AER-based robotic closed-loop control system

A. Jiménez-Fernández, R. Paz-Vicente, M. Rivas, A. Linares-Barranco, G. Jiménez, A. Civit
Arquitectura y Tecnología de Computadores. Universidad de Sevilla. Av. Reina Mercedes s/n, 41012-Sevilla, SPAIN
ajimenez@atc.us.es

Abstract— Address-Event-Representation (AER) is an asynchronous protocol for transferring the information of spiking neuro-inspired systems. Actually AER systems are able to see, to ear, to process information, and to learn. Regarding to the actuation step, the AER has been used for implementing Central Pattern Generator algorithms, but not for controlling the actuators in a closed-loop spike-based way. In this paper we analyze an AER based model for a real-time neuro-inspired closed-loop control system. We demonstrate it into a differential control system for a two-wheel vehicle using feedback AER information. PFM modulation has been used to power the DC motors of the vehicle and translation into AER of encoder information is also presented for the close-loop. A codesign platform (called AER-Robot), based into a Xilinx Spartan 3 FPGA and an 8051 USB microcontroller, with power stages for four DC motors has been used for the demonstrator.

I. INTRODUCTION

Bio-Inspired and Neuro-Inspired systems or circuits are a relatively novel approaches to solve real problems by mimicking the biology in its efficient solutions. Robotic also tries to mimic the biology and more particularly the human body structure and efficiency of the muscles, bones, articulations, etc.

Spiking systems is one of the neuro-inspired alternatives of mimicking the neurons layers of the brain for processing purposes. One of the most important processing applications is the vision processing through spiking systems. It processes the information into a continuous way, without discretization of the visual information into frames. Hardware implementations of these spiking systems are usually composed by several steps: sensors, filters, convolutions … Each of these steps consists into one or several chips that has to process the information like in the brain: they establishes point to point connections between neurons from one layer or chip to other or other neurons of the next layer or chip. Engineers found a great problem at this point because they need to communicate thousands of neurons from one chip to the next chip, but they have a limitation in the number of pins. Address-Event-Representation solves this problem.

AER was proposed by the Mead lab in 1991 [1] for communicating between neuromorphic chips with spikes (Fig. 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 pixel is placed on the external inter-chip digital bus (the AER bus). Additional handshaking lines (Acknowledge and Request) are used for completing the asynchronous communication. In the receiver chip the spikes are directed to the pixels whose code or address was on the bus. In this way, cells with the same address in the 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. Cells that are more active access the bus more frequently than those less active.

Arbitration circuits usually ensure that cells do not simultaneously access the bus. Usually these AER circuits are built using self-timed asynchronous logic by e.g. Boahen [3].

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 [4][5]. The goal of this community is to build large multichip and multi-layer hierarchically structured systems capable of performing massively-parallel data-driven processing in real time [9].

The neuromorphic approach of AER can be also applied to actuators, like the muscles in the biology. The success of such systems will strongly depend on the availability of robust and efficient development, debugging and interfacing AER-tools [8][10]. From the robotic point of view a very interesting tool will allow the connection between an AER system and a robot. This paper presents, describes and tests a spike based control model for a DC motor. Simulation results under Simulink are presented for the model. A demonstrator has been developed under a three wheels vehicle with two independent motors. The spike based control system of the vehicle is able to follow a reference signal of speed and direction to be able to turn right or left. The spike based control model has been implemented into VHDL and tested into a three-wheel vehicle using the AER-Robot AER-tool[1]. This tool is able to interface an AER system with actuators

[1] http://www.atc.us.es/AERtools

and commercial sensors (position, contact, pressure, temperature…) to allow movements and feedback. Therefore, a more complex spike based control algorithm can be developed for a robot.

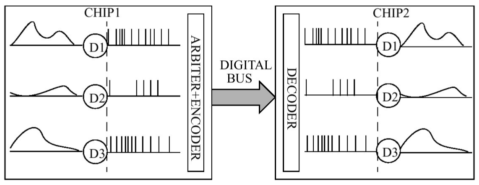

Figure 1. Rate-Coded AER inter-chip communication scheme.

