Adaptive Signal Processing in Mixed-Signal VLSI with Anti-Hebbian Learning

Miguel Figueroa, Esteban Matamala and Gonzalo Carvajal
Department of Electrical Engineering
Universidad de Concepción
Concepción, Chile
Email: mfigueroa@die.udec.cl

Seth Bridges
Computer Science & Engineering
University of Washington
Seattle, WA 98195-2350, USA
Email: seth@cs.washington.edu

Abstract

We describe analog and mixed-signal primitives for implementing adaptive signal-processing algorithms in VLSI based on anti-Hebbian learning. Both on-chip calibration techniques and the adaptive nature of the algorithms allow us to compensate for the effects of device mismatch. We use our primitives to implement a linear filter trained with the Least-Mean Squares (LMS) algorithm and an adaptive decorrelation network that improves the convergence of LMS. When applied to an adaptive Code-Division Multiple-Access (CDMA) despreading application, our system, without the need for power control, achieves more than 100x improvement in the bit-error ratio in the presence of high interference between users. Our 64-tap linear filter uses $0.25mm^2$ of die area and dissipates $200\mu W$ in a $0.35\mu m$ CMOS process.

1 Introduction

A challenging aspect in the design of portable electronic systems is their need to operate under unknown environmental conditions such as interference, noise, and varying input statistics. Adaptive signal-processing techniques, often in the form of adaptive filters and neural networks, effectively optimize system performance under these conditions because they model the varying statistics of the environment by adapting a set of internal weights to optimize a goal function.

Portable systems also face severe restrictions in cost, power dissipation, and die area. In such cases, implementing adaptive signal-processing algorithms on an embedded processor is often infeasible. The computationally-intensive nature of these algorithms means that even custom digital VLSI solutions can be prohibitively large and power-hungry. For problems that require moderate arithmetic resolution, analog and mixed-signal circuits provide an attractive tradeoff in die area and power dissipation; however,
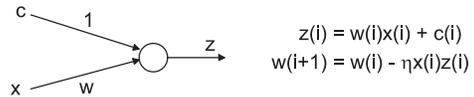
Figure 1. Anti-Hebbian synapse.

these circuits are plagued by problems such as charge leakage, signal offsets, device mismatch, and noise sensitivity.

In this paper, we present a set of analog and mixed-signal primitives that compensates for these problems through on-chip calibration and dynamic adaptation. Using these primitives, we implement adaptive LMS filters based on neural networks trained using anti-Hebbian learning [1] and we show that the same primitives can form decorrelating networks to improve filter convergence in the presence of correlated inputs. Finally, we show an application of our system to improve adaptive CDMA despreading without power control.

2 Anti-Hebbian Learning

Adaptive signal-processing algorithms that update their weights based on correlations between local signals are particularly amenable to hardware implementations because of their local and regular communication structure. Particularly, *anti-Hebbian* learning rules [1] minimize output energy by subtracting an update proportional to the correlation between the input and output. Fig. 1 illustrates this concept on a single-input neural network that computes the synaptic function $z(i) = w(i)x(i) + c(i)$. Using a stochastic gradient descent to minimize the variance of the output, $E[z^2]$, yields the anti-Hebbian learning rule $w(i+1) = w(i) - \eta x(i)z(i)$, where η is a constant learning rate. The independent input $c(i)$ prevents the trivial solution where $w = 0$. The function generalizes naturally to multiple inputs $x_j(i)$.

Despite its simplicity, anti-Hebbian learning is widely used in signal processing. In fact, the learning rule converges when $w(i)x(i)$ is the best approximation to $-c(i)$ in

1
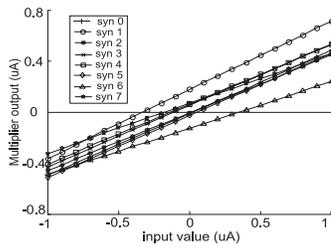
(a) Multiplier output vs. input value.
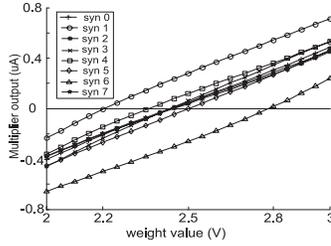
(b) Multiplier output vs. weight value.

Figure 2. Multiplier output for 8 synapses.

(a) Memory cell with linear updates.

(b) Updates in eight memory cells.

