Software Generation of Address-Event-Representation for Interchip Images Communications

Alejandro Linares-Barranco, Gabriel Jiménez, A. Civit, J.L. Sevillano, R. Paz.
Arquitectura y Tecnología de Computadores. Universidad de Sevilla.
Av. Reina Mercedes s/n. ETS Ingeniería Informática. 41012-Sevilla, SPAIN
{alinares, gaji, civit, sevi, rpaz}@atc.us.es

***Abstract*: Address-Event-Representation (AER) is a communications protocol for transferring images between chips, originally developed for bio-inspired image processing systems. Such systems may consist of a complicated hierarchical structure with many chips that transmit images among them in real time, while performing some processing (for example, convolutions). In developing AER based systems it is very convenient to have available some kind of means of generating AER streams from on-computer stored images. In this paper we present a method for generating AER streams in real time from images stored in a computer's memory. The method exploits the concept of linear feedback shift register random number generators. This method has been tested by software and compared to other possible algorithms for generating AER streams. It has been found that the proposed method yields a minimum error with respect to the ideal situation. A hardware platform that exploits this technique is currently under development.**

I. INTRODUCTION

Address-Event-Representation (AER) was proposed in 1991 by Sivilotti [1] for transferring the state of an array of analog time dependent values from one chip to another. It uses mixed analog and digital principles and exploits pulse density modulation for coding information. 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 *ms*). Each cell or pixel includes a local oscillator (VCO) that generates digital pulses of minimum width (a few nano-secconds). The density of pulses is proportional to the state of the pixel (or pixel intensity). Each time a pixel generates a pulse (which is called "Event"), it communicates to the array periphery and a digital word representing a code or address for that pixel 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. The inter-chip AER bus operates at the maximum possible speed. In the receiver chip the pulses are directed to the pixels 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 pixel integrates the pulses and reconstructs the original low frequency continuous-time waveform. Pixels that are more active access the bus more frequently than those less active.

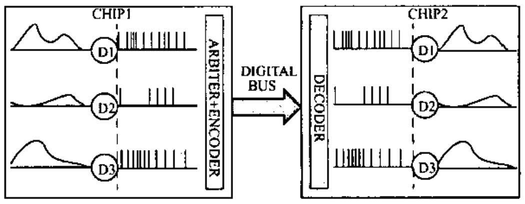

Fig. 1: Illustration of AER inter-chip communication scheme

Transmitting the pixel addresses allows performing extra operations on the images while they travel from one chip to another. For example, inserting properly coded EEPROMs allows 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 success of such systems will strongly depend on the availability of robust and efficient development and debugging AER-tools. One such tool is a computer interface that allows not only reading an AER stream into a computer and displaying it on its 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 [4-6].

II. SYNTHETIC AER STREAM GENERATION

One can think of many software algorithms that would transform a bitmap image into an AER stream of pixel addresses. At the end, 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 because 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 bus that connects to the input of an AER receiver chip. Let us call this vector of addresses the "period". If we have an image of *NxM* pixels and each pixel can have up to *k* grey levels, one possibility would be to place each pixel address in the "period" as many times as the value of its intensity (from 0 to *k*). In the worst case (all pixels with value *k*), the "period" would be filled with *NxMxk* addresses. Each algorithm would implement a particular way of distributing these addresses within the "period". Let us consider 3 different algorithms. The "uniform" method, the "scan" method, and the "random-distribution" method which we propose in this paper.

A. The "uniform" method: in this method the image is scanned pixel by pixel one time. For each pixel, its intensity $x_{ij} \in [0,k]$ is read and x_{ij} pulses are distributed over the "period" at equal distances. As the "period" is getting filled the algorithm may want to place addresses in slots that are already occupied. In this case, it will put the pulse in the nearest empty slot of the "period". This method will make more mistakes at the end of the process and the execution time grows considerably at the end.

B. The "scan" method: in this method the image is scanned many times. For each scan, every time a nonzero pixel is reached its address is put on the "period" in the first available slot, and the pixel value is decremented by one. This method is very fast, because it does not need to look for empty slots, although the image needs to be scanned many times (*k* times in the worst case). However, with this method, the pixels with highest values will appear very frequently at the end of the "period".

