A Fully Parallel Learning Neural Network Chip for Real-time Control

Jin Liu and Martin Brooke
Georgia Institute of Technology
791 Atlantic Drive, MiRC, Atlanta, GA 30332, USA
Emails: jliu@ee.gatech.edu and martin.brooke@ee.gatech.edu

Abstract

Presented in this paper is a parallel learning neural network chip, which is used to perform real-time output feedback control on a nonlinear dynamic plant. The controlled plant is a simulated unstable combustion process. Neural networks provide an adaptive sub-optimal control that does not need any prior knowledge of the system. In addition, the hardware neural network presented here utilizes parallelism to achieve speed independent of the size of the network, enabling real-time control. On-chip learning ability allows the hardware neural network to learn on-line as the plant is running and the plant parameters are changing. Also described is the experimental setup used to obtain the results.

1. Background Review

The goal of our research is to implement a learning neural network chip to control real-time systems. Real-time systems require that the processing occur within a short period. The real-time system considered here is the combustion in jet or rocket engines, for which the plant time constants are associated with the combustion process and the fuel injection delay. Because of this, neural networks operating in a few milliseconds are required.

Neural networks have been successfully applied in many areas, from engineering to economics, from forecasting to control. In the control area, they have been applied to control robot arms, chemical processes, continuous productions of high-quality parts, and aerospace applications [1]. Previous research [2] [3] [4] showed that neural networks could be used to model and control complex nonlinear physical systems with unknown or slowly varying plant parameters.

Our research is part of the Multidisciplinary University Research Initiative (MURI) on Intelligent Turbine Engines (MITE) project, supported by DOD-Army Research Office, under Grant No. DAAH04-96-1-0008.

A neural network controller can be a general-purpose controller. The same neural network can be reconfigured for dissimilar applications. In conventional control methods, an extensive study and understanding of a system is required to build a controller, especially for that system. A neural network can control a system without previous knowledge of the system, and an on-line learning neural network can adjust on-line to unexpected conditions, which is very difficult with traditional control methods.

Another potential advantage of the neural network is its parallelism. Human neural systems solve complicated problems with parallel operations of many neurons. The parallel operations make it possible to solve a complex problem in a short time period, compared with serial operations. However, reported implementations of neural networks do not all exploit parallelism fully.

Originally, most neural networks are implemented by software running on computers. However, as neural networks gain wider acceptance in a greater variety of applications, it appears that many practical applications require high computational power to deal with the complexity or real-time constraints [9]. Software simulations on serial computers can not provide the computational power required, since they transform the parallel neural network operations into serial operations. When the networks get bigger, the software simulation time increases accordingly. With multiprocessor computers, the number of processors typically available does not compare with the full parallelism of hundreds, thousands, or millions of neurons in most neural networks. In addition, software simulations are run on computers, which are usually expensive and can not always be affordable.

As a solution to the above problems raised by software simulation, dedicated hardware is purposely designed and manufactured to offer a higher level of parallelism and speed. It can potentially provide high computational power at a limited cost.

A common principle for all hardware implementations is their simplicity. Mathematical operations that are easy to implement in software might often be very burdensome in the hardware and therefore more costly. Hardware-friendly algorithms are essential to ensure the functionality and cost effectiveness of the hardware implementation.

In this research, a fully parallel learning analog neural network chip that implements a hardware-friendly algorithm, called random weight change (RWC), is used to successfully control a simulated dynamic, nonlinear real-time system.

2. Proposed Real-time Control Scheme with Neural Network

A general direct feedback scheme [5], as depicted in Figure 1, is proposed for real-time nonlinear system control, using learning neural network chips.