Figure 3. A simple PDM analog memory cell.

the sense of the mean-squared value of the error $z(i)$. This is indeed the formulation of the well-known Least-Mean Squares (LMS) algorithm, commonly found in adaptive filtering applications. Another property of this algorithm is that it converges when the output has extracted all the information of $-c(i)$ available from the input $x(i)$ such that the correlation between x and z is minimal. Therefore, anti-Hebbian learning is also used to adaptively decorrelate signals in applications such as dimensionality reduction and blind source separation [2].

3 VLSI Primitives for Anti-Hebbian Learning

To build anti-Hebbian learning networks in VLSI, we need multipliers and adders to implement the forward-path synaptic computation $\sum_{j=1}^{N} x_j w_j$, distributed on-chip memory to store the weights, and memory-update mechanisms to implement the learning rule $\Delta w_j = -\eta z x_j$. This section, based on previous analyses [3] and our own systems simulations [4], describes how the design of these blocks allows us to correct for the effects of device mismatch. Unless otherwise noted, the data shown in this paper was measured from an implementation of these primitives in a $0.35\mu m$ CMOS process.

3.1 Forward-Path Multipliers

Multipliers impose the strongest requirements on die area and power dissipation in VLSI signal-processing sys-

tems. To achieve compact and low-power systems, we use analog multipliers with current outputs in the forward-path computations and we sum the output currents from each synapse on common wires to form the neuron output.

A systems analysis of our filter design shows that the residual error is sensitive to multiplier linearity with respect to the synaptic input x, but relatively robust to their linearity with respect to the weight w, which is automatically compensated by the adaptive algorithm; also, adaptation compensates for weight offsets in the multipliers, provided the weight range is large enough to absorb the offset.

Based on our analysis, we minimize the convergence time and residual error of our filter by using a Gilbert-style multiplier with differential current inputs for x and a differential voltage representation for w. We used long transistors and above-threshold operation to maximize the linearity of x, but favored larger range in w. Fig. 2 shows the transfer function of eight different multipliers on a single chip.

3.2 Weight Storage and Updates

Using analog multipliers in the forward path requires we provide on-chip analog weight storage. Because the performance of the learning algorithm depends directly on the accuracy of the stored weights and update rules, conventional VLSI capacitors are inadequate: Charge leakage requires continuous updates, preventing the open-loop operation common in applications such as adaptive filters. VLSI capacitors are sensitive to charge injection, degrading performance when used with digital pulse-based updates which provide accurate and compact learning rules [5].
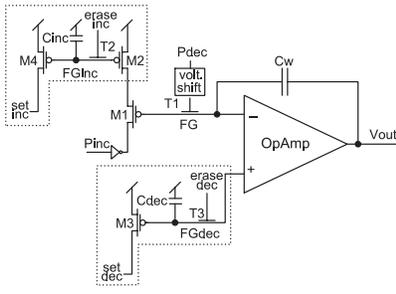
(a) Memory cell with on-chip calibration.
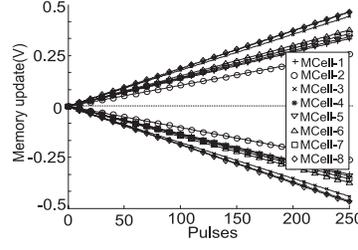
(b) Symmetric updates in 8 memory cells.
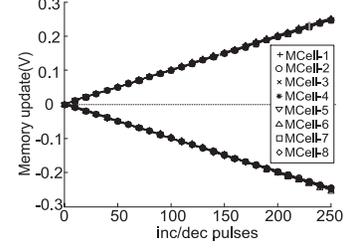
(c) Uniform updates in 8 memory cells.

Figure 4. PDM cell with on-chip calibration.

Instead, we use *synapse transistors* [6] to store and update our analog weights. These devices use charge on a floating gate to provide compact and accurate nonvolatile analog storage. Fowler-Nordheim tunneling adds charge to the floating gate and hot-electron injection removes charge and both mechanisms accurately update the stored value during normal device operation [6]; however, the dynamics of injection and tunneling are highly nonlinear with respect to their control variables (gate, drain and tunneling-junction voltages), which leads to exponential learning rules that do not enable anti-Hebbian learning. This section describes a memory cell based on synapse transistors that supports accurate, linear updates.

