Silicon Auditory Processors as Computer Peripherals

John Lazzaro[1], John Wawrzynek[1], M. Mahowald[2], Massimo Sivilotti[3], Dave Gillespie[4].

Abstract

Several research groups are implementing analog integrated circuit models of biological auditory processing. The outputs of these circuit models have taken several forms, including video format for monitor display [1,2], simple scanned output for oscilloscope display [3], and parallel analog outputs suitable for data-acquisition systems [4]. In this paper, we describe an alternative output method for silicon auditory models, suitable for direct interface to digital computers. As a prototype of this method, we describe an integrated circuit model of temporal adaptation in the auditory nerve, that functions as a peripheral to a workstation running the Unix operating system. We show data from a working hybrid system that includes the auditory model, a digital interface, and asynchronous software; this system produces a real-time X-Windows display of the response of the auditory nerve model.

1. Introduction

Several researchers have implemented computational models of biological auditory processing, with the goal of incorporating these models into a speech recognition system [5,6,7,8,9]. These projects have shown the promise of the biological approach, sometimes showing clear performance advantages over traditional methods.

The application of these computational models is limited by their large computation and communication requirements. Analog VLSI implementations of these neural models may relieve this computational burden; several VLSI research groups have efforts in this area, and working integrated circuit models of many popular representations presently exist [1,2,3,4,10,11,12,13,14,15].

Some neural models map sound into large two-dimensional representations, while other one-dimensional models have important time structure at a time scale of 100 microseconds. In an architecture where special purpose chips compute auditory representations and a general-purpose host computer uses them, an interactive research approach is limited by the ability of the host computer to receive and process the data in a timely fashion.

This paper presents an interface method [16,17] that explicitly addresses the communications issue between analog VLSI auditory implementations and digital processors. To demonstrate this method, the paper describes a silicon auditory nerve model that uses this

[1] Computer Science Division, EECS, University of California at Berkeley.

[2] Computation and Neural Sciences, California Institute of Technology, currently at MRC Anatomical Neurophamacology Unit, Oxford, England.

[3] Computer Science, California Institute of Technology, currently at Tanner Research, Inc.

[4] Computer Science, California Institute of Technology, currently at Synaptics, Inc.

interface method. This chip contains both the analog processing and the digital interface circuits; analog signals are not sent off chip.

2. Communication in Neural Systems

Biological neurons communicate long distances using a pulse representation. Communications engineers have developed several schemes for communicating on a wire using pulses as atomic units. In these schemes, maximally using the communications bandwidth of a wire implies the mean rate of pulses on the wire is a significant fraction of the maximum pulse rate allowed on the wire.

Using this criteria, neural systems use wires very inefficiently. In most parts of the brain, most of the wires are essentially inactive most of the time. If neural systems are not organized to fully utilize the available bandwidth of each wire, what does neural communication optimize? Evidence suggests that energy conservation is an important issue for neural systems. A simple strategy for energy conservation is the reduction of the total number of pulses in the representation. Many possible coding strategies satisfy this energy requirement.

The strategies observed in neural systems share another common property. Neural systems often implement a class of computations in a manner that produces an energy-efficient output encoding as an additional byproduct. The energy-efficient coding is not performed simply for communication and immediately reversed upon receipt, but is an integral part of the new representation. In this way, energy-efficient neural coding is intrinsically different from engineering data compression techniques. Temporal adaptation, lateral inhibition, and spike correlations are examples of neural processing methods that perform interesting computation while producing an energy-efficient output code.

These representational principles are the foundation of the neural computation and communication method we advocate in this paper. In this method, the output units of a chip are spiking neuron circuits that use energy-efficient coding methods. To communicate this code off a chip, we use a distinctly non-biological approach.

3. The Event-Address Protocol

The unique characteristics of energy-efficient codes define the remaining off-chip communications problem. In the spiking neuron protocol, the height and width of the spike carries no information; the neuron imparts new information only at the moment a spike begins. This moment occurs asynchronously; there is no global clock synchronizing the output units. One way of completely specifying the information in the output units is an event list, a tabulation of the precise time each output unit begins a new spike. We can use this specification as a basis for an off-chip communications system, that sends event-list messages to represent output unit activity.

An evaluation of the event-list representation as a communications protocol begins with the specification of the required temporal accuracy of time markers. The time scale of a neural computation determines the required accuracy of time markers. The accuracy requirements and the number of output units of a chip dictate the performance requirements of an implementation of the event-list protocol.