C. The "random distribution" method: this method is similar to the uniform method, but instead of having the algorithm trying to place the addresses in equally distant slots, it will take the slot number from a random number generator. The generator is based on Linear Feedback Shift Registers. Consequently, each slot number is generated only once. If a pixel in the image has a value *p*, then the method will take *p* values from the random number generator and place the pixel address in the corresponding *p* slots of the "period". They will not be perfectly equidistant, but in average they will be reasonably well spaced. This method is fast, because the image is swept only once, and because the algorithm does not need to perform searches for empty slots. Next Sections explains in more details the implementation issues for this method.

III. Random Distribution Method

This method is an implementation of Linear Feedback Shift Register (LFSR) based random number generators [7]. Linear feedback shift register random number generators are based on a linear recurrence of the form:

$$x_n = (a_1 x_{n-1} + \ldots + a_k x_{n-k}) \bmod 2$$

where k>1 is the order of the recurrence, $a_k=1$, and $a_j \in \{0,1\}$ for each j. This recurrence is always purely periodic and the period length of its longest cycle is 2^k-1 if and only if its characteristic polynomial

$$P(z) = -\sum_{i=0}^{k} a_i z^{k-i}$$

is a primitive polynomial over the Galois field with 2 elements.

With these premises and limiting the length of the "period" to the maximum number of address events necessary to transmit an image, we know the number of bits needed for the LFSR and the primitive polynomial.

In the software program shown bellow, a maximum of 22 bits are used in the LFSR, although not all random numbers are always used. Only those numbers that are below a limit are used to generate the AER addresses. This limit is different for each image and corresponds to the following formulation:

Let us suppose that **Im** is a matrix that represents the image, that **N** is the number of rows and **M** the number of columns, and that **l** is the length of the "period". Then

$$l = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \mathrm{Im}(i,j)$$

The characteristics polynomial **P(z)** used for 22 bits is:

$$P(z) = z^{20} + z^{19} + 1$$

which corresponds to the LFSR of Fig. 2.

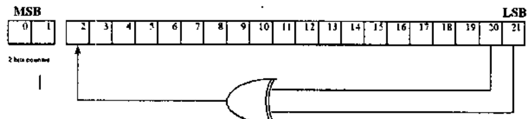

Fig. 2: Left Shift Register for random distribution with a counter for ensurement of the separation

where bits 21, 16, 11 and 6 are set to 1 as a seed value. The two most significant bits are obtained with a counter. This way, the generated random numbers are distributed along quarters of the "period". Consequently, strictly speaking, it is a pseudo-random method.

IV. Software Simulation Results

Using pseudo-randomization to distribute the address events in time, is at the end, a model of the behaviour present in an AER emitter chip with an array of VCOs. Let us check the error introduced by this model. In an ideal AER stream all events for one pixel and image would be equidistant in time. We will now evaluate how much a synthetically generated AER stream deviates from the ideal stream. The three methods in Section II will be compared.

Let us suppose that $d_{i,j}$ is the ideal distance between events of a pixel, and P is the matrix that represents the image.

$$d_{i,j} = \frac{n^{o}\,events}{P(i,j)} = \frac{\sum_{a}^{128}\sum_{b}^{128} P(a,b)}{P(i,j)}$$

And that $e_{i,j}$ is the error transmission for a pixel, defined as the difference between the ideal distance and the average of real distances of events of a pixel. Where $p_k(i,j)$ is the position of the event k of the pixel (i,j) in the temporal array of the AER conversion, and $V_{i,j}$ is the value of the pixel (i,j).

$$e_{i,j} = d_{i,j} - \frac{\sum_{k=1}^{V_{i,j}-1} \left(p_{k+1}(i,j) - p_k(i,j)\right)}{V_{i,j}}$$

Then, the matrix NE is defined as the normalized error for each pixel, respect to the ideal distance as:

$$NE = \begin{pmatrix} e_{1,1}/d_{1,1} & e_{1,2}/d_{1,2} & \cdots & e_{1,128}/d_{1,128} \\ e_{2,1}/d_{2,1} & e_{2,2}/d_{2,2} & \cdots & e_{2,128}/d_{2,128} \\ \vdots & \vdots & \ddots & \vdots \\ e_{128,1}/d_{128,1} & e_{128,2}/d_{128,2} & \cdots & e_{128,128}/d_{128,128} \end{pmatrix}$$

$$\Rightarrow ne_{i,j} = e_{i,j}/d_{i,j}$$

Consider the example image in Figure 3. First, let us compute for each pixel what would be the ideal spacing of its address in the "period". The result is shown in Figure 4(a), normalized to '1'. For each of the synthetic AER generation methods ("uniform", "scan", and "random"), we generate by software the corresponding "period" and compute for each pixel address the average spacing of events. Then we obtain the difference with respect to the ideal case of Figure 4(a). The resulting relative error is shown in figures 4(b)-(d) for the methods "scan", "uniform", and "random", respectively. As can be seen, the "random" method proposed in this paper yields a significant improvement over the other methods. To our knowledge, no other synthetic methods have been reported so far.

