A Multi-Sender Asynchronous Extension to the AER Protocol

John Lazzaro and John Wawrzynek
Computer Science Division
UC Berkeley
Berkeley CA 94720-1776
{lazzaro,johnw}@cs.berkeley.edu

Abstract

The address-event representation (AER) is an asynchronous point-to-point communications protocol for silicon neural systems. This paper describes an extension of the AER protocol that allows multiple AER senders to share a common bus. A fully-functional silicon implementation of the extended protocol is described, as well as a functional board-level system of several of these chips sharing a common bus.

1 Introduction

Computation and communication are deeply interdependent in information processing systems; the proposed benefits of a novel method of computation often disappear once the realities of input and output are considered. This observation holds for biological processing systems as well as engineering computer systems: the methods that neurons use to communicate complement the mechanisms of neural computation, to realize an energy-efficient style of information processing.

Early work in analog VLSI modeling of neural systems focused on mapping the computational methods of neural computation on to analog circuits. To send the results of the computation off chip, these systems use traditional data acquisition techniques. An example of this approach is the original silicon retina design [4]. On this chip, weak-inversion analog circuits model sensing and computation in the mammalian retina, while a uniform-rate sampling system multiplexes the output array into a signal suitable for driving a video monitor.

Spending a few minutes in front of this silicon retina chip while observing its video monitor reveals the poor match between neural representations and uniformly-sampled communication techniques. The analog processing of the retina acts to accentuate spatial and temporal changes in contrast. If you hold your hand still in front of the chip, it fades from the screen in a few seconds; if you move your hand quickly, an outline of your hand appears. By coding changes in space and time with large signals, and regularity in space and time with null signals, the retinal representation is inherently adaptively compressed. However, a video signal lacks the representational power to exploit this compression.

The biological retina uses a different approach for communicating visual information to the brain, that exploits the adaptive compression of the retinal representation. The final analog signal for each image position is converted to series of fixed-width, fixed-height pulses; a dedicated wire for each image position carries these pulses to higher brain centers. A large input signal results in many pulses per second on the wire; a small input

signal results in few pulses per second. Since the retinal representation maps temporal and spatial constancy to small signals, most of the time, most of the retinal outputs will be sending very few pulses per second, and the energy used for communications is small compared to a uniform-sampling approach. We refer to neural representations that minimize the average pulse rate as *energy-efficient codes*.

The neural pulse representation has an interesting attribute: the signal imparts new information only at the moment a new pulse begins. The information in a group of (numbered) neurons can be compactly encoded as a list of pulse events, that tabulates for each pulse event the neuron number and precise time of pulse onset. A natural way of adapting neural pulse communication for silicon neural systems is to implement the output representation of a chip as an array of pulsing neuron circuits, and to send an event list describing pulse activity off chip to communicate the representation.

Several silicon neural models have implemented this approach to off-chip communication [7,5,1,6,2]. In these designs, an asynchronous arbitration tree acts to transform the pulse activity of an array of output units into a series of events; these events are sent off chip on an asynchronous data bus. These designs assume the receiver is listening to the data bus with a constant latency, and do not send timestamp information. An event consists of a single asynchronous bus transaction sending the number of a neuron, and is sent at the moment of pulse onset. We call a protocol that communicates event lists without timestamps an *address-event representation (AER)*.

An off-chip asynchronous bus is capable of transmitting millions of events per second; an output unit using energy-efficient coding may have an average pulse rate of 1-100 pulses per second, depending on the application. This large difference in speed allows many output units to share an off-chip bus without significant loss of timing information. Analysis in [1,2] shows the performance limits of this architecture as a function of the number of output units, the behavior of the units, and desired timing accuracy, for an auditory application.

In auditory applications, a single chip can compute an interesting neural representation; however, real-world applications often require several different representations of an audio signal, that code for different aspects of sound (for example, separate representations for spectral profile and temporal onset detection). Multiple copies of single chip, with different control parameters, can compute these representations independently [2].

However, while connecting a single AER port to a receiver is a simple exercise, connecting several ports to a single receiver is more difficult. As each AER port has its own dedicated data bus, the fan-in of the receiver grows linearly with the number of input ports. Moreover, each AER port has its own asynchronous control signals, forcing the receiver to implement an arbitration system.

This paper describes an extension of the AER protocol, that allows many chips to send events on a shared asynchronous bus. This extension takes the form of a modified version of the AER sender port; no external logic is needed for arbitration or bus-sharing. The paper describes an auditory chip that includes the modified sender port, and a board implementation of several of these chips sharing a common asynchronous bus. The chip has been fabricated and tested, and the board implementation is fully functional.

2 An AER implementation

Our multiple-sender AER implementation is an extension of the AER architecture shown in [2]. We begin the description of our multiple-sender extension with a review of the original architecture.

