# A CMOS Feedforward Neural-Network Chip With On-Chip Parallel Learning for Oscillation Cancellation

Jin Liu, *Member, IEEE*, Martin A. Brooke, *Member, IEEE*, and Kenichi Hirotsu, *Member, IEEE*

***Abstract*—This paper presents a mixed signal CMOS feedforward neural-network chip with on-chip error-reduction hardware for real-time adaptation. The chip has compact on-chip weighs capable of high-speed parallel learning; the implemented learning algorithm is a genetic random search algorithm—the random weight change (RWC) algorithm. The algorithm does not require a known desired neural-network output for error calculation and is suitable for direct feedback control. With hardware experiments, we demonstrate that the RWC chip, as a direct feedback controller, successfully suppresses unstable oscillations modeling combustion engine instability in real time.**

***Index Terms*—Analog finite impulse response (FIR) filter, direct feedback control, neural-network chip, parallel on-chip learning, oscillation cancellation.**

## I. Introduction

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. Software simulations on serial computers cannot provide the computational power required, since they transform the parallel neural-network operations into serial operations. When the networks become larger, 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 cannot always be affordable.

As a solution to the above problems, dedicated hardware is purposely designed and manufactured to offer a higher level of parallelism and speed. Parallel operations can potentially provide high computational power at a limited cost, thus, can potentially solve a complex problem in a short time period, compared with serial operations. However, reported implementations of neural networks do not always exploit the parallelism.

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 hardware-friendly algorithm, called random-weight-change (RWC) algorithm [1], is implemented on CMOS processes. The RWC algorithm is a fully parallel rule that is insensitive to circuit nonidealities. In addition, the error can be specified such that minimizing the error leads the system to reach its desired performance and it is not necessary to calculate the error by comparing the actual output of the neural network with the desired output of the neural network. This enables the RWC chip to operate as a direct feedback controller for real-time control applications.

In the last decade, research has demonstrated that on-chip learning is possible on small problems, like XOR problems. In this paper, a fully parallel learning neural-network chip is experimentally tested to operate as an output direct feedback controller suppressing oscillations modeling combustion instability, which is a dynamic nonlinear real-time system.

## II. Issues on the Design of Learning Neural-Network Hardware

Neural networks can be implemented with software, digital hardware, or analog hardware [2]. Depending on the application nature, cost requirements, and chip size limitations due to manufacturability, each of the implementation techniques has its advantages and disadvantages. The implementations of on-chip learning neural-network hardware differ in three main aspects: the learning algorithm, the synapse or weigh circuits, and the activation function circuits.

### *A. Learning Algorithm*

The learning algorithms are associated with the specific neural-network architectures. This work focuses on the widely used layered feedforward neural-network architecture. Among the different algorithms associated with this architecture, the following algorithms have been implemented in CMOS integrated circuits: the backpropagation (BP) algorithm, the chain perturbation rule, and the random weight change rule.

The BP algorithm requires precise implementation of the computing units, like adders, multipliers, etc. It is very sen-

Manuscript received October 25, 2000; revised July 20, 2001 and January 24, 2002. This work was supported by the Multidisciplinary University Research Initiative (MURI) on Intelligent Turbine Engines (MITE) Project, DOD-Army Research Office, under Grant DAAH04-96-1-0008.

J. Liu is with the Department of Electrical Engineering, the University of Texas at Dallas, Richardson, TX 75080 USA.

M. A. Brooke is with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA.

K. Hirotsu is with the Sumitomo Electric Industries, Ltd., Osaka 541-0041, Japan.

Publisher Item Identifier S 1045-9227(02)05565-0.

sitive to analog circuit nonidealities, thus, it is not suitable for compact mixed signal implementation. Learning rules like serial-weight-perturbation [3] (or Madaline Rule III) and the chain perturbation rule [4] are very tolerant of the analog circuit nonidealities, but they are either serial or partially parallel computation algorithms, thus are often too slow for real-time control. In this research, we use the RWC algorithm [1], which is a fully parallel rule that is insensitive to circuit nonidealities and can be used in direct feedback control. The RWC algorithm is defined as follows.

For the system weights

$$w_i(n+1) = w_i(n) + \Delta w_i(n+1)$$

$$\text{If the error is decreased,}\quad \Delta w_i(n+1) = \Delta w_i(n)$$

$$\text{If the error is increased,}\quad \Delta w_i = \mathrm{Rand}(n)$$

where $\mathrm{Rand}(n)$ is either $+\delta$ or $-\delta$ with equal probability, $\delta$ is a small quantity that sets the learning rate, and $w_i(n)$ and $\Delta w_i(n)$ are the weight and weight change of $i$th synapse at the $n$th iteration. All the weights adapt at the same time in each weight adaptation cycle.

Previously, it has been shown with simulations that a modified RWC algorithm can identify and control an inductor motor [5]. Further simulation-based research has shown that the RWC algorithm is immune to analog circuit nonidealities [6]. An example of analog circuit nonidealities is the nonlinearity and offset in the multiplier, as will be shown in the following section. Replacing the ideal multiplier with the nonlinear multiplier constructed from the measurement result of an integrated circuit implementation of the multiplier, we redo the simulations on identifying and controlling an inductor motor. The results of both conditions are almost identical, with minor difference in initial the learning process [6].

### B. Synapse Circuits

Categorized by storage types, there are five kinds of synapse circuits: capacitor only [1], [7]–[11], capacitor with refreshment [12]–[14], capacitor with EEPROM [4], digital [15], [16], and mixed D/A [17] circuits.

Capacitor weights are compact and easy to program, but they have leakage problems. Leakage current causes the weight charge stored in the capacitor to decay. Usually, the capacitors have to be designed large enough (around 20 pF for room temperature decay in seconds) to prevent unwanted weight value decay. Capacitor weights with refreshment can solve leakage problem, but they need off chip memory. In addition, the added A/D and D/A converters either make the chip large or result in slow serial operation. EEPROM weights are compact nonvolatile memories (permanent storage), but they are process sensitive and hard to program. Digital weights are usually large, requiring around 16-bit precision to implement BP learning. The mixed D/A weight storage is a balanced solution when permanent storage is necessary.

For this research, the chip is to operate in conditions where the system changes continuously and so weight leakage problems are mitigated by continuous weight updates. Thus, the chip described here uses capacitor as weight storage. The weight retention time is experimentally found to be around 2 s for loosing 1% of the weight value at room temperature.

