Spike-based control monitoring and analysis with Address Event Representation

A. Jimenez-Fernandez, A. Linares-Barranco, R. Paz-Vicente, G. Jimenez-Moreno
Dpt. Computer Architecture and Technology
Universidad de Sevilla
Sevilla, Spain
ajimenez@atc.us.es

R. Berner
Institute of Neuroinformatics
Uni/ETH Zurich
Zurich, Switzerland
raphael@ini.phys.ethz.ch

***Abstract*—Neuromorphic engineering tries to mimic biological information processing. Address-Event Representation (AER) is a neuromorphic communication protocol for spiking neurons between different chips. We present a new way to drive robotic platforms using spiking neurons. We have simulated spiking control models for DC motors, and developed a mobile robot (Eddie) controlled only by spikes. We apply AER to the robot control, monitoring and measuring the spike activity inside the robot. The mobile robot is controlled by the AER-Robot tool, and the AER information is sent to a PC using the USBAERmini2 interface.**

I. INTRODUCTION

Biology provides efficient solutions for several problems. Trying to mimic biology, bio-inspired and neuro-inspired systems have been extended in the last years.

One of the neuro-inspired models which mimic the neuron layers in the brain are the spiking models. Hardware implementations of these models can use the AER protocol to communicate spikes between different layers. There is a growing community of AER system developers. One of the goals of this community is to build large multichip and multi-layer hierarchically structured systems, for real-time massively-parallel spike processing. In recent years, a diverse set of AER building blocks have been reported: sensors (cochleae, retinas…)[1][2][3], processing layers (filters, convolutions, WTA…)[4][5][6][7], robotic controlling (Central Pattern Generators, Eddie…)[8][9], computer interface for system monitoring and sequencing (PCI-AER, USB-AER, USBAERmini2…)[10][11], and computer software for real-time processing (jAER, MATLAB interface)[12][13]. One of the most extended fields is the vision processing using an AER chain for objects recognition, tracking, etc…[15][16]. EU CAVIAR project developed the largest AER chain reported [14] (see Figure 1).

Rate coded spiking neurons encode information in a stream of pulses (spikes), whose frequency (or spike rate) is proportional to the neuron's excitation, following a Pulse Frequency Modulation (PFM) scheme. If we have several layers with hundreds or thousands of neurons, it is impossible to use a point to multi-point connection between thousands of neurons in a chip for different layers. AER was proposed by Mead lab in 1991 and tries to avoid this problem by using a common bus, the AER bus, and giving a digital code (address) to every neuron. Every time a neuron fires a spike, it goes to an arbiter, managing collisions [17], and to an encoder, who writes the address to the AER bus. Additional lines of request (REQ) and acknowledge (ACK) are required to communicate the address using a 4-phase asynchronous hand-shake protocol. In the receiver, neurons will be listening to the bus, looking for the spikes sent to them. Neurons are virtually connected by streams of spikes (see Figure 2).

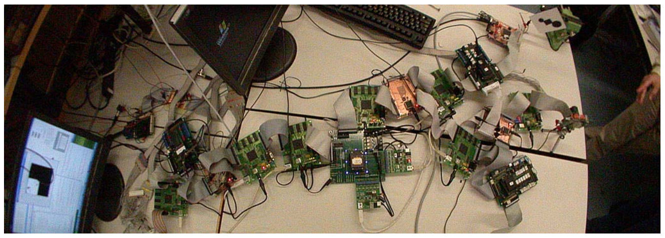

Figure 1 An AER chain for vision processing.

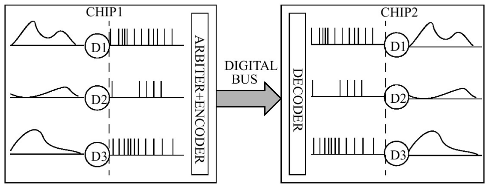

Figure 2 Spiking neurons communication between two layers using AER.

One of the last steps of information processing in biology is controlling a limb. It means to drive many muscles simultaneously and coordinated. Thinking in robots, we can apply neuromorphic technology to robot controlling. We have simulated a closed-loop speed control for DC motors [18], which works like a proportional control, using only spikes, and applying spikes directly to the DC motor. As a practical application of these controls, we have built a differential robot (Eddie) [20]. Eddie is controlled by an AER-Robot board [19].

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

Because Eddie is only controlled using spikes, we can use AER to encode the control loop spikes as AER events, communicating them through the AER ports of the AER-Robot platform, which allows capturing or mapping outside the AER-Robot. We have used the USBAERmini2 board [13] to monitor the AER stream of the DC motor controller for debugging purposes. The USBAERmini2 board can work as an AER monitor and/or sequencer. It transfers the AER information through an USB 2.0 high speed port to a PC. By measuring this information we can adjust the controller parameters in order to improve the performance of the DC motors.

