Low-Power Speech Processing based upon Floating-Gate Circuits

Paul D. Smith, David Graham, and Paul Hasler
Georgia Institute of Technology, Atlanta, GA 30332-0250.
phasler@ece.gatech.edu.

Abstract—**This paper describes our current efforts towards creating cooperative analog/digital signal processing (CADSP) systems for auditory sensor and signal processing applications. We address resolution issues that affect the choice of signal processing algorithms arriving from an analog sensor. We discuss current analog circuit approaches towards the front-end signal processing by reviewing major programmable analog building blocks and showing how they can be interconnected to create a complete system. We also discuss our current analog signal processing front-end system for speech recognition. Experimental data is presented from circuits fabricated using a 0.5μm nwell CMOS process available through MOSIS.**

New advances in analog VLSI circuits have made it possible to perform operations that more closely reflect those done in DSP applications, or that are desired in future DSP applications. With these advances, analog circuits and systems can be *programmable*, reconfigurable, adaptive, and at a density comparable to digital memories (for example, 100,000+ multipliers on a single chip). Therefore, with both digital and analog signal processing (DSP and ASP, respectively) modalities feasible, more options are now available when designing a signal processing system.

Figure 3a shows our current analog signal processing front-end system for speech recognition, modified from ideal DSP blocks, which is comprised of an analog Cesptrum-like processor [1], a Vector-Quantization stage [2], and a continuous-time HMM block built from programmable analog waveguide stages [3]. The comparable digital algorithms are well understood and, since they are not novel, are not discussed here. The purpose, then, of this paper is to demonstrate the analog options available when deciding where to partition the analog and digital parts of a system. First, we will address resolution issues that affect the choice of signal processing algorithms arriving from an analog sensor. Second, we will discuss the building blocks of current analog circuit approaches towards front-end signal processing and the relationship to modeling biological cochleas. Third, we discuss our current work on a continuous-time mel-frequency cepstrum encoding IC using analog circuits and floating-gate computational arrays (more detail given in [1]), our current work on continuous-time vector quantization IC, and our current work on HMM classification models. These approaches are based upon our previous research in programmable floating-gate arrays and analog filters [4], [5]. Experimental data is presented from circuits fabricated on a 0.5μm n-well CMOS process available through MOSIS.

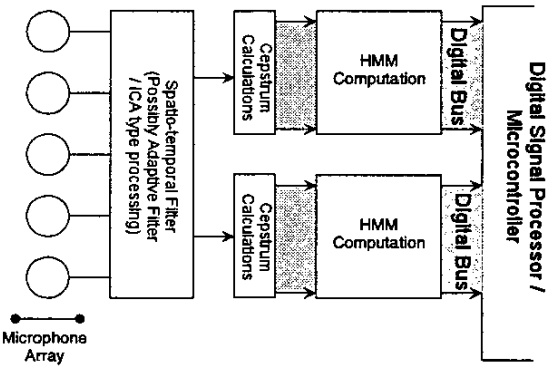

Fig. 1. Block diagram of a potential speech front-end system which takes the outputs of several microphones and could compute phonemes for a higher level digital processing.

I. Analog–Digital Partitioning

By adding functionality to the analog components of our systems, some of the processing requirements on the digital side of the system may be reduced. The placement of this partition will be highly dependent on the engineering constraints for a particular implementation. Analog signal processing is capable of several linear and nonlinear operations [6], [7], [8], [3]. The analog circuits that we present are very small—a characteristic made possible by the floating–gate technology which allows for easy tuning and programming of the circuits. Often the comparison between a custom analog solution and custom digital solution shows a power savings of a factor approximately 10,000.

The primary question once the existance of programmable analog signal processing methods is the effective resolution of these computing systems. The related question is identifying the *cost* of computation at a particular resolution. The computation cost (a wide range of metrics including area, power dissipation, computational delay, required tools, and design time.) of digital computation varies linearly with the required bits of resolution, while, the computation cost of analog computation using a single wire varies exponentially with the required bits of resolution. As a result, computation requiring less resolution than a threshold is less expensive for analog computation, and computation requiring more resolution than a threshold is less expensive for digital computation. One careful study by Sarpeskar [9], shows this threshold to be typically between 8bits to 14bits.

