# A 5 Meps $100 USB2.0 Address-Event Monitor-Sequencer Interface

R. Berner*, T. Delbruck*, A. Civit-Balcells† and A. Linares-Barranco†
*Institute of Neuroinformatics, UNI-ETH Zurich, Switzerland
†Department of Architecture and Technology of Computers, University of Seville, Spain

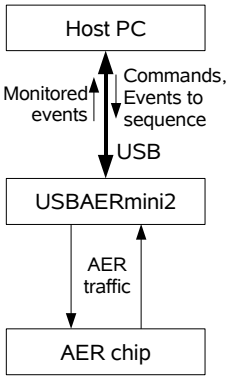
***Abstract*—This paper describes a high-speed USB2.0 Address-Event Representation (AER) interface that allows simultaneous monitoring and sequencing of precisely timed AER data. This low-cost (<$100), two chip, bus powered interface can achieve sustained AER event rates of 5 megaevents per second (Meps). Several boards can be electrically synchronized, allowing simultaneous synchronized capture from multiple devices. It has three parallel AER ports, one for sequencing, one for monitoring and one for passing through the monitored events. This paper also describes the host software infrastructure that makes the board usable for a heterogeneous mixture of AER devices and that allows recording and playback of recorded data.**

## I. Introduction

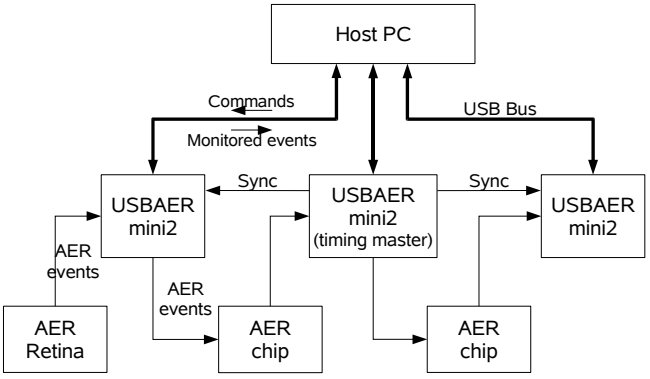
The Address-Event-Representation (AER) is an asynchronous protocol for transmitting spike-based information between chips. AER encodes spike-based data as digital addresses, either of sending or receiving units. There is a growing community of AER protocol users for neuromorphic applications in visual, auditory, and multi neuron learning systems. The goal of this community is to build multi-chip and multi-layer hierarchically structured systems capable of performing spike-based parallel processing in real time.

A number of previous generations of AER-computer interfaces have been built and are in present use [2], [3], [4], [5], but none of them had the convenience of high-speed bus-powered USB combined with synchronous monitoring of AER traffic. For our CAVIAR project [6], [7] we assembled a heterogeneous mixture of AER chips into a visual system, and we needed a device that could record from all parts of the system simultaneously, was simple to operate and use, was easy and cheap to build, and could be reused in other contexts. We also wanted a convenient device that could sequence recorded or synthesized events into AER chips to allow their characterization.

The functionality of the USBAERmini2 board was evolved from the monitoring capability which was developed for [8] to include sequencing, passing through of monitored events to a receiving AER chip and the synchronisation of several devices. The number of jumpers is minimized by auto detection of connected devices. Using the device is very simple. For example, a user connects a sending AER device to the monitor port and plugs the board into a USB port on a computer. Then they are ready to capture and time-stamp AER traffic from the AER device. Fig. 1 shows two example setups.




(a) Basic operation example: the USBAERmini2 is used to stimulate an AER chip and to monitor its response.




(b) Operation example: Several USBAERmini2s are used to synchronously monitor a system [6], [7]. The USBAERmini2s are in pass-through mode, which means they read the incoming AER data and simultaneously send it to the next AER chip and to the host computer. The USBAERmini2s are synchronized with a cable.

Fig. 1. Shows the functionality of the USBAERmini2 by two example setups.

## II. The Hardware

The board consists of two main components, the Cypress FX2LP USB2.0 transceiver and a Xilinx Coolrunner 2 CPLD with 256 macroblocks (XC2C256) (see fig. 2). The board was designed to be simple and cheap to manufacture. We recently hand-assembled ten boards with two people in three days. The components cost less than $100 per board, including the 2-layer PCB.

