AER Neuro-Inspired interface to Anthropomorphic Robotic Hand

A. Linares-Barranco, R. Paz-Vicente, G. Jimenez.

Dept. of Architecture and Technology of Computers

University of Seville

ETSI Informática. Av. Reina Mercedes s/n. Sevilla SPAIN

alinares@atc.us.es

J.L. Pedreño-Molina, J. Molina-Vilaplana

J. López-Coronado

Lab. Of Neurotechnology, Control and Robotics(NEUROCOR)

Technical University of Cartagena.

Campus Muralla del Mar.C/Doctor Flemming s/n.

Cartagena. SPAIN

jl.coronado@upct.es

Abstract **– Address-Event-Representation (AER) is a communication protocol for transferring asynchronous events between VLSI chips, originally developed for neuro-inspired processing systems (for example, image processing). Such systems may consist of a complicated hierarchical structure with many chips that transmit data among them in real time, while performing some processing (for example, convolutions). The information transmitted is a sequence of spikes coded using high speed digital buses. These multi-layer and multi-chip AER systems perform actually not only image processing, but also audio processing, filtering, learning, locomotion, etc. This paper present an AER interface for controlling an anthropomorphic robotic hand with a neuro-inspired system.**

Index Terms **– AER, Neuro-inspired, robotic hand, FPGA, Anthropomorphic.**

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 spikes for coding information. Figure 1. explains the principle behind the AER basics. The emitter chip contains an array of cells (like, for example, the pixels of a camera or an artificial retina chip) where each cell implements a continuously varying time dependent state that change 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-seconds). The rate of pulses is proportional to the state of the cell (or pixel intensity for a retina) assuming spike rate coding is used. Each time a pixel generates a pulse (which is called "event"), 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. 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. In this way, cells with the same address in the emitter and receiver chips are virtually connected with a stream of pulses. The receiver cell integrates the pulses and reconstructs the original low frequency continuous-time waveform. Cells that are more active access the bus more frequently than those less active.

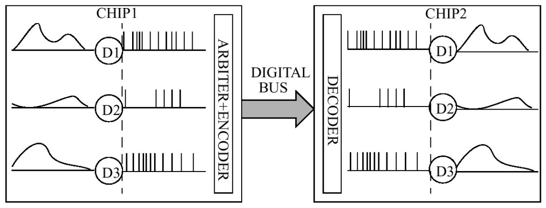

Figure 1. Rate-Coded AER inter-chip communication scheme.

Transmitting the cell addresses allows performing extra operations on the events while they travel from one chip to another. For example in a retina, the activity of the pixels in the array represents the input image. By translating the address of the events during transmission, the image can be shifted or rotated. This translation of the address can be achieved by inserting properly coded EEPROMs. Furthermore, the image transmitted by one chip can be received by many receiver chips in parallel, by properly handling the asynchronous communication protocol. The event-based 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, audition and locomotion 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 complex massively-parallel processing in real time. The success of such systems will strongly depend on the availability of robust and efficient development, debugging and interfacing AER-tools [6].

One such tool is a computer interface that allows not only reading a sequence of events with their timestamps, but also reproduces a sequence of events stored in the computer's memory. Another interesting tool is the interface to actuators and commercial sensors (position, contact, pressure, temperature, …) to allow movements and feedback allowing a more complex and bio-inspired control of a robot.

Factorization of Length and Tension (FLETE) is a bio-inspired control mechanism for robots that computes not only the position of the robot, but also the rigidity of it. In this case the visual information is insufficient, and another kind of mechanical sensor is needed.

With these interfaces you can control a robotic platform using an AER system to give it orders and obtain other kind of sensory information (pressure, contact, position, etc) into AER format. Furthermore, the AER interface allows debugging the robotic platform if you connect it to the computer using the PCI-AER interface [5].

In his paper we present a new AER Interface to connect the AER system to a set of actuators (motors) and sensors. The interface has been used to connect a computer to the UPCT anthropomorphic robotic hand in order to enable an AER system to control a complex and bio-inspired robot. The hand is driven by an agonist-antagonist opponent system. In order to measure joint position, velocity, and direction of rotation, hall-effect position sensors were integrated at each joint of the fingers. Force sensors are mounted on the curved surface of the fingertips, and on the palm.

In the following sections we describe the anthropomorphic hand, the AER interface for a robot, the PCI-AER interface to the computer for debugging purpose, and some results.