The key in looking at the necessary resolution for either the

0-7803-8104-1/03/$17.00 ©2003 IEEE

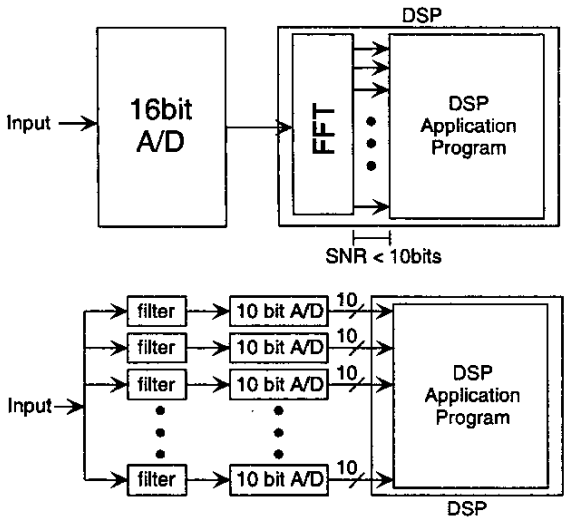

Fig. 2. An example comparison looking at the resulting SNR for two approaches for a particular applications: one case is a purely DSP solution, and the second case is a combined analog-digital solution. A practical example comparing using analog or digital signal processing for a particular output resolution (Signal–to–noise). One common signal processing step with incoming sensor data is taking an FFT, or equivalent Fourier based algorithm. For DSP computation, we would require a 16bit A/D converter to get some output channels at 10bit resolution. For ASP computation, we would require a bank of bandpass filters with 10bits of Signal-to-noise ratio coupled with a bank (or multiplexed) 10bit A/D converter to get the output channels at 10bit resolution. Both *analog* systems have similar design complexity. These computations are transparent (in resolution) to the engineers developing the remainder of the algorithm, and therefore tradeoffs could be made at these levels. In the end, either approach would give similar amount of information at each output channel.

analog or digital signal processing parts depends heavily on the amount of the incoming information and resolution needed to represent it. Figure 2 shows an example comparing how one might apply these results. One common signal processing step with incoming sensor data is taking an FFT, or equivalent Fourier based algorithm. Both *analog* systems have similar design complexity, because the design complexity of a 16-bit A/D converter is exponentially more difficult than the design complexity of a single or multiple 10-bit A/D converters. In terms of resolution, these trade–offs at the A/D conversion level will appear transparent to the engineers developing the remainder of the algorithm. When modeling analog signal processing resolution, typically measured in signal-to-noise ratio (SNR), one must consider the particular circuit effects and continuous-time signal processing to get an accurate estimate. Simply treating analog components as fixed-point arithmetic with finite register effects will always underestimate the SNR of actual computation.

In summary we can say that doing more in digital hardware generally increases flexibility and increases power consumption and, beyond a certain point, can yield increased accuracy. Whereas, analog implementations of parts of a system generally result in significant power savings and space savings at the expense of flexibility.

II. Programmable Analog CMOS Technology for Analog Signal Processing

Programmable and reconfigurable analog signal processing enables a wide range of applications only thought possible in digital signal processing environments. Our Programmable Analog CMOS technology is based on floating-gate circuits [10], which is a modified EEPROM technology that allows for storage and simultanious computation through a transistor element. These floating–gate transistors provide nonvolatile storage (floating-gate surrounded by high-quality Silicon Dioxide), compute a product between this stored weight and the inputs (through capacitive coupling into the floating gate), allow for programming that does not affect the computation (using a combination of electron tunneling and hot–electron injection), and adapt due to correlations of input signals. These single transistor learning synapses [11], [12], named because of the similarities to connectionist synapses, lead to a technology called analog computing arrays [4]. Using these floating-gate analog arrays, we are able to realize a wide range of programmable and adaptive [13] analog signal processing systems.