The Cypress FX2LP is an USB2.0 transceiver with an enhanced 8051 microcontroller and a flexible interface to its

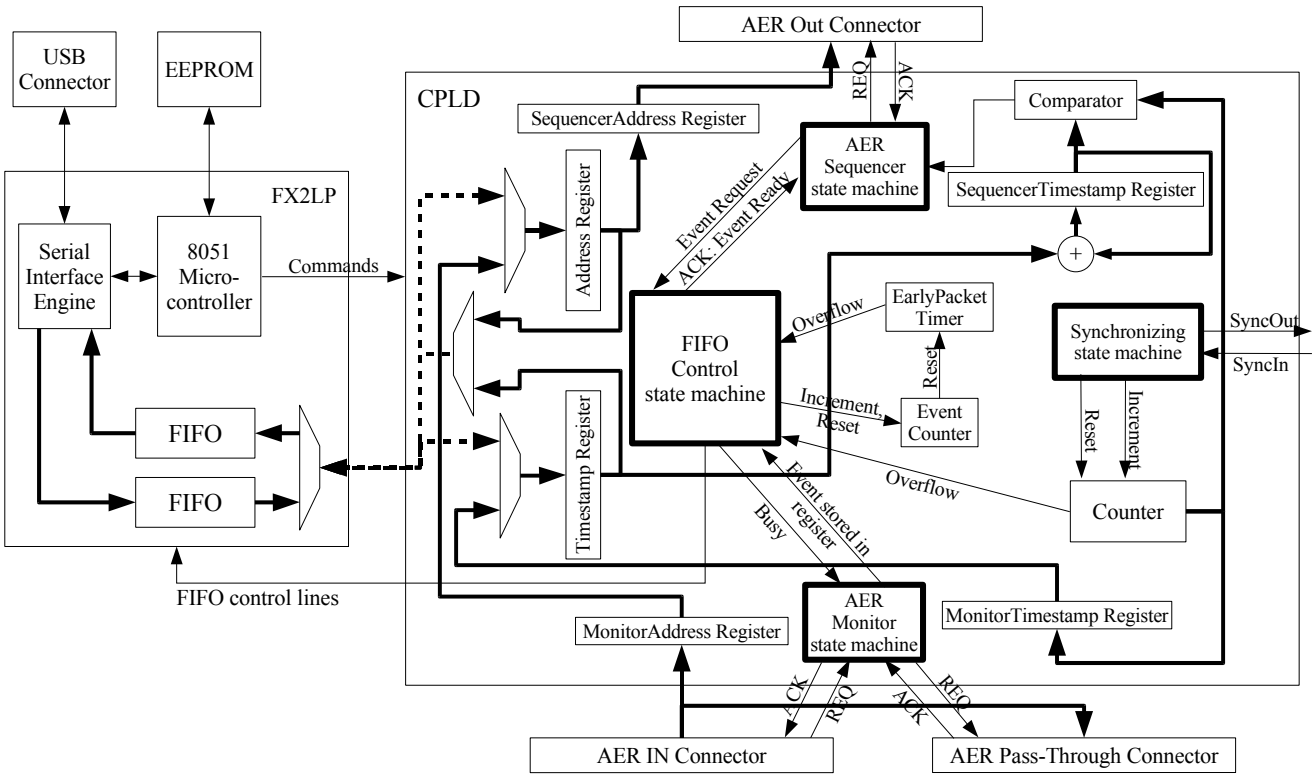



Fig. 2. This figure shows an overview of the device. The CPLD implements four FSMs (bold boxes) to achieve to desired functionality. The AER bus consists of 16 address lines, a request and an acknowledge line. Bold arrows stand either for address or for timestamp buses. The CPLD and the FX2LP are connected through several control lines and a 16 bit wide data bus (bold dashed) which is connected to the FX2LP FIFOs through a multiplexer. This multiplexer is controlled by the FIFO control state machine.

4 kB of first-in first-out (FIFO) buffers, which are committed automatically from or to the USB by the Cypress serial interface engine (SIE). As in [5], we used the FX2LP in its slave FIFO mode, which means that the device handles all low level USB protocol in hardware. We used USB bulk transfers. We used two FIFOs (one in each direction), which are configured to be quad buffered and hold 128 events each. To the CPLD the FX2LP appeared as a FIFO source or sink of AER data.