In comparison to other areas of the brain, biological auditory representations require extraordinary timing accuracy of individual pulses. One secondary neural representation uses the auditory nerve representation in computations that require 70 μs timing accuracy [18]. Using this accuracy requirement, we can calculate the worst-case performance requirements of an event-list communications system.

Using standard signalling conventions in a 2μ CMOS technology, a parallel bus can communicate the binary encoding of a single spiking event in 50ns. With this bus, 1,400 output units of a silicon auditory nerve can spike simultaneously, and all events will be correctly represented within the required $70\mu s$ timing accuracy. This analysis assumes that an output unit only produces one event in $70\mu s$; in biological systems, the absolute refractory period between two neural spikes is much greater than $70\mu s$.

Biological auditory nerves contain about 50,000 fibers; the largest auditory nerve computer models currently used in research have an equivalent of 1,280 fibers. Most biological neural representations require temporal accuracy at least one order of magnitude slower than the auditory nerve. With this relaxed accuracy requirement, at least 14,000 output units can communicate on a single bus.

This analysis addresses the simultaneous firing of many output units. This behavior is not an unlikely occurrence in neural representations; the temporal correlation of the activity in many neurons is a primary mechanism for information encoding. However, the energy-efficient coding ensures the sustained communications rate is much smaller than the maximum bandwidth of the 50ns bus.

Note that an explicit timestamp for each entry in the event list is not necessary, if communication latency between the sending chip and the receiver is a constant. In this case, the sender simply communicates, upon onset of a spike from an output, the identity of the output unit; the receiver can append a locally generated timestamp to complete the event. If simplified in this manner, we refer to the event-list protocol as the event-address protocol.

The event-list representation is one way to specify the information in the output units. Another way of specifying this information is to synchronously sample the state of the entire output unit array, at a constant rate determined by the required temporal accuracy of the representation. The sampling method uses a constant number of bits per unit time to represent information, independent of the amount of activity in the output units. This property reflects the explicit representation of both active and inactive states of each output neuron in the array.

In contrast, the number of bits in the event-list representation per unit time is a linear function of the amount of activity in the output unit array; if no output units are active, the number of bits in the event-list representation is zero. Therefore, if activity in the output unit array is sufficiently sparse, the event-list representation is a more efficient communication method than the sampled representation.

4. System Implementation of the Event-Address Protocol

We have designed a working system that computes a model of auditory nerve response, in real time, using analog VLSI processing. This system takes as input an analog sound source, and uses the event-list representation to communicate the model output to the

host computer.

Figure 1 is a block diagram of this system. A single VLSI chip computes the auditory model response; an array of spiking neuron circuits is the final representation of the model. This chip also implements the event-address protocol, using asynchronous arbitration circuits. The chip produces a parallel binary encoding of the model output, as an asynchronous stream of event addresses. These on-chip operations are shown inside the dashed rectangle in Figure 1, labelled **Chip Architecture**.

Additional digital processing completes the custom hardware in the system. This hardware transforms the event-address protocol into an event-list protocol, by adding a time marker for each event (16 bit time markers with $20\mu s$ resolution). In addition, the hardware implements the bus interface to the host computer, in conjunction with a commercial interface board. These operations are shown in Figure 1 as the box labelled **Board Architecture**, and are implemented with standard logic components.

The commercial interface board supports 10 MBytes/second asynchronous data transfers between our custom hardware and the host computer, and includes 8 KBytes of data buffers. Our display software produces a real-time graphical display of the auditory model response, using the X window system.

Figure 2 shows the screen image of the graphics display, showing the response of the chip to an equal mixture of two sinusoids (200 Hz and 1000Hz). In this figure, the x axis represents time and the y axis represents neural activity along the cochlea. The cochlear axis represents sound frequency using a logarithmic scale, with the lowest frequency at the bottom of the figure. A black dot represents the onset of a spike in an output unit. This figure shows that our implementation of the event-address protocol preserves the fine temporal structure of the cochlear representation; the short black curves in Figure 2 are spaced at 1 millisecond intervals.

