

Analog Floating-Gate, On-Chip Auditory Sensing System Interfaces

Paul Hasler, *Senior Member, IEEE*, Paul D. Smith, *Member, IEEE*, David Graham, *Student Member, IEEE*, Rich Ellis, and David V. Anderson, *Senior Member, IEEE*

Abstract—This paper describes our current efforts toward creating cooperative analog–digital signal-processing 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 toward 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 IC approaches using this technology for noise suppression, as well as our current analog signal-processing front-end system for speech recognition. Experimental data is presented from circuits fabricated using a 0.5 μm nwell CMOS process available through MOSIS.

Index Terms—Analog cepstrum, analog hidden Markov model (HMM), analog signal processing (ASP), analog speech enhancement in noise, analog speech recognition, analog vector quantization (VQ), auditory signal processing, floating-gate circuits.

NEW advances in analog very large-scale integration (VLSI) circuits have made it possible to perform operations that more closely reflect those done in digital signal-processing (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 DSP and analog signal-processing (ASP) modalities feasible, more options are now available when designing a signal-processing system.

In this paper, we will discuss ASP in the context of several audio-processing systems. 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 toward front-end signal processing and the relationship to modeling biological cochleas. Third, we will discuss our current IC

Manuscript received November 8, 2002; revised September 27, 2003. The associate editor coordinating the review of this paper and approving it for publication was Prof. P. M. Sarro.

P. Hasler, D. Graham, and D. V. Anderson are with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA (e-mail: phasler@ece.gatech.edu; dgraham@ece.gatech.edu; dva@ece.gatech.edu).

P. D. Smith was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA. He is now with Gtronix, Inc., Fremont, CA 94538 USA.

R. Ellis is with Medtronic, Tempe, AZ 85281 USA.
Digital Object Identifier 10.1109/JSEN.2005.854488

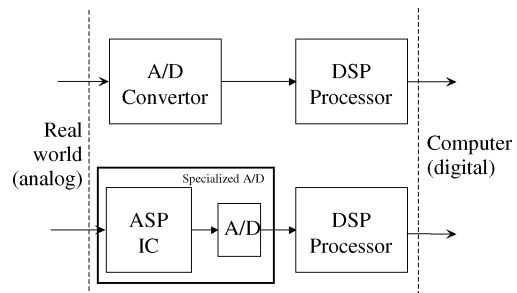


Fig. 1. Cooperative analog–digital signal processing (CADSP) applied toward auditory sensor processing. We assume the typical model of signals coming from real-world sensors, which are analog in nature, that need to be utilized by digital computers. Our approach is to perform some of the computations using ASP, requiring simpler analog–digital converters, and reducing the computational load of resulting digital processors.

approaches using this technology for noise suppression using Wiener gain–control algorithms [1]. Finally, we describe our current ASP front-end system for speech recognition. Experimental data is presented from circuits fabricated using a 0.5- μm n-well CMOS process available through MOSIS.

I. ANALOG–DIGITAL PARTITIONING

One major question lies in where to partition the analog–digital boundary, as shown in Fig. 1(a), in order to enhance the overall functionality of a system by utilizing analog and digital computation in a mutually beneficial way. 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 and a full treatment of the tradeoffs is beyond the scope of this paper and, indeed, much of the tradeoffs are yet to be investigated. However, in this section, we present a brief discussion of power, size, and resolution.

A. Power and Space

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. As a reference point, each analog system discussed in this paper occupies less than 1.5 mm^2 in a 0.5- μm process. These circuits are generally operated in subthreshold mode, yielding tremendous power savings. Again, as a point of reference, each analog system discussed in this paper consumes less than 1 mW of power. For the noise suppression system presented herein, a comparison with

