On Synthetic AER Generation

Alejandro Linares-Barranco[1], Gabriel Jimenez-Moreno[1], Antón Civit-Ballcels[1], and Bernabé Linares-Barranco[2]

[1]Arquitectura y Tecnología de Computadores. ETSI Informática, Av. Reina Mercedes s/n, 41012 Sevilla, SPAIN. Phone: 95-455-6145, Fax: 95-455-6449, E-mail: alinares@atc.us.es

[2]Instituto de Microelectrónica de Sevilla, Sevilla, SPAIN.

Abstract

In this paper several software methods for generating synthetic AER streams from images stored in a computer's memory are proposed and evaluated. Evaluation criteria cover execution time, distribution error and how they perform with two receiver cell models. A hardware PCI to AER interface is presented.

1. Introduction

Address-Event-Representation (AER) was proposed in 1991 by Sivilotti [1] to transfer the state of an array of neurons from one chip to another. It uses mixed analog and digital principles and exploits pulse density modulation to code 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 to complete 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.

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 re- ceiver chips in parallel, by properly handling the asyn- chronous communication protocol. The peculiar nature of the AER protocol also allows for very efficient convolution operations within a receiver chip [2].

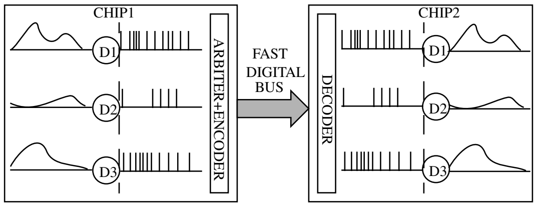

Fig. 1: AER inter-chip communication scheme

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 success of such systems will strongly depend on the availability of robust and efficient development and debugging AER-tools [4][5]. One such tool is a computer interface that allows not only reading an AER stream into a computer 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 interface.

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. If pixel signal time constant is much slower than inter-event timing, 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 $T_{frame} = 40ms$) and the time required to transmit a single event (for example $T_{pulse} = 10ns$). 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* T_{frame}/T_{pulse}. Depending on the total intensity of the image there will be more or less empty slots in the *frame vector*. Each algorithm would implement a particular way of distributing these address events, and will require a certain time.

A. 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*.

B. 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:

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

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

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.

C. 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]. 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 *b*-bit counter for the most significant bits of the address. For each value of the LFSR, four addresses are generated by incrementing the counter. This ensures absence of collisions. Fig. 2 shows the LFSR structure with a 2-bit counter for a 128x128 frame with 256 grey levels.

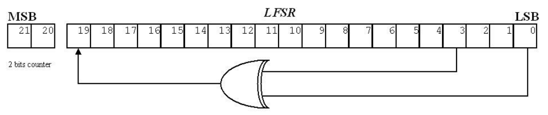

Fig. 2: Random method structure: LFSR with a 2-bit counter.

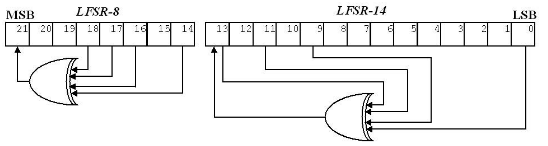

Fig. 3: Random-Square structure: LFSR-8 and LFSR-14 .

D. The Random-Square method

For the *Random* method with a fixed size counter from 1 to the maximum grey level, the event distribution for high activity pixels is acceptable, but poor for low level values. Substituting the counter by another LFSR, the distribution could be improved.

For a *128x128* frame with maximum grey level of *255*, an *8*-bit LFSR (LFSR-8) is used for selecting *255* slices of *128x128* positions, 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, and the LFSR-8 is called as many times as the intensity level of the pixel would indicate. Fig. 3 shows the LFSRs used by this Random-Square method.

E. The Exhaustive method

This algorithm also divides the address event sequence into *K* slices of *NxM* positions for a frame of *NxM* pixels with a maximum grey level of *K*. For slice *k*, an event of pixel *(i,j)* is sent on time *t* if the following condition is asserted:

$$(k \times P_{i,j}) mod K + P_{i,j} \geq K \tag{1}$$

and

$$N \times M \times (k-1) + (i-1) \times M + j = t \tag{2}$$

where $P_{i,j}$ is the intensity value of the pixel *(i,j)* [4][5].

The Exhaustive method tries to improve the Random-Square one by distributing the events of each pixel into the *K* slices at equal distances. The algorithm scans the frame *K* times. In iteration *k*, if the previous condition is true, then the corresponding event is sent, otherwise the algorithm will wait for the following event (no event is sent at time *t*).