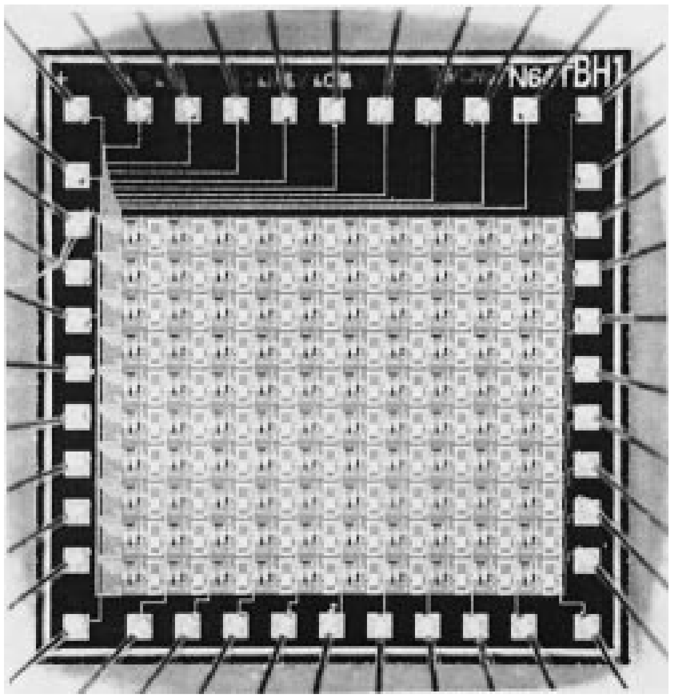

Fig. 1. Chip photo.

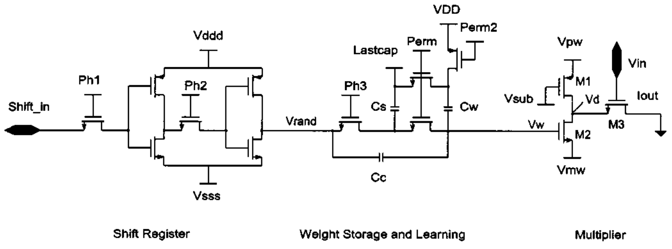

Fig. 2. Schematic of a weight cell.

### C. Activation Function Circuits

Research [11], [18] shows that the nonlinearity used in neural-network activation functions can be replaced by multiplier nonlinearity. In this work, since the weight multiplication circuit has nonlinearity, we uses a linear current to voltage converter with saturation to implement the activation function.

## III. Circuit Design

### A. Chip Architecture

The chip was fabricated through MOSIS in Orbit 2-$\mu$m n-well process. Fig. 1 shows a photomicrograph of the 2 mm on a side chip. It contains 100 weights in a $10 \times 10$ array and has ten inputs and ten outputs. The input pads are located at the right side of the chip, and the output pads are located at the bottom side of the chip. The pads at the top and left sides of the chip are used for voltage supplies and control signals. This arrangement makes it possible for the chip to be cascaded into multilayer networks.

The schematic of one weight cell is shown in Fig. 2. The left part is a digital 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.

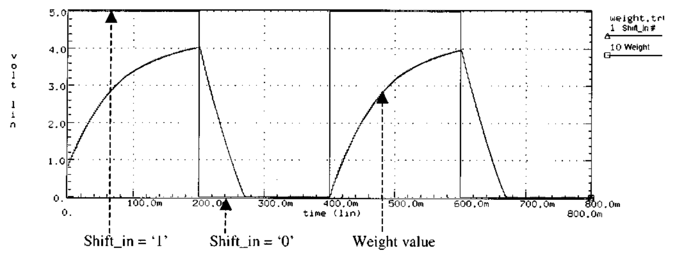

Fig. 3. HSPICE simulation result on the adjustment of a weight value.

The shift registers of all the cells are connected as a chain, therefore, only one random bit needs to be fed into the chip at a time. At a given time, each cell sees a random number at the output of the shift register, being either "1" or "0." If it is "1," the voltage $V_{\rm rand}$ is equal to $V_{\rm ddd}$; if it is a "0," the voltage is equal to $V_{\rm sss}$.

## B. Weight Storage and Adaptation Circuits

The weight charge is stored in the larger capacitor $C_w$, with $V_w$ representing the weight value. Switching clock *Ph3* on, while clock *perm* is off, loads the smaller capacitor $C_s$ with a small amount of charge. Then, connecting $C_w$ in parallel with the smaller capacitor $C_s$ changes the weight value. Suppose that the voltage across $C_s$ is $V_s$ and the voltage across $C_w$ is $V_x$ before connecting them in parallel, after connecting them in parallel for charge sharing, the final voltages across them are the same, supposed to be $V_{x\,\rm new}$. The total charge carried over the two capacitors does not change, $C_sV_s + C_wV_x = (C_s + C_w)V_{x\,\rm new}$, thus the new voltage across $C_w, V_{x\,\rm new}$, will be $V_{x\,\rm new} = (C_s)/(C_s + C_w)V_s + (C_w)/(C_s + C_w)V_x$. In this implementation, the $C_w$ is 100 times of $C_s$, thus $V_{x\,\rm new} = 0.01\,V_s + 0.99V_x$. So, every time, the weight value changes approximately by 1% of the voltage across $C_s$. However, the change is nonlinear, due to the weight decay term, $0.99V_x$ in the above equation. The bottom plate of $C_s$ will be charged to $V_{\rm rand}$, which will be either $V_{\rm ddd}$ or $V_{\rm sss}$. The top plate of $C_s$ is connected to a bias voltage *Lastcap*. The values of $V_{\rm ddd}, V_{\rm sss}$, and *Lastcap* together control the step size of weight change. The $V_{\rm DD}$ is an external biasing to set the range of the actual weight value, which is the sum of the value of $V_{\rm DD}$ and the charge across $C_w$. Clock *perm2* has a complementary phase of clock *perm*.

An individual weight will have its value either increased or decreased every time the clock *Ph3* is activated. When the data shifted into the weight cell is a "1" (5 v), the weight is increased; and when the data shifted in is a "0" (0 v), the weight is decreased. Fig. 3 shows the results of an HSPICE simulation of the weight changing with time. In the simulation, from the time 0 to 200 ms, a series of "1s" are shifted into the cell. Clock *Ph3* is activated to allow the random number to be added to the permanent weight change; clock *perm* turns on and off to make permanent change on the weight value. Clocks *Ph3* and *perm* have complimentary phases with period of 2 ms. As a result, the weight keeps on incrementing for 100 times during the 200 ms period. From time 200 to 400 ms, a series of "0's" are shifted to the cell, so the weight keeps on decrementing. The same process repeats for several cycles in the simulation. The first two cycles are shown in the figure, the rest of the cycles are identical to the second cycle.