Although this robot only has two DC motors, these tools will allow monitoring and adjusting control parameters in more complex robots (e.g. anthropomorphic hand [19]), and also use a standard PC for real-time controlling.

II. SPIKE-BASED CONTROLS FOR DC MOTORS

This section describes the building blocks of our AER spike controlled robot platform.

A. *VHDL blocks for DC motor controlling.*

PWM applies a digital signal directly to a DC motor. This signal has a fixed period (Tpwm) and a variable duty-cycle, or high time (Th). We change this philosophy by fixing the high time, and changing the frequency, following a Pulse Frequency Modulation (PFM) scheme just like the spiking neurons. So, in our control system, spikes are applied almost directly to the DC motor; we only need to increase their width in time before sending them to the DC motor. As described in [18], the DC motor speed is linear with regard to the spike frequency and the motor speed static gain can be set with the time that every spike is increased in time (spike width). The VHDL entity which implements this functionality is the “Spike Expander” component, which has only one parameter, the spike width.

The idea of classic closed-loop control is to compare a reference with a measured value (feed-back information), subtracting them to obtain the error. Then we apply a function of the error to the system. The simplest function is a proportional function, multiplying the error with a fixed value and applying it to the system. Thinking in spikes, if we have a spike stream as reference value, and a feed-back spike stream, we will need an element which is able to calculate the difference of these two spike-based signals and send the result again as a spike stream. This new spike signal represents the error spike stream. We propose a Hold&Fire (H&F) model to perform this operation.

The idea of the H&F is to hold the incoming spikes and wait for input evolution. A received positive reference (U+) spike will be held internally for a fixed time. If the hold time ends, the spike is fired as a positive spike (+), and the H&F is empty for new spikes. Otherwise, if there are input spikes during the hold time, for example a similar spike (positive reference, U+), the old one is fired, and the new one held. If a feed-back positive spike (Y+) is received during the hold time of a positive reference spike, they are mutually killed, and there is no output. Behavior is similar when a feed-back spike is received, but firing negative spikes instead of firing positive spikes. This entity has two parameters, the hold time for the reference and for the feed-back. Figure 3 shows how H&F can evolve after receiving a positive reference spike.

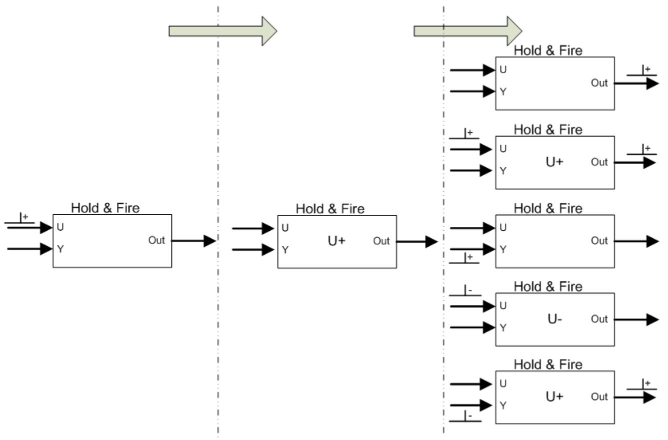

Figure 3 Spike Hold&Fire evolution form a positive reference spike.

We have two different spike sources in our system. The first one is a VHDL component that generates a synthetic spike stream from a discrete number [21]. This entity will generate the spikes for the references in our system, and has only one parameter, the spike rate (16 bit signed number).

Eddie has attached a 2-channel optical encoder for each DC motor. Optical encoders have two signals *A* and *B*, with a 90º phase difference. In both signals the DC motor speed is coded by a 50% duty-cycle square signal. The frequency of this signal represents the DC motor speed. The *A* and *B* signals almost follow a PFM scheme. To translate them to spikes, we only need to fire a spike for every edge in both signals. The spike polarity resides in the phase between *A* and *B*. This element has no parameters.

Finally, we have implemented an SPI slave receiver, which receives the parameters trough an SPI port, and routes them to the right element. This SPI is used to send controller parameters from the PC using a Sillicon Laboratories C8051F320 microcontroller with USB 2.0 full-speed slave transceiver.

