A Cochlear-Implant Processor for Encoding Music and Lowering Stimulation Power

This 75 dB, 357 \( \mu \)W analog cochlear-implant processor encodes fine-phase-timing spectral information in its asynchronous stimulation outputs to convey music to deaf patients.

Cochlear implants (CIs), or bionic ears, restore hearing in profoundly deaf (greater than –90 dB hearing loss) patients. They function by transforming frequency patterns in sound into corresponding spatial electrode-stimulation patterns for the auditory nerve. Over the past 20 years, improvements in sound-processing strategies, in the number of electrodes and channels, and in the rate of stimulation have yielded improved sentence and word recognition scores in patients.\(^1\) Next-generation implants will be fully implanted inside the patient’s body. Consequently, power consumption requirements for signal processing will be very stringent.

The processor we discuss in this article is intended for use in such next-generation implants. It can operate on a 100 mA-hr battery with a 1,000 charge-and-discharge cycle lifetime for 30 years, while allowing nearly 1 mW of electrode-stimulation power. It provides more than an order-of-magnitude power reduction over an A/D-then-DSP (analog/digital, then digital signal processor) solution, which often consumes 5 mW or more. Our processor’s digital outputs, its immunity to power-supply noise and temperature variations, and its high programmability level ensure ease of use with an implant system’s other parts, such as the wireless communication link and the programming interface.

CI users and music perception

A common speech-processing strategy, used in implants and speech-recognition systems, employs a mel-cepstrum filter bank with eight to 20 channels. The mel scale maps frequencies to a perceptually linear scale.\(^2\) Filter banks based on the mel scale use linearly spaced filter center frequencies of up to 1 KHz and logarithmically spaced center frequencies above 1 KHz. (The center frequency is the frequency of maximum response in a bandpass filter output.) Ubiquitous cepstral techniques use a logarithmic measure of the spectral energy in each filter bank channel for further processing. Implants also use eight to 20 functioning electrodes for stimulation; because of spatial interactions among the electrodes, having more electrodes is often not useful.

Compared to normal-hearing listeners, deaf patients who use a cochlear implant have only a very limited ability to perceive music.\(^3\) In an earlier work, we showed that a low-power algorithm providing asynchronous interleaved sampling (AIS) in cochlear implants is well-suited for encoding fine-phase-timing information.\(^4\) The ability to encode such information is
important for music perception by CI users, because even the best-performing CI users appear unable to use more than seven to 10 channels of spectral information. A recent study of two stimulation strategies that don’t include fine-phase-timing information—the Advanced Combination Encoder (ACE) and Spectral Peak (SPEAK) strategies—confirms that music appreciation is less than satisfactory even with the latest implants.

Consequently, researchers have recently proposed several strategies, in addition to AIS, for delivering fine-phase-timing information in CI stimulation. These include frequency-amplitude modulation encoding (FAME) and peak-derived timing (PDT). However, none of these have successfully presented fine-phase-timing information to CI users in a way that can improve music perception. Hence, further tests on CI users are necessary to investigate each technique’s efficacy.

Our 16-channel cochlear-implant processor implements our earlier AIS algorithm. This processor moves the AIS strategy a step closer to testing on CI users. It is suitable for encoding music, and it operates with very low power consumption because of its use of analog processing techniques. The AIS algorithm allows high-rate sampling of high-intensity channels while maintaining a low average rate of sampling for all channels, thus allowing lower stimulation power as well.

**Analog versus digital**

We estimate that an A/D-then-DSP implementation of traditional cochlear-implant processing would use about 0.25 mW to 0.5 mW for the microphone front end and A/D converter, and use 250 µW/MIP x 20 MIPS = 5 mW for the other processing, yielding a total power consumption of about 5.5 mW. These numbers are representative of state-of-the-art cochlear-implant processing, although many commercial processors’ power consumption is significantly worse because of various system inefficiencies. The power consumption for stimulation can range from 1 mW to 10 mW, depending on the patient and stimulation strategy. Our algorithm’s digital implementation, unlike that of a traditional processing algorithm, will likely be extremely power hungry, owing to its need for asynchronous processing and high-speed sampling of certain channels.