The microcontroller controls the communication with the host computer. This involves the setup of the USB communication and the interpretation of commands that are received from the host to the control endpoint (like *start capturing*, *stop capturing*, or *zero timestamps*). It also communicates with the CPLD to select operation mode (monitoring and/or sequencing) and additional configuration possibilities.

The CPLD handshakes with AER sender and receiver and records a 16-bit timestamp whenever an event is received. Addresses and timestamps are written to or read from the FIFOs in the FX2LP. This is done with four finite state machines (FSM). There is an FSM responsible for monitoring, one for sequencing, one for accessing the FX2LP FIFOs and one for synchronizing to a timing master board. Simplified state diagrams for the monitoring state machine and the sequencing state machine are shown in figure 3 and 4 respectively.

In sequencing mode, addresses and inter-spike intervals are read from the FX2LP FIFO and stored in registers. The inter-spike intervals are added to the timestamp of the last event. This is done to get an absolute timestamp which can be compared to the timestamp counter. Because the CPLD has no internal RAM, events are read from the FIFO only one by one. The events are sent from the host computer to the FX2LP as fast as possible and then wait in the FIFO until they are sequenced.

The CPLD uses a 16 bit counter for the generation of timestamps. These timestamps are unwrapped to 32 bits on the host computer. The timestamp tick period can be controlled from the host computer. Two timestamp modes can be used, either a tick of one microsecond or a tick of one clock cycle, which is $33\frac{1}{3}ns$. When the timestamp counter overflows, a special "timestamp-wrapped" event is sent to the host to tell it that the timestamp wrap counter has to be incremented.

We chose to send explicit timestamps for each event because we were interested in precise event timing. [5] did not send timestamps but instead used a special "heartbeat" address sent as regular intervals. Their scheme uses less USB bandwidth but requires interpolation to reconstruct timestamps.

92% of the CPLD macroblocks are used.

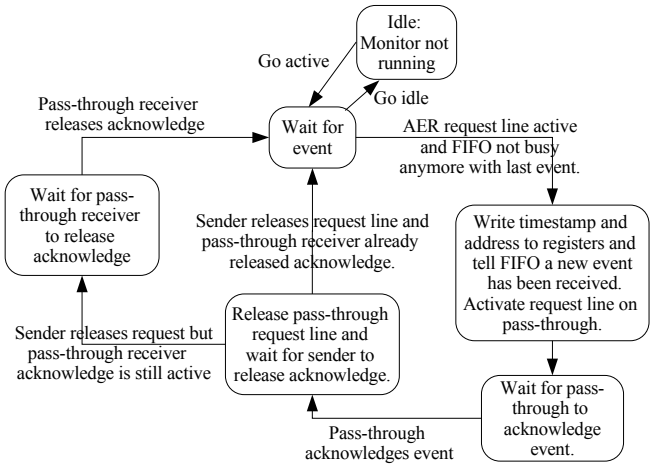



Fig. 3. Simplified monitor state machine, showing only the part which in used in pass-through mode, i.e. when the monitored events are passed to another AER device. There are additional states for the device acting in terminal mode (i.e. as in Fig. 1(a)). By a pull-down resistor on the acknowledge line of the pass-through, the device is able to detect if there is a receiving device present or not. The state machine is activated by an input to the CPLD and then waits for the request line to be active.

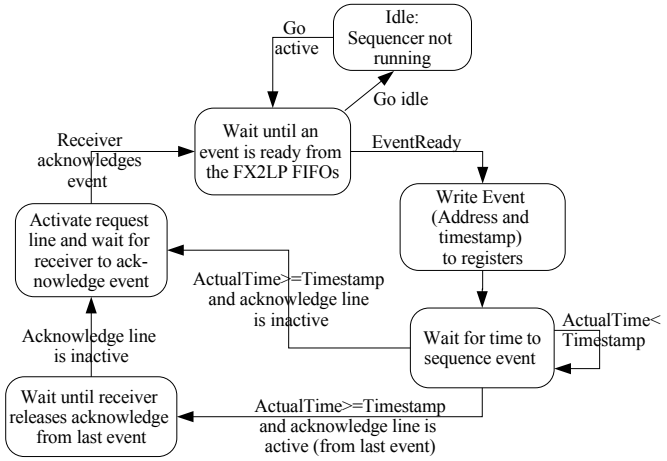



Fig. 4. Sequencer state machine. When this state machine goes active, it signals the FIFO state machine a request for an event. If the FIFO state machine is not busy, it will read an event from the FX2LP FIFO, unless this FIFO is empty. As soon as this event is stored in the registers, its timestamp is compared to the counter, to determine when to activate the request line. Interspike intervals of up to $2^{16} - 1$ timestamp ticks are possible.

