A 20Mevps/32Mev event-based USB framework for Neuromorphic systems debugging

A. Rios-Navarro, J.P. Dominguez-Morales, R. Tapiador-Morales, D. Gutierrez-Galan, A. Jimenez-Fernandez and A. Linares-Barranco
Robotic and Technology of Computers Lab.
University of Seville. Seville, Spain
Email: arios@atc.us.es

***Abstract*— Neuromorphic systems are engineering solutions that take inspiration from biological neural systems. They use spike- or event-based representation and codification of the information. This codification allows performing complex computations, filters, classifications and learning in a pseudo-simultaneous way. Small incremental processing is done per event, which shows useful results with very low latencies. Therefore, developing this kind of systems requires the use of specialized tools for debugging and testing those flows of events. This paper presents a set of logic implementations for FPGA that assists on the development of event-based systems and their debugging. Address-Event-Representation (AER) is a communication protocol for transferring events/spikes between bio-inspired chips/systems. Real-time monitoring and sequencing, logging and playing back long sequences of events/spikes to and from memory; and several merging and splitting ports are the main requirements when developing these systems. These functionalities and implementations are explained and tested in this work. The logic has been evaluated in an Opal-Kelly XEM6010 acting as a daughter board for the AER-Node platform. It has a peak rate of 20Mevps when logging and a total of 32Mev of logging capacity on DDR when debugging an AER system in the AER-Node or a set of them connected in daisy chain.**

Keywords—AER; event-based systems; debugging tools; neuromorphic system; FPGA; VHDL; AER-Node; Opal Kelly.

I. INTRODUCTION

The AER protocol allows asynchronous event-based massive interconnection between silicon-neurons of neuromorphic processing chips or neuron models implemented in programmable hardware devices. This protocol was first proposed in Caltech, in 1991, to solve this problem between populations of neurons located in different chips by Mead Lab. [1]. It uses mixed analog and digital principles and exploits pulse density modulation for coding information and high speed digital communications for time-multiplex information. The state of the neurons is a continuous time varying analog signal. AER is inspired on the transmission of neural information in a biological neural system, by an electrical spike. In a two-chips AER communication the emitter contains an array of cells where each cell shows a state that changes with a slow time constant (in the order of milliseconds). Each cell includes an oscillator that generates pulses of minimum width (a few nanoseconds). Each time a cell generates a pulse (called "event"), it communicates with the periphery and its address is placed on the external digital bus (the AER bus), among additional info such as polarity or synaptic weight. Handshaking lines (Acknowledge and Request) are used for completing the communication. Then, in the receiver chip, the pulses are directed to the cell whose address was on the bus for each handshake cycle. The receiver cell processes the pulses at the time it is received, with a low latency respect to the time it was produced in the emitter. This is the main advantage of AER systems, because real-time properties are considerably improved. Therefore, there is no necessity for time codification or inclusion into the event's data. Since the computational charge assigned per event is very low respect to the inter-event-interval time, computation is distributed among the event's flow, and results are obtained as soon as the amount of sent events is enough to produce desired output events. In other words, the neuromorphic event-based processing is pseudo-instantaneous [2].

Nowadays, there exists a neuromorphic engineering community of AER systems developers for event-based applications in sensors (i.e. vision and auditory [3][4][5].), learning and robotic areas. Some of these systems have been developed to identify a human speaker [6], to estimate the distance of moving targets using AER stereo-vision [7] or to track multiple objects in parallel using cascade architecture of processing neurons [8]. In addition, there are some works that use both kind of sensors (vision and auditory) to develop bio-inspired sensory fusion systems [10].

To make easier the development of these systems, it is necessary to have a set of event-based tools:

- *Monitor*: Observe the output of the AER system in soft real-time through a computer and eventually store this in a file in the computer hard-disk
- *Sequencer*: Take a file with an AER stream previously stored and send it to the hardware interface to have a controlled input for an AER system.
- *Logger*: Save the output information of any sensor or system in local memory with the maximum possible speed (hard real-time). Then this information can be played back or downloaded to a computer file.
- *Player*: Produce a previously stored AER stream in local memory to act as the input for an AER system.
- *Merger*: Multiplex the output information from several sensors onto one AER bus.