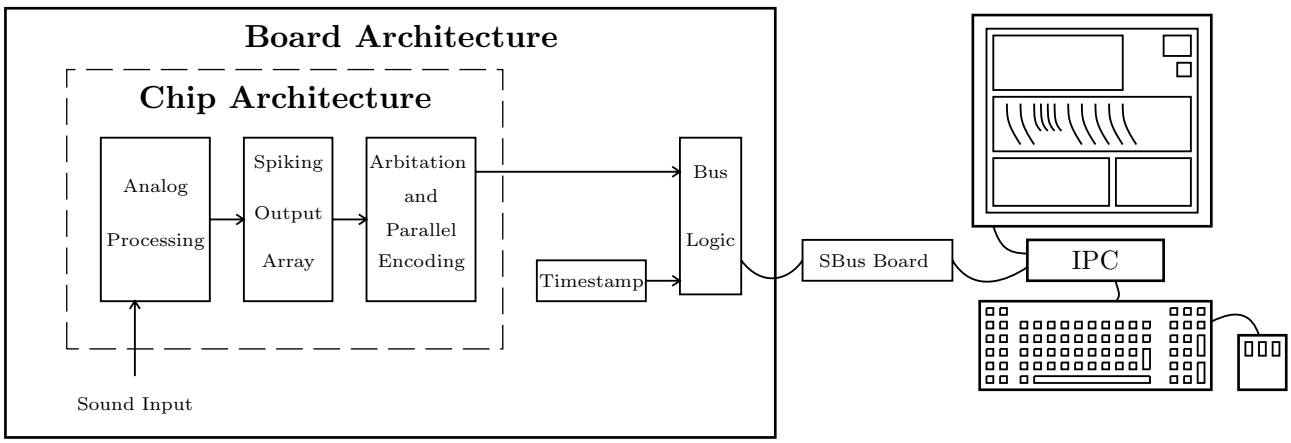


Figure 1. System block diagram, showing chip architecture, board architecture, and the host computer (Sun IPC).

5. VLSI Implementation of the Event-Address Protocol

This section describes the architecture and circuit design of the analog VLSI chip in the system. This description explains our asynchronous implementation of the event-address protocol in detail, to enable other researchers to use the protocol in their designs. Readers not interested in VLSI implementation details may wish to skip to the next section.

We fabricated the auditory nerve chip design through MOSIS, using the Orbit 2μ double-poly low-noise analog process; the chip dimensions are 2220μm by 2250μm. Figure 3 shows a block diagram of the chip, that models the auditory nerve representation. The analog input signal connects to a silicon cochlea [10], shown on the far left of the diagram, that has 30 output taps. Each output tap includes circuits that model inner-hair-cell transduction [12]. Each tap connects to five spiking neuron circuits, shown as a row of boxes, that form the silicon auditory nerve representation. The spiking neuron circuits model the temporal adaptation of the auditory nerve [14].

These 150 spiking neurons, arranged in a 30 by 5 array, are the output units of the chip; the event-address protocol communicates the activity of these units off chip. The temporal adaptation of these units acts as energy-efficient coding, and reduces the mean spike rate of the output array. At the onset of a spike from an output unit, the array position of the spiking unit, encoded as a binary number, appears on the output bus. The asynchronous output bus is shown in Figure 3 as the data signals marked **Encoded X Output** (column position) and **Encoded Y Output** (row position), and the acknowledge and request control signals A_c and R_c.

We implemented the event-address protocol as an asynchronous arbitration protocol in two dimensions. In this scheme, an output unit can access two request lines, one associated with its row and one associated with its column. Using a wire-OR signalling protocol, any output unit on a particular row or column may assert the request line. Each request line is paired with an acknowledge line, driven by the arbitration circuitry outside the array. Row and column wires for acknowledge and request are explicitly shown in Figure 3, as the lines that form a grid inside the output unit array.

