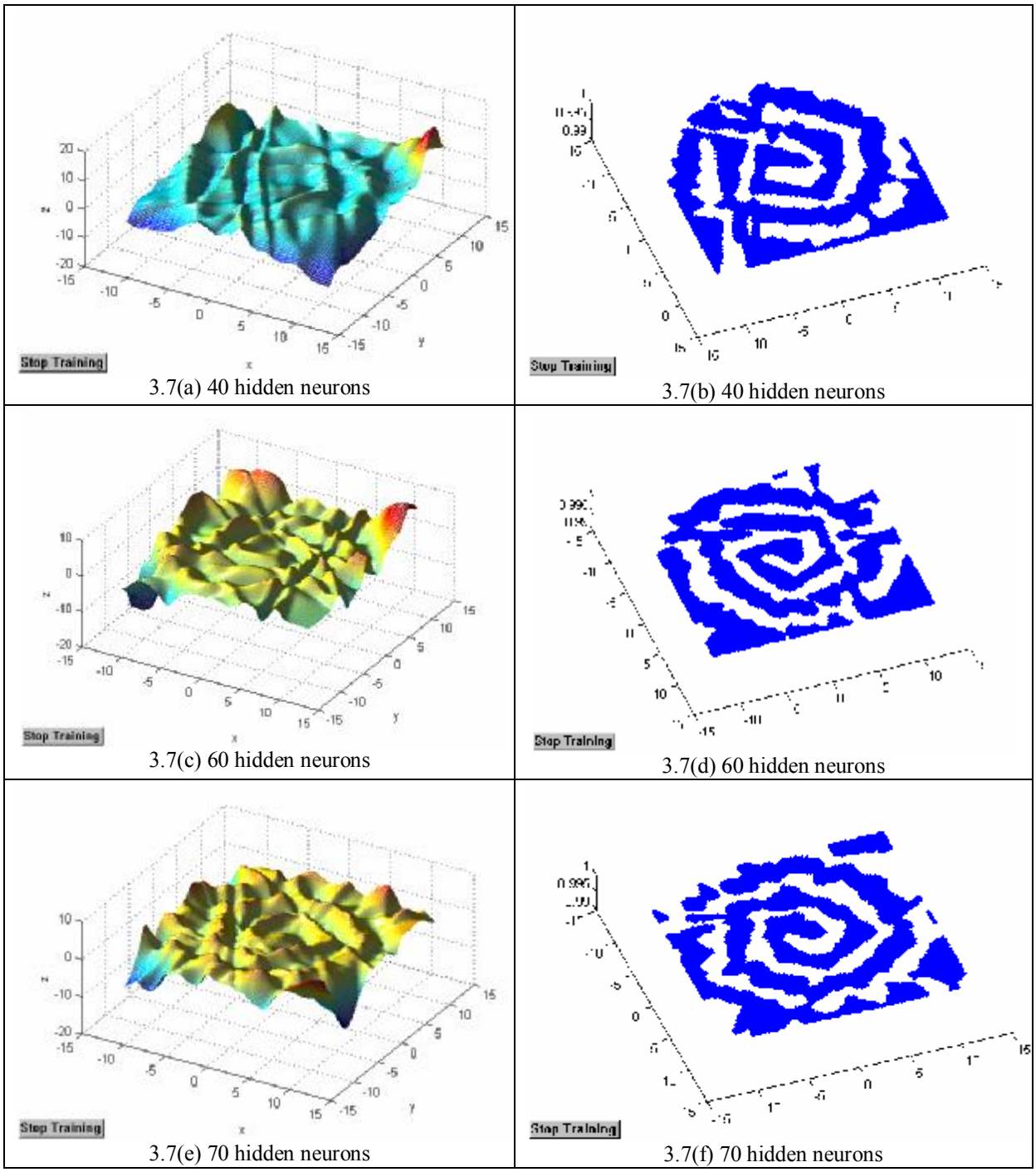


Figure 3.6. Generalization performance of two-spiral problem by different methods.

Based on our theory, 98 neurons are sufficient to separate the data of two classes with zero error. We train and test different single hidden layer neural networks with the following number of hidden neuron numbers: 40, 60, 70, 80, 90, 98 and 145. The trained the network is tested thoroughly with 14400 data points as shown in Figure 3.7a-3.7n.



3.7(a) 40 hidden neurons

3.7(b) 40 hidden neurons

3.7(c) 60 hidden neurons

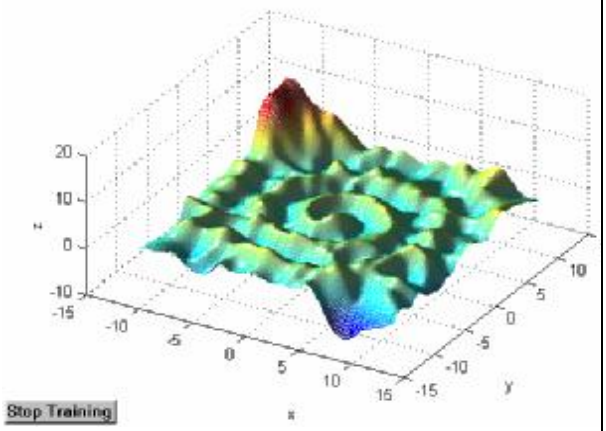
3.7(d) 60 hidden neurons

3.7(e) 70 hidden neurons

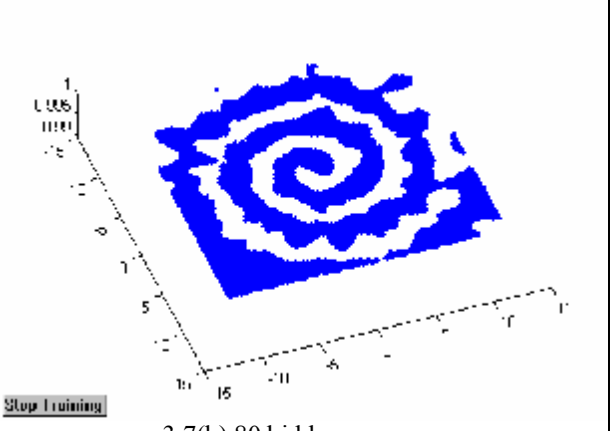
3.7(f) 70 hidden neurons

Figure 3.7. Testing results of two-spirals problem trained with the 3-layered feed forward neural networks.

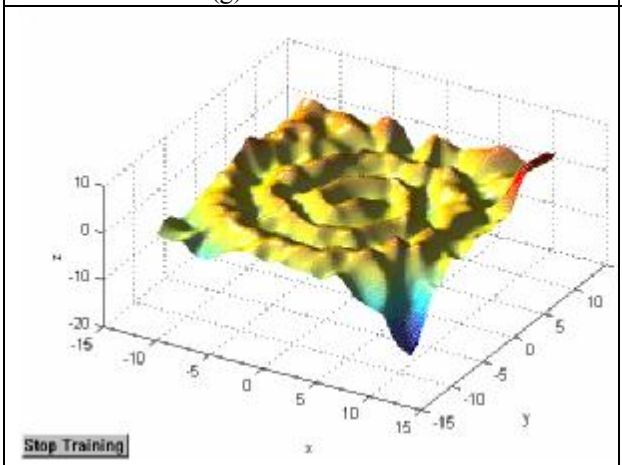
(Simulated in MATLAB/SIMULINK from the MathWorks, Natick, MA, USA)



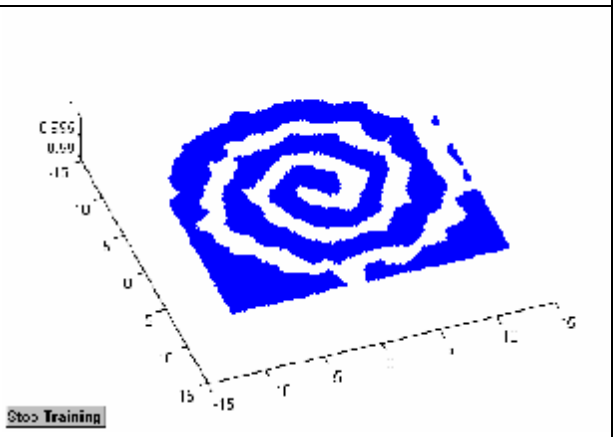
3.7(g) 80 hidden neurons



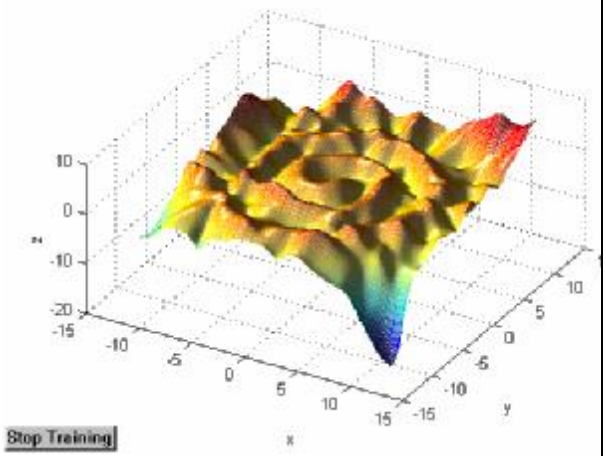
3.7(h) 80 hidden neurons



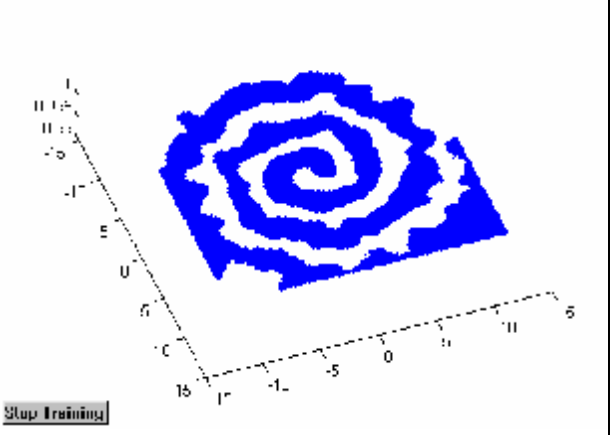
3.7(i) 90 hidden neurons



3.7(j) 90 hidden neurons

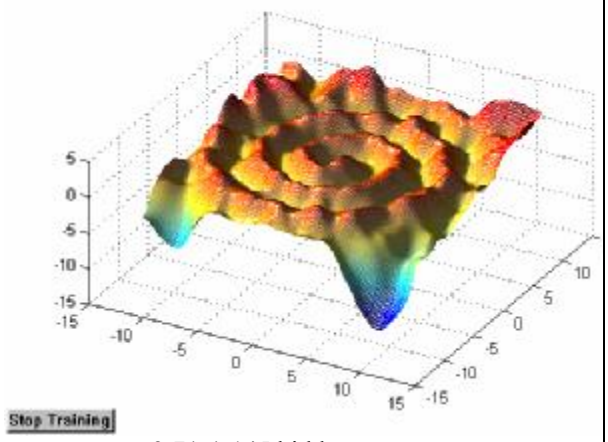


3.7(k) 98 hidden neurons

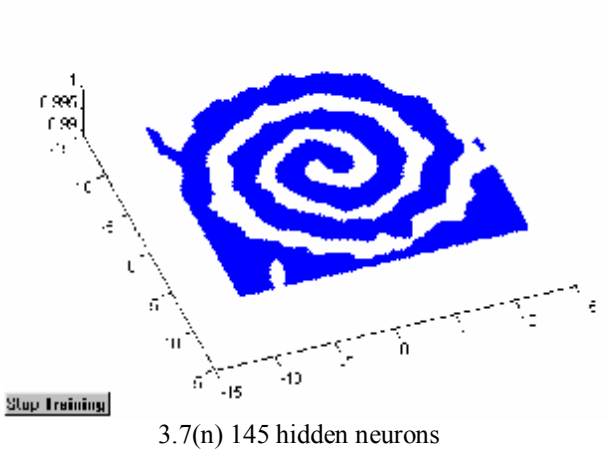


3.7(l) 98 hidden neurons

Figure 3.7 – continued.



3.7(m) 145 hidden neurons



3.7(n) 145 hidden neurons

Figure 3.7 – continued.

As shown in Figure 3.7, the testing results indicate the network classifies the problem better with the increase of the number of the hidden neurons. The network classifies the two groups of data completely when the neuron number reaches 90 in the hidden layer. With 98 and 145 neurons in the hidden layer, the network still classifies the two groups of data points successfully. This means the minimum required neuron numbers is between 80 - 90 which is fairly close to our calculated "data-independent" upper bound. This result would be closer to our upper bound if the problem could somehow be made more difficult.

For the 2-98-1 structure, the classification error on the training set of 194 points drops to zero after 500 back propagation training epochs. The trained network classifies the two groups of data with 100 percent accuracy. The number of epochs is 70.59% less than the result (1700 epochs) obtained by the cascade-correlation algorithm proposed by Fahlman and Lebiere [5].

IV. Conclusion

An upper bound has been presented for the number of neurons in the hidden layer of a 3-layered neural network that can correctly label any N data points of 2 classes. At most N/2+1 hidden neurons is sufficient to classify N such data points of 2 classes with zero error. The hidden neurons have hyperbolic tangent activation functions. The data points are allowed to be *m* dimensional ($m > 1$). The application of the theory is demonstrated with a classification benchmark problem from the literature.

V. References

[1] Z. Qin and Z. Mao, "A new algorithm for neural network architecture study," *presented at Intelligent Control and Automation, 2000. Proceedings of the 3rd World Congress on*, 2000.

[2] Y. LeCun, J. Denker & S. Solla, "Optimal Brain Damage," *In D. Touretzky (ed.), Advances in Neural Information Processing Systems, Morgan Kaufmann*, pp. 598-605. San Mateo, CA, 1990.

[3] B. Hassibi, D. G. Stork, "Second order derivatives for network pruning: Optimal Brain Surgeon," *Proceedings of Neural Information Processing Systems*, pp.164-171, 1993.

[4] M. C. Mozer & P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," *Advances in Neural Information Processing Systems*, pp. 107-115, 1998.

[5] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," *In Touretzky, D. S., editor, Advances in Neural Information Processing Systems*, pp. 524-532, San Mateo, 1990.

[6] J. P. Thivierge, F. Rivest, T.R. Shultz, "A dual-phase technique for pruning constructive networks," *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 559-564, 2003.

[7] E. Baum, "On the capabilities of multilayer perceptrons," *Journal of Complexity*, vol. 4, pp. 193–215, 1988.

[8] K. Hornik, "Approximation capabilities of multilayer feed forward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[9] I. Ciuca, "On the approximation capability of neural networks using bell-shaped and sigmoidal functions," *presented at IEEE International Conference on Systems, Man, and Cybernetics*, 1998.

[10] W. Eppler and H. N. Beck, "Piecewise linear networks (PLN) for function approximation," *presented at Neural Networks, 1999. IJCNN '99. International Joint Conference on*, 1999.

[11] Z. Zhanga, X. Mab and Y. Yang, "Bounds on the number of hidden neurons in three-layer binary neural networks," *Neural Networks* 16, 2003.

[12] K. H. Schindler, M. Sanguineti, "Bounds on the complexity of neural-network models and comparison with linear methods," *International Journal of Adaptive Control and Signal Processing*. Vol, 17, pp. 179-194, 2003.

[13] G. B. Huang, "Learning Capability and Storage Capacity of Two-Hidden-Layer Feed forward Networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, Mar. 2003.

[14] G. B. Huang and H. A. Babri, "Upper Bounds on the Number of Hidden Neurons in Feed forward Networks with Arbitrary Bounded Nonlinear Activation Functions," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 224-229, 1998.

[15] D. Gao and Y. Ji, "Classification methodologies of multilayer perceptrons with sigmoid activation functions, " *Pattern Recognition* 38 pp. 1469 – 1482, 2005

[16] S. Ridella, S. Rovetta, and R. Zunino, "Circular back propagation networks for classification," *IEEE Transactions on Neural Networks*, vol. 8, pp. 84-97, 1997.

[17] K. K. Lang and M. K. Witbrock, "Learning to tell two spirals apart," *in Proceedings of Connectionist Models Summer School*, San Mateo, CA: Morgan Kaufman, pp. 52-61, 1998.

[18] J. Jia and H. C. Chua, "Solving two-spiral problem through input data representation," *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 1, pp. 132 – 135, 27th Nov.-1st Dec. 1995.

[19] S. Ridella, S. Rovetta and R. Zunino, "Representation and generalization properties of class-entropy networks," *IEEE Transactions on Neural Networks*, vol. 10, issue 1, pp. 31-47, Jan 1999.

[20] H. C. Fu, Y. L. Lee, C. C. Chiang and H. T. Pao, "Divide-and-conquer learning and modular perceptron networks," *IEEE Transactions on Neural Networks,* vol 12, issue 2, pp. 250 – 263, March 2001.

[21] C.-C. Yu, Y.-C. Tang, and B.-D. Liu, "An adaptive activation function for multilayer feed forward neural networks," *presented at TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, 2002.

[22] Q. Xiong, K. Hirasawa, J. Hu, and J. Murata, "Comparative study between functions distributed network and ordinary neural network," *presented at Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, 2001.

[23] B. M. Wilamowski and R. C. Jaeger, "Implementation of RBF type networks by MLP networks," *presented at Neural Networks, 1996., IEEE International Conference on*, 1996.

[24] S. R. Waterhouse and A. J. Robinson, "Classification using hierarchical mixtures of experts," *presented at Neural Networks for Signal Processing [1994] IV. Proceedings of the 1994 IEEE Workshop*, 1994.

[25] www.mathworld.com

CHAPTER IV - Implementation of Artificial Neural Network for Real Time Applications Using Field Programmable Analog Arrays

Puxuan Dong[1*]
IEEE Student Member
pdong@ncsu.edu

Griff Bilbro[2]
IEEE Senior Member
glb@ncsu.edu

Mo-Yuen Chow[1]
IEEE Senior Member
chow@ncsu.edu

[1] Advanced Diagnosis Automation & Control Lab,
Department of Electrical and Computer Engineering,
North Carolina State University, Raleigh NC 27695 USA
Phone: +1(919)515-5405

[2] Department of Electrical and Computer Engineering,
North Carolina State University, Raleigh NC 27695 USA

* Corresponding Author

This chapter is to be submitted to IEEE Transactions on Computational Intelligence. A short version of this work is accepted for presentation in WCCI2006 (International Joint Conference on Neural Netwoks, Vancouver, BC, Canada, July 16-21).

Abstract

This paper presents a method of realizing artificial neural networks (ANNs) hardware implementation using field programmable analog arrays (FPAAs). A simplified realization for neurons with piecewise linear activation functions is used to reduce the complexity of the neural network architecture. Several different feedforward neural networks are implemented using single-chip and multi-chip FPAAs. Anadigm's commercially available AN221E04 FPAA chips are adopted as the platform for simulation and experiments. The FPAA based multi-chip ANN classifies two groups of data with zero error at a speed of 6.0 Million Connections Per Second (MCPS). The result is more than 1400 times faster than comparable software implementation. The ANN architecture is also expandable to perform more complicated tasks by incorporating more FPAA chips into the implementation. The programmability of the FPAA makes analog rapid prototyping possible.

Keywords: field programmable analog arrays, neural network hardware, rapid prototyping

I. Introduction

Artificial neural networks (ANNs) have been playing an increasingly important role in areas such as robotics [1], process control [2-3], and motor fault detection [4-6]. Both software and hardware based approaches have been used for implementing ANNs. In general, software instructions executed serially cannot take advantage of the inherent parallelism of ANN architectures. Hardware implementations of neural networks promise higher speed operation when they can exploit this massive parallelism. Different

hardware implements of neural network have been reported [7-25]. Other than the FPGA based approaches [10, 11, 18, 24], most of the hardware implementations provide no programmability. Reconfigurability of an ANN is desirable since many ANN applications, e.g., robots performing different tasks in different environments may benefit from different neural network topologies (e.g., different number of hidden nodes). The best choices for neural network implementations that achieve both high speed and rapid prototyping appear to be programmable hardware approaches like field programmable gate arrays (FPGAs) and field programmable analog arrays (FPAAs). Compared to digital hardware, FPAAs have the advantage of interacting directly with the real world because they receive, process, and transmit signals totally in the analog domain (without the need to do A/D, D/A conversions) and are suitable for real time applications. As reported in [26] on controlling a path-tracking unmanned ground vehicle, an FPAA can easily outperform the digital hardware by processing the signal 8,000 times faster. Other FPAA applications, including a voltage-to-frequency converter and a Hodgkin-Huxley neuron simulator, have been reported [27-28].

Section II of this paper proposes a simple realization of layered neural networks appropriate for FPAAs. Section III applies the neural network architecture simplification method to a single-chip FPAA based neural network to realize the XOR gate with can be converted to other 3 logic gates with very little change of the network architecture. Section IV applies the neural network architecture simplification method to a multi-chip FPAA based neural network to classify the elements of a data set containing two groups of data. Section V analyzes the speed performance of the FPAA implementing the ANN by comparing it to software implementation. Section VI gives some concluding remarks.

II. Neural Network Architecture Simplification in FPAA

A. The Piecewise Linear Activation Function

In the ANN, the output of a neuron is computed by applying its activation function to a weighted sum of its inputs. Some activation functions such as hyperbolic tangent and sigmoid are expensive for digital hardware implementation. To reduce the cost for implementation, the piecewise linear activation function has been used to approximate sigmoid activation function [29]. We chose the Piecewise Linear (PL) activation function for the neurons in the hidden layer of our neural network architecture because it is naturally suited for applying FPAA hardware to the problem of interest (to be described in later sections).

A neural network must be trained to reflect or to generalize a desired relationship between inputs and outputs. During the back propagation training process in a neural network, the error signal at the output of the neuron *j* at iteration *n* (i.e., presentation of the n^{th} training example) is defined by

$$e_j(n) = d_j(n) - y_j(n), \tag{1}$$

where $d_j(n)$ is the desired response of neuron *j* and is used to compute $e_j(n)$, $y_j(n)$ is to the function signal appearing at the output of neuron *j* at iteration *n*. Let $\varphi_j(\bullet)$ be the activation function; then the synaptic weight $\Delta w_{ji}(n)$ change is:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n), \tag{2}$$

where

$$\delta_j(n) = e_j(n)\varphi_j'\left(v_j(n)\right), \tag{3}$$

is called the local gradient and η is the learning rate. In equation (3),

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n), \tag{4}$$

and w_{ji} denotes the synaptic weight connecting the output neuron *i* (there are *m* inputs) to the input of neuron *j* at iteration *n*. The PL activation function is given by

$$\varphi_j(x) = \begin{cases} w_+, & \mathbf{w}_l^T \mathbf{x} + w_0 \geq w_+ \\ \mathbf{w}_l^T \mathbf{x} + w_0, & w_- < \mathbf{w}_l^T \mathbf{x} + w_0 < w_+ \\ w_-, & \mathbf{w}_l^T \mathbf{x} + w_0 \leq w_- \end{cases}, \tag{5}$$

where $\mathbf{x}^T = [x_1, x_2, ...x_d] \in R^d$ is the input vector and $\mathbf{w}^T = [w_0, w_1, ...w_d, w_+, w_-] \in R^{d+3}$, with $w_+ = +1$ and $w_- = -1$, is the parameter vector that characterizes the node function. Figure 4.1 shows the 3D view of input-output relationship of a neuron of 2 inputs with piecewise linear activation function.