There are a set of tools that perform and also mapping operations (transformation of the events flow on-the-fly), but they lack in capacity [11], bandwidth [12] or portability [13]. This paper presents a novel, compact, portable and modular hardware interface based on FPGA where all of these

This research is supported in part by Spanish grant BIOSENSE (TEC2012-37868C04-02) and Andalusian Council Excellence grant MINERVA (P12-TIC-1300), both with support from the European Regional Development Fund.

functionalities have been implemented under the same platform.

The paper is structured as follows: section II presents details of the hardware architecture; section III describes the proposed platform; finally, section IV shows the results and section V presents the conclusions.

II. Hardware Architecture

The previous list of functionalities to be covered by an AER tool requires a hardware platform which main characteristics have to be the reconfigurability, memory capacity and low latency. In this work we propose to improve the current functionality of AER general purpose platform AER-Node [14] with a new daughter board able to implement these functionalities.

The AER-Node board was designed in the context of a Spanish Government funded project, where the aim was to demonstrate that event-based processing under AER is feasible and convenient for high speed frame-free vision [15][17], filtering, processing and actuation, using spikes from vision sensors to DC motors. Under this project we also developed an event-based filter for object detection and tracking [8][9], a scalable mesh network multi-PCB technique of spike convolutions for cortex operation emulation [15], and spike based motor controllers (proportional-integral-derivative [21] and neuro-inspired open-loop VITE [22]). The AER-Node board was designed to allow multi-PCB communication with conventional parallel-handshaked AER chips (retinas and convolutions) or robots with the adequate motor interfaces. To achieve these requirements a Spartan6 XC6S1500FXT FPGA was selected, and a set of daughter boards were designed to increase its functionality (interface to convolution chips, to multiple DVS retinas, USB-computer, Controller-Area-Network, embedded computer and monitoring interfaces). Scalability is provided by four SATA connectors for bidirectional LVDS high-speed communications to enable a mesh of AER-Node boards [14]. Through two parallel 28-bit connectors and two 8-bit data connectors, functionality can be increased with proper daughter boards.

The new daughter board, called OKAERtool, is designed to add AER systems debugging capabilities to the AER-Node, which mainly allow to sequence and monitor a stream of events on the fly, to log and play-back to a from DDR memory; and to merge/split AER channels. The OKAERtool has four AER interfaces, three inputs (1 CAVIAR and 2 ROME connectors [17]) and one output (CAVIAR connector). In addition, it has two 8-bit general purpose connectors, which can be used to expand the width of the AER interfaces or for configuration purpose, i.e. through a Serial Peripheral Interface (SPI).

The OKAERtool is based on an Opal Kelly commercial platform. It can be connected to an AER-Node [14] for mesh networks [15] or it can also work independently. The specific OpalKelly model is the compact XEM6010-LX150 [16] that has the following components: Spartan 6 XC6SLX150-2FGG, 128 MiB DDR2 and a high-speed USB 2.0 interface (Cypress FX2LP - CY68013A). Fig. 1 shows the block diagram of the board.

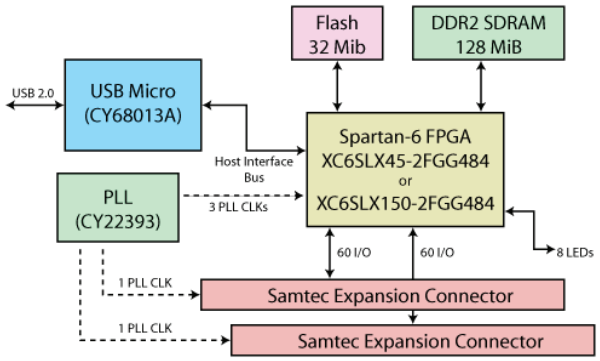

Fig. 1. Opal Kelly XEM6010-LX150 block diagram [16].