Fig. 3(a) shows a pulse-density modulated (PDM) memory cell with linear updates. A negative-feedback loop around an operational amplifier pins the floating-gate voltage FG at V_{bias}. Fixed-width, fixed-amplitude pulses on P_{inc} and P_{dec} trigger electron tunneling and injection, respectively. Because all the control voltages are fixed, the magnitude of the charge updates depends linearly on the density of the update pulses. The charge is integrated on the feedback capacitor C_w, causing a linear update in the output voltage V_{out}. Fig. 3(b) shows the transfer function of eight PDM memory cells on a chip. The integral nonlinearity (INL) of most cells is less than 0.1%, corresponding to a linearity of more than 10 bits.

Fig. 3(b) also highlights an important problem: device mismatch makes it impossible to achieve symmetric updates within a single synapse or equal updates across different synapses using only a single reference voltage V_{bias}. Without symmetric updates, the residual error increases significantly, while unequal updates across synapses result in slower convergence [4]. Fig. 4(a) shows an improved design which adds two local degrees of freedom to the cell. First, we use an additional floating gate FG_{dec} to locally set the reference voltage at each memory cell. By tunneling and injecting to this floating gate, we can vary the voltage at FG, thus changing the relative strengths of tunneling and injec-
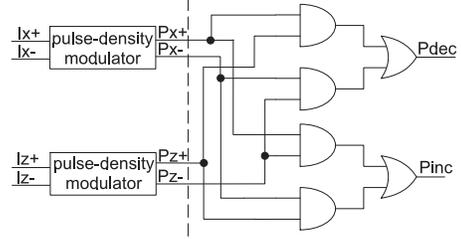
Figure 5. Anti-Hebbian update circuit.

tion at P_{dec} and P_{inc} (increasing the voltage at FG weakens tunneling and strengthens injection and vice versa) until we achieve local symmetry. Second, setting the voltage at the floating gate FG_{inc} controls the source current through M1, limiting injection due to pulses on P_{inc} without affecting the tunneling rate due to pulses on P_{dec}. This added control allows us to achieve symmetric updates and equalize them for different cells across the chip. Fig. 4(b) shows the symmetric updates achieved tuning only FG_{dec}, while Fig. 4(c) shows uniform symmetric updates using both floating gates. The mismatch across cells is now less than 0.25% of their dynamic range (better than 8-bit matching).

3.3 Anti-Hebbian Learning Rules

Fig. 5 shows a block diagram of the digital circuit that implements the anti-Hebbian rules at each synapse. Pulse-density modulators [7] (off-chip in our current implementation) transform the synaptic inputs and neuron outputs into pairs of fixed-width digital pulses (P_x^+, P_x^-; P_z^+, P_z^-). The value of the input is represented as the difference between the density (frequency) of the pulses. We use a single modulator for each input or output in the system. We implement the anti-Hebbian learning rules as:

$$P_{inc} = (P_x^+ \,\&\, P_z^-) \,|\, (P_x^- \,\&\, P_z^+)$$
$$P_{dec} = (P_x^+ \,\&\, P_z^+) \,|\, (P_x^- \,\&\, P_z^-)$$

(a) LMS filter.

(b) Normalized LMS filter performance.
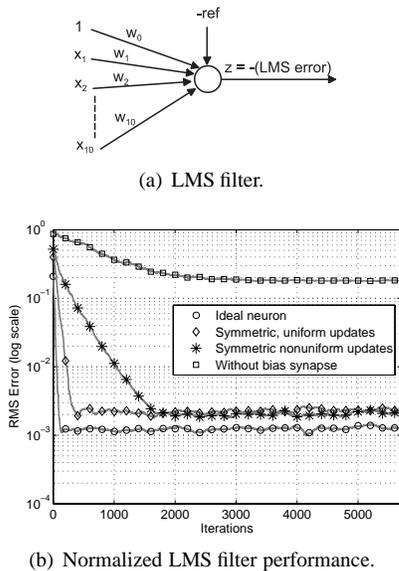
Figure 6. Filter with LMS adaptation.

If the pulse streams are asynchronous and sparse, then the weight updates implemented by the expressions above converge to the negative correlation between x and z, effectively implementing an anti-Hebbian learning rule.

4   A Linear Filter with LMS Adaptation

Using the primitives described in Section 3, we implemented a 10-input LMS adaptive filter in a $0.35\mu m$ CMOS process. Fig. 6(a) shows the architecture of the filter as a neural network adapting with anti-Hebbian learning rules. If the inputs to the neuron are drawn from a tapped-delay line (such as the barrel-shifter analog line described in [8]), then the neuron implements an adaptive FIR filter. Because input offsets in the forward-path multipliers translate into nonzero-mean inputs and degrade the performance of the filter [4], we use a *bias* synapse w_0 with a constant input to compensate for the aggregated value of all the input offsets in the neuron [1]. We train the bias synapse using the same anti-Hebbian learning rule we use to train the other the synapses in the neuron.