In the following sections we present and describe the AER elements developed under Simulink and VHDL for controlling DC motors, the hardware platform used to test this model and the scenario to test the system.

II. Elements for AER control over DC motors

In this section we present an example of a close-loop speed DC motor regulator, with a spike-based control model, implemented into Simulink with Xilinx System Generator.

Figure 2 shows: (1) the complete simulation model of the DC motor with the AER control, called AERActuator; (2) the detailed AER VHDL control blocks; and (3) the simulation results.

Fig. 2-1 shows the complete simulation model. At left, the inputs of the system: step input, pulse width fixed time for the generated spikes, and the feed-back (the motor speed scaled by a K factor). The AERActuator box contains the VHDL AER based control. And finally, the dynamic motor model used, whose parameters are shown in Table I. The feedback measures the motor speed into rad/sec (from -402 to 402 rad/sec) and it is scaled into a 16-bit signed number by the speed scale K factor (in this particular case 81.5).

TABLE I. Motor Model

Motor Parameter	Parameters Detail		
	Sim. Name	***Value***	***Units***
Nominal Voltage	-	12	*V*
Max Speed	-	402	*rad/s*
Time Constant	-	3.19	*ms*
Terminal Resistance	R	2.06	*Ohm*
Terminal Inductance	L	0.238e-3	*H*
Torque Constant	Kp	23.5	*Nm/A*
Speed Constant	Kv	0.0235	*V/(rad/s)*
Rotor Inertia	J	10.7e-7	*Kgm2*
Friction Coefficient	B	7.5e-5	*Nm/(rad/s)*

Fig. 2-2 shows the VHDL AER based control model: two spikes generator for the reference and the feed-back signal, the AER control model, and an AER to PFM adapter to power the motor.

A. *Exhaustive spikes generator*

Two 16-bit speed reference input signals are received reference (U) and feedback motor speed (Y). These entities generate a spikes stream from these discrete value inputs following the Exhaustive method [10][12] for synthetic rate-coded AER generation. The frequency of the output spike stream generated is proportional to the input value. In the Fig. 2-2 this element is used to generate the speed reference codified into spikes, and to generate a spike stream simulating the behavior of an optical encoder for the feedback control.

B. *AER Holder&Fire*

The objective of this closed-loop regulator model is to apply to the motor the difference between the speed reference and the real motor speed, both codified into spikes. To do that, it is needed an element able to produce a spike output stream with a spike rate equal to the difference between the two inputs spikes rate. When a new spike arrives by one of the input port, this component increases it, in time during a fixed time (U delay or Y delay depending on the input port: reference or encoders respectively). During this time, it waits for the inputs evolution, and holds (waiting the U or Y time before firing or killing the output spike, while no new spike is received), kills (no output spike will be produced, if is received a same sign one by the other port) or fires the spike (without any additional wait, if the hold time ends, or if is received a spike with same sign by the same port).

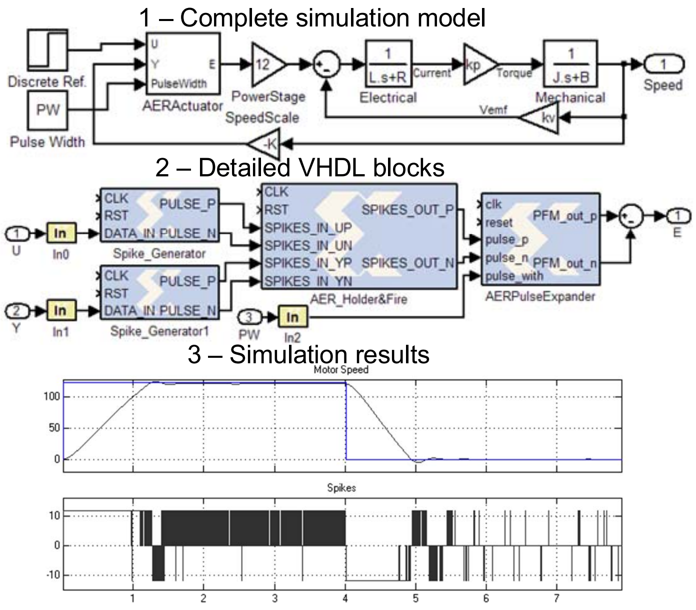

Figure 2. Closed-loop DC motor speed regulator models and results.