The expansion connectivity is through a two 80-pin 0.8mm Samtec board-to-board connectors. All this board functionality and power signals are accessible through these two connectors. Therefore, in order to be able to connect it to the AER-Node, a daughter board with additional CAVIAR and ROME connectors [17] has been developed. This daughter board has been designed to act as an independent board too. Therefore, the AER-Node is not needed for AER systems debugging. In Fig. 2 both, the Opal Kelly XEM6010-LX150 module and the AER-Node daughter board are shown.

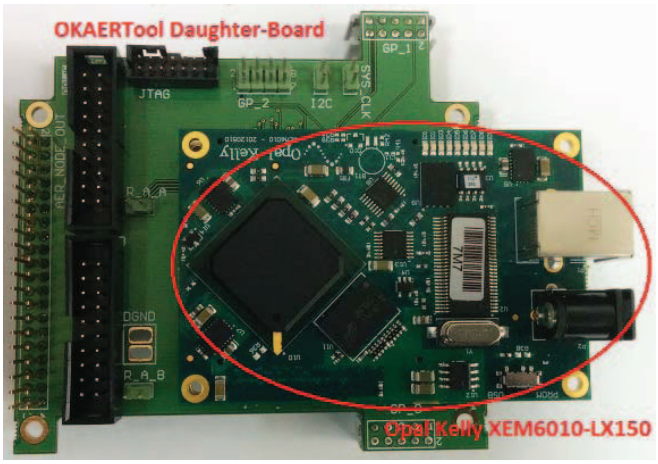

Fig. 2. OKAERTool daughter board for AER-Node (behind) and Opal Kelly XEM6010-LX150 (top) connected

III. OKAERtool Functionalities

As discussed in the previous section, a set of AER tools functionalities are gathered in the same platform and presented in this paper. They allow debugging and testing spike-based systems. Fig. 3 shows the block diagram of the logic in the FPGA. This logic has been described in VHDL in a modular way. A merger block is able to join two AER ROME connectors into one AER input bus. Then, this input AER bus can be selected or not by a multiplexor (*MUX* block) for logging or monitoring purposes by the *Logger* or *Monitor* blocks respectively. The mentioned multiplexor can select this merger output or the input AER bus, which can come from the output of the AER-Node board, for logging / monitoring tasks. For logging operations, all the AER received flow is stored in *DDR2* (accessible through the *DDR2 IPCore & Wrapper*

block), whilst for monitoring, the AER stream is sent to the host computer through the *Cypress FX2* microcontroller (connected to the logic by the *USB IPCore & Wrapper* block). On the other hand, the *OKAERtool* is also able to produce a stream of AER through the *Player* block. These events can come from two different sources: a set of previously stored events in *DDR2 memory*, or an events flow arriving through the *Cypress FX2* microcontroller. A state machine (inside the *Player* block) takes care of the AER handshaking for sending out the AER stream to the output CAVIAR parallel connector. This output can be connected to the input AER port of the AER-Node if the *OKAERtool* daughter board is acting as a daughter board; or this port can represent an output port to be connected to any other AER system.

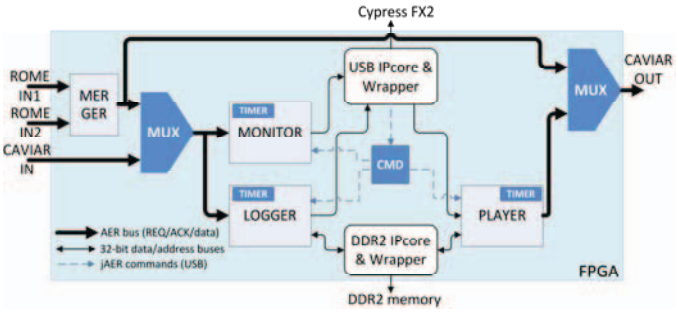

Fig. 3. OKAERtool logic block diagram.

