Time-Recovering PCI-AER interface for Bio-inspired Spiking Systems

R. Paz-Vicente, A. Linares-Barranco, D. Cascado, S. Vicente, G. Jimenez, A. Civit.

[1]Dpto. de Arquitectura y Tecnología de Computadores, Universidad de Sevilla, Av. Reina Mercedes s/n, 41012 Sevilla, Spain. Phone: , Fax: , E-mail: rpaz@atc.us.es

[2]Instituto de Microelectrónica de Sevilla, Ed. CICA, Av. Reina Mercedes s/n, 41012 Sevilla, Spain.

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. Also, neurons generate 'events' according to their activity levels. More active neurons generate more events per unit time, and access the interchip communication channel more frequently, while neurons with low activity consume less communication bandwidth. 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) inject a sequence of events at some point of the AER structure. This is necessary for testing and debugging complex AER systems.

This paper presents a PCI to AER interface, that dispatches a sequence of events received from the PCI bus with embedded timing information to establish when each event will be delivered. A set of specialized states machines has been introduced to recovery the possible time delays introduced by the asynchronous AER bus. On the input channel, the interface capture events assigning a timestamp and delivers them through the PCI bus to MATLAB applications. It has been implemented in real time hardware using VHDL and it has been tested in a PCI-AER board, developed by authors, that includes a Spartan II 200 FPGA. The demonstration hardware is currently capable to send and receive events at a peak rate of 8,3 Mev/sec, and a typical rate of 1 Mev/sec.

1. 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.

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 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.
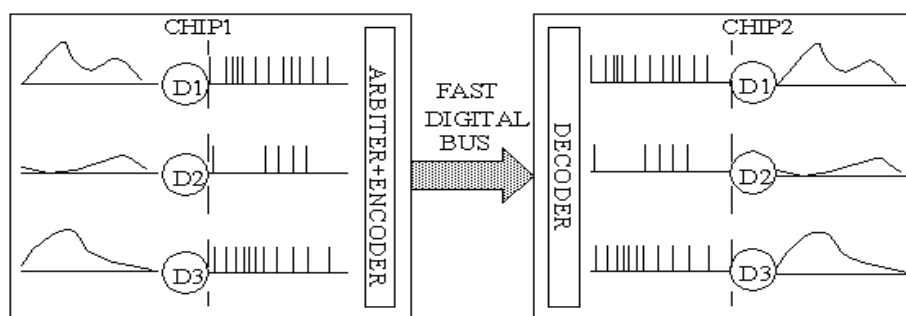

fig. 1 *AER inter-chip communication scheme.*

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.

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 depend on the availability of robust and efficient AER interfaces [4]5. 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]67.

In Section 2 we comment the problems behind the AER sequencing and monitoring. In Section 3 and 4 we present a hardware architecture for the CAVIAR PCI-AER interface developed into VHDL for European project CAVIAR. In Section 5 we present experiment results. Finally Section 6 present the conclusions.

2. 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.

3. 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 (See [2]). 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:
- Extended PCI buses.
- Bus mastering.
- 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 [3] 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) and consider PCI express for future implementations. Speed improvements, thus, should come from the alternative possibilities.

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 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.

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 [4][5].

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 [6] 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.

As a result of these considerations the design and implementation of the CAVIAR PCI-AER board was subdivided into a set of intermediate steps in which initially no mastering was implemented. Later Bus mastering was included and hardware based frame to AER conversion was included as a last step.

The physical implementation of all the steps is equal. They differ in the VHDL FPGA code and in the operating system dependant driver. The first design was a VIRTEX based board which was completely redesigned after the first tests. 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. The Spartan Version of the board is shown in fig. 2.