II. Anthropomorphic Hand

The UPCT anthropomorphic robot design hand is based on the biomechanics modelling of the human hand, as well as on designs by manufacturers of robot hands. The hand has three fingers and an opposing thumb, and four degrees of freedom for each finger. The fingers are mounted on a rigid palm. Figure 2 shows one photography of the UPCT Hand, placed over a industrial robot for grasping, reaching and handling tasks.

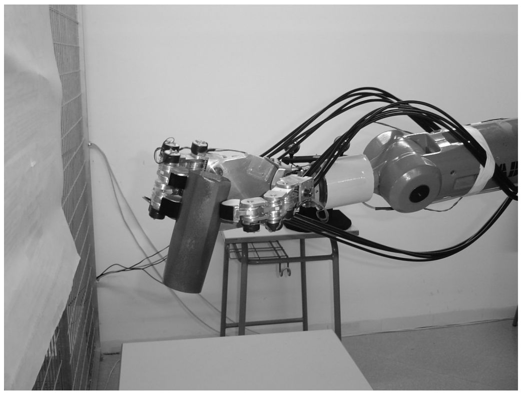
Figure 2. UPCT Robot Hand.

The design of the multi-jointed finger presents three joints (metacarpophalangeal (MCP), proximal interphalangeal (PIP), and distal interphalangeal (DIP) joints, respectively), where DIP and PIP are coupled. Both the PIP and DIP joints have flexion and extension, and the MCP joint consist of two joints that allow flexion-extension and adduction-abduction motions. In the finger design, the muscle-like actuators are DC motors. Each joint of the finger is actuated through 2 polystyrene tendons, routed through pulleys and driven by DC motors. The joints are moved by an agonist-antagonist opponent system. In order to measure joint position, velocity, and direction of rotation, hall-effect position sensors were integrated at each joint of the fingers. Tactile sensors based on FSR (Force Resistive sensing) technology are mounted on all the joints and on the palm emulating artificial tactile surfaces. The flexibility of these sensors is very suitable for the implementation on the curved surface of the fingertips for precision grasping and manipulations tasks. One two-axis sensor placed on the palm is employed to correct the stability of the gross grasping. It permits the tactile guided for the movement of the wrist of the robot hand-arm [7]. Each one of the fingers that conforms the biomechanical hand is driven by a mechanism constituted by an assembly of pulleys that control the movements of the different phalanges. Each finger is comprised of three articulations with possibility of turn and an additional articulation that permits to reproduce the movement of abduction, besides serving of element of union of the digit with the palm of the hand. The pulleys (on articulations) are driven for a system of cables to way of human tendons. Each articulation possesses two tendons, one flexor and another extensor. The tendon flexor causes the movement of contraction of the articulations while the tendon extensor causes the contrary effect. The mechanical system of actuation arranges of a motor to extend and another to contract the tendon. For control the turn of each articulation, in a synchronized way, the wires remains traction in every instant, and is possible to measure the effort done by the tendons.

III. Factorization of Length and Tension

In this way, a FLETE (Factorization of Length and Tension) [8] is implemented at the hand interface in order to simultaneously achieve the position and rigidity control of each robot joint of the fingers. This biologically inspired neural model is the main tool to establish one upper hierarchy for control of the artificial opponent muscles, by measuring the status of the finger potentiometers and the tensions sensors for the tendons.

Two are the main inputs for the FLETE algorithm: the spatial target (in radians) and the strength parameters (in Nw.) for both agonist and antagonist artificial muscle, for each joint of the robot finger. These last parameters are supplied by the PCI-AER interface while the spatial target is generated by means of a VITE (Vector Integration to EndPoint) model. Due to the target evolution, VITE must be modulated by a temporal Go(t) signal, in order to solve the simultaneous differential equations implemented by the FLETE. The implementation of the VITE-FLETE algorithm has been carried out using the Matlab S-function tool and the engine application for Visual C++ and Matlab integration, as Figure 3 shows.

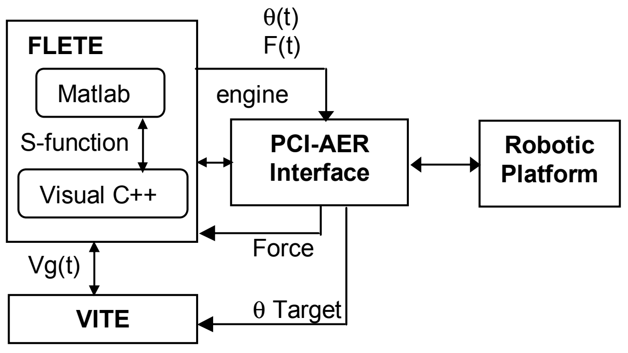

Figure 3. VITE-FLETE and Integration Interfaz.

