AER and dynamic systems co-simulation over Simulink with Xilinx System Generator

A. Jiménez-Fernández, A. Linares-Barranco, R. Paz-Vicente,C.D. Luján-Martínez, 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 a neuromorphic communication protocol for transferring information of spiking neurons implemented into VLSI chips. These neuro-inspired implementations have been used to design sensor chips (retina, cochleas), processing chips (convolutions, filters) and learning chips, what makes possible the development of complex, multilayer, multichip neuromorphic systems. In biology one of the last steps of the processing is to move a muscle, to apply the results of these complex neuromorphic processing to the real world. One interesting question is to be able to transform, or translate, the AER information into robot movements, like for example, moving a DC motor. This paper presents several ways to translate AER spikes into DC motor power, and to control a DC motor speed, based on Pulse Frequency Modulation. These methods have been simulated into Simulink with Xilinx System Generator, and tested into the AER-Robot platform.**

I. INTRODUCTION

The Address-Event Representation (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 [2].

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 [3]. 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 [7].

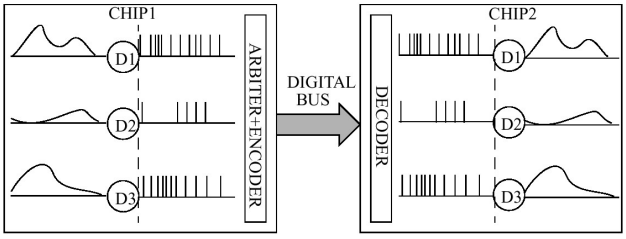

Fig. 1 Rate-coded AER inter-chip communication scheme.

The neuromorphic approach of AER can be also applied to actuators, like the muscles in the biology. In this paper we study and compare two possible transformations of the AER information to DC motor motion, the Pulse Width Modulation (PWM) and the Pulse Frequency Modulation (PFM). We present a Simulink scenario using VHDL blocks to simulate a DC motor speed closed-loop control. We carried out an experiment for controlling a DC motor speed, using VHDL blocks over the Xilinx Spartan3 400 FPGA in the AER-Robot interface, which includes power stages for connecting the FPGA to up to 24 volts and 4 Amps DC motors.

II. AER MODULATIONS FOR MOTOR CONTROL

There are many actuation techniques over DC motors, being two of the most popular: Pulse Width Modulation (PWM) and Pulse Frequency Modulation (PFM). Now, we

propose two ways to translate AER information to these modulation schemes.

A. *AER to PWM translation*

A typical PWM signal has a fixed period (Tpwm) with a variable duty-cycle, or high time (Th). Signal information resides in the duty-cycle. The power applied to a DC motor is proportional to the duty-cycle of the PWM signal.

One possible way to translate AER spikes to a PWM signal, is to fix Tpwm and modify Th according to an AER input spikes rate. We have implemented this with a spikes integrator and a typical PWM modulator. The integrator will integrate the incoming spikes, using a simple counter, along Tpwm. When Tpwm is reached the PWM modulator is reset, using the integrated value for the new Th duty-cycle. The PWM modulator generates a signal with a fixed period and a high time equivalent to the number of spikes counted in the last Tpwm cycle as shown in the Fig. 2.

Fig. 2 AER to PWM translation

B. *AER to PFM adaption*

The PFM scheme is opposed to the PWM one. Now Th is fixed, and the time between pulses (Tpfm) is variable. The information is contained in the pulse frequency, just like the AER information. Therefore, we propose to apply AER spikes almost directly on a DC motor. AER spikes are so shorts in time (about 20ns) that it is not enough to be able to commute the power stage. These spikes may be increased in time (Th) before applying them to a DC motor. Th must have a value according to the DC motor model and power stages features. In the Fig. 3 we show the static speed characteristic of a DC motor model for different Th. Th has been started in 200ns and increased by 200ns until 2us. Simulation shows that the static speed gain can be changed with Th. It also shows that speed saturates with less spike rate when Th is increased, because this entity receives a new spike when it is still increasing the last one.