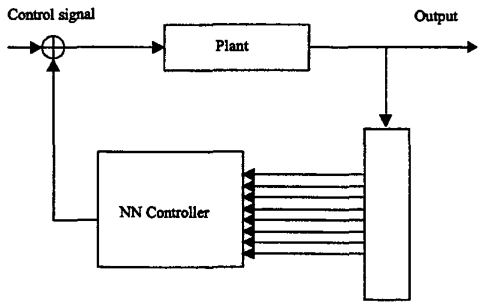

Figure 1: A Neural Network Controller

In Figure 1, the plant is a dynamic nonlinear system. Unlike static systems, whose inputs and outputs are uniquely mapped, the outputs of dynamic nonlinear systems depend on internal states, system parameters, and inputs. Moreover, for nonlinear systems, their system parameters vary with the time and operating conditions. The tapped delay line is used to sample the plant output history; it should be long enough for the neural network to infer the state of the plant. In general, a period of the plant output of the slowest signal frequency is to be covered. At the same time, the tap delay line should be faster than the Nyquist sample frequency of the highest signal frequency. The rule is that the neural network should be provided enough information about the plant dynamics.

3. Issues for On-chip Neural Network Implementation

There are two general categories of hardware neural networks: digital hardware and analog hardware.

Digital hardware has the advantage of implementing precise computation units like adders, shifters, and multipliers. Analog hardware experiences circuit nonidealities, like nonlinear multipliers and the weight leakage. On the other hand, the analog circuit is much more compact than the digital circuit. Because of the size difference, most analog neural chips are implemented by a parallel matrix of neural cells, while most digital neural chips need to share computation units.

The degree of parallelism for the digital hardware varies with different implementations. One category of the digital chips is general-purpose digital processors (i.e., DSP chips) and digital logic arrays (i.e., FPGAs). They are the same circuits as in a computer. However, they are more cost effective and faster than computers. For this category of hardware, the degree of parallelism is either serial, or partially parallel. Another category of digital chips is dedicated chips, designed to implement neural networks. They are similar to analog chips, but use digital circuits instead of analog circuits. They can be made fully parallel. However, since digital computation units are much larger than analog units, a fully parallel digital chip would be very large [6] [7] [8], thus very expensive.

On the other hand, a fully parallel neural network of useful size can be implemented on a single integrated chip using analog circuits. The analog implementation is a more efficient implementation compared with the digital implementation. As mentioned before, there are difficulties for the analog implementation: the need for hardware-friendly learning algorithm with full parallelism, the added hardware on-chip learning ability, and the difficulty with long-term storage, etc. If the analog implementation can overcome these design difficulties, it can provide fully parallel neural networks, which would offer a higher degree of the performance than digital systems.

Full parallelism is desirable, but its advantage is obvious only when the neural network size is large. When the neural network size is small, there may not be a big difference among software networks, partially parallel hardware, and fully parallel hardware. However, when the size of the network increases by several orders of magnitude, the difference in speed between the fully parallel hardware and other networks can increase proportionally.

The major drawback of all hardware implementations is that they must often be designed for a specific application. Therefore, their use can be justified only for either very large quantities or very high-performance requirements [9]. General-purpose digital hardware has an advantage in this perspective. It is available in the market, and only software configurations are needed. In addition, digital hardware is generally easier to design than analog hardware. Analog hardware is more costly to design. However, if the analog design overcomes the design difficulties, and is in mass production, the cost would be much smaller. As mentioned before, neural network controllers can be general-purpose controllers. Once a good design is ready, it can be reconfigured with minor effort for dissimilar applications.

In summary, there are advantages and disadvantages for both digital and analog hardware. When the computation power is not high, it is cost efficient to use software and digital solutions. Only when high computation power is required is the analog circuit preferable, given a good analog implementation.

There are mainly three issues for building parallel on-chip learning hardware neural networks: the chip architecture, the activation function circuit, and the synapse circuit.

We choose the feed forward architecture for the network. The algorithms associated with this type of architecture are the well-known back propagation algorithm (BP), the chain perturbation rule etc. However, BP algorithm requires that the output of the neural network be known and so cannot be directly used in the neural network control scheme proposed above. In addition, the BP algorithm has proven sensitive to analog circuit non-idealities, thus it is not a good choice for this work.

Learning rules like serial-weight-perturbation [10], or Madaline Rule III (MRIII) are very tolerant of the analog circuit non-idealities, but as they are slow serial computation algorithms, the speed advantage of the parallel hardware is lost. The Chain Perturbation Rule [11] is based on the serial-weight-perturbation learning rule. It is tolerant for nonlinear analog multipliers, but it uses partially parallel learning, still is not a fully parallel algorithm.

