50 years...
MILTON S. EISENHOWER RESEARCH AND TECHNOLOGY DEVELOPMENT CENTER

Bang, Click, Thud, or Whack?

Fernando J. Pineda, Gert Cauwenberghs, R. Timothy Edwards, Kenneth T. Ryals, and David G. Steigerwald

Acoustic transients—short, impulsive bursts of acoustic energy—are a rich source of information in the natural world. Biological systems process them quickly and economically. In this article, we describe a biologically inspired analog very-large-scale integration (VLSI) architecture for real-time classification of acoustic transients. Judicious normalization of time–frequency signals allows an elegant and robust implementation of a correlation algorithm. The algorithm replaces analog–analog multiplication with binary multiplexing of analog signals. This removes the need for analog storage and analog multiplication. Simulations show that the resulting algorithm has the same out-of-sample classification performance (about 93% correct) as a template-matching algorithm based on conventional analog correlation. This development paves the way for intelligent acoustic processing in low-power applications such as cellular telephones and debit cards.
(Keywords: Acoustic transients, Analog VLSI, Matched filters, Neural computation, Speech recognition.)

INTRODUCTION

Take a nature walk along a wooded path. A cricket chirps to your right. A woodpecker taps above you. A dry twig snaps underfoot. The sounds that tell you about the cricket, the woodpecker, and the twig are all acoustic transients: short, impulsive bursts of acoustic energy that last between 10 and 100 ms. Transients are a rich source of information in the natural world, and the ability to process them in real time provides a competitive advantage to species. As a result we, like other animals, have evolved the ability to quickly and economically process acoustic transients.

In the digital world of algorithms and computers, analogous evolutionary forces have caused engineers to develop powerful digital signal processing (DSP) algorithms for classification of acoustic signals on fast DSP engines. Using modern signal processing techniques to recognize acoustic transients in real time is straightforward on modern processors. The challenge of extracting information from signals has been met by powerful mathematical techniques such as wavelet analysis[1] and hidden Markov models.[2] The need for real-time performance has been met by fast and powerful central processing units (CPUs) and special-purpose DSP chips. The claim that the problem of acoustic pattern recognition is essentially solved is quite defensible.

Nevertheless, if we take a closer look at the DSP solutions, we find that the burden of real-time processing is borne by increasingly powerful digital processors. The price for success is measured in terms of power dissipation and complexity. Power dissipation scales linearly with the processor's clock rate. Thus, all else being equal, a 100-MHz processor dissipates 1000 times more power than a 100-kHz processor. Each bump up in clock rate requires increasingly heroic methods to control power dissipation. Complexity can be measured by the number of cycles required to perform a calculation and by the surface area of the chip. Increasingly complex algorithms create pressure to increase the complexity of the processor and thus the area of a chip. The problem of greater chip area can be overcome by scaling down the feature size of the fabrication process, but scaling also has physical limits. Moreover, as the feature size scales down, the fabrication process itself becomes more difficult and exacting.

All this contrasts sharply with nature's solution. The characteristics and advantages of nature's acoustic processing algorithms are well documented.[3] Natural systems process acoustic information in real time, with precision and reliability, while dissipating minuscule amounts of energy. Nature accomplishes this with slow and unreliable devices, i.e., neurons. Biological hardware has no clock, but typical time scales are measured in fractions of milliseconds. In effect, biological hardware runs at a 1- to 10-kHz clock rate.

If it were possible to engineer acoustic processors with biological levels of performance and power requirements, a large number of new applications would become feasible. Intelligence based on acoustic pattern recognition could be built into appliances, telephones, and credit cards. Cellular phones could take spoken commands. Smart credit cards could recognize not only passwords, but also the speaker. Digital watches and calculators that run for years on button cells could understand a small vocabulary of spoken words. Self-diagnosing machines could recognize acoustic transients caused by state changes and wear.

Motivated by the observation that biological systems perform very sophisticated tasks while making low demands on power consumption and component precision, we are engaged in developing artificial devices that perform as competently as biological systems while requiring minimal resources. Our long-term goal is to build pattern recognition engines whose performance characteristics rival those of biological systems. To be more specific, we seek to build acoustic processing engines with the following characteristics:

- Real-time operation, so that typical transients are recognized in about 100 ms or less.
- High correct classification rates (near 95%) on hundreds of transient classes while achieving low false alarm rates.
- Implementation of such engines with the highly mismatched metal-oxide-silicon (MOS) transistors that are typical in modern analog very-large-scale integration (VLSI) fabrication processes (feature size <1.2 μm).
- Power dissipation on the order of a milliwatt or less. This requires subthreshold current-mode circuits. Currents in such circuits are in the 0.1- to 10-nA range, while voltage swings are in the 100-mV range. Clock rates will be tens of kilohertz or less.

In this article we describe our recent work toward building processors with these characteristics. In particular, we focus on the impact of algorithm development on hardware requirements. We describe a practical architecture for performing real-time recognition of acoustic transients by means of a correlation-based algorithm. In other words, the algorithm performs pattern recognition by correlating an incoming signal with a stored template. Correlation-based algorithms are generally believed to be so computationally intensive that they cannot be used for real-time applications except in conjunction with fast DSP chips.