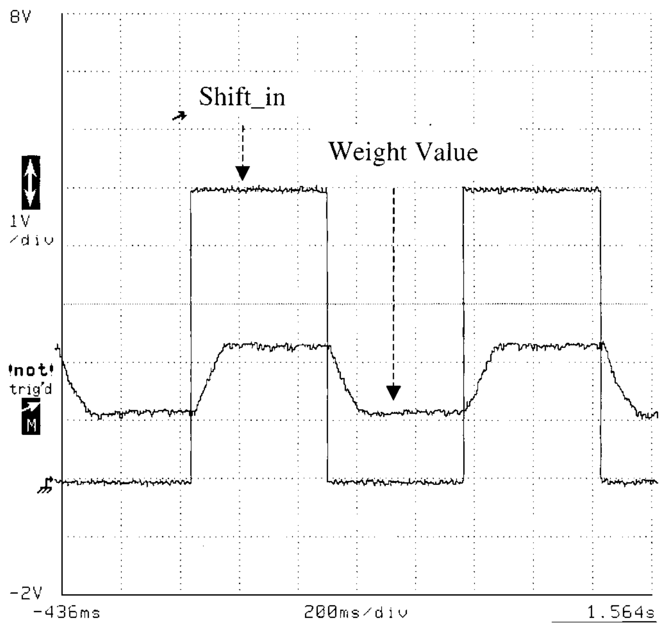

Fig. 4. Measured result on the adjustment of a weight value.

The weight increment and decrement rates are determined by the values of $V_{\rm ddd}, V_{\rm sss}$, and *Lastcap*, as mentioned earlier. In this simulation, $V_{\rm ddd}$ is 5 V, $V_{\rm sss}$ is 0 V, and *Lastcap* is 2.5 V. When a "0" is shifted in, $V_{\rm rand}$ equals to 0 V; when a "1" is shifted in, $V_{\rm rand}$ equals to 5 V. However, the voltage at the bottom plate of $C_s$ does not always equal to $V_{\rm rand}$ exactly, due to the NMOS switching gate controlled by *Ph3,* which is 5 V. Suppose the threshold voltage of the switching gate is 0.7 V, the voltage at the bottom plate of $C_s$ equals to 4.3 V when $V_{\rm rand}$ equals to 5 V, and equals to 0 V when $V_{\rm rand}$ equals to 0 V. Thus, the increment step is approximately 0.018 V while the decrement step is 0.025 V, corresponding to about 7 $1/2$-bit resolution.

Fig. 4 shows the measurement result of the weight increment and decrement, for comparison with the simulated result shown in Fig. 3. The shift_in data are series of "0s" and "1s." In this measurement, the three voltages controlling the weight increment and decrement step size are adjusted so that the up slope and the down slope are almost symmetrical.

The following scheme implements the RWC learning. If the calculated error decreases, clocks *Ph1* and *Ph2* stop. The same random number, representing the same weight change direction, will be used to load $C_s$ with the charge, thus, the weights change in the same direction. If the error is increased, clocks *Ph1* and *Ph2* are turned on, a new random bit will be shifted in, resulting a random change of the weight values.

## C. Multiplier Circuits

The operation of the multiplier, whose schematic is shown in Fig. 2, is as follows. The voltage $V_{\rm sub}$ is the substrate voltage, which is the most negative voltage among all the biasing voltages. In the simulation and experiments, we use complimentary power supplies, i.e., $V_{\rm pw} = -V_{\rm mw}$. The output of the multiplier is a current flowing into a fixed voltage, which should be

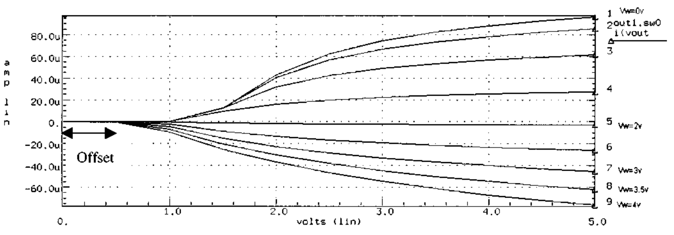

Fig. 5. HSPICE simulation result on the multiplication function.

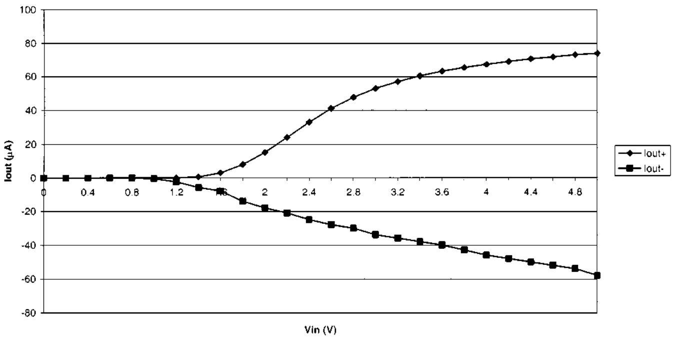

Fig. 6. Measured result on the multiplier output current range.

in the middle of $V_{\rm pw}$ and $V_{\rm mw}$; in this case, it is ground. The weight voltage $V_w$ is added at the gate of M2. Fig. 5 shows the HSPICE simulation result of the multiplier. The horizontal axis is the input voltage $(V_{\rm in})$, the vertical axis is the output current $(I_{\rm out})$, and different curves represent different weight voltage values $(V_w)$. The multiplier attempts to produce a multiplying relationship as follows: $I_{\rm out} = V_{\rm in} * V_w$. When $V_w$ is about 2 V, the drain of M2 $V_d$ is about 0 V; when $V_w$ is below 2 V, $V_d$ is positive and when $V_w$ is above 2 V, $V_d$ is negative. The range of $V_d$ is small to ensure that M3 is operated in the nonsaturation region, thus the output current of M3 is approximately proportional to drain-source voltage, $V_{\rm DS3}$. Depending on the polarity of $V_d$, the output current $I_{\rm out}$ can flow in both directions and is defined as follows:

$$I_{\rm out} = \begin{cases} \beta(V_{\rm in} - V_T)V_d, & V_d \geq 0 \\ \beta(V_{\rm in} - V_d - V_T)V_d, & V_d \leq 0 \end{cases}$$

where $V_T$ is the threshold voltage of M3. The above equation explains why the simulated multiplier has both offset and nonlinearity. The nonlinear relationship is actually desirable as it eliminates the need for a nonlinear stage following the multipliers, as discussed earlier.

