AER Synthetic Generation in Hardware for Bio-inspired Spiking Systems

Alejandro Linares-Barranco[1], Bernabé Linares-Barranco[2], Gabriel Jiménez-Moreno[1], Antón Civit-Balcells[1]
[1]Dpto. de Arquitectura y Tecnología de Computadores, Universidad de Sevilla, Av. Reina Mercedes s/n, 41012 Sevilla, Spain. Phone: , Fax: , E-mail: alinares@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) 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 addresses the problem of converting, in a computer, a conventional frame-based video stream into the spike event based representation AER. There exist several proposed software methods for synthetic generation of AER for bio-inspired systems. This paper presents a hardware implementation for one method, which is based on Linear-Feedback-Shift-Register (LFSR) pseudo-random number generation. The sequence of events generated by this hardware, which follows a Poisson distribution like a biological neuron, has been reconstructed using two AER integrator cells. The error of reconstruction for a set of images that produces different traffic loads of event in the AER bus is used as evaluation criteria. A VHDL description of the method, that includes the Xilinx PCI Core, has been implemented and tested using a general purpose PCI-AER board. This PCI-AER board has been developed by authors, and uses a Spartan II 200 FPGA. This system for AER Synthetic Generation is capable of transforming frames of 64x64 pixels, received through a standard computer PCI bus, at a frame rate of 25 frames per second, producing spike events at a peak rate of 10^7 events per second.

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.

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 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.

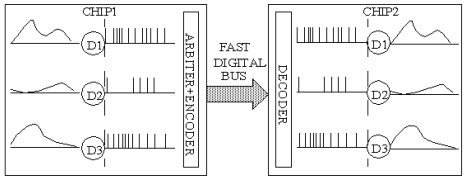

Figure 1. AER inter-chip communication scheme.

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 review some synthetic AER generation methods and present some improvements over earlier presented ones [4]5. In Section 3 different methods are evaluated attending to three criteria: execution time, error of distribution and distance between ideal distribution in two kind of receptors, the Boahen integrator [8] and the Mortara integrator [9]. Finally, section 5 presents the hardware implementation of the Random method for the PCI-AER board developed into VHDL for European project CAVIAR.

2. SYNTHETIC AER GENERATION

One can think of many software algorithms to transform a bitmap image (stored in a computer's memory) into an AER stream of pixel addresses [4]5. In all of them the frequency of appearance of the address of a given pixel must be proportional to the intensity of that pixel. Note that the precise location of the address pulses is not critical. The pulses can be slightly shifted from their nominal positions; the AER receivers will integrate them to recover the original pixel waveform.

Whatever algorithm is used, it will generate a vector of addresses that will be sent to an AER receiver chip via an AER bus. Let us call this vector the "*frame vector*". The *frame vector* has a fixed number of time slots to be filled with event addresses. The number of time slots depends on the time assigned to a frame (for example *Tframe*=40ms) and the time required to transmit a single event (for example *Tpulse*=10ns). If we have an image of *NxM* pixels and each pixel can have a grey level value from *0* to *K*, one possibility is to place each pixel address in the *frame vector* as many times as the value of its intensity, and distribute it with equidistant positions. In the worst case (all pixels with maximum value *K*), the *frame vector* would be filled with *NxMxK* addresses. Note that this number should be less than the total number of time slots in the *frame vector* . Depending on the total intensity of the image there will be more or less empty slots in the *frame vector Tframe/Tpulse*. Each algorithm would implement a particular way of distributing these address events, and will require a certain time.

2.1. The Scan Method

In this method a frame is scanned many times. For each scan, every time a non-zero pixel is reached its address is put on the *frame vector* in the first available slot, and the pixel value is decremented by one. If a pixel value is zero, a blank slot is left in the *frame vector*. This method is very fast. However, the resulting event distribution is very different from the one an AER retina, for example, would produce. Particularly, the events of pixels with low intensity will appear only at the beginning of the *frame vector*.