The algorithm and architecture that we describe are expected to perform a correlation calculation on a special-purpose parallel analog VLSI chip, using a slow clock (about 10 kHz) and (we estimate) with just a few milliwatts of power dissipation. Moreover, neither digital-to-analog conversion nor the storage of analog values is required. The algorithm leads to a correlator whose computing surface bears a strong resemblance to conventional dynamic random access memory (RAM).

To appreciate the advantages of our approach, it is useful to understand the constraints and advantages of correlation calculations in analog VLSI. Traditionally, correlation in analog VLSI poses two fundamental implementation challenges: first, the problem of template storage; second, the problem of accurate analog multiplication. Both problems can be solved by building sufficiently complex circuits. For example, analog values can be stored by sample-and-hold circuits or by storing digital values and converting them into analog values via digital-to-analog converters. These solutions are generally inferior to digital correlation algorithms because they lead to analog processors that are large compared with their digital counterparts. Another, more compact solution to the template storage problem is to employ the recently developed floating gate devices. Presently, such devices can store precise analog values for years without significant degradation. Moreover, this approach can result in very compact devices. Unfortunately, programming floating gate devices is not particularly easy. It is relatively slow and requires high voltage. Worse yet, the floating gate degrades each time it is reprogrammed. The fabrication of high-quality floating gates also requires advanced fabrication processes that may not be compatible with circuits for other kinds of on-chip processing. Finally, even if the analog storage problem could be solved effectively, the problem of building accurate analog–analog multipliers remains. High-quality analog multipliers are notoriously difficult to build. Effective solutions require considerable area on the chip.

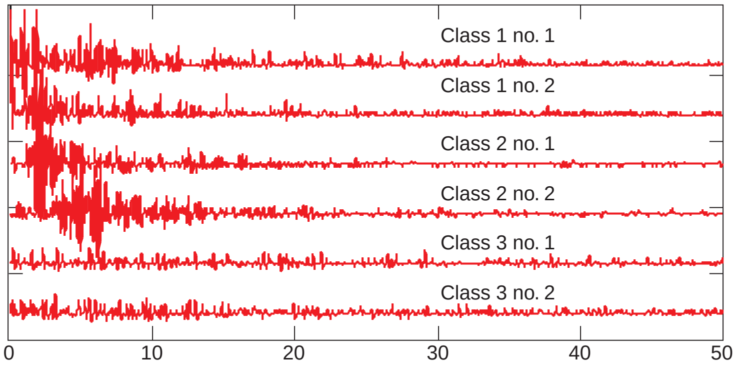


Figure 1. Six acoustic transients from three classes. Note the similarity between classes and the dissimilarity within classes.

Our "solution" to these problems is to sidestep them completely and to develop an algorithm and architecture that require neither analog storage nor analog multiplication. One instance of this approach is to binarize the input and then to correlate it with a binary template. Thus, the correlations can be performed by simple "AND" gates. This approach is compact and fast, but it generally tosses out too much information, so it is not compatible with high classification rates. In contrast, the algorithm we present in the following section is a hybrid approach that replaces analog–analog multiplication with analog–binary multiplication. In analog hardware this operation corresponds to simple binary multiplexing. We demonstrate that a high level of classification performance on real-world data can be achieved with no measurable loss of performance in comparison with a traditional, computationally intensive correlation algorithm. Moreover, the algorithm is not significantly harder to implement than binary–binary correlation. In the acoustic case, the approach requires neither digital-to-analog nor analog-to-digital converters.

THE BASELINE ALGORITHM

In this section we describe both the acoustic transient data and a conventional correlation algorithm used to classify the data. Two of us (K. R. and D. S.) collected the transients.[4] These transients consist of isolated bangs, claps, clicks, cracks, dinks, pings, pops, slaps, smacks, snaps, thuds, and whacks that were recorded on digital audio tape in an office environment. The ambient noise level was uncontrolled but was typical of a single-occupant office. Approximately 221 transients in 10 classes were collected. Figure 1 shows six exemplars from three classes. As can be seen, most of the energy in one of our typical transients was dissipated in the first 10 ms. The rest was dissipated over the course of about 100 ms. The transients had durations of about 20 to 100 ms. There was considerable in-class and between-class variability in duration. The duration of a transient was determined automatically by a segmentation algorithm, described later in this article. The segmentation algorithm was also used to align the templates in the correlation calculations.

Pineda et al.[4] described the baseline classification algorithm and its performance. Here we summarize only its salient features. As in many biologically motivated acoustic processing algorithms,[3] the preprocessing steps included time–frequency analysis, rectification, and smoothing and compression via a nonlinearity. Classification was performed by correlation against a template that represented a particular class followed by selection of the class with the greatest correlation. Creating the templates also required a "training" step. This training step is described under "Correlation" later in this article. We turn now to a more detailed description of each processing step.