Fig. 6(b) shows the evolution of the RMS value of the output error compared to an ideal mathematical implementation of the filter. We normalize the current output to its full-scale value ($20\mu A$ differential) to ease the comparison to the ideal filter. We drew the inputs from a uniform random distribution and trained our filter using a reference generated by a non-adaptive, mathematically ideal filter with the same inputs. We added Gaussian noise to the reference, which resulted in a 60dB signal-to-noise ratio (SNR).
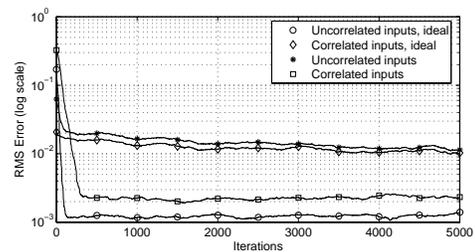
Figure 7. Effect of input correlation.

Trained with this reference, the ideal LMS filter achieves a normalized RMS error of 10^{-3}. The output reference has unity variance, therefore this RMS error is equivalent to a digital resolution of 10 bits. Calibrating the memory cells for symmetric and uniform updates as depicted in Fig. 4(c), the residual error reaches a RMS value of 2×10^{-3}, corresponding to a 9-bit output resolution. The performance of the circuit is mainly limited by the linearity of the forward-path multipliers. Fig. 6(b) also shows the performance of the filter with the memory cells calibrated for local symmetry only. In this case, the residual error is the same, but the convergence time is about four times longer because the learning rate η must be adjusted to stabilize the synapse with the fastest updates [4] (we tune the learning rule globally changing the gain of the pulse modulators). Without a bias synapse, the multiplier offsets limit the performance of the filter to a RMS error of 2×10^{-1} (2 bits).

In the experiment described above, the filter inputs were uncorrelated. Unfortunately, many real-world signals are generated by distributed sensors such as antenna arrays, and show a significant correlation between them. This correlation severely degrades the performance of the LMS algorithm, as shown in Fig. 7 for both the hardware and ideal filter. In this experiment, we mixed the inputs to achieve an average cross-correlation of 0.75. Algorithms such as Recursive Least Squares (RLS) keep an estimate of the inverse of the input correlation matrix and use it to improve the performance of the adaptation, but their computational structure is a poor match for custom VLSI implementations. In the next section, we explore an alternative approach.

5   A Triangular Decorrelating Network

As described in Section 2, we can train an anti-Hebbian synapse to decorrelate two signals. We can use this property to build a decorrelation network and use it as a pre-processing stage to the LMS filter. Fig. 8(a) shows the architecture of the system, based on the direct form of the triangular decorrelating filter described in [1]. Synapse a_{21} decorrelates y_2 from x_1, which is equal to y_1. Synapses a_{31} and a_{32} decorrelate y_3 from x_1 and x_2. Because y_1 and y_2 are linear combinations of x_1 and x_2, y_3 is decorrelated from

(a) LMS filter with adaptive decorrelation stage.

(b) Normalized LMS performance with adaptive decorrelation stage.

(c) Decorrelation stage weights.

(d) LMS filter weights.
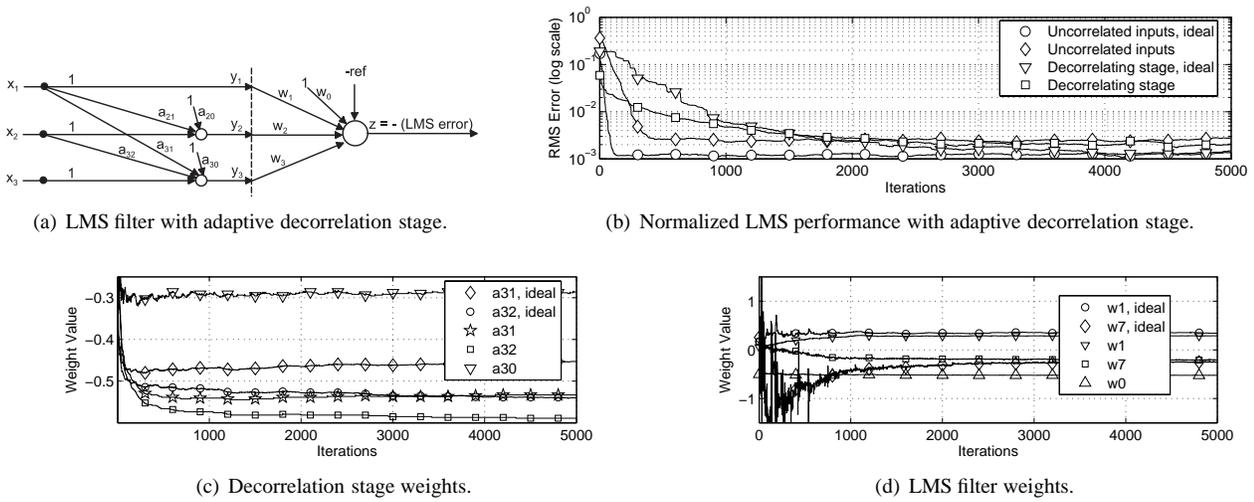
Figure 8. Filter with adaptive decorrelation.