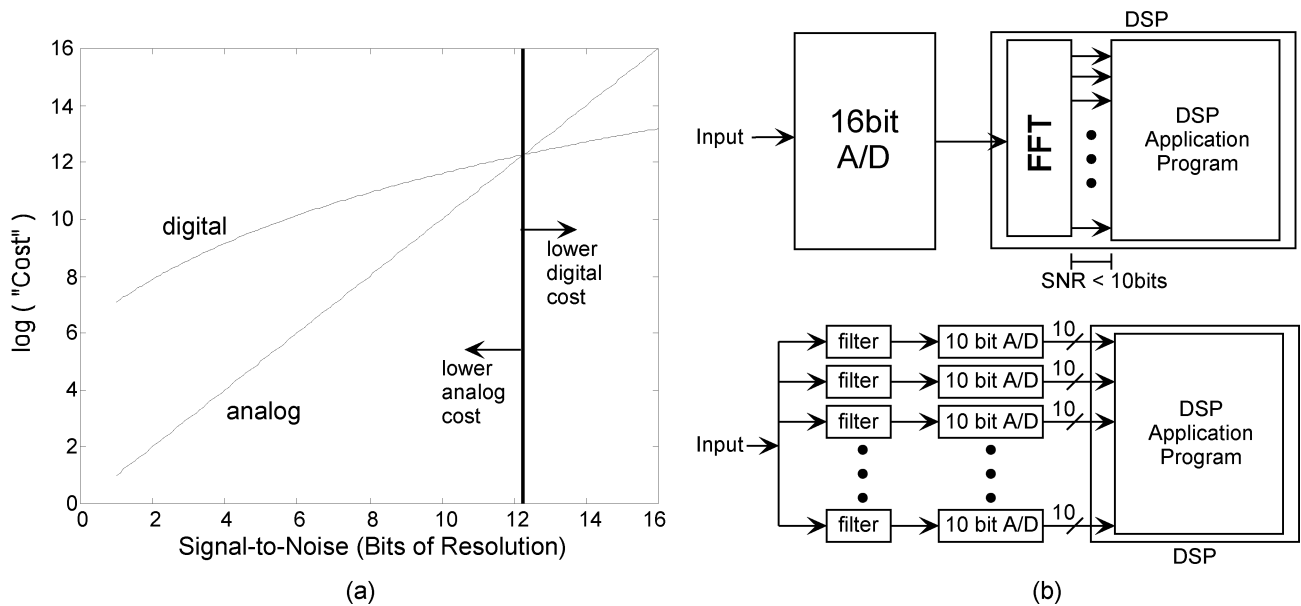


Fig. 2. Guidelines on using ASP or DSP depending upon required resolution (signal-to-noise). (a) As discussed elsewhere [2], the computation cost of digital computation varies linearly with the required bits of resolution, while the computation cost of digital computation varies exponentially with the required bits of resolution. This threshold is typically between 8 bits to 14 bits, depending upon the particular application. (b) 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 DSP for a particular output resolution (signal-to-noise). One common signal-processing step with incoming sensor data is taking an fast Fourier transform, or equivalent Fourier-based algorithm. For DSP computation, we would require a 16 bit A/D converter to get some output channels at 10-bit resolution. For ASP computation, we would require a bank of bandpass filters with 10 bits of SNR coupled with a bank (or multiplexed) 10-bit A/D converter to get the output channels at 10-bit 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.

a state-of-the-art, optimized DSP solution shows a power savings of a factor of 10 000.

B. Signal-to-Noise Versus Cost

ASP is capable of several linear and nonlinear operations [3]–[6]. Even if ASP is capable of several important functions, and is programmable, the primary question is the effective resolution of these computing systems. The related question is identifying the *cost* of computation at a particular resolution. Fig. 2(a) shows a typical plot of signal-to-noise as bits of resolution versus the net cost [2]. One gets similar results when computing cost using a wide range of metrics involving area, power dissipation, computational delay, required tools, expenses associated with the design and manufacture, and design time. The computation cost 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 [2], shows this threshold to be typically between 8–14 bits.

The key in looking at the necessary resolution for either the analog or DSP parts depends heavily on the amount of the incoming information and resolution needed to represent it. Fig. 2(b) shows an example comparing how one might apply these results. One common signal-processing step with incoming sensor data is taking an (fast Fourier transform) FFT, or equivalent Fourier-based algorithm. Both *analog* systems

have similar design complexity, because the design complexity of a 16-bit analog–digital (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 tradeoffs at the A/D conversion level will appear transparent to the engineers developing the remainder of the algorithm. When modeling ASP 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

Programmable and reconfigurable ASP enables a wide range of applications only thought possible in DSP environments. Our programmable analog CMOS technology is based on floating-gate circuits [7], which is a modified EEPROM technology that allows for storage and simultaneous 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

