Neuro-Inspired Real-Time USB & PCI to AER Interfaces for Vision Processing

A. Linares-Barranco, R. Paz, A. Jiménez-Fernandez, C.D. Luján, M. Rivas, J.L. Sevillano, G. Jiménez, A. Civit.
ETSI Informática. Dpto. ATC. University of Seville.
http://www.atc.us.es

Abstract

Address-Event-Representation (AER) is an emergent neuromorphic interchip communication protocol that allows for real-time virtual massive connectivity between huge number neurons located on different chips. By exploiting high speed digital communication circuits (with nano-seconds timings), synaptic neural connections can be time multiplexed, while neural activity signals (with mili-seconds timings) are sampled at low frequencies. When building multi-chip muti-layered AER systems it is absolutely necessary to have a computer interface that allows (a) to read AER interchip traffic into the computer and visualize it on screen, and (b) convert conventional frame-based video stream in the computer into AER and inject it at some point of the AER structure. This is necessary for test and debugging of complex AER systems.

This paper describes a set of PC interfaces to neuro-inspired systems, analyses the performance and power consumption. The interfaces use PCI or USB bus connections that have been developed under an EU project, where they have been tested in a stressed situation.

Keywords: AER, Neuro-Inspired, USB, PCI, FPGA, VHDL, Codesign.

INTRODUCTION

Computers have evolved in a vertiginous way in the last years, increasing the parallelism, increasing the clock frequency, increasing the performance, reducing the power consumption, reducing the volume, … But today, there isn't any hardware comparable to the most powerful 'computer' in biology, the human brain, with millions of relative slow components (neurons) working together in parallel, with a total power consumption of 20W per day in average [1]. For vision processing, the human brain is much more powerful, smaller and with very low power consumption, compared with any computer.

Primate brains are structured in layers of neurons, in which the neurons in a layer connect to a very large number (~10^4) of neurons in the following layer [2]. Many times the connectivity includes paths between non-consecutive layers, and even feedback connections are present. Artificial bio-inspired software models based on such connectivity models have overwhelmed the specialized literature presenting many ways of performing bio-inspired processing systems that outperform more conventionally engineered machines [3][4]. Since these models are software based, they operate at extremely low speeds, because of the massive connectivity they emulate. For real-time solutions direct hardware implementations are required. However, hardware engineers face a very strong barrier when trying to mimic the bio-inspired hierarchically layered structure: the massive connectivity. In present day state-of-the-art very large scale integrated (VLSI) circuit technologies it is plausible to fabricate on a single chip many thousands (even millions) of artificial neurons or simple processing cells. However, it is not viable to connect physically each of them to even a few hundreds of other neurons. The problem is greater for multi-chip multi-layer hierarchically structured bio-inspired systems. AER is an incipient bio-inspired spike-based technique capable of providing a hardware solution to the inter-chip massive connectivity problem.

AER was proposed in 1991 by Sivilotti [5] for transferring the state of an array of analog time dependent values from one chip to another. It uses mixed analog and digital principles and exploits pulse density modulation for coding information. Fig. 1 explains the principle behind the AER basics. The Emitter chip contains an array of cells (like, for example, a camera or artificial retina chip) where each pixel shows a continuously varying time dependent state that changes with a slow time constant (in the order of *ms*). Each cell or pixel includes a local oscillator (VCO) that generates digital pulses of minimum width (a few nano-secconds). The density of pulses is proportional to the state of the pixel (or pixel intensity). Each time a pixel generates a pulse (which is called "Event"), it communicates to the array periphery and a digital word representing a code or address for that pixel is placed on the external inter-chip digital bus (the AER bus). Additional handshaking lines (Acknowledge and Request) are also used for completing the asynchronous communication. The inter-chip AER bus operates at the maximum possible speed because of its asynchronous nature. In the receiver chip the pulses are directed to the pixels whose code or address was on the bus. Thus, pixels with the same code or address in the emitter and receiver chips will "see" the same pulse stream. The receiver pixel integrates the pulses and reconstructs the original low

frequency continuous-time waveform. Pixels that are more active access the bus more frequently than those less active.