C. *PWM translation vs PFM adaption*

The advantage of using this PWM scheme is that we can introduce a Z-transfer function (e.g.: proportional-integral-derivative controller) between the spikes integrator and the PWM modulator, because AER information is integrated to a discrete value, and the sample time is known, Tpwm. In the other hand, the PWM translation introduces a time delay equivalent to Tpwm.

With a PFM scheme, the spikes are directly applied to the motor, without delay. But, pulse-based controls are not extended. PFM eliminates the current drain associated with constant-frequency PWM controllers, caused by switching the power stage unnecessarily.

They not only have opposed schemes, have just opposed advantages.

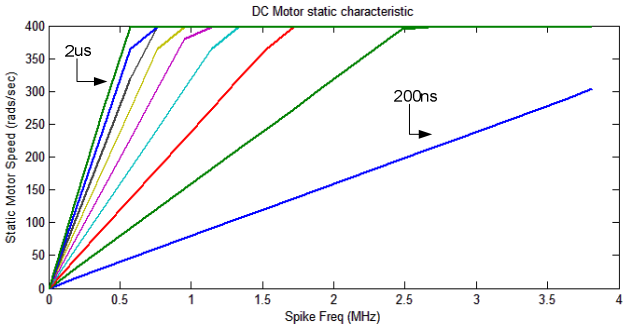

Fig. 3 Static speed charasteristic of a DC motor using a PFM scheme

III. AER CLOSED-LOOP ENTITIES

To implement an AER closed-loop control, there is needed an element that subtracts two spike signals. This means that this element may have two inputs ports, and one output port. The output port must generate a spike-based signal, whose spike rate is equivalent to the inputs spike rate subtraction.

Like in a classical motor speed closed-loop control, in the positive port we apply a reference spike signal (U) and in the negative one, the real motor speed codified by spikes (Y). The DC motor speed is sensed by a quadrature encoder, because these devices modulate the rotation speed following a PFM scheme, just like AER spikes.

Thinking about doing an AER subtraction several mechanisms can be done, for example: a) one way is to modify the inter-spike-interval in the output spikes; b) other way is to generate positive and negative spikes; and finally, c) the last way is to combine both techniques. We propose two different strategies to make this: 1) AER Hold&Fire (focused in mechanism c) and 2) Integrate&Generate (focused in the mechanism a). To test both methods, we have simulated them according with the Fig. 4 model. Simulations have been made with a Maxon motor (whose parameters are shown in Table I).

A. *AER Hold&Fire*

Fig. 6-1 shows this model. It is divided in three parts. The first ones generate spikes [8] from an input signal (U and Y), the next one implements the subtraction method, and the last one increase the Th time of the output spikes of the method before sending them to the motor.

Regarding to this subtraction method, once a spike is received, it is increased by a fixed time (U delay or Y delay depending on the input kind: reference or encoders respectively), waiting for the inputs evolution, and holding (waiting the U or Y time before firing the output spike), killing (no output spike will be produced) or firing spikes (without any wait). For example, if we receive a signed (U) spike, instead of propagate it; we hold it for a fixed time (U delay). If no other spike is received along this time (neither U nor Y), this spike is fired. But, if a same sign spike (U) is received, the first spike is fired and the incoming one hold.

TABLE I. MOTOR MODEL

Motor Parameter	Parameters Value	
	Value	*Units*
Nominal Voltage	12	*V*
Terminal Resistance	2.06	*Ohm*
Terminal Inductance	0.238e-3	*H*
Torque Constant	23.5	*Nm/A*
Speed Constant	0.0235	*V/(rad/s)*
Rotor Inertia	10.7e-7	*Kgm2*
Friction Coefficient	7.5e-5	*Nm/(rad/s)*

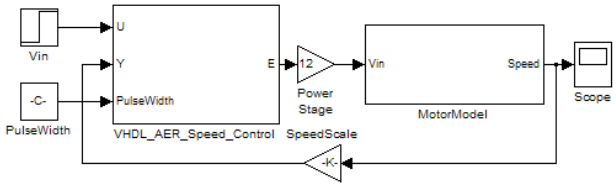

Fig. 4 Motor simulation model

And if a sing event is received (Y), both are killed mutually, and no spike is fired. The fired spike is increased in time until the Th by the next block of the model, before delivering it to the DC motor.