Although the PL activation function is less popular than the hyperbolic tangent activation function, the piecewise nature has attractive features such as ease of implementation and amenability to VLSI implementation [30-31]. It is also simpler to find $\varphi_j'(\bullet)$ in equation (5) since it requires only addition, multiplication and comparison operations in contrast to the trigonometric function that must be evaluated for the hyperbolic tangent function.

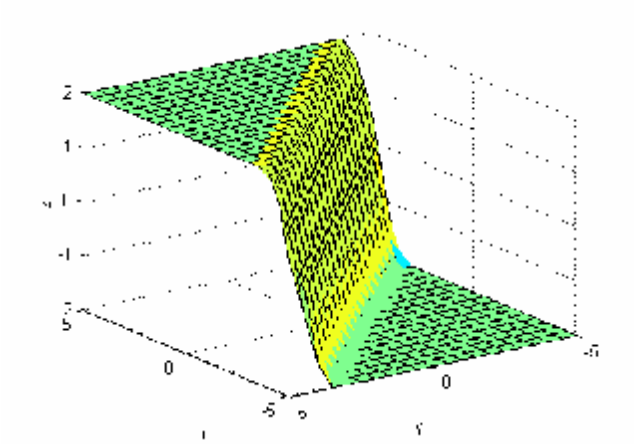


Figure 4.1. The piecewise linear (PL) activation function for $\mathbf{w}^T = [1,1]$ in two dimensions.

B. Implementing the PL Function on FPAA

This section develops a realization of the standard PL activation function that uses two gain amplifier functional blocks. A standard PL function has the following form:

$$PL(x) = \begin{cases} -1 & x < -1 \\ x & -1 \le x < +1 \\ +1 & x \ge +1 \end{cases}. \tag{6}$$

In a FPAA circuit which saturates symmetrically at V_+ and V_-, where $V_+ = V_0 > 0$, $V_- = -V_0 < 0$, a standard PL activation function can be obtained with two cascade gain stages G_1 and G_2 if $V_0 > 1$, where $G_1 = \frac{V_+ - V_-}{2} = V_0$ and $G_2 = \frac{2}{V_+ - V_-} = \frac{1}{V_0}$ (which will be explained in the following paragraphs). Note that the product of G_1 and G_2 is unity and $G_1 > 1 > G_2$.

Since the circuit saturates at V_+ and V_-, the relationship between input voltage x and output voltage $F_1(x)$ of a "through" circuit is:

$$F_1(x) = \begin{cases} -V_0 & x < -V_0 \\ x & -V_0 \le x < V_0 \\ V_0 & x \ge V_0 \end{cases}. \tag{7}$$

A gain stage G_1 after $F_1(x)$ establishes the following relationship between the new output $F_2(x)$ and x:

$$F_2(x) = F_1(x) \times G_1 = \begin{cases} -V_0 & x < -1 \\ G_1 \times x & -1 \le x < 1 \\ V_0 & x \ge 1 \end{cases}. \tag{8}$$

Adding another gain stage G_2 after $F_2(x)$ gives the following relationship between $F_3(x)$ and x:

$$F_3(x) = F_2(x) \times G_2 = \begin{cases} -1 & x < -1 \\ x & -1 \le x < +1 \\ +1 & x \ge +1 \end{cases}. \tag{9}$$

Thus the standard piecewise linear activation function is obtained by inserting these two particular gain stages between the input and the output of a through circuit. Figure 4.2 shows the three functions (with using $V_0 = 2.5$).

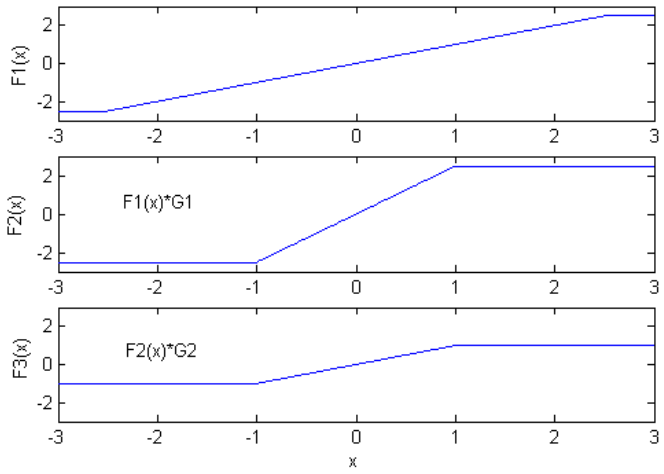


Figure 4.2. Obtaining the PL function using two gain stages.

C. Merging the Gain Stages of Cascade Blocks on the FPAA

As shown in Figures 3 and 4, the neural network can be simplified further by merging the two gain blocks G_1 and G_2 into the input and output weights of the neurons. G_1 and G_2 form the standard piecewise linear transfer function for neuron j. The neural network architecture in Figure 4.3 can be simplified by multiplying every weight of neuron j by G_1 and multiplying w_{k1} by G_2 as shown in Figure 4.4. As a result, addition and multiplication are the only two operations required for a neural network implementation on the FPAA. The addition operation is performed by inverting sum amplifier blocks. The weights that a neuron uses to compute the weighted sum of its inputs are realized as the gain parameters of the inverting sum amplifiers on the FPAA. These weights are obtained from an offline training procedure using MATLAB/SIMULINK software to accurately simulate the network topology and to optimize the weights. The optimal weights are downloaded to the Anadigm FPAA chips for the corresponding real-time operation, such as controlling a mobile robot or, in this

paper, classifying data points.

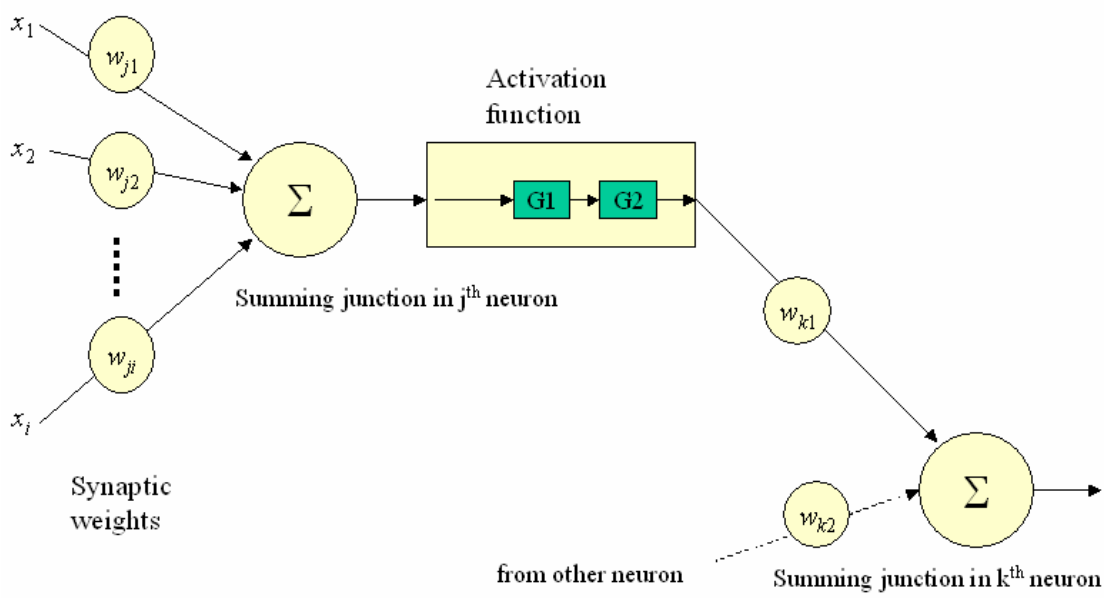


Figure 4.3. Neural network architecture with unmerged gain blocks.

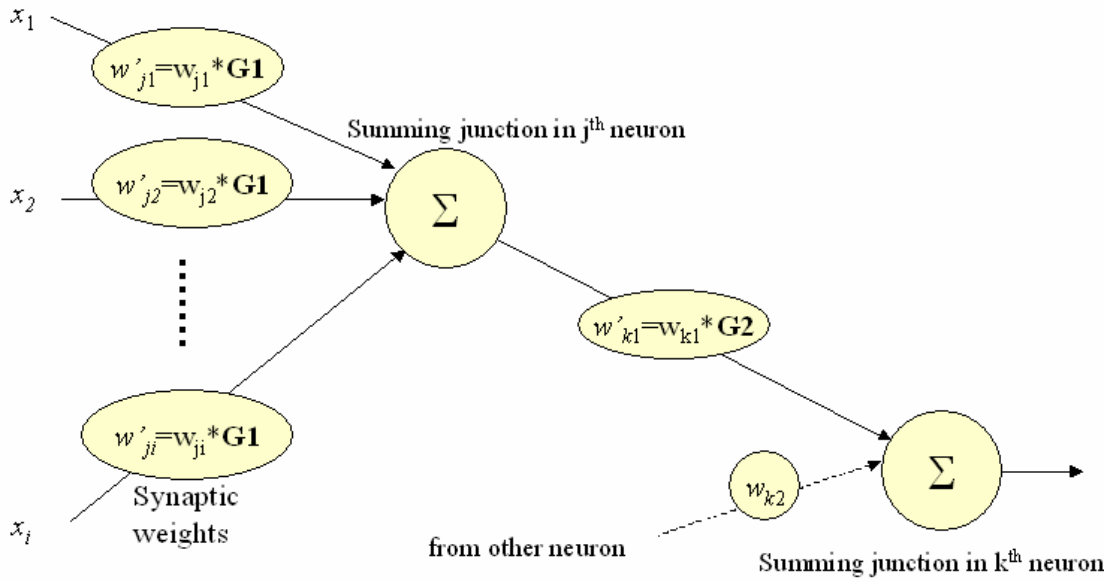


Figure 4.4. Neural network architecture with merged gain blocks.

Note that merging G_1 into w_{ji} will not change the input to G_2 in Figure 4.4. In the

meantime, merging G_2 into w_{k1} will not change the input to the summing junction of k^{th} neuron. The merging procedure for G_1 implemented in FPAA is depicted in Figure 4.5. The circuits shown reflexes the FPAA circuits except that resistors are replaced by equivalent switched capacitors and signals are differential inside the actual FPAA. The top circuit is the one before merging and the bottom one is after merging.

This section explains why voltage saturation at the upper and lower limits of circuit does not invalidate our merging simplification. In the circuits shown in Figure 4.5, G_1 is equal to $R5/R4$. In all cases the output of the Op Amps saturates at $\pm v_0$ (the Op Amps are assumed to be ideal).

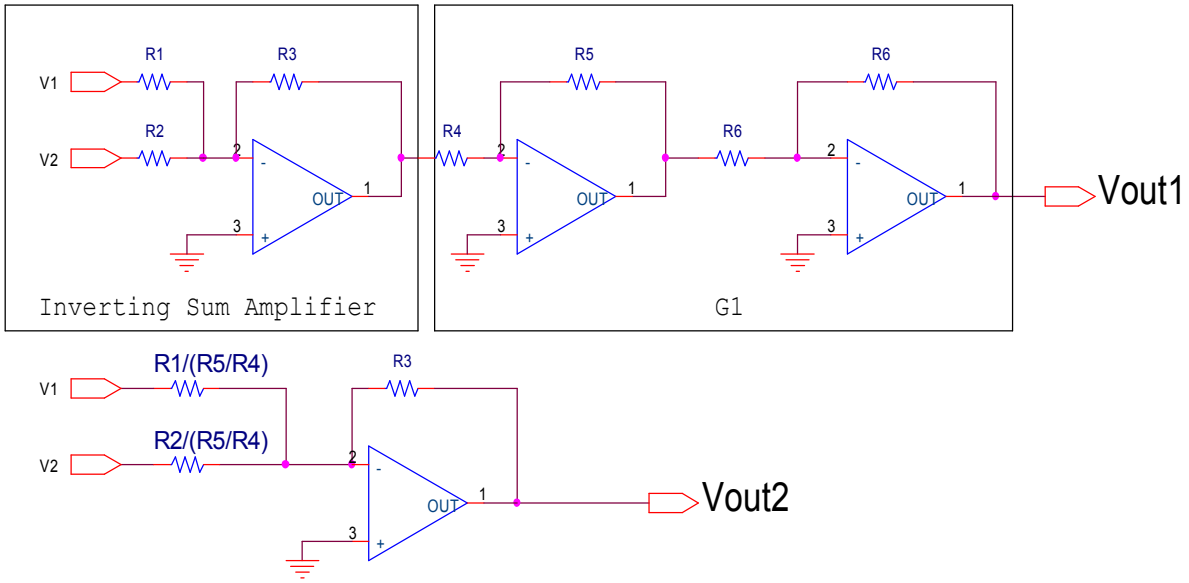


Figure 4.5. Simplifying the FPAA circuit by merging G1 into the inverting sum amplifier.

The output V_{out1} of the circuit with unmerged gain blocks is

$$V_{out1}=\begin{cases} V_0, & R_3(\frac{V_1}{R_1}+\frac{V_2}{R_2})\geq 1 \\ -\frac{R_5R_3}{R_4}(\frac{V_1}{R_1}+\frac{V_2}{R_2}), & -1<R_3(\frac{V_1}{R_1}+\frac{V_2}{R_2})<1 \\ -V_0 & R_3(\frac{V_1}{R_1}+\frac{V_2}{R_2})\leq -1 \end{cases}. \quad (10)$$

The output V_{out2} of the circuit with merged gain blocks is

$$V_{out2}=\begin{cases} V_0, & \frac{R_5R_3}{R_4}(\frac{V_1}{R_1}+\frac{V_2}{R_2})\geq V_0 \\ -\frac{R_5R_3}{R_4}(\frac{V_1}{R_1}+\frac{V_2}{R_2}), & -V_0<\frac{R_5R_3}{R_4}(\frac{V_1}{R_1}+\frac{V_2}{R_2})<V_0 \\ -V_0, & \frac{R_5R_3}{R_4}(\frac{V_1}{R_1}+\frac{V_2}{R_2})\leq -V_0 \end{cases}. \quad (11)$$

Since $\frac{R_5}{R_4}=V_0$, thus $R_3(\frac{V_1}{R_1}+\frac{V_2}{R_2})\geq 1$ is equivalent to $\frac{R_5R_3}{R_4}(\frac{V_1}{R_1}+\frac{V_2}{R_2})\geq V_0$, $-1<R_3(\frac{V_1}{R_1}+\frac{V_2}{R_2})<1$ is equivalent to $-V_0<\frac{R_5R_3}{R_4}(\frac{V_1}{R_1}+\frac{V_2}{R_2})<V_0$ and $R_3(\frac{V_1}{R_1}+\frac{V_2}{R_2})\leq -1$ is equivalent to $\frac{R_5R_3}{R_4}(\frac{V_1}{R_1}+\frac{V_2}{R_2})\leq -V_0$. Thus the two circuits are equivalent.

Similar proof can be applied to the merging of G2 into the subsequent functional blocks of the FPAA.

III. Logic Gates Implementation Using Single-chip FPAA

This section applies the network architecture simplification method to solve the classic neural network benchmark problem: XOR problem. Moreover, the versatility of FPAA chips is shown by the implementation of other three logic gates XNOR, AND or OR gate

with slight change of network parameters and/or circuit configuration for the XOR gate. Speed performance of the implementation is also addressed.

The XOR problem is a classic problem for neural networks since it is a simple, non-linearly separable problem that can be solved by neural networks.

The piece-wise linear transfer function was chosen for the neural network because it can be easily realized by inverting gain stages in two low cost Configurable Analog Modules (CAMs). The circuit configuration can be further simplified by merging the two gain modules into inverting summer modules required by the neural network.

A 2-2-1 feedforward neural network topology was trained in MATLAB for XOR, XNOR, AND and OR gates, and the 4 resulting sets of weights were stored for downloading.

The FPAA used in our simulation and experiments is the AN221E04 FPAA from Anadigm^{TM} Inc. The AN221E04 is a dynamically reconfigurable analog chip composed of op-amps, comparators and switched programmable capacitors. FPAA technology enables rapid-prototyping of analog circuits by programming the configurable analog modules supported by the chip, such as gain blocks, inverters, summing inverters, adders, multipliers, integrators, quadratic/linear analog filter blocks, and sine wave generators. With the aid of design software AnadigmDesigner 2, the FPAA can translate complex analog circuits into the simple set of system/block level design instead of transistor level design, and thus gives designers the analog equivalent of an FPGA. Moreover, it places analog functions under real-time software control within the system.

Figure 4.6, 4.7, 4.8 and 4.9 show the actual circuit configurations and simulation results for the XOR, XNOR, AND and OR gates. Note that the circuit configuration of XOR is same as OR gate as well as the identity of the circuit configurations between XNOR and AND gate. The two types of circuit configurations differ by only one analog functional block. Each network has its own set of parameters. The parameters are obtained by training the network in MATLAB. Thus it is easy to change the circuit from one type of logic gate to any of other three gates.

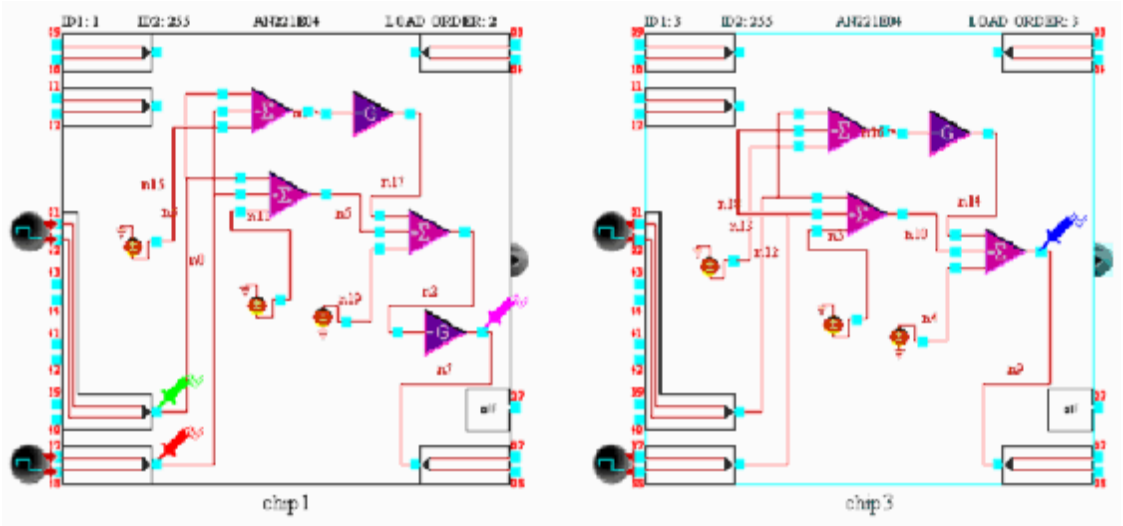


Figure 4.6. XNOR-gate FPAA circuit (left) and XOR-gate FPAA circuit (right)

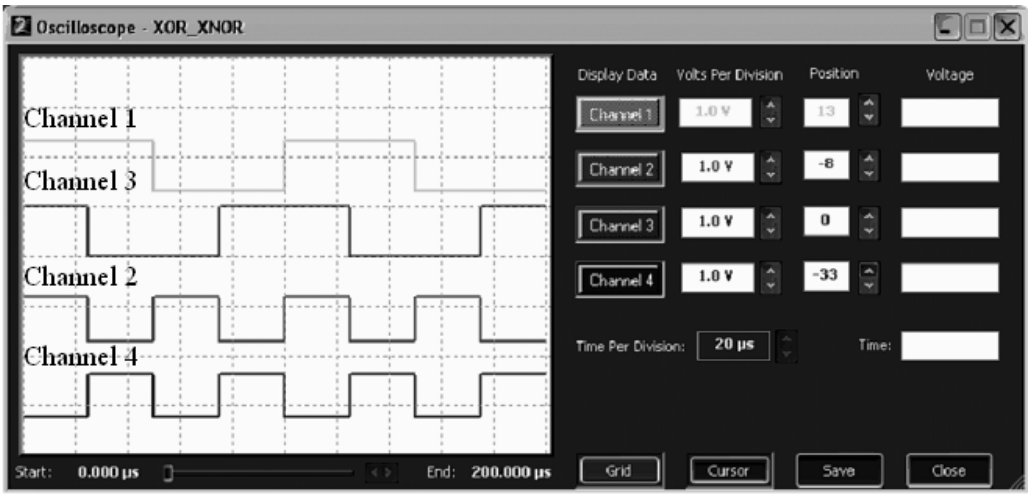


Figure 4.7. Simulation results of XNOR (channel 2) and XOR gate (Channel 4). Channel 1 and 3 are two inputs.

