AER Filtering Using GLIDER: VHDL Cellular Automata Description

A. Linares-Barranco[1a], J.L. Sevillano[a], M. S. Obaidat[c], Fellow, IEEE, N. Ferrando[b], J. Cerdá[b], D. Cascado[a], G. Jimenez[a] and A. Civit[a]

[a] Dept. of Computer Architecture and Technology. Univ. of Seville. Spain. [1] alinares@atc.us.es

[b] Dept. of Electronic Engineering. Tech. University of Valencia, Spain.

[c] Computer Science Dept. Monmouth University, NJ, USA

Abstract—**Cellular Automata (CA) is a bio-inspired processing model for problem solving, initially proposed by Von Neumann. This approach modularizes the processing by dividing the solution into synchronous cells that change their states at the same time in order to get the solution. The communication between them is crucial to achieve the correct solution. On the other hand, the Address-Event-Representation (AER) is a neuromorphic communication protocol for transferring asynchronous events between VLSI chips. These neuro-inspired implementations have been used to design sensor chips (retina, cochleas), processing chips (convolutions, filters) and learning chips, which makes it possible to develop complex, multilayer, multichip neuromorphic systems. This paper presents the fusion of both bio-inspired solutions for implementing a vision filter based on 3x3 convolutions. The GLIDER software tool for developing CA has been used to implement the filter in VHDL and synthesize it into the Spartan II 200 of the USB-AER.**

I. INTRODUCTION

Cellular organization in biology has been an inspiration in several fields, like the description and definition of automata. Each cellule principal characteristic is the internal state, defined by a set of information bits. Von Neumann suggests the possibility of the evolution of the internal state into discrete steps like a rudimentary automaton that knows just one simple way to calculate the new internal state. The evolution rule of the system should be the same for all the cellules and a function of the actual internal state and the actual internal state of the neighbourhood [1][2]. Like in biology, the activity of the cellules occurs simultaneously: the same clock signal marks the evolution of the cellules and the internal state actualization in a synchronous way. Von Neumann refers to this system as a Cellular Space and today is know as Cellular Automata (CA) [1].

The first self-reproducing CA, proposed by von Neumann was composed of 2D matrix of cells, and the self-reproducing structure was composed of several hundreds of elemental cells. Each of these cells presented 29 possible states [3]. The evolution rule started from the actual state of each cell and the four closer ones (up, down, right and left). Due to the high complexity of the model, the von Neumann rule has never been implemented in hardware, but some partial implementations has been obtained [4].

Cellular automata are discrete models that consist of a regular grid of cells. The cells have different states along time. The grid dimension is variable. The state of a cell at time t is a function of the states of a number of cells (called its neighbourhood) at time (t – 1). These neighbors are a selection of cells relative to the specified cell, and this selection does not change. Every cell has the same rule for updating, based on the values in this neighbourhood. Each time the rules are applied to the whole grid a new generation is created.

Another bio-inspired approach is the Address-Event-Representation (AER) that consists basically of multiplexing in time the behaviour of the cells of a VLSI neuro-inspired system.

AER was proposed by the Mead lab in 1991 [5] for communicating between neuromorphic chips with spikes (Figure 1). Each time a cell on a sender device generates a spike, it communicates with the array periphery and a digital word representing a code or address for that cell is placed on the external inter-chip digital bus (the AER bus). Additional handshaking lines (Acknowledge and Request) are used to complete the asynchronous communication. In the receiver chip, the spikes are guided to the cell whose code or address was on the bus. In this way, cells with the same address in the