2.2. The Uniform method

In this method, the objective is to distribute equidistantly the events of one pixel along the *frame vector*. The image is scanned pixel by pixel only once. For each pixel, the generated pulses must be distributed at equal distances. As the *frame vector* is getting filled, the algorithm may want to place addresses in slots that are already occupied. This situation is called a *'collision'*. In this case, we propose three solutions:

A) The *Back-Forward* (Uniform-BF method) solution will put the event in the nearest empty slot of the *frame vector*.

B) The *Forward* (Uniform-F method) solution will put the event in the following empty slot in the *frame vector*.

C) And the *Winner-Takes-All* (Uniform-WTA method) solution will put in the collision position of the vector the event that produces a lower error and will ignore the others. The winning event is the one of the pixel with the lowest intensity.

Uniform-BF, Uniform-F and Uniform-WTA methods, apparently, will make more mistakes at the end of the process than at the beginning. The execution time grows considerably because the collisions consume an important amount of time to be resolved.

2.3. The Random method

This method places the address events in the slots obtained by a pseudo-random number generator based on Linear Feedback Shift Registers (LFSR) [10]11. Due to the properties of the LFSR used, each slot position is generated only once, except position zero, and no collisions appear. If a pixel in the image has intensity *p*, then the method will take *p* values from the pseudo-random number generator and places the pixel address in the corresponding *p* slots of the *frame vector*. They will not be equidistant but will appear along the complete address sequence randomly. This method is faster than any of the *Uniform* methods.

Note that by using an LFSR it would be possible to obtain two very close addresses in a few calls. This can be avoided using a *n*-bit counter for the most significant bits of the address. Figure 2 (left) shows the LFSR structure with a 2-bit counter for a 128x128 frame with 256 gray levels.

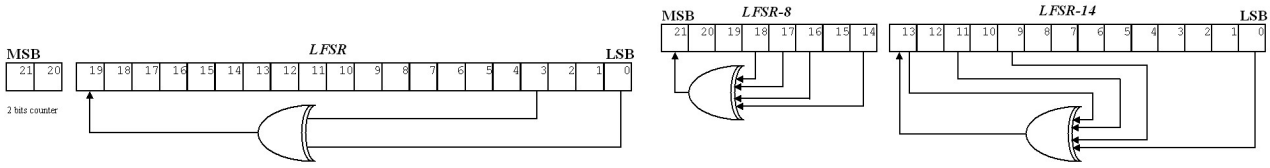

Figure 2. Random method structure on the left and Random-Square on the right.

2.4. The Random-Square method

For the *Random* method with a fixed size counter, the event distribution is poor for low activity pixels. The distribution can be improved substituting the counter by another LFSR.

For a *128x128* frame with maximum gray level of *255*, an *8*-bit LFSR (LFSR-8) is used for selecting *255* slices of *128x128* slots, and another *14*-bit LFSR (LFSR-14) selects the position inside the slice. The image is scanned only once. For each pixel a *14*-bit number is generated by the LFSR-14, which is used to select a slot in a slice. Then, the LFSR-8 is called as many times as the intensity level of the pixel indicates, that is used for selecting the slices to place the events. Figure 2 (right) shows the LFSR structure used.

2.5. The Exhaustive method

The Exhaustive method was proposed in [4]5. This algorithm also divides the address event sequence into K slices of NxM positions for a frame of NxM pixels with a maximum gray level of K. For each slice (k), an event of pixel (i,j) is sent on time t if the following condition is asserted:

$$(k \cdot P_{i,j}) \bmod K + P_{i,j} \geq K \quad \text{and} \quad N \cdot M \cdot (k-1) + (i-1) \cdot M + j = t \qquad (1)$$

where $P_{i,j}$ is the intensity value of the pixel *(i,j)*.

The Exhaustive method tries to improve the Random-Square one by distributing the events of each pixel in equidistant slices.

3. EVALUATION RESULTS

In this Section we compare the methods proposed above and estimate how the performance of the methods is affected by the traffic or load of events in the AER bus. To carry out this analysis a set of random images have been generated, which represent a population of images.