Time–Frequency Analysis

Time–frequency analysis for the baseline algorithm and for the simulations performed in this work was performed by a low-power (5.5-mW) analog VLSI filter bank intended to mimic the processing performed by the mammalian cochlea.[5] This real-time device created a time–frequency representation that would ordinarily require hours of computation on a high-speed workstation. More complete descriptions of the hardware can be found in the references (Pineda et al.,[4] Goldstein et al.,[5] and references therein). The time–frequency representation produced by the filter bank was qualitatively similar to that produced by a wavelet transformation. The center frequencies and Q-factors of each channel were uniformly spaced in log space. The low-frequency channel was tuned to a center frequency of 100 Hz and a Q-factor of 1.0, while the high-frequency channel was tuned to a center frequency of 6000 Hz and a Q-factor of 3.5. There were 31 output channels. The 31-channel cochlear output was digitized and stored on disk at a raw rate of 256,000 samples per second. This raw rate was distributed over 32 channels, at rates appropriate for each channel (six rates were used, 1 kHz for the lowest-frequency channels up to 32 kHz for the highest-frequency channels and the unfiltered channel).

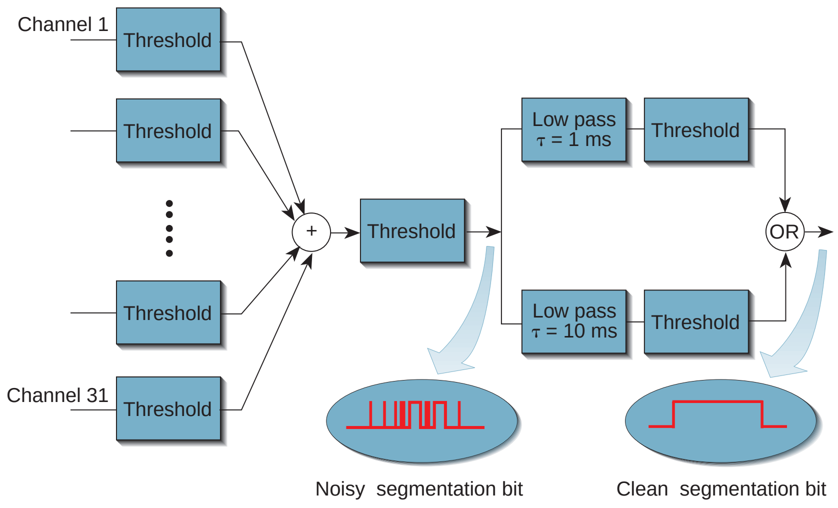


Figure 2. Schematic of the segmenter network (τ indicates the time scale of the low-pass filters).

Segmentation

Both the template calculation and the classification algorithm depend on having a reliable segmenter. In our experiments, the transients were isolated and the noise level was low; therefore, a simple segmenter was adequate. Figure 2 shows a block diagram of our segmenter.

The raw output of each channel was high-pass filtered to subtract the mean and then was rectified. The signal in each channel was then passed through a threshold function. In principle, each channel can have its own threshold, but in practice, the same threshold was used for all channels. The resulting bits were summed and again passed through a threshold function. The result is a noisy segmentation bit that was set to 1 if two or more channels exceeded their thresholds. A clean segmentation bit was generated from the noisy segmentation bit by passing the noisy segmentation bit through two parallel channels. Each channel first low-pass filtered the noisy segmentation bit and then passed it through a threshold function. The first channel used a 10-ms low-pass filter to fill in dropouts; the second channel used a faster (1-ms) low-pass filter to catch the onset of a transient. The outputs of the two channels were passed through an "OR" gate to produce a clean segmentation bit. Essentially, the segmenter was a three-layer neural network composed of linear threshold units. The network has four adjustable thresholds that were determined in an ad hoc manner so as to maximize the number of true transients that are properly segmented while minimizing the number of transients missed or cut in half. No effort was made to control the duration of the segments generated by the segmenter. A software simulation of the segmenter was used to segment the raw cochlear output files into events that were then written out as smaller disk files. Segmenting a 15-s stretch of raw data took about 1 h of computation on an RS/6000 workstation (rated at 10 MFLOPS). If this segmenter were realized as an analog circuit, it would operate in real time. The segmented files were used as the starting point for all the experiments described in the following paragraphs.

Smoothing and Normalization

The raw output of the filter bank was rectified and smoothed with a single pole filter and subsequently normalized. Smoothing was done with the same time scale (1 ms) in all frequency channels. The instantaneous output of the normalizer was

$$\hat{\mathbf{X}}(t) = \frac{\mathbf{X}(t)}{\theta + \left\|\mathbf{X}(t)\right\|}, \tag{1}$$

where $\mathbf{X}(t)$ was the instantaneous vector of rectified and smoothed channel data and θ was a small positive constant whose purpose was to prevent the normalization stage from amplifying noise in the absence of a transient signal. With this normalization we have

$$\left\|\hat{\mathbf{X}}(t)\right\|_1 \approx 0 \quad \text{if } \left\|\mathbf{X}(t)\right\|_1 << \theta , \tag{2}$$

and

$$\left\|\hat{\mathbf{X}}(t)\right\|_1 \approx 1 \quad \text{if } \left\|\mathbf{X}(t)\right\|_1 >> \theta . \tag{3}$$

Thus, θ effectively determined a soft input threshold that transients must have exceeded if they were to be normalized and passed on to higher-level processing.

A sequence of normalized vectors over a time window of length T was used as the feature vector for the correlation and classification stages of the algorithm.

Figure 3 shows the normalized feature vectors corresponding to the first four examples of a typical class. These have been concatenated into a single plot.