Hardware test results, presented in Fig. 6, show that the measured multiplier function is close to the HSPICE simulation result. The two lines are constructed from the measured points when the weight is programmed to be at its maximum and minimum. The horizontal axis is input voltage, with units of V and the vertical axis is current, with units of $\mu$A.

## IV. Learning Process

In the learning, a permanent change is made every time a new pattern is shifted in. If the change makes the error decrease, the weights will keep on changing in the same direction in the following iterations, until the error is increased. If the change makes the error to increase, the weights keep this change, and try on a different change for the next iteration.

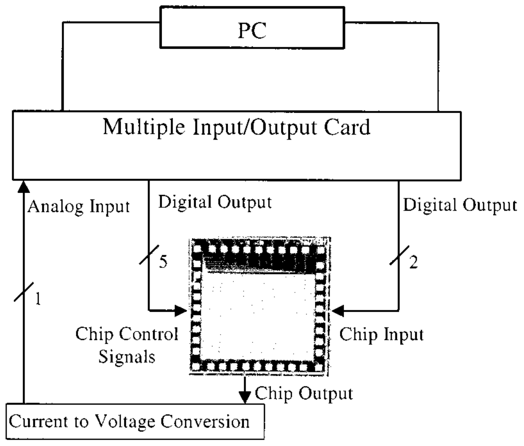

Fig. 7. Test setup for the inverter experiment.

The test setup shown in Fig. 7 is used to demonstrate the random-weight-change learning process. The task is to train a two-input–one-output network to implement an inverter. It is configured so that one input is always held high as the reference, while the second one alternates between high and low. The desired output is the inverse of the second input.

In the test, the high and low are set to two voltage values for the network outputs to reach. The network then is trained to minimize an error signal, which is calculated as follows. Suppose that the desired output values of high and low are $D_h$ and $D_l$, and the actual output values of high and low are $O_h$ and $O_l$, the error is calculated as error $= \sqrt{(D_h - O_h)^2 + (D_l - O_l)^2}$. Thus, when the error is small enough, the network implements an inverter. The error is not calculated on current chip. Rather, it is calculated on PC in the test setup and is sent to the chip, as a 1-bit digital signal. However, the error calculation can be incorporated on the same chip, with additional digital circuits for error calculation.

The desired low and high output voltages, in this experiment, are 1 and 2 V. So, the desired output should oscillate between 1 and 2 V. Fig. 8 shows a typical initial learning process captured from the oscilloscope. The figure shows that, within 0.8 s, the network is trained to behave as an inverter with the specified high and low output voltages.

After the initial training, the network converges to the inverter function. Then, the network tries to maintain the performance as an inverter by continuously adjusting itself. Fig. 9 shows a 100-ms time slice of the continuously adjusting process; the desired high and low output voltages are 1.5 and 0.5 V for this case. There are two signals in the plot. One of them is the *ph2* clock, which is represented by the spikes shown in the figure. A high of *ph2* means that a new random number is sifted in and indicates that the error starts to increase. The other signal is the inverter output. It oscillates between high and low, since the input alternates between low and high. The two horizontal markers indicate the desired high and low voltages.

Starting from point A (time 0), indicated by the trigger arrow, clock *ph2* is high, thus, a new random pattern is introduced. From point A to point B, the output signal oscillates between high and low, converging to the desired high and low values.

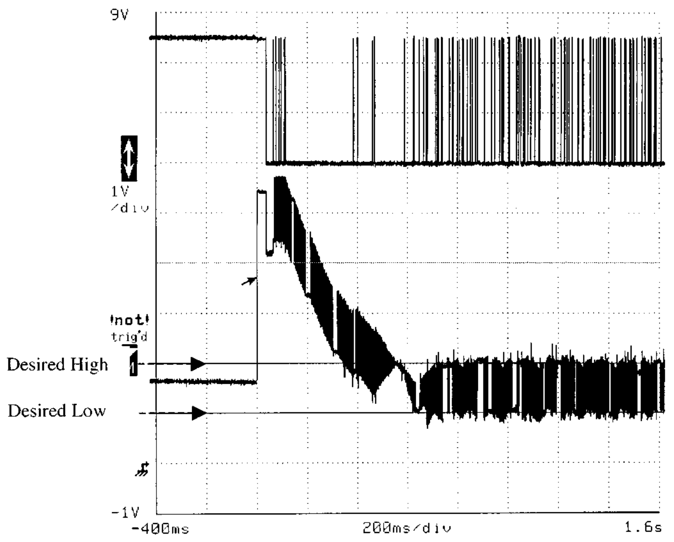

Fig. 8. Oscilloscope screen capture of the initial learning process for the inverter experiment, with the desired low and high voltages as 1 and 2 V.

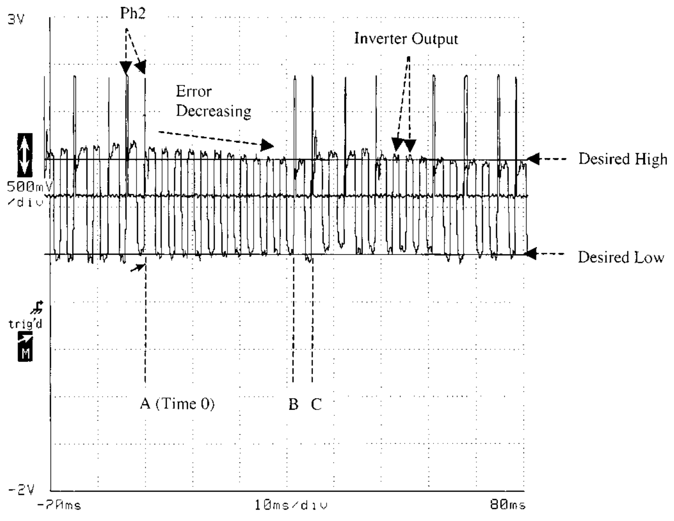

Fig. 9. Oscilloscope screen capture of the detailed learning process of the inverter experiment.

According to the error calculation equation, the error decreases. During this process, clock *Ph2* stops to let the network keep on using the weight change, which is consistent with the algorithm. The error decreases until point B, when the error starts to increase. Thus, the network stops using this weight change direction and tries a new pattern, indicated by a spike of *ph2* at point B. Unfortunately, this pattern cause the error to increase; the network gives up this direction pattern and tries a new one, indicated by another spike at point C. As this process goes on, the network dynamically maintains its performance as an inverter by continuously adjusting its weights.

The above experimental results show that the recorded hardware learning process complies with the random weight change algorithm and the weights of neural-network chip can be trained in real time for the neural network to implement simple functions. Next, we apply the chip to a more complicated application—direct feedback control for combustion oscillation cancellation.