Routinely programming thousands to millions of floating–gate elements requires systematic, automated methods for programming. Fast programming is critical to mass production programming of large arrays of floating-gate devices. We have developed a standard method using standardized custom hardware and algorithms that allows for flexibile floating-gate array programming over a wide range of IC processes and allows for nearly transparent operation to the user [5], [4], [14]. Our programming scheme minimizes interaction between floating–gate devices in an array during the programming operation. This scheme also measures results at the circuit's operating condition for optimal tuning of the operating circuit. Most elements are currently programmed in roughly 10 iterations; the injection time for a single iteration is a constant typically between $10\mu s$ and $100\mu s$. Once programmed, the floating-gate devices retain their channel current in a non-volatile manner.

We commonly use several basic circuit elements for our auditory signal processing structures. Floating–gate circuit techniques enable using these circuits for a wide range of signal processing functions. We have been using coupled bandpass IC filter models for cochlear modeling, which are designed to be used for front-end signal processing [15]. The spectrum decomposition is done using differential C^4 second-order-section bandpass filters [15]. Floating gate pFETs are used to set the bias currents that control the corner frequencies of the filter. Therefore, the spacing of the bandpass filters is arbitrary because each can be programmed to have a desired high-frequency corner and low-frequency corner [15]. As a bandpass filter array, the C^4SOS structure is not cascaded as in cochlea models [6], therefore eliminating the typical distortion or noise accumulation. We also utilize peak detector circuits to estimate the magnitude of output signals at a particular timescale. , often the output signals of these bandpass filter elements, We program the peak detectors to the desired frequency response of each frequency band; When connected to arrays of filter elements, each peak detector has

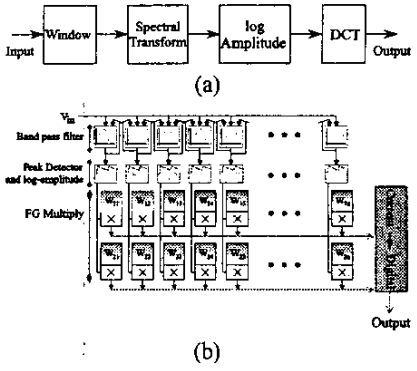

Fig. 3. (b) The traditional cepstrum computation as performed in digital circuitry. (c) Block diagram of a floating-gate system to perform cepstrum front-end computation for speech processing systems. The system contains 32 frequency taps that can be spaced arbitrarily by programming the corner frequencies for the bandpass filter banks. The peakdetectors provide a power spectrum of the input signal for any given time slice.

an individually programmable corner frequency. We also use analog differential multiplier [4] that multiplies the incoming differential voltage signal with a stored differential weight. We program the positive and negative weights by setting programmable floating-gate charge.

III. Continuous-Time Cepstrum

The Mel-cepstrum is often computed as the first stage of a speech recognition system [16]. Figure 3c shows the block diagram for the analog cepstrum which is an approximation to either the mel-cepstrum or cepstrum. The output of each filter contains information similar to the short-time Fourier transform and can likewise be assumed to represent the product of the excitation and vocal-tract within that filter band. The primary difference is that the DSP mel-cepstrum approximates the critical band log frequency analysis of the human ear by combining DFT bands while the analog system actually performs a critical band-like analysis on the input signal. Thus higher frequency critical band energies are effectively computed using shorter basis functions than the lower frequency bands and are similar with in the human auditory system and is better suited to identifying transients. We present a detailed discussion on the signal processing foundation of analog and digital Mel-Cepstrum computations elsewhere [1]; the primary difference between the analog and digital computation approaches is in the frequency decomposition ad amplitude dection method. We compute a DCT on these results using a matrix multiply using arrays of floating-gate circuits where each row of the matrix is another DCT basis vector.

