PCI-AER interface for Neuro-inspired Spiking Systems

R. Paz-Vicente, A. Linares-Barranco, D. Cascado, MA. Rodriguez, G. Jimenez, A. Civit, JL. Sevillano.
Dpto. de Arquitectura y Tecnología de Computadores. Universidad de Sevilla, SPAIN.
rpaz@atc.us.es

***Abstract*— Address Event Representation (AER) is a neuromorphic interchip communication protocol that allows for real-time connectivity between huge number neurons located on different chips. By exploiting high speed digital communication circuits (nano-seconds), synaptic neural connections can be time multiplexed (mili-seconds). 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, and (b) inject a sequence of events to the AER structure. This paper presents a PCI to AER interface, that dispatches a sequence of events with timing information. It is able to recovery the possible delays introduced by AER bus. It has been implemented in real time hardware using VHDL and tested in a PCI-AER board, developed by authors, that currently capable to send and receive events at a peak rate of 16 Mev/sec, and a typical rate of 10 Mev/sec.**

I. Introduction

Address-Event-Representation (AER) was proposed in 1991 by Sivilotti [1] for transferring the state of an array of neurons from one chip to another. It uses mixed analog and digital principles and exploits pulse density modulation for coding information. The state of the neurons is a continuous time varying analog signal.

Figure 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 change with a slow time constant (in the order of milliseconds). Each cell or pixel includes a local oscillator that generates digital pulses of minimum width (a few nanoseconds). The density of pulses is proportional to the state or intensity of the pixel. Each time a pixel generates a pulse (which is called "event"), it communicates with the array periphery and a digital word representing its code or address 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.

In the receiver chip the pulses are directed to the pixels or cells whose code or address was on the bus. This way, pixels with the same code or address in the emitter and receiver chips will "see" the same pulse stream. The receiver cell integrates the pulses and reconstructs the original low frequency continuous-time waveform. Pixels that are more active are accessing the bus more frequently than those less active.

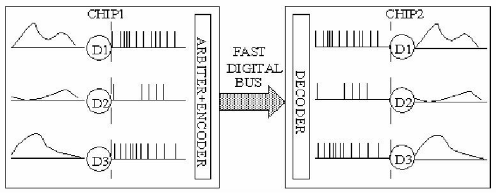

Figure 1. AER inter-chip communication scheme.

Transmitting the pixel addresses allows performing extra operations on the images while they travel from one chip to another. For example, inserting properly coded memories (ie. EEPROM) allows transformation (ie. 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 [2].

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 [3]. 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 powerful of these systems can be used under computer based systems under co processing. This purpose strongly depends on the availability of robust and efficient AER interfaces [4]. One such tool is a PCI-AER interface that allows not only reading an AER stream into a computer memory and displaying it on 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 [1][5][6].

In Section II we comment the problems behind the AER sequencing and monitoring. In Section III and IV we present a hardware architecture for the CAVIAR PCI-AER interface developed under the European project CAVIAR. In Section V we present experiment results. Finally Section VI presents the conclusions.

II. Sequencing and Monitoring AER Events

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. Let's start explaining the meaning of interfacing in the context.

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) is critical for this communication mechanism. Thus, a well designed tool shouldn't modify the ISIs of the AER.

The ISIs may be difficult to preserve depending on the nature of the emitter and/or receiver chips. Let's suppose the case of having an AER emitter chip connected to an AER receiver chip, and we want to debug their communication. In principle, there are two possibilities: connecting to the bus an AER sniffer element, or to introducing a new AER element in between the emitter and the receiver.