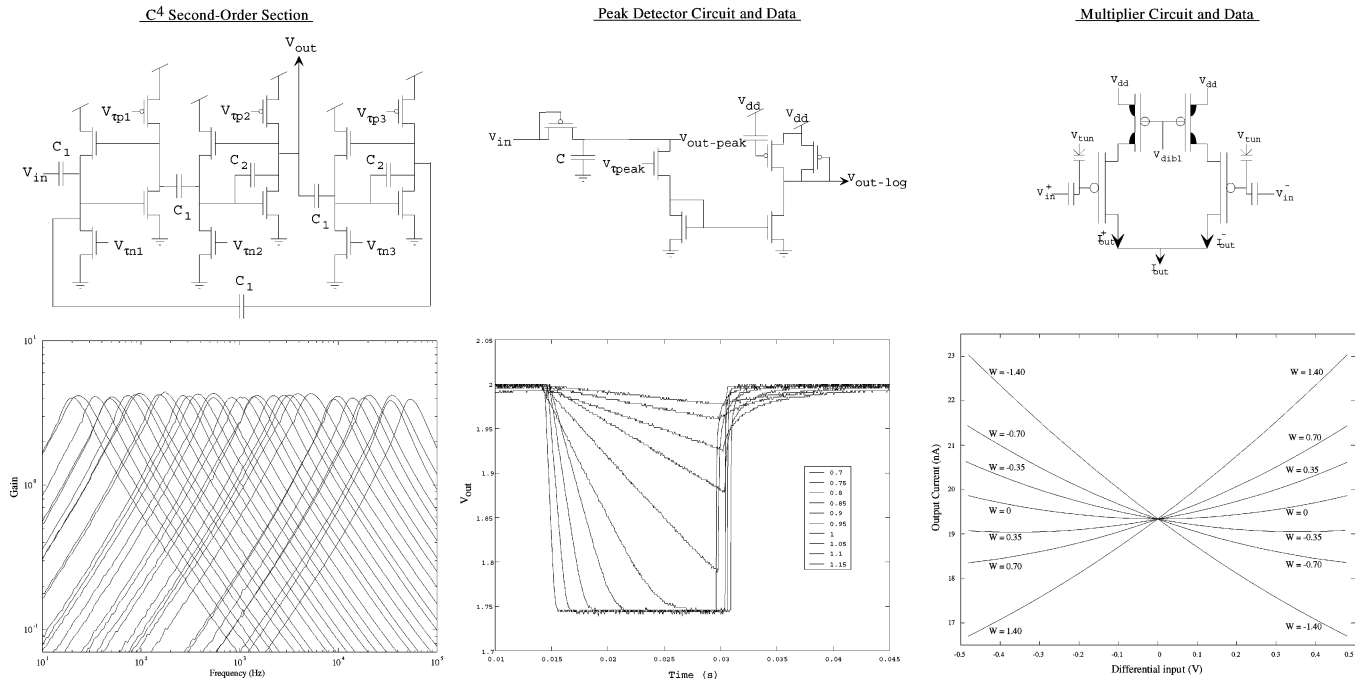


Fig. 3. Typical circuit elements used in auditory signal processing. Second-order section: Floating-gate C^4 second-order-section and its corresponding frequency response. The high- and low-corner frequencies can be independently tuned for each filter bank. Arbitrarily programmable corner frequencies allow these filters to be spaced linearly, octave, logarithmically, or any other values desired by the user. Floating-gate multiplier: Differential floating-gate multiplier structures multiply two differential signals by constant factors that are stored on the floating-gate elements. Floating-gate peak detectors: The frequency response of the peak detector is controlled by a bias voltage which controls the gate of nFET M3. This element sets a constant resistance and the total R, C value shifts the high corner frequency. The frequency response is shown for different values of v_{τ} .

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 [8], [9], named because of the similarities to connectionist synapses, lead to a technology called analog computing arrays [10]. Using these floating-gate analog arrays, we are able to realize a wide range of programmable and adaptive [11] ASP 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 flexible floating-gate array programming over a wide range of IC processes and allows for nearly transparent operation to the user [12], [10], [13]. 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 ten iterations; the injection time for a single iteration is a constant typically between 10–100 μs . Once programmed, the floating-gate devices retain their channel current in a nonvolatile manner.

III. SIGNAL-PROCESSING CIRCUITS

We commonly use several basic circuit elements for our auditory signal-processing structures. We will examine the circuits shown in Fig. 3 in the following sections. Floating-gate circuit techniques enable using these circuits for a wide range of signal-processing functions.

A. Frequency Decomposition

We have been using coupled bandpass IC filter models for cochlear modeling, which are designed to be used for front-end signal processing [14]. The spectrum decomposition is done using differential C^4 second-order-section bandpass filters [14]. For simplicity, only one half of the differential structure is shown in Fig. 3(a). 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 [14]. The programming structure allows each corner frequency to be accurately and precisely tuned throughout the entire filter bank. The corner frequencies were programmed within five percent.

As a bandpass filter array, the $C^4\text{SOS}$ structure is not cascaded as in cochlea models [3], therefore eliminating the typical distortion or noise accumulation. In speech, particularly in noisy environments, the signal power is more evenly distributed across a broad frequency range than a simple tone and, therefore, allowing for larger input amplitudes with minimal output distortion (higher system SNR). As a result, we typically have signal amplitudes through each filter that are 10–30 mV or less for input amplitudes between 0.25–1 V, resulting in harmonic distortion through the system less than -30 dB at each tap; differential circuits will further reduce these effects.

B. Amplitude Detection

The magnitude of each spectrum passes through a peak detector stage to produce a constant magnitude output. This magnitude is similar to taking the power spectrum density or real