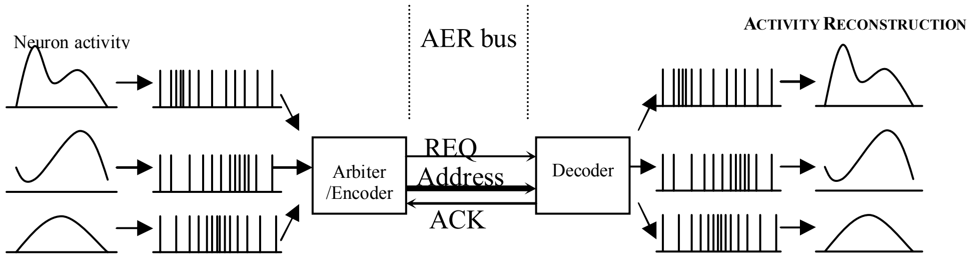


Figure 1. Rate-coded AER inter-chip communication scheme. REQ is the Request line, ACK is the Acknowledge line and the bus Address hold the neuron code.

emitter and receiver chips are virtually connected by streams of spikes. These spikes can be used to communicate analog information using a rate code, but this is not a requirement. More active cells access the bus more frequently than those less active. Arbitration circuits usually ensure that cells do not access the bus simultaneously. Usually, these AER circuits are built using self-timed asynchronous logic such as the one in Boahen [6].

Transmitting the cell addresses allows performing extra operations on the events while they travel from one chip to another. For example, the output of a silicon retina can be easily translated, scaled, or rotated by simple mapping operations on the emitted addresses. These mapping can either be lookup-based (using, e.g. an EEPROM) or algorithmic. Furthermore, the events transmitted by one chip can be received by many receiver chips in parallel, by properly handling the asynchronous communication protocol. There is a growing community of AER protocol users for bio-inspired applications in vision, audition systems and robot control, as demonstrated by the success in the last years of the AER group at the Neuromorphic Engineering Workshop series [7]. The goal of this community is to build large multi-chip and multi-layer hierarchically structured systems capable of performing massively-parallel data-driven processing in real time [8].

The main purpose of this paper is to show that these objectives may be covered by the use of CA with AER intercommunication. As an example, a general purpose 2D filter implementation is presented. This filter is based on a 3x3 kernel to implement AER based convolutions using Cellular Automata VHDL descriptions by GLIDER Java application.

Next section describes a tool for designing CA graphically over a specific CA description language proposed in [9][10]. Section III defines the AER interfaces necessaries to implement this solution. Then in section IV, the proposed AER filter is presented. Finally, section V presents the conclusions.

II. GLIDER: CELLULAR AUTOMATA

GLIDER is a graphic interface that allows: (a) describing a cellular automata composed of cellules and a cellular network, (b) translating the graphical definition into a Cellular Specification Description Language (CSDL) and (c) translating CSDL into VHDL for hardware implementation of the Cellular Automata [9][10].

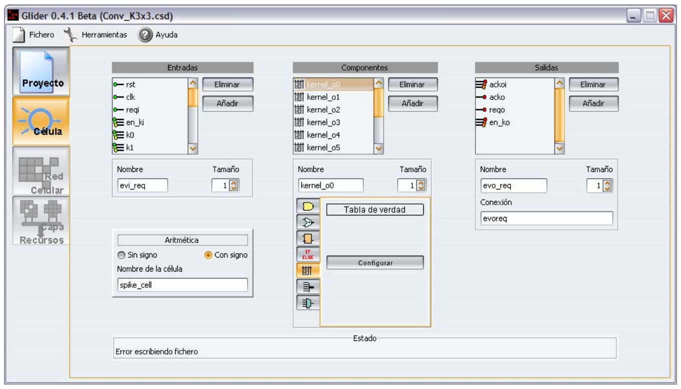
GLIDER has been implemented in Java 1.5, which makes it compatible with any platform and operating system. This software has been developed by the department of Electronic Engineering of the Technical University of Valencia.

A. *Graphic Interface*

There are three main descriptions that define Cellular Automata: cellules, cellular network and resource layer.

1) *Cellule:*
   The cellule definition consists of defining the set of inputs and outputs, that could come from or go to other cellules or could be common for all the cellules; and defining the operations that cellule have to implement with the inputs to generate the outputs. As the cellular automata depend on the time for the state definition, both sequential and combinational logic must be used for a cellule.
   A Cellular Automata could be formed by several cellule definitions.
2) *Cellular Network:*
   The connectivity of the cellules, the morphology of the network and type of cellules used are the parameters that define the Cellular Network.
3) *Resource Layer:*
   This layer defines how a cellular network is connected to another cellular network. It defines also the global inputs and outputs, and the initialization of the cellules.