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

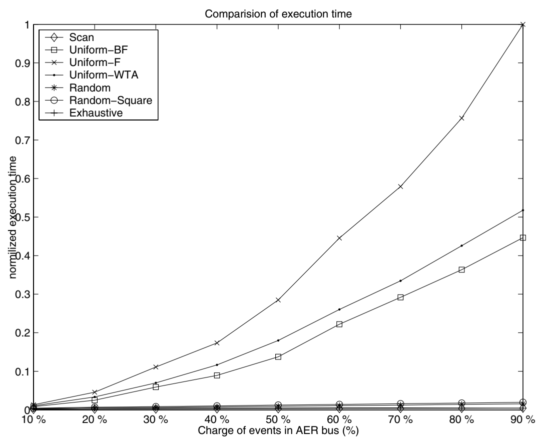

Fig. 4: Execution time comparision of sotware implementation

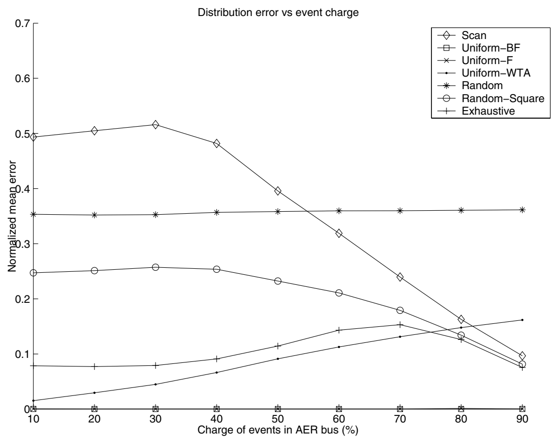

Fig. 5: Mean of NE matrix for methods along incremental charge of events in AER bus.

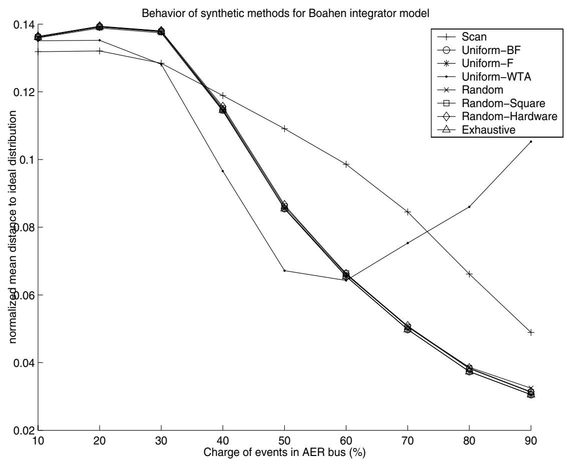

Fig. 6: Normalized mean distance between methods and ideal distribution for Boahen integrator.

compare the algorithms according to the following criteria:

A. Execution Time

Fig. 4 shows the execution time versus the event load of the images. The *Scan* and *Exhaustive* methods follow an almost constant relation because the event load does not affect much the execution time for these algorithms.

B. 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.

Let us suppose *Dij* is the ideal distance between events of pixel (*i,j*) of a *NxM* image with *K* grey level values. Then

$$D_{i,\,j} = (N \times M \times K) / P_{i,\,j} \tag{3}$$

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

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

$$d^{k}{}_{i,\,j} = p^{k+1}{}_{i,\,j} - p^{k}{}_{i,\,j} \tag{4}$$

where p_k is the position of event *k* in the *frame vector*.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:

$$e_{i,\,j} = \frac{\sum_{k=1}^{P_{i,\,j}} \left| D_{i,\,j} - d^{k}{}_{i,\,j} \right|}{P_{i,\,j}} \tag{5}$$

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*):

$$me_{i,\,j} = 2 \cdot (D_{i,\,j} - 1) \cdot (1 - 1/P_{i,\,j}) \text{ with } P_{i,\,j} \neq 1 \tag{6}$$

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*).

$$NE_{i,\,j} = e_{i,\,j} / (me_{i,\,j}) \tag{7}$$

Fig. 5 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.

C. 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 modelled 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. Fig. 6 and Fig. 7 show the distance between the ideal distribution of events and the real distribution due to each method using our "Test Image Set" (TIS) and for each receptor model.

4. Hardware Interface

All simulations presented have been performed in software. However, the final goal is to build a dedicated

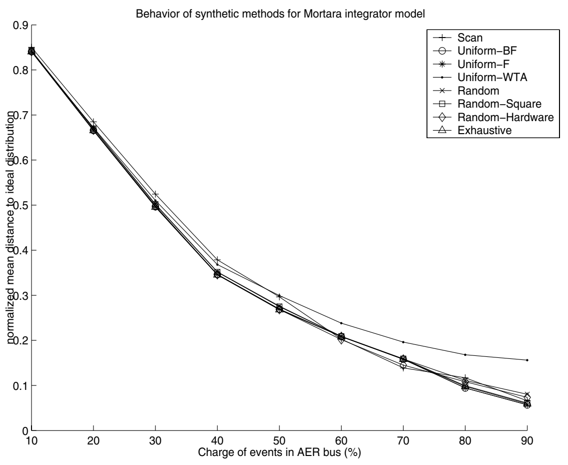

Fig. 7: Normalized mean distance between methods and ideal distribution for Mortara integrator.