2.1 The output unit abstraction

An output unit converts an analog signal into a stream of fixed-width, fixed-height pulses. In this paper, we are unconcerned about the specifics of this conversion, but instead focus on the digital interface of an output unit. Figure 1 shows an output unit; a typical pulse pattern for an output unit is shown as P_i. The unit has a request output, r, that signals the onset of each P_i, and an acknowledge input, a, that acts to reset the r signal.

Typical r and a signals are shown in Figure 1; these lines implement a 4-cycle signalling convention. The first two pulses of P_i illustrate that the width of r is only dependent on the acknowledge signal a, and can be shorter or longer than the pulse width of P_i. The third and fourth pulses of P_i illustrate the lack of buffering in an output unit: data can be lost if an acknowledge signal happens too slowly.

2.2 Generating a signals

Our goal is to multiplex the onset events of N output units onto a single bus. One way to realize this goal is to design a system that observes the $r_1 \ldots r_n$ request signals from the output units, and generates the $a_1 \ldots a_n$, in such a way that only one output unit is being acknowledged at a moment in time.

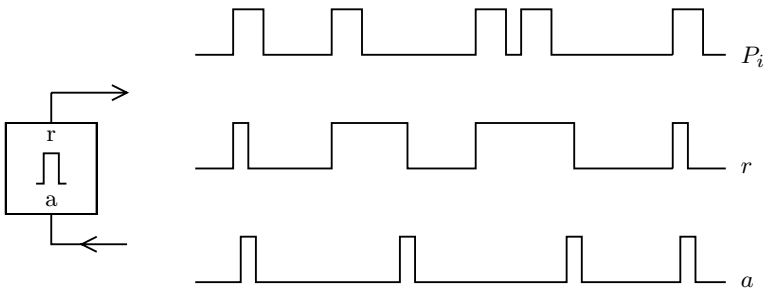

Figure 1. Symbol for an output unit, with request output r and acknowledge input a. Traces show typical behavior of unit; P_i is the pulse waveform whose onsets are coded by r.

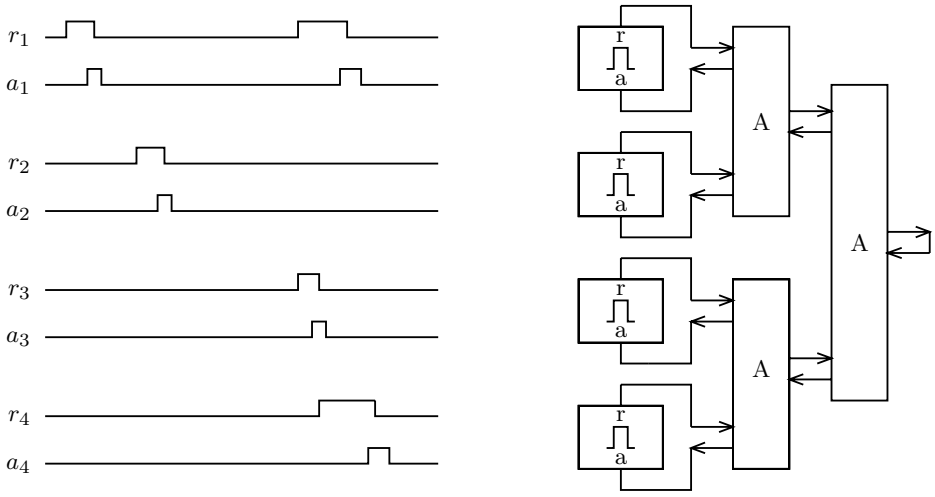

Figure 2. Four output units, being reset by a 4-input arbitration tree. Each box marked with an A is a two input arbiter. Note that no two a_i signals are high simultaneously.

An N input asynchronous arbiter, using the 4-cycle signalling convention, will generate $a_1 \ldots a_n$ given $r_1 \ldots r_n$ under this constraint. We use a tree of 2-input arbiters to implement this N input arbiter. Figure 2 shows an implementation of 4 output units serviced by an arbitration tree; the boxes marked A are two-input arbiters as implemented in [2]. The traces in Figure 2 show typical behavior for staggered and simultaneous pulse patterns.

2.3 Generating an asynchronous output bus

The arbitration tree in Figure 2 acts to order the asynchronous activity of the output units into a sequence of individual events. To communicate this event sequence on a parallel bus with 4-cycle asynchronous control signals, we make the following additions to the system:

- We generate a request control signal for the bus, R_c, by a logical OR of the a_i signals.
- We generate a binary encoding of the a_i signals to create the data outputs of the bus, D_o and D_1.
- We add control logic between the output units and the arbitration tree, to latch the current state of the inputs to the arbitration tree at the moment an R_c signal occurs. This latch is released when the bus acknowledge signal, A_c, occurs.