Furthermore, thanks to the connection of the *Merger* output to the output CAVIAR port, the *OKAERtool* can work in parallel as an input gangway for an AER system running the AER-Node platform and, at the same time, it can work as a *Monitor* / *Logger* of that AER system for debugging its operation. Therefore, one or two sensors can be connected to the ROME connectors and, through the *Merger* block and the output CAVIAR MUX, they can represent the input of the AER-Node board. Then, the output of the AER-Node board can be connected to the *Monitor* or *Logger* block, through the input MUX, for debugging purposes.

Following subsections describe with more details each component of the logic.

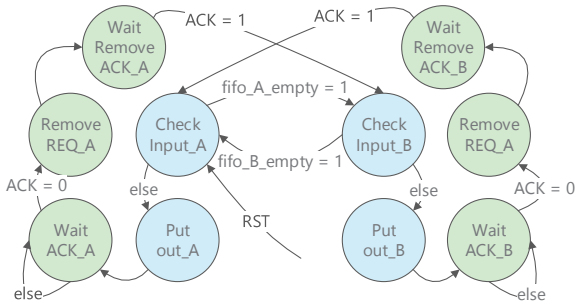

Fig. 4. Merger finite state machine: left side states manage the transmission of one event stored at FIFO-A, and right side is devoted to manage FIFO-B events. While both FIFOs have events, the state machine keep transmiting interlaced events of both FIFOs. Green states for AER handshake and blue implements the block operation.

A. Merger

This block is able to combine events from multiple sources into a single AER bus. Two external AER inputs are allowed (2:1 merger). Each input has a logic that includes the AER handshake controller and a small FIFO to store events and reduce stall situations probability. Then, an arbiter reads both FIFOs and multiplexes their outputs in time. Fig. 4 shows the merger finite state machine, which includes a procedure for fair arbitration. In case of collision, alternative turns between both FIFOs.

B. Monitor

This block can observe the output stream of an AER system. This observation is done by assigning a timestamp when an event arrives and then, by transmitting the couple address/timestamp to the USB interface. Monitors are used to observe the performance or right behavior of a neuromorphic system. This monitor functionality comprises several blocks in the block diagram of Fig. 3: the *Monitor* finite state machine, the USB interface to the FX2 uC (which is an IPCore from OpalKelly) and the command listener block (CMD). The monitor input can come from the Merger output or from the CAVIAR input connector. The maximum measured bandwidth is 5 Mevps. This limit is due to the bottleneck of the USB 2.0 host computer. Fig. 5 shows the monitor finite state machine. When the received command (through FX2 uC) is for monitoring, the state machine leave *"Idle"* state and get the *"Wait Event"* state in order to wait for an active *REQ* (active low). Once this REQ is received, the state *"Capture"* read both the input address and the timestamp (from its *Timer* block) and sends the ACK. "Capture" state is operated for only one clock cycle and the state machine reach the *"Wait Req"* state. This *"Wait Req"* state ensures the right handshake behaviour. It has a timeout mechanism when no events are received, making the state machine to come back to Idle state wait for a new command.

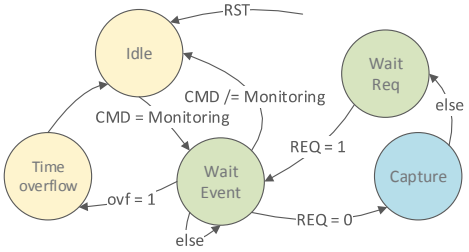

Fig. 5. Monitor finite state machine. Yellow states for commands suport, green states for AER handshake and blue implements the block operation.