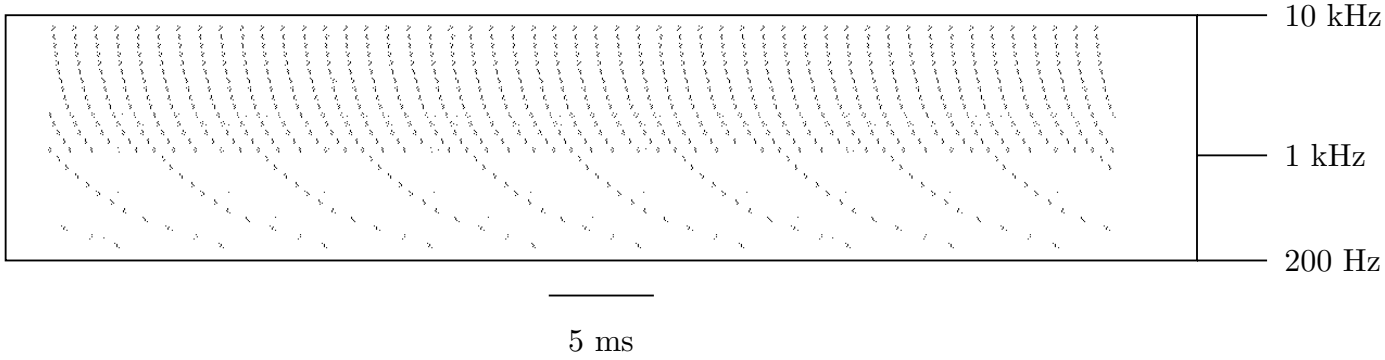


Figure 2. Screen image of the display program, showing auditory nerve model response to an equal combination of 1000 Hz and 200 Hz sinusoids. The spiking rates of the output units are set higher than in normal operation, to produce a high-contrast figure suitable for reproduction.

At the onset of a spike, an output unit asserts its row request line, and waits for a reply on its row acknowledge line. An asynchronous arbitration system, shown in Figure 3 as a triangle marked **Y Arbitration Tree**, assures only one output row is acknowledged. After row acknowledgement, the output unit asserts its column request line, and waits for a reply on its column acknowledge line. The column arbitration system is shown in detail in Figure 3; four two-input arbiter circuits, shown as rectangles marked with the letter A, are connected as a binary tree to arbitrate among the 5 column inputs. Upon the arrival of both row and column acknowledgements, the output unit releases both row and column request lines. Static latches, shown in Figure 3 as the rectangles marked **Control Logic,** retain the state of the row and column request lines.

Binary encoders transform the row and column acknowledge lines into the output data bus. Another column encoder senses the acknowledgement of any column, and asserts the bus control output R_c. When the external device has secured the data, it responds by asserting the A_c signal. The A_c signal clears the static latches in the **Control Logic** blocks and resets R_c. When A_c is reset, the data transfer is complete, and the chip is ready for the next communication event.

Figure 4 shows the details of the communications circuits of Figure 3. Figure 4(a) shows the two-input arbiter circuit used to create the binary arbitration trees in Figure 3. This digital circuit takes as input two request signals, R_1 and R_2, and produces the associated acknowledge signals A_1 and A_2. The acknowledgement of a request precludes the acknowledgement of a second request. The circuit asserts an acknowledge signal until its associated request is released.

R_o is an auxiliary output signal indicating either R_1 or R_2 has been asserted; A_o is an auxiliary input signal that enables the A_1 and A_2 outputs. The auxiliary signals allow the two-input arbiter to function as an element in arbitration trees, as shown in Figure 3; the R_o and A_o signals of one level of arbitration connect to the R_k and A_k signals at the next level of arbitration. In two-input operation, the R_o and A_o signals are connected together, as shown in the root arbiter in Figure 3.

In the two-input arbiter circuit, the primary inputs R_1 and R_2 are connected to cross-coupled static NAND gates; this configuration ensures the internal signals s_1 and s_2 are never asserted simultaneously. The auxiliary request output R_o is derived from the internal signals of the NAND gates. A static logic gate computes the primary outputs A_1 and A_2 from the s_1, s_2, and A_o signals, implementing the logic equations $A_1 = s_1 \cdot A_o$ and $A_2 = s_2 \cdot A_o$.

Figure 4(b) shows the circuit implementation of the **Control Logic** blocks in Figure 3; this circuit is repeated for each row and column connection. This circuit interfaces the output bus control input A_c with the arbitration circuitry. If output communication is not in progress, A_c is at ground, and $\overline{A}_c$ is at V_{dd}.