Figure 3 shows these changes, and a typical signal sequence for an event. By expanding the depth of the arbitration tree and adding additional data lines, chips with linear arrays of 120 output units have been implemented [1]; in addition, a straightforward extension to this architecture supports chips with two-dimensional arrays of output units [2].

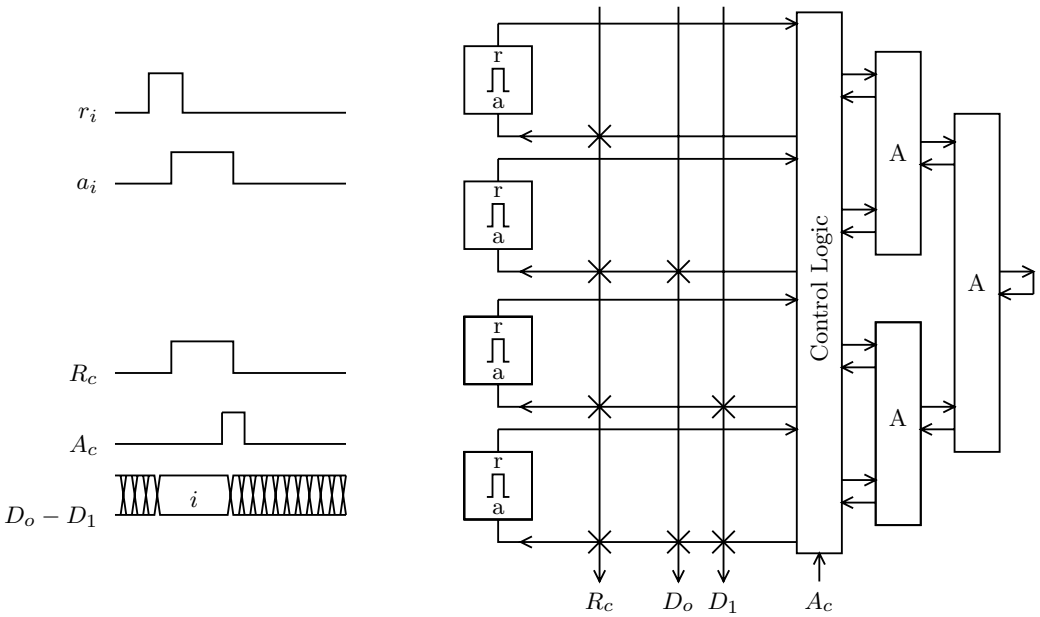

Figure 3. A complete implementation of the address-event protocol. The encoder lines form the data (D_o, D_1) and request (R_c) outputs of the off-chip bus. The box marked "Control Logic" acts to latch the inputs of the arbiter tree; off-chip bus input A_c clears the latched state and resets the system to send the next event.

3 Requirements for a multi-sender extension

To connect several senders to a single receiver, additional hardware is needed to arbitrate between the senders, create new control signals, and multiplex the data bus. The design presented in this paper modifies the implementation in Figure 3, so that many senders chips can share the control and data lines of a single 4-cycle asynchronous bus without additional hardware. Figure 4 shows a conceptual block diagram of several senders sharing a common bus; this diagram illustrates the three functions needed for bus sharing:

• Arbitrate among the senders, to ensure only one chip is communicating an event at a time. In Figure 4, connections between senders illustrate this function.

• Communicate which sender is currently using the bus. In Figure 4, the bus lines W_o and W_1 encode the identity of the sender.

• Electrically share wires. Many chip outputs connect to common wires in Figure 4; the physical layer of bus sharing must be specified.

Several design goals guided our implementation choices for these three functions. Our primary goal was the integration of all three functions into the sender, so that no additional hardware would be needed to build a system. This requirement dictated a distributed approach to arbitration and sender encoding. We also wanted a system that did not require programming a unique identification number into each sender: in our design, the sender information encoded on the W lines is computed in a distributed fashion by the arbitration system.

4 The multi-sender AER extension

In our extension, each AER sender becomes a node in a board-level arbitration tree, that is an extension of the chip-level arbitration tree of the original sender design. Figure 5 shows the design of the modified sender. The arbitration tree on the left side of Figure 5 is identical to tree in Figure 3, and acts to sequence the acknowledgements of output units. The R_c, A_c, and D_i signals have identical roles as in the original sender.