In order to solve the differential equations for the VITE-FLETE model by means of the S-function blocks, several parameter have been defined, such as sigmoid function for Go(t) in VITE, or force gain, mass moment, extern force, stretch feedback gain, etc. for FLETE. By other side, the inputs to VITE-FLETE neural model are: simulation time (*t*) and step signal (*U*) for VITE, and initial position for the joints (*A*, from the interface), the neurons activity (*n*, from a cerebellar equation) and the target position for both tendons at each joint, modulated for Go(t) (*VG(t)* from VITE).

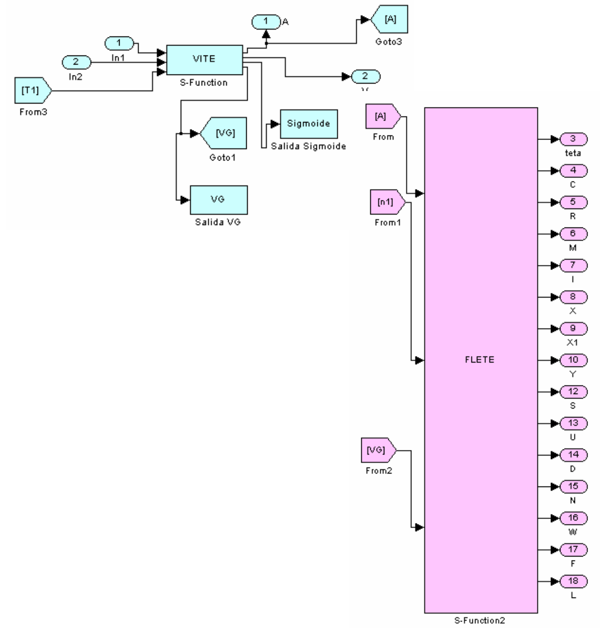

Figure 4. VITE-FLETE scheme

The output for VITE-FLETE model, -see Figure 4-, is given by a matrix containing the temporal evolution for each 18 variables obtained for the two artificial muscles in one joint: agonist and antagonist. The main output parameters are related to the incremental spatial position and the force exerted for each tendon, while other parameters measure aspects such as static gamma motoneurons activity, intrafusal dynamic gamma muscle contraction, Renshaw population output signal or Dynamic gamma muscle contraction.

Especially important for the satisfactory implementation of this neural model is the fast interaction between the robotic platform and the control software due to the AER protocol advantages.

IV. AER-Robot Interface

This section describes in detail an AER interface to manage actuators and to read analog commercial sensors and convert it to AER format. These actuators are based on DC motors.

The AER-Robot interface can control up to 16 up/down DC motors, each one doted with a two channel encoder. The DC motors are controlled digitally using Pulse Width Modulation (PWM). AER-Robot can read up to the following sensors: (a) 12 potentiometers for the finger articulations position, (b) 16 contact resistors for the fingertip and the palm object detection, (c) 16 tension sensor for the tendons of the fingers, and (d) 16 current sensor for the power consumption of the motors of the fingers. These sensors information are fundamental for the control algorithms in the hand platform.

The AER-Robot interface has been developed to communicate AER systems with an anthropomorphic robotic hand using two AER buses: one for incoming commands and another for outgoing information of the motors and the sensors. It is based around a Spartan 3 400 FPGA that allows co-processing. This FPGA receives commands through the input AER bus and sends motor and sensor information back. These commands allows to:

- Configure the PWM period that manages all the motors.
- Move a motor attending to PWM intensity and an estimated position through the encoder's information.
- Ask for a motor state.
- Ask for a sensor state.

Figure 5. shows the block diagram of the circuit of the FPGA, described into VHDL. This circuit is composed by several processes. CMDin receives commands and sends them to the corresponding process. There are 16 independent processes (Motor i) to control de PWM signal to be sent to each motor. There are four processes to attend the 64 possible sensors of the hand (16 potentiometers, 16 contact, 16 hall effect current and 16 tension tendon sensors). These last four processes attend to four Cygnal microcontrollers. Each of them is continuously converting an analog signal to digital from 16 possible analog inputs.

