ConvNets Experiments on SpiNNaker

T. Serrano-Gotarredona, B. Linares-Barranco
Instituto de Microelectrónica de Sevilla
(IMSE-CNM)-CSIC and Univ. de Sevilla, Sevilla, Spain
{terese, bernabe}@imse-cnm.csic.es

F. Galluppi[2], L. Plana[1], S. Furber[1]
[1]Dept. Comp. Science, University of Manchester, UK
[2]Institut de la Vision, Univ. Pierre et Marie Curie, Paris, France.

***Abstract*—The SpiNNaker Hardware platform allows emulating generic neural network topologies, where each neuron-to-neuron connection is defined by an independent synaptic weight. Consequently, weight storage requires an important amount of memory in the case of generic neural network topologies. This is solved in SpiNNaker by encapsulating with each SpiNNaker chip (which includes 18 ARM cores) a 128MB DRAM chip within the same package. However, ConvNets (Convolutional Neural Network) posses "weight sharing" property, so that many neuron-to-neuron connections share the same weight value. Therefore, a very reduced amount of memory is required to define all synaptic weights, which can be stored on local SRAM DTCM (data-tightly-coupled-memory) at each ARM core. This way, DRAM can be used extensively to store traffic data for off-line analyses. We show an implementation of a 5-layer ConvNet for symbol recognition. Symbols are obtained with a DVS camera. Neurons in the ConvNet operate in an event-driven fashion, and synapses operate instantly. With this approach it was possible to allocate up to 2048 neurons per ARM core, or equivalently 32k neurons per SpiNNaker chip.**

Keywords—Convolutional Neural Networks, SpiNNaker Platform, Event-driven Computation, Object Recognition

I. Introduction

Convolutional Neural Networks (ConvNets) are biologically inspired generic architectures for intelligent data processing [1]. The ConvNet architecture is inspired in the structure of layers of the biological visual cortex, where the cortex is composed of several layers of processing neurons. Each neuron in a layer is connected to a neighborhood of neurons in the next layer implementing a projection field, and the connectivity pattern is quite homogeneous within the neurons located in the same layer. This gives rise to the "weight sharing" property.

However, conventional ConvNets operate in a frame-based fashion. That is, the sensed data ('image' or 'frame' in case of vision processing) is acquired at a given *frame rate* (normally in the order of 30-40ms) and each acquired image is processed sequentially by the different convolution layers. In that way, the operation of each ConvNet layer must be completely finished before the processing of the next ConvNet layer can be started. But, this is not what happens in biological brains. Biological neurons operate with asynchronous spikes. When a pixel in the retina has relevant information to transmit, it emits a spike which is delivered with just the propagation delay of the synaptic connections to the neurons of the next layers. When a neuron in the next layer receives enough spikes to reach some threshold it would emit another spike to the projection neurons of the next layer. That way, relevant information is processed in real time up to the higher recognition layer levels.

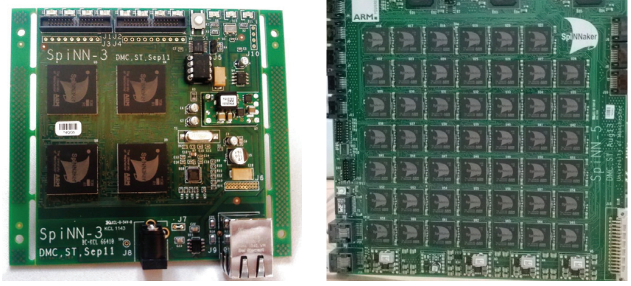
In this work, we have been able to implement a fully event-driven 5-layer ConvNet recognition system using the generic bioinspired highly parallel fully programmable SpiNNaker platform [2]. At the same time, we have made several software optimizations in the SpiNNaker platform to speed-up the processing and optimize the number of convolutional neurons that can be allocated. SpiNNaker is a versatile platform for many other experiments [3][4].

II. Description of the SpiNNaker Platform

The SpiNNaker platform is a bioinspired highly parallel hardware based on event-driven data communication and computation [2]. This platform is one of the basic computation hardware platforms brought into the EU Flagship *Human Brain Project* (HBP) and the ultimate goal of SpiNNaker is to emulate 1% of the human brain capacity [5].