This cepstrum processor can act as the front-end for larger digital or analog speech processing systems. Figure 4 shows experimental results from different stages of our Cepstrum computation. The 14 output taps of our analog cepstrum computation closely agrees with the DSP equivalent algorithm when starting from a set of bandpass filter. Early data from a related project gives confidence that this approach will improve the state of the art at a given power dissipation level [17]. Recent experimental and computational studies have shown 98% to 99% percent recognition on TI digit databases.

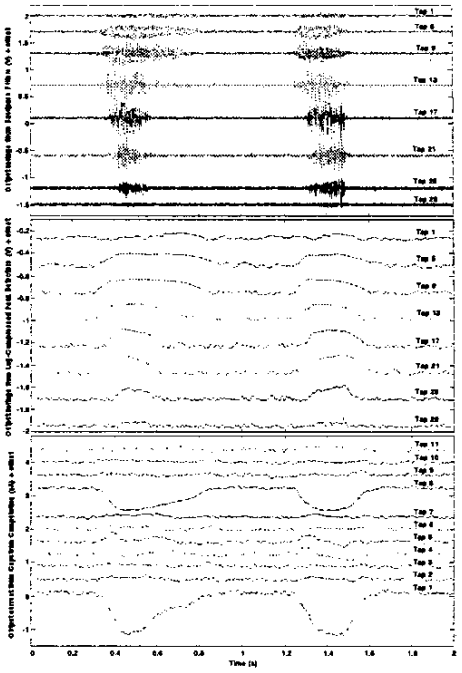

Fig. 4. Cepstrum system output. The system input is a sequence of speech using a standard speech database; each letter or phrase is separated by a short period of silence. There are 12 continuous cepstrum coefficients calculated for this section of speech and more coefficients is only a matter of chip area since the calculation is performed in parallel analog circuits. From the graph one can see the two distinct periods of speech.

IV. Continuous-Time VQ

In this section, we provide an overview of Vector quantization (VQ), which is typically used in data compression and in classifying signals to symbols [18]. A VQ system will compute how far away a particular input vector is from the desired target vectors, and pick the code vector that is *closest* to the input vector. For VQ some information is lost in the representation, but the goal is that it should be a sufficient representation for the problem at hand.

VQ computes the closest input vector by choosing an appropriate distance metric. The question is how to choose the distance metric between the incoming vector signal and the desired or target mean value for these signals. Figure 5 shows the circuit and measured data from the VQ classifier array [2]. Each cell in the array compares the value of that column's input to the value it has memorized; the output current flows out of the V_{out} node. We will use a metric close to an ideal Guassian metric using sech(·) function, based upon a floating-gate circuit variation on the bump circuit [19], which compares the two inputs to this circuit; this cell returns a large current if the two values match (minimal difference). This system outputs a measure of the similarity; therefore, the outputs of all the elements can be added (by KCL) and the largest output is the vector with the maximum similarity. One can also build a a metric based upon a rough exponential function of this metric, which effectively turns the summation into a product, resulting in a more Gaussian-like formulation