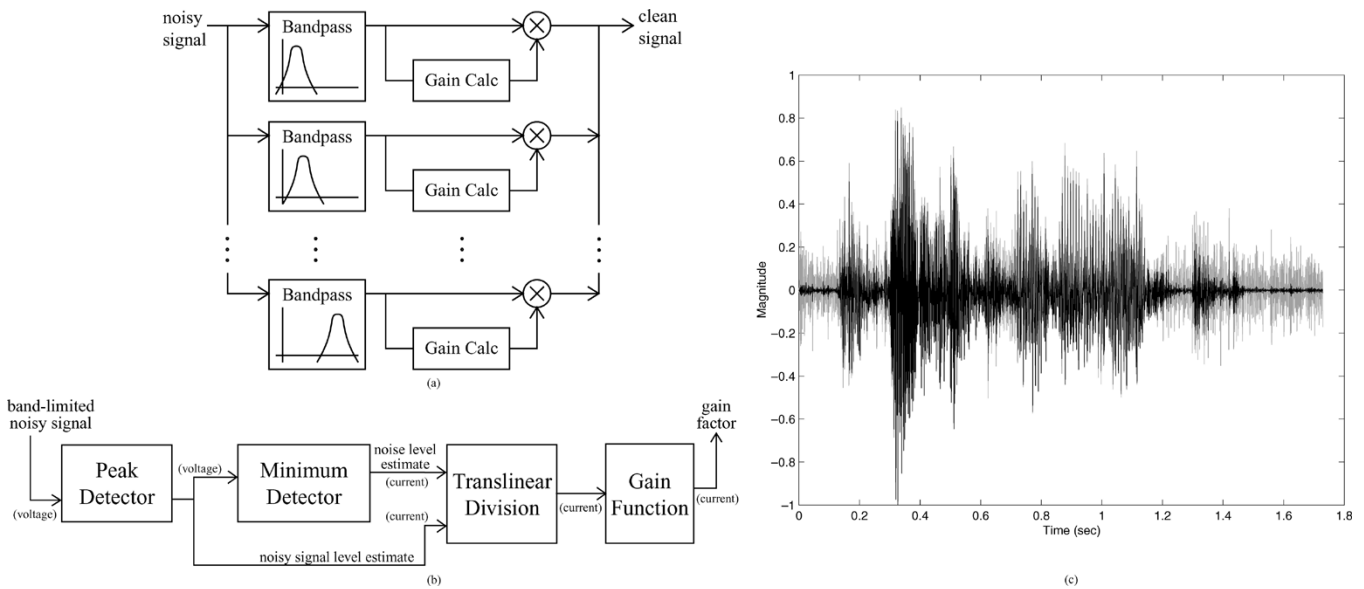


Fig. 4. Our continuous-time noise suppression system. (a) The overall structure of the system. The incoming noisy signal is divided into exponentially-spaced frequency bands using C^4 second-order sections. Next, the optimal gain (gain calculation block) for each band is computed. If the band has sufficient estimated SNR, then the signal passes through with maximal gain, otherwise the gain is reduced dependent upon the estimated SNR in that particular band. The resulting gain factor is multiplied with the band-limited noisy signal to produce a band-limited “clean” signal. Finally, the output of all of the bands are summed to reconstruct the signal with the noise components significantly reduced. (b) Details of the gain calculation block. (c) Experimental measurements of noise suppression in one frequency band. The light gray data is the subband noisy speech input signal; the black waveform is the corresponding subband output, after the gain function has been applied. The noise-only portions of the signal have been significantly attenuated while a lesser attenuation is applied appropriately to the speech+noise portions.

spectrum of an input signal. The circuit is shown in Fig. 3(b). We program the peak detectors to the desired frequency response of each frequency band. The floating-gate transistor on the output provides an offset current to set the dc output voltage. Each peak detector has an individually programmable corner frequency. Because the output magnitude is continuous, this allows us to capture additional high frequency content within each band. The peak detector programming blocks are isolated similarly to the C^4 s. The entire bank is treated as a single row and within that row the individual elements are accessed by column. Control circuitry on the rows and columns ensures isolation.

C. Weighted Multiplication

Fig. 3 shows our analog differential multiplier 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. These values are programmed to arbitrary values; their differential operation requires each pair to have a dc bias current.

IV. NOISE SUPPRESSION FOR SPEECH ENHANCEMENT

Audio signal enhancement by removing additive background noise has recently received increased attention with the prosperity of portable communication devices. We use a real-time, low-power technique for noise suppression in the continuous-time domain [Fig. 4(a)]. The goal is to design a real-time system that generates some optimal estimate of the actual signal from an additive mixture of signal and noise. We assume that the additive noise is stationary over a long time period relative to the short term nonstationary patterns of normal speech. We separate the noisy signal into 32 bands that are exponentially spaced

in frequency [Fig. 3(a)]. Then, a gain factor is calculated based on the envelopes of each observed subband signal and subband noise signal, which serves to estimate the SNR of the incoming signal in that band. Bands with low SNR are attenuated, and bands with high SNR result pass through. The gain factor is multiplied with the band-limited signal and summed to reconstruct the full-band signal estimate, with the additive noise components suppressed.

The first step in the gain calculation algorithm [Fig. 4(b)] estimates both the levels of the noisy signal and the noise (using a minimum statistics approach). Because one can not accurately determine the actual signal component of the incoming signal, the noisy signal is accepted as a reasonable estimate. The noisy signal envelope is estimated using a peak detector circuit, and the noise level is estimated using a minimum detector operating on the signal envelope at a slower rate. Currents that are representative of the noisy signal level and the noise level are divided (using a translinear division circuit) to create an output current as an estimate for SNR. An optimal weiner gain function is applied (computed in current mode) to the SNR current to calculate each gain factor. We present motivation for these concepts and the details of the signal-processing theory, the algorithm for gain calculation, and the circuit elements that perform these functions elsewhere [15], [1].

Fig. 4(c) shows a noisy speech signal that has been processed by the components in our system. The system is effective at adaptively reducing the amplitude of noise-only portions of the signal while leaving the desired portions relatively intact. Initial perceptions between the initial played signals and resulting outputs by these researchers show similar improvements. Any noise or distortion created by the gain calculation circuits minimally affects the output signal because these circuits are not directly