C. Player

The use of sensors connected to an AER systems under debugging is not the correct method for ensuring a repetitive behavior of the under evaluation system. For this purpose it is necessary to be able to sequence a synthetically generated or a repetitive AER stream previously logged from a sensor. This tool lets us to download AER data packets from a computer, through USB 2.0 high speed bus, to onboard DDR2 memory, for later sequencing operation through the *AER_OUT* port. Since this downloaded AER stream includes time-stamps for each event, the tool has to reproduce events respecting such

temporal representation. The player functionality is divided in two states machines. First one, shown in Fig. 6, takes care of the DDR2 memory interface to and from FIFOs. These FIFOs are later managed by a second state machine that is sequencing these events out of the tool. First finite state machine has two different operation modes: (1) config mode (right side) for receiving a stream of events with timestamps from USB FX2 microcontroller to store them in DDR2 memory; and (2) operation mode (left side) for playing events according to their temporal waits represented by timestamps. Different USB commands allow to change between each modes.

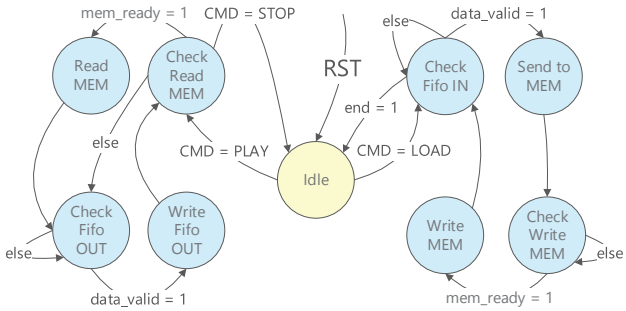

Fig. 6. Player DDR2 control finite state machine. Yellow states for commands suport and blue implements the block operation.

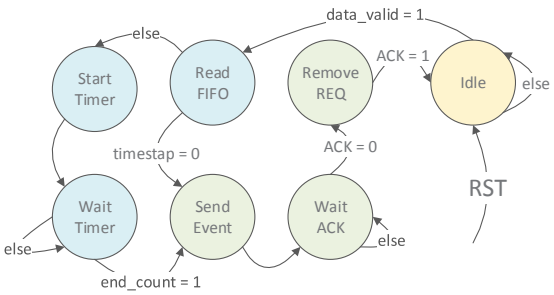

Fig. 7. Player sequencer finite state machine. Yellow state waits for Player FIFO data, green states for AER handshake and blue implements the block operation.

The sequencer part of the Player module of this OKAERtool sends out DDR2 events using their timestamp information. This timestamp represents the inter-event-time between two consecutive events. Therefore, an internal timer is used for waiting precise time before sending out the event through the *AER_OUT* port. This sequencer will compensate those possible receiver handshaking delays. Fig. 7 shows the sequencer finite state machine. It is waiting for next event coming from *Player* (data_valid=1) through a FIFO. Then it waits the timestamp using an internal timer, which is decremented until it underflows, before sending out the event.

D. Logger

This functionality is similar to the monitor one, but with the difference that the input events are time-stamped and stored on DDR2 memory instead of being sent through USB FX2 microcontroller in soft real-time. This allows a faster manipulation than the *Monitor* of a considerable amount of incoming events according to DDR2 capacity. These stored events can be later downloaded to a computer or sequenced by the *Player*. Maximum bandwidth of captured AER streams between *Monitor* and *Logger* is very relevant: 5Mevps for the monitor and 20Mevps for the logger. The capacity of DDR2 on board memory is 128 MB. If we use 16-bit events and 16-bit timestamps, it can be stored up to 32 Mevents, what makes possible to record several seconds of AER typical activity.

Fig. 8 shows the *Logger* finite state machine. It also has two different modes: (1) operation mode (right side), for capturing input events with their timestamps and store them in DDR2 memory; and (2) config mode (left side), for downloading DDR2 stored stream of events to the PC through USB 2.0 high speed interface of the FX2 microcontroller.