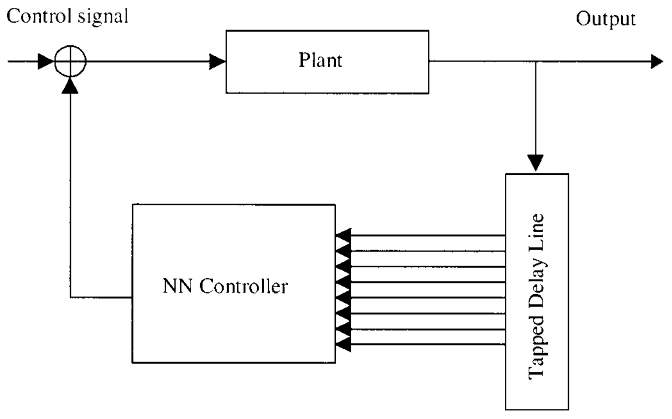

Fig. 10. Direct feedback control scheme with a neural-network controller.

## V. Combustion Instability and Direct Feedback Control

The combustion system is a dynamic nonlinear system, with randomly appearing oscillations of different frequencies and unstable damping factors. When no control is applied, this system is unstable and eventually reaches a bounded oscillation state. The goal of the control is to suppress the oscillation. There are several well-known passive approaches for reducing the instabilities [19], [20]. However, the implementation of such passive approaches is high cost and 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. Since the combustion system is a nonlinear system, the system parameters vary with time and operating conditions. The active controllers that developed to suppress the oscillation in fixed modes cannot deal with the unpredicted new oscillation modes. In addition, the actuation delay presented in the control loop also causes difficulties for the control.

In this research, we use the neural-network chip for direct feedback control [21] of the oscillation. The RWC chip has on-chip learning ability; the weights on the chip are adjusted in parallel, which enables the chip to adapt fast enough for many real-time control applications. The adaptation time of each weight update is about 2 ns. Fig. 10 shows the direct feedback control scheme with the neural-network chip as controller. The tapped delay line in control setup is used to sample the plant output (combustion chamber pressure). In general, a period of the plant output of the lowest signal frequency is to be covered. At the same time, the sampling rate of the tap delay line should also be faster than the Nyquist sampling rate of the highest frequency component of the plant output. The rule is that the neural network should be provided enough information on the plant dynamics. Software simulation [22], [23], using the setup in Fig. 10, suggests that it is possible to suppress the combustion oscillation with the direct feedback control scheme using the neural-network controller with the RWC algorithm.

### A. Combustion Model With Continuously Changing Parameters

In this simulation, the combustion process is modeled by the limit cycle model: $\ddot{x} + 2\zeta\omega(x^2 - b)/(b)\dot{x} + \omega^2 x = u$, where

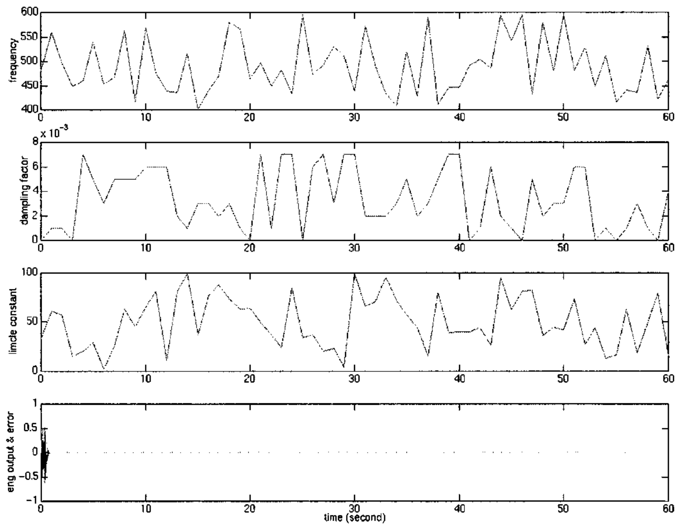

Fig. 11. Simulation result on the control of an unstable engine with continuously changing parameters.

$u$ is the input to the engine, $x$ is the output, $\omega$ is the oscillation frequency, $\zeta$ is the damping factor, and $b$ is limit cycle constant. The simulation runs with a sampling rate of 8 kHz. The feedforward neural network had eight inputs, three hidden neurons, and one output. The combustion engine output is tap delayed; the original engine output and the tap-delayed signals are the inputs of the neural-network controller. The model parameters of the combustion process change with time and with prefixed ranges: 400–600 Hz for frequency, 0–0.008 in damping factor, and 1–100 in limit cycle constant.

Fig. 11 shows the simulation result when the parameter change rate is 1 point/s. It means that every second, a new set of plant model parameters are randomly picked within the parameters' predefined range and the value of the parameters between these randomly picked pointed is defined by linearly interpolating between the two neighbor points. In the figure, the horizontal axis is time. The top three plots are the plant parameters, which change with time and will cause unstable oscillations for the engine. The bottom plot is the engine output, the engine is stabilized around 2 s. The error signal is the low-passed signal of the engine output.

## B. Noise Tolerance

This section presents the simulation result of the neural-network control of the simulated limit cycle single frequency combustion instability with 10% random noise. The purpose of these simulations is to determine the noise tolerance of the system. The 10% noise is a fair estimation of the real combustion system. The plant parameters were as follows: the frequency was 400 Hz, the damping factor was 0.005, and the limit cycle constant was one. The initial engine pressure was 0.1, without control it eventually enters a limit cycle that with an amplitude of 2 in about 400 ms as shown in the top plot in Fig. 12. The bottom plot shows the engine output with the control of the neural network; only the additive noise remains in the engine output after about 5 s.

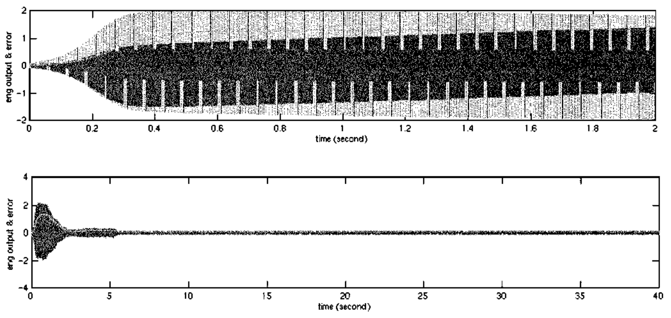

Fig. 12. Combustion oscillation with 10% additive noise under no control and under the control of the neural network.