Figure 2 shows the GLIDER interface for the description of a Cellular Automata.



Entradas: rst, clk, reqi, en_ki, k0, k1
Nombre: evi_req Tamaño: 1
Aritmética: Sin signo [✓ Con signo]
Nombre de la célula: spike_cell
Componentes: kernel_o0, kernel_o1, kernel_o2, kernel_o3, kernel_o4, kernel_o5
Nombre: kernel_o0 Tamaño: 1
Salidas: ackoi, acko, reqo, en_ko
Nombre: evo_req Tamaño: 1
Conexión: evoreq
Estado: Error escribiendo fichero

Figure 2. GLIDER Java application. Cellule definition.

Figure 3 shows a block diagram of the GLIDER internal operation. The graphics interface generates the CSDL files of the Cellular Automata definition. The CSDL compiler is invoked automatically from this Java application and the VHDL files are generated. The console messages are redirected to the application.

An interesting capability of GLIDER is that it keeps continuously checking the consistency of the CSDL generated from the graphics developed by the user. The error or warning messages are reported to the user automatically and in real-time.

Section IV describes a Cellular Automata using GLIDER that is able to implement 3x3 kernel convolutions to filter the AER information of a neuro-inspired system.

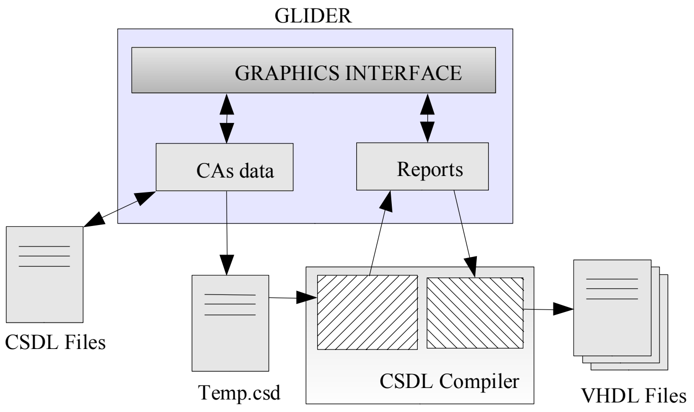


Figure 3. GLIDER Java application. Internal block diagram.

III. AER INTERFACING

The USB-AER tool [11] allows standalone operation and has several functionalities, like hardware mapping and more.

This standalone operating mode requires to be able to load the FPGA and the mapping RAM from some type of non volatile storage that can be easily modified by the users, like MMC/SD cards. However, USB input is also provided for development stages.

Figure 4. USB-AER Board.

The USB-AER board, see Figure 4, includes a relatively large FPGA (Spartan-II 200) that can be loaded from MMC/SD or USB (through the C8051F320 microcontroller), a large SRAM bank (512Kx32 12ns) and two AER ports. Thus the board can be used as a sequencer, a monitor, or an event processor of the AER bus. Due to the bandwidth limitations of full speed USB (12Mbit/s), hardware based event to frame conversion is essential in this board for high, or even moderate event rates.

The board will act as a different device according to the module that is loaded in the FPGA (through MMC/SD or USB), for example, as Mapper, Generator or Sequencer, Monitor or Framegrabber, Datalogger, Player, and Filter [11].

An input AER bus and a output AER bus connected directly to the FPGA allow it to implement any hardware for manipulating or processing AER information, like the AER filter using CA approach described in the next section.

IV. IMAGE FILTERING

The CA described with GLIDER application and implemented in the USB-AER tool consists of a 3x3 kernel convolution filter for AER information.

Let us assume that the AER information transmitted by a visual AER sensor, like an AER retina or a synthetic AER generator from static frames [12], consists of a stream of pixel addresses that identifies the gray level of a visual stimulus. The number of events transmitted through the AER bus by a pixel identifies the gray level of that pixel.