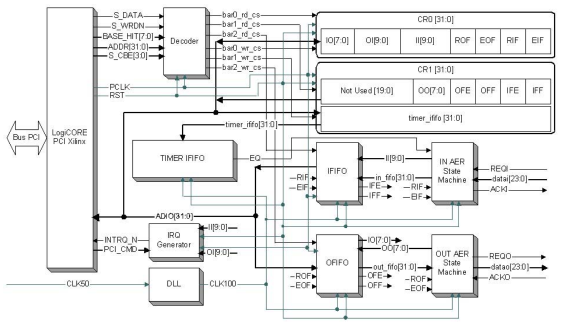

Fig. 8: Hardware Interface Architecture.

hardware that transforms a video frame sequence into an AER stream in real time. Such hardware is presently under development. At this moment a PC based system is available where the frame-AER transformations are performed in software but the resulting *frame vector* is dumped through the computer PCI bus on to an AER bus.

Fig. 8 shows the architecture of the present hardware interface. This is a PCI interface based on the LogiCORE PCI of Xilinx that uses I/O space for configuration and memory space for AER format reading and writing. It has two AER buses, one for incoming AER data and another for outgoing one. There are two FIFOs for both directions. It has a programmable timestamp assignment for incoming AER and programmable wait states for outgoing AER. There is an interrupt generation for avoiding overflows at the incoming FIFO.

The system has been implemented using VHDL and synthesized into a VirtexE 600 FPGA. It has been tested on a Nallatech Ballyinx prototyping board under Linux operating system. It can read or write an AER event every $T_{pulse} = 40ns$. If $T_{frame} = 40ms$, then this implies $N \times M \times K \leq T_{frame}/T_{pulse} = 10^6$.

5. Conclusions

Algorithms for transforming synchronous frame based video streams in asynchronous address event streams are presented and evaluated. Three criteria (execution time, error distribution and distance to ideal behavior with two integrator models) have been evaluated for the seven software methods. A hardware interface between a computer (PCI) and a bioinspired system (AER) has also been presented.

The results presented in Section 3 show that: (a) Software based Unifom methods are not valid for real-time due to the overhead introduced by collision resolution. A hardware version is currently under development to solve these problems. (b) Uniform methods have lower distribution error than others for the test set (TIS). (c) Reconstruction of images, using two models of spike based integrators, show that any method could be valid with small differences among them. The Uniform-WTA has the worst results in this aspect due to the reduction of events by collisions.

6. Acknowledgements

This work was partially supported by spanish grants TIC1999-0446-C02-02, TIC2000-0406-P4-05 (Victor), FIT-07000/2002/921 (Arquimedes), TIC2002-10878-E, TIC-2003-08164-C03-01 (Samanta), and EU grant IST-2001-34124 (Caviar).

7. References

[1] M. Sivilotti, *Wiring Considerations in analog VLSI Systems with Application to Field-Programmable Networks*, Ph.D. Dissertation, Caltech, Pasadena CA, 1991.

[2] T. Serrano-Gotarredona, A. G. Andreou, B. Linares-Barranco, "AER Image Filtering Architecture for Vision-Processing Systems," *IEEE Trans. Circ. and Syst*. Part-I, vol. 46, No. 9, September 1999.

[3] A. Cohen, R. Etienne-Cummings, T. Horiuchi, G. Indiveri, S. Shamma, R. Douglas, C.Koch and T. Sejnowski, *Report on the 2003 Workshop on Neuromorphic Engineering*, Telluride, CO, June 29 to July 19, 2003. {www.ini.unizh.ch/telluride}

[4] A. Linares-Barranco, *Study and Evaluation of AER Interfaces for Neuromorphic Systems*, Ph.D. Dissertation, University of Seville, Spain, 2003. (In spanish)

[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," *Proc. ETFA 2003*, Lisbon, September, vol. 2, pp. 462-467.

[6] K. Boahen, "Communicating Neuronal Ensembles between Neuromorphic Chips," *Neuromorphic Systems*, Kluwer Academic Publishers, Boston 1998.

[7] M. Mahowald, *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function*. Ph.D. Dissertation. Caltech, Pasadena, California 1992.

[8] K. Boahen, "Retinomorphic vision systems II: Communication channel design," *Proc. of the IEEE ISCAS, vol. supplement*, pp. 14-17. May 1996.

[9] A. 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] S.W. Golomb, *Shift Register Sequences*. Laguna Hills, CA: Aegean Park Press, 1982.