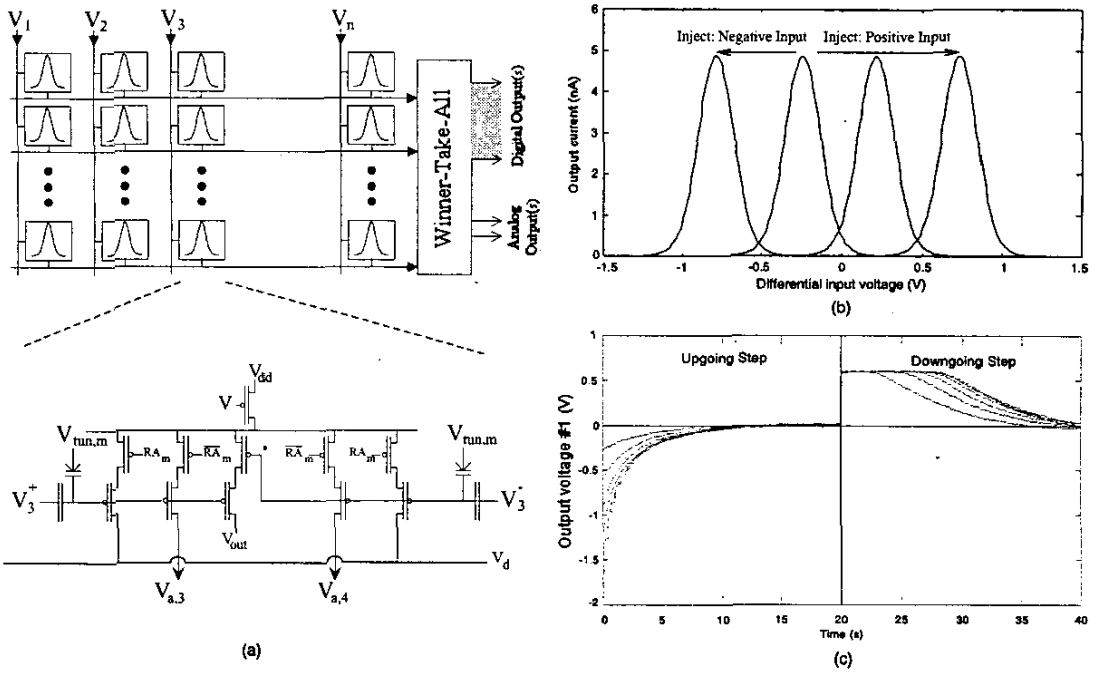

Fig. 5. Basic circuit, architecture, and measurements from the VQ circuit. (a) The core cell is built from a floating-gate bump circuit, which allows the target mean value to be stored and subtracted from the broadcasted input signal. (b) Results from programing the VQ circuit. We see that output current (experimental measurements) of the middle leg of the bump circuit reaches a maximum at its center value, and falls off exponentially as one moves from that center value. This output current is summed together with the output from other bump circuits. We use the RA_m signal to select between adaptation or computation / programming along a given row; if only programming and computation are required, then the circuit can be significantly reduced. We reconfigure the VQ circuit so that it fits within the standard floating-gate programming architecture and algorithms [14]. We reset the floating-gate charge using electron tunneling, and program positive or negative offsets using hot-electron injection. If we inject the floating-gate associated with the positive input terminal, then we increase the offset, If we inject the floating-gate associated with the negative input terminal, then we decrease the offset. (c) The results of adapting the input signal mean. The Common-Mode FeedBack (CMFB) circuitry is switched in from the bottom of the array. We show experiemental measurements showing the convergance of the floating-gate bump element with the CMFB circuitry. We show one drain voltage when connected to the CMFB circuitry; if the drain voltage reaches equilibrium between the operating rails, then the circuit has converged to the signal mean.

We utilize floating gates elements [20] at the inputs to provide the ability to store and subtract off the each cell's mean value. Setting the floating-gate charge establishes the mean value as well as eliminating the mismatch between the two-transistor pairs [2]. Figure 5b shows that the means in a VQ array can be programmed to an arbitrary level. Using this approach, we have estabilished approaches to program arrays of floating-gate elements. Figure 5c shows the circuit and architecture for our adaptive VQ system [2], [20]; We adapt the floating-gate charge to the mean of the input signal. The sum of these current outputs are sent through a Winner-Take-All circuit that outputs the N largest results [21].

V. Continuous-Time HMM

A Hidden Markov Model (HMM) can be viewed as a state machine in which the states themselves are not observable, but an output, whose statistics are determined by the current state, is observable. For example, in using an HMM to model speech production the states are the desired utterance (phonemes and words) and the observations are features of the audio signal produced by the talker. The audio features are determined by the spoken word but they are randomly distributed since each time that same word is spoken it will sound a little different.

For recognition problems, the goal is to estimate the underlying states of the state machine based on the observed outputs. For speech recognition, the HMM decoder takes as inputs the signal statistics or features and generates a probability of occurrence on any one of a set of speech "symbols." These "symbols" can be grouped over multiple short windows to generate larger symbols, such as phonemes or words. The ongoing input train of symbols is used to map a path through a probability trellis for the larger blocks [22].