The random-weight-change algorithm [12] is more suitable for the industrial control applications. It is tolerant to sever analog circuit nonidealites. The error to be reduced is not specified in the algorithm, instead it can be any error. For the control scheme shown in Figure 1, the error can be at the plant output as desired. The disadvantage of the random-weight-change algorithm is that it takes longer to learn than the back propagation algorithm.

A linear current to voltage converter with saturation is used to implement the activation function [13] [14].

There are five kinds of synapse circuits currently used. They are categorized by storage type. They are capacitor only [14] [15] [16] [17] [18] [19], capacitor with EEPROM [11], capacitor with refreshment [20] [21] [22], digital [23] [24], and mixed digital/analog circuits [25].

Digital weight circuits are large, requiring 16-bit precision to implement BP learning. EEPROM weights are very compact nonvolatile memory, but are process sensitive, and extra compensation circuits are required. Capacitors with weight refreshment need off chip memory, which is not a compact solution, and the added A/D and D/A circuits either make the chip large or result in serial operation. Capacitor weights are easy to use, and compact. However, they have leakage problems. To prevent leakage, the capacitor may have to be large, a 20pF capacitor is around the same size as an eight-bit digital weight. However, for the application for which leakage is not a problem, capacitor weights are a good solution. The mixed digital/analog type weight storage is a balanced solution when nonvolatile memory is necessary.

The random weight chip described here uses capacitor weight storage. Potentially, there will be a leakage problem, and a nonvolatile weight may be desirable eventually. However, for the current research, the controlled plant is a dynamic system, which changes fast with time. The weights continue to adjust on-line, so that the leakage is compensated for and is not a problem.

4. The Random Weight Change Algorithm

The Random Weight Change (RWC) algorithm is defined as follows for weights w_i

$$\begin{aligned}
&w_i(n+1) = w_i(n) + \Delta w_i(n+1); \\
&\text{if error decrease,} \qquad \Delta w_i(n+1) = \Delta w_i(n); \\
&\text{if error increase,} \qquad \Delta w_i = \text{rand}(n); \\
&\text{rand}(n) = +\delta \text{ or } -\delta \text{ with equal opportunities, and} \\
&\delta = \text{a small enough quantity,}
\end{aligned}$$

where $w_i(n)$ and $\Delta w_i(n)$ are the weight and weight change of i^{th} neuron, at the n^{th} iteration respectively.

Software simulations using modified RWC algorithm to identify and control an inductor motor [26] was successful. Further research proved that the RWC algorithm is immune to the analog circuit non-linearity [27].

5. Software Simulation of Combustion Instability

The dynamic nonlinear system we seek to control is combustion instability. When no control is applied, this system is unstable and eventually reaches a bounded oscillation state. The goal of our control is to suppress the oscillation.

There are several well-known passive approaches for reducing the instabilities [28]. However, the implementation cost of such approaches is high and the design process time consuming, and they often fail to adequately damp the instability. The effort of developing active control systems for damping such instabilities has increased in recent years. However, since the combustion system is a nonlinear system, the system parameters vary with time and operating conditions. Thus active controllers that are developed to suppress the oscillation in fixed modes cannot deal with unpredicted new oscillation modes. Also the actuation delay present in the control loop also causes difficulties for the control.

We have carried out extensive combustion instability control with the random-weight-change algorithm. The models used are the simple oscillators and limit cycle oscillators. The simulation performed includes single frequency, multi frequency, continuously changing system parameters, and added noise. They are all successful.

6. Neural Network Hardware Control of Combustion Instability

The chip layout is shown in Figure 2. It consists of 10x10 arrays of weight cells. The schematic of a weight cell is shown in Figure 3. The left part is a shift register for shifting in random numbers. The right part is a simple multiplier. The circuits in the middle are the weight storage and weight modification circuits. The weight is stored in the capacitor, and it is changed by charge sharing.