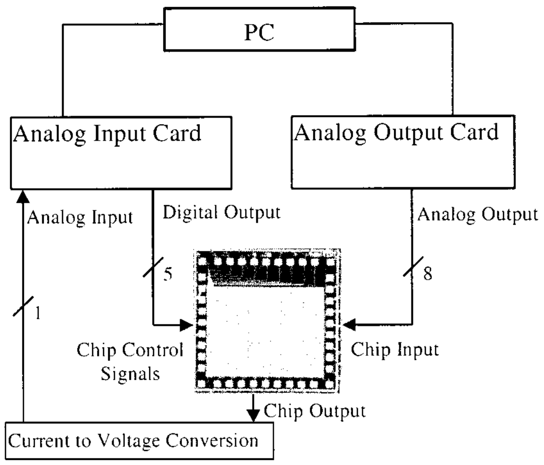

Fig. 13. Experimental test setup.

## VI. Experimental Results

### A. Test Setup

The hardware test setup is shown in Fig. 13. To test the hardware chip, we simulate the oscillation process on a PC, which only provides digital outputs. The analog-to-digital converter (ADC) and the digital-to-analog converter (DAC) cards provide interface between the oscillating process and the hardware chip. The neural-network chip itself requires several interface pins like power supply, two nonoverlapping clocks to synchronize the learning process, a random bit, and one-bit error increase or decrease signal. Most of the interface signal can be generated by standard circuits available in the literature, except the one-bit error increase or decrease signal, which has to be designed *ad hoc* depending on the application.

In the test, the chip and the PC simulating the combustion process form a closed loop. The computer runs the program simulating unstable combustion process and generates the engine output. The tap-delayed engine outputs are sent to the chip in parallel through the DAC. The chip functions as a direct feedback controller and generates a control signal for the combustion engine. The chip output is a current, it is first transformed to a voltage and then read into the computer by the ADC. The weights of the chip are updated every time an error is calculated. Due to system delay (mainly combustion actuator delay), we have to wait for about two cycles to calculate the error based on the previous weight update. Within these two cycles, the combustion simulation process keeps on generating outputs, and

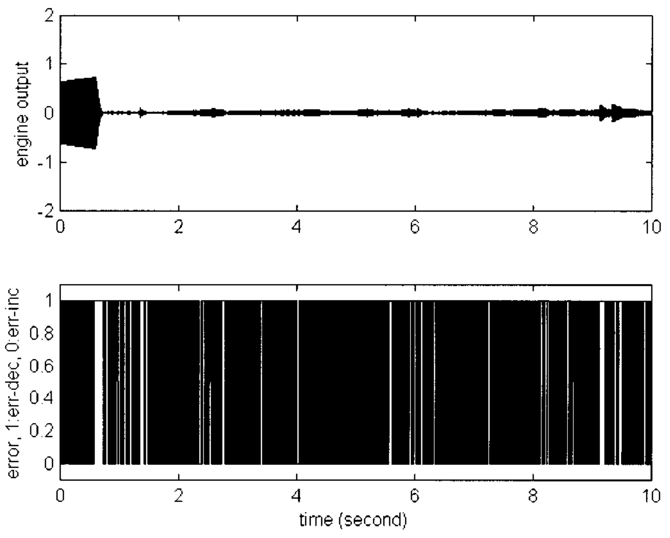

Fig. 14. Experimental result on the suppression of a stable oscillation with the chip as a feedback controller.

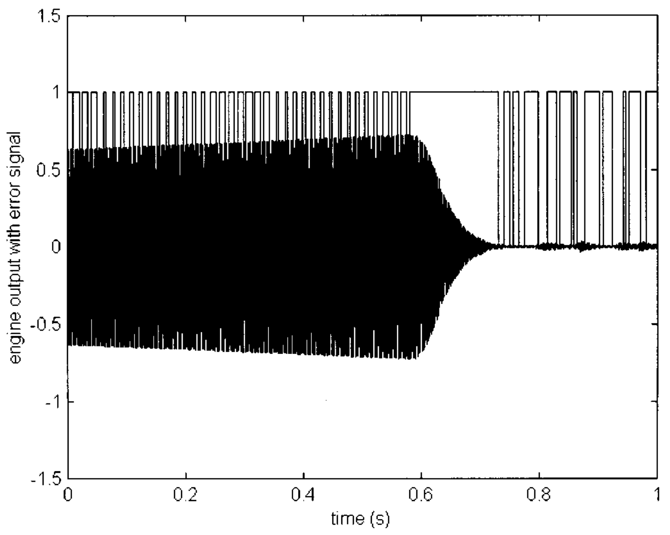

Fig. 15. Details of the initial learning process of the test result shown in Fig. 14.

these outputs are forward propagated through the chip and fed back to the engine input.

## B. Stable Oscillations

In this experiment, the combustion process is simulated using the limit cycle model. The frequency $\omega$ is 400 Hz and the damping factor $\zeta$ is zero. The run time for the simulation is 10 s. Assigned an initial state with the oscillation magnitude of 0.5, the process will reach a fixed oscillation magnitude of one, if no control is applied. The error signal is specified to be proportional to the magnitude of the oscillation and is calculated by low-passing the rectified engine output oscillation signal. Thus, minimizing the error signal will minimize the oscillation magnitude, which in turn suppress the oscillation appeared at the engine output.

Fig. 14 shows one of the test results. The first plot in the figure is the engine output and the second plot is the error-decrement signal. A "1" of the error-decrement signal means that the error decreases and a "0" means the error is increased. At around 1 s, the oscillation magnitude is greatly reduced and the oscillation is limited to a small magnitude afterwards. Fig. 15 shows the details of the learning process. At the beginning of the learning process, the chip explores weight changes in different directions, the error-decrement signal oscillates and the output magnitude increases slowly. At about 0.6 s, the chip finds a good weight change direction, it then keeps on changing the weights in this direction. As a result, the error-decrement signal remains to be "1" for this period and the magnitude of the oscillation is greatly reduced. For the rest of the process, the weights adjust to stay around the optimal value; the error-decrement signal frequently oscillates between "0" and "1" and the oscillation magnitude fluctuates, within a small range.

The above results show that the RWC chip is able to suppress the oscillation to a small magnitude. It takes about 1 s to suppress the oscillation initially. After which, the oscillation is limited to a small magnitude.