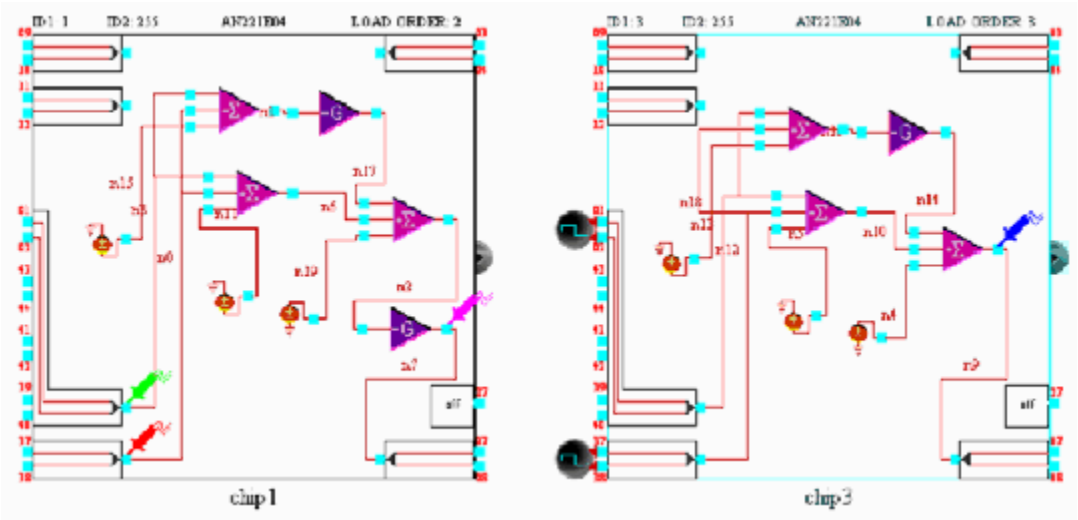


Figure 4.8. AND-gate FPAA circuit (left) and OR-gate FPAA circuit (right)

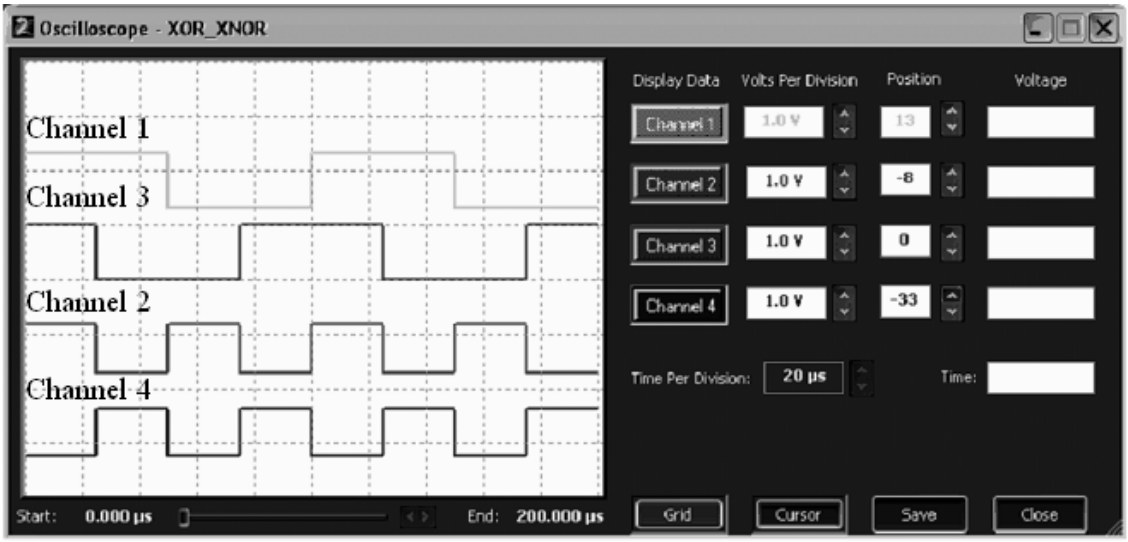


Figure 4.9. Simulation results of AND (channel 2) and OR gate (Channel 4). Channel 1 and 3 are two inputs

The measured time delay between input and output for the XOR gate is 0.2 microseconds. Our 3-layer neural network has 6 connections so the neural network has the performance of 6/0.2 = 30 Million Connections per Second (MCPS) [33].

IV. Multi-chip FPAA Based Neural Network Classifying 2 Groups of Data

A. The Two Classes of Data

The 8-point version of the "alternate labels" problem [32] is chosen to as an example to demonstrate the speed advantage of using FPAA implementation. The problem has two classes of data points. Let the two class be A and B. Each class has 4 data points alternating with the 4 data points of the other group in two dimensions. Each data point is represented in the usual way as an ordered pair of numbers as shown in Figure 4.10; we call the elements of the n^{th} pair x_n and y_n (n = 1, 2, …, 8). All 8 data points have the same the y values thus $y_1 = y_2 = \ldots = y_8$. "a" (represented by squares) and "b" (represented by

circles) are two different real numbers representing the two classes. Without loss of generality, we can assume that the interval between successive x_n's is constant, say 0.4, and that the values of all y_n are the same, say 1.0, as shown in Table I. A feedforward neural network with several neurons in the hidden layer can generate the decision boundaries.

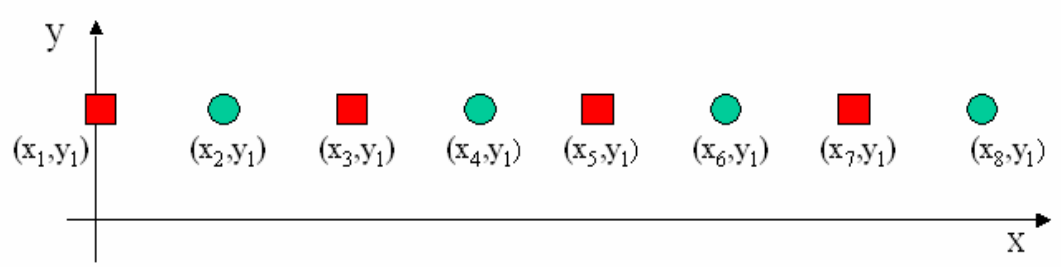


Figure 4.10. Seven decision boundaries separating 8 data points of 2 classes.

B. Classifying the Two Groups of Data

Table 4.1 shows the input and output values of the 8 data points used in our simulation and experiment.

Table 4.1. Two classes of data: class a = 0 and class b = 1.

Input x	0	0.4	0.8	1.2	1.6	2.0	2.4	2.8
Input y	1	1	1	1	1	1	1	1
Output z	0	1	0	1	0	1	0	1

The neural network needs to implicitly generate the desired decision boundaries based the input data pairs in order to make proper classification. To classify these 8 data points, a 2-5-1 neural network is trained using the training data in Table I in MATLAB to obtain the weights. The neural network has 5 neurons in the hidden layer which has the PL

activation functions and one output neuron to construct a linear combination of the outputs of the 5 hidden neurons. Before the neural network is mapped onto the FPAA, it is simulated in MATLAB/SIMULINK to verify the separation capabilities of the network. As a result, the neural network achieves 100 percent classification accuracy as shown in Figure 4.11 with the output (z) threshold chosen to be 0.5.

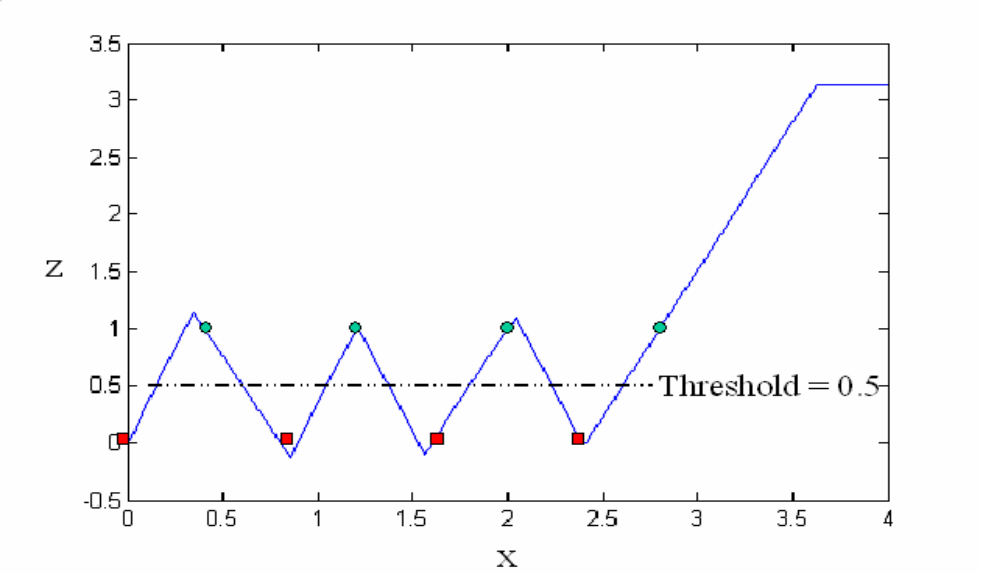


Figure 4.11. The output of the trained neural network view in x-z plane at y=1.

(Simulated in MATLAB/SIMULINK from the MathWorks, Natick, MA, USA)

C. Simulation and Experimental Results

In this section we map the trained neural network onto the FPAA devices.

The 2-5-1 neural network with parameters obtained by the MATLAB/SIMULINK model was mapped onto the FPAA programmed using only Inverting Gain Amplifier functional blocks (represented by "Inv G" in the Figure) and Inverting Sum Amplifier functional blocks (represented by "Inv Sum" in Figure 8). The element "b" in the figure is the trained bias input for each neuron. We programmed the Inverting Sum Amplifier to

accept at most 3 inputs; and several of the Inverting Sum Amplifiers are cascaded between the hidden layer and the output layer to realize the sum operation for the output neuron. The accumulated sign-flips of Inverting Gain Amplifiers are correctly accounted for by additional inversions when necessary. For example, in Figure 4.12, the trained weight for the Y input to the first neuron in the hidden layer has the negative sign but there are 4 Inverting Sum Amplifiers between the input Y and the final output which provides a positive sign; thus an Inverting Gain Amplifier is needed in the signal path to generate the negative sign. The exact location of the Inverting Gain Amplifier in the signal path is chosen based on the available programming resources of each chip.

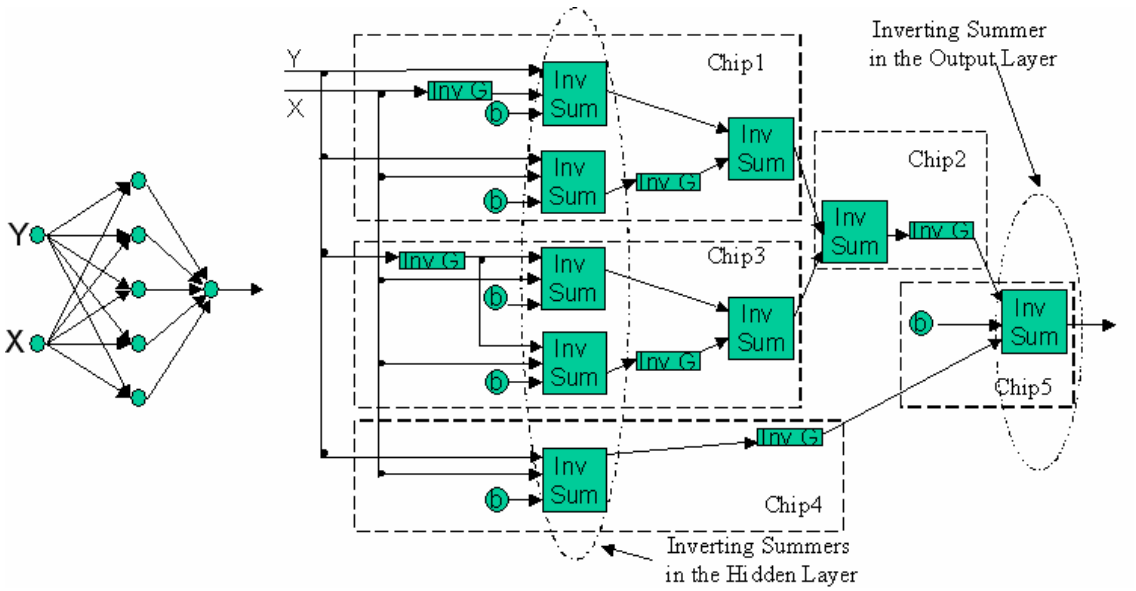


Figure 4.12. Constructing a 2-5-1 neural network using configurable analog modules of the FPAA.

Five AN221E04 chips are integrated together to realize the neural network as shown in Figure 4.13. The network is decomposed into five modules as shown in Figure 4.12 and each module is encapsulated in one chip. The simulation result using AnadigmDesigner 2 is shown in Figure 4.14. Input Y is a test signal of 1*v* constant voltage and Input X is the

triangular voltage input peaking at 3*v*. Setting the output threshold at 0.5*v*, the network classifies the data with 100% accuracy. The experimental result showing more details of the classification is shown in Figure 4.15, which is the oscilloscope screen shot of the experiment result.

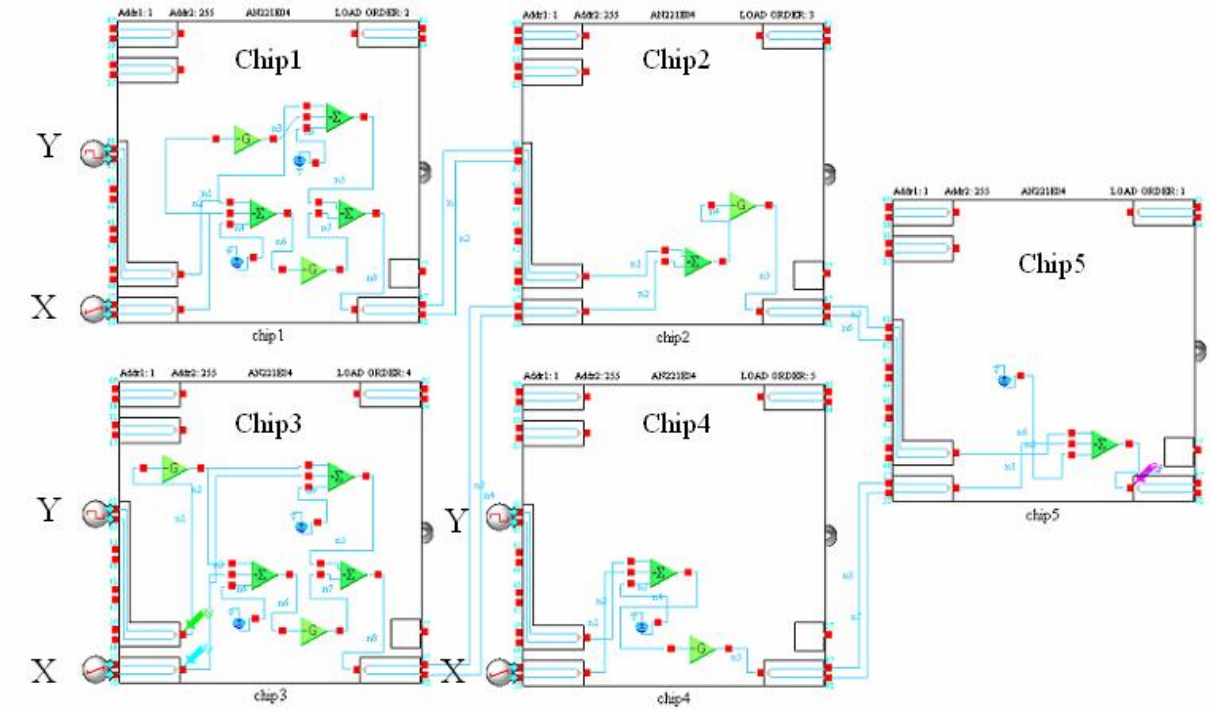


Figure 4.13. The multi-chip FPAA based neural network programmed using software AnadigmDesigner 2.

Figure 4.14. The simulation results of neural network classifying two classes of data.

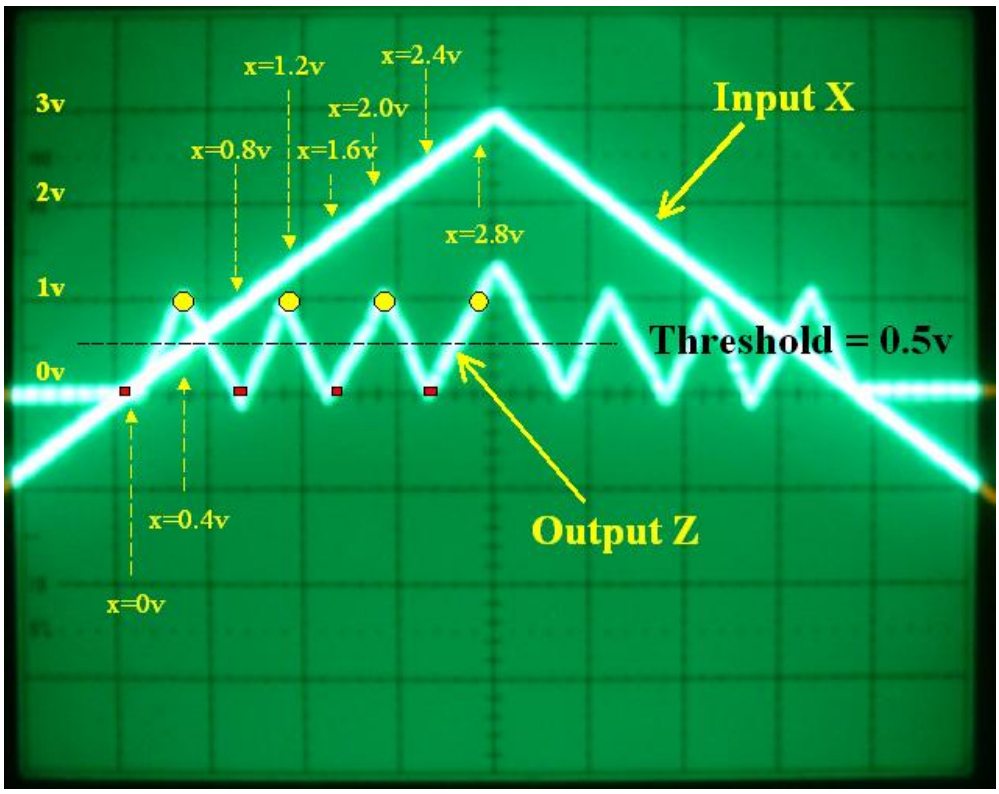


Figure 4.15. Experimental results of neural network classifying two classes of data.

As shown in Figure 4.15, the neural network trained from a 2-5-1 neural network separates the two classes of data into 2 regions and makes correct classifications of all data points with the threshold chosen to be 0.5*v*. We would also like to evaluate the speed performance for our multi-chip neural network using the standard neural network hardware measuring criteria: Millions of Connections Per Second (MCPS) [33]. The measured delay from the network input to the network output is 2.5 microseconds and there are 15 connections, yielding 6.0 MCPS in actual measured speed performance. Figure 4.16 shows the 5 FPAA evaluation boards for the experiments.

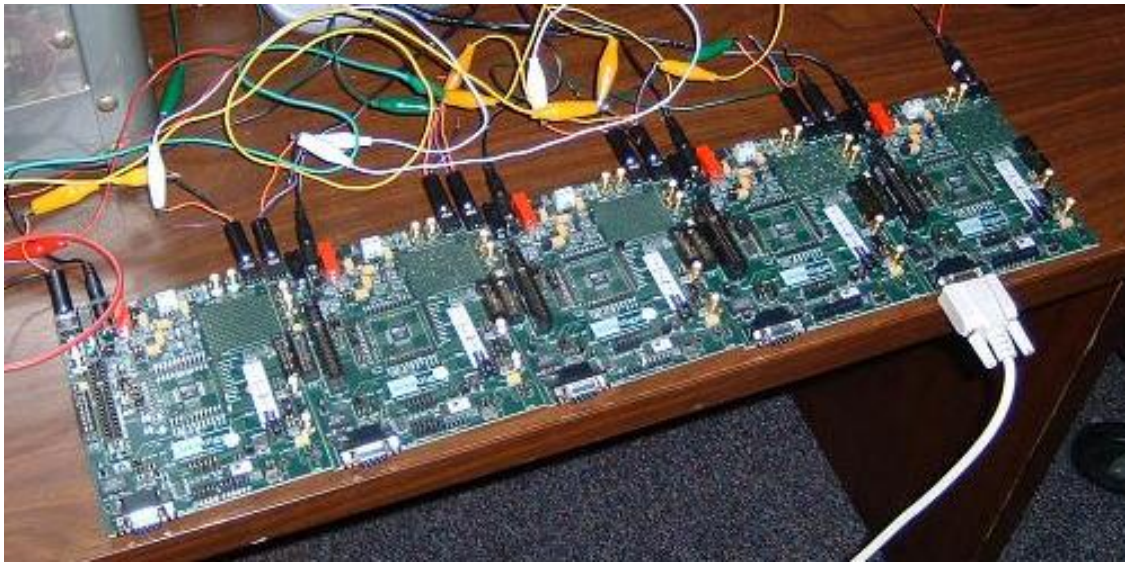


Figure 4.16. The five FPAA evaluation boards for the experiment.

V. Analysis of Speed Performance

To compare the speed performance of neural network implementation using FPAA to the software implementation (MATLAB on an Intel Celeron 2 GHz machine), neural networks with 4 architectures: 2-2-1, 2-3-1, 2-4-1 and 2-5-1 are implemented using both FPAA and the software. The measured implementation time of the neural network (time

delay from the input to the output of the network) of all four architectures is 2.5 microseconds (error bound is below 0.5%) on the FPAA, independent of the number of neurons in the hidden layer. One the other hand, the software implementation time is more than 3.6 milliseconds. As a result, the FPAA implements the neural network more than 1400 times faster than the software implementation. Figure 4.17 shows the relationship between the software implementation time and the number of the neurons in the hidden layer of the network. It is shown that adding neurons into the hidden layer increases the overall software implementation time. This is because software instructions that are executed serially cannot take advantage of the inherent parallelism of ANN architectures as FPAA does. This indicates that the speed difference between two implementation approaches become more obvious with the increase of the number of the hidden nodes. Note the experiment results are only to qualitatively show how the implementation time is affected with different neuron numbers instead of showing the exact functional relationship between software implementation time and the neuron numbers. All in all, the FPAA implementation of the neural network has superior performance over software implementation.

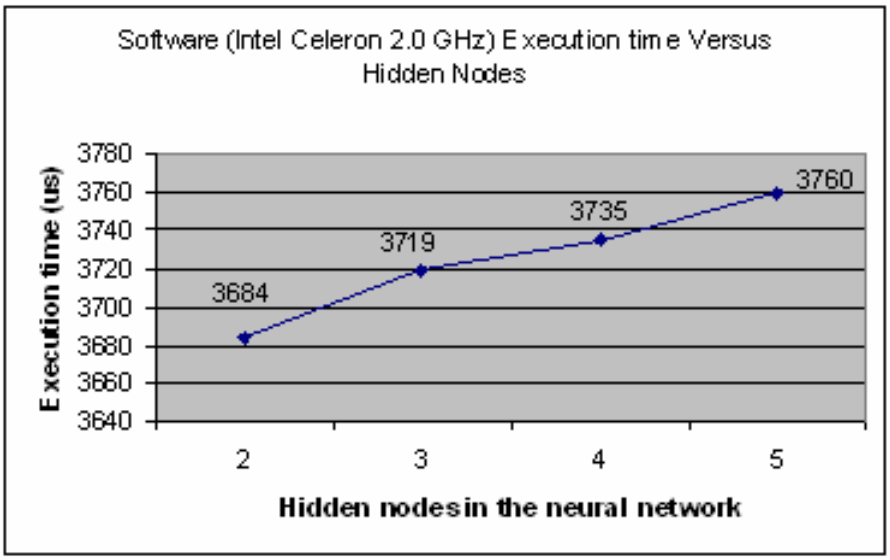


Figure 4.17. Neural network execution time by software versus number of hidden nodes.