- The sniffer element will consist on an AER receptor that captures the address and stores it with a timestamp in memory for each request that appears on the AER bus. The problem in this case is that the speed of the emitter and receiver protocol lines could be faster than the maximum speed supported by the sniffer (15 ns per event in some existing chips), causing events to be lost. Another typical problem could be that the throughput of the AER bus (unknown in principle) would be so high that the interface memory cannot be downloaded to the computer's memory on time. This also implies that events are lost.
- The other possibility is to introduce a new AER element between the two chips. In this case the emitter sends the event to the AER element and the AER element sends the same event to the receiver chip. The problem now is that the new AER element will always introduce a delay in the protocol lines, and may also block the emitter if it is not able to keep up with its throughput. Therefore, ISIs are not conserved. But the behaviour will be the same than if we connect the emitter to a slower receiver.

The throughput problem requires using very fast PC interfaces and the problem of very fast emitter or receiver protocols can be reduced by using a very high frequency clock for the stages that interface with the AER protocols.

III. PCI-AER INTERFACE: Considerations and PCB

Before the development of our tools the only available PCI-AER interface board was developed by Dante at ISS-Rome [3]. 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 proposes 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: (a) extended PCI buses, (b) bus mastering and (c) hardware based Frame to AER and AER to Frame conversion.

When the development of the CAVIAR PCI-AER board was started, using 64bit/66MHz PCI seemed an interesting alternative as computers with this type of buses were popular in the server market. When we had to make implementation decisions the situation had altered significantly. Machines with extended PCI buses had almost disappearing and, on the other hand, serial LVDS based PCI express was emerging clearly as the future standard but almost no commercial implementations were in the market. Therefore, the most feasible solution was to stay with the common PCI implementation (32 bit bus at 33MHz).

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 increase 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.

Hardware based frame to AER conversion doesn't increase PCI throughput but, instead, it reduces PCI traffic. First some important facts have to be explained. It is well known that some AER chips, especially grey level imagers where pulse density is proportional to the received light intensity, require a very large bandwidth. This is also the case of many other chips when they are not correctly tuned. For example let's consider a Grey level 128*128 imager with 256 grey levels. In a digital frame based uncompressed 25fps format, it would require a bandwidth of 128*128*25= 0.39MBytes/s. The maximum requirements for an "equivalent" system that would output AER supposing the number of events in a frame period is equal to the gray level and considering the worst case where all pixels spike with maximum rate is:

2bytes/event*256events/pixel*number of pixels/ frame period= 200MBytes/s

The meaning of this figure should be carefully considered. A well designed AER system, which produces events only when meaningful information is available, can be very efficient but, an AER monitoring system should be prepared to support the bandwidth levels that can be found in some real systems. These include systems that have not been designed carefully or that are under adjustment. Currently the available spike rates, even in these cases, are far from the value shown above but, some current AER chips may exceed the 40Mevents/s in extreme conditions.

The theoretical maximum PCI32/33 bandwidth is around 133Mbytes/s. This would allow for approximately

33Mevent/s considering 2 bytes per address and two bytes for timing information. Realistic figures in practice are closer to 15Mevents/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.

Implementing AER to Frame conversion is a relatively simple task as it basically requires counting the events over the frame period. Producing AER from a frame representation is not trivial and several conversion methods have been proposed 0[4].

The theoretical event distribution would be that where the number of events for a specific pixel is equal to its associated grey level and those events are equally distributed in time. The normalized mean distance from the theoretical pixel position in time to the resulting pixel timing with the different methods is an important comparison criterion. In 0[4] it is shown that, in most circumstances, the behavior of the methods is similar and, thus, hardware implementation complexity is an important selection criteria. From the hardware implementation viewpoint random, exhaustive and uniform methods are especially attractive.