them as well. As a result, the triangular network computes a linear combination of its inputs that minimizes the correlation between the outputs. The LMS filter can now operate on these outputs and achieve better performance.

We tested a 10-input decorrelating filter using the architecture described above, training it with the same correlated inputs discussed in Section 4. Fig. 8(b) compares its performance to a filter trained with uncorrelated inputs. We include a mathematically ideal implementation for comparison purposes. Even though the decorrelation stage and the LMS filter learn concurrently, the residual error reaches the same value as the LMS filter with uncorrelated inputs after less than 2000 iterations, which corresponds to about four times the convergence time of the filter with uncorrelated inputs. This is a substantial improvement over the results shown in Fig. 7. In fact, the VLSI filter converges faster than its mathematically-ideal counterpart. This is partly because the ideal filter converges to a smaller error, but also because the discrete updates of the VLSI filter damp the variance of the weights in the decorrelation network, allowing better LMS performance.

Figs. 8(c) and Fig. 8(d) depict the evolution of selected weights in the decorrelation network and the LMS filter. We observe that the bias weights (a_{30} and w_0) converge to nonzero values to compensate for the multiplier offsets. We also see that the weights converge to different values than their ideal counterparts, to compensate for offsets in the memory cells and analog multipliers. Fig. 8(c) shows that it takes about 1500 iterations for the weights in the decorrelation network to approach their final value, at which point the cross-correlation of the LMS filter inputs is low enough to achieve good performance.

6 Application: Adaptive CDMA despreading

Direct Sequence Code Division Multiple Access (DS-CDMA) is a spread-spectrum technique widely used in wireless data communications. In this scheme, each bit transmitted by a user is encoded using a sequence of shorter binary values (chips), called the *user signature*. Ideally, the cross-correlation between signatures is zero, which makes it possible to detect a single user's message by simply correlating the received signal with the desired user's signature, thus canceling the contribution of other users and recovering only the bit stream of interest. In practice, finite correlation between signatures, unequal power among users and multipath fading causes interference between users, which lead to high error rates, low channel utilization and/or expensive power-control techniques.

An alternative to traditional CDMA detection is to use an adaptive filter to compute an optimal user signature using a training sequence. Because of multiple-user interference and different signal power, the optimal signature is often not the one used to encode the original bit stream (and it is, in fact, not binary). In this section, we show an application of the filters presented in Sections 4 and 5 to adaptively recover user data in a CDMA system. Our filter supports 64 inputs and occupies a die area of 0.25mm ^{2} in a 0.35μm CMOS process, dissipating 200μW of total power.

We configured a CDMA system with 16 simultaneous users and 64-chip signatures. The worst-case effective cross-correlation between user signatures is 0.4 and the worst power ratio between user signals is 10. Fig. 9(a) shows the estimation errors as a result of decoding a 2048-bit message with the traditional method for the user with the

(a) Open-loop BER for user 13.   (b) BER for user 13 with adaptive detection.   (c) BER for 16 users with adaptive detection.
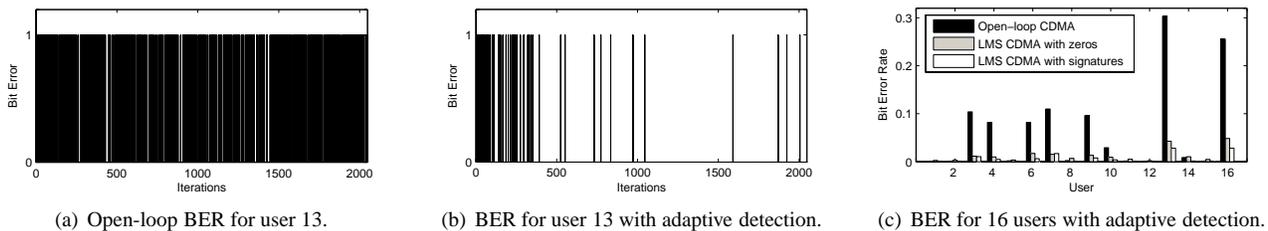
Figure 9. Application to CDMA despreading.