VI. Discussion on the Scalability of the Structure

The structure is scalable for the neural network which has same number of inputs/outputs and more neurons in the hidden layer. More summer blocks are required to obtain the final output. The positions of inverting gain blocks may need to be adjusted according the signs of the weights. An example of scaling is shown in Figure 4.18.

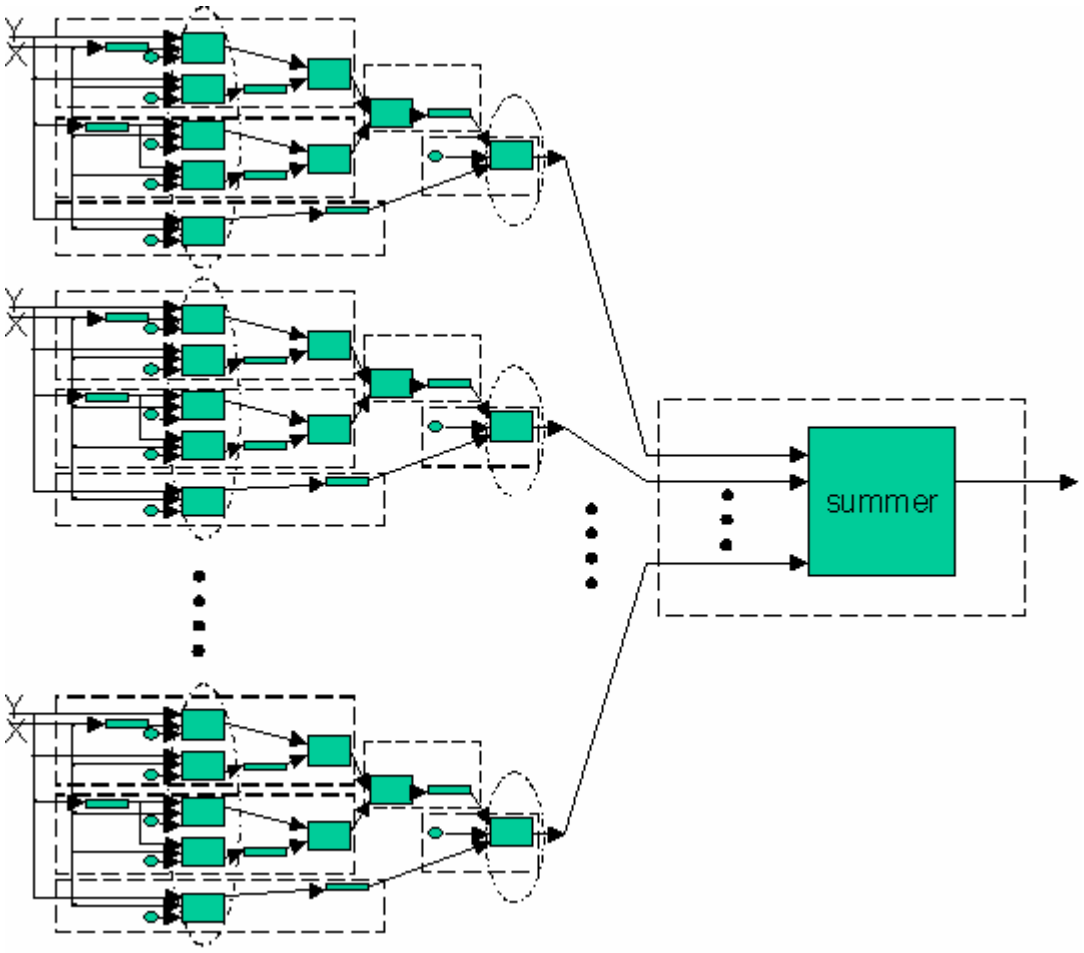


Figure 4.18. Scalability of the FPAA based ANN

VII. Conclusion

This paper demonstrates the hardware implementation a feedforward artificial neural network using low-cost commercially available FPAA chips. We proposed a simplified realization for neurons with piecewise linear activation functions and thereby reduced the complexity of the neural network architecture correspondingly. Our final ANN requires only two types of analog function blocks: the Inverting Gain Amplifier and the Inverting Sum Amplifier. In this effort, we did not require the many other functional blocks available on the Anadigm FPAA chip, but these additional resources can be combined with ANNs for conventional signal processing at the input or output of an ANN.

Demonstrated with simulation and experimental results, a single chip FPAA is programmed to realize 4 different logic gates with similar circuit configurations; moreover, a multi-chip FPAA circuit correctly performs a classification task at the speed of 6.0 MCPS. We used 5 chips to realize the 2-5-1 ANN for the classification task, which suggests more complicated network architectures can be realized by integrating more FPAA chips. We found that FPAA-based ANNs are convenient to implement fast to operate and scalable. We conclude that the proposed approach to realizing ANNs is suitable for real time applications.

VIII. References

[1] F. L.Lewis, "Neural-network control of robot manipulators," *IEEE Expert*, pp. 64-75, June 1996.

[2] J. Teeter and M.-Y. Chow, "Application of Functional Link Neural Network to HVAC Thermal Dynamic System Identification," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 1, pp. 170-176, 1998.

[3] M.-Y. Chow and J. Teeter, "A Knowledge-Based Approach for Improved Neural Network Control of a Servomotor System with Nonlinear Friction Characteristics," *Mechatronics*, vol. 5, no. 8, pp. 949-962, 1995.

[4] B. Ayhan, M.-Y. Chow, and M.-H. Song, "Monolith and Partition Schemes with LDA and Neural Networks as Detector Units for Induction Motor Broken Rotor Bar Fault Detection," *KIEE International Transactions on Electrical Machinery and Energy Conversion Systems*, June 1, 2005 (invited).

[5] M.Y. Chow, "Methodologies of Using Artificial Neural Network and Fuzzy Logic Technologies for Motor Incipient Fault Detection," *World Scientific Publishing Co.* Pte. Ltd., 1998.

[6] M.-Y. Chow, G. Bilbro, and S. O. Yee, "Application of Learning Theory to a Single Phase Induction Motor Incipient Fault Detection Artificial Neural Network," *International Journal of Neural Systems*, vol. 2, no. 1&2, pp. 91-100, 1991.

[7] M. Holler, S. Tam, H. Castro and R. Benson, “An electrically trainable artificial neural network (ETANN) with 10240 `floating gate' synapses ,” *Neural Networks, 1989. IJCNN, International Joint Conference on*, pp. 191 - 196 vol.2, 18-22 June 1989.

[8] S. Tam, B. Gupta, H. Castro and M. Holler, “Learning on an Analog VLSI Neural Network Chip,” *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*, 1990.

[9] Y. Maeda, H. Hirano and Y. Kanata, “AN Analog Neural Network Circuit with a Learning Rule via Simutaneous Perturbation,” *Proceedings of the IJCNN-93-Nagoya*, pp. 853-856, 1993.

[10] S. S. Kim and S. Jung, "Hardware implementation of a real time neural network controller with a DSP and an FPGA," *presented at Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference* on, 2004.

[11] W. Qinruo, Y. Bo, X. Yun, and L. Bingru, "The hardware structure design of perceptron with FPGA implementation," *presented at Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 2003.

[12] S. B. Yun, Y. J. Kim, S. S. Dong, and C. H. Lee, "Hardware implementation of neural network with expansible and reconfigurable architecture," *presented at Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*, 2002.

[13] H. Withagen, “Implementing Backpropagation with Analog Hardware,” *Proceedings of the IEEE ICNN-94-Orlando Florida*, pp. 2015-2017, 1994.

[14] T. Szabo, L. Antoni, G. Horvath, and B. Feher, "A full-parallel digital implementation for pre-trained NNs," presented at Neural Networks, 2000. IJCNN 2000, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, 2000.

[15] S. Popescu, "Hardware implementation of fast neural networks using CPLD," *presented at Neural Network Applications in Electrical Engineering, Proceedings of the 5th Seminar on*, 2000.

[16] B. Girau, "Digital hardware implementation of 2D compatible neural networks," *presented at Neural Networks, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, 2000.

[17] H. Abdelbaki, E. Gelenbe, and S. E. EL-Khamy, "Analog hardware implementation of the random neural network model," *presented at Neural Networks, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, 2000.

[18] J. Zhu, G. J. Milne, and B. K. Gunther, "Towards an FPGA based reconfigurable computing environment for neural network implementations," *presented at Artificial Neural Networks, Ninth International Conference on*, 1999.

[19] J. Liu and M. Brooke, "A fully parallel learning neural network chip for real-time control," *presented at Neural Networks, International Joint Conference on*, 1999.

[20] J. Liu and M. Brooke, "Fully parallel on-chip learning hardware neural network for real-time control," *presented at Circuits and Systems, Proceedings of the IEEE International Symposium on*, 1999.

[21] E. J. Brauer, J. J. Abbas, B. Callaway, J. Colvin, and J. Farris, "Hardware implementation of a neural network pattern shaper algorithm," *presented at Neural Networks, International Joint Conference on*, 1999.

[22] P. M. Engel and R. F. Molz, "A new proposal for implementation of competitive neural networks in analog hardware," *presented at Neural Networks, Proceedings. 5th Brazilian Symposium on*, 1998.

[23] J. Tang, M. R. Varley, and M. S. Peak, "Hardware implementations of multi-layer feedforward neural networks and error backpropagation using 8-bit PIC microcontrollers," *presented at Neural and Fuzzy Systems: Design, Hardware and Applications, IEE Colloquium on*, 1997.

[24] D. S. Reay, T. C. Green, and B. W. Williams, "Field programmable gate array implementation of a neural network accelerator," *presented at Hardware Implementation of Neural Networks and Fuzzy Logic, IEE Colloquium on*, 1994.

[25] A. Achyuthan and M. I. Elmasry, "Mixed analog/digital hardware synthesis of artificial neural networks," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 13, pp. 1073-1087, 1994.

[26] P. Dong, G. Bilbro, and M.-Y. Chow, "Controlling a Path-tracking Unmanned Ground Vehicle with a Field-Programmable Analog Array," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Monterey, CA, 24-28 July, 2005.

[27] P. I. Yakimov, E. D. Manolov, and M. H. Hristov, "Design and implementation of a V-f converter using FPAA," *presented at Electronics Technology: Meeting the Challenges of Electronics Technology Progress, 2004. 27th International Spring Seminar on*, 2004.

[28] M. Sekerli and R. J. Butera, "An implementation of a simple neuron model in field programmable analog arrays," *presented at Engineering in Medicine and Biology Society, 2004. EMBC 2004. Conference Proceedings. 26th Annual International Conference of the*, 2004.

[29] K. Basterretxea, J. M. Tarela, and I. del Campo, "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons," *Circuits, Devices and Systems, IEE Proceedings [see also IEE Proceedings G- Circuits, Devices and Systems]*, vol. 151, pp. 18-24, 2004.

[30] A. Atiya, E. Gad, S. Shaheen, and A. El-Dessouky, "On training piecewise linear networks," *presented at Neural Networks for Signal Processing IX, Proceedings of the IEEE Signal Processing Society Workshop*, 1999.

[31] A. Bermak and A. Bouzerdoum, "VLSI implementation of a neural network classifier based on the saturating linear activation function," *presented at Neural Information Processing, Proceedings of the 9th International Conference on*, 2002.

[32] S. Ridella, S. Rovetta, and R. Zunino, "Circular backpropagation networks for classification," *Neural Networks, IEEE Transactions on*, vol. 8, pp. 84-97, 1997.

[33] E. van Keulen, S. Colak, H. Withagen, and H. Hegt, "Neural network hardware performance criteria," *presented at Neural Networks, IEEE World Congress on Computational Intelligence*, 1994 IEEE International Conference on, 1994.

CHAPTER V - Controlling a Path-tracking Unmanned Ground Vehicle with a Field-Programmable Analog Array

Puxuan Dong[1*]
IEEE Student Member
pdong@ncsu.edu

Griff Bilbro[2]
IEEE Senior Member
glb@ncsu.edu

Mo-Yuen Chow[1]
IEEE Senior Member
chow@ncsu.edu

[1] Advanced Diagnosis Automation & Control Lab
Department of Electrical and Computer Engineering,
North Carolina State University, Raleigh NC 27695 USA
Phone: +1(919)515-5405

[2] Department of Electrical and Computer Engineering,
North Carolina State University, Raleigh NC 27695 USA

* Corresponding Author

This chapter was accepted to IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2005), 24–28 July 2005, Monterey, CA.

Abstract

Unmanned ground vehicle (UGV) path-tracking has been an important topic in mechatronics real-time applications. This paper describes and compares the implementation and performance of path-tracking unmanned ground vehicle using a field programmable analog array (FPAA) and conventional digital microcontroller. The FPAA AN10E40 is a general-purpose, digitally reconfigurable analog signal-processing chip. Its current commercial applications center on signal conditioning and base-band analog signal processing and rapid prototyping. This paper will show that the AN10E40 can also readily implement a control system for a path-tracking UGV. Using PI control for the path tracking, the FPAA controlled UGV made about 38% fewer tracking error with 22% faster traveling speed than the digital microcontroller (MC68HC11) controlled UGV due to the fast processing time of the FPAA. The results indicate the great potential of using FPAA for real-time control in mechatronics systems.

I. Introduction

Unmanned ground vehicle (UGV) path-tracking is an important topic in mechatronics, robotics and automation. Although much research work has been done in UGV path-tracking [1-5], controlling the UGV with programmable analog controller is presented for the first time in our paper. In [2], Koh and Cho formulated a path-tracking problem for an unmanned ground vehicle, which moves along a pre-defined path. Tipsuwan and Chow proposed the use of gain scheduling to optimally control a mobile robot over IP network [3]. Kanayama and Fahroo proposed a steering function as a line tracking method for nonholonomic vehicles [4]. Besides the research work dedicated to this area, there are

regular competitions of path-tracking UGVs held by several organizations including Dallas Personal Robotics Group (DPRG) and Chicago Area Robotics Group (Chibotics). Most of today's path-tracking UGVs at these competitions use off-the-shelf digital microcontrollers such as Atmel AVR microprocessor, PIC16C74A microprocessor, PIC16F84 microprocessor, or the Motorola MC68HC11 processor. Digital microcontrollers require analog-to-digital and digital-to-analog signal processing. No field programmable analog controller for path-tracking UGVs has been reported. The FPAA, a new programmable analog technology, has potential to play a major role in future unmanned ground vehicle real-time applications. The FPAA AN10E40 is a general-purpose, digitally reconfigurable analog signal-processing chip. Its current commercial applications center on signal conditioning and base-band analog signal processing and rapid prototyping. With its dynamic reconfigurability and fast analog signal processing speed, FPAA can be used in real time adaptive control such as [6]. The FPAA is chosen as controller for the path-tracking UGV because the signal processing is totally in the analog domain. This has the advantage of producing simpler systems than are possible with digital microcontrollers. Moreover, the convenient software design environment provided with FPAA evaluation kits and its dynamic reconfigurability make it attractive for developing control systems. FPAAs enable many popular controllers including P, PI, PD and PID to be conveniently and quickly realized. In FPAA-based systems the signal remains in the analog domain at all times, but the configuration of a FPAA chip is digital and its open-ended functionality is programmable. The resulting designs require minimum hardware, are convenient to prototype and refine, and are highly reliable [7].

The task in this paper is a simple path tracking problem with reflective sensors and 2 DC motors. Although DC motor control has been investigated intensively by many researchers [8-11], FPAA is employed as the controller for the first time in our paper. For our system, two FPAA-based PI controllers were designed for each DC motor of the UGV that drives each wheel, to control the UGV to move along a predefined path as shown in Figure 5.1 using photomicrosensors and control implementation performed by FPAA. The path is composed of three half-circles and two quarter-circles connected by straight lines. The UGV will start from one point on the straight line and travel along the whole loop and return to its starting point. The goal of the control is to make the UGV track the path as accurate as possible and run as fast as possible. The UGV will start from one point on the track and run along the track. The performance of the FPAA controlled UGV is evaluated based on two aspects: the traveling time C1 and the error rate C2. The traveling time is the time for the UGV to run along the path until it reaches back to its starting point. The definition of the error rate will be described in section III.

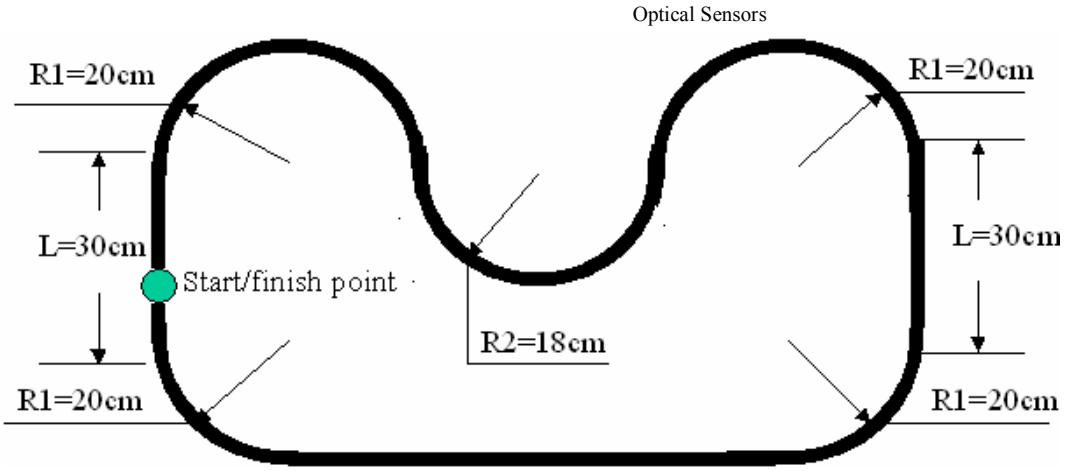


Figure 5.1. UGV and the path to track. Total length of the path is 380.92cm

II. Design

A. The Unmanned Ground Vehicle

The unmanned ground vehicle, as shown in Figure 5.2, used in the Advanced Diagnosis, Automation, and Control (ADAC) lab of North Carolina State University is a UGV which was previously controlled by MC68HC11 microcontroller embedded in Handy Board. The Handy Board is a commercially available digital microcontroller system originally developed at MIT for educational uses [12]. In addition to the control, the UGV has optical sensors, an H-bridge circuit for driving DC motors, two DC motors and the power supply. This paper will compare the UGV path-tracking performance by using the MC68HC11 and by using the FPAA AN10E40.

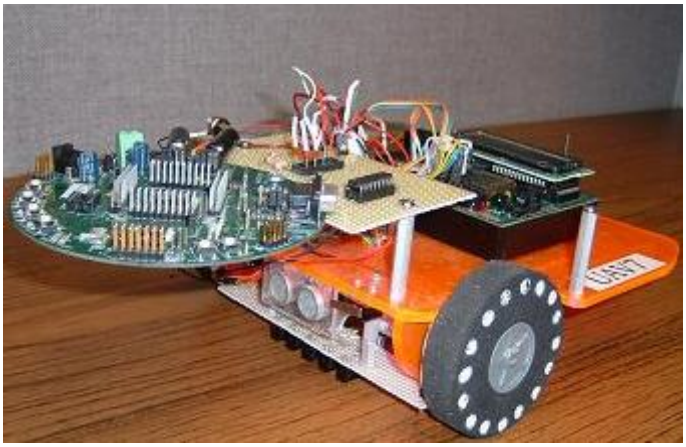


Figure 5.2. UGV used in ADAC lab at North Carolina State University.

B. System Architecture

The path-tracking UGV estimates its track position with optical sensors mounted at its front end as shown in Figure 5.3. The overall path-tracking closed-loop control system is shown in Figure 5.4. The FPAA calculates the required control voltage for the DC motors, and sends control signals to an H-bridge circuit, which is used as an interface

between the FPAA and the DC motors [13]. The FPAA was configured to output a PWM voltage signal in the range from 0 to 5v in order to comply with the input specifications of the H-bridge circuit. The five optical sensors sense the relative position of the UGV with respect to the track. These optical sensor signals were sent to the FPAA from which it produces signals for controlling the DC motors to adjust the positions of the UGV with respect to the track.

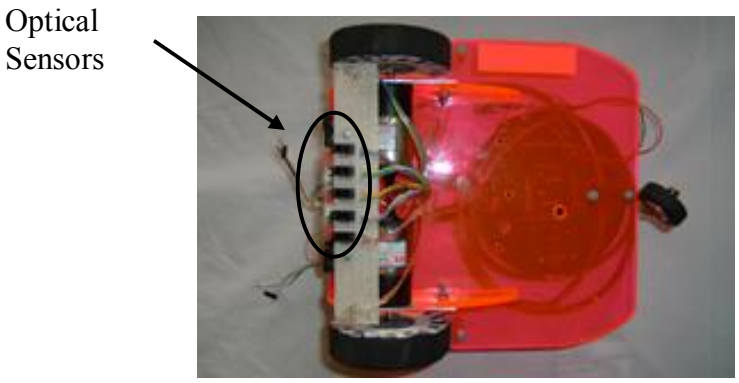


Figure 5.3. Birdseye view of the UGV.

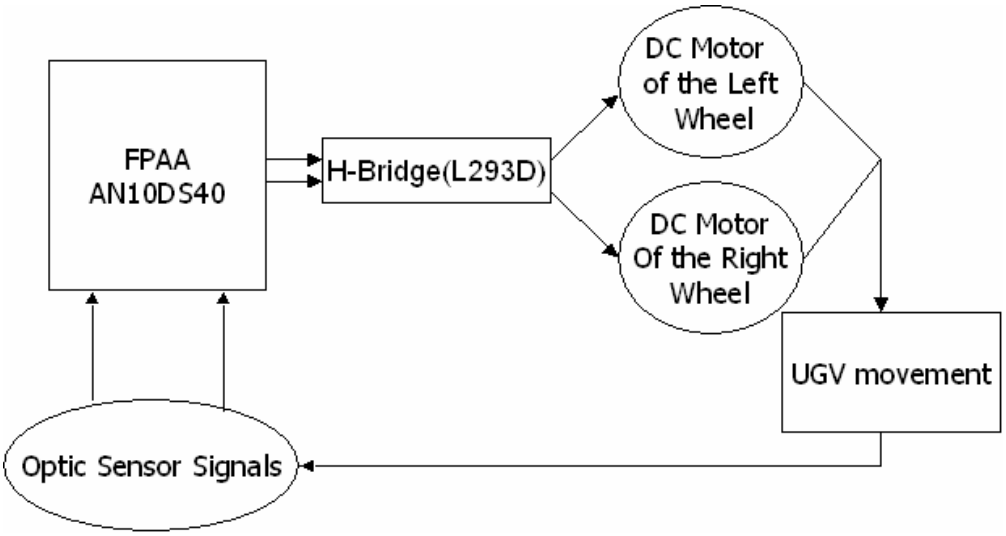


Figure 5.4. System architecture of the FPAA - controlled unmanned ground vehicle.