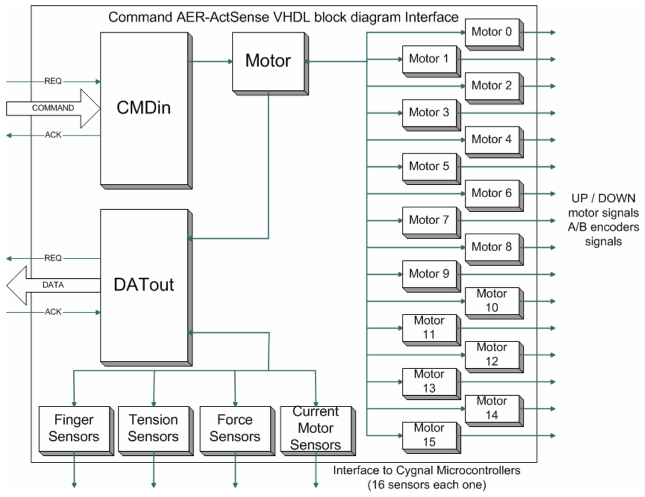

Figure 5. Circuit block diagram of the FPGA.

These processes work in parallel to allow real-time control of the hand; therefore the interface can receive new commands while it is executing another one. There is one last process for the AER output bus traffic control (DATout).

For each input command received by the AERin process, the order is sent to the corresponding motor process or sensor process, and then this AERin process is free to attend a new command.

Each motor process is in charge of one motor. If this process receives an order, its motor will go up or down, for a number of encoder pulses and with a programmed intensity.

There are four sensor processes. Each of them is asked for a value of one the sensors. Each process keep updated a 16-address internal RAM memory with the digital value of their sensors, and the digital value is sent to the AERout process immediately.

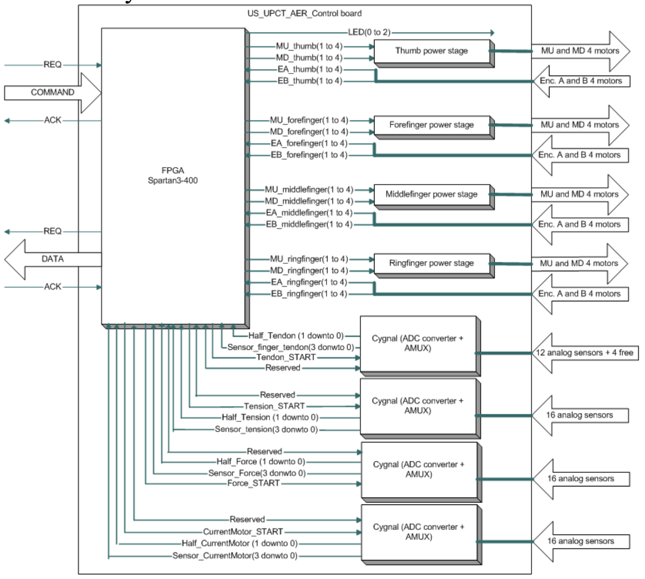

Figure 6. AER-Robot block diagram interface.

To keep this RAM-table updated, each sensor process communicates with a microcontroller, outside the FPGA, that scan the 16 analog output of the sensors, convert it into 8-bit and send it to the FPGA. The RAM-table is updated every 184μs. Thus, when a command is asking for the value of one sensor, the sensor process doesn't ask it to the microcontroller, but it just has to read it from the internal RAM memory of the FPGA (1 clock cycle).

The AER-Hand interface count with 4 microcontrollers. Each of them is in charge of 16 sensors of 4 different sets: articles potentiometers, fingertip and palm contacts, tendon tension and power consumption of the motors.

Figure 6. shows the block diagram of the interface. The real-time is warranted by the independent process architecture.

Figure 7. shows a photograph of the prototype of the AER-Robot Interface PCB. The digital part of the PCB is in the middle. The board has 16 power steps for the 16 motors, 16 10K x amplifiers for the tendon sensors, 16 hall effect sensors for the power consumption measurement of the motors, 4 Cygnal 80C51F320 microcontrollers for the analog to digital conversion (200Ksamples/second and 10-bits) of the sensor measurements, and all the connectors to the Hand.

This AER input bus and AER output bus is connected to a PC using the PCI-AER interface, explained in the next section.