This set of images has been obtained considering two aspects: (a) its histogram must be close to a Gaussian distribution and (b) the number of events required to transmit them. This way, a 100% event load corresponds to an

image with all pixels at maximum value. Consequently, an image with 10% of event load, represents an image that uses 10% of the possible events. Let us generate a 'Test Image Set' (TIS) composed of nine images with event load of 10%, 20%, 30%, ... and 90%. This set will be used to compare the algorithms according to the following criteria:

3.1. Execution Time

Figure 3.a shows the execution time versus the event load of the images, using the same hardware conditions. The *Scan* and *Exhaustive* methods follow an almost constant relation because the event load does not affect much the execution time for these algorithms.

3.2. Distribution Error

In an ideal AER distribution all events for one pixel are equidistant in time: constant frequency of events. In this section, the distribution of events obtained with each method is evaluated. Let us call '*Distribution Error*' how much the event distribution generated by a method deviates from the ideal distribution.

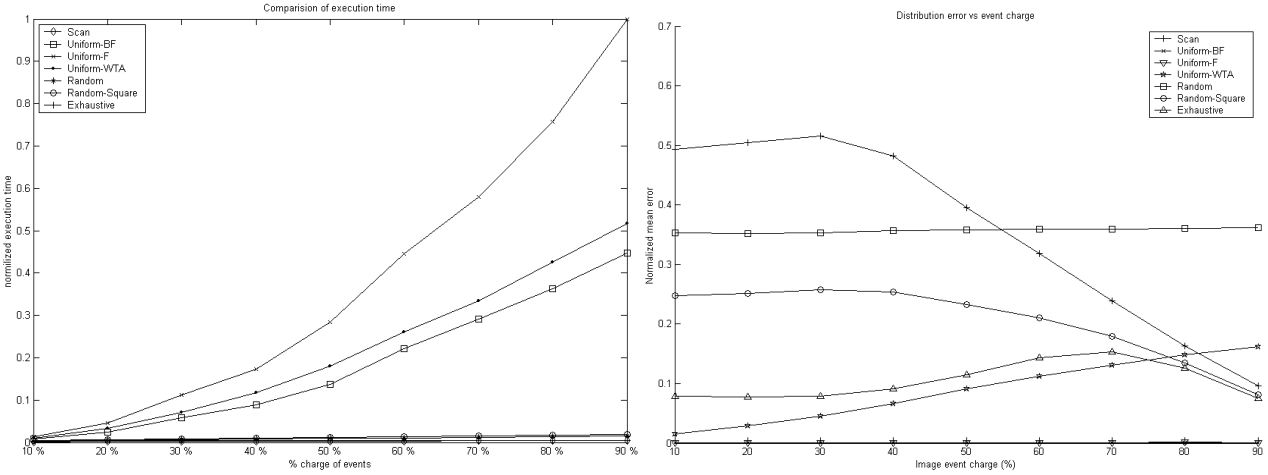

Figure 3. A)Execution time comparison of software implementation (left). B)Mean of NE matrix for methods along incremental charge of events in AER bus (right).

Let us suppose *Dij* is the ideal distance between events of pixel (*i,j*) of a *NxM* image with *K* gray level values. Then $D_{i,j} = \frac{N \cdot M \cdot K}{P_{i,j}}$, where $P_{i,j}$ is the intensity value of pixel (*i,j*).

Let us suppose $d_{i,j}^{k}$ is the distance between the *k*-th event and the (*k*+1)-th one.

$$d_{i,j}^{k} = p_{i,j}^{k+1} - p_{i,j}^{k} \qquad (2)$$

where *p* is the position inside the memory array.