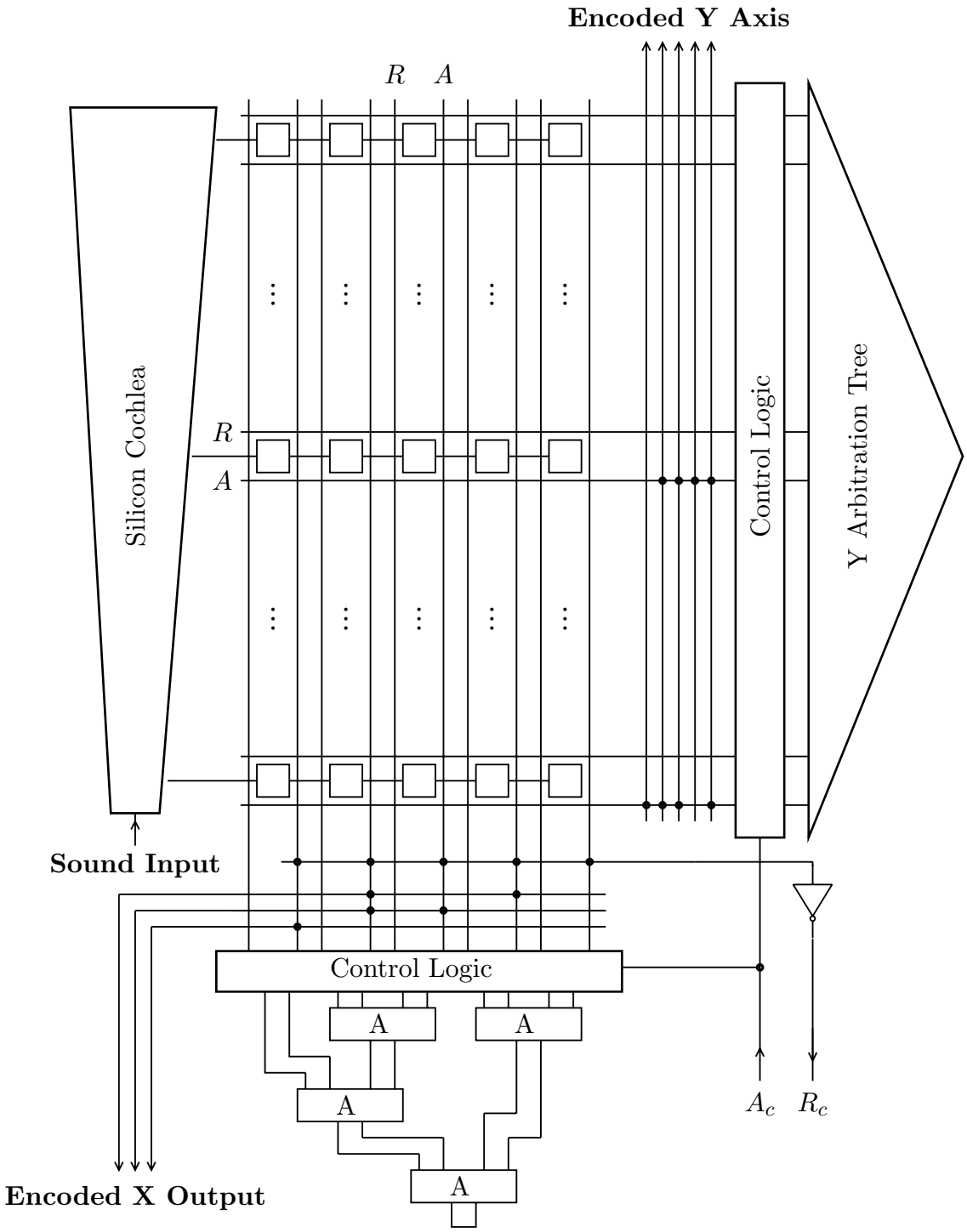
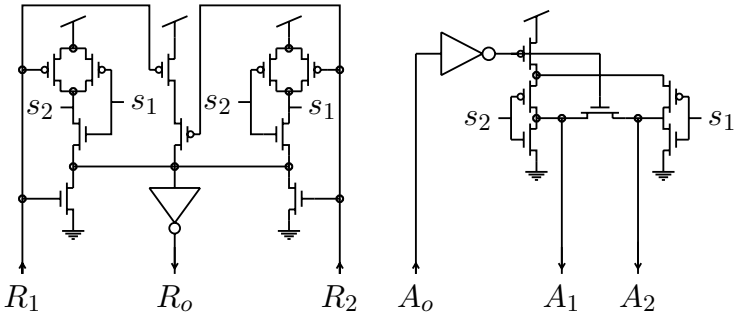
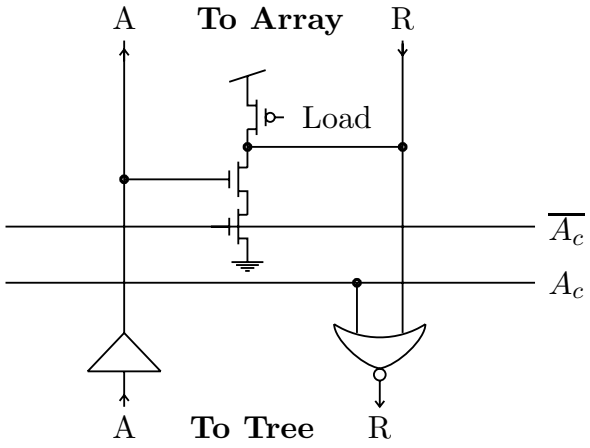