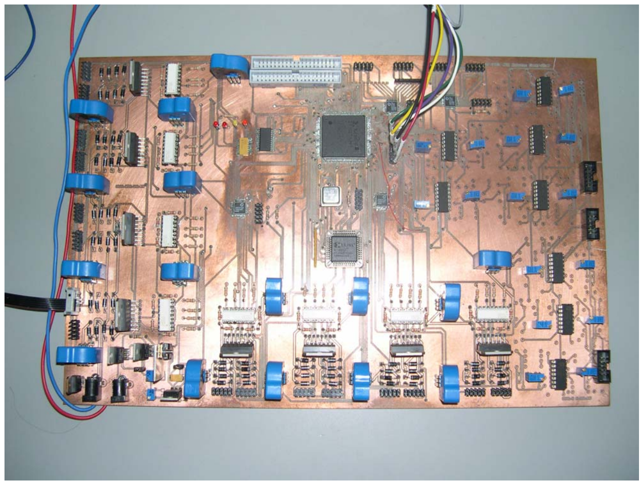

Figure 7. AER-Robot board photograph.

IV. PCI-AER INTERFACE

Before the development of this interface the only available PCI-AER interface board was developed by Dante at ISS-Rome [3]. This board is very interesting as it embeds all the requirements mentioned above: AER generation, remapping and monitoring. Anyhow its performance is limited to 1Mevent/s approximately. In realistic experiments software overheads reduce this value even further. In many cases these values are acceptable but, currently many address event chips can produce (or accept) much higher spike rates.

As the computer interfacing elements are mainly a monitoring and testing feature in many address event systems, the instruments used for these proposes should not delay the neuromorphic chips in the system. Thus, speed requirements are at least 10 times higher than those of the original PCI-AER board. Several alternatives are possible to meet these goals: (a) extended PCI buses, (b) bus mastering and (c) hardware based Frame to AER and AER to Frame conversion.

When the development of the CAVIAR PCI-AER board was started, using 64bit/66MHz PCI seemed an interesting alternative as computers with this type of buses were popular in the server market. When we had to make implementation decisions the situation had altered significantly. Machines with extended PCI buses had almost disappearing and, on the other hand, serial LVDS based PCI express was emerging clearly as the future standard but almost no commercial implementations were in the market. Therefore, the most feasible solution was to stay with the common PCI implementation (32 bit bus at 33MHz).

The previously available PCI-AER board uses polled I/O to transfer data to and from the board. This is possibly the main limiting factor on its performance. To increase PCI bus mastering is the only alternative. The hardware and driver architecture of a bus mastering capable board is significantly different, and more complex, than a polling or interrupt based implementation.

Hardware based frame to AER conversion doesn't increase PCI throughput but, instead, it reduces PCI traffic. First some important facts have to be explained. It is well known that some AER chips, especially grey level imagers where pulse density is proportional to the received light intensity, require a very large bandwidth. This is also the case of many other chips when they are not correctly tuned. For example let's consider a Grey level 128*128 imager with 256 grey levels. In a digital frame based uncompressed 25fps format, it would require a bandwidth of 128*128*25= 0.39MBytes/s. The maximum requirements for an "equivalent" system that would output AER supposing the number of events in a frame period is equal to the gray level and considering the worst case where all pixels spike with maximum rate is:

2bytes/event*256events/pixel*number of pixels/ frame period= 200MBytes/s

The meaning of this figure should be carefully considered. A well designed AER system, which produces events only when meaningful information is available, can be very efficient but, an AER monitoring system should be prepared to support the bandwidth levels that can be found in some real systems. These include systems that have not been designed carefully or that are under adjustment. Currently the available spike rates, even in these cases, are far from the value shown above but, some current AER chips may exceed the 40Mevents/s in extreme conditions.

The theoretical maximum PCI32/33 bandwidth is around 133Mbytes/s. This would allow for approximately 33Mevent/s considering 2 bytes per address and two bytes for timing information. Realistic figures in practice are closer to 15Mevents/s. Thus, in those cases where the required throughput is higher a possible solution is to transmit the received information by hardware based conversion to/from a frame based representation. Although this solution is adequate in many cases, there are circumstances where the developers want to know precisely the timing of each event, thus both alternatives should be preserved.

Implementing AER to Frame conversion is a relatively simple task as it basically requires counting the events over the frame period. Producing AER from a frame representation is not trivial and several conversion methods have been proposed [4].

The theoretical event distribution would be that where the number of events for a specific pixel is equal to its associated grey level and those events are equally distributed in time. The normalized mean distance from the theoretical pixel position in time to the resulting pixel timing with the different methods is an important comparison criterion. In [4] it is shown that, in most circumstances, the behavior of the methods is similar and, thus, hardware implementation complexity is an important selection criterion. From the hardware implementation viewpoint random, exhaustive and uniform methods are especially attractive.