Our analog, 357 µW processor can improve performance, reduce processing power consumption by more than an order of magnitude, and significantly lower stimulation power consumption. This processor makes a fully implantable system feasible and practical. Thanks to the use of analog processing, its effective computational efficiency is between 1 µW/MIP and 5 µW/MIP in a 1.5 µm process. This is considerably better than even the most power-efficient DSPs, dominated by switching capacitance only in the DSP core, implementing their most favorable applications, and implemented in an advanced submicron process. The effective computational efficiency for such DSPs is between 50 µW/MIP and 100 µW/MIP.

Needless to say, the efficiency for DSPs will continually improve with Moore’s law, but such improvements are increasingly more modest. Even if we generously assume that the power consumption of the DSP is actually zero at the end of Moore’s law, the power consumption of a very low-power microphone front end, anti-alias filter, and A/D converter would still likely exceed 357 µW. Moreover, A/D scaling in speed, power, and precision is far slower than Moore’s law, and some circuits in our analog implementation could also benefit from these improvements. A custom digital solution would certainly narrow the gap between a DSP’s and our processor’s power consumption, but the high cost of the A/D converter, the microphone, and the asynchronous processing in the digital domain would still give our analog processor a significant advantage.

It’s useful to understand why our processor operates more efficiently than an A/D-then-DSP implementation. An A/D converter immediately creates a representation of the incoming information as a series of relatively high-precision and high-speed numbers (16 bits at 44 KHz is typical in such applications) that by themselves carry very little meaningful information. This digitization consumes considerable power because doing any task with high speed and high precision is expensive. (High precision is necessary if an operation requires a wide dynamic range and all computations, including gain control, are performed in the digital domain. High speed is necessary to avoid aliasing.) Then, a DSP takes all these numbers and crunches them with millions of multiply-accumulate operations per second, burning power in several switching transistors. It finally extracts more meaningful log spectral-energy information—but, because of speech data's
high variability, at a far slower rate of 100 Hz to 1 KHz in 16 parallel bands and at 6-bit-to-8-bit precision.

In contrast, analog preprocessing lets our processor efficiently compress the incoming data such that low-speed and low-precision A/D converters at a later stage of the computation quantize the meaningful information. Some of our prior work analyzes the optimal point for digitizing information in more general systems. Too much analog preprocessing before digitization is inefficient because the costs required to maintain precision begin to rise steeply. Too little analog preprocessing before digitization is inefficient because the digital system ignores analog degrees of freedom that can be exploited to improve computational efficiency.

Analog systems are more efficient than digital systems at low output precision, whereas digital systems are more efficient than analog systems at high output precision. In our processor, the output precision in each channel is 7 bits, and we intentionally limited the maximum output firing rate to 1 KHz, to lower stimulation power and to avoid a firing rate in the auditory nerve that is limited to the refractory period of recovery (when the nerve is overstimulated at a rate that is too high). Our processor’s internal dynamic range (IDR) is near 55 dB, with gain control allowing 75 dB of input dynamic range. An analog solution can therefore compete with a digital solution if the entire system maintains the necessary precision. If a task required 14 bits of output precision, 72 dB IDR and 100 KHz bandwidth at each channel, the A/D-then-DSP strategy would definitely be more efficient than our solution.

An analog solution must preserve its efficiency advantage by carefully monitoring robustness (that is, immunity or insensitivity to process variation, power supply noise, crosstalk between signals, and pickup of other interfering noise sources). Such robustness need not be present in every device and every signal, as in a digital solution, but only at important locations in the signal-flow chain, where it truly matters. Our processor is robust in the face of power-supply noise, thermal noise, temperature variations, and transistor mismatches, owing to its use of feedforward and feedback calibration circuits, robust biasing techniques, and careful analog design. Thus, an analog system addresses the robustness-efficiency trade-off very differently than a digital system does.