Figure 1 shows two of the five developed generations of SpiNNaker boards. Figure 1a shows the spiNN-3 board hosting 4 SpiNNaker chips (the one that we have used in the present work) and Figure 1b shows the most advanced spiNN-5 board hosting 48 SpiNNaker chips. About 1,200 spiNN-5 boards would eventually be used to assemble a 1-million-core SpiNNaker system. Each SpiNNaker chip contains 18 ARM cores with a local 32kB of ITCM (instruction tightly coupled RAM) to store instructions and 64kB of DTCM (data tightly coupled RAM) to store locally neuron states and parameters. Read/Write operations on these memories need only one 200MHz clock cycle. Additionally, each SpiNNaker chip package contains a commercial 128MB SDRAM shared by the 18 ARM cores through a direct memory access (DMA) controller per core. This external in-package memory is intended to store all the synaptic connectivity weights.

When implementing a neural network on SpiNNaker, there are several abstraction levels of software system descriptions as represented in Figure 2(a). At a higher level, the user can describe the system in the high-level language PyNN [6] using the neuron models and synaptic connectors supported (see Figure 2b) in the (expandable) model library. These are described in C language, to be compiled and downloaded for


(a) (b)
Figure 1. (a) 4 chip SpiNNaker board, (b) 48 chip SpiNNaker board

This work has been supported by the Salvador Madariaga Mobility Program of the Spanish Ministry of Education, Culture and Sports under grant numbers PRX12/00562 and PRX12/00558, by the EU HBP project under grant number FP7-ICT-2013-FET-F-604102, by the ERANET grant PRI-PIMCHI-2011-0768 (PNEUMA) funded by the Spanish Ministry of Economy and Competivity, by Spanish research grant (with support from the European Regional Development Fund) TEC2012-37868-C04-01 (BIOSENSE), and by Andalusian research grant TIC-2010-6091 (NANONEURO)

execution on the SpiNNaker cores. At the intermediate level, the PACMAN tool (partition and configuration manager) [7] distributes the neuron populations among the different cores, downloads routing tables, neuron states, synaptic weights, …

In its original configuration, PACMAN stores states of the neurons allocated in each core in the corresponding internal DTCM. However, as the number of synaptic connections is much higher than the number of neurons, the synaptic routing information for each neuron and the corresponding weights are stored in the in-package SDRAM shared by all the cores. Each time a synaptic event arrives at a neuron, a DMA process is initiated to read the corresponding weight from the SDRAM. The synaptic events received by each neuron are accumulated in an internal ring buffer per neuron. Neuron states are updated at fixed time-steps (1ms), controlled by a timer event [2].

In section IV, we will explain the modifications we have done to PACMAN to exploit the resources available in the SpiNNaker chip in a time-step-less way, adapted to the peculiarities of pure event-driven ConvNet processing.

	ConvNet Structure					
	C1	***S2***	***C3***	***S4***	***C5***	***C6***
Feature Maps (FM)	***6***	***6***	***4***	***4***	***8***	***4***
FM Size	***28x28***	***14x14***	***10x10***	***5x5***	***1x1***	***1x1***
Kernels size	***10x10***	-	***5x5***	-	***5x5***	***1x1***
kernels	***6***	-	***24***	-	***32***	***32***
weights	***600***	-	***600***	-	***800***	***32***
Trainable weights	***0***	-	***600***	-	***800***	***32***

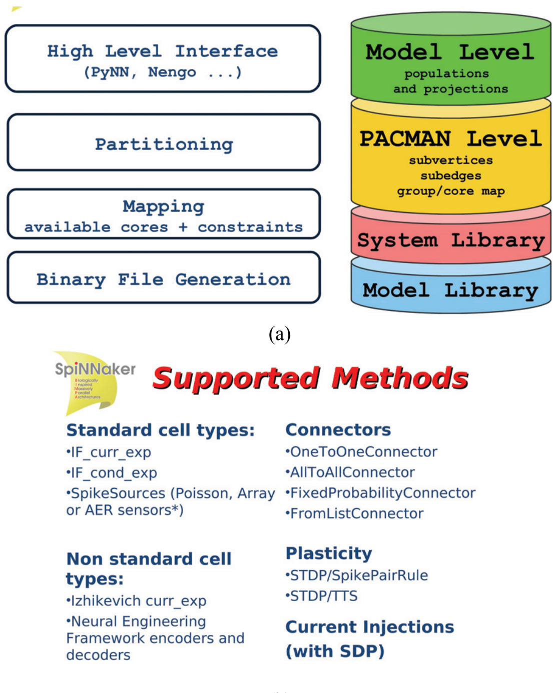
Table 1. ConvNet Architecture for card symbols recognition