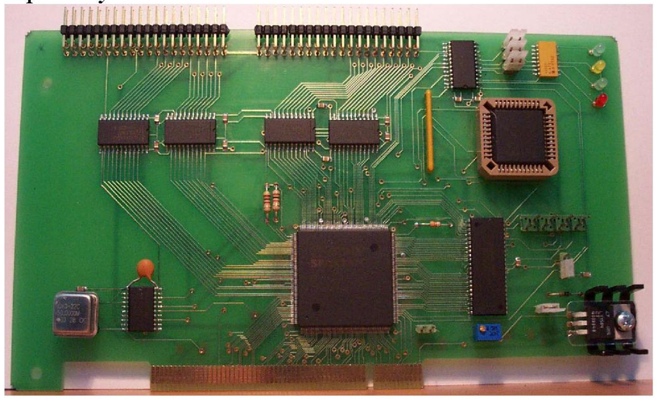

Figure 2. CAVIAR PCI-AER board

As a result of these considerations the design and implementation of the CAVIAR PCI-AER board was developed including the bus mastering. The hardware based frame to AER conversion has been developed for the CAVIAR USB-AER board [10].

The physical implementation is implemented into VHDL for a FPGA. It was established that most of the functionality, demanded by the users, could be supported by the larger devices in the less expensive SPARTAN-II family. Figure 2. shows the CAVIAR PCI-AER board.

A Windows driver and an API that implements bus mastering and a Matlab interface are currently available. A Linux version of the driver is still under development.

IV. PCI-AER INTERFACE: HARDWARE DESIGN

The final goal is to transmit an AER sequence to an AER based system (for example a convolution chip) to perform video processing. An adequate sequence of events can be generated by software for testing an AER based system. This sequence of events needs to be sent to the AER based system. For this purpose it is necessary an interface between the computer and the AER bus. Figure 3. shows the VHDL architecture of the present hardware interface. This is a PCI interface developed under the European project CAVIAR. The interface, called CAVIAR PIC-AER, has two operation modes that can work in parallel:

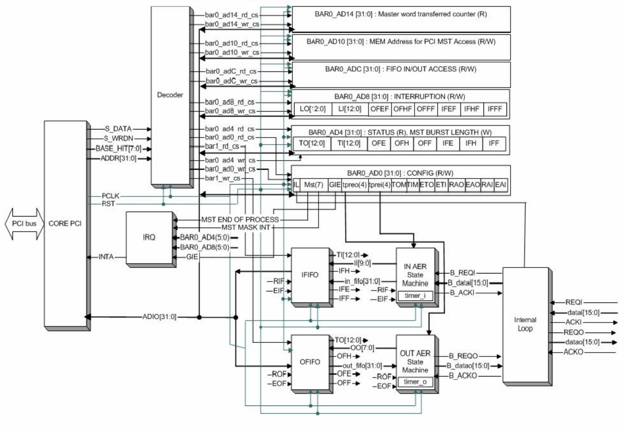

Figure 3. Hardware Interface Architecture.

A. *From PCI to AER.*

The AER-stream is stored in the computer memory and then it is sent to the AER system through the PCI bus and the OFIFO. This stream is saved in memory using 32 bits for each address event. The sixteen less significant bits represents the address of the pixel that is emitting the event. The sixteen more significant bits represent a time difference from the previous event in clock cycles. The clock cycle is 30 ns, but can be scaled up 16 times. Special words can be used in the OFIFO to make the state machine to wait the maximum time, coded with 16 bits, and then it reads a new word of the OFIFO without any event transmission. The OUT-AER state machine keeps continuously reading 32-bit words from OFIFO if it is enabled. For each word the state machine will wait for the configured number of clock cycles before transmitting the address through the AER output bus. If the acknowledge is delayed, the timer of the OUT-AER state machine will discount this time to the wait state of the next event. If the result of the discount is negative no wait will be done for the next event and this value will be used as initial wait for the following event. With this treatment the delay between events is not relative to the previous one, and a delay in the ACK reception will not cause a distortion in the time distribution of all the events along the time period.

B. *From AER to PCI.*

The AER sequence arrives to the CAVIAR PCI-AER interface through the input AER port. The AER-IN state machine stores the incoming event (16 bits LSbits) into the IFIFO with temporal information. This temporal information (16 bits MSbits) is the number of clock cycles since the last event.

The connection to the PCI bus is done by a VHDL bridge [9] that attends to the Plug & Play protocol of the PCI bus, decodes the access to the base address by the operating system, allows the bus mastering and the interruptions.