Programmability is certainly not as great in an analog system as in a digital one. However, as in our case, this is less of an issue when implementing an algorithm that’s known to work. Our processor’s programmability of 165 parameters with 546 bits allows sufficient but not excessive programmability. Our processor’s efficiency is high because it exploits the transistor’s analog degrees of freedom for computation without treating it as a mere switch.

The AIS algorithm

This algorithm uses half-wave-rectified (HWR) and phase-locked current outputs from spectral-analysis channels to charge an array of neuronal capacitors that compete with one another in a race-to-spike paradigm: The first neuron to reach a fixed voltage threshold wins the race and gets to fire a spike.
(pulse). Thus, the AIS algorithm prevents simultaneous channel stimulation, to avoid spectral smearing through electrode interactions.\textsuperscript{13} Once a spike fires, all the capacitors reset, and the race to spike begins again, except that the algorithm applies a negative current to the neuron that just spiked, to inhibit it from winning in subsequent races. This inhibition current remains active for the duration of a predetermined relaxation time constant. The algorithm thereby enforces a minimum interspike interval, which the relaxation time constant sets. This prevents the maximum stimulation rate from ever exceeding the refractory rate of neuronal recovery, which would otherwise cause unnatural distortions in the temporal discharge patterns of cochlear implants.

However, stimulation is not constrained to fire only at the maximum rate. The algorithm naturally adapts the stimulation rate (effectively, the rate at which the algorithm samples the input) in both time and spectral space to the signal’s information content, so that the processor doesn’t spend any power during quiet periods or on quiet channels. Therefore, high-intensity channels win the race to spike more frequently and are sampled at a high phase-encoded rate, whereas low-intensity channels win less frequently and are sampled at a lower phase-encoded rate. This adaptable stimulation rate lowers the average stimulation power and allows more natural, asynchronous stimulation of the auditory nerve.\textsuperscript{4} Figure 1 shows a MATLAB simulation of the AIS algorithm on a segment of speech.

**The AIS processor**

Figure 2 shows a block diagram of the AIS processor, which implements the AIS algorithm, building on our prior work.\textsuperscript{10,11} We modified the envelope detector in our prior work so that it could quickly output HWR currents to the AIS circuit. When a spike fires, the spike activates tristate buffers within the winning channel to report the log envelope amplitude as a 7-bit digital number onto a common output bus, thus providing both amplitude information and fine-phase-timing information in a single output event. The only constraint on the rate of spikes arriving from multiple channels onto the output bus is that they not overlap. Hence, our 16-channel analog AIS processor provides high temporal resolution, without the need for a high-rate sampling clock that constantly runs whether or not events occur.

Figure 3 shows a 16-channel, voltage-mode winner-take-all (WTA) circuit that forms the AIS circuit’s core by detecting the first channel whose neuronal state variable, \(V_{i,x}^{\text{ss}}\), crosses a fixed voltage threshold, \(V_{\text{thresh}}\). (Throughout our description, the letter \(x\) in a signal variable denotes the signal variable corresponding to a channel with a channel number \(x\). The value of \(x\) can range from 1 to 16.) Output voltage \(V_{s,x}^{\text{ss}}\) goes high only in the winning channel, thereby suppressing all other channel outputs from rising by pulling up strongly on the common source voltage \(V_s\) through positive feedback in the