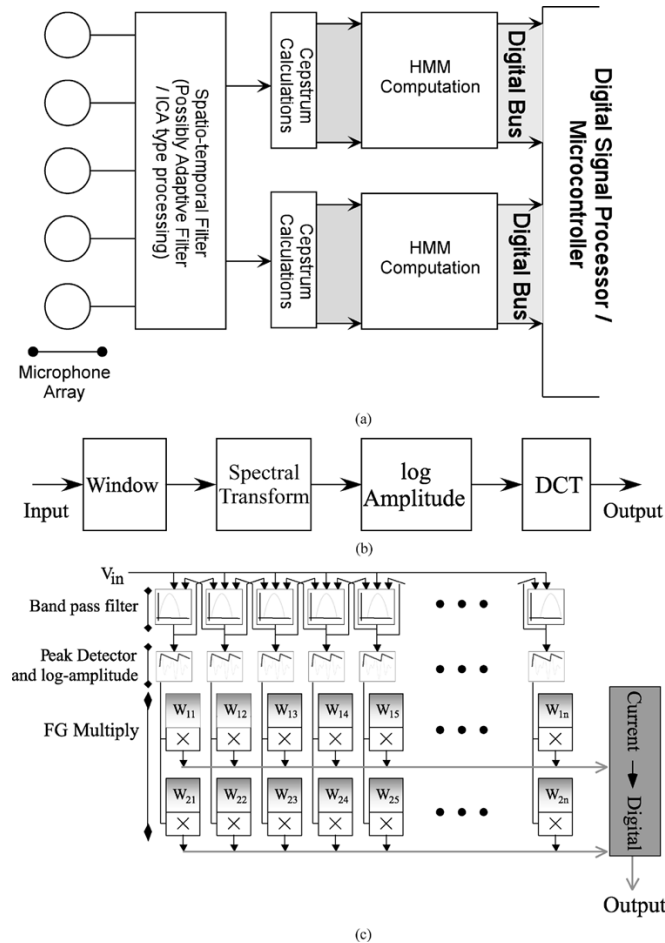


Fig. 5. (a) Block diagram of a potential speech front-end system, which takes the outputs of several microphones and could compute phonemes for a higher level digital processing (b) The traditional cepstrum computation as performed in digital circuitry. (c) Block diagram of a floating-gate system to 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.

in the signal path. While the bandpass filters and the multipliers will inject a certain amount of noise into each frequency band, this noise will be averaged out by the summation of the signals at the output of the system.

V. ANALOG SIGNAL-PROCESSING FRONT END FOR SPEECH RECOGNITION

Fig. 5(a) shows our current ASP front-end system for speech recognition, modified from ideal DSP blocks, which is comprised of an analog Cepstrum-like processor [16], a vector-quantization (VQ) stage [17], and a continuous-time hidden Markov model (HMM) block built from programmable analog waveguide stages [6]. This section discusses our current work on a continuous-time mel-frequency cepstrum encoding IC using analog circuits and floating-gate computational arrays (more detail given in [16]), and the following section discusses our current work on continuous-time VQ IC. Both approaches are based upon our previous research in programmable floating-gate arrays and analog filters [10], [12]. Experimental data is presented from circuits fabricated on a $0.5\text{-}\mu\text{m}$ n-well CMOS process available through MOSIS.

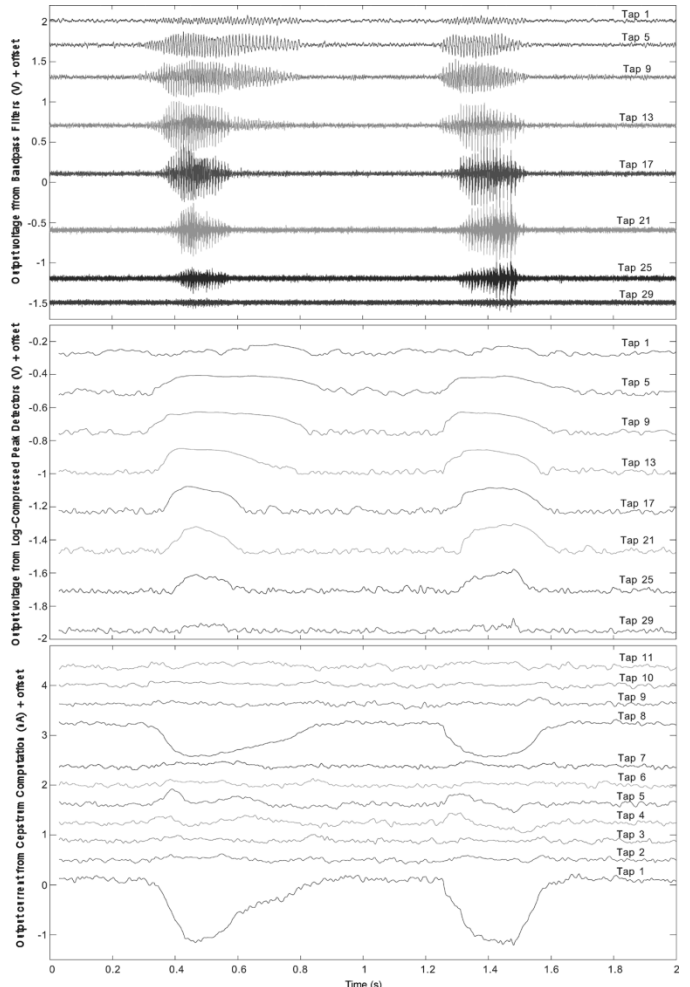


Fig. 6. Cepstrum system output. The system input is a sequence of speech using a standard speech database; each letter or phrase is separated by a short period of silence. There are 12 continuous cepstrum coefficients calculated for this section of speech and more coefficients is only a matter of chip area since the calculation is performed in parallel analog circuits. From the graph, one can see the two distinct periods of speech.

A. Continuous-Time Cepstrum

The mel-cepstrum [Fig. 5(b), as used in DSP] is often computed as the first stage of a speech recognition system [18]. Fig. 5(c) 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 discrete Fourier transform (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 [16]; the primary difference between the analog and digital computation approaches is in the frequency decomposition and amplitude detection method.