Then we can measure the mean error for a pixel as the average of the differences between the ideal and real distance. The error expression is:

$$e_{i,j} = \frac{\sum_{k=1}^{P_{i,j}} \left| D_{i,j} - d_{i,j}^{k} \right|}{P_{i,j}} \qquad (3)$$

It is easy to see that the worst case for this error measurement is when all the events are together in the address sequence. Therefore, in order to compare the error obtained for different methods and images, the error of each pixel must be normalized with respect to the maximum error associated to the pixel. The following expression is the maximum error for pixel (*i,j*): $me_{i,j} = 2 \cdot (D_{i,j} - 1) \cdot (1 - \frac{1}{P_{i,j}})$, with $P_{i,j} \neq 1$

For $P_{i,j} = 1$, the distribution error is zero, because only one event has to be sent.

Finally, we define a matrix (*NE*) with the same size of the test image, and where each element (*i,j*) represents the error normalized for pixel (*i,j*).

$$NE_{i,j} = {e_{i,j}}/{me_{i,j}} \qquad (4)$$

Figure 3.b shows the measure of the *NE* matrix calculated for the nine test images using the methods proposed. The x-axis represents the image *event load* and the y-axis is the mean normalized error.

3.3. Integrator Cells

Consider the receptor cells proposed by Boahen [8] (diode-capacitor integrator) and by Mortara [9] (two capacitors working in two phases). We have modeled the ideal behavior of these cells in MATLAB. Then for each synthetic AER generation method, different *frame vectors* were obtained. These *frame vectors* were then used to feed an array of integrators of either the Boahen type or the Mortara type. Figure 4 shows the distance between the ideal distribution of events and the real distribution due to each method using the TIS and for each receptor model. No significant difference is observed, except that both Scan and Uniform-WTA methods have the worst behavior.

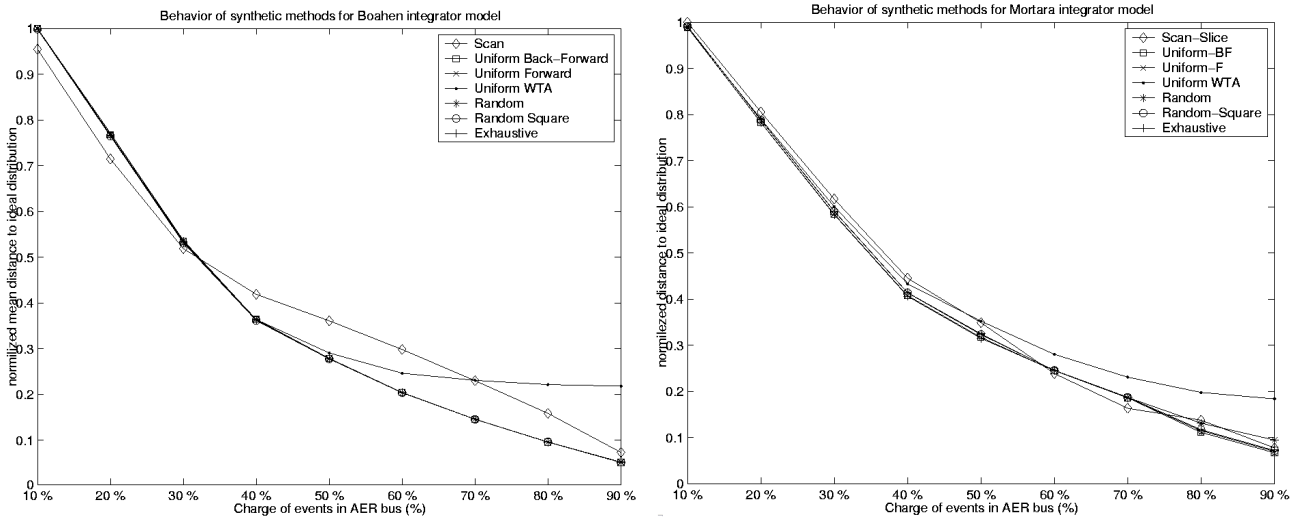

Figure 4. A) Normalized mean distance between methods and ideal distribution for Boahen integrator (left). B) Idem for Mortara integrator(right).