III. EVENT-DRIVEN CONVNET ARCHITECTURE FOR POKER CARD SYMBOL RECOGNITION

A generic n-layer architecture of a ConvNet for visual object recognition is depicted in Figure 3a. The dynamic visual scene coming out of the retina is fed to a sequence of feed forward layers. Each layer consists of the parallel application of 2D-filters to extract visual features. Each representation obtained is called a *feature map*. The first layer extracts oriented edges according to different angles and different spatial scales. Subsequent layers combine features extracted at previous layers trying to detect progressively more complex shapes and figures, until achieving recognition of complex objects/symbols in the last layer. Along the layers, size of feature maps is progressively reduced through subsampling. This subsampling process is intended to introduce invariance to object size and position [1].

The particular ConvNet architecture implemented in SpiNNaker for poker card symbol classification is shown in Figure 3b. This event-driven ConvNet has been reported before, and tested on a custom software simulator [8]. It consists of four convolution layers (C1-C3-C5-C6) interleaved with two subsampling stages (S2-S4). Table 1 details number and size of the feature maps in each layer as well as the number of kernels and kernel sizes defining each layer synaptic connections. From here, the total number of weights defining each layer can be deduced. The first convolution layer implements Gabor filters at 3 different orientations and 2 spatial scales and their weights are not trained. The rest of the network weights were obtained by backpropagation, training a frame-driven ConvNet version and afterwards applying a conversion method to obtain the corresponding parameters for the event-driven ConvNet [8]. In an event-driven ConvNet there are some additional timing parameters (refractory time and leakage rate for each layer) that depend on the dynamics of the events generated by each layer [8]. Timing parameters were optimized afterwards for maximum recognition performance. We used a very high speed recording from a 128x128 resolution DVS [9] sensor, observing the browsing of a full poker card deck in about 1 second (10-30ms per poker card). This produced about 10Meps (million events per second), which is about a factor 100 of typical sensor output event traffic for normal visual scenes.

For the recorded sensor output events, we used an event-driven clustering algorithm [10] to track the passing symbols and at the same time we adjusted the tracking area to a 32x32 pixel window. Each symbol appeared during about 10-30ms producing 3k-6k events, thus providing a peak event rate of 0.6Meps. The achieved recognition success rate varied between 90.1-91.6% [8]. DVS cameras also have been shown to be very



Supported Methods

Standard cell types:
•IF_curr_exp
•IF_cond_exp
•SpikeSources (Poisson, Array or AER sensors*)

Non standard cell types:
•Izhikevich curr_exp
•Neural Engineering Framework encoders and decoders

Connectors
•OneToOneConnector
•AllToAllConnector
•FixedProbabilityConnector
•FromListConnector

Plasticity
•STDP/SpikePairRule
•STDP/TTS

Current Injections (with SDP)

Figure 2. (a) Description levels of a neural system on SpiNNaker, and (b) supported neural models and connectors