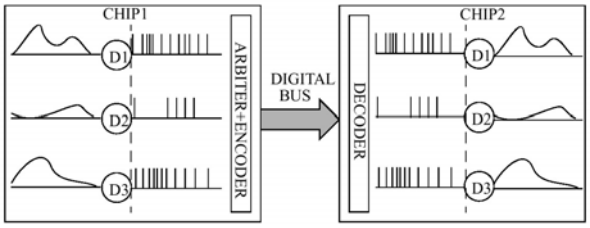

Fig. 1: Illustration of AER inter-chip communication scheme.

Transmitting the pixel addresses allows performing extra operations on the images while they travel from one chip to another. For example, inserting properly coded EEPROMs allows shifting and rotation of images. Also, the image transmitted by one chip can be received by many receiver chips in parallel, by properly handling the asynchronous communication protocol. The peculiar nature of the AER protocol also allows for very efficient convolution operations within a receiver chip [6].

There is a growing community of AER protocol users for bio-inspired applications in vision and audition systems, as demonstrated by the success in the last years of the AER group at the Neuromorphic Engineering Workshop series [7]. The goal of this community is to build large multi-chip and multi-layer hierarchically structured systems capable of performing complicated array data processing in real time. The success of such systems will strongly depend on the availability of robust and efficient development and debugging AER-tools. One such tool is a computer interface that allows not only reading an AER stream into a computer and displaying it on its screen in real-time, but also the opposite: from images available in the computer's memory, generate a synthetic AER stream in a similar manner as would do a dedicated VLSI AER emitter chip [8][9][10].

The CAVIAR EU project had the objective to demonstrate this technology by targeting and following a moving ball. The planned AER system under CAVIAR used the following AER chips: one Retina, four Convolutions, one Winner-Take-All (Object) and one Learning chip. To make possible the right communication of these chips and for debugging purposes it is essential to have a set of instruments that would allow to:

- Sequence: Produce synthetic AER event streams that can be used as controlled inputs while testing and adjusting a chip or set of chips.
- Monitor: Observe the output of any element in the system.
- Map: Alter the stream produced by an emitter and send the modified stream to a receiver

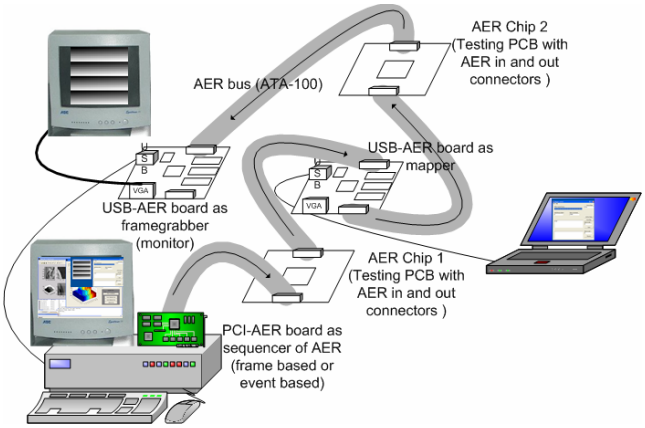

Fig. 2: AER tools usage scenario

For these purposes we have designed and implemented two different instruments: a PCI board capable of sequencing and monitoring events at a peak rate of 15Mevents/s, an equivalent USB2.0 (high speed) board with a peak rate of 5Mevents/s, and a versatile USB2.0 (full speed) board that can be used for sequencing, monitoring and mapping. This last board can be used either in a stand alone mode or connected to an external computer (through a USB bus). A possible scenario for these tools is shown in Fig. 2 where a computer with a PCI-AER board produces output for AER chip1 input. The output from this chip is remapped by a USB-AER board and fetched to AER chip 2. The output stream produced by chip 2 is monitored by another USB-AER board which can send its output directly to a VGA monitor or to a computer through USB bus.

To be useful for debugging an AER tool should be able to receive and also send a long sequence of events interfering as little as possible with the system under test.

As neurons have the information coded in the frequency (or timing) of their spikes, the pixels that transmit their address through an AER bus also have their information coded in the frequency of appearance of those addresses in the bus. Therefore, inter-spike-intervals (ISIs) are critical for this communication mechanism. Thus, a well designed tool shouldn't modify the ISIs of the AER.