The controller selected for the UGV is AN10DS40 Evaluation and Development System for AN10E40 Field Programmable Analog Array as shown in Figure 5.5.

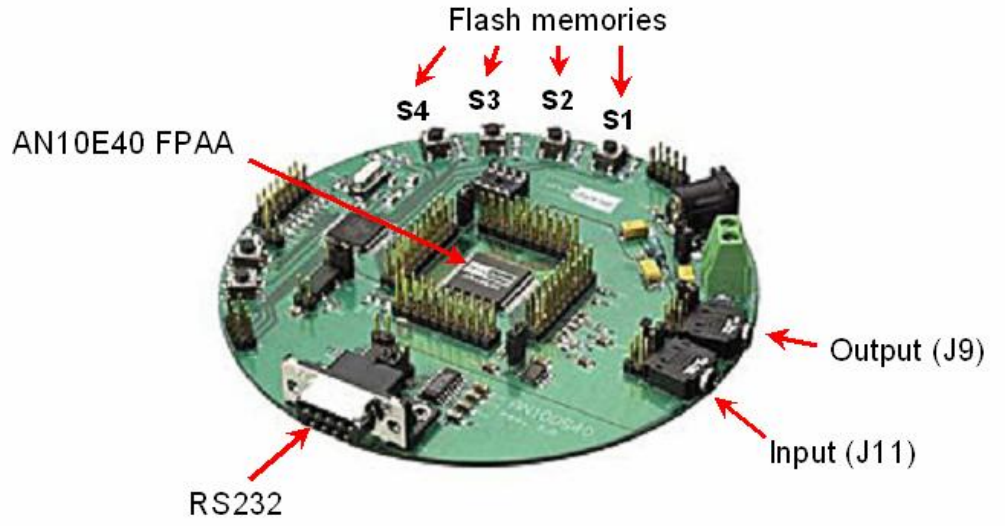


Figure 5.5. The AN10DS40 Evaluation and Development System.

The AN10E40 FPAA is the square chip at the center of the board. The rest of the board facilitates the development. There is no encoder or microcontroller inside the AN10E40 chip. The chip itself is a stand-alone system, and requires only a 5V external power supply and an EEPROM for non-volatile memory for power-up. The array is supported by 13 input/output cells as part of its monolithic integrated circuit.

After programming a control algorithm using AnadigmDesigner, the control configuration can be downloaded to the chip directly or to the flash memory on evaluation board through a RS232 cable. The on-board microcontroller provides four FPAA configurations in its flash memory. The development board allows any of four pre-programmed alternative configurations for the AN10E40 to be loaded after powered up from flash memory by pushing buttons S1-S4, as labeled in Figure 5.6. The sensors used in the UGV are EE-SF5 Reflective Photomicrosensors manufactured by OMRON. The internal circuit is showed in Figure 5.6.

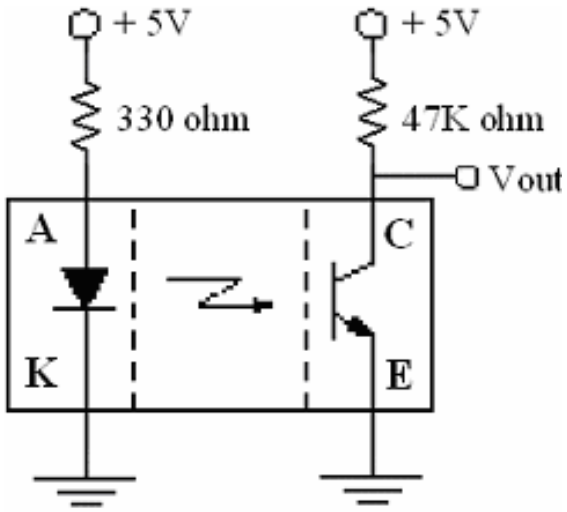


Figure 5.6. Internal circuit of photomicrosensors manufactured by OMRON.

A – Anode, K – Cathode, C – Collector and E – Emitter

A H-Bridge circuit is used as the interface between the FPAA and the DC motors. The H-bridge is specifically designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors [14] while FPAA cannot directly drive DC motors by itself. H-Bridges allow forward and reverse motor control by closing one or the other pair of diagonally opposing switches.

As shown in Figure 5.7, a separate supply voltage connected to VCC1 is provided for the logic input circuits to minimize device power dissipation. Supply voltage VCC2 is used for the output circuits.

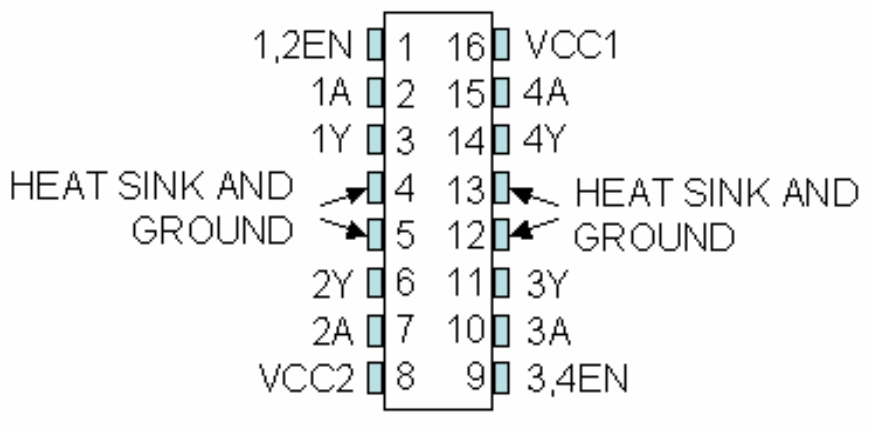


Figure 5.7. The H-Bridge circuit (L293D) and its function table for each driver.

We have two 12VDC Reversible Gear Head Motors [15] to drive the two wheels of the UGV. The H-bridge circuit drives the two DC motors simultaneously. Figure 5.8 shows the diagram of the H-bridge circuit connecting to the two DC motors.

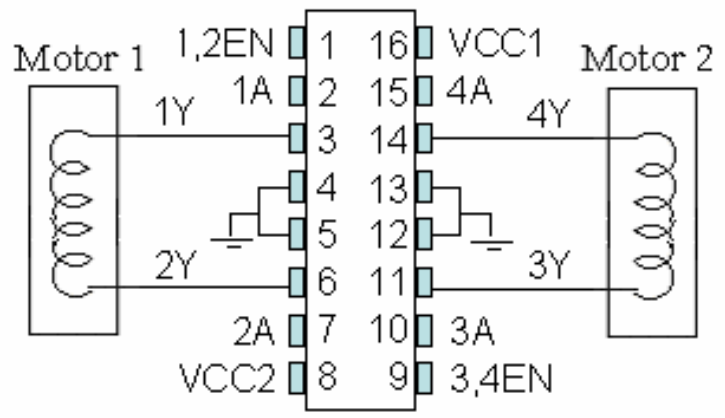


Figure 5.8. Connection between the H-bridge circuit (L293D) and DC motors.

To turn the motor, a high (+5 volts or logic 1) is sent to the 1A line while a low (0 volts or logic 0) is delivered to the 2A line. To turn the motor in the opposite direction, a high is sent to the 2A line while a low is sent to the 1A line. The other motor is controlled similarly based on the 4A and 3A inputs.

The motor runs at full speed when logic 1 is applied to 1A and logic 0 to 2A. When both motors run at full speed the measured UGV's maximum speed is 10.2 cm/sec. We controlled the motor speed by adjusting the duty cycle of a conventional pulse width modulation (PWM) signal [16], which the FPAA can conveniently be configured to synthesize.

C. The Control System

The path selected for tracking is in the Advanced Diagnosis, Automation, and Control (ADAC) Laboratory of Electrical and Computer Engineering Department at North Carolina State University as shown in Figure 5.9. The width of the track is generally less than the distance between the left edge of sensor 1 and the right edge of sensor 5 but

more than the distance between the right edge of sensor 1 and the left edge of sensor 5. Let the track width be W, then it has the following relationship to L1 and L2 as shown in Figure 11: L1 < W < L2.

Figure 5.9. Track for the unmanned ground vehicle testing.

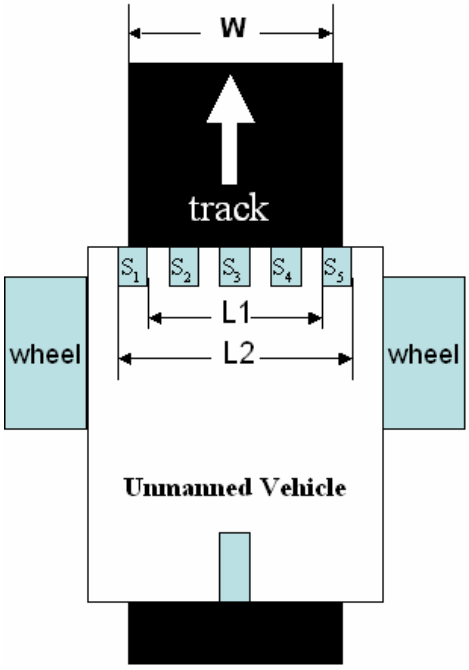


Figure 5.10. The unmanned ground vehicle with 5 optical sensors mounted at its front end.

The error processing here refers to the computing of the errors in the control loop. The unmanned ground vehicle has 5 sensors to sense its position with respect to the path. When any sensor is fully on the black track it outputs a nominal 5v signal; when it fully off the track (on white background) it outputs a nominal 0v. When partial sensor is the off the track the output voltage will be between 0 and 5v which is proportional to the area of the sensor that is on the track. The sensors are labeled from left to right as S_1, S_2, S_3, S_4 and S_5 as shown in Figure 10. The voltage output of each sensor is represented by V_1, V_2, V_3, V_4 and V_5.When the middle sensor S_3 is centered on the track and sensor line is perpendicular to the tangent line of the track, the voltage outputs of S_1 and S_5 are the same and sensors S_2, S_3 and S_4 are at 5v.This ideal position is shown in Figure 5.10 and requires no position correction.

The speed of DC motors are controlled by PI gain according to the signals of optical sensors. A derivative control could be used to decrease the overshoot of a system but for our system the output signal updating of the controller is much faster than the mechanical response of the UGV so there is very little overshoot resulted. Thus D component is not included in our design. When the voltage difference $\Delta V_{1,5}$ between S_1 and S_5 is 0, the DC motors are configured to run at full speed and S_1 and S_5 are in symmetrical positions on the track. When S_1 and S_5 are in asymmetrical positions of the track as the right picture of Figure 5.11 shows, there will be a voltage difference between the outputs of S_1 and S_5 that is used to regulate the speed of the DC motors.

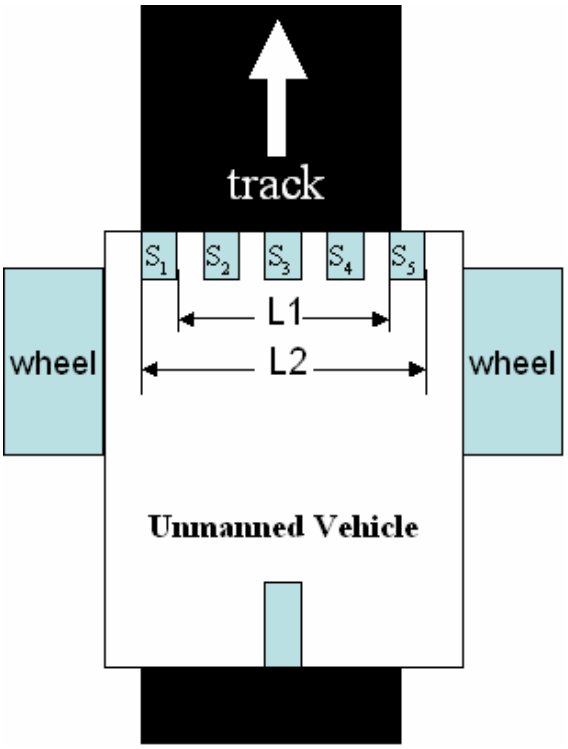


Figure 5.11. Sensor S1 and S5 are in asymmetrical positions of the track.

The control is closed-loop as shown in Figure 5.12.

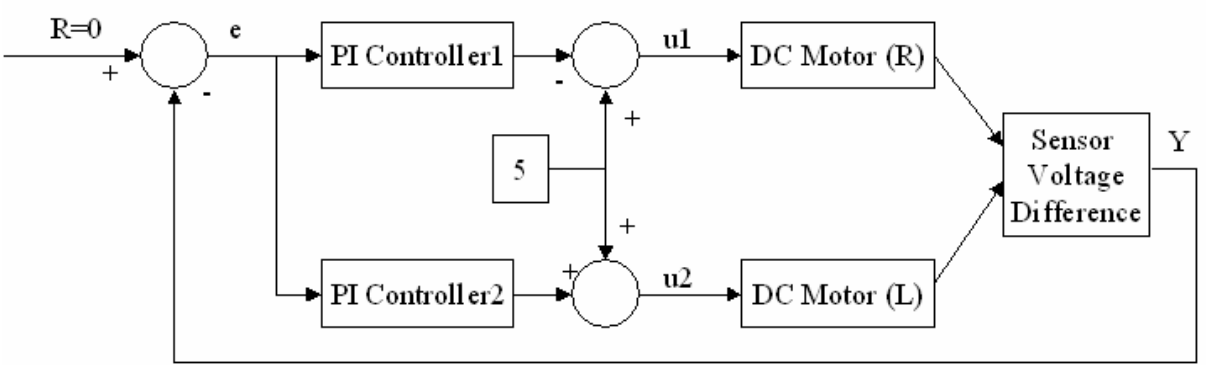


Figure 5.12. The closed-loop control of the system.

R is the reference signal that corresponds to the zero voltage in our design. Y is the actual output that corresponds to the voltage difference between sensor 1 and sensor 5:

$$Y = V_1 - V_5 = \Delta V_{1,5} \tag{1}$$

Y is equal to zero volts when the UGV is right on the path. The e represents the error where

$$e = R - Y \tag{2}$$

The error e is sent to the PI controller that also produces the control signal u1 and u2 for the DC motors. For the right motor, the control signal u subtracted from 5v is equal to the proportional gain (K_p) times the magnitude of the error plus the integral gain (K_I) times the integral of the error:

$$5\text{-}u = K_p e + K_I \int e dt \tag{3}$$

For the left motor, the control signal u minus 5v is equal to the proportional gain (K_p) times the magnitude of the error plus the integral gain (K_I) times the integral of the error:

$$\text{-}5\text{+}u = K_p e + K_I \int e dt \tag{4}$$

Control signals saturate at 5v since that is the upper bound of FPAA output signal. When error is equal to zero, the control signal will be 5v for both motors so the UGV will run at full speed. When error is not equal to zero, the calculated control signal will be

sent to corresponding DC motor to decrease its speed then the position of the UGV can be adjusted.

The H-bridge circuit acts as the interface between the FPAA and the DC motors as shown in Figure 5.13. The FPAA is configured to output PWM signals that are compatible with the H-bridge circuit.

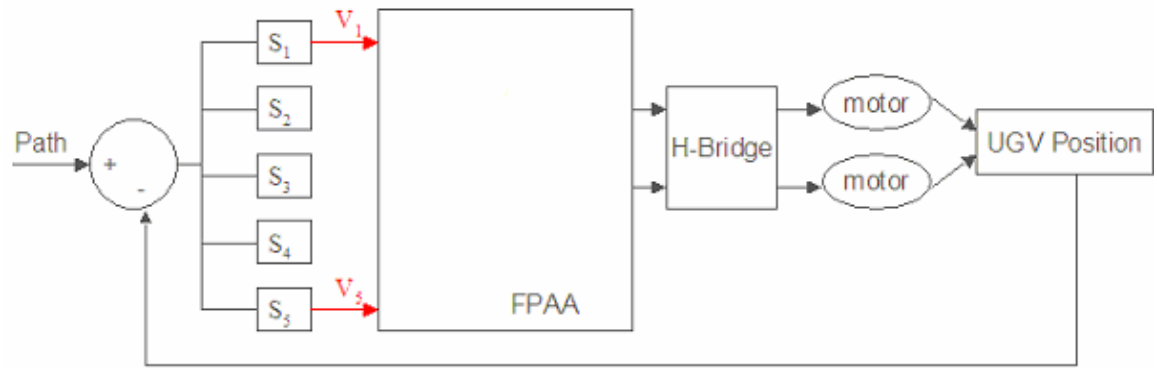


Figure 5.13. H-bridge circuits acts as the interface between the FPAA and the DC motors

The DC voltage signals can be converted to PWM signals by comparing itself to sine waves. The DC-to-PWM signal conversion is realized with the FPAA by configuring the resources in the AN10E40 FPAA as a sine wave oscillator, another as a comparator, and connecting them as if they were discrete components or cells in an ASIC design to process the DC signal in the usual way.

Configurations for the FPAA such as the PWM generator are developed using AnadigmDesigner that represents the design logically. Figure 5.14 shows the simulation results of PWM generation. The sine oscillator function block in the FPAA, like most other function blocks is parameterizable. Any such parameters, such as the oscillator

frequency in this case, can be modified by the user with a dialog box displayed by AnadigmDesigner.

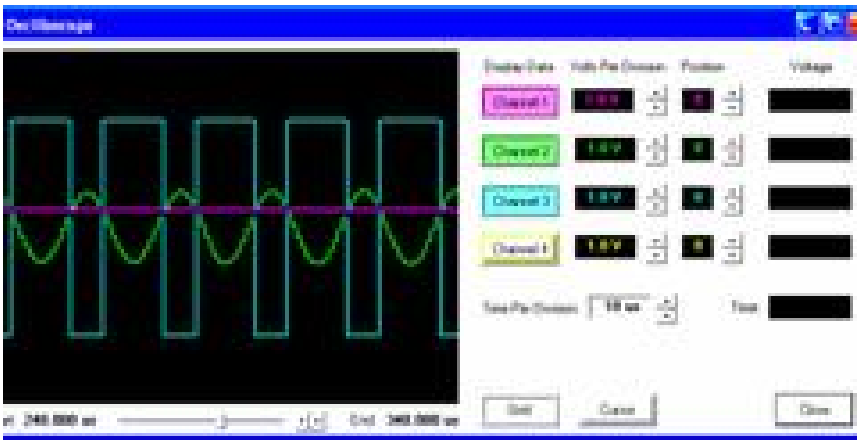


Figure 5.14. Simulation result showing the generated PWM signal.

To conclude, the FPAA is configured to be a PI controller and it also converts the DC control signal u to PWM signals for the H-bridge circuit. The final FPAA circuit programmed with AnadigmDesigner is shown in Figure 5.15.

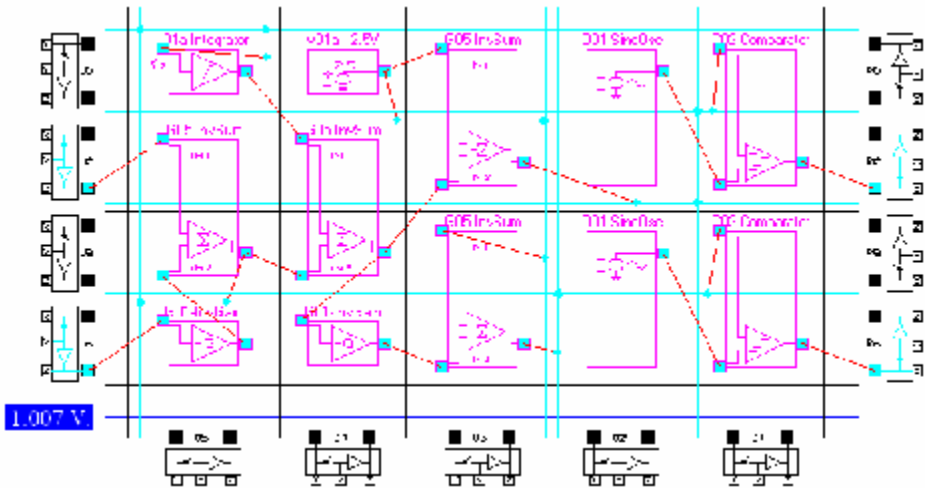


Figure 5.15. FPAA circuit that controls the path-tracking unmanned ground vehicle.

III. Experimental Results and Comparison to Microcontroller controlled UGV

We compared our FPAA controlled UGV with microcontroller controlled UGV under identical test conditions. The MC68HC11 controlled UGV system architecture is exactly the same as the FPAA controlled UGV as shown is Figure 5.16. The MC68HC11 needs is programmed using the language Interactive C.

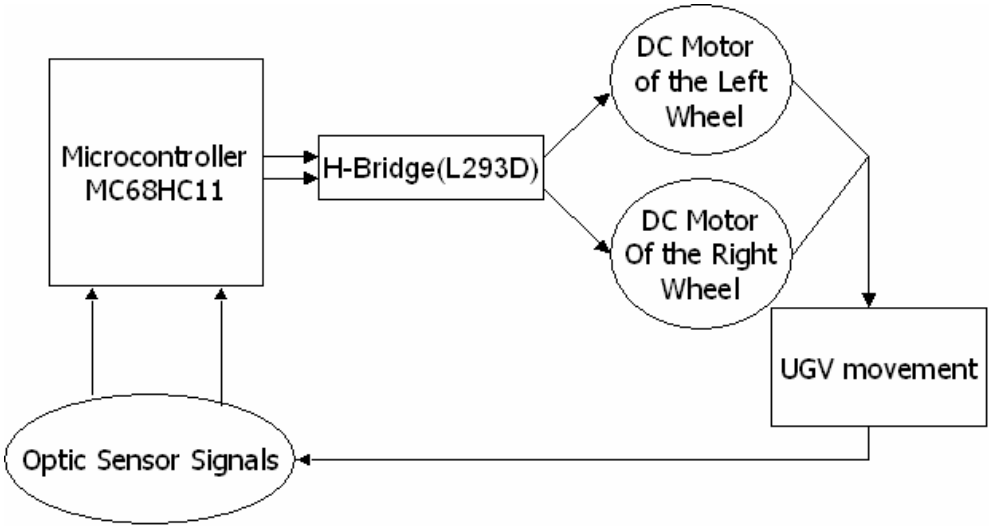


Figure 5.16. System architecture of microcontroller - controlled unmanned ground vehicle.