Correlation

The feature vectors were correlated in the time–frequency domain against a set of k time–frequency templates. The kth feature vector template was precalculated by averaging over an ensemble of normalized feature vectors from the kth class. Thus, if C_k represented the kth transient class and $\langle\ \rangle_k$ represented an average over the elements in a class, e.g.,

$$\left\langle \hat{\mathbf{X}}(t) \right\rangle_k = E\left\{ \hat{\mathbf{X}}(t) \middle| \hat{\mathbf{X}}(t) \in C_k \right\}, \tag{4}$$

then the template was of the form

$$b_k(t) = \left\langle \hat{\mathbf{X}}(t) \right\rangle_k . \tag{5}$$

The instantaneous output of the correlation stage is a k-dimensional vector $\mathbf{c}(t)$ whose kth component is

$$c_k(t) \equiv \sum_{\tau=1}^{T} \hat{\mathbf{X}}(t-\tau) \cdot \mathbf{b}_k(\tau) . \tag{6}$$

The time–frequency window over which the correlations were performed is of length T and is advanced by one time-step between correlation calculations.

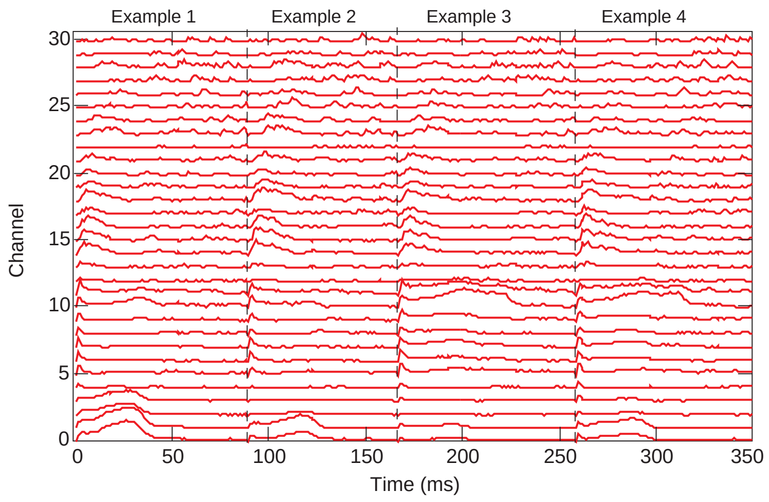


Figure 3. Normalized representation of the first four examples from a typical transient class.

Classification

The classification stage was a simple winner-take-all algorithm that assigns a class to the feature vector by picking the component of $c_k(t)$ that has the largest value at the appropriate time,

$$class = \arg_k \max\left\{ c_k(t_{\text{valid}}) \right\} . \tag{7}$$

The segmenter was used to determine the time t_{valid} when the output of the winner-take-all was to be used for classification. This corresponds to properly aligning the feature vector and the template.

Leave-one-out cross-validation was used to estimate the out-of-sample classification performance of all the algorithms described here. The rate of correct classification for the baseline algorithm was 92.8%. Of the 221 events detected and segmented, 16 were misclassified.

A CORRELATION ALGORITHM FOR ANALOG VLSI

We now take a closer look at the correlation step in the baseline algorithm. Can we perform classification without performing analog–analog multiplication and without having to store analog templates? In the following section we show that the answer to this question is yes. To provide a better understanding of our approach, we present it in two steps. In the first step, we construct a similarity measure that uses a binarized template and show that this template achieves a high level of classification performance. In the second step, we show how a slightly modified version of this similarity measure leads to a particularly elegant implementation in analog VLSI.

Examination of the normalized representation in Fig. 3 reveals that the features in the normalized representation vary slowly over time (compared with 1 ms). Moreover, adjacent frequency channels are very similar. Accordingly, the information content of any single time–frequency bin cannot be very high. This observation motivates a highly compressed representation for the stored template. To be useful, such a representation must not degrade the classification rate. Accordingly, we redefine the vector $\mathbf{c}(t)$ to be the following similarity measure:

$$c_k(t) \equiv \sum_{\tau=1}^{T} \hat{\hat{\mathbf{X}}}(t-\tau) \cdot \mathbf{b}_k(\tau) , \tag{8}$$

where the overdot represents differentiation with respect to t. In this expression, the time derivative of the normalized input vector is correlated against a binary valued [−1, +1] template vector $\mathbf{b}(\tau)$. This template vector is precomputed from examples by averaging over a set of typical transients and by setting each element of the template equal to one when the corresponding average is increasing in time and minus one if it is decreasing. In other words, the kth template is given by

$$b_k(\tau) = \mathbf{sgn}\left(\left\langle \dot{\hat{\mathbf{X}}}(\tau) \right\rangle_k\right), \tag{9}$$

where

$$\mathbf{sgn}(\mathbf{x}) = \mathbf{x}/|\mathbf{x}| \tag{10}$$

is the vector-valued function that takes the sign of each component of $\mathbf{x}$. Despite the fact that we have apparently removed a lot of information from the template, experiments with our office transients reveal that classification performance is not measurably degraded. In fact, in our experiments we found that exactly the same errors were made as in the baseline algorithm.