To control the computer simulated combustion plant with the real neural network chip, the test setup shown in Figure 4 was used. The computer runs a program that simulates the unstable limited-cycle engine. The RWC and the computer are connected by two data acquisition cards to form a closed loop. The engine output generated by differential equation models is tap-delayed and sent to the input of the chip through the analog output card (AT-AO-10). These inputs are forward propagated through the neural network chip, and an output is generated. The weights are adjusted according to the random weight change algorithm in order to minimize the magnitude of the engine output. The chip output is current, it is transformed to voltage and read in to the computer by the analog input card (AT-MIO-16E).

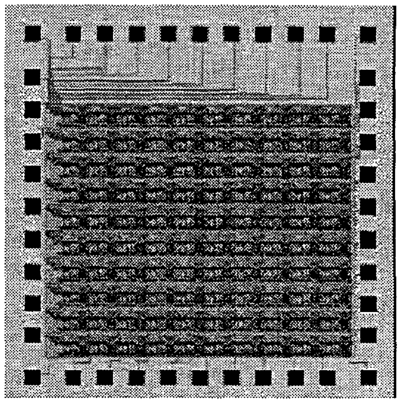
Figure 2: Chip Layout

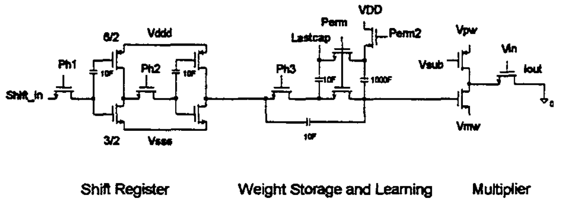

Figure 3: Weight Cell Schematics

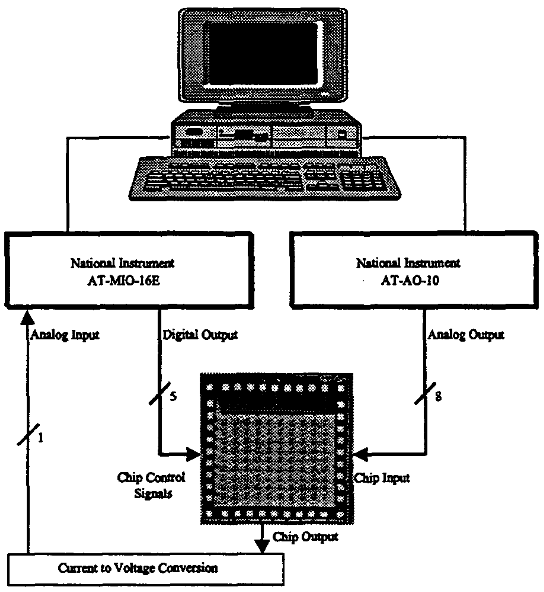

Figure 4: Test Setup

The result of the neural network chip controlling computer simulated combustion instability is shown in Figure 5. The run time is 10 seconds. The first plot of the results is the engine output under the control of the RWC chip. The second plot of the results is the error signal used to control the learning. A '1' means the error is decreasing, thus the weights should keep the last good change, a '0' means the error is increasing, thus the weights should change to another random number. The plots show that when the chip finds a good set of weight changes, it keeps on using them to decrease the magnitude of the engine output. It happens when the error signal remains at '1' for a period of time, shown by the big gaps in the error signal plot. Otherwise, the chip randomly searches for the right weight change, the error sometimes decreases, sometimes increases. This happens when the error signal frequently oscillates between '0' and '1', indicated by the dark blocks in the error signal plot.

Figure 6 shows the details of the initial learning process and Figure 7 shows more details of the continuously adjusting process.

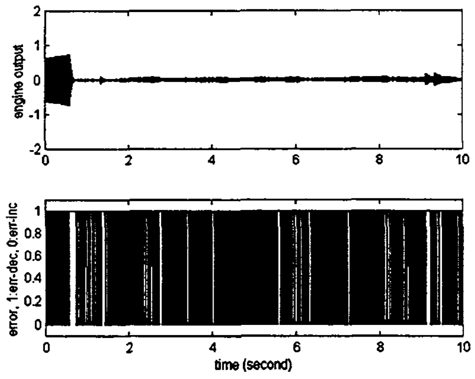

Figure 5: Hardware Neural Network Controlling Computer Simulated Combustion Instability