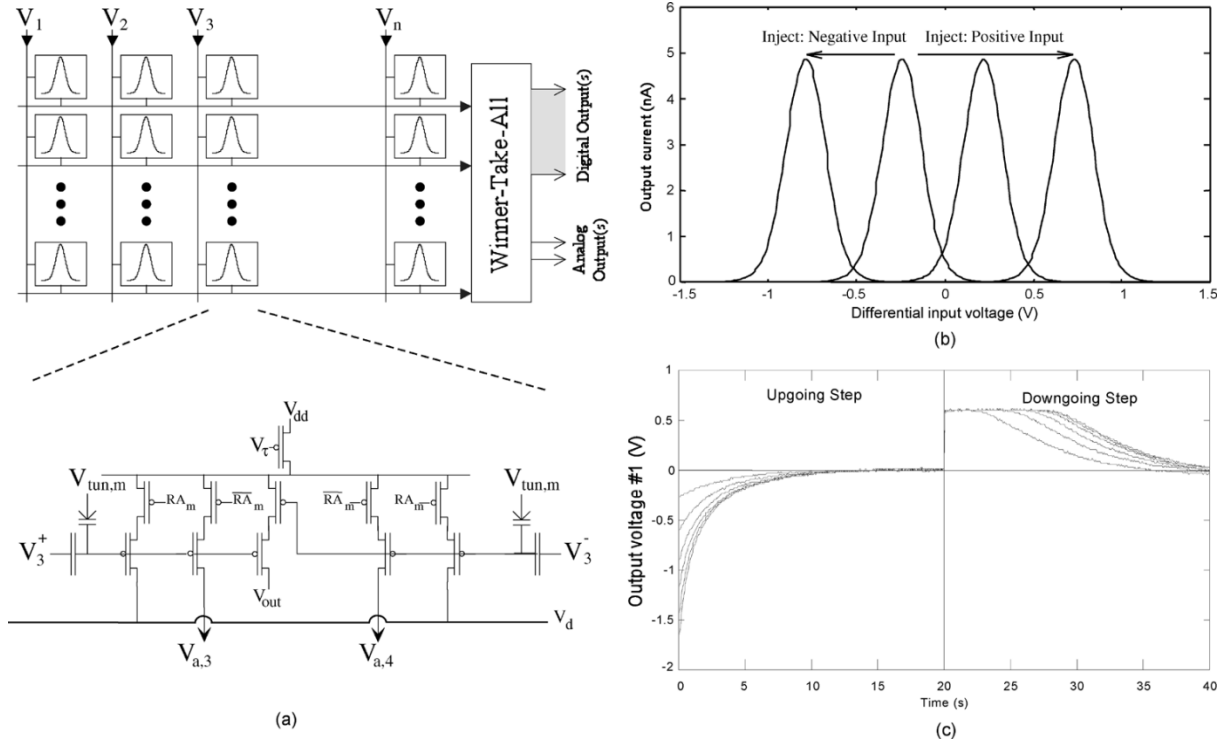


Fig. 7. Basic circuit, architecture, and measurements from the VQ circuit. (a) The core cell is built from a floating-gate bump circuit, which allows the target mean value to be stored and subtracted from the broadcasted input signal. (b) Results from programming the VQ circuit. We see that output current (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 [13]. 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 experimental measurements showing the convergence 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.

The basic building block of the continuous-time cepstrum implementation begins with a continuous spectrum decomposition and amplitude detection, similar to a DFT. The spectrum decomposition is done using differential C^4 second-order-section bandpass filters. The magnitude function (inside the log) is estimated using a peak detector rather than using the true magnitude of the complex spectrum. Finally, 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. Fig. 6 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 [19]. Recent experimental and computational studies have shown 98%–99% percent recognition on TI digit databases.

B. Continuous-Time VQ

In this section, we provide an overview of VQ, which is typically used in data compression and in classifying signals to sym-

bols [20]. 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. Fig. 7 shows the circuit and measured data from the VQ classifier array [17]. 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 Gaussian metric using $\text{sech}(\cdot)$ function, based upon a floating-gate circuit variation on the bump circuit [21], 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 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 previous IC implementations (nonfloating-gate) have used simple norm metrics for their difference functions [22], [23].