To gain insight into this unexpectedly high level of performance, we observe that differentiation throws out only an additive constant in each channel. This additive constant contains no information because the vectors are normalized. Next, we consider the effect of reducing the template vector to a single bit of information. This effect can be understood by first considering the dot product of two random normalized vectors, $\mathbf{x}$ and $\mathbf{y}$. If $\mathbf{x}$ and $\mathbf{y}$ are statistically independent, then the expected value of their dot product is zero, while the dot product of either vector with itself is just the Euclidean norm of the vector, e.g.,

$$\mathbf{x} \cdot \mathbf{x} = \|\mathbf{x}\|_2 . \tag{11}$$

Thus, if we normalize with respect to the Euclidean norm, identical vectors will have dot products equal to one, whereas vectors that are statistically independent will have dot products close to zero. Now, consider the dot product between a random vector $\mathbf{x}$ and a binary vector whose components are just the signs of a random vector $\mathbf{y}$. As before, if $\mathbf{x}$ and $\mathbf{y}$ are statistically independent, the dot product $\mathbf{x} \cdot \mathbf{sgn}(\mathbf{y})$ has an expected value near zero. Moreover, the dot product of a random vector $\mathbf{x}$ with $\mathbf{sgn}(\mathbf{x})$ will be equal to the 1-norm of $\mathbf{x}$, i.e.,

$$\mathbf{x} \cdot \mathbf{sgn}(\mathbf{x}) = \|\mathbf{x}\|_1 . \tag{12}$$

Thus, if we normalize with respect to the 1-norm, identical vectors will have dot products equal to one, whereas vectors that are statistically independent will have overlaps close to zero. This heuristic analysis leads to the insight that using binary template vectors amounts to performing classification based on a 1-norm rather than a more traditional 2-norm. One expects differences in classification rates, but these differences will depend on subtleties of the distribution of input vectors. Work to characterize these subtleties is under way. Empirically, it is clear that for office transients these effects are insignificant.

Next we turn to two changes to the algorithm that allow us to implement it in analog VLSI in a particularly elegant fashion. We observe that we have eliminated the need for 4-quadrant multiplication. Instead, we need only to multiply a positive or negative real-valued $\dot{\hat{\mathbf{X}}}$ with a plus or minus one. In other words, we have reduced the computational requirements from 4-quadrant multiplication of two real-valued quantities to 4-quadrant multiplication of one real quantity with one binary-valued quantity. In what follows we show that we can further reduce the computation to a 1-quadrant multiplication of a positive real-value with zero or one.

First we observe that differentiation and addition commute; thus, we can write Eq. 8 as

$$c_k(t) = \frac{d}{dt} \sum_{\tau=1}^{T} \hat{\mathbf{X}}(t-\tau) \cdot \mathbf{b}_k(\tau) . \tag{13}$$

By performing the differentiation after we perform the correlation, we only have to perform 2-quadrant multiplications of the positive components of $\hat{\mathbf{X}}$ with plus or minus one. The final simplification is achieved by observing that normalization implies that when one channel is increasing in amplitude, one or more other channels must be decreasing in amplitude so as to maintain the normalization. In effect, normalization introduces a new approximate symmetry that we can exploit to further simplify the computation. To see how this comes about, consider a positive vector $\mathbf{x}$ normalized with respect to the 1-norm, i.e.,

$$\sum_{\omega} x_{\omega}(t) = 1 . \tag{14}$$

Taking the time derivative of this expression yields

$$\sum_{\omega} \dot{x}_{\omega}(t) = 0 . \tag{15}$$

We can rewrite this as the sum of positive and negative contributions

$$\sum_{\dot{x}_\omega > 0} \dot{x}_\omega(t) + \sum_{\dot{x}_\omega < 0} \dot{x}_\omega(t) = 0\,, \tag{16}$$

which immediately implies

$$\sum_{\dot{x}_\omega > 0} \mathbf{sgn}(\dot{x}_\omega)\dot{x}_\omega - \sum_{\dot{x}_\omega < 0} \mathbf{sgn}(\dot{x}_\omega)\dot{x}_\omega = 0\,. \tag{17}$$

In other words, the negative terms in $\dot{\hat{\mathbf{x}}} \cdot \mathbf{sgn}(\dot{\hat{\mathbf{x}}})$ exactly equal the positive terms. This means we need only accumulate the positive contributions and multiply by a factor of 2 to recover the complete result. Empirically, we find that the difference between the [0, 1] representation and the [−1, +1] representation remains a factor of 2 (to several digits of precision), even when the input vectors and the template vectors do not correspond to the same class. Our classification experiments show that the out-of-sample performance of the [0, 1] representation is *identical* to that of the [−1, +1] representation. Changing to the [0, 1] representation has no impact on the storage requirements, since both representations require the storage of a single bit per time–frequency bin. The big payoff is that the multiplication hardware is now very simple: 1-quadrant multiplication of a positive number with [0, 1] scarcely deserves the name multiplication, because in current-mode analog VLSI it can be implemented by a simple transistor on–off switch.

To summarize, we have developed a correlation algorithm that empirically performs as well as a baseline correlation algorithm but that requires only binary multiplexing to perform the correlation. We find that even with only 16 frequency channels and 64 time bins (1024-bits/template), we are able to achieve exactly the same level of performance as the original analog–analog correlation algorithm (31 frequency channels and 128 time bins).