AER TOOLS

1. PCI-AER Interface

Before the development of our tools the only available PCI-AER interface board was developed by Dante at ISS-Rome (See [7]-2001). This board is very interesting as it embeds all the requirements mentioned above: AER generation, remapping and monitoring. Anyhow its performance is limited to 1Mevent/s approximately. In realistic experiments software overheads reduce this value even further. In many cases these values are acceptable but, currently many address event chips can produce (or accept) much higher spike rates.

As the Computer interfacing elements are mainly a monitoring and testing feature in many address event systems, the instruments used for these purposes should not delay the neuromorphic chips in the system. Thus, speed requirements are at least 10 times higher than those of the original PCI-AER board. Several alternatives are possible to meet these goals: extended PCI buses, bus mastering or hardware based Frame to AER and AER to Frame conversion.

The previously available PCI-AER board uses polled I/O to transfer data to and from the board. This is possibly the main limiting factor on its performance. To improve the performance PCI bus mastering is the only alternative. The hardware and driver architecture of a bus mastering capable board is significantly different, and more complex, than a polling or interrupt based implementation.

The theoretical maximum PCI32/33 bandwidth is around 133Mbytes/s. This would allow for approximately 44Mevent/s considering 2 bytes per address and two bytes for timing information. Realistic figures in practice are closer to 20Mbyte/s. Thus, in those cases where the required throughput is higher a possible solution is to transmit the received information by hardware based conversion to/from a frame based representation. Although this solution is adequate in many cases, there are circumstances where the developers want to know precisely the timing of each event, thus both alternatives should be preserved.

The physical implementation of all the steps is equal. They differ in the VHDL FPGA code and in the operating system dependent driver. The design is a SPARTAN-II based board. The Spartan Version of the board is shown in Fig. 3. All the functionality supported by the interface is implemented in the FPGA. Thus the VHDL implements the PCI-bridge, the AER interface and all the states-machines and FIFOs used to adapt both buses. These state machines were designed to guarantee the time between events (ISI).

Currently a Windows driver that implements bus mastering is working and the performance is shown in this paper. The Linux version with bus mastering is still under development. An API that is compatible, as much as permitted by the different functionality, with that used in the current PCI-AER board has been implemented. MEX files to control the board from MATLAB have also been developed.

Current performance of PCI-AER board is around 15 Mevents/second using PCI mastering capabilities.

Fig. 3: CAVIAR PCI-AER board

2. USB2AER

As PCI bus is almost obsolete, and this kind of interface requires the use of a host computer instead of laptops, a USB version was developed during the project.

This board is a high-speed bus-powered USB co-design platform based on Cypress USB2.0 microcontroller and Xilinx CPLD 95xx family. The interface is able to monitor synchronous AER traffic and the opposite, sequence AER traffic. For our CAVIAR project [11], [14] 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 USB2AER board was evolved from the monitoring capability which was developed for [12] 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.

3. USB-AER

The CAVIAR PCI-AER interface and the USB2AER can perform Address Event sequencing and monitoring functions but has no hardware mapping capabilities. Although software based mapping is feasible a specific device for this purpose is needed if we want to build AER systems that can operate without requiring any standard computer. This standalone operating mode requires to be able to load the FPGA and the mapping RAM from some type of non volatile storage that can be easily modified by the users. MMC/SD cards used in digital cameras are a very attractive possibility. However in the development stage the users prefer to load the board directly from a computer and, for this purpose USB seems the most suitable solution.

Many AER researchers would like to demonstrate their systems using instruments that could be easily interfaced to a laptop computer. This requirement can also be supported with the USB-AER board as it includes a relatively large FPGA that can be loaded from MMC/SD or USB, a large SRAM bank and two AER ports. Thus the board can be used also as a sequencer or a monitor. Due to the bandwidth limitations of full speed USB (12Mbit/s) hardware based event to frame conversion is essential in this board for high, or even moderate, event rates.

The USB-AER board is based around a Spartan-II 200 Xilinx FPGA, with a 512K*32 12ns SRAM memory bank. The board uses a Silicon Laboratories C8051F320 microcontroller to implement the USB and the MMC/SD interface. A simple VGA display interface is also provided to allow the board to act as a monitor (frame grabber).