---

**Figure 2. The AIS processor.** The FG3329 microphone picks up sound, which goes through a preamplifier and then to a broadband automatic gain control (AGC) circuit. The AGC compresses this sound and converts the input dynamic range of 75 dB to an internal dynamic range of 55 dB. A bank of bandpass filters then filter the AGC’s compressed output. Envelope detectors perform rectification and peak detection on the filter outputs to create inputs for the AIS circuit and log A/D (analog/digital) blocks, respectively. The AIS circuit then generates the asynchronous timing events, while the log A/D converter digitizes the envelope of each channel.
When a channel wins and $V_{ox}$ rises, signaling an asynchronous firing event, a method of setting a one-shot pulse width on $V_{ox}$ is required. It’s also necessary to immediately inhibit the winning channel from firing again until after an absolute refractory period. Figure 4 shows a single channel of the AIS circuit. The attack-and-release subcircuit (shown in bold in the figure), defines both $V_{ox}$ pulse width $T_a$, and the absolute refractory period of inhibition $T_r$ immediately after a pulse fires. This subcircuit works as follows: A two-transistor superbuffer of cascaded n and p source followers ($T_x07$ and $T_x08$ in figure 4) is biased with a large pull-up current $I_a$ and a small pull-down current $I_r$. By design, the superbuffer output voltage $V_{ax}$ initially sits below comparator input threshold voltage $V_{inhTH}$, so comparator output voltage $V_{hst}$ is low, and devices $T_{x10}$ and $T_{x11}$ are off. At the rising edge of $V_{ox}$, $V_{ax}$ initially undershoots because $T_{x07}$ shorts out the threshold drop on $T_{x08}$ such that $V_{ax}$ falls to approximately the value of the release voltage $V_{rx}$. Current $I_a$ then charges up the release capacitor $C_s$, and $V_{ax}$ ramps up from a minimum value until it crosses $V_{inhTH}$ and causes $V_{hst}$ to go high, turning on $T_{x10}$ and $T_{x11}$ and terminating the pulse by pulling $V_{ox}$ low again. Fixing $C_s$ and $V_{inhTH}$ lets us make pulse width $T_p$ programmable by varying $I_a$. The pulse width’s programmability is necessary to accommodate different time profiles for charge transfer in each stimulation event.

Turning on $T_{x10}$ and $T_{x11}$ pulls $V_{ox}$ low and inhibits input voltage $V_{ax}$ from rising. Upon the falling edge of $V_{ox}$, $T_{x07}$ shuts off, causing $V_{ox}$ to step back up to a threshold drop above $V_{ox}$, $V_{ax}$ then follows $V_{ax}$ which ramps down as $I_a$ discharges $C_s$. The inhibition from $V_{hst}$, therefore, remains high until $V_{ax}$ falls below $V_{inhTH}$ again, and the time

---

**Figure 4.** One channel of the AIS circuit, with the attack-and-release subcircuit shown in bold.
it takes to do so sets $T_r$, programmable by varying $I_r$. Programmability in $T_r$ is necessary to enforce a minimum inter-pulse interval, which prevents a channel that has just won the race from immediately winning again. This programmability is also necessary for setting a minimum refractory period, which allows the auditory nerves stimulated by the winning channel to recover.

To perform pre-emphasis or equalization across channels, which might be necessary due to patient variability or fabrication offsets, a translinear input stage programs the effective threshold in each channel by varying the current gain, $A$, applied to each input. Rather than fixing $A$ and varying $V_{\text{thresh}}$ across channels, we equivalently fix $V_{\text{thresh}}$ and vary $A$. Because PMOS (p-channel metal-oxide semiconductor) devices $T_{x21}$, $T_{x22}$, $T_{x23}$, and $T_{x24}$ are biased below threshold, the translinear loop yields output current $I_o$ as $I_{\text{in}}(I_2/I_3) = I_o(I_2/I_3)$; thus, $A = I_2/I_3$, and we make $A$ programmable by fixing $I_2$ and varying $I_3$.

The AIS algorithm requires resetting all input capacitors, $C_{\text{in}}$, to ground whenever a stimulation pulse is generated. A reset digital signal, which is a Schmitt-triggered, buffered version of analog signal $V_a$, accomplishes this resetting. Thus, $C_{\text{in}}$ discharges through $T_{x9}$ when $V_a$ goes high, for a stimulation pulse’s duration. We program $I_3$, $I_r$, and $I_s$ using 3-bit current D/A converters. Figure 5 shows the range of programmability in the AIS circuit from measured chip waveforms. The chip has a total of 546 programmable bits, allowing the adjustment of 165 spectral-analysis and AIS parameters through a three-wire serial peripheral interface. We employed robust biasing of D/A currents, immune to both power supply noise and temperature, as described in our earlier work.\textsuperscript{10,11}

**Performance comparisons**

We played various sound clips from the computer into the AIS processor, and we recorded the asynchronous stimulation pulses along with their 7-bit log envelope values. One of these clips, taken from Handel’s Messiah, was analyzed by a bandpass filter bank in MATLAB that was mathematically equivalent to the filters on the chip. Figure 6a shows this clip as a spectrogram. Figure 6b shows the asynchronous spike pulses scaled by the log envelope energy, which we reconstructed into a continuous-time signal (shown in figure 6c) using low-pass filtering.\textsuperscript{9} As the spectrograms show, the chip reconstruction matches the ideal MATLAB simulation very well.