As a result of these considerations the design and implementation of the CAVIAR PCI-AER board was developed including the bus mastering. The hardware based frame to AER conversion has been developed for another board under the project: the CAVIAR USB-AER board [6].

The physical layer has been implemented into VHDL for a FPGA. It was established that most of the functionality, demanded by the users, could be supported by the larger devices in the less expensive SPARTAN-II family. Figure 8. shows the CAVIAR PCI-AER board.

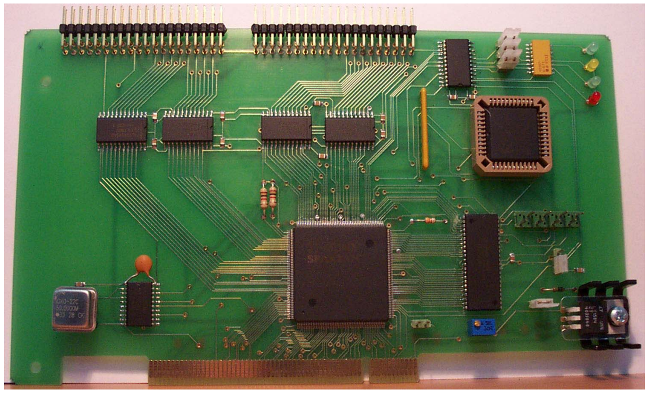
Figure 8. CAVIAR PCI-AER board

A Windows driver and an API that implements bus mastering and a Matlab interface are currently available. The Linux version with bus mastering is still under development.

The final goal is to transmit an AER sequence to an AER based system (for example a convolution chip) to perform video processing. An adequate sequence of events can be generated by software for testing an AER based system. This sequence of events needs to be sent to the AER based system. For this purpose it is necessary an interface between the computer and the AER bus. Figure 10 shows the VHDL architecture of the present hardware interface. This is a PCI interface developed under the European project CAVIAR. The interface, called CAVIAR PIC-AER, has two operation modes that can work in parallel:

A) From PCI to AER.

The AER-stream is stored in the computer memory and then it is sent to the AER system through the PCI bus and the OFIFO. This stream is saved in memory using 32 bits for each address event. The sixteen less significant bits represents the address of the pixel that is emitting the event. The sixteen more significant bits represent a time difference from the previous event in clock cycles. The clock cycle is 30 ns, but can be scaled up 16 times. Special words can be used in the OFIFO to make the state machine to wait the maximum time, coded with 16 bits, and then it reads a new word of the OFIFO without any event transmission. The OUT-AER state machine keeps continuously reading 32-bit words from OFIFO if it is enabled. For each word the state machine will wait for the configured number of clock cycles before transmitting the address through the AER output bus. If the acknowledge is delayed, the timer of the OUT-AER state machine will discount this time to the wait state of the next event. If the result of the discount is negative no wait will be done for the next event and this value will be used as initial wait for the following event. With this treatment the delay between events is not relative to the previous one, and a delay in the ACK reception will not cause a distortion in the time distribution of all the events along the time period.

B) From AER to PCI.

The AER sequence arrives to the CAVIAR PCI-AER interface through the input AER port. The AER-IN state machine stores the incoming event (16 bits LSbits) into the IFIFO with temporal information. This temporal information (16 bits MSbits) is the number of clock cycles since the last event.

The connection to the PCI bus is done by a VHDL bridge [12] that attends to the Plug & Play protocol of the PCI bus, decodes the access to the base address by the operating system, allows the bus mastering and the interruptions.

V. RESULTS

Both the hand and the AER interfaces have been connected to debug the FLETE algorithm developed under matlab.

The Antrophomorphic robotic hand was developed by NEUROCOR group from Cartagena (Spain). Under the Spanish grant project SAMANTA the hand has been connected to an AER system. One of the objectives of the project was to control the hand using AER vision systems together with other sensor information. Therefore the hand needs to be controlled under the AER protocol. In that project the visual information comes from an AER retina. Thanks to the PCI-AER interface the visual information is sent to a personal computer. A boundary-contour-system feature-control-system (BCS-FCS) algorithm for image processing was implemented under Matlab. With the visual processing results and the hand sensors information, the FLETE algorithm gives orders to the hand, for example to catch an object, by computing not only the visual information, but also the rigidity and fatigue of the muscles.