When a cell of the CA receives an event address, this address is decoded and a spike is sent to the target cell of the CA filter. Each cell of the filter is composed of a register and some logic. When the spike arrives, several operations are arranged to transform the state of the target cell and the neighborhood. These are summarized below:

- The register is updated with an increment of the center of the 3x3 kernel, and an ack signal is produced.
- Registers of the 8 neighbors are incremented by the corresponding 3x3 kernel elements. Each neighbor cell produces an ack signal.
- Once all the ack signals are received from the target cell and the neighbor cells, a global ack signal is sent to decoder to enable the acknowledgment to the AER emitter.

Figure 5 shows the digital logic that implement a cell of the CA to change its state when a spike arrives or when a spike arrives to a neighbor cell and to stimulate the neighborhood cells to change their states.

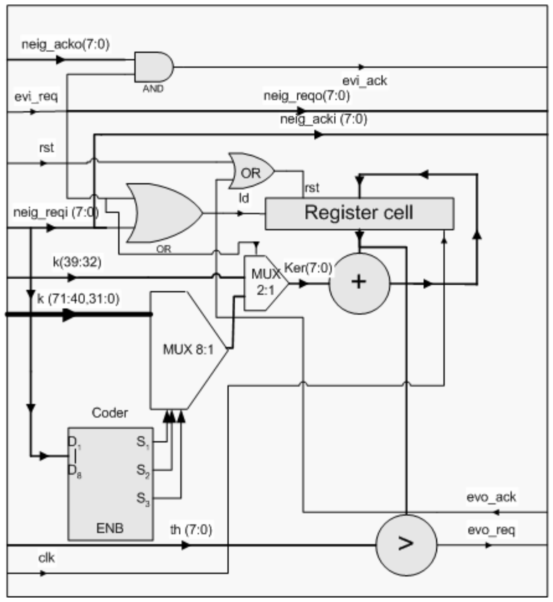


Figure 5. CA 3x3 kernel convolution filter cell logic.

The internal register is incremented when a spike arrives or when a spike arrives to a neighbor cell. The amount to be incremented or decremented depends on the value of the kernel matrix that is completely visible for all the cells. This implies a restriction in the system what makes the kernel sizes to be small. Figure 6 shows the communication between a set of nine neighbor cells when a spike is received.

The target cell is activated by a spike signal, called *evi_req* in the Figure. This signal is produced by a decoder block that receives the AER event, decodes it and sends the spike to the target. The decoder waits for an acknowledgment signal of the CA before starting the decode operation of the next event.

The spike signal produces in the target cell an increment operation according to kernel matrix center (K(40:32)), and simultaneously a spike is sent out to the 8 neighbors. Each neighbor can receive a signal from any of the 8 adjacent cells. Depending on which neighbor cell sends the spike, the corresponding element of the kernel matrix is incremented in that cell.

With this simple operation, for each spike, different increment operations are calculated in the kernel size neighbor of the target cell, according to the kernel values. As several spikes are received by the same pixel address to quantify the pixel gray level, the kernel matrix is incremented to the neighbor of the target pixel as many times as the gray level, implementing the multiplication of the convolution operation.