We want to revisit the mathematical modeling for HMM classification and translate the formulation to be more easily implemented in continuous-time analog hardware. Starting from the stereotypical speech production HMM (Fig. 6), rewriting the resulting likelihood update equation in continous-time, and expanding in first order Taylor expansion, results in the family of differential equations:

$$\tau \frac{d\phi_i(t)}{dt} + \left(\frac{1}{b_i(t)} - 1\right)\phi_i(t) = a_i\left(\phi_{i-1}(t) - \phi_i(t)\right). \quad (1)$$

where ϕ_i represents the current state at time (t) or time index (n), τ is the time between index (n) and (n-1) for the discrete time formulation, δ is the distance between node (i) and (i-1), b_i is the input to the current state and $a_i(n)$ are the transition probabilities between adjacent states. Expanding this

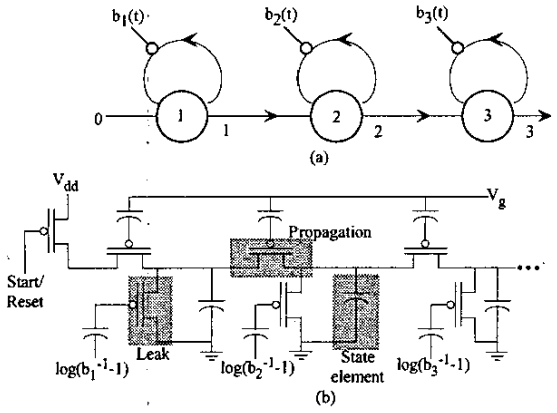

Fig. 6. Circuit design for our HMM branch element. (a) State diagram for a branch of an HMM classifier network typical in speech recognition. (b) Circuit diagram of our HMM network. Our branch element design is based upon diffusor elements to perform the classical HMM calculation. In this framework, each branch element exhibits wave propagation. We can build these branch elements into an array for classification. The shaded areas show the transistors corresponding to terms in the continuous HMM equation.

equation in position results in a wave propagating PDE, where a_i modifies the velocity of the resulting wave, and the input probabilities, $b_i(t)$, set the decay rate of the $\phi_i(t)$ values. The velocity of this wave is $a_i\delta/\tau$.

Figure 6b shows our HMM branch implementation and HMM network implementation. For our implementation, we look at HMMs as propogating waves and the probabilities relate to the velocity of propogation. We can build compact, programmable wave-propagating structures using a floating-gate programmable diffusor circuit with each voltage programmed such that we get either forward or backward propagating waves with minimal diffusion components. A comparison with the terms in (1) shows a direct correspondance to the circuit implementation as

- State Element → Capacitor (storage),
- Wave propagation → Propagation transistor, and
- Decay term → Leak transistor ($b_i(t)$ input).

We implement the $b_i(t)$ terms through log-compressed voltage signals modifying the floating-gate input of the i^{th} leak transistor. We represent $\phi_i(n) = I_i/I_{so}$ for I_i as the leakage current at that node for the reference level of $b(t)$, where I_{so} as a reference current for the array of elements. Although $b_i(t) = 0$ requires ∞ leakage conductance and $b_i(t) = 1$ requires zero leakage conductance, we practically do not reach these limits, although the variation between these two conductances is typically 9 orders of magnitude.

REFERENCES

[1] Paul Smith, Matt Kucic, Rich Ellis, Paul Hasler, and David V. Anderson, "Cepstrum frequency encoding in analog floating-gate circuitry," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Phoenix, AZ, May 2002, vol. IV, pp. 671–674.

[2] Paul Hasler, Paul Smith, Chris Duffy, Christal Gordon, Jeff Dugger, and David Anderson, "A floating-gate vector-quantizer," in *IEEE Midwest Circuits and Systems*, Tulsa, OK, Aug. 2002.