The results shown in table 1 are the ranges of the parameters that can be programmed in the AER-Robot board. These results have been measured under the following scenario: the AER-Robot interface has been connected to a PC through the PCI-AER interface. Therefore, commands where sent from matlab to the PCI-AER interface into AER format, and then they are sent to the AER-Robot interface.

	Freq min	***Freq max***	***High res.***	
PWM	763 Hz	25 MHz	8-bits	
	Dig. Res.	***Range (volts)***	***Kind***	***Sensor range***
Potentiometers	8-bit	0-3,3v	R	
Contact	8-bit	0-3,3v	R	
Tension	8-bit	2-2,5v	R	0,1mv/V
Hall-Effect current	8-bit	2-3,3v	L	
PCI-AER	***Max Th.***	***Pulse width***		
	6 Mev/s	120 ns		
AER-Robot	3 Mev/s	240 ns		

Table 1: Measured ranges for DC motor and Sensors.

Finally, several spatial and force consigns have been applied to the VITE-FLETE neural model from the interface, in order to measure the temporal evolution from the current value toward the indicated targets, which is generated in real time for FLETE. One result is shown in Figure 9. for only two joints (distal and medium) of one robot finger, where the targets were $\theta_d = 0{,}12$ rad., $\theta_m = 0{,}17$ rad., F_d=3,2 Nw. and F_d=5,3 Nw.

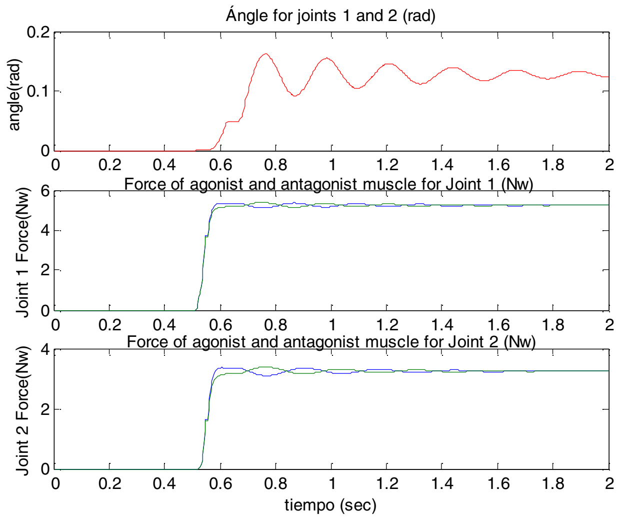

Figure 9. Force and espatial position ouputs form FLETE

V. Conclusions and Future Work

An AER to anthropomorphic robotic hand interface has been presented. This AER-Robot interface can be connected to a computer through a PCI-AER interface. The AER-Robot interface is based around a Spartan 3 FPGA that allows it to be configurable and easily modified for other robots based on DC motors, potentiometers sensors and tension sensors.

The AER neuro-inspired communication channel is connected with the robot. This implies a neuro-inspired control of the robot. This control is based on visual processing using AER retinas and convolution chips, and neuro-inspired FLETE algorithm in software.

The present state of the interface is able to receive and send 16-bit AER data. Coded under these 16-bit is placed the command or the sensor information. Therefore one event is enough to send a command to a motor or ask for a sensor information, then one or two events are sent back with the information required. The future work is focused on the spike based information. In such way, the motor PWM frequency will be sent translating the frequency of one address in the AER bus. Each address corresponds with one motor, and with one sensor in the other way. These VHDL improvements are under development. A more compact (6 motors and 16 analog sensors) and simple PCB is under design. This new board will allow connecting several AER-Robot interfaces in chain using the same AER bus. Therefore, in that case where more than 6 motors are needed to be controlled, a second board with a different address space can be connected to the first one and controlled from the PC or the AER system in the same way.