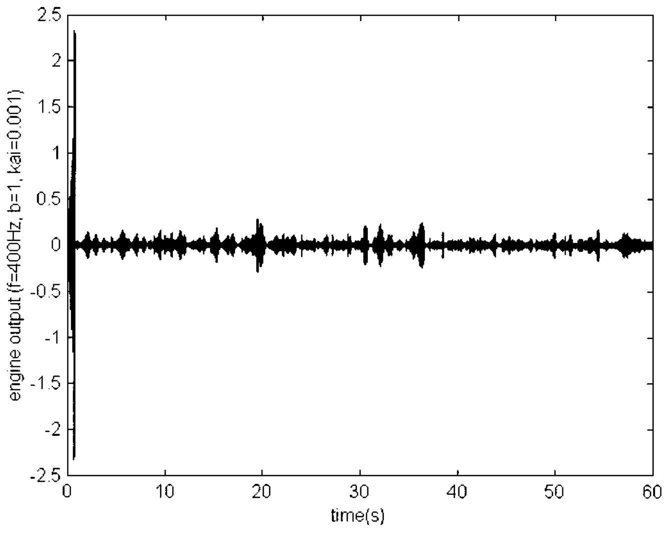

Fig. 16. Experimental result on the suppression of an unstable oscillation with the chip as a feedback controller ($\zeta = 0.001$).

## C. Unstable Oscillations

The above experimental test shows the chip suppressing a stable oscillation, in this section, we present the experimental result of the chip suppressing unstable oscillations. The oscillation frequency is 400 Hz and limit cycle constant is one. Fig. 16 shows the result with damping factor of 0.001. The chip suppresses the oscillation within around 1 s and limits the magnitude to be within 0.3. Fig. 17 shows the result with damping factor of 0.002. The chip suppresses the oscillation within around 1 s. There are two big reoccurred blowups with magnitude of 1.8 and 0.7, after which the engine output is limited to the magnitude to be within 0.5.

In both experiments, the weights are adjusted continuously after the initial oscillation suppression. It is observed that as the damping factor increases, the chip has harder time to limit the engine output within a small magnitude, the reason is that a larger damping factor results in an oscillation with larger magnitude. In addition, the I/O card delay and the software simulation

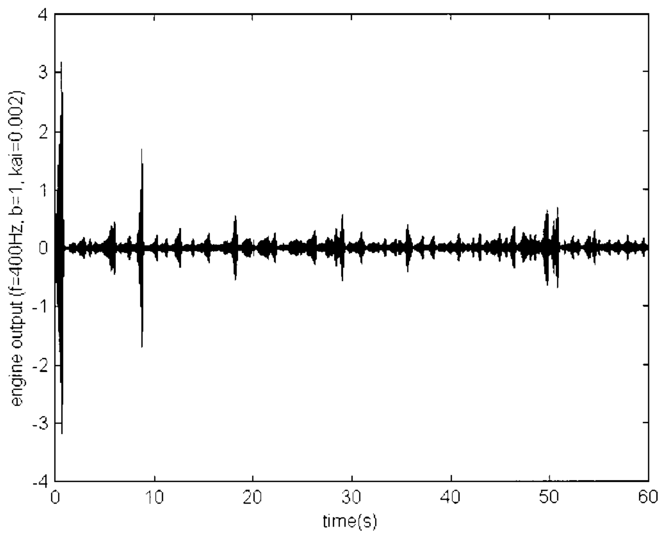

Fig. 17. Experimental result on the suppression of an unstable oscillation with the chip as a feedback controller ($\zeta = 0.002$).

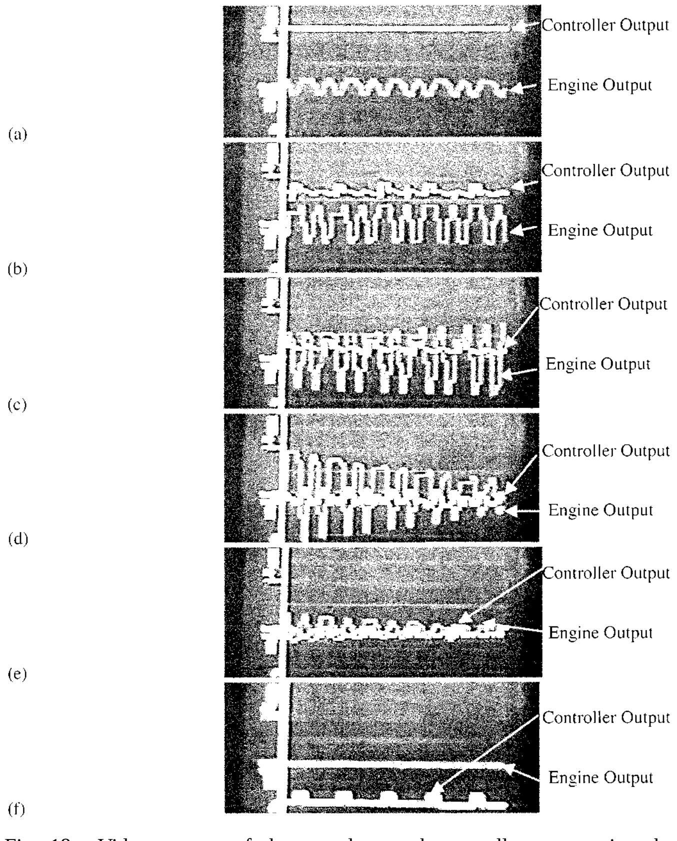

Fig. 18. Video capture of the neural-network controller suppressing the oscillation.

time of the engine cause the weights to decay and drive them away from the learned values.

### D. Controller Output

In this section, the control signal generated by the neural-network controller is presented and analyzed. At the beginning, the neural-network controller is not trained and the engine output starts to oscillate, as shown in Fig. 18(a). As a result, the amplitude of the engine oscillation increases, as shown in Fig. 18(b). The neural network tries on random weight changes, Fig. 18(c) shows that at certain time, the weight change can cause the controller output to be in phase with the engine oscillation and the amplitude of the engine oscillation increases. Fig. 18(d) shows that the neural network finds a weight change direction that suppressing the oscillation and the oscillation amplitude is decreasing. The neural-network controller output and the engine output almost have complementary phases. As mentioned in Section V, there is actuator delay in the combustion engine model, so the controller output signal cannot be exactly 180° out of phase with the engine oscillation in order to suppress the oscillation. Fig. 18(e) shows that the engine oscillation is further suppressed to very small amplitude. After the neural network controller is trained, the engine output is limited to a small residue oscillation, as shown in Fig. 18(f). The result shown in Fig. 18 is for unstable limit cycle oscillation with damping factor of 0.001.