Figure 4 illustrates the key architectural features of the correlator/memory. The rectified and smoothed frequency-analyzed signals are input from the left as

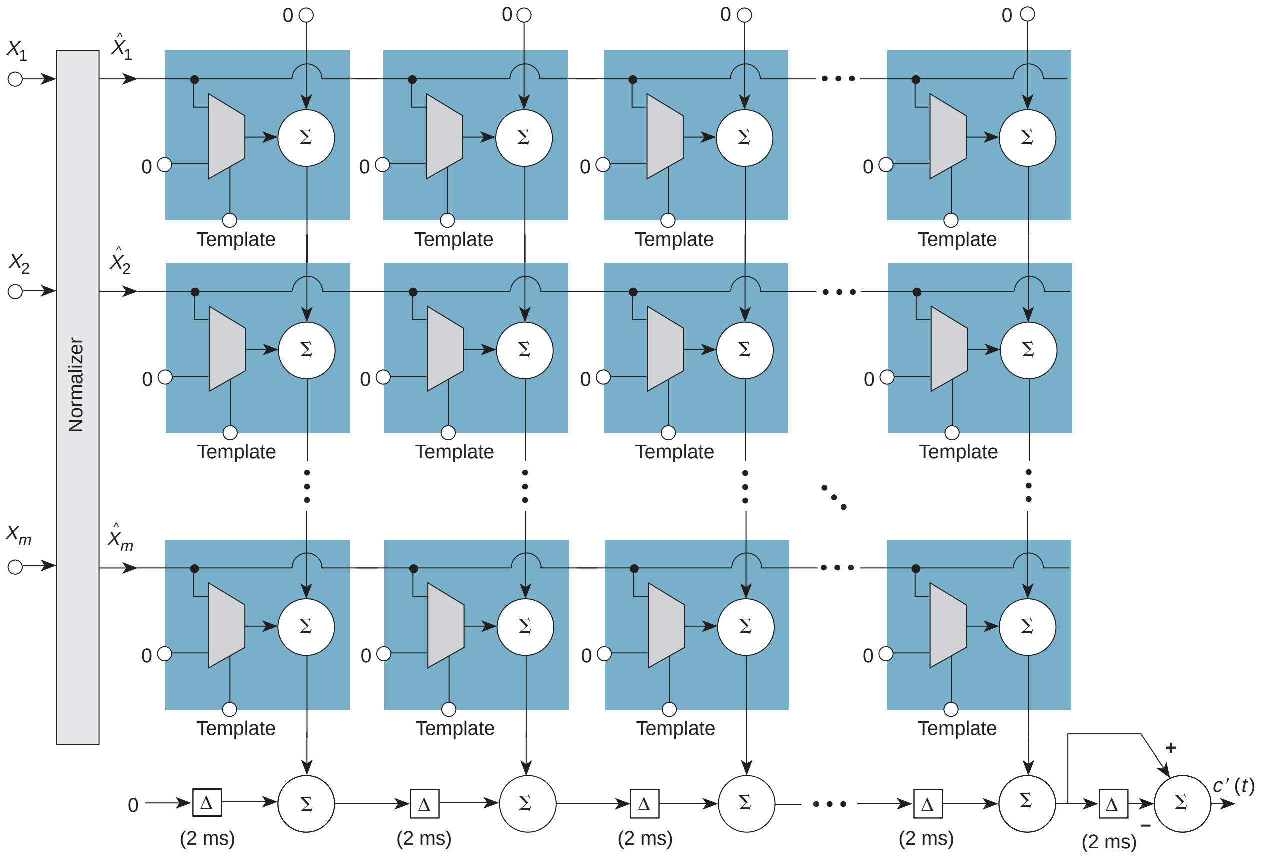


Figure 4. Schematic architecture of the *k*th correlator/memory. Δ and Σ are the usual symbols for delay elements and additive elements, respectively.

currents. The currents are normalized before being fed into the correlator. A binary time–frequency template is stored as a bit pattern in the correlator/memory. A single bit $b''_{\nu\tau}$ is stored at each time τ and frequency ν bin. If this bit is set, current is mirrored (switched) from the horizontal (frequency) lines onto vertical (aggregation) lines. Current from the aggregation lines is integrated and accumulated in a bucket brigade device.[6] The number of time bins and the clock speed of the bucket brigade determine the width of the correlation window in time. In our simulations, shifts occur every 2 ms. The last two stages of the bucket brigade are differenced to estimate a time derivative.

DISCUSSION, CONCLUSIONS, AND FUTURE DIRECTIONS

The proposed architecture uses an algorithm that correlates an analog value with a binary template. The incoming signal is not significantly compressed. Only the templates used for correlation are significantly compressed. Accordingly, the entire processing path from transduction until the accumulate-and-shift step can be performed in a fully analog, data-driven fashion. The only clock that appears in the system is used for the analog shift register. This clock is very slow (about 10 kHz) as compared with conventional microprocessor speeds. The correlator/memory array can be implemented as an array of cells bearing a strong resemblance to dynamic or static RAM cells. Thus, storing templates is as easy as loading conventional RAM, which is much easier than storing analog values in a floating gate array.