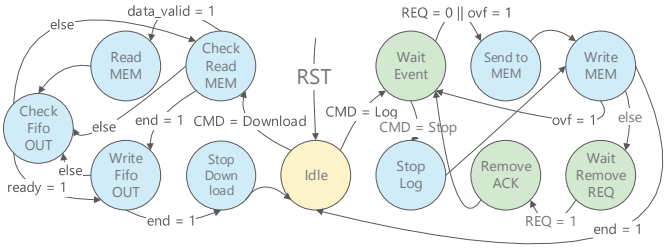

Fig. 8. Logger finite state machine. Yellow state waits for Player FIFO data, green states for AER handshake and blue implements the block operation.

The whole system is controlled by commands that are received through USB. These commands are interpreted by the *CMD* unit that is responsible for enabling different modules state machines. The two remaining blocks (*"USB IPcore / Wrapper"* and *"DDR2 IPcore & Wrapper"*) are provided by the manufacturer along with the Opal-Kelly platform. USB one for sending USB packets between the FPGA and the USB2.0 Cypress FX2 microcontroller; and DDR2 one is configured and obtained from Xilinx Core Generator tool for interfacing the FPGA to the on-board DDR2 memory. If needed, FSMs (figures 4-8) current states can be monitored through Chip-Scope tool available under ISE-Xilinx license. This tool operates through JTAG programming / debugging connector of the Opal-Kelly module. The system presented in this paper works with a 100MHz clock.

IV. Testing Scenarios and Performance

In this work we achieve faster and more compact AER tools than previous works [11][12][13]. TABLE I. shows a comparison between earlier USB AER interfaces (USBAER and USBAERmini2), and the presented interface. The OKAERtool interface is an "all-in-one" tool with better performances that supports wider event buses (16-bit and 32-bit event sizes are allowed). The whole debugging system has been successfully synthesized, implemented and tested on a Spartan 6 XC6SLX150 and also in a XC6SLX45, which is smaller, cheaper and it is provided by the Opal-Kelly module XEM6010-LX45.

The OKAERtool has x64 more RAM capacity for logging than the USBAER, a 33% increment on monitoring and 73% on sequencing bandwidth over the USBAERmini2 and a 66% increment of logging / playing bandwidth over the USBAER interface.

Fig. 9 shows a real experiment scenario where two neuromorphic sensors are joined and connected to an AER system in the AER-Node through the OKAERtool interface. This testing/debugging configuration has been successfully used in previous works like [18], where the experiment consists of estimating the motor speed at different values through a visual and audio neuromorphic sensory integration. Two event-based algorithms were proposed to estimate the speed (rpm) of the motor and they were implemented as filters under the open source project jAER [19]. The system used the DVS128 [20] (max 1Mevps) and a neuromorphic audio frequencies decomposer on FPGA (max 3Mevps). Its estimation accuracy was measured to be 94.33% with a tolerance of 10% with this OKAERtool.

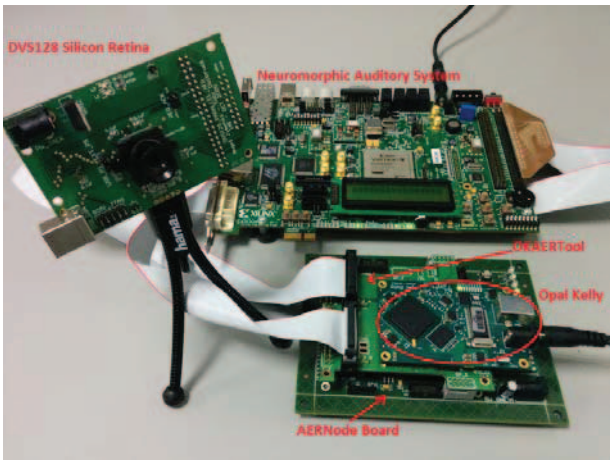

Fig. 9. DVS128 silicon retina (left) and neuromorphic auditory system (top). Complete neuromorphic processing system for sensory integration (bottom), composed of an AER-Node board, an OKAERtool daughter board on the middle and an Opal-Kelly XEM6010 on the top of stack.