B. *AER-Robot board.*

The AER-Robot platform is an interface between AER buses and DC motors which allows driving and measuring them, using the AER technology. This platform has been designed following a FPGA plus microcontroller co-design architecture, its internal blocks can be seen in Figure 4. Each board is able to drive and measure 4 DC motors, and we have included 4 AER ports for full scalability. The FPGA is connected to the AER buses, to H-bridges to drive the DC motors and to the encoder channels. The microcontroller has an USB 2.0 port and a powerful ADC. The microcontroller is connected to the FPGA trough the SPI port. We have included a hall-effect current sensor to every DC motor.

The idea is that the FPGA processes the AER information and controls the DC motors. The microcontroller provides USB connection to a PC for parameter managing, and sends analog (and digital) sensors values to the FPGA.

The main features of this board are:

- Spartan 3 FPGA (XC3S400) clocked at 50MHz
- 24-MHz 8051 microcontroller from Sillicon Laboratories (C8051F320/342).
- 4 AER parallel ports for full scalability.
- 4 H-bridges to drive DC motors with a hall-effect current sensor and 2 encoder channels for every motor.
- USB 2.0 Full-Speed connectivity.
- 12 analog inputs (200kSamples – 10 bits).

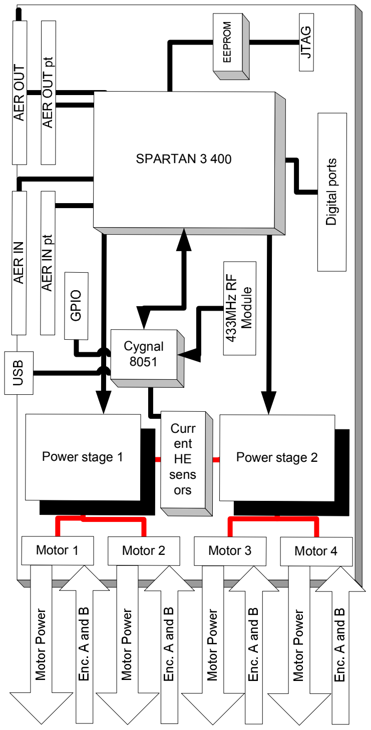

Figure 4 AER-Robot board block diagram.

C. EDDIE

Eddie [20] is a differential robot, driven by an AER-Robot board, with a spike-based control inside. It has two DC motors driving the rear wheels directly and a free wheel in the front. Attached to the rear motors there are two high speed optical encoders to measure the real DC motor speed. The robot is controlled using a standard R/C remote controller for now. In this kind of robots, the main speed of the robot is the average between the speed of the rear wheels, and the turn is the difference between the speed of the wheels. In Figure 5 we can see the robot block diagram and a picture.

The idea of the AER-Robot board is to have the FPGA and the microcontroller working together. In this application, the FPGA will process the spiking information, controlling the robot, and the microcontroller will manage the firmware inside the FPGA. There is a 40MHz R/C receiver connected to the 8051. This kind of devices use a Pulse Position Modulation (PPM), the microcontroller will decode the value of two channels (desired speed and turn), and will send them to the FPGA. The microcontroller also has an USB 2.0 port, allowing managing the FPGA firmware parameters from a PC. Using this feature we can change the parameters of the VHDL components from MATLAB (through a mex file talking with a USB driver).

Inside the FPGA resides the spike-based control that is shown in Figure 6. It is a Multiple Inputs – Multiple Outputs (MIMO) control system. The inputs are represented by the speed and turn references (from the remote controller) and the feed-back information (DC motor real speed, measured using a high speed optical encoder). Every wire in the control diagram is a two bit bus, one bit for positive and one for the negative spikes. The exhaustive spike generators are in green, in blue the spike H&F, in purple the spike extenders, and in light blue the spike generators from the encoder signals. All parameters in VHDL components are controlled by the SPI receiver component.

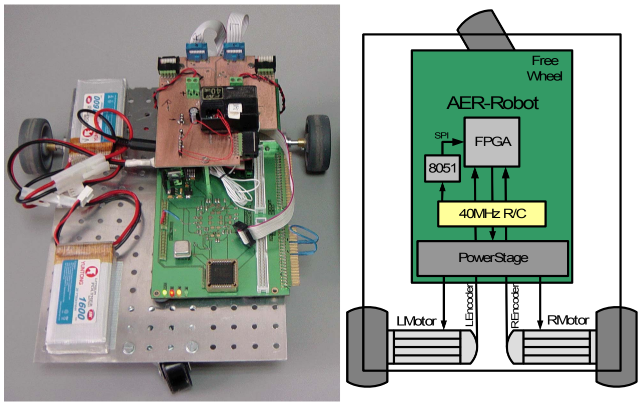

Figure 5 Eddie picture and block diagram

As we can see in Figure 6, spikes flow from left to right, until they reach to the DC motors. Spike-based references are generated in the exhaustive spikes generators. There are three control loops, one for each individual DC motor and a third one to control the turn (speed difference). Speed reference spikes are added or subtracted to the turn reference spikes; this represents the individual DC motor speed reference. Before applying spikes to the DC motors, we increase them in time using the spike extender component. DC motor real speed is measured by optical encoders, and transformed to spikes. Turn close-loop can be disabled for measuring purposes.