Figure 7 compares the performance of the AIS processor (tested with various speech and music sound clips), an ideal MATLAB AIS simulation, and a traditional non-phase-based tone-vocoding simulation representing continuous-interleaved-sampling (CIS) synchronous stimulation. The tone-vocoding simulation represents only a traditional CIS strategy; to achieve higher performance, most modern cochlear implants implement more sophisticated strategies, which could be based on CIS\textsuperscript{7} or on some other sampling technique.\textsuperscript{9} Nevertheless, cochlear implant simulations are still helpful for normal-hearing listeners to gauge the best possible outcomes in CI users. The reason is that electrical stimulation creates many artificial problems, such as cross-fiber synchrony and perceptual dissonance,\textsuperscript{15} that don’t exist in natural acoustic stimulation.

In Figure 7a, we correlated the sound reconstruction (given by the summation of all channels) with the original sound signal, and the vertical bars represent the correlation coefficient, $r$. A high correlation coefficient between
Figure 6. Spectrograms comparing (a) an ideal MATLAB simulation of the bandpass filter outputs in each channel, where \( f \) indicates the center frequency of each filter; (b) asynchronous spike outputs from the AIS processor, where \( N \) is the number of spikes recorded in that channel; (c) spike-reconstructed filter outputs from the AIS processor, where \( r \) shows the correlation coefficient of each channel ranging from 0 to 1, or NaN (meaning “not a number,” resulting from \( N \) in that channel being 0). Note that fine phase-timing information is preserved. (Sound source: 1.46 s of the Hallelujah chorus from Handel’s Messiah.)

The reconstruction and original sound captures the fidelity of both envelope and fine-phase-timing preserved in the signal. A high correlation coefficient can also predict a normal-hearing listener’s increased ability to recognize words and music while listening to a reconstruction. The AIS processor’s correlation coefficients are comparable to those from the MATLAB AIS simulation. Thus, the AIS processor, unlike traditional CIS, encodes fine-phase-timing information, which is necessary for preserving music.

Figure 7b compares the average firing rate (AFR) of the AIS processor, the ideal AIS MATLAB simulation, and the CIS tone-vocoding simulation. The CIS stimulation rate is the fixed rate at which a conventional CIS processor samples the envelope of each analysis channel. In practice, clinicians typically set this fixed rate to between 800 Hz and 2.5 KHz and then adjust it to maximize performance. So, for comparison purposes, we chose a rate of 2 KHz in figure 7b. AIS achieves a lower AFR than conventional CIS, without compromising signal fidelity; in fact, AIS increases this fidelity. Hence, the AIS processor demonstrates that adapting the sampling rate to the signal’s needs can substantially save stimulation power. This benefit comes at the cost of increased signal-processing.
However, in our analog implementation with the AIS processor, the increase is a modest 106 μW (or 6.6 μW per channel) over our analog CIS processor, which consumes 251 μW.

Figure 8 shows a die photo of the 9.23 mm × 9.58 mm AIS processor, with labels describing the various blocks. The entire processor, built in a 1.5 μm process, consumes 357 μW. This is very efficient compared to typical A/D-then-DSP cochlear-implant processors, which often consume 5 mW or more.

The AIS processor demonstrates an example of how simple analog circuit-building blocks can help implement a complex signal-processing
algorithm with minimal resources of power and silicon area. In this example, the implemented algorithm is one that encodes music and lowers stimulation power, making fully implanted cochlear implants with good performance possible. Future work needs to combine work such as ours with other improvements to allow fully implanted systems to enter clinical practice. These improvements include lowering electrode impedances to further reduce current-spreading interactions among electrode channels.

REFERENCES