C. *AER Pulse Expander*

Finally, the AER Holder&Fire spikes output may be applied to the DC motor. These AER spikes are so short in time (1 clock cycle, in this case 20ns), that is not enough to make the power stage commutate. So, the final VHDL entity receives spikes and it increases them in time, for a fixed time (Th), before applying them to a DC motor. Th must have a value according to the DC motor model and power stages features. Note that if a new spike is received while this entity is increasing another spike (with the same sign), this entity will reset his internal time counter, extending the new one along Th, saturating the output. However, if an opposite sign spike is received along Th, the output is reset, and returned to

‘0’. In other case, the DC motor is powered by a PFM modulation.

Fig. 2-3 shows the simulation result, for a simulation time of 8ms, with a discrete reference of 10000≈1.5MSpikes (122 rad/s), and a pulse width of 100 clock cycles (100*20ns =2us). In the top plot we show the speed reference (U, in blue) and the real motor speed (Y, in black), like a discrete number. We can see there how the real motor speed reaches the speed reference with a short rise time, low over-shoot ratio, and low static error (whose measures and pulse width dependence are out of the scope of this paper). In the plot above (Fig. 2-3 bottom), there is shown the spikes applied to the DC motor, with a spike rate proportional the difference between the reference and the real motor speed. We can see how at the simulation start there is applied a great spike rate (saturating the pulse expander with a continuous ‘1’ output for a long time), and how the spike rate is progressively going down until the speed reach the reference.

III. AER –BASED ROBOTIC PLATTAFORM

The objective of this demo is to demonstrate that a differential control, for a differential wheeled robot, based into AER spikes is possible. In this kind of robots, there are two independent wheels, with its own DC motor, and a free wheel. The final robot speed and turn are proportional to the relative wheels speed, just like a wheelchair.

The desired speed and orientation is given to the robot from a PC through a 433MHz radio module, which receives the command and send it to the FPGA. The internal exhaustive spikes generators of the FPGA generate these spikes stream for the references (speed and orientation). The global system diagram is shown in the Fig. 3.

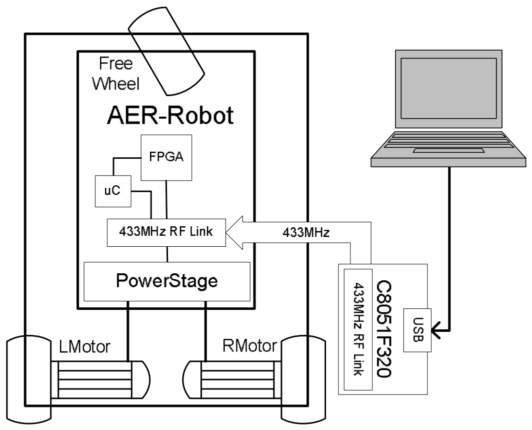

Figure 3. Global System Diagram

A. Robotic Platform

The developed platform has two independent wheels with its own motor. Both wheels have a DC motor manufactured by Maxon Motor, with a gear box and an optical encoder from the same manufacturer. Each DC motor has an 11W power, the gear-boxes have a 1:19 relation, and the optical encoders generate 500 pulses (spikes) per revolution, until 100 KHz.