The board will act as a different device according to the module that is loaded in the FPGA either through a MMC/SD card or from the USB bus. Currently the following **modes** are implemented:

- Mapper: 1 event to N events, where N can be from 0 to 8 and each possible output event can be transmitted or not depending on a probabilistic function.
- Monitor (frame-grabber): using either USB or VGA as output. For the VGA output there are two possibilities: B/W VGA, using the VGA connector of the board. And Gray VGA, using a VGA-DAC board connected to the out-AER connector of the board. In any case, a set of events are integrated in the FPGA into a frame and this frame is represented. The integration time can be configured through USB commands.
- Sequencer: based on hardware frame to AER conversion using the Random or Exhaustive methods [15], [16]. Can produce up to 25 Mevents/second. (40 ns per event).
- Datalogger: allows capturing sequences of up to 512K events with timestamps and sending them to the PC through USB bus. No real-time is supported.
- Player (under development): to play up to 512Kevents with their timestamps. No real-time is supported.

These two last modules are very interesting when a researcher wants to use the output stream produced by a chip from another researcher (probably in other country) as input to his or her chip.

These interfaces has been connected to many AER chips, like in Telluride 2004 [7] to CAVIAR retina and to an imager developed at JHU; in Telluride 2006, with an AER cochlea; during CAVIAR project with a retina, four convolution chips, a WTA filter chip, a learning chip. The USB-AER board is shown in Fig. 4.

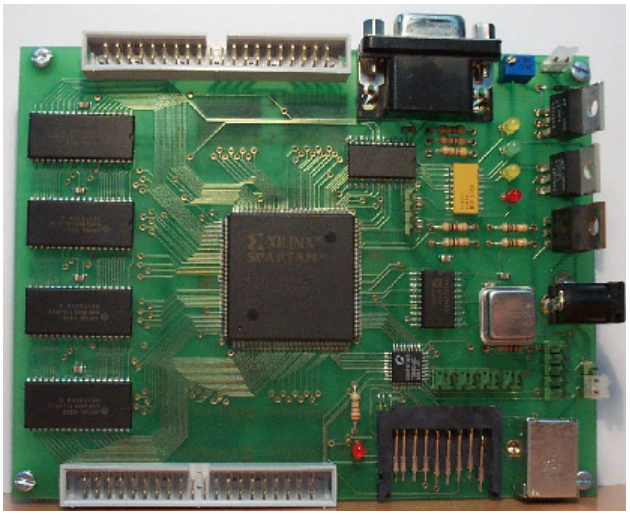

Fig. 4: USB-AER Board.

A visual interface to control this board is available under Windows XP. It allows loading modules into the FPGA, uploading or downloading data to the FPGA, and showing the received images when the board acts as a monitor. There is also available a MATLAB interface that support the same functionality. A Linux driver for the USB-AER has been developed at INI. With this driver the USB-AER board can be easily integrated with several MATLAB applications developed at INI [22].

4. SWITCH-AER

A 4 to 1 AER bus Merger and a 1 to 4 AER Splitter was a fundamental tool in order to configure the AER vision chain. These two functionalities are integrated in the same interface, the Switch-AER. This board allows:

- The connection of more complex AER systems.
- An easier debugging by inserting PCI-AER or USB-AER board without modifying the structure of the global system to be tested.

A CPLD acts as a communication centre, which manages the different modes and controls asynchronously the protocol lines. It can work in two modes: 4 inputs - 1 output or 1 input - 4 outputs, both in unicast mode (selecting one output) or broadcast mode. This functionality should be configured by jumpers. There are 5 different AER ports, where one of them works always as an output, and another as an input. The other three are bidirectional. Fig. 5 shows the current version of this board.

This interface doesn't need a PC link to be managed. Once the CPLD is programmed, the configuration is managed only by jumpers. The clock is integrated in the board, but it is not used by the CPLD, which VHDL is completely asynchronous.