Currently a Windows driver that implements bus mastering is being tested. 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.
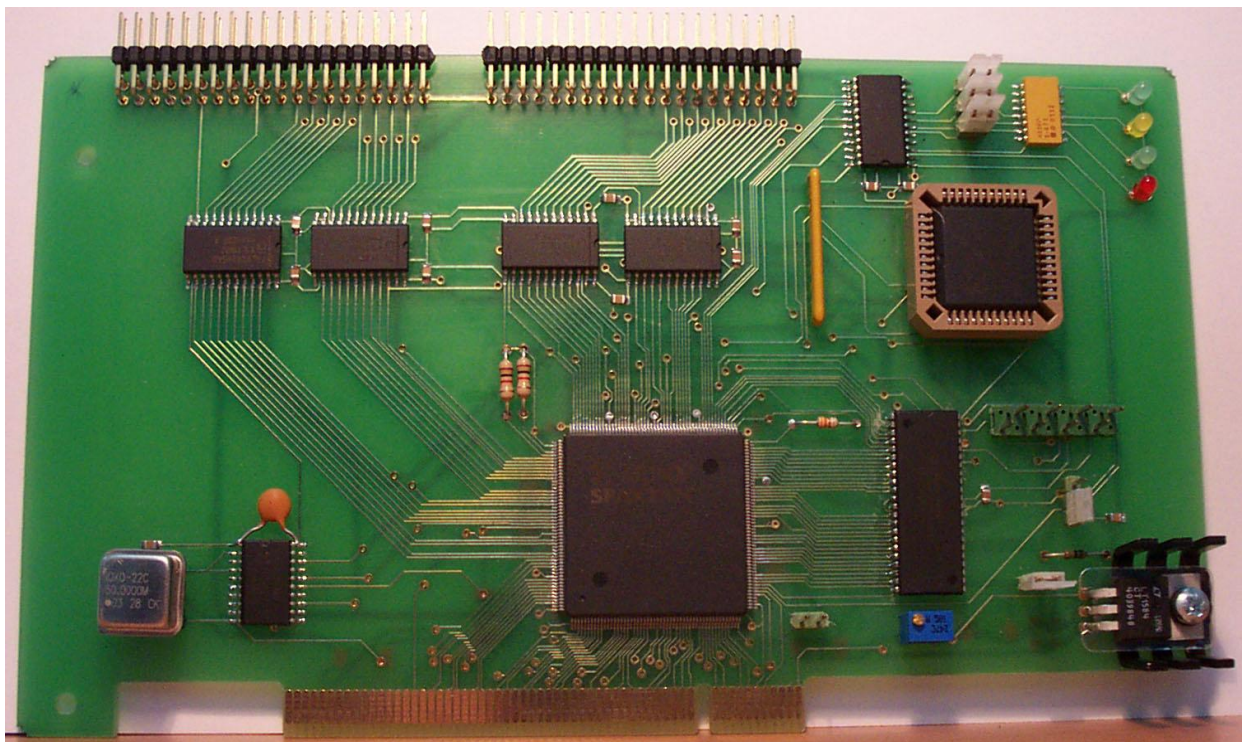

fig. 2 CAVIAR PCI-AER board

4. 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 need to be sent to the AER based system. For this purpose it is necessary an interface between the computer and the AER bus. Fig. 3 shows the architecture of the present hardware interface. This is a PCI interface developed under the European project CAVIAR. The interface, called CAVIAR PIC-AER G1, has two operation modes that can work in parallel:

4.1. From PCI to AER.

The AER-stream is stored in the computer memory and then it is sent to the AER system through 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. And the another more significant bits represent a time difference from the previous event in clock cycles. The clock cycle can be configured. The OUT-AER state machine keeps continuously reading 32-bit words from OFIFO if the ENOF signal is active. 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.

4.2. From AER to PCI.

The AER sequence arrives to the CAVIAR PCI-AER interface through the input AER port. The AER-IN state machine keeps storing the incoming data into the IFIFO. This sequence of events is stored with temporal information. Every time a new event arrives, the number of clock cycles since the last event is stored in the IFIFO in the sixteen more significant bits of the 32 bit word, and the counter of clock cycles is cleared.

Counters for both IFIFO and OFIFO can be clock divided.