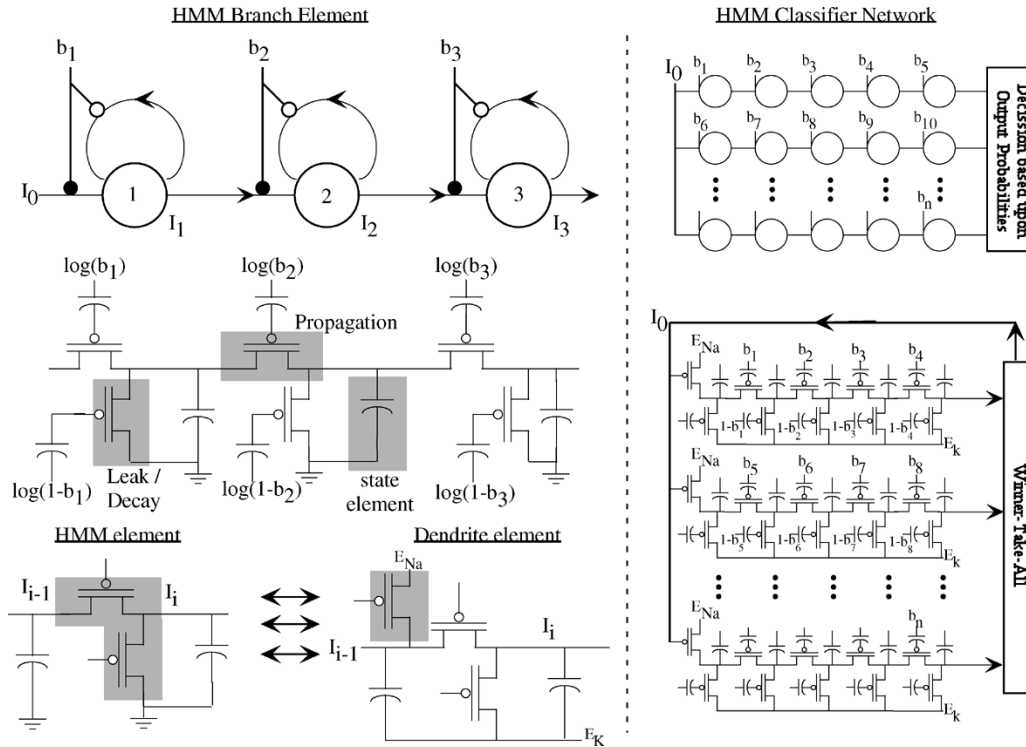


Fig. 8. Circuit design for our HMM branch element as well as the corresponding HMM classifier network. Our branch element design is based upon diffusor elements to perform the classical HMM calculation. In this framework, each branch element exhibits wave propagation. We can build these branch elements into an array for classification. In a practical implementation, we need to choose the largest *useful* result, which, in practice, is a WTA circuit, where only a subset of winning outputs are real outputs. These real outputs reset the HMM function. At the bottom of our branch element, we also show the relationship between a dendrite cell and an HMM cell. The difference between the two approaches is the implementation of the wave propagation mechanisms and to allow corresponding inputs to enhance wave propagation. The HMM cell explicitly combines two state elements and eliminates the leakage from the target cell. The dendrite first combines the probability to the resulting state element and then uses nonlinear gain to transmit the result to the next element. The shaded areas show the transistors that are signal dependent in each case.

We utilize floating-gate elements [7] 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 [17]. Fig. 7(b) shows that the means in a VQ array can be programmed to an arbitrary level. Using this approach, we have established approaches to program arrays of floating-gate elements. Fig. 7(c) shows the circuit and architecture for our adaptive VQ system [17], [24]; we adapt the floating-gate charge to the mean of the input signal. To get a stable adaptive behavior we incorporate our CMFB circuit on the bottom of the array, and connect to the selected element for adaptation; typically, we would only be adapting only one row at a given time. This approach requires some circuit reconfiguration at the core cell; if only adaptation or programming would be used, then the circuit remains simpler than shown in Fig. 7(a) [17]. The sum of these current outputs are sent through a winner-take-all circuit that outputs the N largest results [25].

C. Continuous-Time HMM

An 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, one of which is phonemes. The ongoing input train of symbols is used to map a path through a trellis of probabilities for these larger blocks of phonemes and words [26].

Fig. 8 shows our HMM branch implementation and HMM network implementation. HMMs may be looked at as probabilistic state machines or some sequential processing structure. For the stereotypical speech production HMM, the likelihood update equation is

$$\phi_i(n) = b_i(n) ((1 - a_i) \phi_i(n - 1) + a_{i-1} \phi_{i-1}(n - 1)). \quad (1)$$

where $\phi_i(n)$ represents the current state at time (n), $b_i(n)$ is the input to the current state and $a_i(n)$ are the transition probabilities between adjacent states. For our implementation, we look at HMMs as propagating waves and the probabilities relate to the velocity of propagation. 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 [27].

By rewriting (1) in the continuous case, Taylor series expansion in time, and retaining first-order terms, we get the following set of differential equations and its connection to the resulting transistors as:

$$\underbrace{\delta \frac{\partial \phi_i(t)}{\partial t}}_{\text{capacitor}(a)} = \underbrace{[2b_i(t) - 1]\phi_i(t)}_{\text{leaktransistor}(b)} + \underbrace{\phi_{i-1}(t) - \phi_i(t)}_{\text{diffusiontransistor}(c)} \quad (2)$$

where $\phi_i(n) = I_i/I_{SO}$ for I_i as the leakage current at that node for $b_i(t) = 0$, I_{SO} as a reference current for the array of elements. We have set our terms to $a_i = 1$ for clarity; we can program these effects using the floating-gate circuit structure. We implement the $b_k(t)$ and $1 - b_k(t)$ terms through log-compressed voltage signals modifying the k th horizontal and vertical conductance elements, respectively. Previous work in this area used analog circuitry to decode the HMM states [28]; this work very clearly explained the computational paradigm for HMM classification although the circuits were not elegant implementations. In a longer work, we show the connection between this HMM classifier circuits and circuits modeling dendritic and synaptic computation [29].