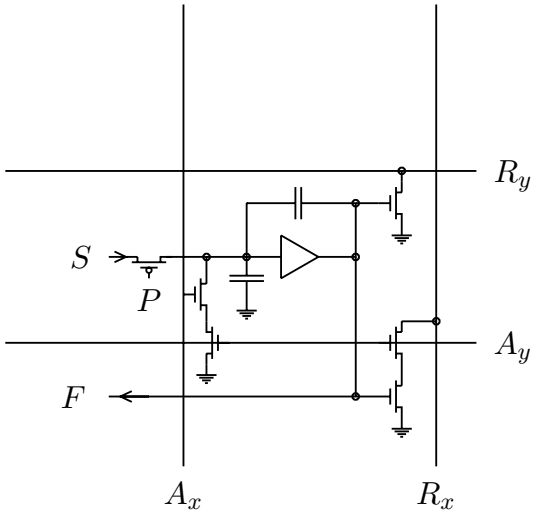
Figure 3. Block diagram of the chip. See text for details.



(a)



(b)



(c)

Figure 4. Diagrams of communication circuits in the chip. **(a)** Two-input arbiter circuit. **(b)** Control logic to interface arbitration logic and output unit array. **(c)** Output unit circuit.

The PFET transistor marked as **Load** acts as a static pullup to the array request line (R); output units pull this line low to assert a request. The NOR gate inverts the array request line, and routes it to the arbitration tree. When a pending request is acknowledged by the tree acknowledge line, the two NFET transistors act to latch the array request line. The assertion of A_c releases the array request line and disables the arbitration tree request input; these actions reset all state in the communications system. When A_c is released, the system is ready to communicate a new event.

Figure 4(c) shows the circuit implementation of a unit in the output array. In this implementation, each output unit is a two-stage low-power axon circuit [19]. The first axonal stage receives the cochlear input, and models the short-term adaptation of the auditory nerve [14]; this axon stage is not shown in Figure 4(c). The first stage couples into the second stage, shown in Figure 4(c), via the S and F wires.

To understand the operation of this circuit, we consider the transmission of a single spike. Initially, we assume the request lines R_x and R_y are held high by the static pullup PFET transistors shown in Figure 4(b); in addition, we assume the acknowledge lines A_x and A_y are at ground, and the noninverting buffer input voltage is at ground.

When the first axonal stage fires, the S signal changes from ground potential to V_{dd}. At this point the buffer input voltage begins to increase, at a rate determined by the analog control voltage P. When the switching threshold of the buffer is reached, the buffer output voltage F swings to V_{dd}; capacitive feedback ensures a reliable switching transition. At this point, the output unit pulls the request line R_y low, and the communications sequence begins.

The Y arbitration logic replies to the R_y request by asserting the A_y line. When both F and A_y are asserted, the output unit pulls the request line R_x low. The X arbitration logic replies to the R_x request by asserting the A_x line. The assertion of both A_x and A_y resets the buffer input voltage to ground. As a result, the F line swings to ground potential, the output unit releases the R_x and R_y lines, and the first axon stage is enabled. At this point, the latch circuit of Figure 4(b) maintains the state of the R_x and R_y lines.

The two axonal stages decouple the millisecond-timescale pulse widths required by the temporal adaptation circuitry from the nanosecond-timescale pulse widths necessary for fast communication. The subthreshold control voltage P sets the pulse width of the first axonal stage, while the above-threshold currents of the NFET transistors draining the buffer input node and the request lines set the communications latency.