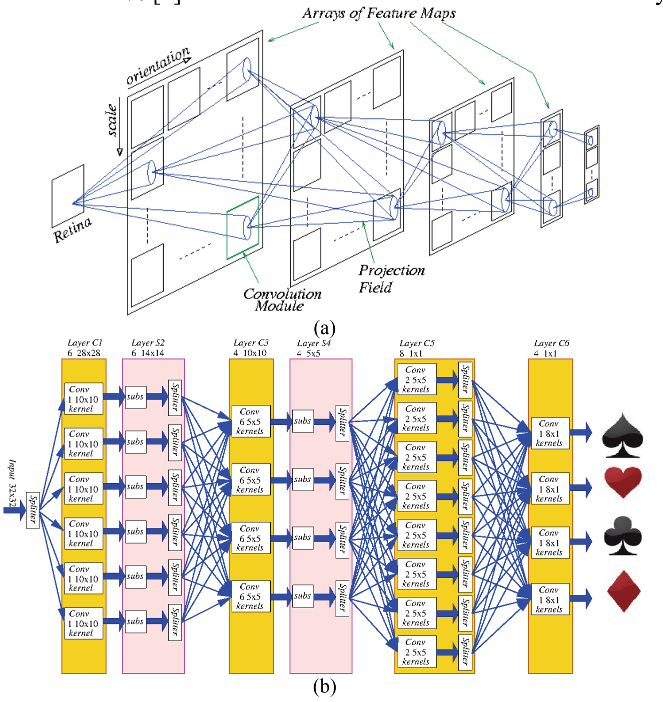


Figure 3. (a) 6-Layer Architecture of a Convolutional Neural Network, and (b) particular architecture designed for symbol cards recognition

efficient for other high speed experiments [11],[12].

IV. Event-Driven ConvNet on SpiNNaker

One of the peculiarities of ConvNet architectures is the "weight sharing" property. The weights of the kernels that connect neurons in two consecutive feature maps do not depend on the particular neuron positions but just on the relative positions of the two neurons in the origin and destination feature maps. Because of that "weight sharing" property the number of synaptic weights that must be stored for a neuronal population is very reduced compared to the case of populations with full connectivity and independent non-shared weights. To optimize the processing speed, the original PACMAN has been modified to admit a special "convolution connector". The "convolution connector" is shared by all the neurons belonging to the same convolutional feature map population and contains the kernel weights which are stored in the local DTCM of the corresponding population. This solution avoids the DMA access of the kernel weights from the external in-package SDRAM each time an event arrives at the convolution module. Each time an event arrives at a convolution module, depending on the source population of the incoming event, the corresponding kernel is read from its DTCM memory (see Figure 4) and the neuron states of the neighbour pixels are updated correspondingly. If any of the updated neurons overcomes the firing threshold, an output event is generated and sent to the next processing layer without waiting for any timer, contrary to how it is done in the conventional implementation of neural systems on SpiNNaker. That way, the implemented ConvNet is truly event-driven and fully independent of the standard 1ms state update time step.

Another characteristic of ConvNets is that most of the neuron parameters (as neuron threshold voltage, reset voltage, leakage rate as well as refractory time) are shared by all the neurons in the same population. Only the particular neuron state and firing times are individual for each neuron. In the standard PACMAN version, all neuron parameters are replicated and stored individually for each neuron in DTCM. Thus the DTCM capacity sometimes limits the number of neurons that can be implemented per core. PACMAN has also been modified to distinguish between the parameters that are individual for each neuron and the parameters shared by all the population. With this approximation, we are able to implement 2048 convolution neurons per core. This number is determined by the maximum number of addressable neurons, limited by the implemented routing scheme. Therefore, if future releases of PACMAN allow for a more flexible neural addressing routing scheme, it could be possible to further increase this limit.

V. EXPERIMENTAL SET-UP AND RESULTS

The event-driven ConvNet architecture with optimized parameters to perform card symbol recognition [8] has been programmed on a 4-chip SpiNNaker board. To test the recognition success rate we have used a test sequence of 40 32x32 tracked symbols obtained from a very high speed recording with our high sensitivity DVS [9]. Although our DVS has a contrast sensitivity of down to 1.5%, we had to adjust the sensitivity to the 20% range so that its output event rate would not exceed 10Meps, which is the sensor's maximum peak communication bandwidth [9].

We loaded the recorded event sequence on a data player board [13]. The data player board stores the addresses and timestamps of the recorded events in a local memory and reproduces the events through a parallel AER link in real time (see Figure 4). A splitter board [13], replicates this event-flow on two separate AER links. One of them enters a commercial FPGA prototyping Raggestone Board. The Raggestone board is programmed to convert the parallel AER events coming into its input connector to the SpiNNaker input accepting events coded in a very low power 2-of-7 asynchronous protocol [14]. In a similar way, the board converts the output events generated by the SpiNNaker hardware to the parallel AER protocol and emits them through a separate AER output link. The output of the SpiNNaker board coming out of the Raggestone board is combined with the second replica of the input stimulus (provided by the splitter board) into one single AER event flow using a merger board [13]. The merger board adds one extra bit to the output AER (merged) flow to encode the input port. That way, we have the input stimulus synchronized with the Spinnaker classifier output. As can be observed in Figure 4, the events coming out of the merger board are fed into a jAER Board [13] (usually known as USBAERmini2 PCB) which timestamps the arriving events and sends them to the computer through USB-2.0 for use by the jAER java software environment [15].