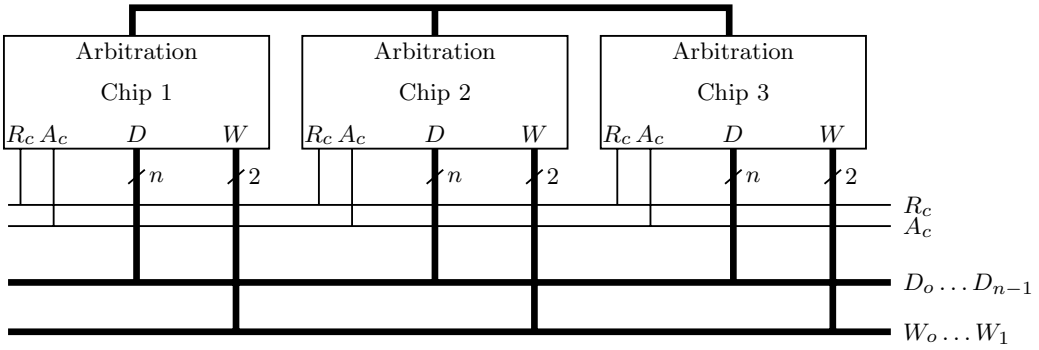

Figure 4. Three sender chips sharing the same bus. Control signals R_c and A_c implement a 4-cycle handshake for the data bus, consisting of intra-chip event address $D_o \ldots D_{n-1}$, and chip number $W_o \ldots W_1$. Arbitration communication between chips coordinates bus sharing.

The auxiliary tree on the right side of Figure 5 arbitrates between the root handshake pair of the internal tree and the off-chip handshake pairs (R_l, A_l) and (R_r, A_r). The root handshake pair (R_m, A_m) of the auxiliary tree is sent off chip. By the appropriate interconnection of the (R_l, A_l), (R_r, A_r) and (R_m, A_m) signals of different senders, we can design a board-level arbitration tree that ensures only one output unit in the entire system is acknowledged at a given time.

In Figure 5, several signals are derived from the acknowledge signals of the auxiliary tree:

- $D_l = A_l + A_{\text{internal}}$
- $D_r = A_r + A_{\text{internal}}$
- $R_t = A_l + A_r + A_{\text{internal}}$.

The D_r and D_l signals are sent off chip. By the appropriate interconnection of the D_r and D_l signals of different senders, we can produce a distributed computation of the W_i signals that encodes the identity of the current sender. The R_t signal, in concert with the R_c signal produced by the internal tree decoder, is used to coordinate the physical level of bus sharing. Figure 6 shows the schematic symbol we will use to represent the modified sender.

5 Interconnecting modified senders

Figure 7 shows the interconnection of 7 senders to form a fully-populated arbitration tree of depth 3. The R_m and A_m signals of the sender at the root of the tree (sender 4) are connected together; the R_l and R_r inputs of the senders at the leaves of the tree (1,2,5,6) are grounded. The R_c signals from all senders are tied together; the A_c and D_i signals are connected in the same fashion.

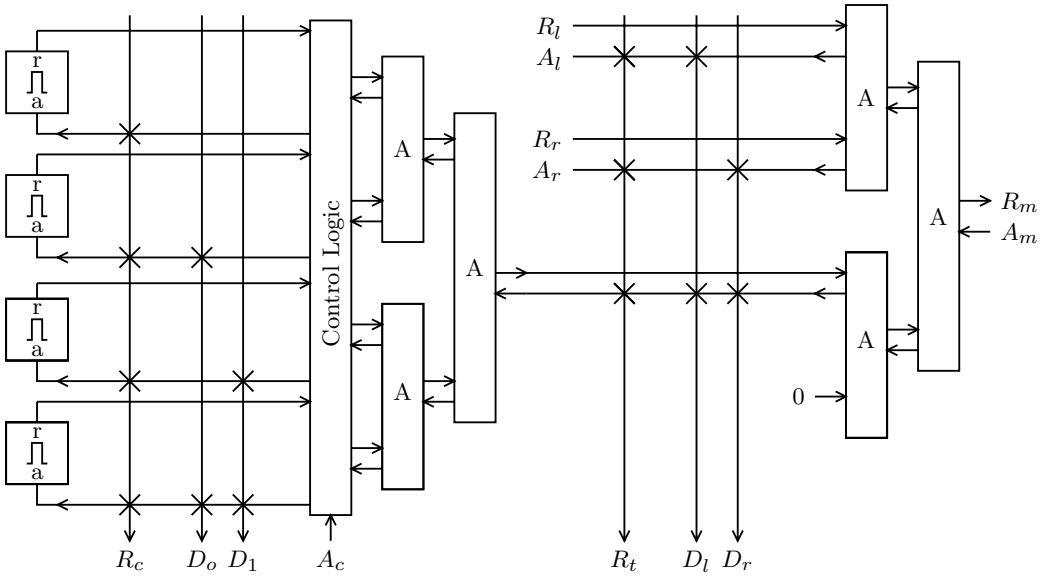

Figure 5. The new implementation of an AER sender. See main text for details.