[3] P. D. Smith and P. Hasler, "Analog speech recognition project," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Orlando, FL, 2002, vol. 4, pp. 3988–3991.

[4] Matt Kucic, Paul Hasler, Jeff Dugger, and David V. Anderson, "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *2001 Conference on Advanced Research in VLSI*, Erik Brunvand and Chris Myers, Eds. IEEE Computer Society, March 2001, pp. 148–162.

[5] M. Kucic, A. Low, P. Hasler, and J. Neff, "A programmable continous-time analog fourier processor," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 90–99, Jan. 2001.

[6] Carver Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, MA, 1989.

[7] Carver A. Mead, "Neuromorphic electronic systems," *IEEE Proceedings*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.

[8] Paul Hasler and David V. Anderson, "Cooperative analog-digital signal processing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Orlando, FL, May 2002, vol. IV, pp. 3972–5.

[9] Rahul Sarpeshkar, *Efficient precise computation with noisy components: extrapolating from an electronic cochlea to the brain*, PhD thesis, California Institute of Technology, Pasadena, CA, 1997.

[10] P. Hasler and T. S. Lande, "Special issue on floating-gate devices, circuits, and systems," *IEEE Journal of Circuits and Systems*, vol. 48, no. 1, Jan. 2001.

[11] P. Hasler, C. Diorio, B. A. Minch, and C. A. Mead, *Advances in Neural Information Processing Systems 7*, chapter Single transistor learning synapses, pp. 817–824, MIT Press, Cambridge, MA, 1995.

[12] Paul Hasler, Bradley A. Minch, Jeff Dugger, and Chris Diorio, *Learning in Silicon*, chapter Adaptive Circuits and Synapses Using pFET Floating-Gate Devices, pp. 33–65, Kluwer Academic, 1999.

[13] Tyson S. Hall Paul Hasler and David V. Anderson, "Field-programmable analog arrays: A floating-gate approach," in *12th International Conference on Field Programmable Logic and Applications*, Montpellier, France, Sept. 2002.

[14] P. Smith, M. Kucic, and P. Hasler, "Accurate programming of analog floating-gate arrays," in *International Symposium on Circuits and Systems*, Phoenix, AZ, May 2002, vol. 5, pp. 489–492.

[15] David Graham and Paul Hasler, "Capacitively-coupled current conveyer second-order section for continuous-time bandpass filtering and cochlea modeling," in *Proceedings of the International Symposium on Circuits and Systems*, 2002, vol. to appear.

[16] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan, 1993.

[17] Todd M. Massengill, Denise M. Wilson, Paul Hasler, and David Graham, "Emperical comparison of analog and digital auditory perprocessing for automatic speech recognition," in *Proceedings of the International Symposium on Circuits and Systems*, Phoenix, AZ, May 2002.

[18] J. Schurmann, Ed., *Pattern Classification, A Unified View of Statistical and Neural Approaches*, John Wiley and Sons, Inc., New York, NY, 1996.

[19] T. Delbruck, "Bump circuits for computing similarity and disimilarity of analog voltages," in *Proceedings of the International Joint Conference on Neural Networks*, Seattle, WA, 1991, vol. I, pp. 475–479.

[20] Paul Hasler, "Continuous-time feedback in floating-gate mos circuits," *IEEE Transactions on Circuits and Systems*, vol. 48, no. 1, pp. 56 – 64, Jan. 2001.

[21] John Lazzaro, S. Ryckebusch, M.A. Mahowald, and Carver A. Mead, *Advances in Neural Information Processing Systems 1*, chapter Winner-take-all networks of O(N) complexity, pp. 703–711, Morgan Kaufman Publishers, San Mateo, CA, 1988.

[22] Steve Ranals, Nelson Morgan, Harve Bourlard, Michael Cohen, and Horacio Franco, "Connectionist probability estimators in hmm speech recognition," in *IEEE Transactions On Speech And Audio Processing*, January 1994, vol. 2, pp. 161–174.