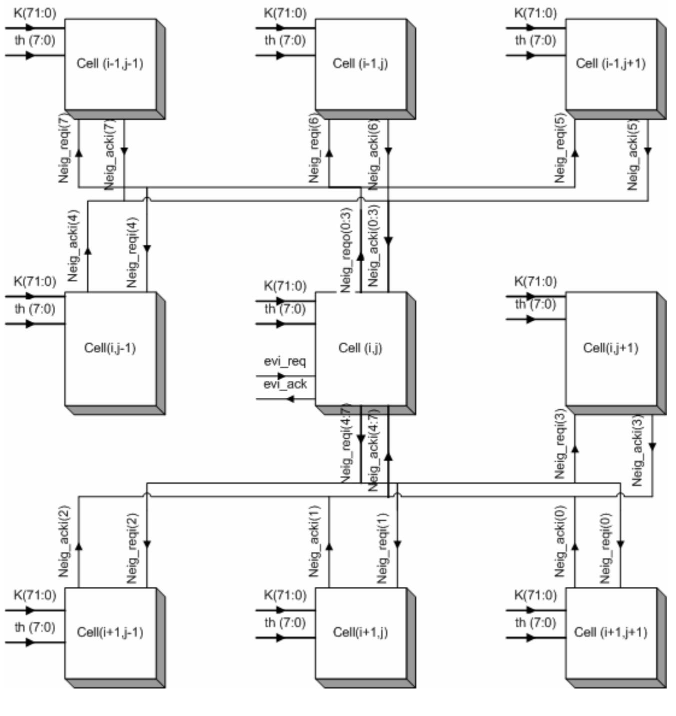


Figure 6. Communication scheme between neighbor cells when a spike is received.

The output of each cell consists also of a stream of events. An event is produced if the value accumulated in the internal register of each cell is greater than a threshold value, that is constant to all the cells, called "th" in Figure 6. When the event is produced, the cell register is cleared. This behavior corresponds to the Integrate and Fire neuron model.

V. Conclusions

Cellular Automation approach has been used to implement an AER neuro-inspired filter for vision processing.

GLIDER covers all the functionalities of the CSDL language and implements them graphically. This makes the design of Cellular Automata fast, easy and intuitive.

An AER based filter based on 3x3 kernel convolutions has been implemented into VHDL using the Cellular Automata approach under the GLIDER software, plus a simple decoder to redirect the input event and an arbiter to encode the output event generated by each possible cellule or cell.

Acknowledgment

This work has been supported in part by the Andalusia Council with the Brain-System project (P06-TIC-01417), and by the Spanish project SAMANTA II (TEC2006-11730-C03-02).

References

[1] J. von Neumann, *The Theory of Self-reproducing Automata*, A. Burks, ed., Univ. of Illinois Press, Urbana, IL, 1966.

[2] M. S. Obaidat, G. I. Papadimitriou and A. S. Pomportsis, "Learning Automata: Theory, Paradigms and Applications," IEEE Transaction on Systems, Man and Cybernetics-Part B, Vol. 32, No. 6, pp. 706-709, December 2002.

[3] A. Burks (ed). *Essays on Cellular Autómata*. University Illinois Press. 1970.

[4] U. Pesavento. An implementation of von Neumann's self-reproducing machine. *Artificial Life*, Vol. 2, pp. 337-354, 1995.

[5] M. Sivilotti, "Wiring Considerations in analog VLSI Systems with Application to Field-Programmable Networks", Ph.D. Thesis, California Institute of Technology, Pasadena CA, 1991.

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

[7] A. Cohen et al., Report to the National Science Foundation: Workshop on Neuromorphic Engineering, Telluride, Colorado, USA, June-July 2006. [www.ine-web.org]

[8] M. Mahowald. "VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function". Ph.D. Thesis. California Institute of Technology, Pasadena, California 1992.

[9] J. Cerdá, R. Gadea, V. Herrero, A. Sebastià. On the Implementation of a Margolus Neighborhood Cellular Autómata on FPGA. En *Field-Programmable Logic and Applications. Lecture Notes in Computer Science* 2778, pp. 76-785, 2003.

[10] J. Cerdá. Arquitecturas VLSI de autómatas celulares para modelado físico. (in Spanish). PhD Thesis. Universidad Politécnica de Valencia, Valencia, Spain, 2004.

[11] F. Gomez-Rodriguez, R. Paz, A. Linares-Barranco, M. Rivas, L. Miró, G. Jimenez, A. Civit. "AER tools for Communications and Debugging". Proceedings of the IEEE ISCAS 2006, Kos, Greece. May 2006.

[12] A. Linares-Barranco, G. Jimenez-Moreno, A. Civit-Ballcels, and B. Linares-Barranco. "On Algorithmic Rate-Coded AER Generation". *IEEE Transaction on Neural Network, Vol. 17, No. 3, pp. 771-788,* May, 2006.