These DC motors and encoders are driven by the AER-Robot platform (Fig. 4). This platform has been designed and developed under the Spanish grant SAMANTA II and BrainSystem, for controlling an anthropomorphic AER hand [8]. The platform is designed around a Spartan 3 FPGA with four parallel AER connectors (whose are not used for this demo), 4 power stages to manage 4 DC motors, for each motor there are two encoder channels, and 4 Hall Effect current sensors to measure the torque applied to a motor. The FPGA is also connected to an 8051 microcontroller (Cygnal 80C51F320), this microcontroller includes a general purpose ADC (16 channels, 10-bits at 200Ksps), an USB 2.0 full-speed controller, and other many peripherals, like a SPI port.

Fig. 5 top shows the block diagram of the PCB and the VHDL loaded on the FPGA. For this particular application we have attached to the microcontroller a 433MHz radio module over RS-232 for implementing a radio link between the platform and a common PC.

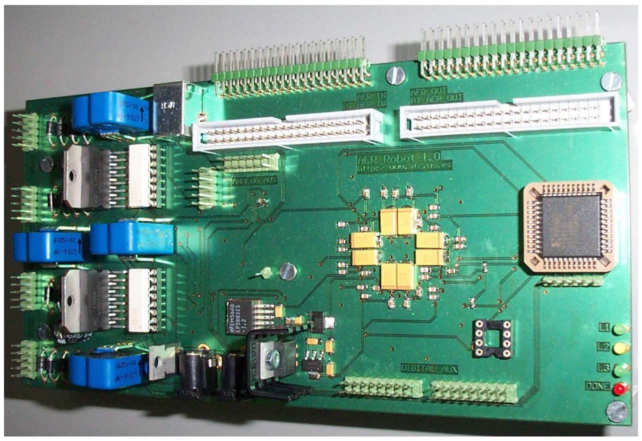

Figure 4. AER-Robot board photograph

B. AER-based VHDL

This subsection describes the demo core. The VHDL components used in this demo have already been described in previous section. For this demo, as there are two DC motors, two AER-based controllers correctly connected are needed.

Furthermore a Motorola SPI [11] slave VHDL component has been included in order to receive the references (speed and turn) from the microcontroller and to write configuration registers to generate the spikes stream references, pulse width, stop / run the platform, reset the FPGA, etc…

The internal VHDL control blocks are shown on the bottom in Fig. 5. Spikes flow form left to right. Figure left shows the references spike generators, one for the reference speed and other for the desired turn. The spike stream applied to the right motor is the result of adding the reference and the positive spikes from the turn reference. However, for the left one it is added to the negatives ones from the turn reference. Appling this, we obtain the desired DC motor speed difference to make the desired turn at the reference speed. Other typical technique is to add the reference turn to one DC motor and subtracts it to the other DC motor. If we use this other technique, the difference between both DC motors speeds may be the double of the reference, so we apply only the positives or negative spikes to both DC motors.

On the bottom of Fig. 5, we can also see three control loops, two for the individual DC motor speed, and other to control the DC motor speed difference. The motors speed are sensed by optical encoders, whose codifies the DC motor speed following a PFM scheme, these information is sent directly to the AER controller, and it is the feedback information.

Because these encoder pulses are so long in time, depending on the DC motor speed, we need to shrink them to an impulse or a spike. In our system every spike takes only 1 clock cycle (20ns), so it is needed to introduce a very easy finite state machine (FSM) to determine, from the typical encoder A and B signals (shifted digital signals that determine the speed and direction of a motor), the spike sign (DC motor speed direction) and to reduce them in time.

The last step, before to apply the spikes directly to the DC motor, is to increase the spikes pulse width. This is made by the AER Pulse Expander component, whose behavior has been described in section 2.C.