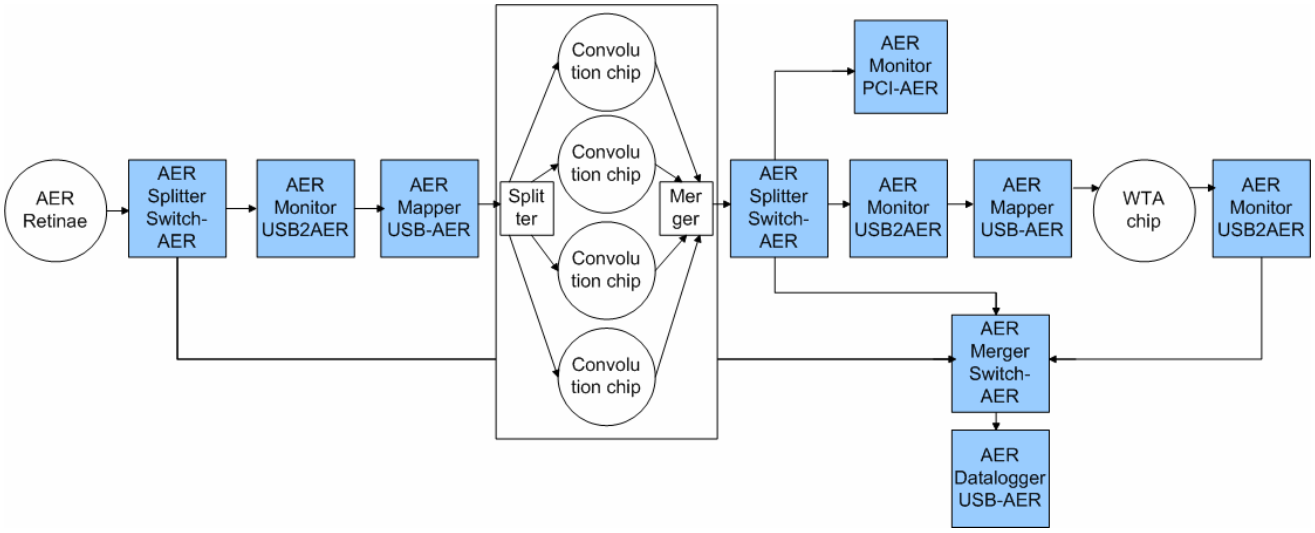

Fig. 6: CAVIAR Scenario. Blue boxes are the AER-Tools and white boxes are the AER chips.

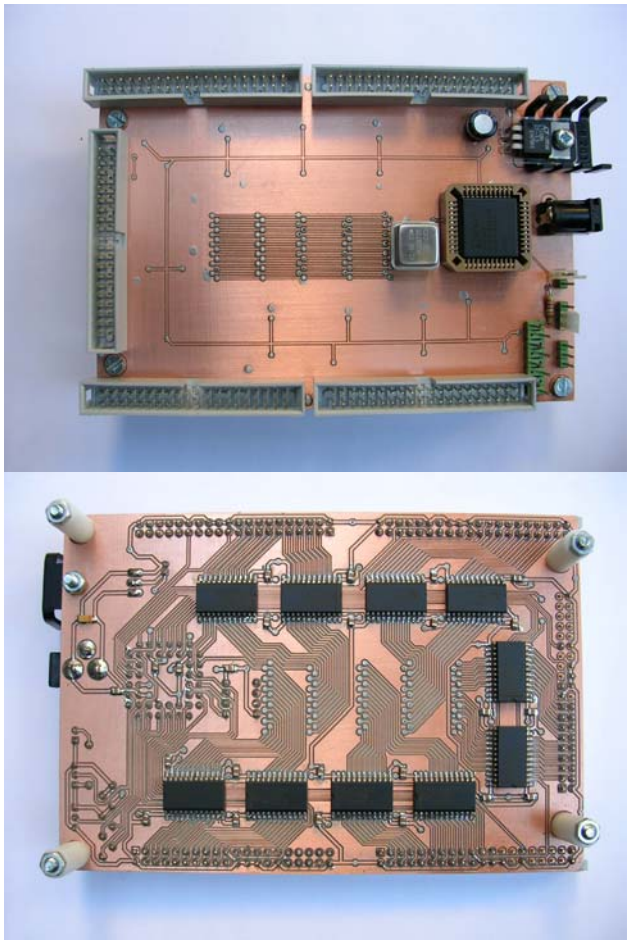

Fig. 5: AER-Switch Board.

CAVIAR SYSTEM SCENARIO

Complex systems developed by Neuromorphic Engineers require interfaces to interconnect them and to connect them to PCs for debugging and/or other purposes. This milestone was the start point for the development of a set of AER Tools under the European Project CAVIAR. We were four different partners working together in the design of a neuromorphic vision system totally based on AER. CAVIAR has connected the biggest AER chain at the moment [14]. This chain is composed by a 128x128 retina that spikes with temporal and contrast changes [17], four 32x32 convolution chips joined to detect a ball in a 64x64 space [18], a 32x32 object chip to filter the convolution activity [19] and a learning stage composed by two chips: delay line and learning to catalogue the different trajectories [20]. To make all this vision system possible, a set of AER-Tools for debugging and interconnection [13] purposes are not only useful, but also necessary.