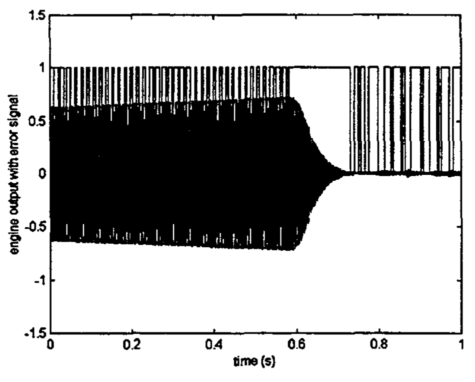

Figure 6: Details of the Initial Learning Process

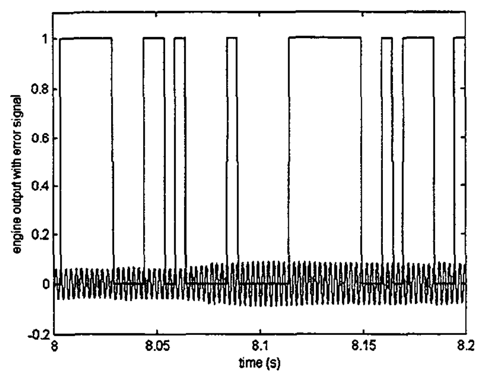

Figure 7: Details of the Continuously Adjusting Process

The result shows that the RWC chip can successfully suppress the initial instability of the engine within about 0.8 second. After which, the chip continuously adjusts on-line to limit the engine output to be within a small magnitude.

7. Conclusion and Future Work

The above result shows that the RWC chip is able to control a simulated combustion engine to the desired performance.

An improved version of the chip in 0.35-μm CMOS process is being fabricated. It contains two layers of neural networks, with 720 weights. The new chip has varying learning step size proportional to the error signal. When the error is small, the weights make small adjustments around the current learned value. Thus, a wrong adjustment will not drive the weights far from the learned value, which is relatively good when the error is already small.

8. References

[1] Werbos, P.J., "Neurocontrol and elastic fuzzy logic: capabilities, concepts, and applications," *IEEE Transactions on Industrial Electronics*, vol. 40, no. 2, pp. 170-80, April 1993.

[2] Ahmed, R.S., Rattan, K.S. and Khalifa, I.H., "Real-time tracking control of a DC motor using a neural network," *Proceedings of the IEEE National Aerospace and Electronics Conference*, vol. 2, pp. 593-600, 1995.

[3] Shaffner, and C., Schroder, D., "An application of general regression neural network to nonlinear adaptive control," *Fifth European Conference on Power Electronics and Applications*, vol. 4, pp. 219-24, September 1993.

[4] Kalanovic, V.D. and Tseng, W.-H., "Back-propagation in feedback error learning," *Proceedings of Neural, Parallel and Scientific computations*, vol. 1, pp. 239-42, might 1995.

[5] Thomas Kailath, *Linear Systems*, p. 188, Prentice-Hall, Inc., Englewood Cliffs, N. J. 07632, 1980.

[6] Yuzo Hirai, "Recent VLSI Neural Networks in Japan", *Journal of VLSI Signal Processing*, 6, 7-18, 1993.

[7] H. Eguchi, T. Furuta, H. Horiguchi, S. Oteki, and T. Kitaguchi, "Neural Network LSI Chip with On-Chip *Proceedings of the International Joint Conference on Neural Networks*, Vol. 1, pp. 453-6, 1991.

[8] Akira Masaki, Yozo Hirai, and Minoru Yamada, "Neural Networks in CMOS: a Case Study", *IEEE Circuits and Device Magazine*, Vol. 6, No. 4, P.12-17, July 1990.

[9] L. M. Reyneri, "Neuro-Fuzzy Hardware: Design, Development and Performance", Private Communications.

[10] M. Jabri and B. Flower, "Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks", *IEEE Transaction on Neural Networks*, Vol.3, No.1, January 1992.

[11] A. J. Montalvo, R. S. Gyurcsik, and J. J. Paulos, "An Analog VLSI Neural Network with On-Chip Perturbation Learning", *IEEE Journal of Solid-State Circuits*, Vol.32, No.4, April 1997.