The connection to the PCI bus is done by a VHDL bridge [12] that attend to the plug & play protocol of the PCI bus, decodes the access to the base address by the operating system, allows the burst access and the interruption. This interruption tries to avoid overflows at the incoming FIFO.
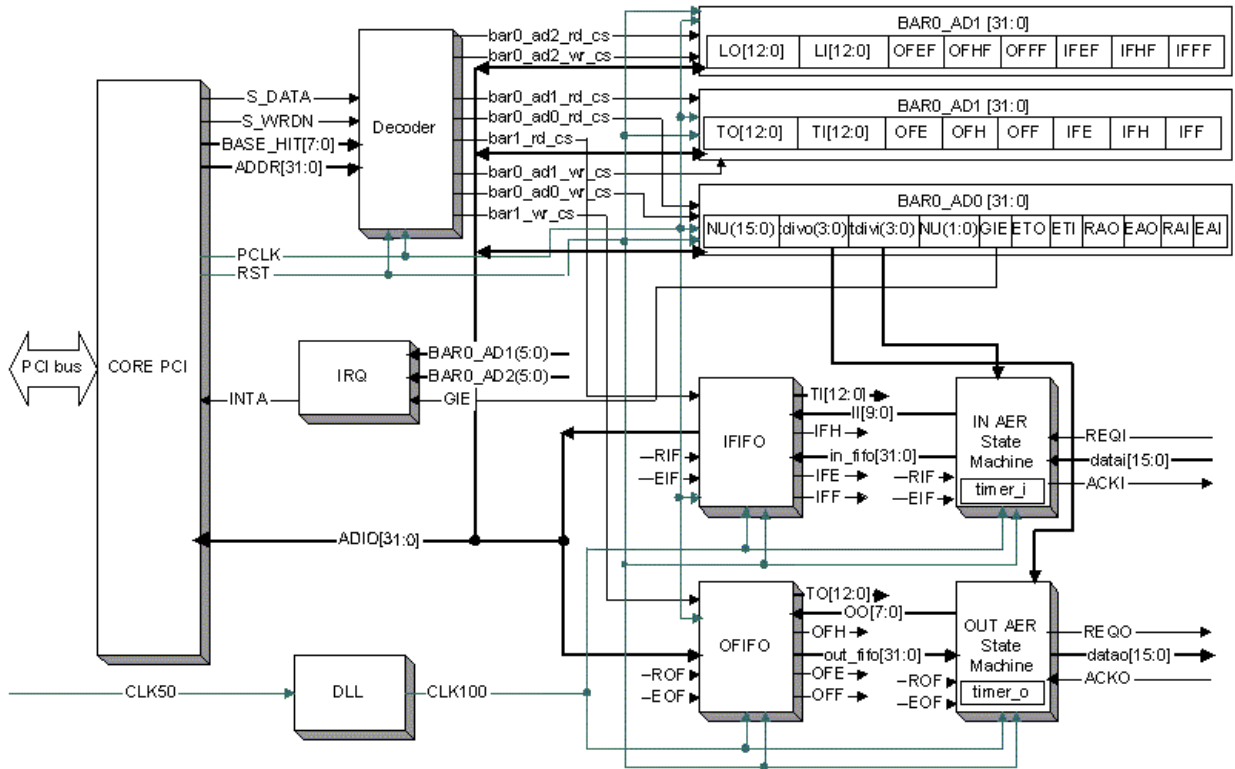
fig. 3. *Hardware Interface Architecture.*

5. EXPERIMENT

The output AER bus has been connected with the input AER bus of the same board. With this configuration the board, in burst mode is able to read or write an AER event every *Tpulsemin*=60ns. This implies the restriction that $NxMxK \leq Tframe/Tpulse$=3,3·10^5. This restriction implies that for avoiding errors in the channel the resolution of the image has to carry out with the previous condition. This restriction doesn't imply that a large image can't be transmitted without error or with a minimum error. In fact, due to the pause or wait states between events along the sequence of events associated to one image, the delay that the channel includes to the transmission can be compensated by reducing the pauses if there exist enough pauses and they are well distributed.

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][5]. Then it can be calculated the maximum and minimum time between events. The minimum one is equal to *Tpulsemin=120ns*, and this has been obtained during a burst transmission with the OFIFO full as initial condition. And the maximum one is equal to *Tpulsemax=1,14μs* and it was obtained by transmitting a sequence of events through the PCI bus with the OFIFO empty as initial condition.