The DC motors have been manufactured by Maxon Motors. They have a gear box with 1:19 gear transmission ratio and an optical high-speed encoder attached. Table I lists technical specifications of Eddie.

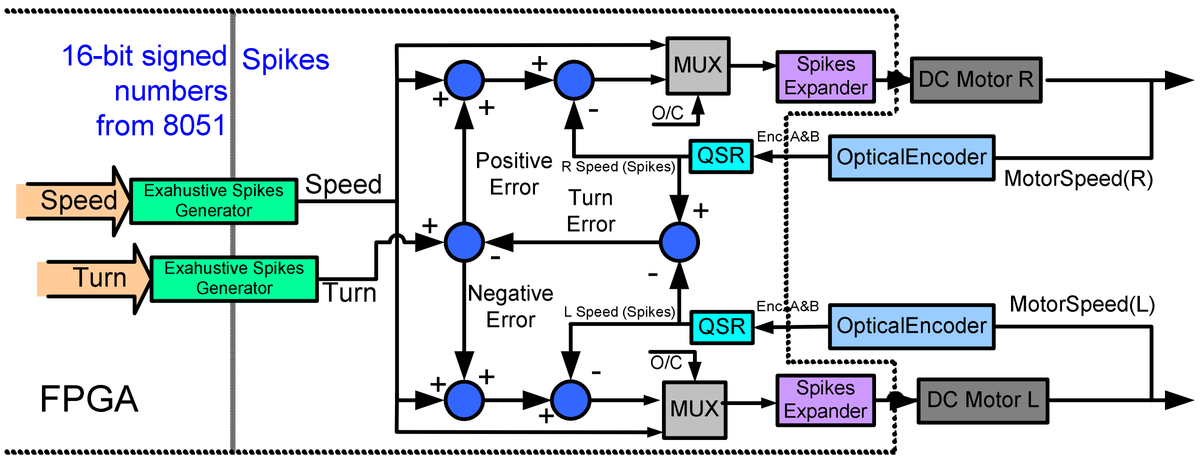

Figure 6 Eddies control components.

D. *VHDL Massive Spike Monitor*

We have implemented a VHDL component to monitor the spike traffic in the different connections between the control blocks. The idea is to use an AE representation, so we need to give an address to every spike (also accounting its polarity) and to send this address (the event) through a parallel AER port. We need three blocks, the first to avoid collisions, the second to encode the spike with its address, and the third to finally send the address with the hand-shake protocol. We show the block diagram in Figure 7.

TABLE I. DC MOTORS PARAMETERS

Parameters	Values and units	
	Value	*Units*
Motor nominal voltage	12	V
Motor start current	4.8	A
Max. motor ang. speed	133.5	r.p.s.
Gear box relation	1:19	n. a.
Max. gear box speed	7.02	r.p.s.
Encoder channels	3	n. a.
Encoder counts per turn	500	n. a.
Max. encoder freq.	100	kHz
Max. references spike rate	267	kSpikes/s
Max. robot linear speed	110.2	cm/s

To avoid collisions, we chose align every spike bus (2 bits) in a single word (10 buses x 2 bits = 20 bits word), and store it in a FIFO, if any spike has been fired (Figure 7 top). If there are many '1' in a single word means that more than one spike has been fired at the same time.

Now we need to encode every spike with its address. We have designed a Finite State Machine (FSM) which, if there is at least one word in the FIFO, loads it into a register and looks for a spike bit by bit. If it finds one, it looks for its address in a ROM, and writes the address in another FIFO (Figure 7 middle), the AER FIFO.

The AER FIFO contains the spikes address, and they are ready to be sent trough the AER port. We use another FSM for AER handshaking (Figure 7 bottom).

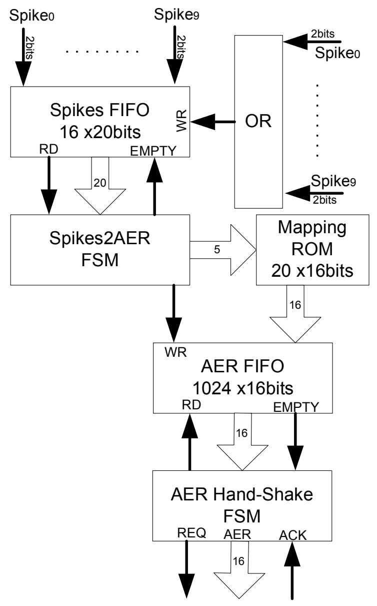

Figure 7 Massive Spike Monitor.