lowest power. The bit-error rate (BER) is 0.3 (30% of the bits were incorrectly detected). We used our adaptive filter to detect the same user's message, using an off-chip comparator to discriminate between bits, and using only the first 700 bits as a training sequence. The performance is greatly improved, reaching a BER of 0.05 including the incorrect estimations during training. The BER after training is only 0.01. Experiments published in [4] show that the output resolution of the 64-input filter is equivalent to 10 bits.

Fig. 9(c) shows the BER for all 16 users traditional CDMA detection and our adaptive filter (including training). In the adaptive case, we obtain slightly better performance if we initialize the weights to the original user signature instead of zeros. Considering only the detection after training, our filter improves the BER performance by a factor between 25 and 114, depending on the user. Other hardware implementations of CDMA detection [9] use less power and area than our filter, but they rely fundamentally on nonadaptive, strictly binary user signatures. Thus, these implementations can only operate in open loop using a traditional detection scheme.

7   Conclusions

We have described analog and mixed-signal primitives for adaptive signal processing in CMOS VLSI using anti-Hebbian learning. Analog-weight storage using synapse transistors enables the use of analog hardware while maintaining accurate weight updates. The adaptive nature of the algorithms compensates for some analog-hardware nonidealities such as gain and offset mismatches. We include on-chip calibration circuitry to tune those features not compensated by the adaptation, such as asymmetric and nonuniform learning rates. We demonstrated the effectiveness of our approach building a 10-input filter that adapts with a pulse-based implementation of the LMS algorithm, achieving an output resolution of 9 bits. We also used anti-Hebbian learning to implement an adaptive decorrelating stage for the LMS filter, greatly improving convergence time and residual error (by more than an order of magni-

tude) in the presence of correlated inputs. Finally, we built a 64-input LMS filter to perform adaptive DS-CDMA despreading, improving the BER over traditional detection by a factor of more than 100 in the presence of user interference and unequal signal power. The circuit die area is 0.25mm^2 and its power dissipation is 200μW in a 0.35μm CMOS process.

Acknowledgments

References

[1] F. Palmieri, J. Zhu, and C. Chang, "Anti-Hebbian Learning in Topologically Constrained Linear Networks: A Tutorial," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 748–761, 1993.

[2] A. Hyvärinen and E. Oja, "Independent Component Analysis: Algorithms and Applications," *Neural Networks*, vol. 13, no. 3, pp. 411–430, 2000.

[3] B. K. Dolenko and H. C. Card, "Tolerance to Analog Hardware of On-Chip Learning in Backpropagation Networks," *IEEE Transactions on Neural Networks*, vol. 6, no. 5, pp. 1045–1052, 1995.

[4] M. Figueroa, S. Bridges, and C. Diorio, "On-chip compensation of device-mismatch effects in analog VLSI neural networks," in *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press, 2005.

[5] Y. Hirai and K. Nishizawa, "Hardware Implementation of a PCA Learning Network by an Asynchronous PDM Digital Circuit," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, vol. 2, 2000, pp. 65–70.

[6] C. Diorio, P. Hasler, B. Minch, and C. Mead, "A Complementary Pair of Four-Terminal Silicon Synapses," *Analog Integrated Circuits and Signal Processing*, vol. 13, no. 1/2, pp. 153–166, 1997.

[7] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[8] M. Q. Le, P. J. Hurst, and J. P. Keane, "An Adaptive Analog Noise-Predictive Decision-Feedback Equalizer," in *IEEE Symposium on VLSI Circuits*, D. Scott and M. Yamashina, Eds. Honolulu, Hawaii, USA: IEEE Solid-State Circuits Society, 2000, pp. 216–217.

[9] T. Yamasaki, T. Fukuda, and T. Shibata, "A Floating-Gate-MOS-Based Low-Power CDMA Matched Filter Employing Capacitance Disconnection Technique," in *IEEE Symposium on VLSI Circuits*, Kyoto, Japan, 2003, pp. 267–270.