### *Special Features*

An "early packet" circuit ensures a maximum inter-packet interval of 10 ms. This feature is necessary for real time visualization of applications with a low event rate and for real-time control based on AER data. This circuit commands the FX2LP to commit a FIFO buffer to the SIE when the early packet timer expires, even if the buffer is not full. The timer is reset every time a packet is sent.

Electrical synchronization enables timestamp-synchronized capture from multiple AER devices. A USBAERmini2 in "slave" mode can be synchronized to a timing "master", which

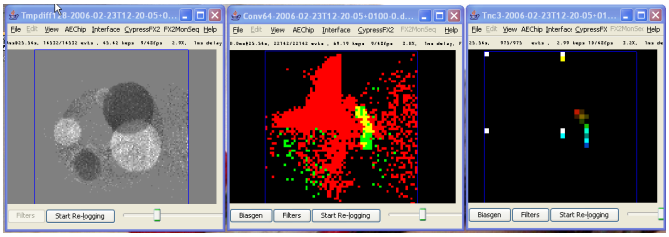


Fig. 5. Screen shot of host application playing synchronized recording from 3 different AER chips [6], connected in series with the USBAERmini2 monitoring each chip's output (Fig. 1(b)). Each chip uses a different rendering method.

generally is another USBAERmini2. This synchronization is implemented by a dedicated FSM and an input and an output pin.

As soon as the synchronization state machine detects a timestamp master at the synchronization input (i.e. the sync input goes high), it resets the timestamp, signals to the host by sending a USB control transfer message, and changes to slave mode. The host resets its timestamp wrap counter so that all slaves have the same absolute timestamp as the master.

## III. Software

The host-side software was designed for usability and flexibility in interfacing to a variety of AER chips over a variety of hardware interfaces. This host side of the interface has been implemented entirely in Java (1.5 JVM). An object-oriented software architecture consisting of about 200 classes allows for capturing events from multiple hardware sources, rendering events to the screen (as viewable frames or other representation, e.g. space-time) and recording and playing back AER activity (Fig. 5). Using the Java Bindings for OpenGL (JOGL) greatly increased graphics rendering performance. These Java classes can be used directly from Matlab.

We used the Thesycon (www.thesycon.de) USB device driver for Windows. It is reliable and has a convenient Java interface. The USB host-side interface uses a high-performance multi-threaded buffer-pool for both reading and writing event packets. This infrastructure is provided as part of the Thesycon USB driver kit. The reader and writer threads run asynchronously from other threads, e.g the rendering thread. Monitoring events means that a set of asynchronous reads are launched using a pool of memory buffers. When these reads complete because data is received from the device, a handler is called and the events can immediately be processed or written to another "user" buffer that the rendering thread can later access. After each buffer is processed, a new read is launched on that buffer. This standard approach for decoupling between device data and data processing was not previously used in AER interfaces and it makes software development much easier.

Each AER device (e.g. retina, cochlea, multi neuron chip) is specified as a subclass of a general AEChip class. This subclass includes a method that maps raw AER addresses to meaningful x,y,type information. The software framework is

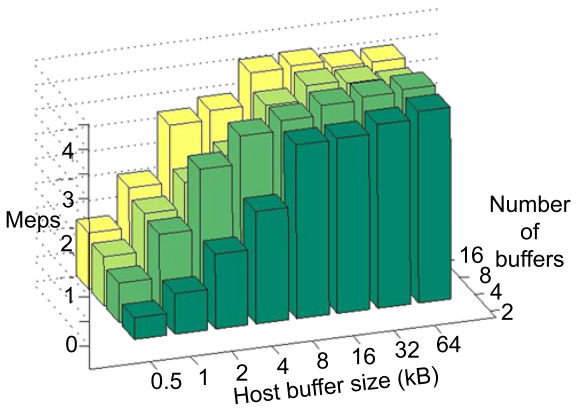



Fig. 6. Measured monitoring performance versus number and size of host memory buffers.

therefore easily extended to support new AER chips, only a new subclass of AEChip has to be generated, which involves very little programming. Each device can have its own set of rendering and event-processing methods. Multiple devices can be viewed simultaneously and their data recorded and played back in synchrony (Fig. 5).