E. *The AER-computer interface*

For monitoring the spike traffic in the robot and visualization and analysation of the AER traffic the USBAERmini2 [13] is used.

This device allows monitoring and sequencing of AER traffic with a time resolution of either one microsecond or 200 nanoseconds. It can be inserted into a connection between two AER devices (pass-through mode) or it can be used in terminal mode, which is the case in this work.

The device consists of a Cypress FX2LP USB2.0 transceiver with 8051 microcontroller and a Xilinx Coolrunner 2 CPLD with 256 macroblocks. In the CPLD, four finite state machines are implemented to handle the handshaking with AER sender or receiver, to control the timestamp counter and to read and write data to and from the FIFOs in the FX2LP. See the block diagram in Figure 8. The timestamp counter on the device is 14 bits wide. Each time it overflows, it sends a message to the host computer which then increments its MSB counter to achieve a total timestamp resolution of 32 bits.

The CPLD is clocked with 30 MHz and achieves a peak monitor rate of 6 megaevents per second and a sustained rate of 5 megaevents per second, which is limited by the host computer. The maximum sequencer rate is 3.75 megaevents per second.

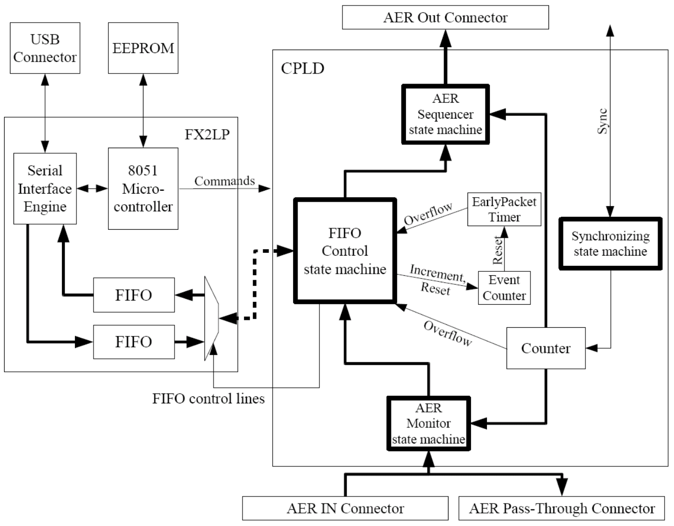

Figure 8 This figure shows a block diagram of the USBAERmini2. The bold boxes inside the CPLD are finite state machines. Bold arrows are signal busses. The device also has synchronization pins which allow two or more boards to capture events with synchronized timestamps.The purpose of the EventCounter and the EarlyPacketTimer blocks are to ensure a minimal USB packet rate of about 100 Hz, even at low event rate.

The device was designed to be simple and cheap to manufacture and also simple to use, e.g. by auto detection whether a receiver is connected to the pass-through port. Several USBAERmini2's can capture data with synchronized timestamps. The timestamps are synchronized using an additional connection and the jitter on the timestamp is less than 33ns (one clock cycle of the CPLD).

As host software, jAER [13], [22] provides the interface between the USBAERmini2 and MATLAB. jAER is an open-source java software for interfacing to AER devices and provides visualization and processing tools for AER data. The jAER classes are accessible from MATLAB, providing an easy way to stimulate, capture and analyze AER data.

III. Real DC motor data monitoring

Now, we are ready to stimulate Eddie with different functions and parameters and analyze the real DC motors response. Using de Open/Close bit (O/C in Figure 6) we can select between an open or closed loop control for each DC motor.

Both AER tools, AER-Robot and USBAERmini2, have USB ports. The AER-Robot USB port (attached to the microcontroller) is used to generate references and set the control parameters (speed, turn, spike width and hold times) for the DC motors from a PC. The USBAERmini2 uses the USB port to communicate the monitored traffic to a PC. AER information has been analyzed using MATLAB. jAER classes have been used as an interface between MATLAB and an USBAERmini2 board. Figure 9 shows both AER-tools connected through an AER bus (gray ribbon), and with a USB wire to a PC for each board. At left there is Eddie, and at right an USBAERmini2 board.