Fig. 6 shows the AER system mounted under the CAVIAR project. In this chain the blue box are the AER-tools:

- USB-AER Mapper to implement transformations of the events during the transmission time, what allows simple operations like translations, rotations, address map shifting, compressing the image space (from 128x128 to 64x64 joining the AER traffic of 4 neighbour pixels into one of the new address space), increasing the AER traffic (from convolution to WTA to make the filter faster). This interface is based on the USB-AER tool. A special VHDL for mapping purposes is downloaded in the FPGA. This board implies a delay in the AER chain from input to output of 80ns, and a USB data transfer of 150Kbps for mapping tables uploading.
- USB-AER Framegrabber for transforming the events traffic into a sequence of frames. The USB-AER FPGA is configured with a different VHDL that integrates events into an internal 64x64 block-RAM and transfers the bitmaps to the PC through the USB2.0 full-speed interface. This tool allows

monitoring the behaviour of the chain without interfering in the performance or the AER bus traffic of the chain, thanks to the pass-through AER port. The PC or laptop used to monitor is provided with a dedicated XP driver and C++ application that can configure the interface to set up the hardware integration time. The performance of the interface is 25Mevps at the input and at the output. The software application uses around 10MIPS to monitor a frame every 100 ms with a 100 ms hardware integration time or lower. The maximum speed is 10 ms of hardware integration. But the software application depends on the USB speed to reach the maximum speed (around 50 ms per frame).

- AER Merger & Splitter. This board has been used to send the same stimulus to several convolution chips in parallel for implementing several filters at the same time (for example for detecting several ball radius), and to merge all the outputs of the convolution chips into one AER bus.
- PCI-AER Monitor. This interface allows capturing the AER traffic directly to the PC RAM memory using PCI mastering at a maximum sustained rate of 10Mevps.

PERFORMANCE

Several performance criteria have been analyzed:

- AER bus throughput: table 1 shows the maximum and the averaged throughput of each interface. The PCI-AER interface bandwidth depends on the PCI bridge throughput. The maximum is obtained while the events FIFOs of the FPGA are not saturated.
  The USB2AER bandwidth also depends on the PC USB bandwidth.
  These results have been obtained using only one interface board with an Intel PIV 3GHz PC.
  For the USB-AER and for Switch-AER interfaces, the PC bandwidth does not interfere to the AER bandwidth.

AER-Tool.	Peak	Sustained
PCI-AER. Sequencer	25 Mev/s	8 Mev/s
PCI-AER. Monitor	25 Mev/s	10 Mev/s
USB-AER. Generator	25 Mev/s	25 Mev/s
USB-AER. Mapper	12 Mev/s	12 Mev/s
USB-AER. FrameGrabber	25 Mev/s	25 Mev/s
USB-AER. Datalogger	12 Mev/s	12 Mev/s
USB2AER. Monitor	5 Mev/s	4’5 Mev/s
USB2AER. Sequencer	3,75 Mev/s	2,5 Mev/s
USB2AER. Passthrough	10 Mev/s	10 Mev/s
Switch-AER. Merger	100 Mev/s (1 to 1)	25 Mev/s (4 inputs)
Switch-AER Splitter	100 Mev/s (1 to 1)	25 Mev/s (4 outputs)

Table 1: AER bus bandwidth supported by interfaces.

- PC interface throughput
  Table 2 shows the AER-Tools bandwidth regarding to the PC connection. Each interface is limited to the maximum bandwidth of the bus used. The peak and average bandwidth differs thanks to the internal FIFOs of each interface that allows small burst transfers at high speed.
  The Switch board has no PC bandwidth since no PC connection is provided by the Switch-AER.