We first tested the correct functionality of the card symbol classification ConvNet at slowed down speed. The original recording shows the symbols at a speed of about 40 symbols per second. For this experiment, we multiplied by a factor 100 the timestamps of all the events of the recorded test sequence. This also implies to multiply by the same 100 factor all timing parameters (refractory time and leakage time) of the ConvNet architecture.

During the interval of each input symbol appearance, we counted the number of output events generated by each output category. The classification is considered successful if the number of output events of the correct category is the maximum one. Figure 5(a) plots (with red crosses) the classification performed by the system with a slow-down factor of 100. With solid blue trace, we have marked the input category (0 to 3) of each symbol. The presentation of the whole test sequence lasts approximately 120s; each symbol is presented during approximately 3s. Red crosses indicate the output category events generated by the SpiNNaker classifier. A total of 189 category output events are generated, and the recognition success rate is 97.5.

Once we tested the correct SpiNNaker ConvNet classifier functionality, we tested the maximum input event flow it could handle correctly. For that purpose, we repeated the experiment using a slow-down factor of 10 for the input stimulus event sequence. Fig. 5b shows the classification output (with red crosses) of the same test sequence for the slow-down factor of

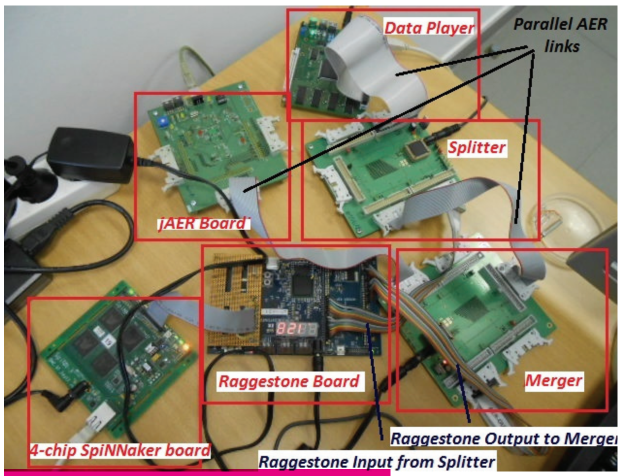
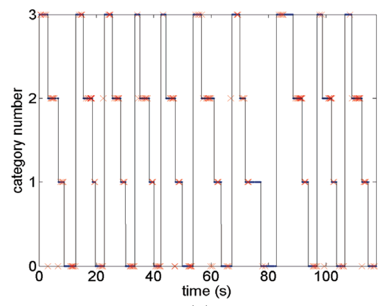


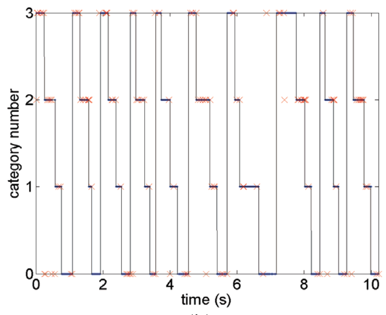
Figure 4. Experimental measurement set-up

10. In this case, the success recognition rate decreases to 80% and the total number of output events generated by the SpiNNaker ConvNet has also decreased to 121 events. Figure 5c plots the classification results for no slow-down factor. In this case, the number of output events has dramatically decreased to 8 events, and the recognition success rate is just 2.5%.

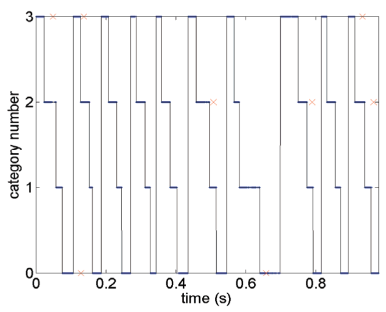
The main bottleneck limiting maximum event throughput is the number of events produced by the first ConvNet layer, programmed with Gabor filters. Each Gabor filter receives all the events from the input stimulus, while generating each a significant number of events, which need to be transmitted and processed by the second layer. Also, we tried to maximize the number of neurons that can be allocated per core, achieving 2000 neurons per core. This has the down side that each core has to process the input events arriving to all these 2048 neurons, which ends up being the processing bottleneck of the overall system. The SpiNNaker architecture is very flexible and it allows to trade off maximum number of neurons per core versus maximum event processing throughput. By eventually reducing the maximum number of neurons per core, one effectively achieves parallelization of event processing, and the system would be capable of handling higher input event rates. Thus the trade-off between maximum neurons per core and maximum event throughput should be optimized according to the desired environment.