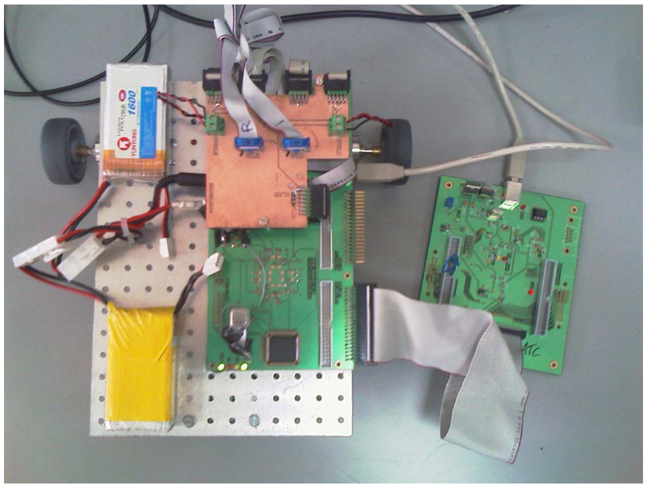

Figure 9 Eddie with USBAERmini2 picture.

First we are going to analyze the open-loop response for different parameters. A synthetic spike generator is connected to a spike expander for each motor. Previous simulations [18] showed that static speed was linear respect to the spike rate, and the static gain could be set with the spike width. Figure 10 shows the open-loop motor response for a constant spike rate reference (95kSpikes/sec) and different spike widths (from 4us to 7us, by 1us). The DC motor speed reaches different static values for different spike widths. Understanding the static gain as the relation between an input spike stream and the DC motor speed feed-back spikes reveals that the gain is a function of the spike width, as we show in the next equation:

$$\frac{motorSpeed(Spikes/s)}{refSpeed(Spikes/s)} = K_{MotorStaticGain} * K(SpikeWidth)$$

Trying to verify previous simulation results, we have stimulated the DC motors with different spike rate inputs, and different spike widths. In Figure 11 we can find the static DC motor speed (y-axis) for different input spike rate (x-axis), increasing the spike widths linearly (starting at 40us, increasing by 4us, until 100us). This verifies the simulation results; the DC static motor speed is linear with respect to the input spike rate. Static gain (the slope in Figure 11) depends on the spike width value.

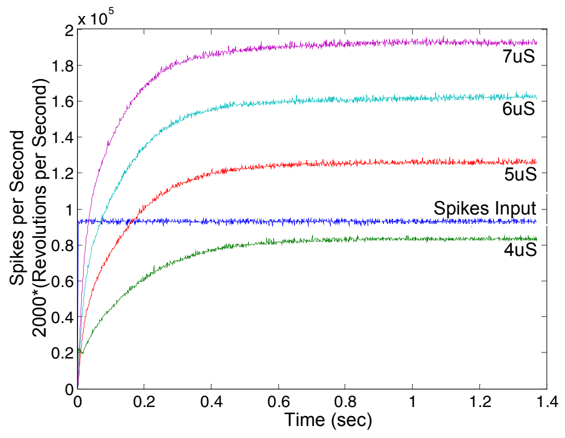

Figure 10 Open-loop step response for constant spike rate input and different spike witdhs.

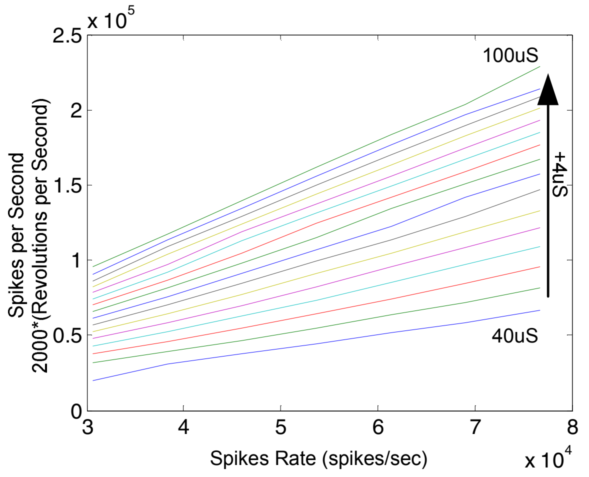

Figure 11 DC motor static speed for different inputs and spike witdhs

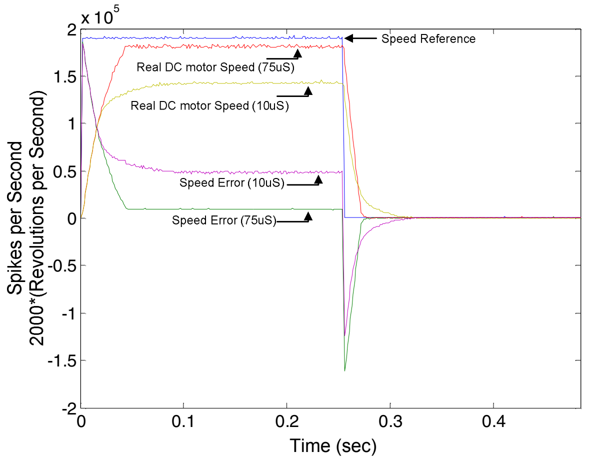

Figure 12 Closed-loop step response with the error.