4. HARDWARE IMPLEMENTATION

All simulations presented have been performed in software. However, the final goal is to implement them in hardware. In this paper we present a hardware version of the Random method. This method has been implemented into VHDL for the PCB called CAVIAR PCI-AER. Figure 5 shows the architecture of the presented hardware. This is a PCI interface developed under the European project CAVIAR. The hardware has two modules that work in parallel:

4.1. PCI interface.

A MATLAB code can send images to the PCI address space of the board. The images are sent to the internal memory one by one. The memory is able to store only one of them at the same time. The time between two consecutive frames sent to the PCI board is used to generate Address-Events using the Random method.

To make the Random method work properly, a PCI interface is needed. We have used the PCI LogiCORE of Xilinx. This core has the following characteristics:

- From 32bit/33MHz to 64bit/66MHz PCI bus compatible. Up to 528MBps.
- Up to 3 BAR available for I/O or Memory access.
- One interrupt line: INTA.
- Support both target implementations and master device with burst access to memory.

Thanks to the Xilinx University Program, we had the PCI LogiCORE available as a donation. This core is the interface between the Random method and the windows XP driver. A MATLAB code can transfer real-time video to an AER chip, through them.

4.2. AER Sequencer.

The received frame through the PCI interface is stored into the internal memory. Figure 5 shows the architecture of the PCI-AER Random generator. Both the control unit and the LFSR are continuously accessing to the frame pixels to decide if an event has to be sent or not. The way to make this decision is the following:

- The LFSR produces a 20-bit random number.
- The 12 LSB are used to address the memory and read the pixel value.
- This pixel value is compared with the 8 MSB of the LFSR.
- If the pixel value is greater than the 8 MSB of the LFSR, an event with the 12 LSB of the LFSR is produced, and emitted through the AER bus.
- In other case the process starts. Due to the properties of the LFSR, the same number is not going to be produced until the others 2^{20}-2 numbers were obtained from the LFSR (except zero).

The LFSR is called as many times as numbers it can produce (2^{20}-1) to conclude the processing of one frame. The PCI interface can produce an interrupt to make the software to send a new image to the internal memory. In other case, the same sequence of events can be produced.

This implementation of the method has been developed under VHDL and tested in a Spartan II FPGA. This FPGA is integrated into a PCB developed by our RTCAR[1] group under the CAVIAR project. With this method the board can produce events at a maximum rate of 5Mevents/second.