If hold time is too large, the latency grows, but the ripple error decreases before being stable the motor speed. In contrast, if hold time is too short, the spike rates that manage the motor signal increase the ripple, and could be unstable, but the latency decreases. So it's very important to fix these times into a dynamic system like a DC motor.

B. Integrate&Generate

This method implements an incoming spikes integrator and then generates a new spike stream based on the integrated value. The integrator is an accumulator and the generator follows the exhaustive spikes generator [8]. The integrator has positive (U) and negative (Y) spikes input, increasing by one or decreasing by one respectively. The spikes are generated according to the integrated value. Fig. 6-2 shows this simulation model.

One disadvantage is that the DC motor speed is increased very slowly, so the integrator uses to go to the maximum positive value. Later, when the spike rate is greater than the reference spike rate, the integrator goes to the minimum negative value. It causes a severe motor oscillation. It would be corrected with forgotten techniques. However, it has an important advantage: we can introduce a PID control between the integrator and the spikes generator.

IV. SIMULATION. VHDL OVER SIMULINK

Simulink, in addition with Xilinx System Generator, became a very useful tool for VHDL and dynamic model system co-simulation. The two methods explained in the previous section have been modeled as shown in Fig. 6. Results are shown in Fig. 5. There are many speed motor responses for the different methods with different parameters. The simulation parameters are contained in the table II.

TABLE II. SIMULATION PARAMETERS

Sim. number	Simulation Parameters				
	Spike Freq	*AER Method*	*Pulse Width*	*U delay*	*Y delay*
1	2.28MHz	Holder&Fire	2uS	1.3ms	20ns
2	2.28MHz	Holder&Fire	2us	1.3ms	4ms
3	2.28MHz	Holder&Fire	0.1us	1.3ms	20ns
4	1.14MHz	Holder&Fire	2uS	1.3ms	20ns
5	1.14MHz	Integrate&Generate	2uS	N.A.	N.A.

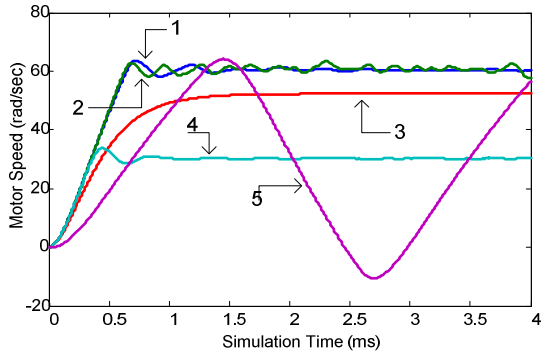

Fig. 5 Motor simulation responses

Fig. 5 shows the different DC motor speed responses, controlled by the different methods and parameters, listed in the Table II. There can be seen the high oscillation of the Integrate&Generate method. Responses 1 and 2 only differs in the Y delay time, where response 2 has a high Y delay time; again we can see an oscillation caused by this delay. If we compare response 1 and 3, with different pulse width, we can see how the pulse width affects to the static gain, and consequently affects to the oscillation. It occurs because a high static gain reduces the phase margin, causing this oscillation. Finally, comparing 1 and 4 responses, whose spikes frequency differs in a half, the 4 responses reach to the half of speed than response 1, showing a linear behavior at same conditions.