AER-Tool.	Max.	Peak	Sustained
PCI-AER. Sequencer	132Mb/s	40 Mb/s	32 Mb/s
PCI-AER. Monitor	132Mb/s	48 Mb/s	40 Mb/s
USB-AER.	1,5Mb/s	750 Kb/s	750 Kb/s
USB2AER. Monitor	50Mb/s	20 Mb/s	16 Mb/s
USB2AER. Sequencer	50Mb/s	15 Mb/s	10 Mb/s
USB2AER. Passthrough	50Mb/s	0 b/s	0 b/s
Switch-AER. Merger	0 b/s	0 b/s	0 b/s
Switch-AER Splitter	0 b/s	0 b/s	0 b/s

Table 2: PC to AER-interface bandwidth

- Power consumption. In normal working operations, the power consumption of the boards is shown in table 3.
  PCI and USB boards have higher power consumption because they are based on FPGA, whose power is higher than for CPLD.

AER-Tool.	Board consumption
PCI-AER.	~100 mA
USB-AER.	220 mA
USB2AER.	60 mA
Switch-AER.	40 mA

Table 3: Power consumption of AER-Tools

- Software performance.
  Using an Intel PIV 3GHz PC with 1 Gb RAM, the software C++ and JAVA applications consume a

percentage of CPU time, an amount of RAM memory and they need an I/O bandwidth when the interfaces are working in a stress situation. Table 4 shows these results.

AER-Tool Sw.	% CPU	Memory Usage	E/S access rate
PCI-AER.	5	~5 Mb	~40 Mbps
USB-AER.	2	~10 Mb	~40 Kbps
USB2AER.	25	~ 85 Mb	~18 Mbps
Switch-AER.	N/A	N/A	N/A

Table 4: PC performance of the AER-Tools.

Table 4 has been obtained reading the corresponding columns of the Task Manager of the Microsoft XP OS while each interface is working in a stress mode:

- The PCI-AER working as a timestamped event monitor through the C++ application which integrates the received events to represent a bitmap in the PC screen.
- The USB-AER working as an AER monitor, integrating the events in the FPGA and transmitting the bitmaps through USB to the C++ application, which simply represents it on the PC screen.
- The USB2AER working also as a timestamped event monitor through high-speed USB. The JAVA application represents the reconstructed bitmap on the PC screen.

The JAVA software is used only by the USB2AER interface. This interface, as commented before, transmits to the PC address-events with timestamp information and it gets a bandwidth that allows up to 18 Mbps (~4.5 Mevps), as demonstrated the 'task manager' I/O bytes transmission column. The PCI-AER interface transmits the same data format with a higher bandwidth of 40 Mbps (~10 Meps). In contrast, the USB-AER interface transmits to the PC a hardware processed bitmap collecting events for a period of time. The 40 Kbps transmitted allows maintaining a transmission of 10 fps (4 Kb per 64x64 8 bit pixels frame).

Regarding to PC resources consumption, the C++ applications are clearly more efficient than the JAVA application, but they are dependent of the OS. C++ consume less %CPU and less RAM, but JAVA application needs x5 more %CPU and x8 more RAM respect to the worst case of each C++ specific application.

CONCLUSIONS

Several AER to PC interfaces have been presented and analyzed regarding to AER communication and PC performance. With these results we can conclude that the tool to be used depends on the requirements of each scenario and the purpose of the interface.

These interfaces allow stimulating and monitoring an AER system under development or testing in order to debug it properly, but they also allow capturing long sequences of events to process them off-line in order to characterize the behavior of the tested AER system.

FUTURE IMPROVEMENTS

The maximum PC to interface bandwidth is obtained through the PCI bus. Since the PCI-Express is the new serial version of the PCI that allows faster transfers and reduced interfaces, the solution seems to be to transform parallel buses into serial ones.

In this way, some advances have been done by authors, implementing parallel to serial converters for AER buses and a new prototype interface for LVDS based serial AER buses [21]. The first experiments, using commercial LVDS transceivers, have reached a peak rate of 1.66 Gbps, with 16bits transceivers, what implies an event rate of 100Mev/s.

ACKNOWLEDGMENTS

This work has been supported by the following projects: Spanish Science And Education Ministry Research Projects TEC2006-11730-C03-02 (SAMANTA 2) and TIN2006-15617-C03-03 (AmbienNet), Andalussian Council grant P06-TIC-01417 (BrainSystem) and EU grant IST-2001-34124 (CAVIAR).

Reference List

[1] Drubach, Daniel. The Brain Explained. New Jersey: Prentice-Hall, 2000.

[2] G. M. Shepherd, *The Synaptic Organization of the Brain*, Oxford University Press, 3rd Edition, 1990.