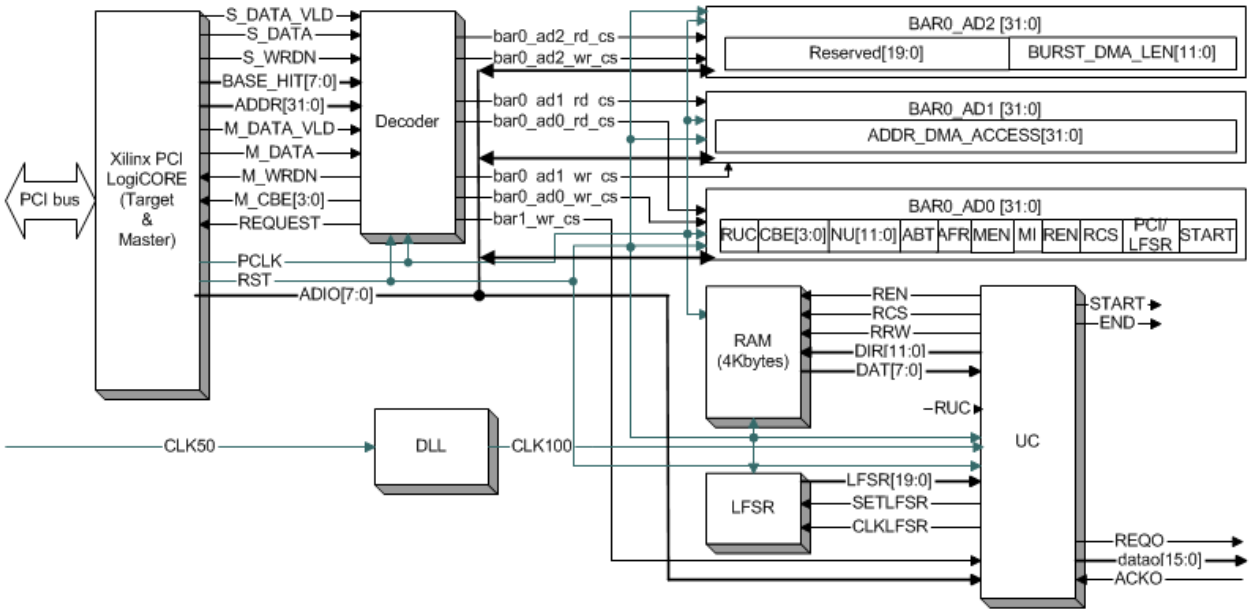

Figure 5. Hardware Interface Architecture.

5. CONCLUSIONS

There are three kind of methods: the scan based, the uniform based and the random based methods. Along the evaluations, uniform ones seem to be the most efficient in distribution of error, but they have the worst time of execution, what makes them unviable for real-time in software. The scan method and exhaustive method have the best results in execution time, but the distribution error is not so good. Random methods are though for an easy hardware implementation, what implies real-time for frames translations. The methods have been tested with the TIS. This set of

[1] Robotic and Technology of Computers Applied to Rehabilitation group. http://www.atc.us.es/

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 of *5 Mevent/second.*

Figure 6 shows an example scenario under CAVIAR EU project, where the hardware implementation of the Random method, working under the CAVIAR PCI-AER G1 board, produces events rates to feed a Winner-Take-All chip. This WTA chip has been developed by INI[2] group. The output of this chip is reconstructed using another board based on FPGA, and represented in a computer screen through the USB bus.

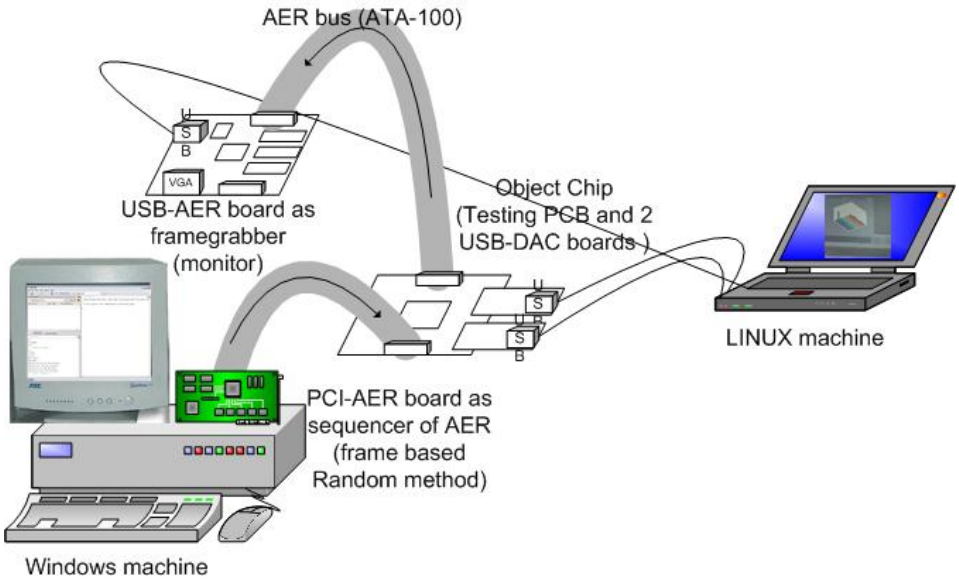

Figure 6: AER system example scenario.

ACKNOWLEDGEMENTS

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

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

[2] Institute of NeuroInformatic. ETZH. http://www.ini.ethz.ch/

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.