Setting the O/C bit to '0' (Figure 6), we set a close-loop control, stimulating speed reference with different input functions, and setting turn reference to zero. Figure 12 shows the close-loop response including errors for an input step for different spike widths (10us and 75us). As we can hope from a close-loop proportional (P) control, DC motor speed rise time has been decreased (from 0.45ms to 0.085ms), and the static gain of a close-loop system, as is well known (G/(1+G)), is a number between (0,1). So we always find a static error, but that decreases when the spike width increases. The VHDL block seems to behave just like a classic close-loop P control. DC motor responses are stable with a low oscillation value. This DC motor model shows overdamped responses, so we can't study other variables (like peak time, over peak value, etc…) of a underdamped system.

Figures 13 and 14 show the DC motor behavior with different input functions (ramp and sine) and also with different spike width values. Again it behaves like classical P closed-loop control. In Figure 13 it's clear that the static gain is below one, because the DC motor response slopes are lower than the reference input slope. But when we increase the spike width, the closed-loop static gain becomes close to one. Figure 14 shows a 2Hz sine response, we can see how changing the spike width not only affects to the static response; it also affects the system phase and modifies the system transfer function.

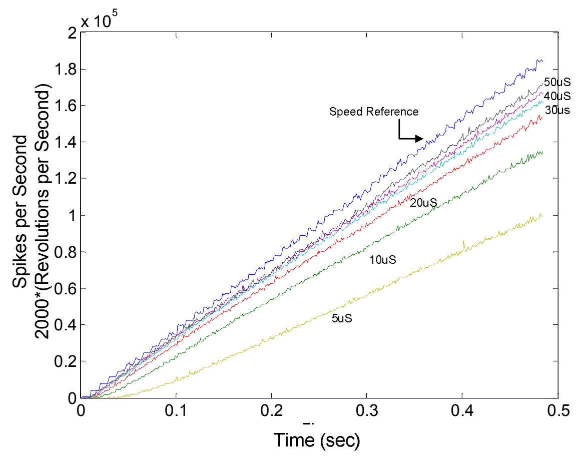

Figure 13 Closed-loop ramp response for diverse spike witdhs.

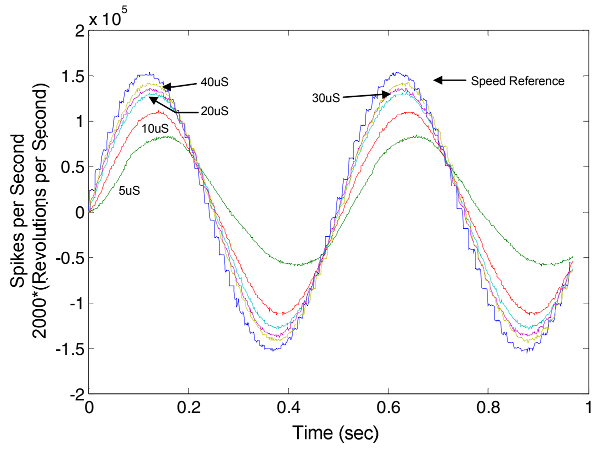

Figure 14 Closed-loop sin response for diverse spike witdhs.

IV. Conclusions

In this paper we presented VHDL components to implement closed-loop spiking control. We have built a robotic platform as a demonstration tool. Using diverse AER-tools, we have monitored the robot behavior, verifying previous simulations results. These controls represent an alternative neuro-inspired way for DC motor control. We use a massive parallel model, like biological models, so we can control a high number of DC motors in real-time. These controls don't need complex hardware, without any algorithmic logic unit (ALU), processor or DSP, only simple time counters, reducing the hardware resources and the power consumption.

Although in this work we show a simple control system, the integration of this kind of spiking controls with other AER systems will allow us to design more complex robotic platforms with higher performance. These platforms will be able to perform high speed sensory information processing, as well as taking decisions and controlling its motion.

Acknowledgment

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

References

[1] P. Lichtsteiner, C. Posh, T. Delbruck., "A 128×128 120dB 15 us Asynchronous Temporal Contrast Vision Sensor". IEEE Journal on Solid-State Circuits, vol. 43, No 2, pp. 566-576, Feb-2008.