(a)



(b)



(c)

Figure 5. (a) Solid blue trace indicates the category (0 to 3) of the input card symbol for a presentation of a 40-card test sequence. The red crosses mark the category output events generated by the SpiNNaker ConvNet. (a) Categorization with a slow-down factor of 100, (b) for a slow-down factor of 10, and (c) without slow-down factor.

VI. CONCLUSIONS

We have implemented an event-driven ConvNet using the SpiNNaker hardware. In order to maximize the size of the neuron populations that can be implemented in each core, we have done some modifications for the synaptic storage exploiting the "weight sharing" property of ConvNets. Also, a truly asynchronous event-driven convolution neuron has been programmed. We have successfully implemented a card symbol classifier on a 4-chip SpiNNaker board achieving an 80% success recognition rate with card symbols passing at approximately 4 symbols/second. In the future this can be combined with a real time tracker [16], so that the recognition system can be directly interfaced to the DVS sensor.

REFERENCES

[1] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Science and Neural Networks*, M. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 255–258.

[2] Steve B. Furber, et al., "Overview of the SpiNNaker system architecture," *IEEE Trans. on Computers*, vol. 62, no. 12, pp. 2454-2467, Dec. 2013.

[3] G. Orchard et al. "Real-time Event-driven SNN for OR on the SpiNNaker platform," ISCAS 2015, Special Session on Real-Time Event-based Sensor Proc.

[4] P. Klein et al. "Scene Stitching with ED Sensors on a Robot Head Platform," ISCAS 2015, Special Session on Real-Time Event-based Sensor Proc.

[5] https://www.humanbrainproject.eu/es

[6] A. P. Davison et al., "PyNN: a common interface for neuronal network simulators," *Front. Neuroinformatics* 2, 2008

[7] F. Galluppi, et al., "A Hierarchical Configuration System for a Massively Parallel Neural Hardware Platform," in *ACM Int. Conf. on Comp. Front.*, 2012.

[8] J. A. Pérez-Carrasco, et al., "Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate-Coding and Coincidence Processing. Application to Feed-Forward ConvNets," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 35, No. 11, pp. 2706-2719, Nov. 2013.

[9] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128x128 1.5% contrast sensitivity 0.9% FPN 3µs latency 4mW asynchronous frame-free dynamic visión sensor using transimpedance amplifiers," *IEEE J. .Solid-State Circ.*, vol. 48, no. 3, pp. 827-838, March 2013.

[10] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic procedural system," Proc. of the *IEEE Int. Symp. CAS,* 2007, pp. 845-848.

[11] T. Delbruck et al., "Human vs Comp Slot Car Racing using an Event and Frame-based DAVIS Vision Sensor," ISCAS 2015, Special Session on Real-Time Event-based Sensor Proc.

[12] J. Conradt, "A pencil balancing robot using a pair of AER DVS," ISCAS 2009, pp.781-784, 2009.

[13] R. Serrano-Gotarredona et al., "CAVIAR: A 45k Neuron, 5M Synapse, 12GConnects/s AER Hardware Sensory–Processing–Learning-Actuating System for High-Speed Visual Object Recognition and Tracking," *IEEE Trasns. on Neural Networks*, vol. 20, N. 9, pp. 1417- 1438, Sept. 2009.

[14] L. Plana et al., "A GALS Infrastructure for a Massively Parallel Multiprocessor", *IEEE Design & Test of Computers*, Vol. 24 , Issue: 5, pp. 454 - 463, Sept.-Oct. 2007

[15] Jaer open source project.http://sourceforge.net/apps/trac/jaer/wik

[16] A. Linares-Barranco et al. "A USB3.0 FPGA Event-based Filtering and Tracking Framework for DVS," ISCAS15, Special Session on Real-Time Event-based Sensor Proc.