To make a fair comparison, both controllers were well tuned before testing. Moreover, both the FPAA evaluation board and the Handy Board were mounted on the UGV during all test runs to maintain the weight of the system constant. A second MC68HC11 controller does the error recording and time recording for performance comparison.

The error rate (error/sec) and the average running time of one trip are two aspects of the performance comparison. As for the error rate, we count an error occur when more than half of the sensor is off the track. The range of voltage outputs from the sensors are from 0 to 5v, so if V_i is equal 2.5v then half of the sensor S_i is on/off the track. 2.5v correspond

the integer number 255*(2.5/5) = 128 for sensor output read by the microcontroller MC68HC11. The integer number 128 is thus set as the threshold for error recording. An error occurs when the sensor output is below 128. However, when the two side sensors output values lower than 128 but the middle three sensors are on track, the position of the UGV needs no correction as shown in Figure 10. In this case, the UGV is deemed as right on track and no error are recorded by the second microcontroller. The second MC68HC11 microcontroller records the error every 100 milliseconds. The second MC68HC11 microcontroller also measures how long it takes the UGV to finish one roundtrip along the path. The experimental data of error recording and time recording are based on the average of 15 test runs for each controller. Since the experiment might have randomness which is treated as random error, we use Monte Carlo simulation approach and compare the resulted means and medians to draw statistical conclusion.

The Error rate comparison results are shown is Table 5.1. Running time comparison results are shown in Table 5.2.

Table 5.1. Error rate comparison between the FPAA–controlled UGV and the microcontroller–controlled UGV.

Error Rate (error/sec)	**Mean**	**Median**
Microcontroller MC68HC11 (e_m)	7.423	7.385
FPAA AN10E40 (e_f)	4.659	4.511
Decrease in Error Rate $(e_m-e_f)/e_m$	37.24 %	38.92 %

Table 5.2. Running time comparison between the FPAA–controlled UGV and the microcontroller–controlled UGV.

Running Time (sec)	Mean	Median
Microcontroller MC68HC11 (t_m)	52.31	53.27
FPAA AN10E40 (t_f)	40.89	41.06
Increase in Speed $[(1/t_m)-(1/t_f)]/(1/t_m)$	21.83 %	22.92 %

By comparing the error rate, we can see from Table 5.1 and Table 5.2 that the FPAA-controlled UGV showed better performance by making about 38% fewer error than the microcontroller-controlled UGV. Moreover, the FPAA controlled UGV runs 22% faster than the microcontroller controlled UGV. This results from the microcontroller MC68HC11's relatively slower speed of error processing. The error processing of the digital microcontroller includes analog to digital signal conversion, error calculation and digital to analog signal conversion. We have optimized our code to minimize the time required for the controller to run the code of error calculation. Nevertheless, it takes 16 milliseconds for the microcontroller to calculate the error. Apart from the time required for error calculation in the microcontroller, the controller also needs time to convert analog signals of optic sensors to digital signals and digital signals to back to analog signals to control the DC motors.

Error processing speed for the FPAA is much faster because of two reasons. First, it processes signals in the analog domain and requires no analog to digital conversion. Secondly, the circuit inside the FPAA was programmed by combining analog circuit blocks together instead of running the Interactive C code. The measured signal input to output delay of the FPAA circuit is less than 2 microseconds. From above analysis we

can say FPAA processes the error less than 2 microseconds while the MC68HC11 microcontroller processes the error more than 16 milliseconds. Consequently, FPAA process more errors than MC68HC11 does during the same time period. Thus FPAA can send more frequently updated control signals to DC motors to adjust position of UGV faster and then the UGV can track the path more accurately.

Besides the error rate comparison, from Table 2 we can see that the speed increases by 22% when FPAA takes the place of microcontroller to control the UGV. This is because the FPAA controlled UGV follows the path more accurately so it runs less distance than the microcontroller controlled UGV, thus it takes less time to finish running the whole loop.

To summarize, the analog FPAA is faster than the digital microcontroller. Moreover, the error calculation in the digital circuits of the microcontroller creates more time delay in signal processing. These factors lead to better performance for the FPAA than the microcontroller to control the UGV.

IV. Conclusion

The paper describes and compares the implementation of a field programmable analog array (FPAA) controlled and conventional digital microcontroller controlled path – tracking unmanned ground vehicle. The FPAA is a general-purpose, digitally reconfigurable analog signal-processing chip. Its current commercial applications center on signal conditioning and base-band analog signal processing and rapid prototyping. We have shown that the AN10E40, a kind of FPAA, can also readily implement a control system for a path-tracking UGV. The performance of the FPAA controlled UGV was

compared to a digital microcontroller (MC68HC11) controlled UGV that has been very popular in this field [12]. The FPAA controlled UGV made about 38% fewer error with 22% more running speed than the digital microcontroller controlled UGV. As a result, the FPAA showed much better performance over microcontroller for path-tracking unmanned ground vehicle and thus FPAA showed great potential in control systems.

The FPAA technology is new and is rapidly developing. The second-generation chip ANxE04 supports an IP module specifically intended for PID controllers and is also suitable for path-tracking UGVs as we will report in a separate article.

V. References

[1] J. Wit, C. Crane, D. Armstrong, "Autonomous ground vehicle path tracking," *Journal of Robotic Systems*, Vol. 21, Issue 8, pp. 439-449, August, 2004

[2] Koh, K. & Cho, H. "A smooth path tracking algorithm for wheeled mobile robots with dynamic constraints," *Journal of Intelligent and Robotic Systems: Theory and Applications* 24(4), pp.367-385, 1999.

[3] S. Kim, W. Lee, and J. Kim, "Research of the unmanned vehicle control and modeling for lane tracking and obstacle avoidance," in *Proc of the International Conference on Control, Automation, and Systems (ICCAS), Gyeongju, Korea,* October 2003.

[4] Y. Tipsuwan, M-Y Chow, "Neural Network Middleware for Model Predictive Path Tracking of Networked Mobile Robot over IP Network," *IEEE IECon'03, Roanoke, VA*, Nov 2 – Nov 6, 2003.

[5] Y. J. Kanayama and F. Fahroo, "A new line tracking method for nonholonomic vehicles," *Robotics and Automation, 1997. Proceedings, 1997 IEEE International Conference on* , Vol. 4, pp. 2908-2913, 20-25 April 1997.

[6] News from Anadigm Inc., "Analogue Arrays Turn to Loudspeaker Control," *Edited by: Electronicstalk Editorial Team,* Nov 1st. 2004.

[7] D. Anderson, C. Marcjan, D. Bersch, H. Anderson, P. Hu, O. Palusinki, D. Gettman, I. Macbeth, A. Bratt, "A Field Programmable Analog Array and its Application," *1997 Custom Integrated Circuits Conference Proceedings*, May, 5-8, 1997, Santa Clara, California, USA.

[8] M.-Y. Chow, Y. Tipsuwan, "Gain Adaptation of Networked Dc Motor Controllers on QoS Variations," *IEEE Transactions on Industrial Electronics*, Vol. 50, no. 5, October, 2003.

[9] J. T. Teeter, M.-Y. Chow, and J. J. B. Jr., "A Novel Fuzzy Friction Compensation Approach to Improve the Performance of a DC Motor Control System," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 1, pp. 113-120, 1996.

[10] C. Canudas, K. J. Astrom and K. Braun, "Adaptive friction compensation in DC-motor drives," *IEEE J. Robot. Automat*., vol. RA-3, no. 6, pp. 681-685, 1987.

[11] Y. Tipsuwan, M.-Y. Chow, "Fuzzy Logic Microcontroller Implementation for a DC Motor Speed Control", *IECon 99, San Jose, CA*, April 1999.

[12] www.handyboard.com

[13] B. Kim, C. Kim, S. Han, S. Kim, H. Park and H. Park, "1.2-μM non-epi CMOS smart power IC with four H-bridge motor drivers for portable applications," *Circuits and Systems, ISCAS '96., 'Connecting the World'., 1996 IEEE International Symposium on* , Vol.1 pp: 633 – 636, 12-15 May 1996

[14] Datasheet of L293D by Texas Instrument Inc.

[15] Product GH12-1828Y of Jameco Electronics Inc.

[16] N.N. Bengiamin and M. Jacobsen, "Pulse-width modulated drive for high performance DC motors," *Industry Applications Society Annual Meeting, 1988., Conference Record of the 1988 IEEE*, Vol.1, pp:543 – 550, 2-7 Oct. 1988.

[17] M.-Y. Chow, G. Bilbro, and S. O. Yee, "Application of Learning Theory to a Single Phase Induction Motor Incipient Fault Detection Artificial Neural Network," *International Journal of Neural Systems*, vol. 2, no. 1&2, pp. 91-100, 1991.

CHAPTER VI - Field Programmable Analog Array Based Artificial Neural Network for the Navigation of Unmanned Ground Vehicles

Puxuan Dong[1*]
IEEE Student Member
pdong@ncsu.edu

Griff Bilbro[2]
IEEE Senior Member
glb@ncsu.edu

Mo-Yuen Chow[1]
IEEE Senior Member
chow@ncsu.edu

[1] Advanced Diagnosis Automation & Control Lab
Department of Electrical and Computer Engineering,
North Carolina State University, Raleigh NC 27695 USA
Phone: +1(919)515-5405

[2] Department of Electrical and Computer Engineering,
North Carolina State University, Raleigh NC 27695 USA

* Corresponding Author

This chapter is to be submitted to IEEE Transactions on Mechatronics

Abstract

A field programmable analog array (FPAA) based artificial neural network (ANN) is designed to control a path-tracking unmanned ground vehicle. The ANN controller can be viewed as a nonlinear proportional, integral and derivative (PID) controller of which the adaptive gain is realized by the ANN to achieve better disturbance rejection performance and lower sensitivity to noise than a conventional PID controller. The stability of proposed controller is guaranteed by Popov stability analysis. UGV path-tracking experiments are reported. The performance of the proposed ANN controller compares favorably to a conventional PID controller. The ANN controller shows 35.9% improvement for noise rejection simulations. Moreover, a path-tracking UGV controlled by the ANN controller shows 6.4% less travel time and 15.4% few tracking error than the UGV controlled by a comparable PID controller. We conclude that the FPAA based ANN controller has better disturbance rejection and noise rejection than the conventional PID controller.

I. Introduction

Unmanned ground vehicle (UGV) path tracking has been used to test control technologies and various hardware controllers. Koh and Cho formulated a path-tracking problem for an unmanned ground vehicle [1], which moves along a pre-defined path. Tipsuwan and Chow proposed Gain Scheduling Middleware (GSM) technology to optimally control a mobile robot over IP network [2-3]. Kanayama and Fahroo proposed a novel steering function as a path-tracking method for nonholonomic vehicles [4]. Besides the research work dedicated to this area, there are regular competitions of path-

tracking UGV held by several organizations including Dallas Personal Robotics Group (DPRG) and Chicago Area Robotics Group (Chibotics). Most of today's path-tracking UGVs at these competitions use off-the-shelf digital microcontrollers such as Atmel AVR microprocessor, PIC16C74A microprocessor, PIC16F84 microprocessor, or the Motorola MC68HC11 processor.

Digital microcontrollers require analog-to-digital and digital-to-analog signal processing while FPAAs process the signal in analog domain. The first FPAA based PI controller for UGV path tracking was reported by Dong, Bilbro and Chow [5]. The FPAA based controller outperforms a digital controller MC68HC11 by processing the signal 8000 times faster.

Artificial neural networks are parallel information processing structures that have been used in path-tracking UGV controlling [6-9]. However, the ANN controller's immunity to disturbance and noise in UGV path tracking application hasn't been investigated to our knowledge. Moreover, no commercially available FPAA based ANN controller has ever been reported although a Hodgkin-Huxley neuron simulator has been implemented with FPAAs by Sekerli and Butera [21].

This paper presents a new ANN structure which is equivalent to a nonlinear PID controller. A neuron with hyperbolic tangent transfer function maps the nonlinear relationship between error and the exiting linear PID controller input, which we call an ANN controller. The ANN controller is used to control a path-tracking UGV. We find that the ANN controller has improved immunity to disturbance and noise compared to the linear PID controller which is also designed in FPAA chip. We demonstrate the improved

immunity with both simulation and experimental results. We also estimate the stability region of the ANN controller.

In our experiment, we choose a commercial FPAA as the implementation platform for our ANN controller. The FPAA adopted in this paper is the AN221E04 from Anadigm Inc [10]. The AN221E04 is a dynamically reconfigurable analog chip composed of op-amps, comparators and switched capacitors. Analog circuits can be rapid-prototyped by programming the configurable analog modules such as gain blocks, inverters, summing inverters, adders, multipliers, integrators and sine wave generators. The FPAA gives designers the analog equivalent of an FPGA. Moreover, it places analog functions under real-time software control.

The chip has 4 configurable analog modules (CAM) each of which has two fully differential Op Amps, capacitor banks, an 8-bit successive approximation register (SAR) for analog to digital converter, a high speed comparator, and a look up table (LUT).

The organization of the paper is as follows: In section II, we describe our ANN controller. In section III, we present simulation results that compare the disturbance rejection capability and the noise immunity of our ANN controller to a conventional PID controller. The stability analysis of the controller is also included in this section. In section IV, we experimentally compare ANN controllers with conventional PID controllers for a UGV path-tracking application. Section V presents some concluding remarks.

II. The Artificial Neural Network Controller

A standard PID controller is described formally by the following equation:

$$u(t) = K_p e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt}, \tag{1}$$

where *u(t)* is the output of the controller, K_P, K_I and K_D are the gain parameters of the proportional, integral and derivative part of the controller [25]. Here, the error *e* is the measured difference between the desired output and the obtained output. The block diagram of this controller is shown in Figure 6.1.

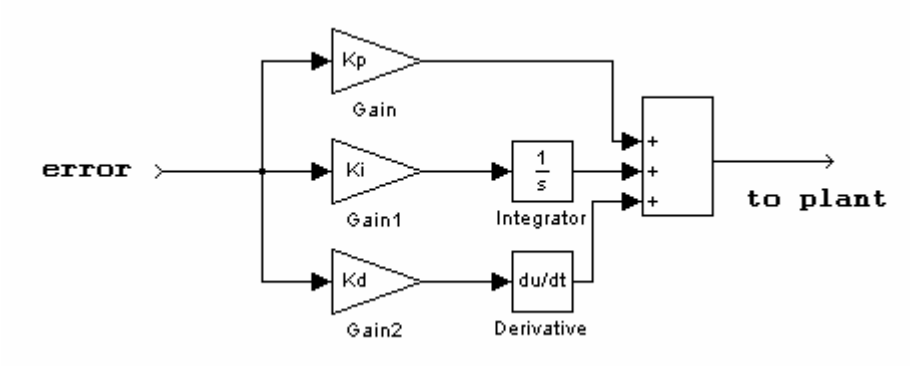


Figure 6.1. Block diagram of the conventional PID controller.

Nonlinear PID controllers have been used to adapt the controller to changes in operating conditions or environmental parameters which is beyond the capabilities of conventional fixed-gain PID controllers [11-14]. Armstrong, Neevel and Kusik describe the nonlinear PID controller in the following way:

$$u(t) = K_p(\bullet)e(t) + K_I(\bullet)\int_0^t e(t)dt + K_D(\bullet)\frac{de(t)}{dt}, \tag{2}$$

where $K_P(\bullet)$, $K_I(\bullet)$ and $K_D(\bullet)$ are time-varying controller gains [15].

We construct a nonlinear PID controller using a 3-layer artificial neural network. As shown in Figure 6.2, our design has one neuron with a nonlinear activation function in the input layer which provides the nonlinear gain, one P neuron, one I neuron, one D neuron in the middle layer, and one neuron with pure linear activation function in the output layer. The input-output function of the P neuron is a proportional function, the input-output function of the I neuron is an integral function and the input-output function of the D neuron is a derivative function [24].

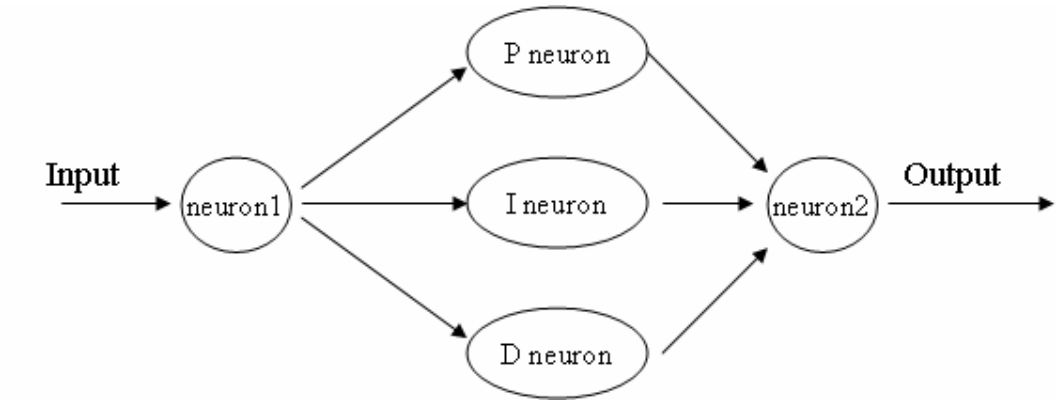


Figure 6.2. Artificial neural network with 3 layers.

Figure 6.3 shows further details of our ANN controller (nonlinear PID controller) in a particular closed-loop control application.

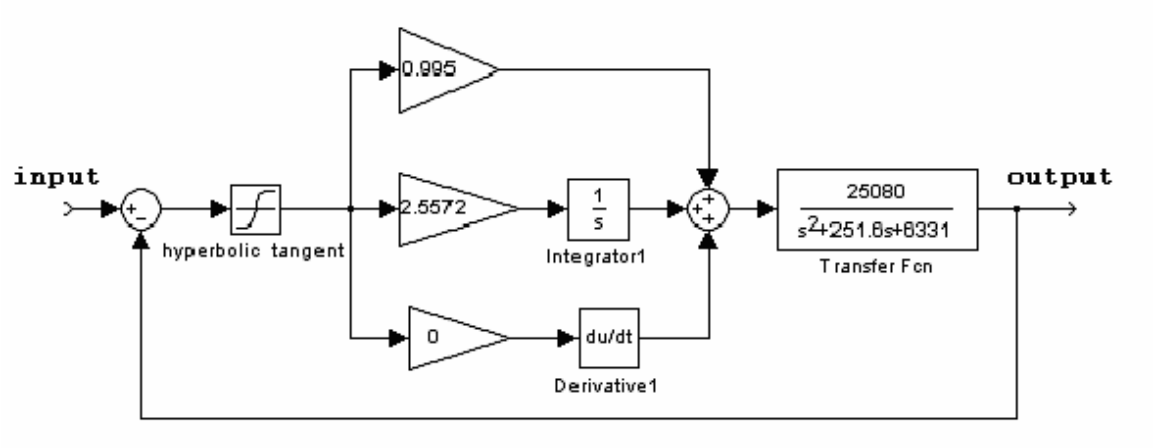


Figure 6.3. Example of an ANN controller in a closed-loop control.

In our design, a nonlinear activation function – hyperbolic tangent is cascaded with the linear PID controller to provide the nonlinear gain. The corresponding nonlinear gain obtained by the hyperbolic tangent function is

$$k = \frac{c \times \tanh(q \times e)}{e}, \tag{3}$$

where e is the error, c and q are positive real values. As an example, the activation function with q = 100 is shown in Figure 6.4. The nonlinear gain k is shown in Figure 5 for c = 0.6.

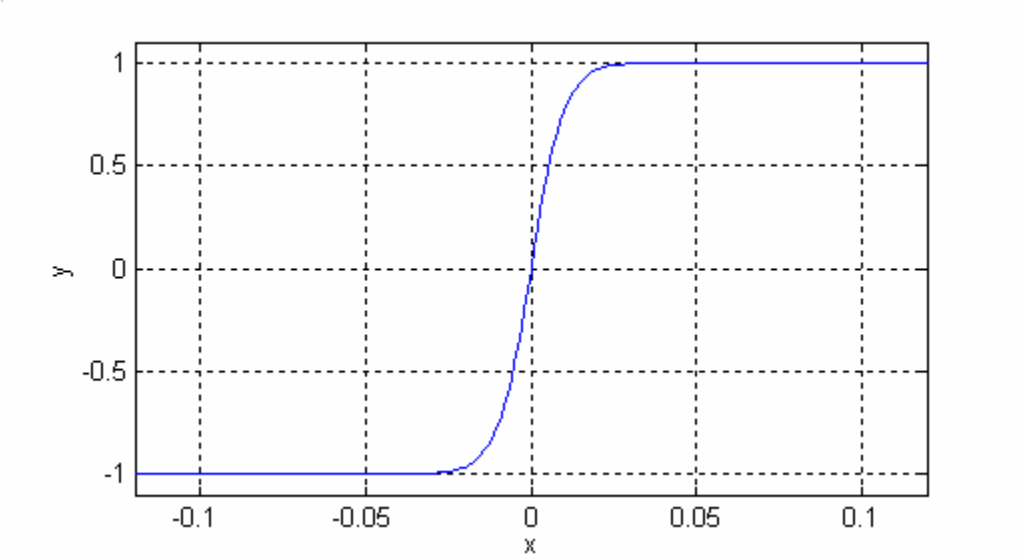


Figure 6.4. Activation function of the hidden node in the ANN.

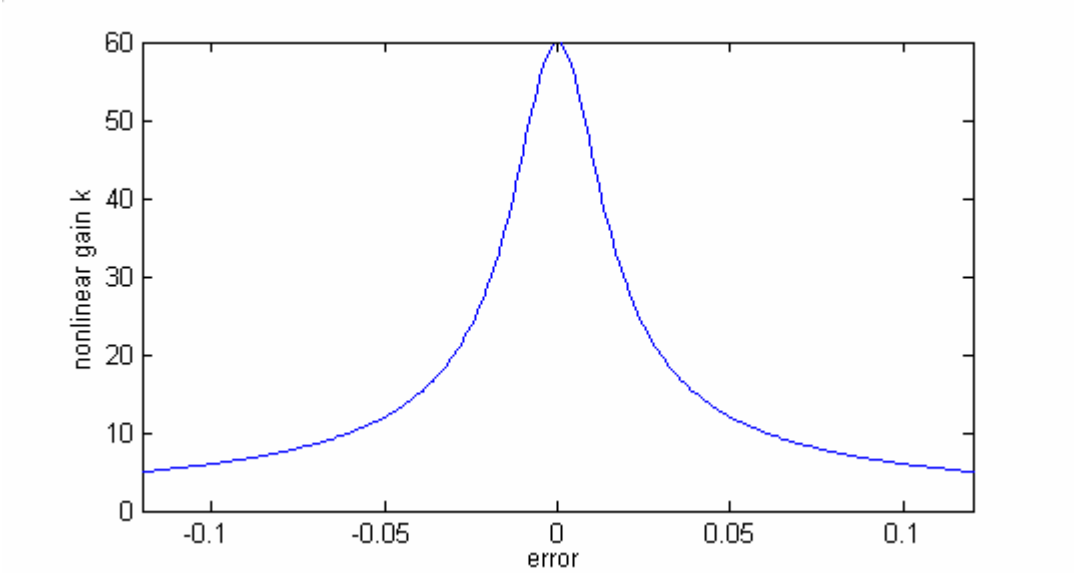


Figure 6.5. Variation of nonlinear gain k with respect to the error

As shown in Figure 6.5, the nonlinear gain k increases as the absolute value of error decreases.