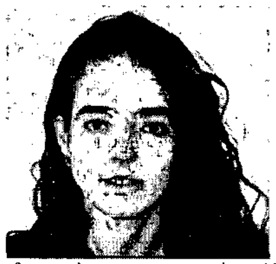
Fig. 3: Reference image to convert into AER format via software.

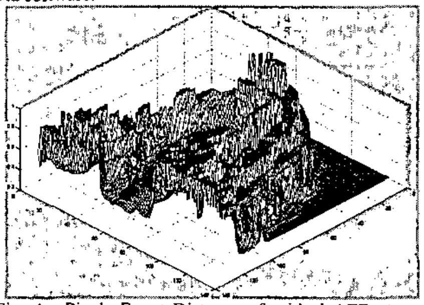
Fig. 4a: Pixels Event Distances for ideal AER stream (normalized to 1).

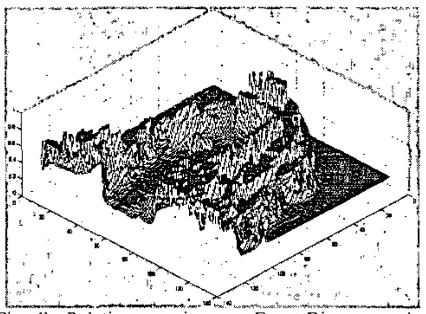
Fig. 4b: Relative error in mean Event Distances using the "scan" method, respect to ideal distribution. Max error of 0,8146.

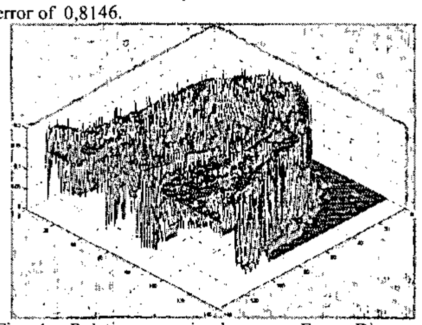
Fig. 4c: Relative error in the mean Event Distances using the "uniform" method, respect to ideal distribution. Max error of 0,1874

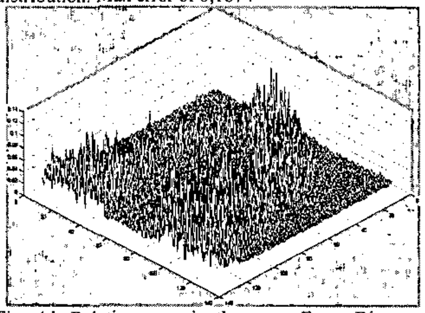
Fig. 4d: Relative error in the mean Event Distances using the "random" method, respect to ideal distribution. Max error of 0,1282

V. COMPUTER-AER INTERFACE

A complete computer-AER interface that exploits the above principle is currently under development. It has two components. One is the software that runs on a standard PC system and manages the conversion of the input image to AER format by generating the corresponding "period". This "period" is then fed to a hardware interface that connects physically to AER chips. The input image for the PC can be a file, a TV signal or a WebCam signal. Let us explain in more detail the hardware and the software interfaces.

A. Hardware Interface.

The hardware printed circuit board whose block diagram is shown in Figure 5 is currently under development. It can read AER streams into the computer as well as generate AER streams from the computer to feed AER receiver chips. It interfaces the computer through its PCI bus. The PCI Switch Core is the interface between the board and the PC, via the PCI bus. The SR Register is a status register, which can be read under a read operation of its PCI address, and the SR Read Control Process will manage the processes.

The interface supports write operations into the AER bus, and read operations from it. For a write operation the OFIFO memory saves an amount of Address Events from the PCI bus and takes control of the AER bus protocol signals. For read operations a process is always waiting for events in the AER input bus and puts them into the IFIFO. When the IFIFO is filled, an interrupt is generated in the PCI bus and the PCI bus reads the IFIFO. A read operation from an AER bus has to have priority over the write operation because data may be lost. This is assured by using interruptions for reading the IFIFO.