The hardware can reduce the delays due to the transmission by avoiding or reducing the wait states. It has been transmitted and received a Test Image Set (TIS) synthesized by all the methods using the CAVIAR PCI-AER. TIS is composed by 9 Gaussian histogram images that implies a growing charge of events in the AER bus. Figure 5 shows the average inter-spike time difference between the expected (10 ns per event) and the transmitted/received by the interface (120 ns per event). In the worst case, the difference is 2,4 ms per event. Around the 30 % and 40 % of charge of events there exist a local maximum due to the proximity to the saturation of the input FIFO. Although the IFIFO is almost collapsed, there still are some pauses that allow to the board to make some wait state. This situation affect to the error due to the reduced hoped time.
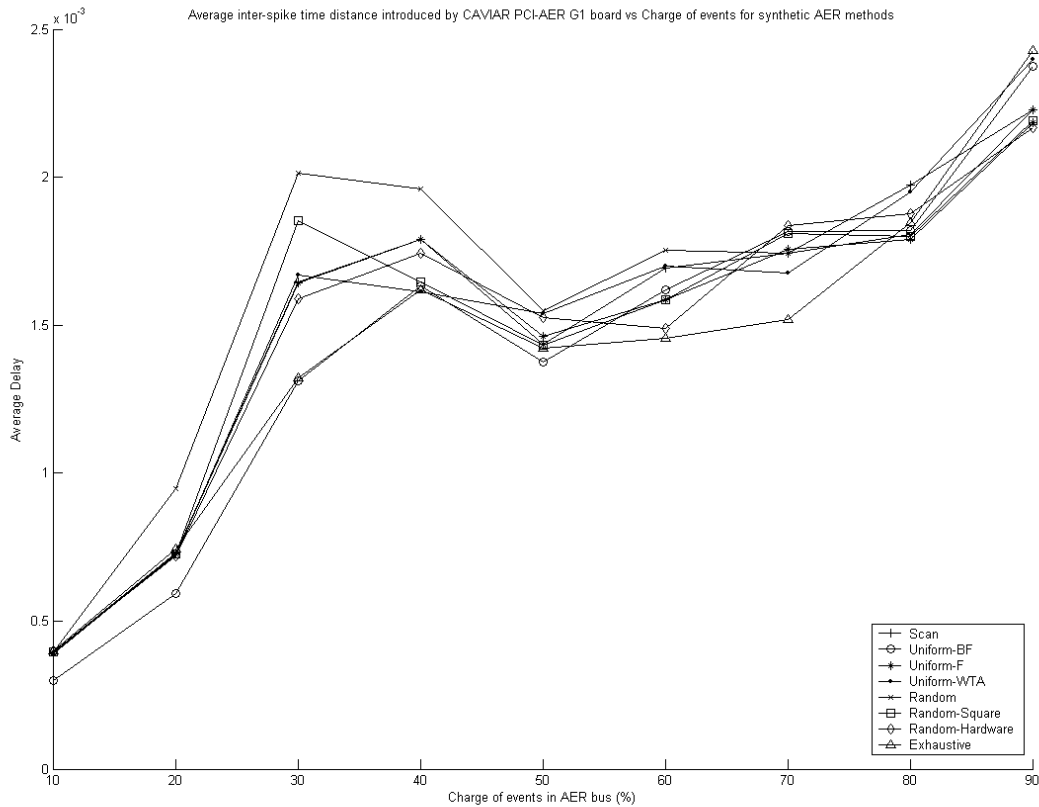
fig. 4. *Average Time Delay of CAVIAR PCI-AER for TIS and all the methods.*

6. CONCLUSIONS

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

The hardware has been tested with the TIS. This set of images carry out with the same characteristics. There exists another kind of population of images with different characteristics that will cause a different respond along the methods. For example radar images, x-ray images, ultrasound-scan images, ... Therefore every method will result more appropriate switch the population of images selected.

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

Our group has available other kind of AER interfaces, based on USB bus and MMC, that can be also used from MATLAB. Some of the functionalities supported are:
- Remapping the AER information while it is transmitted.
- Synthesizing AER format from digital video, frame by frame.
- AER data logging and reproduction.

ACKNOWLEDGEMENTS

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. "Estudio y evaluación de interfaces para la conexión de sistemas neuromórficos mediante Address- Event-Representation". Ph.D. Thesis, University of Seville, Spain, 2003
5. A. Linares-Barranco, R. Senhadji-Navarro, I. García-Vargas, F. Gómez-Rodríguez, G. Jimenez and A. Civit. "Synthetic Generation of Address-Event for Real-Time Image Processing". ETFA 2003, Lisbon, September. Proceedings, Vol. 2, pp. 462-467.
6. Kwabena A. Boahen. "Communicating Neuronal Ensembles between Neuromorphic Chips". Neuromorphic Systems. Kluwer Academic Publishers, Boston 1998.
7. Misha Mahowald. "VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function". Ph.D. Thesis. California Institute of Technology Pasadena, California 1992.
8. Kwabena A. Boahen. "Retinomorphic vision systems II: Communication channel design". Proceedings of the IEEE ISCAS, volume supplement, pp. 14-17. May 1996.
9. 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.
10. Pierre L'Ecuyer, François Panneton. A New Class of Linear Feedback Shift Register Genoerators. Proceedings of the 2000 Winter Simulation Conference.
11. Linear Feedback Shift Register V2.0. Xilinx Inc. October 4, 2001. http://www.xilinx.com/ipcenter.
12. 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.