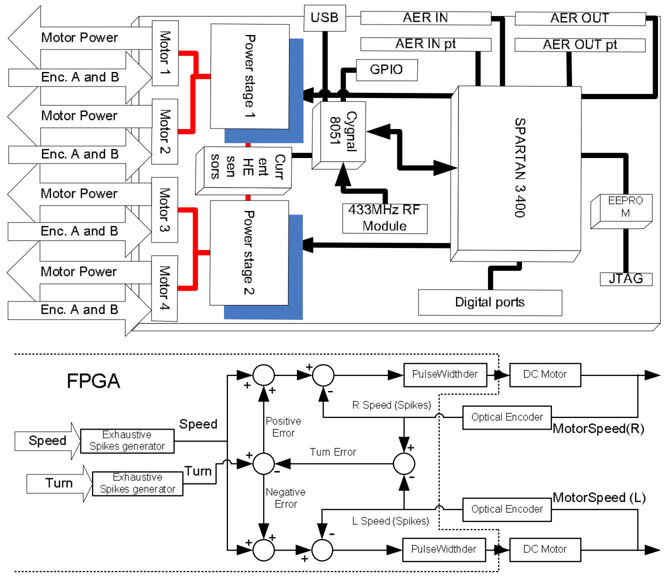

Figure 5. AER-Robot block diagram, and detailed robot control VHDL architecture.

Conclusions

This paper presents a demonstrator of an AER-based controller for a three wheels vehicle. The differential control implemented is based only in spikes flows. Creating spikes-based closed-loops for dynamic systems control.

These control technique is very stable and faster enough for the delay of the power stages of the interface.

An AER-Robot co-design interface based into FPGA (Spartan 3) and an 8051 USB microcontroller has been used to implement the VHDL code to control the three wheels vehicle. The commands of the vehicle are sent through a 433MHz wireless serial radio link form a standard PC.

Acknowledgments

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

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

[4] A. Cohen, R. Douglas, C. Koch, T. Sejnowski, S. Shamma, T. Horiuchi, and G. Indiveri et al., *Report to the National Science Foundation: Workshop on Neuromorphic Engineering*, Telluride, Colorado, USA, June-July 2004. [www.ini.unizh.ch/telluride]

[5] A. Cohen et al., *Report to the National Science Foundation: Workshop on Neuromorphic Engineering*, Telluride, Colorado, USA, June-July 2004. [www.ini.unizh.ch/telluride]

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

[7] P. Lichtsteiner, et al., "A 128×128 120dB 30mW Asynchronous Vision Sensor that Responds to Relative Intensity Change," ISSCC Dig. of Tech. Papers, San Francisco, 2006, pp. 508-509 (27.9).

[8] A. Linares-Barranco, R. Paz-Vicente, G. Jimenez, J.L. Pedreño-Molina, J. Molina-Vilaplana, J.L. Coronado et al.. “AER Neuro-Inspired interface to Anthropomorphic Robotic Hand”. IEEE World Conference on Computational Intelligence. IJCNN. Vancouver, July-2006.

[9] R. Serrano-Gotarredona , M. Oster, P.. Lichtsteiner, A. Linares-Barranco, R. Paz, F. Gomez-Rodriguez, H. Kolle Riis, T. Delbrück, S.C. Liu, S. Zahnd, A.M. Whatley, R. Douglas, P. Häfliger, G. Jimenez, A. Civit, T. Serrano-Gotarredona, A. Acosta, B. Linares-Barranco et al. AER Building Blocks for Multi-Layer Multi-Chip Neuromorphic Vision Systems. NIPS. Vancouver , December-2005.

[10] F. Gomez-Rodriguez, R. Paz, L. Miró, A. Linares-Barranco, G. Jimenez, A. Civit. “Two Hardware Implementation of the Exhaustive Synthetic Aer Generation Method”. LNCS. Vol. 3512. 2005. Pag. 534-540

[11] J. Catsoulis. “Designing Embedded Hardware”. O’Reilly. ISBN: 0-596-00755-8.

[12] A. Linares-Barranco, G. Jimenez-Moreno, A. Civit-Ballcels, and B. Linares-Barranco. “On Algorithmic Rate-Coded AER Generation”. IEEE Transaction on Neural Networks. May-2006