III. Simulation Results

To demonstrate the advantages of the ANN controller over the linear PID controller, a DC motor is used as a case study for simulation.

A. DC Motor Characteristics

The dynamics of a DC motor for the UGV can be described by the differential equation:

$$\begin{aligned} L\frac{di}{dt} + Ri &= V - K_b\frac{d\theta}{dt} \\ J\frac{d^2\theta}{dt^2} + B\frac{d\theta}{dt} &= Ki + T_l \end{aligned}, \tag{4}$$

where L is the armature winding inductance, i is armature winding current, R is the armature winding resistance, V is the armature winding input voltage, K_b is back EMF constant, θ is the position of the shaft, J is moment of inertia of the motor and the wheel, B is the damping coefficient, K is the torque constant and T_l is the load torque. The Laplace transformations of equations (4) are:

$$\begin{aligned} (Ls+R)I(s) &= V - K_b s\theta(s) \\ s(Js+B)\theta(s) &= KI(s) + T_l \end{aligned}. \tag{5}$$

For zero T_l we can eliminate $I(s)$ to get the following transfer function for $\omega = \frac{d\theta}{dt}$:

$$\frac{\omega}{V} = \frac{K}{(Js+B)(Ls+R)+K_bK} = \frac{\frac{K}{JL}}{s^2 + \frac{BL+JR}{JL}s + \frac{BR+K_bK}{JL}}. \tag{6}$$

The parameters used [17] in our simulation are shown in Table 6.1:

Table 6.1. Parameters of the DC motor used in the simulation.

	Parameters	Values
K	Torque constant	2.55e-3 N-m/A
R	Armature winding resistance	6.43 Ω
K_b	Back EMF constant	0.255e-3 V-sec/rad
L	Armature winding inductance	28.8e-3 H
B	Damping coefficient	0.1e-3 N-m-sec/rad
J	Moment of inertia	3.53e-6 Kg-m^2

Substituting the parameters in Table I into equation (6), the obtained transfer function becomes

$$\frac{\omega}{V} = \frac{25080}{s^2 + 251.6s + 6331}. \tag{7}$$

When the DC motor is controlled by the linear PID controller, the gain is tuned with the following restrictions and specifications: percentage overshoot is less than 5%, settling time is less than 0.0321 sec and rise time is less than 0.014 second. The PID controller satisfying the above constraints is found to have (K_P, K_I K_D) = (0.995, 25.572, 0) [18-19]. Thus the controller becomes a PI controller.

An ANN controller is designed by cascading the nonlinear function shown in Figure 6.3 with the tuned PI controller.

B. Disturbance Rejection Capability and Sensitivity to Noise

A closed-loop controller with disturbance D is shown in Figure 6.6.

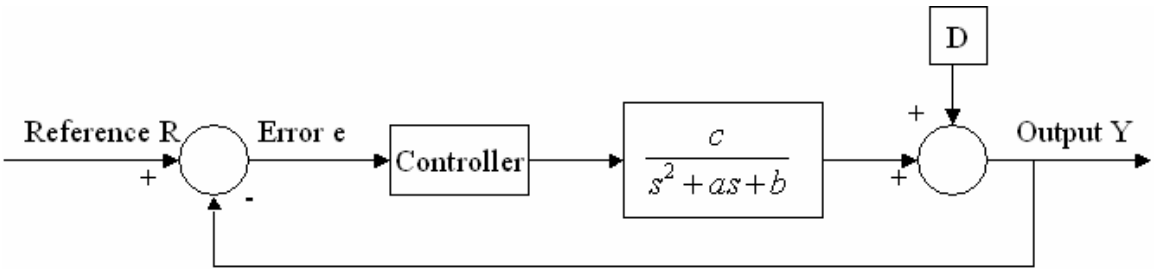


Figure 6.6. Closed-loop control with disturbance.

To analyze the disturbance rejection capabilities of the ANN and PI controller, we inject a series of step inputs as disturbance signals as shown in Figure 6.7.

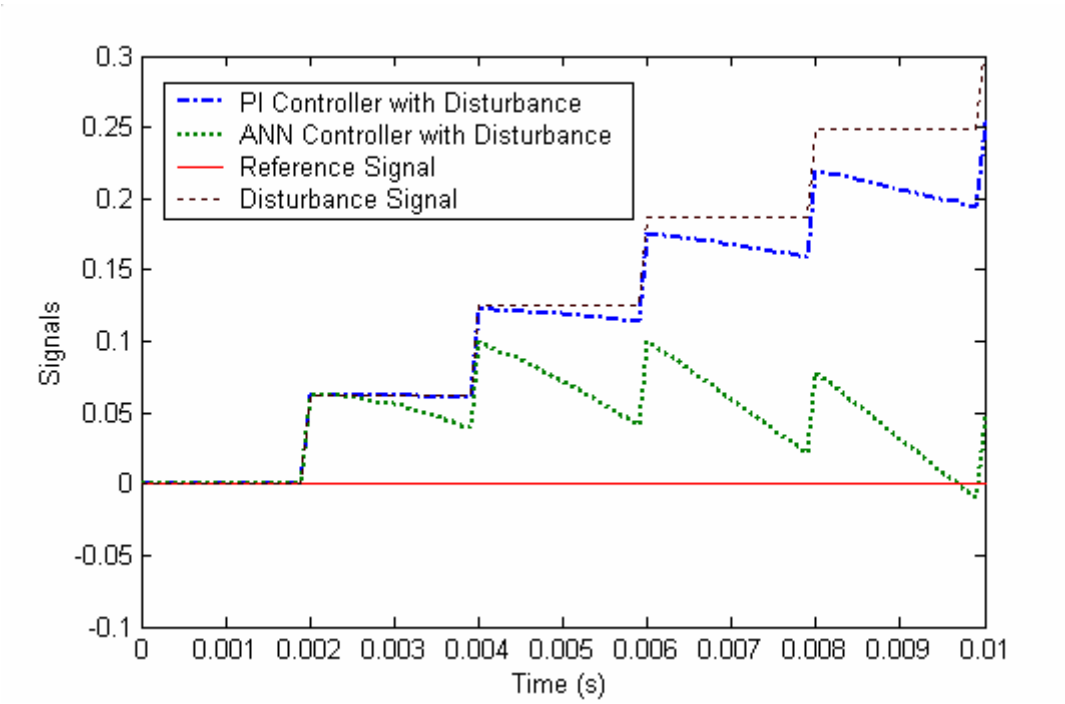


Figure 6.7(a) Initial responses to 4 consecutive step inputs.

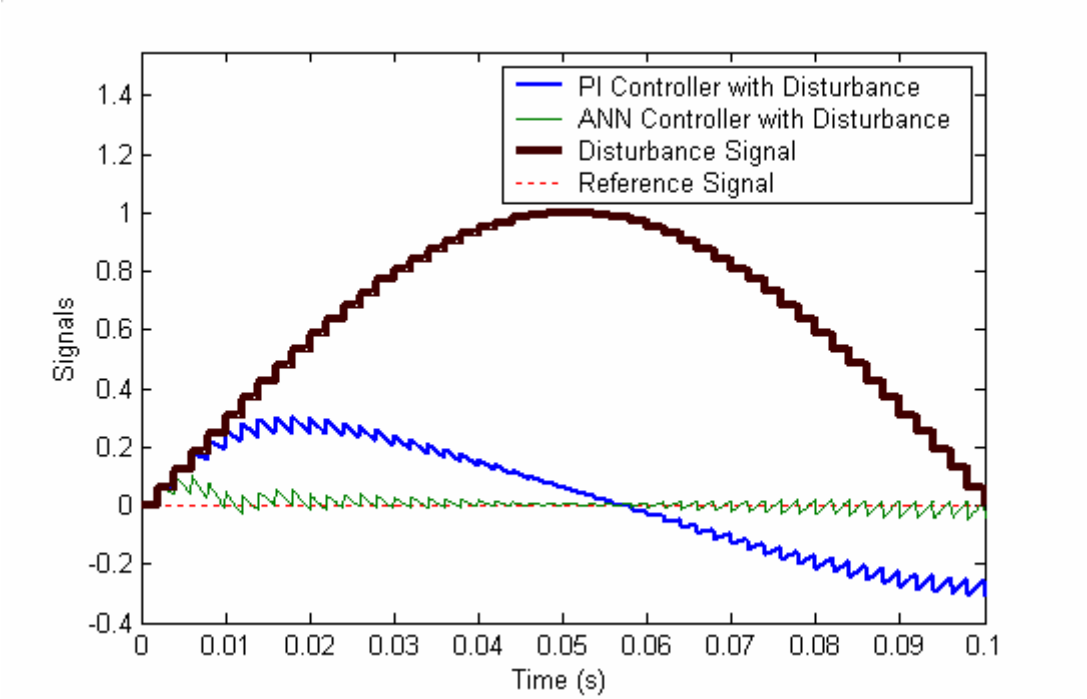


Figure 6.7(b). Overall responses to multi consecutive step inputs.

Figure 6.7. Disturbance rejection performance of the PI controller and the ANN controller.

As mentioned before, the gain of the ANN is relatively large for small error thus it enjoys faster response for smaller error compared to the fixed-gain linear PI controller.

For the series of step inputs comprising disturbance signal as shown in Figure 6.7, the PI controller exhibits more severe error buildup. In the Figure 6.7(a), four step inputs are injected into the system at time (in seconds) t = 0.002, 0.004, 0.006 and 0.008 as the disturbance signals. It is shown that immediately after the first disturbance at $t_1 = 0.002$s, both system start to recover towards these reference signals. Before the disturbance is fully rejected by either controller, the second step input is injected at time $t_2 = 0.004$s. During the 2 ms time interval t_2-t_1, the output of the ANN controller decreases more because of its larger gain. The responses to the second step input are superimposed on the

responses to the first step input. The PI controller suffers more since its initial recovery is slower than the ANN controller. Such arguments with series step inputs can be extended to draw similar conclusions for continuously changing disturbance signals and noise signals.

Disturbance rejection capabilities can be further compared by injecting sine wave disturbance signal. Figure 6.8 shows the step responses of both controllers when there is no disturbance. Figures 6.9 – 6.12 depict the step responses of both controllers under various disturbances. It can be seen that each of these disturbances has more influences on the PI controller's performance than it does on the ANN controller.

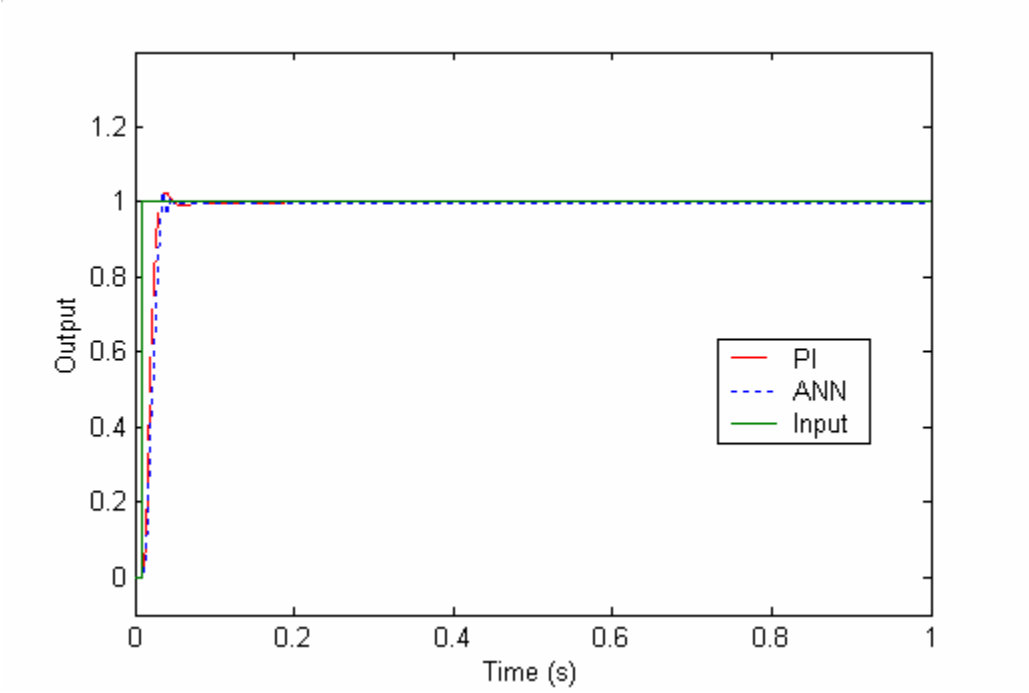


Figure 6.8. Step response comparison of ANN and PI controllers without disturbances.

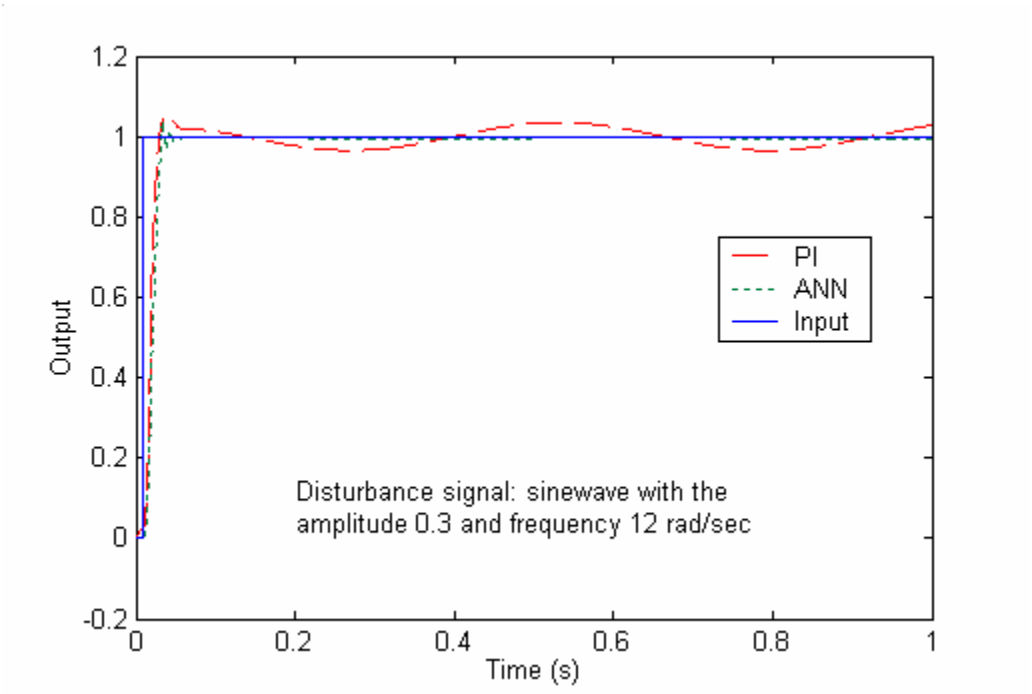


Figure 6.9. Step response comparison of ANN and PI controllers with small disturbances.

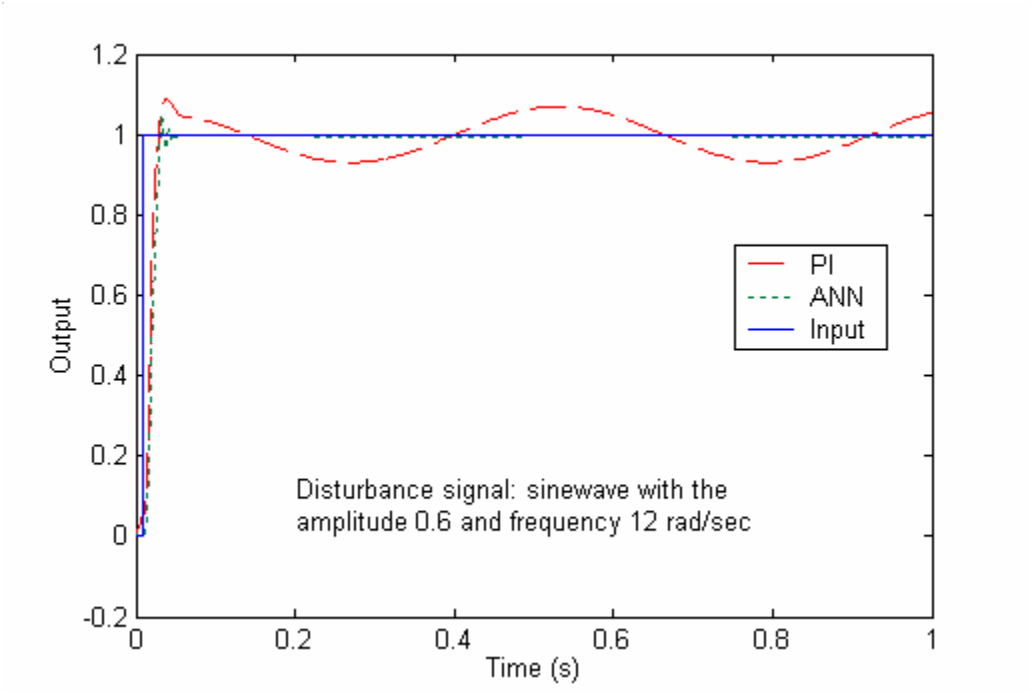


Figure 6.10. Step response comparison of ANN and PI controllers with moderate disturbances.

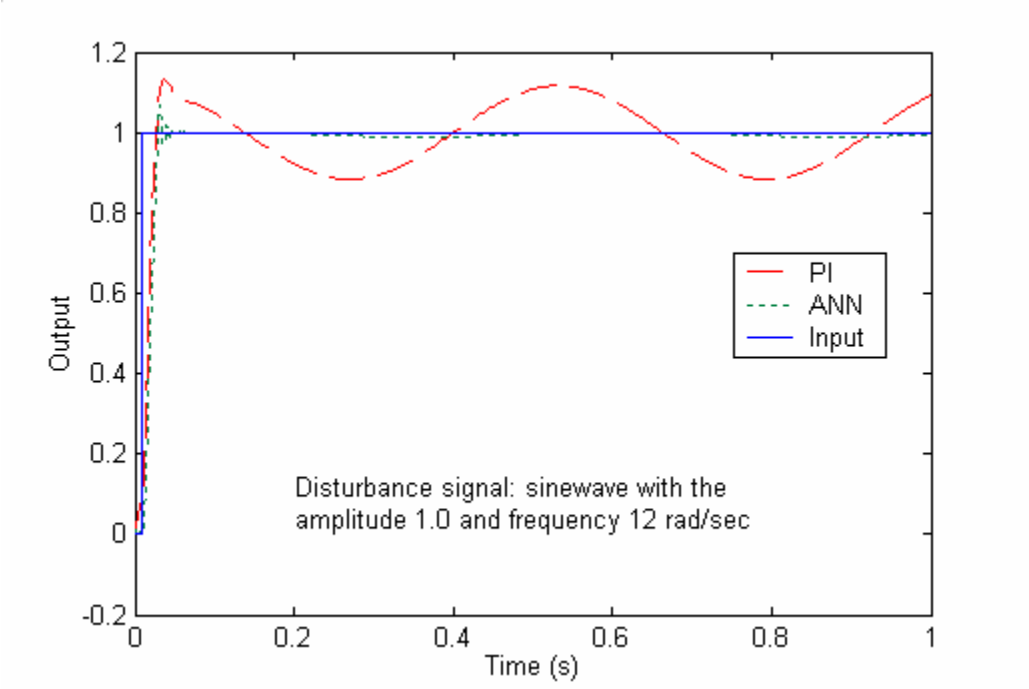


Figure 6.11. Step response comparison of ANN and PI controllers with large disturbances.

The ANN controller demonstrates higher gain for smaller error relative to the linear PI controller and noise can be regarded as small disturbances which is better rejected by large gains. Gaussian white noise is injected into the control loop as disturbance signal as shown in Figure 6.12. The noise chosen in the simulation is Gaussian white noise with mean μ and standard deviation σ.

In our simulation, $\mu = 0$ and $\sigma^2 = 0.001$, 0.002, 0.005, 0.008, 0.010 and 0.012. Noises with these different standard deviations are plotted in Figure 6.12.