[2] E. Culurciello, R. Etienne-Cumming, and K.A. Boahen, " A biomorphic digital image sensor". IEEE Journal of Solid-State Circuits, vol. 38, p 281-204, 2003

[3] V. Chan, S.C. Liu, A.van Schaik, "AER EAR: A Matched Silicon Cochlea Pair with Address-Event-Representation Interface". IEEE Transactions on Circuits and Systems-I. Vol. 54, No 1. pp. 48-59. Jan-2007.

[4] 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.

[5] Oster, M.; Douglas, R.; Shih-Chii Liu, "Quantifying Input and Output Spike Statistics of a Winner-Take-All Network in a Vision System" Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on 27-30 May 2007 Page(s):853 – 856

[6] P. Hafliger. "Adaptive WTA with an Analog VLSI Neuromorphic Learning Chip". IEEE Transactions on Neural Networks, vol. 18, No 2, pp. 551-572. March-2007.

[7] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, C. Serrano-Gotarredona, J.A. Perez-Carrasco, B. Linares-Barranco, A. Linares-Barranco, G. Jimenez, A. Civit. "On Real-Time AER 2-D Convolutions Hardware for Neuromorphic Spike-Based Cortical Processing. IEEE Transactions on Neural Networks, Vol. 19, No 7, pp. 1196-1219. July-2008.

[8] F. Gomez-Rodríguez, A. Linares-Barranco, L. Miró, S.C. Liu, A. van Schaik, R. Etienne-Cummings, M.A. Lewis, "AER Auditory Filtering and CPG for Robot Control". ISCAS 2007.

[9] R. J. Vogelstein, R. Etienne-Cummings, N.V. Thakor, A. H. Cohen. "Dinamic Control of Spinal Locomotion Circuits". ISCAS 2007.

[10] Gomez-Rodriguez, F.; Paz, R.; Linares-Barranco, A.; Rivas, M.; Miro, L.; Vicente, S.; Jimenez, G.; Civit, A.; "AER tools for communications and debugging" Circuits and Systems, 2006. Proceedings. 2006 IEEE International Symposium on 21-24 May 2006. ISCAS.2006.

[11] R. Paz-Vicente, A. Linares-Barranco, D. Cascado, S. Vicente, G. Jimenez, A. Civit. "PCI-AER interface for Neuro-inspired Spiking Systems". ISCAS 2006. Kos, Greece.

[12] Perez-Carrasco, J.A.; Serrano-Gotarredona, T.; Serrano-Gotarredona, C.; Acha, B.; Linares-Barranco, B.; "High-speed character recognition system based on a complex h ierarchical AER architecture" Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on 18-21 May 2008 Page(s):2150 – 2153. ISCAS.2008

[13] R. Berner, Tobi Delbrück, Antón Civit Balcells, Alejandro Linares-Barranco. "A 5 Meps $100 USB2.0 Address-Event Monitor-Sequencer Interface". ISCAS 2007.

[14] 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-Barrancoet "AER Building Blocks for Multi-Layer Multi-Chip Neuromorphic Vision Systems". NIPS 2005.

[15] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez,L. Camuñas-Mesa, R. Berner, M. Rivas, T. Delbrück, S. C. Liu, R. Douglas, P. Häfliger, G. Jiménez-Moreno, A. Civit,T. Serrano-Gotarredona, A. Acosta-Jiménez, B. Linares-Barranco. "CAVIAR: A 45k-Neuron, 5M-Synapse, 12G-connects/sec AER Hardware Sensory-Processing-Learning-Actuating System for High Speed Visual Object Recognition and Tracking" Transactions on neural networks. TNN. Unde review.

[16] Telluride Neuromorphic Engeeniering Workshop 2008 report: https://neuromorphs.net/ws2008/wiki/WorkgroupResults08

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

[18] A. Jiménez-Fernández, A. Linares-Barranco, R. Paz-Vicente,C.D. Luján-Martínez, G. Jiménez, A. Civit. "AER and dynamic systems co-simulation over Simulink with Xilinx System Generator". ICECS 2008

[19] 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.

[20] A. Jiménez-Fernández, R. Paz-Vicente, M. Rivas, A. Linares-Barranco, G. Jiménez, A. Civil. "AER-based robotic closed-loop control system". ISCAS 2008.

[21] F. Gomez-Rodriguez, R. Paz, L. Miro-Amarante, Alejandro Linares-Barranco, Gabriel Jiménez, "Two Hardware Implementations of the Exhaustive Synthetic Aer Generation Method". LNCS 2005.

[22] jAER open-source software project. http://jaer.wiki.sourceforge.net/