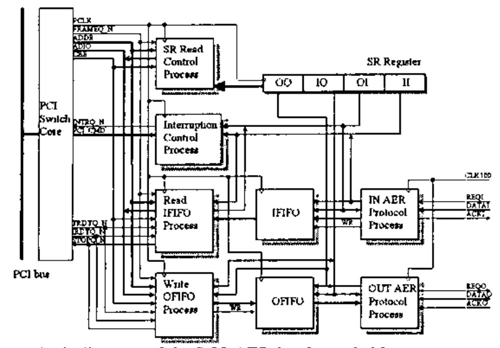

Fig. 5: Block diagram of the PCI-AER hardware bridge.

When AER data has to be written into the AER output bus, the REQ signal has to be activated at the same time the address is placed on the AER bus, and it has to be active until the ACK signal is activated by the receiver chip. Then both the REQ and Address will be cleared, and the bus will be ready for a new address event cycle.

When AER data is waiting in the AER input bus the process will be waiting until REQ is activated. Then the address is given to the IFIFO and the process activates the ACK signal and waits for the REQ low edge to deactivate the ACK signal. Figure 6 shows a typical AER cycle with these 8 clock steps.

1	2	3	4	5	6	7	8	i
Init Read RAM operation	Wait low level ACK signal	Write in AER bus RAM data. Set high REQ	Wait ACK from Receiver	Wait ACK from Receiver	Wait ACK from Receiver	Arrives high level ACK	Set low REQ. Cycle finished	Init Read RAM operation

Fig. 6a: Typical PCI to AER cycle

1	2	3	4	5	6	7	8	i
Waiting for REQ high level	REQ detected, Data latched. Set ACK high	Waiting for REQ low level	Waiting for REQ low level	Waiting for REQ low level	Set low ACK. Cycle finished	Waiting for REQ high level	Waiting for REQ high level	Waiting for REQ high level

Fig. 6b: Typical AER to PCI cycle.

B. Software Interface

The software interface is shown in Figure 7.

Input Image shows the image to be converted to AER format. In this case the image is read from a static file. Consequently, the conversion of the image occurs only one time. The image can be found selecting File from the left Combo list. Parallel port was our first option, but the 300Kbps maximun rate is not enough, so a PCI dedicated system is under constrution to reach the rate of an AER bus.

The "Show AER" option is to activate the Chart at the middle of the window, which represents all the AER bus (or "period") content for the image. This information can be zoomed with the 2 scroll bars under the chart. The maximum address is 16383 for the 128x128 input image.

The "From AER" button is used to make the software read an AER stream stored in memory from the parallel port or PCI board and transform it into a bitmap image, and place it in the "Output Image" square.

The right Combo list allows to make an AER conversion using different methods: Random, Scan and Uniform. There are also two option boxes to calculate distances between the method selected and the ideal distribution, and to write results in files.

When the "To AER" button is clicked the "Input Image" is transformed into an AER stream in the temporal "period", using the method selected. The "period" content is then sent through the hardware board to the AER bus, writing all the events from position 0 to end of the "period". When a period position has no information, a pause is reached for that absent address.

VI. CONCLUSIONS

A windows based application has been developed to convert bitmaps to AER format using pseudo-random number generators to distribute equal events over time, and to compare this distribution with the scan and uniform methods. A dedicated hardware is proposed to write and read to/from an AER bus. An FPGA and RAM memories based implementation is currently under development.

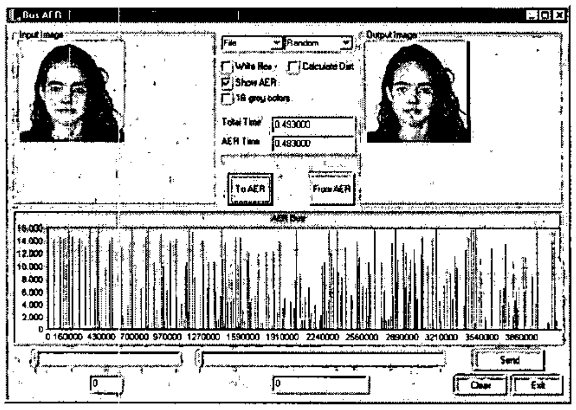

Fig. 7: Software interface.

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

[5] Charles M. Higgins and Christof Koch. "Multi-Chip Neuromorphic Motion Processing". January 1999.

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

[7] Pierre L'Ecuyer, François Panneton. "A New Class of Linear Feedback Shift Register Generators". Proceedings of the 2000 Winter Simulation Conference.