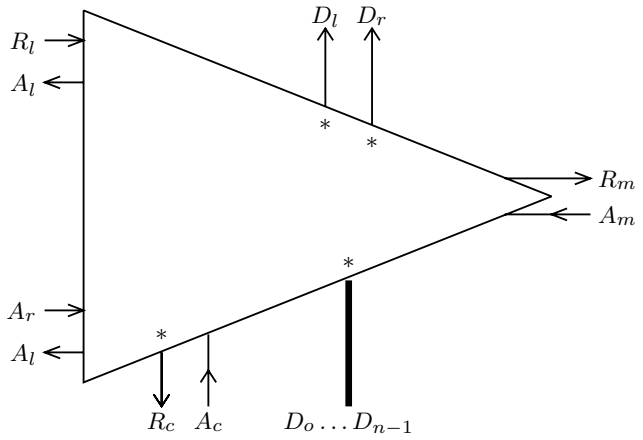

Figure 6. Schematic symbol for the AER sender in Figure 5. Signals marked with a $*$ use tri-state pads shown in Figure 7, all other pads are normal digital input or output pads. Note that signal R_t in Figure 5 is not sent off-chip, but is used internally for tri-state pad control.

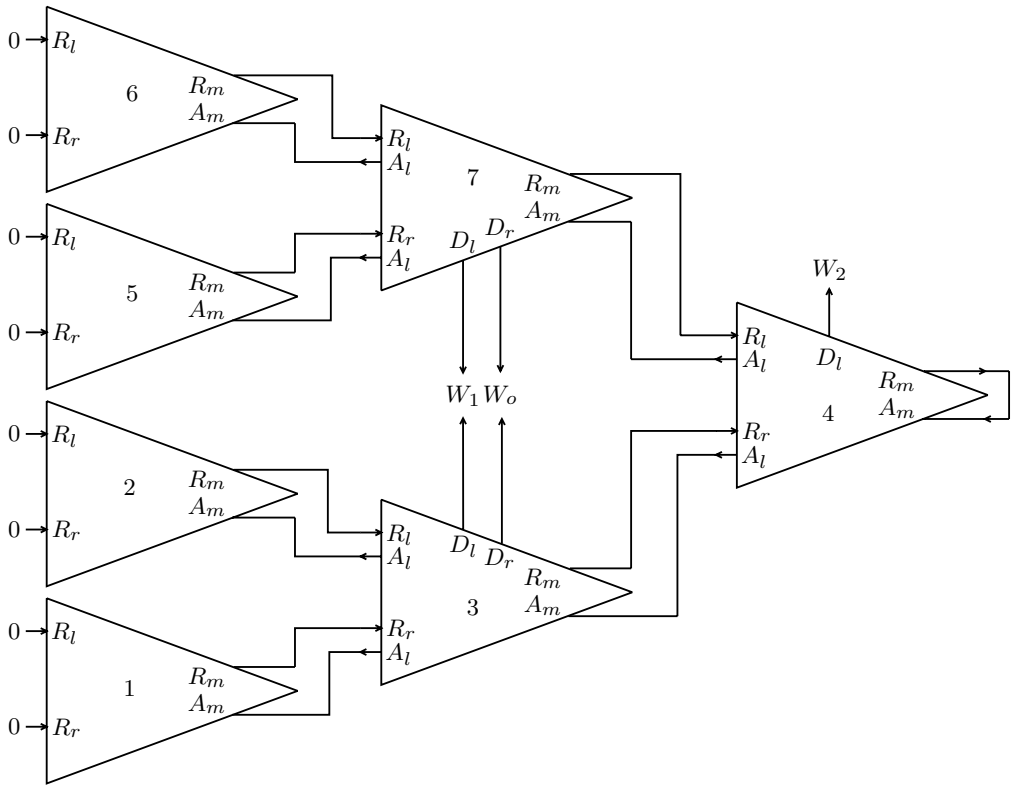

Figure 7. Seven senders connected to form a 3-level binary tree. The R_c, A_c, and $D_o \ldots D_{n-1}$ signals of all senders are tied together, and are not shown. Signals $W_o \ldots W_2$ encode the identity of the sender driving the bus; the number in each sender symbol is the value driven on the $W_o \ldots W_2$ lines as it drives the bus.

The D_l and D_r outputs of senders in particular tree positions are connected together to produce the W_i signals that identify the chip currently sending an event. The D_l and D_r signals of the leaf-level senders (level 1) are unused. All D_r signals of level 2 are connected to produce W_o, and all D_l signals of level 2 are connected to produce W_1. For all tree levels $i > 2$, the D_r outputs are not used, and the D_l outputs are all connected together to produce W_{i-1}. The number label of each sender in Figure 7 is the binary number encoded by (W_2, W_1, W_o) during an event from this sender.