6. Summary

This paper describes a working hybrid system that combines novel analog VLSI computation with mainstream digital computers. We describe an interface protocol that provides appropriate signal representations for both types of computation. We show an efficient implementation of this interface in a standard technology, including a complete logic and circuit level description, to encourage the incorporation of the protocol in other designs. Our system implementation, using a standard workstation running the Unix operating system, demonstrates the practicality of the interface in environments not specifically designed for real-time data processing.

7. Acknowledgements

Research and prototyping of the event-address interface took place in Carver Mead's laboratory at Caltech; we are grateful for his insights, encouragement, and support. The Caltech-based research was funded by the ONR, HP, and the Systems Development Foundation. Research and prototyping of the auditory-nerve demonstration chip and system took place at UC Berkeley, and was funded by the NSF (PYI award MIPS-895-8568), AT&T, and the ONR (URI-N00014-92-J-1672).

8. References

[1] R. F. Lyon, "CCD correlators for auditory models," in *IEEE Asilomar Conference on Signals, Systems, and Computers*, 1991.

[2] C. A. Mead, X. Arreguit, and J. P. Lazzaro, "Analog VLSI models of binaural hearing," *IEEE Transactions of Neural Networks* **2**: 230–236, 1991.

[3] L. Watts, R. Lyon, C. Mead, "A bidirectional analog VLSI cochlear model," *Advanced Research in VLSI, Proceedings of the 1991 Santa Cruz Conference,* C. Sequin (ed.), Cambridge, MA: MIT Press, pp. 153–163, 1991.

[4] W. Liu, A. Andreou, M. Goldstein, "Voiced-speech representation by an analog silicon model of the auditory periphery," *IEEE Transactions of Neural Networks* **3 (3)**: 477–487, 1992.

[5] R. F. Lyon, "Computational models of neural auditory processing," in *IEEE ICASSP,* 1984.

[6] S. Seneff, "A Joint Synchrony/Mean-Rate Model of Auditory Speech Processing," *Journal of Phonetics,* **16(1):** 55-76, 1988.

[7] Y. Muthusamy, R. A. Cole, M. Slaney, "Speaker independent vowel recognition : spectrograms versus cochleagrams," *IEEE ICASSP*, pp. 533-536, 1990.

[8] S. Greenberg, "The ear as a speech analyzer," *J. Phonetics,* vol 16, pp. 139-149, 1988.

[9] C. R. Jankowski, "A Comparison of Auditory Models for Automatic Speech Recognition," S.B. Thesis, MIT Dept of Electrical Engineering and Computer Science, May 1992.

[10] R. F. Lyon, C. Mead, "An analog electronic cochlea," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 36, pp. 1119–1134, 1988.

[11] J. P. Lazzaro, C. Mead, "Silicon models of auditory localization," *Neural Computation,* vol. 1, pp. 47–57, 1989.

[12] J. P. Lazzaro, C. Mead, "Circuit models of sensory transduction in the cochlea," in *Analog VLSI Implementations of Neural Networks,* Mead, Ismail, (eds.), Norwell, MA: Kluwer, pp. 85-101, 1989.

[13] J. P. Lazzaro, C. Mead, "Silicon models of pitch perception," *Proc. Natl. Acad. Sci. USA,* vol 86, pp. 9597–9601, 1989.

[14] J. P. Lazzaro, "Temporal adaptation in a silicon auditory nerve," In Tourestzky, D. (ed), *Advances in Neural Information Processing Systems 4.* San Mateo, CA: Morgan Kaufmann Publishers, 1991.

[15] J. P. Lazzaro, "A silicon model of an auditory neural representation of spectral shape," *IEEE Journal Solid State Circuits,* **26**: 772–777, 1991.

[16] M. Mahowald, Ph.D. Thesis, Computation and Neural Systems, California Institute of Technology, 1992.

[17] M. Sivilotti, "Wiring Considerations in Analog VLSI Systems, with Applications to Field-Programmable Networks," Computer Science Technical Report (Ph. D. Thesis), California Insitute of Technology, 1991.

[18] M. Konishi, "Centrally synthesized maps of sensory space," *Trends in Neuroscience,* **4**: 163–168, 1986.

[19] J. P. Lazzaro, "Low-power silicon spiking neurons and axons," *IEEE International Symposium on Circuits and Systems,* San Diego, CA, 1992, p. 2220–2224.