The OKAERtool interface is fully integrated on the jAER software project, which is extensively used by the neuromorphic community. All possible commands can be sent from jAER to the OKAERtool through a filter GUI (see fig 10). Furthermore, a USB to SPI bridge is included in the logic of the FPGA in order to send configuration parameters to the AER systems synthesized in the AER-Node FPGA through 8-bit general purpose connectors. Finally, a MATLAB library is also available.

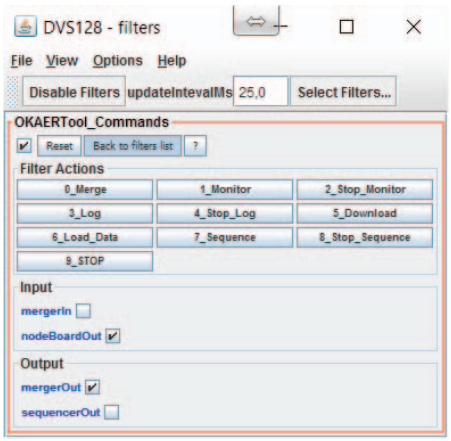

Fig. 10. jAER filter for OKAERtool management. All commands functionalities are activated through buttons. Input and output MUX are controlled also through this GUI.

V. CONCLUSION

A new AER tool has been presented in this paper which functionalities are faster in some cases and more flexible in others than previous tools. The system has been implemented on a compact commercial hardware platform. It can be connected to the AER-Node or it can work independently thanks to the expansion board developed.

ACKNOWLEDGMENTS

This research is supported by the Spanish grants (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02) and the Andalusian Council Excellence grant MINERVA (P12-TIC-1300).

TABLE I. COMPARISON TO PREVIOUS USB SYSTEMS

Comparison	Debugging USB AER Tools		
	USBAER mini2 [11]	***USBAER [12]***	***OKAERtool***
Capacity	*N/A*	2Mbytes	128Mbytes
Event width	16 bits	16 bits	16/32 bits
Monitor bandwidth	5Mevps (Peak 6Mevps)	*N/A*	6.5Mevps (Peak 8Meps)
Player/Logger bandwidth	*N/A*	3.75Mevps 12Mevps	6.5Mevps 20Mevps
Power (mA)	60mA	30mA	32mA

REFERENCES

[1] M. Sivilotti: Wiring Considerations in analog VLSI Systems with Application to Field-Programmable Networks. Ph.D. Thesis, Caltech (1991).

[2] C. Farabet, R. Paz, J. Perez-Carrasco, C. Zamarreño, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, B. Linares-Barranco. “Comparison Between Frame-Constrained Fix-Pixel-Value and Frame-Free Spiking-Dynamic-Pixel ConvNets for Visual Processing”. Frontiers in Neuroscience. Vol. 6. Num. 32. ISSN: 1662-453X. Year. 2012. DOI=10.3389/fnins.2012.00032

[3] 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(2008).

[4] J. A. Leñero-Bardallo, T. Serrano-Gotarredona and B. Linares-Barranco, "A 3.6µs latency asynchronous frame-free event-driven dynamic-vision-sensor," in IEEE Journal of Solid-State Circuits, vol. 46, No. 6, pp. 1443-55, June, 2011

[5] S.-C. Liu, A. van Schaik, B. Minch, and T. Delbruck,"Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," Proceedings of IEEE International Circuits and Systems, 2010, pp. 2027-2030.

[6] Li, Cheng-Han; Delbruck, T.; Liu, Shih-Chii, "Real-time speaker identification using the AEREAR2 event-based silicon cochlea," *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on* , vol., no., pp.1159,1162, 20-23 May 2012.

[7] Domínguez-Morales, M.; Jimenez-Fernandez, A.; Paz-Vicente, R.; Jiménez, G.; Linares-Barranco, A., "Live demonstration: On the distance estimation of moving targets with a Stereo-Vision AER system," *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on* , vol., no., pp.721,725, 20-23 May 2012.