Acknowledgment

This work was in part supported by EU grant IST-2001-

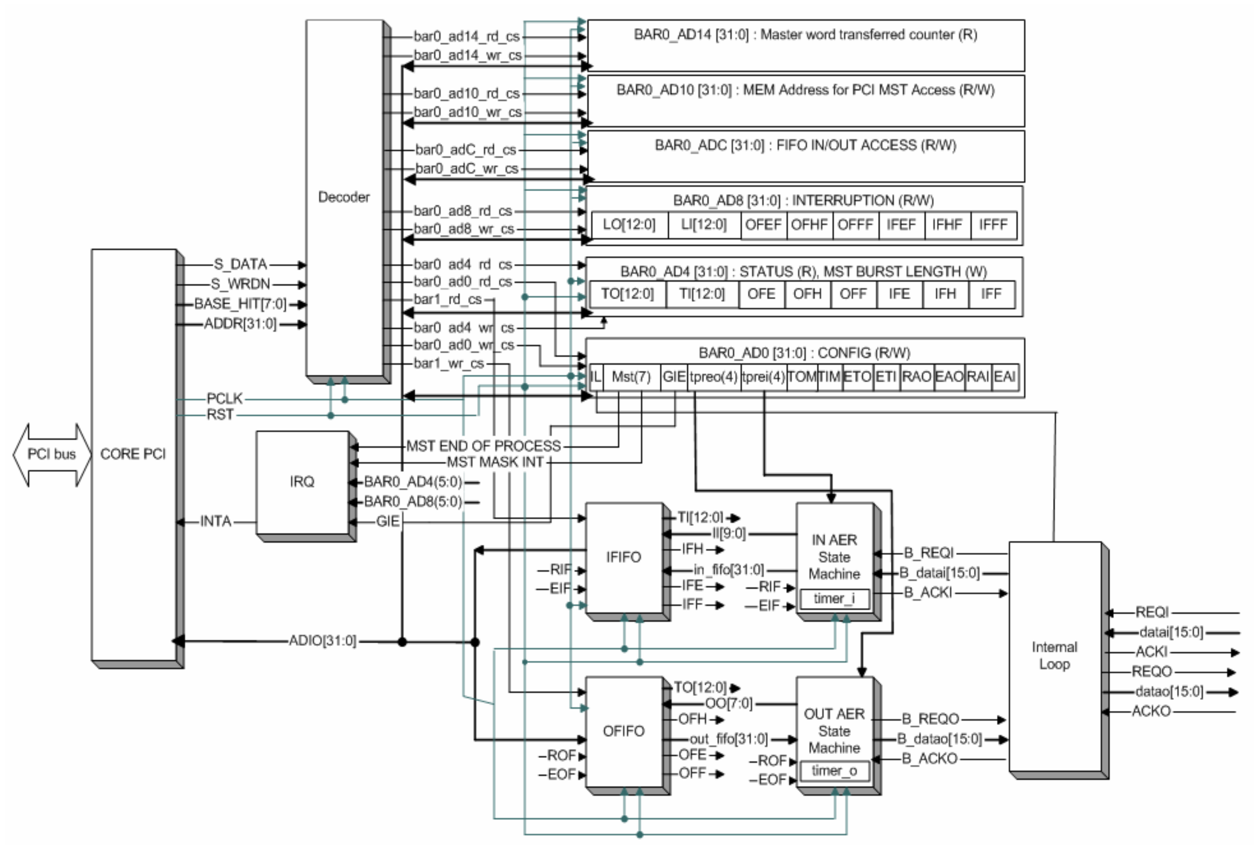

Figure 10. Hardware Interface Architecture.

34124 (CAVIAR), Spanish Science Fundation grants: TIC-2003-08164-C03-02 (SAMANTA) & TIC2000-0406-P4-05(VICTOR) and Plan de Ciencia y Tecnología Región Murcia 2I04SU043 (ROBOTEP)

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] A. Linares-Barranco, G. Jimenez-Moreno, A. Civit-Ballcels, and B. Linares-Barranco. "On Algorithmic Rate-Coded AER Generation". Accepted for publication on IEEE Transaction on Neural Networks.

[5] R. Paz. "Análisis del bus PCI. Desarrollo de puentes basados en FPGA para placas PCI". Trabajo de investigación para obtención de suficiencia investigadora. Sevilla, Junio 2003.

[6] R. Paz, F. Gomez-Rodriguez, M. A. Rodriguez, A. Linares-Barranco, G. Jimenez, A. Civit. Test Infrastructure for Address-Event-Representation Communications. International Work-Conference on Artificial Neural Networks. Vilanova I la Gertru. SPAIN. June-2005.

[7] J. López-Coronado, J.L. Pedreño-Molina, A. Guerrero-González, P. Gorce. A neural model for visual-tactile-motor integration in robotic reaching and grasping tasks. Robotica, Volume 20, Issue 01, pp. 23-21. Cambridge Press. (January 2002).

[8] D. Bullock, J. Contreras-Vidal, S. Grossberg. "Equilibria and dynamics of a neural network model for opponent muscle control". In G. Bekey, K. Goldberg (eds.): Neural Networks in Robotics Kluwer Academic.