We can make several observations about the behavior of the shared output lines in this system (R_c, D_i, W_i). Because the arbitration system enables only one sender at a time, and because of the definitions of R_c and D_i, only one sender at a time can drive the R_c and D_i lines to the high state. In addition, the definitions of D_l and D_r, coupled with the interconnection rules stated above, guarantee that only one sender can drive the W_i lines high at a time.

These observations, and the properties of the 4-cycle bus handshake, prompted us to use a well-known technique for tri-state bus sharing. Our tri-state pad design is shown in Figure 8. In the absence of a driving signal, the cross-coupled inverters to the right of the pad act to maintain the state of the pad output. The signal associated with the pad is connected to the active-high input P_{in}, and acts to drive the pad to V_{dd}. The active-low input N_{in} acts to drive the pad to ground, and is connected to a control signal that briefly pulses low at the end of every 4-cycle handshake on the (R_c, A_c) signals.

At the end of every 4-cycle handshake, the output of the tri-state pad is reset to ground by the control signal connected to N_{in}. In the course of the next event transaction, the signal connected to P_{in} may drive the pad output V_{dd}; if not, the pad remains at ground. At the end of the handshake, the pad is reset to ground again by the signal connected to N_{in}.

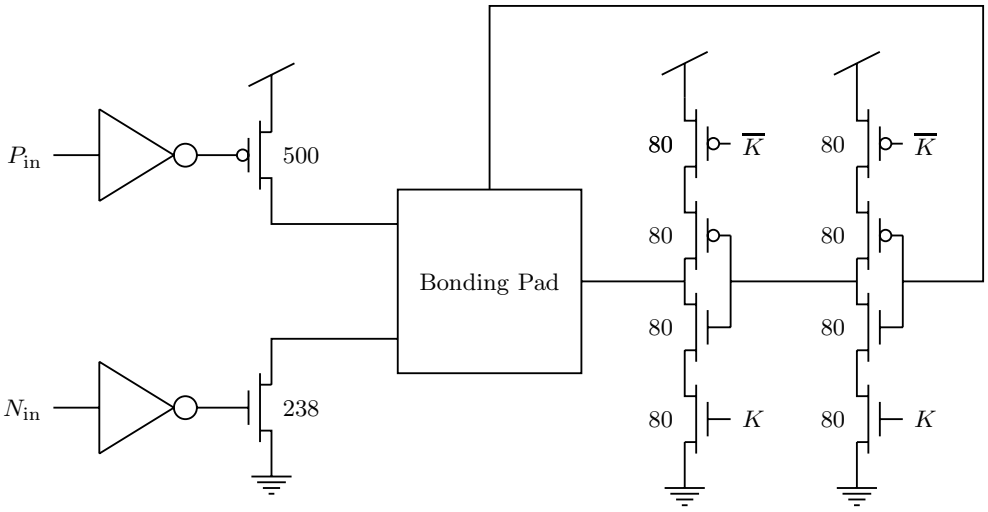

Figure 8. Tri-state pad design. The cross-coupled inverters on the right act to retain the state of the bus; the drivers on the left force the state of the pad to change. The cross-coupled inverters can be disabled on a per-chip basis using the signal K. Numbers next to transistors indicated total gate length in microns.

The circuit shown in Figure 9 generates the control signal for N_{in}. For the D_l and D_r pads, the R signal in Figure 9 connects to the R_t signal shown in Figure 5; for the R_c and D_i pads, the R signal in Figure 9 connects to the R_c signal shown in Figure 5.

The output driver transistors controlled by N_{in} and P_{in} are strong compared to the transistors in the cross-coupled inverters; this difference allows the driver transistors to overpower the stored state of the inverter pair easily. Power is dissipated only during these transitions; an open-drain bus using passive pullups, in contrast, has static power consumption in the low logic state.

The K control signal on each pad supports the selective disabling of the cross-coupled inverters. If multiple tri-state pads are connected together, only one inverter pair should be enabled; this practice ensures the state of the pad can be overdriven. A simple approach is to connect the K's of R_c and all D_i's together to form signal K_1, and connect the K's of D_l and D_r together to form signal K_2, and provide off-chip inputs for setting K_1 and K_2.