## VII. Conclusion and Future Work

We presented a mixed signal CMOS integrated circuit implementation of feedforward type neural networks; all the weights on the chip are adjusted on-chip in parallel for fast real-time control. The implemented RWC learning algorithm enables the chip to function as a direct feedback controller for real-time control applications. Through hardware experiments, we have demonstrated the successful oscillation cancellation using the chip as a direct feedback controller. Future work includes adjustable learning step size proportional to the error signal and nonvolatile weight storage.

## References

[1] K. Hirotsu and M. Brooke, "An analog neural network chip with random weight change learning algorithm," in *Proc. Int. Joint Conf. Neural Networks*, Oct. 1993, pp. 3031–3034.
[2] G. Cauwenbergh and M. A. Bayoumi, *Learning on Silicon: Adaptive VLSI Neural Systems*. Amsterdam, The Netherlands: Kluwer, 1999.
[3] M. Jabri and B. Flower, "Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks," *IEEE Trans. Neural Networks*, vol. 3, Jan. 1992.
[4] A. J. Montalvo, R. S. Gyurcsik, and J. J. Paulos, "An analog VLSI neural network with on-chip perturbation learning," *IEEE J. Solid-State Circuits*, vol. 32, Apr. 1997.
[5] B. Burton, F. Kamran, R. G. Harley, T. G. Habetler, M. A. Brooke, and R. Poddar, "Identification and control of induction motor stator currents using fast on-line random training of a neural network," *IEEE Trans. Ind. Applicat.*, vol. 33, May 1997.
[6] J. Liu, B. Burton, F. Kamran, M. A. Brooke, R. G. Harley, and T. G. Habetler, "High speed on-line neural network control of an induction motor immune to analog circuit nonidealities," in *Proc. IEEE Int. Symp. Circuits Syst.*, June 1997, pp. 633–636.
[7] T. Morie and Y. Amemiya, "An all-analog expandable neural network LSI with on-chip backpropagation learning," *IEEE J. Solid-State Circuits*, vol. 29, Sept. 1994.
[8] C. R. Schneider and H. C. Card, "Analog CMOS deterministic Boltzmann circuits," *IEEE J. Solid-State Circuits*, vol. 28, no. 8, Aug. 1993.
[9] Y. He and U. Cilingiroglu, "A charge-based on-chip adaptation Kohonen neural network," *IEEE Trans. Neural Networks*, vol. 4, May 1993.
[10] Y. K. Choi and S. Y. Lee, "Subthreshold MOS implementation of neural networks with on-chip error back-propagation learning," in *Proc. Int. Joint Conf. Neural Networks*, July 1993, pp. 849–852.
[11] C. Schneider and H. Card, "Analog CMOS synaptic learning circuits adapted from invertebrate biology," *IEEE Trans. Circuits Syst.*, vol. 38, Dec. 1991.
[12] J. A. Lansner and T. Lehmann, "An analog CMOS chip set for neural network with arbitrary topologies," *IEEE Trans. Neural Networks*, vol. 4, May 1993.
[13] J. B. Lont and W. Guggenbuhl, "Analog CMOS implementation of a multilayer perceptron with nonlinear synapses," *IEEE Trans. Neural Networks*, vol. 3, May 1992.

[14] 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 Trans. Neural Networks*, vol. 4, May 1993.

[15] 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 J. Solid-State Circuits*, vol. 27, Dec. 1992.

[16] P. W. Hollis and J. J. Paulos, "Artificial neural networks using MOS analog multipliers," *IEEE J. Solid-State Circuits*, vol. 25, June 1990.

[17] T. Lehmann, E. Bruun, and C. Dietrich, "Mixed analog/digital matrix-vector multiplier for neural network synapses," *Analog Integrated Circuits Signal Processing*, vol. 9, 1996.

[18] P. W. Hollis and J. J. Paulos, "A neural network learning algorithm tailored for VLSI implementation," *IEEE Trans. Neural Networks*, vol. 5, Sept. 1994.

[19] Y. Neumeier, N. Markopoulos, and B. T. Zinn, "A procedure for real-time mode decomposition, observation, and prediction for active control of combustion instabilities," presented at the IEEE Conf. Control Applications, Oct. 5–7, 1997.

[20] D. Harjee and F. Reardon, Eds., *Liquid Propellant Rocket Combustion Instability*, 1972, NASA SP 194.

[21] T. Kailath, *Linear Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1980, p. 188.

[22] J. Liu and M. A. Brooke, "A fully parallel learning neural network chip for combustion instability control," in *Proc. Int. Joint Conf. Neural Network*, July 1999, pp. 2323–2328.

[23] J. Liu, Ph.D. dissertation, Georgia Inst. Technol., Atlanta, June 1999.

**Jin Liu** (S'96–M'99) received the B.S. degree in electronics and information systems from Zhongshan University, P.R. China, in 1992, the M.S. degree in electrical and computer engineering from University of Houston, Houston, TX, in 1995, and the Ph.D. degree in electrical and computer engineering from Georgia Institute of Technology, Atlanta, in 1999.

Since 1999, she has been an Assistant Professor of Electrical Enginering at the University of Texas at Dallas, Richardson. Her research interests include high-speed adaptive signal processing involving analog and mixed signal integrated circuit design, low-noise equalization for Gb/s data transmission in CMOS technology, high-speed current mode driver, wide-dynamic range optical sensor interface circuits, delta modulation technique for test and automatic adjustment of high-frequency waveform rise/fall time, and power management circuits for batteryless wireless sensors.

**Martin A. Brooke** (S'85–M'86) received the B.E.(elect.) degree with first-class honors from Auckland University, New Zealand, in 1981. He received the M.S. and Ph.D. degrees in electrical engineering from The University of Southern California, Los Angeles, in 1984, and 1988, respectively.

He is currently Associate Professor of Electrical Engineering at the Georgia Institute of Technology, Atlanta. He has four U.S. patents. He has published more than 100 articles in technical jurnals and proceedings, and articles on his work have appeared in several trade publications. His interests include high-speed high-performance signal processing, learning neural-network hardware development, neural-network prediction of turbulent flow, focal plane image processing hardware development, 1–20 Gb/s digital CMOS transceiver circuits for low-cost fiber optic communication, nonlinear filtering algorithms for telecommunications, nonlinear analog to digital converter design, accurate modeling of high-speed circuit parasitics, and statistically relevant device models for accurate prediction of high-performance integrated circuit yield.

Dr. Brooke won a National Science Foundation Research Initiation Award in 1990, and the 1992 IEEE Midwest Symposium on Circuits and Systems, Myril B. Reed Best Paper Award.

**Kenichi Hirotsu**, photograph and biograpy not available at the time of publication.