[3] J. Lee, "A Simple Speckle Smoothing Algorithm for Synthetic Aperture Radar Images," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-13, pp. 85-89, 1983.

[4] T. Crimmins, "Geometric Filter for Speckle Reduction," *Applied Optics*, vol. 24, pp. 1438-1443, 1985.

[5] M. Sivilotti, Wiring Considerations in analog VLSI Systems with Application to Field-Programmable Networks, Ph.D. Thesis, California Institute of Technology, Pasadena CA, 1991.

[6] Teresa Serrano-Gotarredona, Andreas G. Andreou, Bernabé Linares-Barranco. AER Image Filtering Architecture for Vision-Processing Systems. IEEE Transactions on Circuits and Systems. Fundamental Theory and Applications, Vol. 46, N0. 9, September 1999.

[7] A. Cohen, R. Douglas, C. Koch, T. Sejnowski, S. Shamma, T. Horiuchi, and G. Indiveri, Report to the National Science Foundation: Workshop on Neuromorphic Engineering, Telluride, Colorado, USA, 2001,2004, 2006. [http://www.ini.unizh.ch/telluride] [http://www.ine-web.org]

[8] Kwabena A. Boahen. Communicating Neuronal Ensembles between Neuromorphic Chips. Neuromorphic Systems. Kluwer Academic Publishers, Boston 1998.

[9] A. Mortara, E.A. Vittoz and P. Venier. A Communication Scheme for Analog VLSI Perceptive Systems, IEEE Journal of Solid-State Circuits. Vol. 30, No. 6, pp. 660-669, 1995.

[10] Misha Mahowald. VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function. PhD. Thesis, California Institute of Technology Pasadena, California, 1992.

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

[12] 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.
[13] R. Paz, F. Gomez-Rodriguez, M. A. Rodriguez, A. Linares-Barranco, G. Jimenez, A. Civit. Test Infrastructure for Address-Event-Representation Communications. IWANN 2005. LNCS 3512. pp 518-526. Springer Verlag.
[14] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, H. Kolle Riis, T. Delbrück, S. C. Liu, P. Häfliger, G. Jimenez-Moreno, A. Civit, T. Serrano-Gotarredona, A. Acosta-Jiménez, B. Linares-Barranco, AER Building Blocks for Multi-Layer Multi-Chip Neuromorphic Vision Systems, NIPS’05, Vancouver, December-2005.
[15] A. Linares-Barranco, G. Jimenez-Moreno, A. Civit-Ballcels, and B. Linares-Barranco. “On Algorithmic Rate-Coded AER Generation”. IEEE Transaction on Neural Networks. May-2006.
[16] A. Linares-Barranco, M. Oster, D. Cascado, G. Jimenez, A. Civit, B. Linares-Barranco. “Inter-spike-intervals analysis of AER Poisson-like generator hardware”. Neurocomputing. Vol. 70, Nos 16-18. pps 2692-2700.
[17] P. Lichtsteiner, T. Delbruck, 64x64 Event-Driven Logarithmic Temporal Derivative Silicon Retina, 2005 IEEEWorkshop on Charge Coupled Devices and Advanced Image Sensors, Nagano, Japan, June-2005.
[18] Rafael Serrano, Teresa Serrano, Antonio José Acosta, Bernabé Linares-Barranco, An Arbitrary Kernel Convolution Aer- Transceiver, ISCAS’06, Kos, Greece, May-2006.
[19] Oster, M., Liu, S.C., A Winner-take-all Spiking Network with Spiking Inputs, ICECS 2004, Tel Aviv, 2004.
[20] H. Kolle Riis, P. Haefliger, Spike based learning with weak multi-level static memory, ISCAS’04, vol. 5, pp. 393-395, Vancouver, Canada, May-2004.
[21] A. Jiménez-Fernández, C.D. Luján, A. Linares-Barranco, F. Gómez-Rodríguez, M. Rivas, G. Jiménez, A. Civit. “Address-Event based Platform for Bio-inspired Spiking Systems”. Proceedings of the SPIE Conference of Microelectronis for the New Millenium. Maspalomas, Gran Canaria, May-2007.
[22] M. Oster, Serverbased Software Architecture for AER systems. [http://www.ini.unizh.ch/~mao/AerSoftware/SoftwareOverview.pdf]