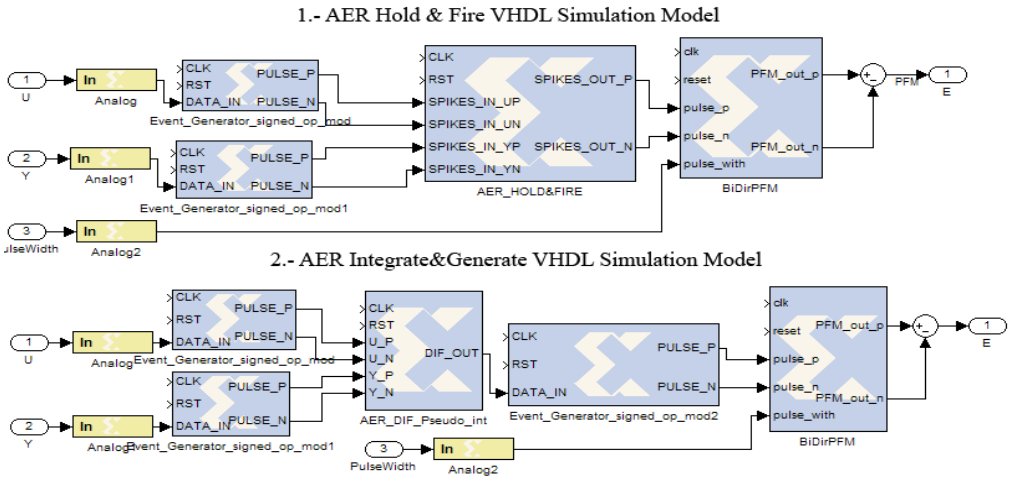

Fig. 6 VHDL blocks for simulation

V. HARDWARE IMPLEMENTATION. THE AER ROBOT PLATFORM

This section describes the hardware implementation of these DC motor modulations based on AER spikes. These modulations have been implemented into VHDL and tested on an AER platform (AER-Robot) (shown in Fig. 7). This platform has been designed and developed under the Spanish grant SAMANTA II to control an anthropomorphic AER hand [6]. The platform is designed around a Spartan3 400 FPGA, with 4 parallel AER connectors (2 input and 2 output), 4 power stages to manage 4 DC motors with two encoder channels, and 4 hall effect current sensor to measure the power consumption of the motors. The interface also has 12 analog sensor inputs and 36 general purpose digital ports. With this FPGA, the interface is able to receive high AER rates, process them together with the input form the robot sensors and encoders, and control the motors of the robot. The platform was developed as an interface between AER systems and robots using two AER buses: one for incoming events and another for outgoing information (AER events) about the state of the motors and the sensors. The input AER bus can be replicated into an output AER bus, called AER IN pt, to conveniently allow a chain of several boards connected by the AER buses. The board has also a Cygnal 80C51F320 microcontroller for the analog to digital conversion (200Ksamples/second and 10-bits) of the sensor measurements and a USB port for the PC connectivity.

Fig. 7 shows a photograph of the AER-Robot Interface PCB. The digital part of the PCB is in the middle.

VI. CONCLUSIONS

Neuromorphic engineers can use Simulink, with the addition of Xinlinx System Generator, to study the behavior of the VHDL AER components applied to dynamic systems. We showed two different techniques to implement a spike-based closed-loop control system. Present work is focused in integrate an AER retina, with a real time AER-based object detection system, in a robotic platform controlled by the exposed control techniques.

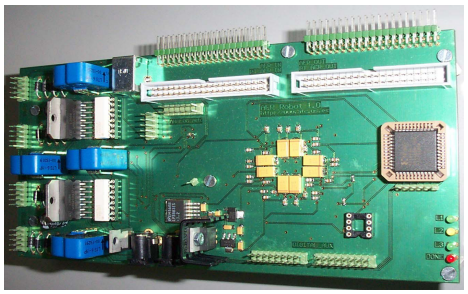

Fig. 7 AER-Robot board photograph

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

[3] 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]

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

[5] 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).

[6] A. Linares-Barranco et al.. "AER Neuro-Inspired interface to Anthropomorphic Robotic Hand". IEEE World Conference on Computational Intelligence. IJCNN. Vancouver, July-2006.

[7] R. Serrano-Gotarredona et al. AER Building Blocks for Multi-Layer Multi-Chip Neuromorphic Vision Systems. NIPS 2005.

[8] F. Gomez-Rodriguez, et al. "Two Hardware Implementation of the Exhaustive Synthetic Aer Generation Method". LNCS. Vol. 3512. 2005. Pag. 534-540