The correlation algorithm described in the previous section is related to the zero-crossing representation analyzed by Yang et al.[3] To see the relationship, suppose that the bit pattern stored in each channel of the correlator/memory were run-length encoded. The resulting bit flips would correspond exactly to the zero crossings of the expected time derivative of the normalized "energy envelope." Yang et al. [3] argue that a zero-crossing representation enhances robustness with respect to noise while maintaining the information content of the signal. An important difference between the present approach and the conventional zero-crossing approach, as exemplified by Yang et al., is that we do not encode the incoming acoustic signal with a zero-crossing representation. Instead, the acoustic signal is maintained as an analog signal throughout the processing path, and *the templates are encoded as a zero-crossing representation*. Interestingly enough, if both the analog signal and the template are reduced to a binary representation, then the classification performance drops dramatically. Clearly, maintaining some analog information in the processing path is important.

The frequency domain normalization used in the preceding approach is essentially instantaneous compared with the characteristic time scales of the signal. Absolute intensity information is mostly thrown away, but at each instant, the *relative* amplitude of the frequency channels is preserved. Because of the normalization, all information in the transient is equally important, regardless of its intensity. Thus, low-intensity resonances that might be excited by the initial injection

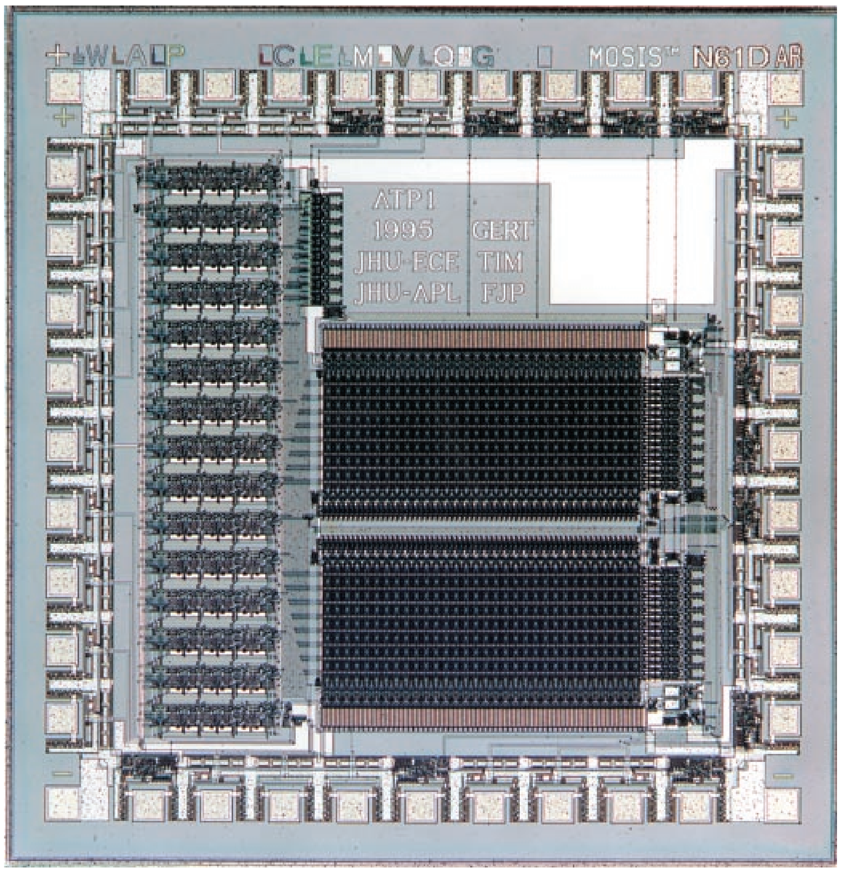


Figure 5. Prototype acoustic transient processing chip.

of energy are treated on the same footing as the loud onset of the transient. These resonances can contain significant information about the nature of the transient but would have less weight in an algorithm with a different normalization scheme. Another consequence of the normalization is that even a transient whose spectrum is highly concentrated in just a few frequency channels will spread its information over the entire spectrum through the normalization denominator. The use of a normalized representation thus distributes the correlation calculation over very many frequency channels and thereby mitigates the effect of device mismatch.

We consider the proposed architecture (without the trigger and without the winner-take-all classifier) as a potential component in more sophisticated acoustic processing systems. For example, the continuously generated output of the correlators $\mathbf{c}(t)$ is itself a feature vector that could be used in more sophisticated segmentation or classification algorithms, such as the time-delayed neural network approach of Unnikrishnan et al.[7] It is evident that the architecture could be useful in applications where the transients are not clearly separated. Perhaps the most significant of such applications is the recognition of continuous speech. Human speech is composed largely of transients, and speech recognizers based on transients can perform as well as recognizers based on phonemes.[8]

Three of us (T. E., G. C., and F. P.) have undertaken the design of a prototype acoustic transient processing chip. The time–frequency analysis and correlation parts of the algorithm all fit on a 2 × 2 mm chip in a 1.2-μm double-poly analog VLSI process. The chip is clocked at approximately 10 kHz and should dissipate power on the order of 1 mW. Figure 5 shows the layout of the test chip. A bank of bandpass filters with center frequencies ranging from 100 to 6000 Hz sits on the left-hand side of the chip. Two 16 × 64 pixel correlators can be seen on the right-hand side of the chip. We are currently testing the various subsystems in this chip and hope to report progress soon.