REFERENCES

- [1] R. Ellis, H. Yoo, D. Graham, P. Hasler, and D. Anderson, "A continuous-time speech enhancement front-end for microphone inputs," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. II, Phoenix, AZ, 2002, pp. 728–31.
- [2] R. Sarpeshkar, "Efficient precise computation with noisy components: Extrapolating from an electronic cochlea to the brain," Ph.D. dissertation, Dept. Comput. Neur. Syst., California Inst. Technol., Pasadena, CA, 1997.
- [3] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.
- [4] C. A. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.
- [5] P. Hasler and D. V. Anderson, "Cooperative analog-digital signal processing," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. IV, Orlando, FL, May 2002, pp. 3972–5.
- [6] P. D. Smith and P. Hasler, "Analog speech recognition project," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 4, Orlando, FL, 2002, pp. 3988–3991.
- [7] P. Hasler and T. S. Lande, "Overview of floating-gate devices, circuits, and systems," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 1–3, Jan. 2001.
- [8] P. Hasler, C. Diorio, B. A. Minch, and C. A. Mead, "Single transistor learning synapses," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: MIT Press, 1995, vol. 7, pp. 817–824.
- [9] P. Hasler, B. A. Minch, J. Dugger, and C. Diorio, "Adaptive circuits and synapses using pFET floating-gate devices," in *Learning in Silicon*. Norwell, MA: Kluwer, 1999, pp. 33–65.
- [10] M. Kucic, P. Hasler, J. Dugger, and D. V. Anderson, "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *Proc. Conf. Advanced Research in VLSI*, E. Brunvand and C. Myers, Eds., Mar. 2001, pp. 148–162.
- [11] T. S. Hall, P. Hasler, and D. V. Anderson, "Field-programmable analog arrays: A floating-gate approach," presented at the 12th Int. Conf. Field Programmable Logic and Applications, Montpellier, France, Sep. 2002.
- [12] M. Kucic, A. Low, P. Hasler, and J. Neff, "A programmable continuous-time floating-gate Fourier processor," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 90–99, Jan. 2001.
- [13] P. Smith, M. Kucic, and P. Hasler, "Accurate programming of analog floating-gate arrays," in *Proc. Int. Symp. Circuits and Systems*, vol. 5, Phoenix, AZ, May 2002, pp. 489–492.
- [14] D. Graham and P. Hasler, "Capacitively-coupled current conveyer second-order section for continuous-time bandpass filtering and cochlea modeling," presented at the Int. Symp. Circuits and Systems, 2002.
- [15] H. Yoo, D. V. Anderson, and P. Hasler, "Continuous-time audio noise suppression and real-time implementation," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. IV, Orlando, FL, May 2002, pp. 3980–3.
- [16] P. Smith, M. Kucic, R. Ellis, P. Hasler, and D. V. Anderson, "Cepstrum frequency encoding in analog floating-gate circuitry," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. IV, Phoenix, AZ, May 2002, pp. 671–674.
- [17] P. Hasler, P. Smith, C. Duffy, C. Gordon, J. Dugger, and D. Anderson, "A floating-gate vector-quantizer," presented at the IEEE Midwest Circuits and Systems, Tulsa, OK, Aug. 2002.
- [18] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals*. New York: Macmillan, 1993.
- [19] T. M. Massengill, D. M. Wilson, P. Hasler, and D. Graham, "Empirical comparison of analog and digital auditory preprocessing for automatic speech recognition," presented at the Int. Symp. Circuits and Systems, Phoenix, AZ, May 2002.
- [20] J. Schurmann, Ed., *Pattern Classification, A Unified View of Statistical and Neural Approaches*. New York: Wiley, 1996.
- [21] T. Delbruck, "Bump circuits for computing similarity and dissimilarity of analog voltages," in *Proc. Int. Joint Conf. Neural Networks*, vol. I, Seattle, WA, 1991, pp. 475–479.
- [22] G. T. Tuttle, S. Fallahi, and A. A. Abidi, "An 8 b CMOS vector A/D converter," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Monterey, CA, 1993, pp. 257–259.
- [23] G. Cauwenberghs and V. Pedroni, "A low-power CMOS analog vector quantizer," *IEEE J. Solid-State Circuits*, vol. 32, no. 8, pp. 1278–1283, Aug. 1997.
- [24] P. Hasler, "Continuous-time feedback in floating-gate mos circuits," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 56–64, Jan. 2001.
- [25] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of O(N) complexity," in *Advances in Neural Information Processing Systems*. San Mateo, CA: Morgan Kaufmann, 1988, vol. 1, pp. 703–711.
- [26] S. Ranals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Trans. Speech Audio Process.*, pt. 2, vol. 2, no. 1, pp. 161–174, Jan. 1994.
- [27] P. Smith and P. Hasler, "A programmable diffuser circuit based on floating-gate devices," presented at the Midwest Circuits and Systems, Tulsa, OK, 2002.
- [28] J. Lazzaro, J. Wawrzynek, and R. Lippmann, "A micropower analog VLSI HMM state decoder for wordspotting," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1996, vol. 9, pp. 727–733.
- [29] P. Hasler, P. D. Smith, E. Farquhar, and D. V. Anderson, "A neuro-morphic IC connection between cortical dendritic processing and HMM classification," presented at the IEEE DSP Workshop, Taos, NM, 2004.

Paul Hasler, photograph and biography not available at the time of publication.

Paul D. Smith, photograph and biography not available at the time of publication.

David Graham, photograph and biography not available at the time of publication.

Rich Ellis, photograph and biography not available at the time of publication.

David Anderson, photograph and biography not available at the time of publication.