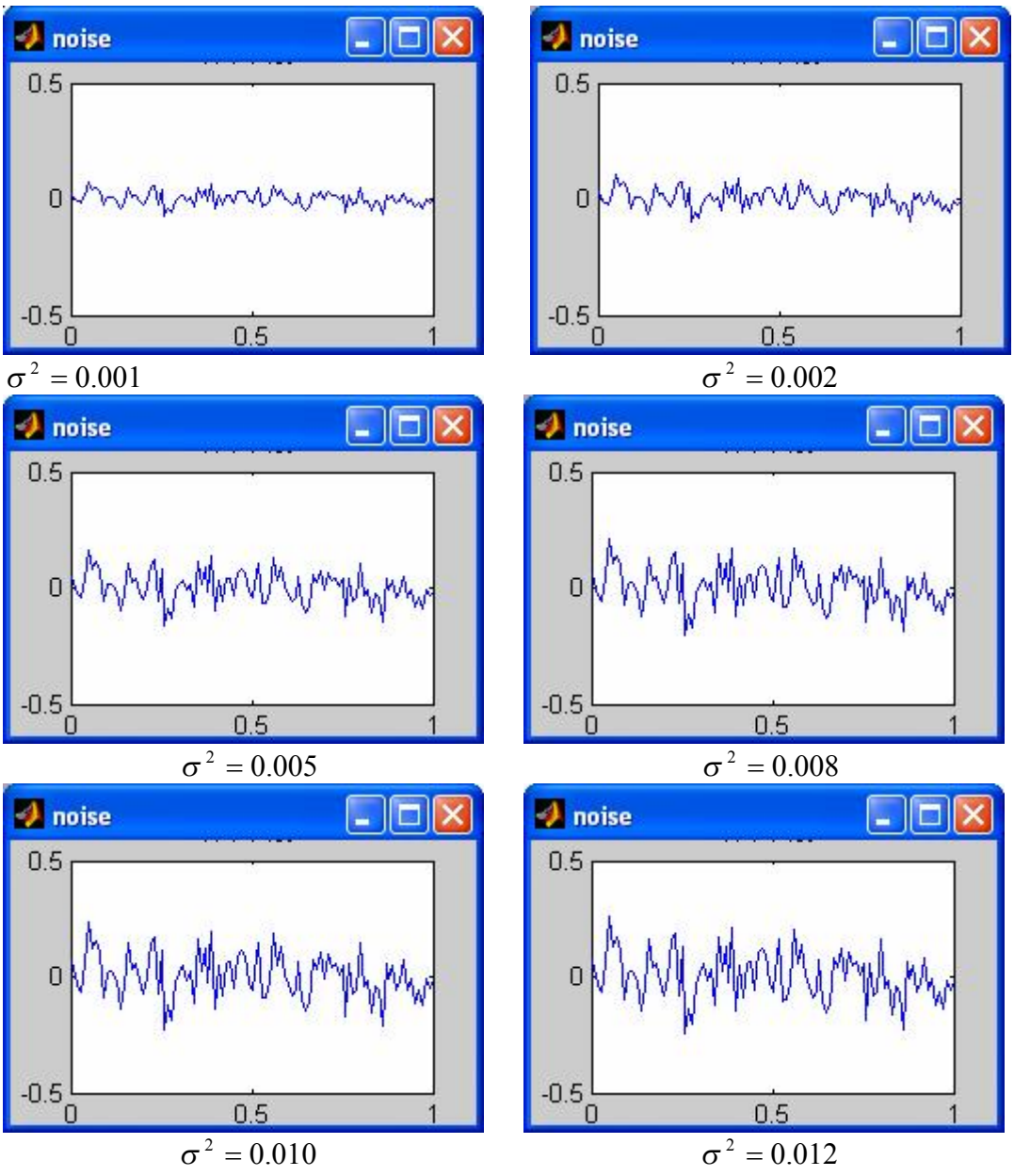


Figure 6.12. Gaussian white noise with different noise variances

Since $k = \dfrac{c \times \tanh(q \times e)}{e} \cong \dfrac{c \times (q \times e)}{e} = cq$ for small value of e, k increases as q increases which is shown in Figure 6.13. As a result, the noise rejection capability of the ANN controller increases with q, as shown in Figure 6.14.

On average, the ANN controller with q = 100 has a noise response 35.93% smaller than the PID controller.

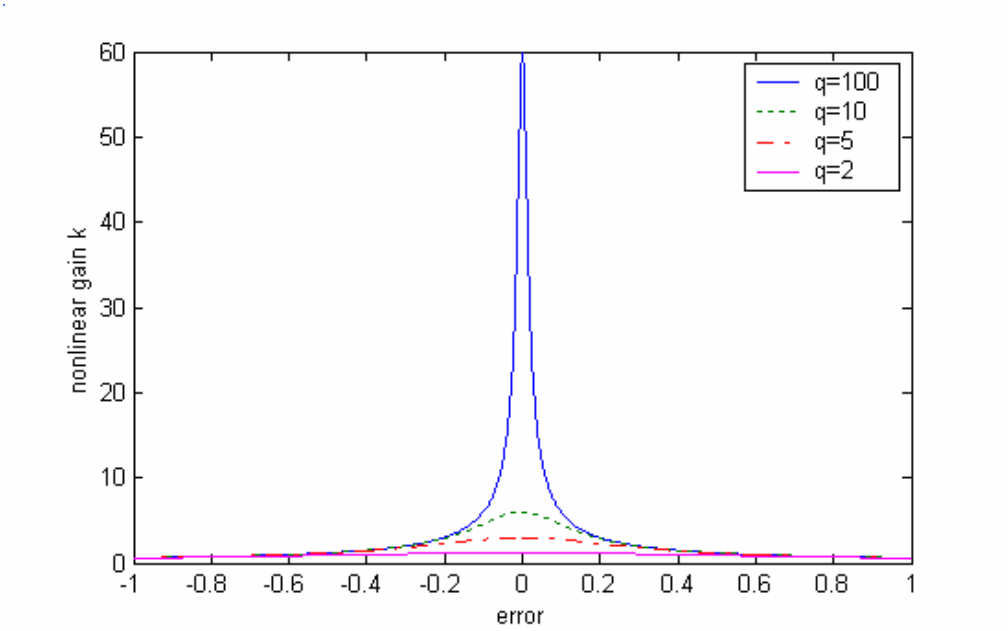


Figure 6.13. Variations of gain k with different parameters.

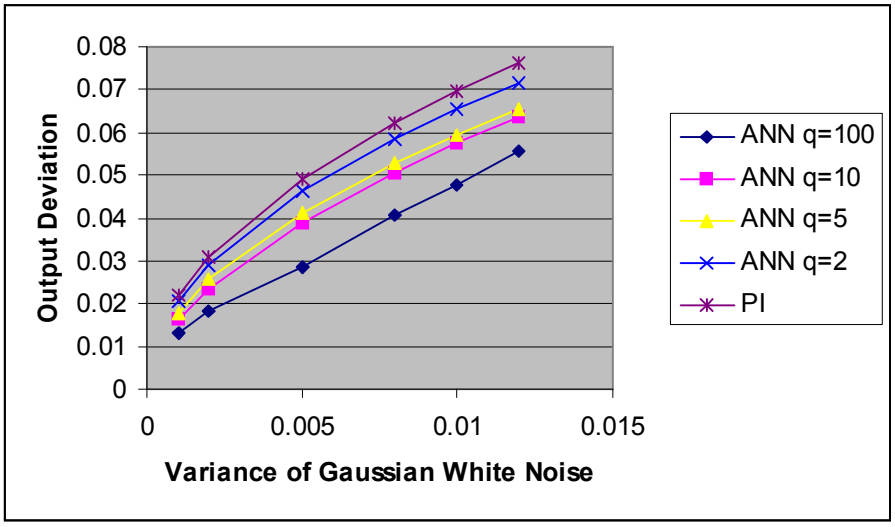


Figure 6.14. Analysis of controllers' sensitivity to noise.

C. Stability Region Estimation for the ANN Controller

Following Seraji [22], the stability of the ANN controller can be analyzed using the Popov criteria [23]. The ANN controller includes linear PI controller component in cascade with the nonlinear gain k. The transfer function of the linear PI controller is given by

$$K(s) = K_P + \frac{K_I}{s}, \tag{8}$$

and the plant transfer function is given by

$$G(s) = \frac{c}{s^2 + as + b}. \tag{9}$$

let the allowable range of nonlinear gain k to be (0, k_{max}), According to Seraji's result, $k_{max} \to \infty$ when $K_I \le a K_P$ and

$$k_{max} = \frac{ab}{(K_I - aK_P)c} \tag{10}$$

when $K_I > a K_P$.

IV. Experimental Setup and Results

A. The Controller

Artificial neural networks are parallel information processing structures inspired by biological neural networks. Software and hardware represent two valid approaches for implementing ANNs, but software instructions which are executed serially cannot take advantage of the inherent parallelism of ANN architectures. Hardware implementations

of neural networks promise higher speed operation when they can exploit this massive parallelism.

B. Hardware Setup for Experiment

The unmanned ground vehicle, as shown in Figure 6.15, used in the Advanced Diagnosis, Automation, and Control (ADAC) lab of North Carolina State University is a UGV which was previously controlled by digital microcontrollers. In addition to the control, the UGV has optical sensors, an H-bridge circuit for driving DC motors, two DC motors and the power supply.

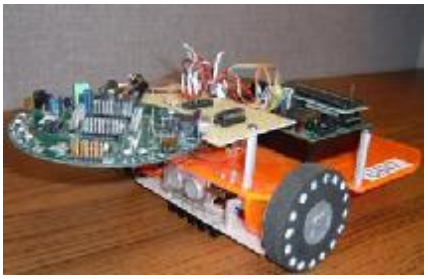


Figure 6.15. UGV used in ADAC lab at North Carolina State University.

The experiment is carried out to test the disturbance rejection capabilities of the ANN controller. A path-tracking UGV is controlled by a FPAA based ANN to run on a surface that has been inclined relative to horizon by lifting one edge of the track board off the ground. The UGV must track the path as close as possible and to finish the round trip in shortest possible time. The UGV and the path to track are shown in Figure 6.16.

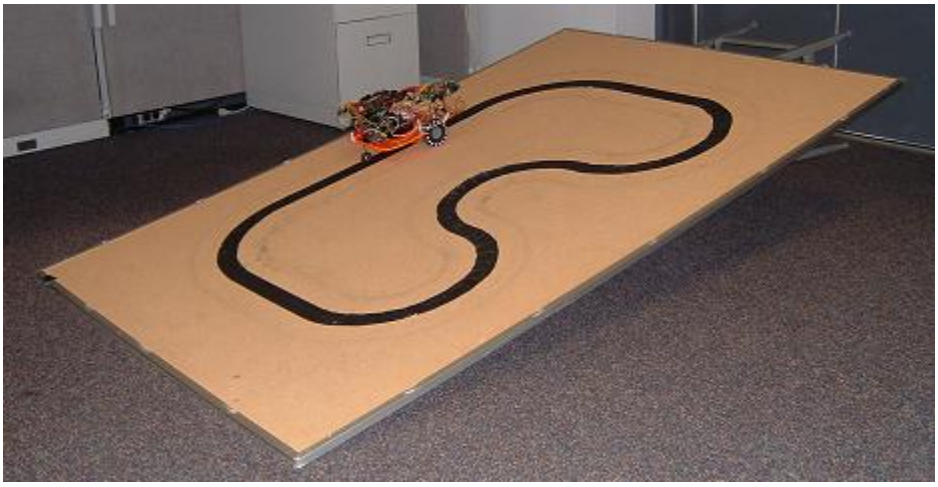


Figure 6.16. The path-tracking UGV and the track.

As the UGV runs up and down the surface, a varying load disturbance is continuously applied to the DC motors spinning the wheels. A PI controller programmed with FPAA is also tested with same experimental conditions for comparison. In closed-loop control, either the ANN or PI controller accepts optic sensor signals from the UGV as inputs and output the control signals to the H-bridge circuit which drives the DC motors as reported in [5]. The system overview is shown in Figure 6.17. The PWM signals driving the DC motors are generated by the FPAA circuit. Performance evaluation criteria for UGV path-tracking can be found in [5]. The round trip time and error rate are two important measures of the performance: good performance is generally associated with small error rate and short round-trip time.

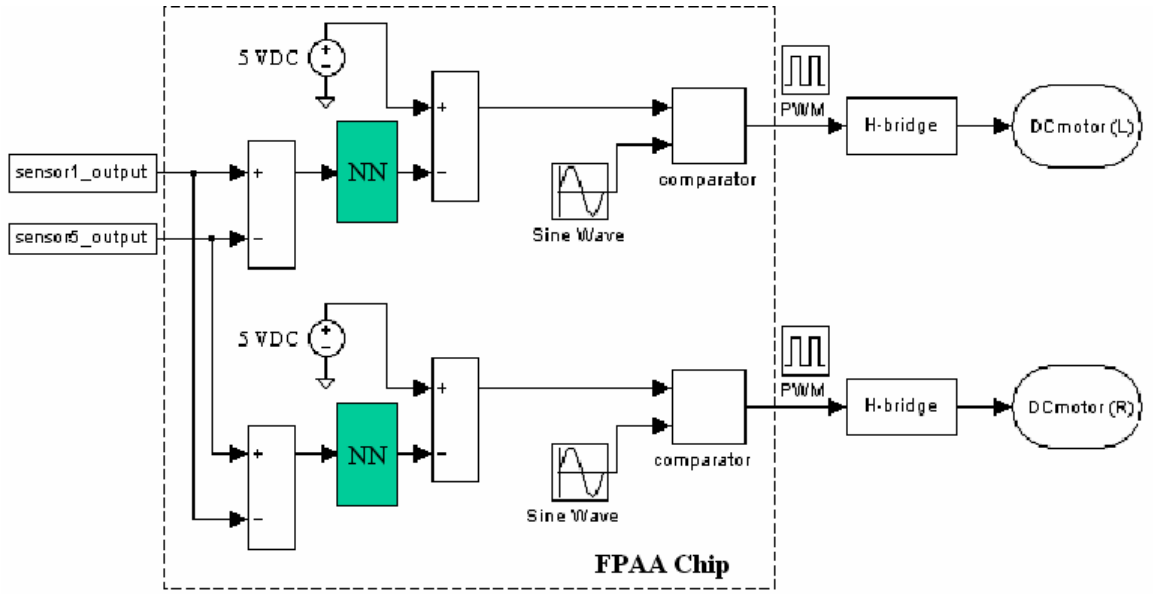


Figure 6.17. System overview of the path-tracking UGV controlled by FPAA based ANN.

C. Experimental Results

As shown in Figure 6.18, the varying load disturbance introduced by the inclined track has more influence on the PI controller since its round trip time increases substantially with the increase of the edge height. As the edge height increases from 0 to 45 centimeters the round trip time increases by 6.41% for the PI controller. For the ANN controller the round trip time is roughly constant with less than 0.1% variation from the average of 40.589 seconds. Regarding error rate, the ANN controller also outperforms the PI controller as shown in Figure 6.19. The error rate increases 15.38% with edge height increase for the PI controller. The ANN controller has less than 1.07% variation around the average of 5.028 error/sec. As a result, we conclude that the ANN controller exhibits better disturbance rejection than the PI controller does under same experimental conditions.

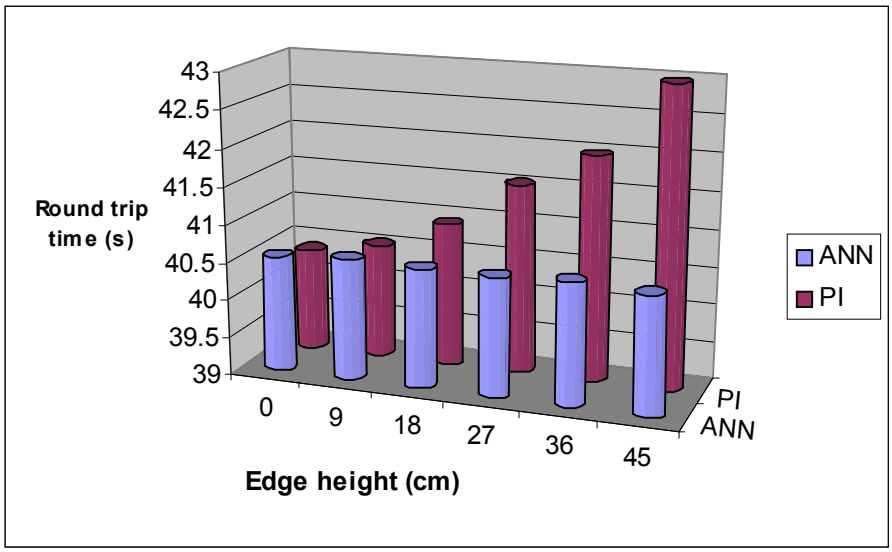


Figure 6.18. UGV round trip time comparison.

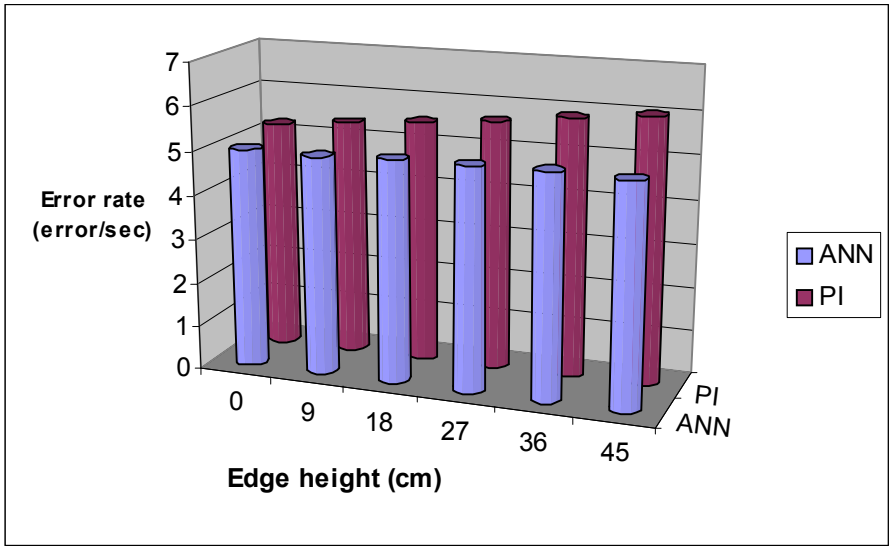


Figure 6.19. Comparison of error rate for UGV path tracking.

V. Conclusion

An ANN controller with improved disturbance rejection capability and immunity to noise is presented. Its performance is evaluated by comparison with a linear PI controller. The ANN is equivalent to a nonlinear PID controller which adapts its gain to make it

more sensitive to small error. Simulation shows that the ANN controller has better capabilities of disturbance rejection and stronger immunity to noise. Stability analysis is addressed for the ANN controller. Experimental evaluations are also carried out. A FPAA based ANN controller is designed to control a path tracking UGV running on an inclined surface, resulting in a load disturbance applied to the DC motors. Experimental results indicate that same load disturbance has more influence on the PI controller's performance than it does on the ANN controller; thus we can conclude the ANN controller has superior performance to the conventional linear PI controller for disturbance rejection. The experiments also employ a whole new media for ANN controller for robotics applications – commercial FPAA technology. With its ease of implementation, low cost and high speed, FPAA's potential advantage for real-time control applications is considerable.

VI. References

[1] Koh, K. & Cho, H. "A smooth path tracking algorithm for wheeled mobile robots with dynamic constraints", *Journal of Intelligent and Robotic Systems: Theory and Applications* 24(4), pp.367-385, 1999.

[2] Y. Tipsuwan, and M.-Y. Chow, "Gain scheduler middleware: A methodology to enable existing controllers for networked control and teleoperation: PART I: Networked control," *IEEE Transactions on Industrial Electronics*, December, 2004.

[3] Y. Tipsuwan, and M.-Y. Chow, "Gain scheduler middleware: A methodology to enable existing controllers for networked control and teleoperation: PART II: teleoperations," *IEEE Transactions on Industrial Electronics*, December, 2004.

[4] Y. J. Kanayama and F. Fahroo, "A new line tracking method for nonholonomic vehicles," *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on* , Vol. 4, pp. 2908-2913, 20-25 April 1997.

[5] P. Dong, G. Bilbro, and M.-Y. Chow, "Controlling a Path-tracking Unmanned Ground Vehicle with a Field-Programmable Analog Array," *2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Monterey, CA, 24-28 July, 2005.

[6] S. Baluja, "Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller," *IEEE Transactions on Systems, Man and Cybernetics*, Part B, Vol. 26, No. 3, pp. 450-463, June, 1996.

[7] J. Rosenblatt, "DAMN: A Distributed Architecture for Mobile Navigation," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9, No. 2/3, pp. 339-360, 1997

[8] R. Fierro and F. L. Lewis, " Control of a Nonholonomic Mobile Robot Using Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 9, No. 4, 1998.

[9] Y. B. Veryha and L. Kourtch, "Neural Network Based Manipulator Control with Time-Optimal Robot Path Tracking," *IEEE International Conference on Control Applications*, Anchorage, Alaska, USA, September 25-27, 2000.

[10] http://www.anadigm.com

[11] J. Taylor and K. Strobel, "Nonlinear compensator systhesis via sinusoidal-input describing functions," *in Proceedings of American Control Conference*, Boston: AACC, 1985, pp. 1242–1247.

[12] J. H. Taylor and K. J. Astrom, "Nonlinear PID Autotuning Algorithm," *in Proceedings of American Control Conference*, Seattle, WA, pp. 2118–2123, 1986.

[13] Y. Xu, J. M. Hollerbach, and D. Ma, "A Nonlinear PD Controller for Force and Contact Transient Control," *IEEE Control System Magazine*, vol. 15, no. 1, pp. 15–21, 1995.

[14] B. Armstrong, D. Neevel, and T. Kusik, "New results in NPID control: Tracking, Integral Control, Friction Compensation and Experimental Results," *in Proceeding of International Conference of Robotics and Automation*, pp. 837–842, 1999.

[15] B. Armstrong, D. Neevel, and T. Kusik, "New results in NPID control: Tracking, integral control, friction compensation and experimental results," *Control Systems Technology, IEEE Transactions on*, vol. 9, pp. 399-406, 2001.

[16] H. Huang , “Nonlinear PID Controller and its Applications in Power Plants,” *International Conference on Power System Technology*, Vol. 3, pp. 1513-1517, Oct. 2002.

[17] Y. Tipsuwan, M.-Y. Chow, “Gain adaptation of mobile robot for compensating QoS deterioration”, *Proceedings of IECon’02*, Sevilla, Spain, 2002.

[18] Tyler Richards, Mo-Yuen Chow, "Performance Characterization of IP Network-based Control Methodologies for DC Motor Applications - Part I", *Proceedings of IECON05, The 31st Annual Conference of the IEEE Industrial Electronics Society*, Raleigh, NC, Nov. 6-10, 2005, pp. 2405 - 2410.

[19] Tyler Richards, Mo-Yuen Chow, Fen Wu, "Performance Characterization of IP Network-based Control Methodologies for DC Motor Applications – Part II", *Proceedings of IECON05, The 31st Annual Conference of the IEEE Industrial Electronics Society*, Raleigh, NC, Nov. 6-10, 2005, pp. 2411 -2416.

[20] B. C. Kuo, “Automatic Control Systems,” 7th edition, *Pretence Hall, Inc*. ISBN 0-13-304759-8, 1995.

[21] M. Sekerli and R. J. Butera, "An implementation of a simple neuron model in field programmable analog arrays," *presented at Engineering in Medicine and Biology Society, 2004. EMBC 2004. Conference Proceedings. 26th Annual International Conference of the*, 2004.

[22] H. Seraji, “NPID Controllers with Robotic Applications*,” Journal of Robotic Systems* 15(3), pp. 161-181, 1998.

[23] S. M. Shinners, “Advanced Modern Control System Theory And Design,” *John Wiley & Sons, Inc.* ISBN 0-471-31857-4, 1998.

[24] H. Shu and X. Guo, “Decoupling control of multivariable time-varying systems based on PID neural network,” 5th Asian Control Conference, vol. 1, pp. 682-685, July, 2004.

[25] W. Bolton, “Mechatronics, Electronic Control Systems in Mechanical and Electrical Engineering,” *Pretence Hall*, ISBN 0-582-35705-5, 1999.