PCI to AER Hardware/Software interface for Real-Time Vision processing

Rafael Paz-Vicente, Alejandro Linares-Barranco, Gabriel Jimenez, Anton Civit
Architecture and Technology of Computers, University of Seville, SPAIN.
{rpaz, alinares, gaji, civit} @ atc.us.es

ABSTRACT

In this paper we present a mechanism that allows the coprocessing of video in real-time based into Address-Event-Representation (AER) convolutions chips. Several software methods for generating synthetic AER streams from frames stored in a computer's memory are described and evaluated. Evaluation criteria cover execution time, distribution error and how they perform with two receiver cell models. A hardware bridge PCI to AER for connecting to the convolution chips is presented and evaluated.

KEYWORDS: AER, image processing, convolutions, FPGA, VHDL, PCI.

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.

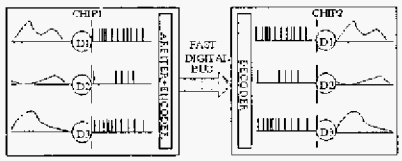

Figure 1. AER inter-chip communication scheme.

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][6][7].

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 4 presents a hardware architecture for the CAVIAR PCI-AER interface 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.

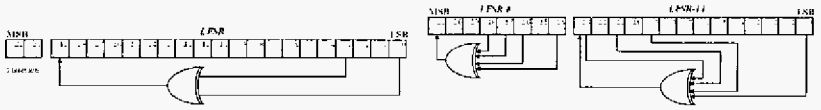

Figure 2. Random method structure on the left and Random-Square on the right.

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.

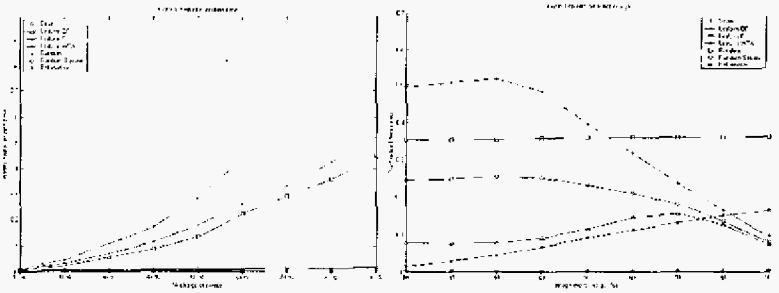

Figure 3. A)Execution time comparison of software implementation (left). B)Mean of NE matrix for methods along incremental charge of events in AER bus (right).

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_{ij} 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.

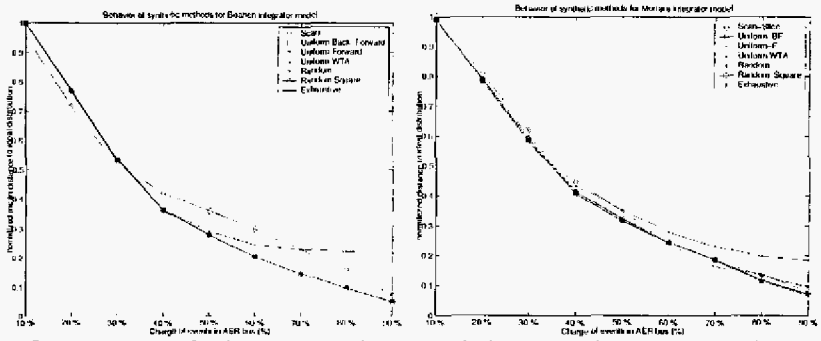

Figure 4. A) Normalized mean distance between methods and ideal distribution for Boahen integrator (left). B) Idem for Mortara integrator(right).

4. HARDWARE INTERFACE

All simulations presented have been performed in software. However, the final goal is to transmit the AER sequence to an AER based system (for example a convolution chip) to perform video processing. For this purpose it is necessary an interface between the computer and the AER bus. Figure 5 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 associated to one image developed by one of the previous methods 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.

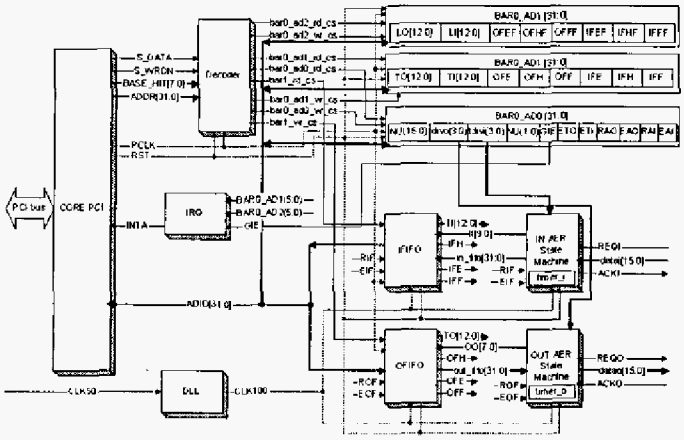

Figure 5. Hardware Interface Architecture.

5. EXPERIMENT

The hardware interface has been implemented using VHDL and synthesized into a Virtex 300 FPGA. It has been tested into a Nallatech Ballyinx prototyping board under Windows 98 operating system.

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 $NxMxK \leq Tframe/Tpulse=3{,}3 \cdot 10^5$. 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. 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 TIS synthesized by all the methods using the CAVIAR PCI-AER. 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.

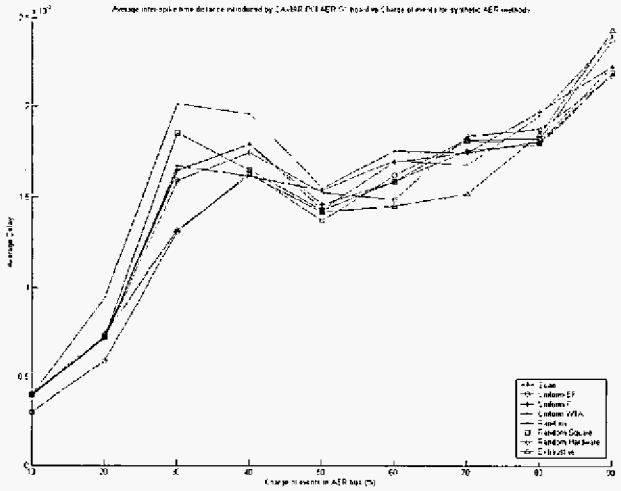

Figure 6. Average Time Delay of CAVIAR PCI-AER for TIS and all the methods.

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 several methods for translating frames of video into AER format, and it have evaluated them.

Although the AER bus has a maximum theoretical bandwidth, those images with a low charge of events can be transmitted through the AER bus with a lower bandwidth than necessary with the introduction of a low error that depends on the events distribution of the image.

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 in viable 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. Our group is now working in the hardware implementation of the random methods.

The methods have 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).

7. ACKNOWLEDGEMENTS

This work was in part supported by EU grant IST-2001-34124 (CAVIAR), and Spanish grant TIC-2000-0406-P4 (VICTOR).

5. 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.