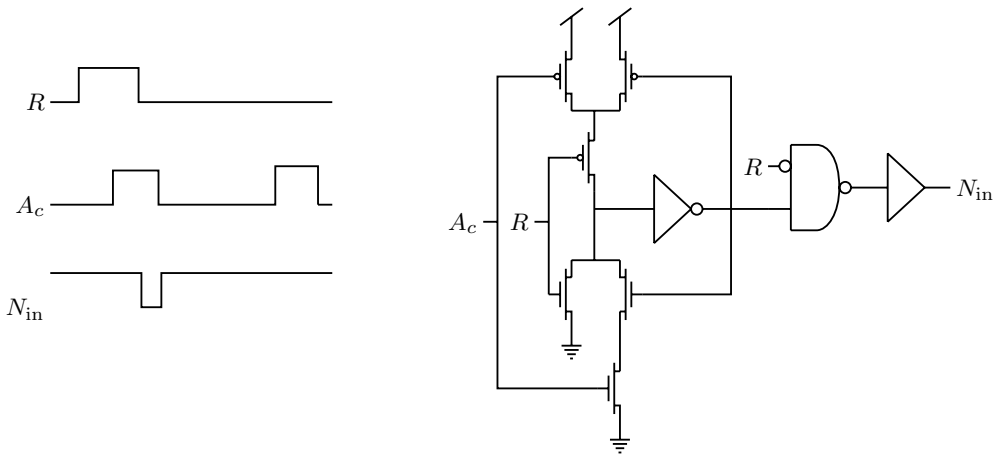

Figure 9. Circuit to reset the cross-coupled inverters in Figure 8 at the end of every bus transaction. To reset the W_i pads, use the R_t signal in Figure 5 for R; to reset the D_i and R_c pads, use the internal R_c signal generated by the decoder line in Figure 5 for R.

6 Experimental results

We designed a test chip containing two logically separate modified AER senders, each with 8 output units; only the A_c signal is shared between the two senders. The output units were driven by an auditory nerve model, as described in [3]. The chip was fabricated on the Orbit 2μ low-noise analog n-well process, in a TinyChip 40-pin package. Pin positions of tri-state pads were placed at adjacent positions at the ends of the package, to induce worse-case noise coupling.

Using 4 chips, we constructed a wire-wrap board implementation of the system shown in Figure 7. We connected the asynchronous bus from this board to the AER data display system described in [2,1]; this S-bus-based system allows continuous, real-time data acquisition and display of AER data up to 1 Mega-event per second.

We configured the analog auditory model so that each chip was tuned to a different audio frequency range. Experiments with tone sweeps confirmed correct operation of the

seven chips; other auditory experiments with a low-frequency square wave input confirmed the gross temporal performance of the communications system.

By driving high-amplitude white noise into the cochlear model inputs, we were able to generate an aggregate pulse rate on the 56 output units (8 output units for 7 senders) sufficient to cause back-to-back bus cycles. Figure 10 shows a typical bus cycle from the 7-sender system in this situation. The 1 Mega-event per second performance of our data acquisition system clearly limits the speed of the bus cycle; the $\tau_{a\overline{r}}$ and $\tau_{\overline{a}r}$ signals are short compared to the acquisition-system delays (see caption).

To place an upper bound on the cycle time of our 7-sender board, we disconnected the board from the data acquisition system and connected the R_c and A_c signals together. Under these conditions, the period of R_c during a multiple-event transmission varied from 100 to 140 ns. The variations reflect the delays of different nodes on the arbitration tree.

Because of the pin and area limitations of combining two auditory models and AER sender ports in a TinyChip payload, the functionality of the auditory model is limited; many auditory model control voltages were grounded internally because pads were in short supply. The practical use of this system in auditory applications is limited; the purpose of the chip was to test the modified AER sender design in an economical fashion.

A version of the 120-channel auditory model presented in [1], retrofitted with this modified AER sender design, has also been fabricated. A three-sender system built with this chip functions correctly, and exhibits bus transaction speeds consistent with the data above.

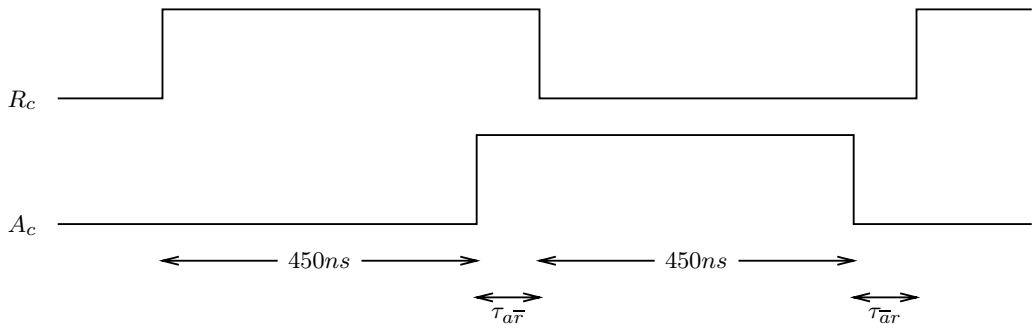

Figure 10. Typical waveforms for R_c and A_c from the 7-sender board implementation, connected to an SBus interface board with a $1\mu s$ cycle time. Typical $\tau_{a\overline{r}}$ were $25ns$; typical $\tau_{\overline{a}r}$ were 40-70ns.