[8] Gómez-Rodríguez, F. et al, "Live demonstration: Real time objects tracking using a bio-inspired processing cascade architecture," *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on* , vol., no., pp.1398,1398, May 30 2010-June 2 2010.

[9] A. Linares-Barranco, F. Gómez-Rodríguez, V. Villanueva, L. Longinotti and T. Delbrück, "A USB3.0 FPGA event-based filtering and tracking framework for dynamic vision sensors," Circuits and Systems (ISCAS), 2015 IEEE International Symposium on, Lisbon, 2015, pp. 2417-2420.

[10] P. O'Connor et al, "Real-time classification and sensor fusion with a spiking deep belief network", Frontiers in Neuroscience, Vol. 7, 2013, Number 178.

[11] Berner, R.; Delbruck, T.; Civit-Balcells, A.; Linares-Barranco, A., "A 5 Meps $100 USB2.0 Address-Event Monitor-Sequencer Interface," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on* , vol., no., pp.2451,2454, 27-30 May 2007.

[12] Linares-Barranco, A.; Paz, R. et al, "Neuro-inspired real-time USB & PCI to AER interfaces for vision processing," in Performance Evaluation of Computer and Telecommunication Systems, 2008. SPECTS 2008. International Symposium on , vol., no., pp.330-337, 16-18 June 2008

[13] Fasnacht, D.B.; Indiveri, G., "A PCI based high-fanout AER mapper with 2 GiB RAM look-up table, 0.8 µs latency and 66MHz output event-rate," in Information Sciences and Systems (CISS), 2011 45th Annual Conference on , vol., no., pp.1-6, 23-25 March 2011

[14] Iakymchuk, et al. An AER handshake-less modular infrastructure PCB with x8 2.5 Gbps LVDS serial links. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on* (pp. 1556-1559). IEEE.

[15] Zamarreno-Ramos, C et al, "Multicasting Mesh AER: A Scalable Assembly Approach for Reconfigurable Neuromorphic Structured AER Systems. Application to ConvNets," in Biomedical Circuits and Systems, IEEE Transactions on , vol.7, no.1, pp.82-102, Feb. 2013

[16] Opal Kelly XEM6010-LX150 board. Avaible on https://www.opalkelly.com/products/xem6010/

[17] R. Serrano-Gotarredona, et al., "CAVIAR: A 45k Neuron, 5M Synapse,12G Connects/s AER Hardware Sensory–Processing–Learning–Actuating System for High-Speed Visual Object Recognition and Tracking," Neural Networks, IEEE Transactions on , vol.20, no.9, pp.1417,1438, Sept. 2009

[18] A. Rios-Navarro et al. "Live-Demonstration: Real-Time Motor Rotation Frequency Detection by Spike-Based Visual and Auditory Sensory Fusion on AER and FPGA". *Artificial Neural Networks and Maching Learning (ICANN)*. 2014 (pp 847-849).

[19] jAER Open-Source Software Project. Avaible online http://jaer.wiki.sourceforge.net/

[20] 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(2008).

[21] Jimenez-Fernandez, A.; Jimenez-Moreno, G.; Linares-Barranco, A.; Dominguez-Morales, M.J.; Paz-Vicente, R.; Civit-Balcells, A. A Neuro-Inspired Spike-Based PID Motor Controller for Multi-Motor Robots with Low Cost FPGAs. Sensors 2012, 12, 3831-3856.

[22] Perez-Peña, F.; Morgado-Estevez, A.; Linares-Barranco, A.; Jimenez-Fernandez, A.; Gomez-Rodriguez, F.; Jimenez-Moreno, G.; Lopez-Coronado, J. Neuro-Inspired Spike-Based Motion: From Dynamic Vision Sensor to Robot Motor Open-Loop Control through Spike-VITE. Sensors 2013, 13, 15805-15832.