## IV. Performance

The peak limit on monitoring and sequencing performance is set by the number of clock cycles used by the state machines for a handshake cycle. The monitor state machine requires 5 clock cycles and the sequencer requires 8 clock cycles. The CPLD clock frequency is 30 MHz, so the peak event rate for monitoring is 6 million events per second (Meps) and for sequencing it is 3.75 Meps.

The measured performance depends on the number and size of host memory buffers. For high frame-rate visualization at low event rate the host buffer size can be set to the USB transaction size of 512 bytes; for highest performance, the buffers need to be at least 4 kB. Figure 6 shows the measured event rate versus buffer size and number running on a 3 GHz desktop PC for a single board. With a buffer size of 8 kB and 4 buffers, an event rate of 5 Meps is achieved when capturing events from one board, which is 83% of the limit defined by handshake timing. USB2.0 high speed mode has a data rate of 480 megabits per second (Mbps). Because each event consists of four bytes, 5 Meps is equivalent to 160 Mbps. Monitoring with 2 or 3 interfaces simultaneously on one computer using the same USB host controller achieves a total rate of 6 Meps=192 Mbps, about half the basic data bandwidth of USB2.0.

## V. Conclusion

The main achievement of this work is the development of a simple, cheap, user-friendly USB-AER interface device for monitoring and sequencing.

TABLE I
Device Specifications

| | |
|---|---|
| Interface | 16 bit word-parallel AER |
| Protocol | 4 phase handshake |
| Host interface | USB2.0 High speed<br>128-event transactions<br>16 bit address, 16 bit timestamp |
| Timestamp | 1 $\mu s$ / 33 $ns$ timestamp<br>(switchable) |
| Cost | <$100 |
| Min. packet rate | 100 Hz |
| Power consumption | 60 $mA$ USB 5V while<br>monitoring and sequencing |
| Bandwidth | peak: 6 Meps monitor<br>3.75 Meps sequencer<br>sustained: 5 Meps monitor |

The board was successfully used to monitor and log 40000 neurons spread over four different AER chips [7] and is presently in regular use in three different labs. The complete design of this board along with software (excluding device driver) will be open-sourced [9].

## Acknowledgment

The authors thank J. Arthur and J. Wittig for initial guidance in the use of the Cypress USB interface, A.M. Whatley for the feedback on this paper, and M. Gutiérrez and Á. Jiménez for their help during the design of the PCB and the assembling of the prototypes.

## References

[1] R. Berner, "Highspeed USB2.0 AER Interfaces," Master's thesis, Swiss Federal Institute of Technology, Zurich, Switzerland and University of Seville, Spain, 2006.

[2] V. Dante, P. D. Giudice, and A. M. Whatley, "PCI-AER – Hardware and Software for Interfacing to Address-Event Based Neuromorphic Systems," *The Neuromorphic Engineer*, pp. 5–6, Mar. 2005.

[3] F. Gomez-Rodriguez, R. Paz, A. Linares-Barranco, M. Rivas, L. Miro, S. Vicente, G. Jimenez, and A. Civit, "AER tools for communications and debugging," in *IEEE International Symposium on Circuits and Systems*, 2006, pp. 3253–3256.

[4] R. Paz-Vicente, A. Linares-Barranco, D. Cascado, M. Rodriguez, G. Jimenez, A. Civit, and J. Sevillano, "PCI-AER interface for neuro-inspired spiking systems," in *IEEE International Symposium on Circuits and Systems*, 2006, pp. 3161–3164.

[5] P. Merolla, J. Arthur, and J. Wittig, "The USB Revolution," *The Neuromorphic Engineer*, pp. 10–11, Dec. 2005.

[6] CAVIAR project. [Online]. Available: http://www.imse.cnm.es/caviar

[7] R. Serrano-Gotarredona *et al.*, "AER Building Blocks for Multi-Layer Multi-Chip Neuromorphic Vision Systems," in *Advances in Neural Information Processing Systems 18*, 2005, pp. 1217–1224.

[8] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120dB 30mW Asynchronous Vision Sensor that Responds to Relative Intensity Change," *2006 IEEE ISSCC Digest of Technical Papers*, pp. 508–509, 2006.

[9] AER tools. [Online]. Available: http://avlsi.ini.unizh.ch/aer