V. Experiment

The output AER bus has been connected with the input AER bus of the same board. The experiment consist on transmitting a sequence of events associated to an image using different methods for synthetic AER generation: Scan, Uniform, Random and Exhaustive[4]. It has been transmitted and received a Test Image Set (TIS) synthesized by all the methods. The TIS is composed by 9 Gaussian histogram images that imply a growing charge of events in the AER bus. Figure 4. shows the average inter-spike time difference between the expected (10 ns per event) and the transmitted/received by the interface (40 ns per event). In the worst case, the difference is 5 ms per event. The delay is growing with the charge of events because of the saturation of the states machines local time recovery algorithm. The IFIFO is almost collapsed for the 90% charge of events, so there still are some pauses between events that allow to the state machine to make some wait state. This situation affect to the error due to the reduced hoped time.

VI. CONCLUSIONS

AER format is a neuroinspired communication way between neuroinspired systems. Many efforts have been done in real-time vision processing. This paper has presented a hardware interface with time delays recovery for transferring events from a PC to an AER system.

The hardware has been tested with several charges of events produced by several methods of synthetic AER generation using the TIS.

A hardware interface that allows the communication between a PC and a AER based system, through the PCI bus, is detailed and it has been tested with a bandwidth support from *1 Mevent/second* (worst case) to *16,6 Mevent/second* (best case), using PCI-mastering capabilities.

Acknowledgment

This work was in part supported by EU grant IST-2001-34124 (CAVIAR), and spanish grant TIC-2003-08164-C03-02 (SAMANTA).

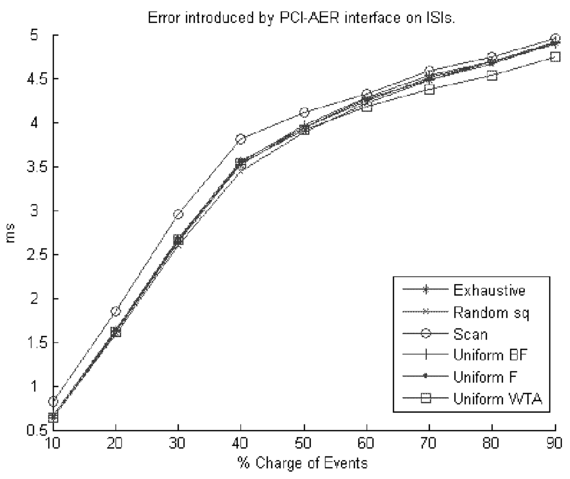

Figure 4. Average IS time difference for TIS and synthetic methods.

References

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

[2] 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.

[3] 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, June-July 2001. [www.ini.unizh.ch/telluride]

[4] A. Linares-Barranco, G. Jimenez-Moreno, A. Civit-Ballcels, and B. Linares-Barranco. "On Algorithmic Rate-Coded AER Generation". Accepted for publication on IEEE Transaction on Neural Networks.

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

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

[7] Kwabena A. Boahen. "Retinomorphic vision systems II: Communication channel design". Proceedings of the IEEE ISCAS, volume supplement, pp. 14-17. May 1996.

[8] Mortara, Eric A. Vittoz, Philippe Venier. A communication Scheme for Analog VLSI Perceptive Systems. IEEE Journal of Solid-State Circuits, vol. 30, No. 6, pp. 660-669, June 1995.

[9] R. Paz. "Análisis del bus PCI. Desarrollo de puentes basados en FPGA para placas PCI". Trabajo de investigación para obtención de suficiencia investigadora. Sevilla, Junio 2003.

[10] R. Paz, F. Gomez-Rodriguez, M. A. Rodriguez, A. Linares-Barranco, G. Jimenez, A. Civit. Test Infrastructure for Address-Event-Representation Communications. International Work-Conference on Artificial Neural Networks. Vilanova I la Gertru. SPAIN. June-2005.