REFERENCES

[1] Kronland-Martinet, R., Morlet, J., and Grossman, A., "Analysis of Sound Patterns Through Wavelet Transforms," *Int. J. Pattern Recognit. Artif. Intell.*, Special Issue on Expert Systems and Pattern Analysis **1**, 97–126 (1989).

[2] Rabiner, L. R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, reprinted in *Readings in Speech Recognition*, A. Waibel and K.-F. Lee (eds.), Morgan Kaufmann, San Mateo, CA (1990).

[3] Yang, X., Wang, K., and Shamma, S. A., "Auditory Representations of Acoustic Signals," *IEEE Trans. Inf. Process.* **38**, 824–839 (1992).

[4] Pineda, F. J., Ryals, K., Steigerwald, D., and Furth, P., "Acoustic Transient Processing Using the Hopkins Electronic Ear," *World Conf. Neural Networks 1995*, Washington, DC, IEEE Press (1995).

[5] Goldstein, M. H., Liu, W., and Jenkins, R. E., "Speech Processing by Real and Silicon Ears," *Johns Hopkins APL Tech. Dig.* **12**(2), 115–128 (1991).

[6] Coggins, R., Jabri, M., Flower, B., and Picard, S., "A Hybrid Analog and Digital VLSI Neural Network for Intracardiac Morphology Classification," *IEEE J. Solid State Circuits* **30**, 542–550 (1995).

[7] Unnikrishnan, K. P., Hopfield, J. J., and Tank, D. W., "Connected-Digit Speaker-Dependent Speech Recognition Using a Neural Network with Time-Delayed Connections," *IEEE Trans. Signal Process.* **39**, 698–713 (1991).

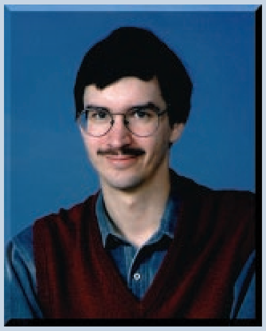
[8] Morgan, N., Bourlard, H., Greenberg, S., Hermansky, H., and Wu, S. L., "Stochastic Perceptual Models of Speech," *IEEE Proc. Int. Conf. Acoust., Speech Signal Process.*, Detroit, MI, pp. 397–400 (1996).

ACKNOWLEDGMENTS: The work reported here was supported by a Whiting School of Engineering/Applied Physics Laboratory Collaborative Grant. Preliminary work was supported by APL Independent Research and Development funds.

THE AUTHORS

FERNANDO J. PINEDA received his B.S. degree in physics from the Massachusetts Institute of Technology in 1976 and his M.S. and Ph.D. degrees in physics from the University of Maryland, College Park, in 1981 and 1986, respectively. He joined APL in 1986. Currently, he is a member of the Principal Professional Staff in the Milton S. Eisenhower Research and Technology Development Center. His research interests include neural computation, analog VLSI, and machine learning. He formerly served on the editorial board of the *Johns Hopkins APL Technical Digest* and is a member of the editorial boards of *Neural Computation* and *Applied Intelligence*. His e-mail address is Fernando.Pineda@jhuapl.edu.

GERT CAUWENBERGHS received the Engineer's degree in applied physics from the Vrije Universiteit, Brussel, Belgium, in 1988, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology in 1989 and 1994, respectively. In 1994, he joined The Johns Hopkins University as an assistant professor in electrical and computer engineering. His research covers VLSI circuits, systems and algorithms for parallel signal processing, and adaptive neural computation. His e-mail address is gert@jhunix.hcf.jhu.edu.



R. TIMOTHY EDWARDS received his B.S.E.E. degree from Duke University in 1990, where he also majored in physics, and he received his M.S.E.E. degree from Stanford University in 1992. Currently, he is a Ph.D. candidate at The Johns Hopkins University, where he is a student of Gert Cauwenberghs. His main area of research is analog VLSI circuit and systems design, particularly relating to speech and sound recognition. His e-mail address is tim@bach.ece.jhu.edu.

KENNETH T. RYALS received a B.S. degree in physics from Wake Forest University and an M.S. degree in mechanical engineering from North Carolina State University. Immediately thereafter, he joined APL's Strategic Systems Department and has been involved in the Sonar Evaluation Program ever since. During this time, he has also participated in the analysis of theater ballistic missile defense systems and mine warfare systems. He currently supervises the Performance Assessment Section of the Undersea Systems Evaluation Group. His e-mail address is Kenneth.Ryals@jhuapl.edu.

DAVID G. STEIGERWALD received his B.S. degree in physics from Georgia Tech in 1977 and his M.S. in technical management (systems engineering) from the G.W.C. Whiting School of Engineering in 1995. He has worked at APL since 1977 in various positions related to submarine sonar. Over the last few years, his work has covered a wider range, including several pen-based PC logging applications for submarines and for doctors. His e-mail address is David.Steigerwald@jhuapl.edu.