7 Discussion

The multi-sender AER protocol has direct applications in auditory processing. A variety of interesting auditory representations coding spectral shape, pitch, auditory localization, and other properties can be implemented on a single die, along with non-volatile electrically programmable parameter storage and an AER sender port [1].

Routing input to an auditory model chip is simple: a shielded audio cable suffices. This technique scales very well to hundreds of auditory model chips. If these auditory model chips use energy-efficient coding and the multi-sender AER protocol, combining the output of many auditory models is also straightforward. In this way, a large collection of auditory model chips function as special purpose analog-to-digital conversion system, producing specialized representations of sound in digital form from an analog audio input.

Would such a system be useful? Recent research in audio signal processing has centered on building auditory scene analysis systems: sound processing systems that break up a sound signal into many different representations before making decisions on the data [8]. These systems are analogous to machine vision systems that compute motion, color, and edge representations on a raw input image before performing a task. A collection of silicon auditory model chips communicating with a host computer would function as a real-time, low-power hardware accelerator for auditory scene analysis.

The multi-sender AER technology, in its present state, is less relevant to silicon visual models. Multiple chips are needed to compute a single non-trivial visual representation, in suitable resolution for visual scene analysis. The partitioning of a visual representation over several chips can take many forms, but all methods require individual chips to both send and receive visual scene information. If the chips communicate using AER, this requires an AER receiver port and an AER sender port on each chip. Building vision systems using many chips requires an AER extension that handles multiple receivers and multiple senders, with sufficient flexibility to implement the required interconnections for vision algorithms.

One general communications architecture for vision chips is broadcast: every vision chip sends all event information to every other chip in the system [5]. The multi-sender AER technology presented in this paper can be extended to support broadcast in a straightforward way. In its present form, the control information (R_c, A_c) and intra-chip event information $(D_o \ldots D_{n-1})$ sent to the AER receiver is implicitly broadcast to every chip in the system. If the (R_c, A_c) handshake is sufficiently long, any chip in the system is able to "snoop" on the transaction between the sender chip and the host AER receiver. If the W bus is also distributed to every chip in the system, the identity of the chip sender is also available to be snooped, and a full broadcast operation would be implemented.

The speed limitation on the (R_c, A_c) handshake is a function of the physical size of the complete system; a dense packaging technology would improve the performance of the system. Alternatively, an asynchronous approach to bus snooping could be implemented.

Acknowledgements

This research was funded by the Office of Naval Research (URI-N00014-92-J-1672), the National Science Foundation (PYI award MIPS-895-8568 and Infrastructure Grant CDA-8722788) and AT&T. Thanks to Krste Asanovic, James Beck, Buster Boahen, Steve Deiss, Bertrand Irissou, Dick Lyon, Alan Kramer, Brian Kingsbury, Carver Mead, Misha Mahowald, and Massimo Sivilotti for suggestions about the design and presentation of this work.

References

[1] Lazzaro, J. P., Wawrzynek, J., and Kramer, A (1994). Systems technologies for silicon auditory models. *IEEE Micro*, **14**:3. 7-15.

[2] Lazzaro, J. P., Wawrzynek, J., Mahowald., M., Sivilotti, M., Gillespie, D. (1993). Silicon auditory processors as computer peripherals. *IEEE Journal of Neural Networks* **4**:3 523–528.

[3] Lazzaro, J. and Mead, C. (1989). Circuit models of sensory transduction in the cochlea. In Mead, C. and Ismail, M. (eds), *Analog VLSI Implementations of Neural Networks.* Norwell, MA: Kluwer Academic Publishers, pp. 85-101.

[4] Mahowald, M.A. and Mead, C. (1991). The silicon retina. *Scientific American*, V264 N5:76-82.

[5] Mahowald, M. A. (1992). "Computation and Neural Systems," Ph.D. dissertation, California Institute of Technology.

[6] Mortara, A. and Vittoz, E. A. (1994). "A communications architecture tailored for analog VLSI artificial neural networks – intrinsic performance and limitations", *IEEE Journal of Neural Networks*, 1994 May, V5 N3:459-466.

[7] M. Sivilotti (1991) "Wiring concerns in analog VLSI systems, with applications to field-programmable networks," Computer Science Technical Report, Ph. D. dissertation, California Institute of Technology.

[8] M. Cooke, S. Beet, and M. Crawford (1993). "Visual Representations of Speech Signals." John Wiley & Sons, New York.