[12] K. Hirotsu and M. Brooke, "An analog neural network chip with random weight change learning algorithm", *Proceedings of the International Joint Conference on Neural Networks*, pp. 3031-3034, October 1993.

[13] P. W. Hollis and J. J. Paulos, "A Neural Network Learning Algorithm Tailored for VLSI Implementation", *IEEE Transaction on Neural Networks*, Vol.5, No.5, September 1994.

[14] C. Schneider and H. Card, "Analog CMOS Synaptic Learning Circuits Adapted from Invertebrate Biology", *IEEE Transaction on Circuits and Systems*, Vol.38, No.12, December 1991.

[15] T. Morie and Y. Amemiya, "An All-Analog Expandable Neural Network LSI with On-Chip Backpropagation Learning", *IEEE Journal of Solid-state Circuits*, Vol.29, No.9, September 1994.

[16] K. Hirotsu and M. Brooke, "An analog neural network chip with random weight change learning algorithm", *Proceedings of the International Joint Conference on Neural Networks*, pp. 3031-3034, October 1993.

[17] C. R. Schneider and H. C. Card, "Analog CMOS Deterministic Boltzmann Circuits", *IEEE Journal of Solid-state Circuits*, Vol.28, No.8, August 1993.

[18] Y. He and U. Cilingiroglu, "A Charge-Based On-Chip Adaptation Kohonen Neural Network", *IEEE Transactions on Neural Networks*, Vol.4, No.3, May 1993.

[19] Y. K. Choi and S. Y. Lee, "Subthreshold MOS Implementation of Neural Networks with On-Chip Error Back-Propagation Learning", *Proceedings of International Joint Conference on Neural Networks*, pp.849-852, 1993.

[20] J. A. Lansner and T. Lehmann, "An Analog CMOS Chip Set for Neural Network with Arbitrary Topologies", *IEEE Transactions on Neural Networks*, Vol.4, No.3, May 1993.

[21] J. B. Lont and W. Guggenbuhl, "Analog CMOS Implementation of a Multilayer Perceptron with Nonlinear Synapses", *IEEE Transaction on Neural Networks*, Vol.3, No.3, May 1992.

[22] B. Linares-Barranco, E. Sanchez-Sinencio, A. Rodriguez-Vazquez, and J. L. Huertas, " A CMOS Analog Adaptive BAM with On-Chip Learning and Weight Refreshing", *IEEE Transactions on Neural Networks*, Vol.4, No.3, May 1993.

[23] T. Shima, T. Kimura, Y. Kamatani, T. Itakura, Y. Fujita, and T. Iida, "Neuro Chips with On-Chip Back-Propagation and/or Hebbian Learning", *IEEE Journal of Solid-State Circuits*, Vol.27, No.12, December 1992.

[24] P. W. Hollis and J. J. Paulos, "Artificial Neural Networks Using MOS Analog Multipliers", *IEEE Journal of Solid-State Circuits*, Vol.25, No.3, June 1990.

[25] T. Lehmann, E. Bruun, and C. Dietrich, "Mixed Analog/Digital Matrix-Vector Multiplier for Neural Network Synapses." *Analog Integrated Circuits and Signal Processing*, 9, pp. 55-63, 1996.

[26] B. Burton, F. Kamran, R. G. Harley, T. G. Habetler, M. A Brooke; R. Poddar, "Identification and control of induction motor stator currents using fast on-line random training of a neural network", *IEEE Transactions on Industry Applications*, Vol.33, No.3, p.697-704, May 1997.

[27] J. Liu, B. Burton, F. Kamran, M. A. Brooke, R. G Harley, T. G. Habetler, "High Speed On-line Neural Network Control of an Induction Motor Immune to Analog Circuit Non-idealities", *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1997.

[28] Y. Neumeier, N. Markopoulos, and B. T. Zinn, "A Procedure for Real-Time Mode Decomposition, Observation, and Prediction for Active Control of Combustion Instabilities", *IEEE Conference on Control Applications*, Oct 5-7, 1997.