Neuromorphic Approach Sensitivity Cell Modeling and FPGA Implementation

Hongjie Liu[1], Antonio Rios-Navarro[2], Diederik Paul Moeys[1], Tobi Delbruck[1], and Alejandro Linares-Barranco[2]

[1] Institute of Neuroinformatics, ETHZ-UZH, Zurich, Switzerland
hongjie@ini.uzh.ch
[2] Robotic and Technology of Computers Lab, University of Seville, Sevilla, Spain
alinares@atc.us.es

Abstract. Neuromorphic engineering takes inspiration from biology to solve engineering problems using the organizing principles of biological neural computation. This field has demonstrated success in sensor based applications (vision and audition) as well in cognition and actuators. This paper is focused on mimicking an interesting functionality of the retina that is computed by one type of Retinal Ganglion Cell (RGC). It is the early detection of approaching (expanding) dark objects. This paper presents the software and hardware logic FPGA implementation of this approach sensitivity cell. It can be used in later cognition layers as an attention mechanism. The input of this hardware modeled cell comes from an asynchronous spiking Dynamic Vision Sensor, which leads to an end-to-end event based processing system. The software model has been developed in Java, and computed with an average processing time per event of 370 ns on a NUC embedded computer. The output firing rate for an approaching object depends on the cell parameters that represent the needed number of input events to reach the firing threshold. For the hardware implementation on a Spartan6 FPGA, the processing time is reduced to 160 ns/event with the clock running at 50 MHz.

Keywords: Neuromorphic engineering · Event-based processing · Address-Event-Representation · Dynamic Vision Sensor · Approach Sensitivity cell · Retina Ganglion Cell

1 Introduction

In [2], Münch et al. identified a ganglion cell type, the approach sensitivity cell (**AC**), in the mouse retina that is sensitive to approaching motion of objects. The detection of approaching motion elicits behaviors such as startle and protective motor responses in animals and humans. These responses are also important to predict collisions. This kind of function is also required in autonomous vehicles and robotics for obstacle avoidance. In [7], a time-to-contact algorithm in this kind of application based on event-based vision sensor [11] was reported. In this work, we report a more bio-inspired way of detecting approaching objects more efficiently, but also with more restrictions on the visual input.

The Dynamic Vision Sensor [1] (**DVS**) mimics the temporal dynamic responses of the retina by asynchronously outputting events signaling brightness changes. Every pixel works independently from others in such a way that when the detected brightness (log intensity) changes by more than a preset threshold from the pixel's memorized value of brightness, a spike is produced by that pixel in the sensor output. The communication protocol between event based sensors and other neuromorphic hardware is called the Address Event Representation (**AER**). The AER protocol encodes the x-y address of the pixel where the temporal change has surpassed the threshold and it transmits that address using an asynchronous handshake protocol. There are many promising applications in the literature that take advantage of this event-based processing concept, such as in [8] (one of the first) where DVS sensor output was connected to several convolutional event-based chips in parallel to detect particular objects, plus winner-take-all filters, to make it possible to move motors in order to follow one particular object in real time with sub-millisecond visual processing latencies. In [9], a spiking neural network is implemented in SpiNNaker [10] for DVS event processing to drive a mobile robot in a cognitive way.

This paper is structured as follows: the next section explains the AC biological model. Section 3 presents a software implementation of the model in Java, for the open-source jAER[1] project. Section 4 presents the hardware implementation of an AC on a field programmable gate array (**FPGA**) using a set of AER platform tools. Finally, Sects. 5 and 6 presents results and conclusions.

2 The Approach Sensitivity Cell Biological Model

In [2], it is reported that the AC receives excitatory and inhibitory inputs from small subunits. The excitatory inputs are from the so called OFF type subunits which respond to the decrease of brightness, and the inhibitory inputs are from the so called ON type subunits which respond to the increase of brightness. The ON and OFF type subunits cancel out each other when there is lateral motion. To be sensitive to approaching motion, the crucial point is that the cell **nonlinearly** integrates the potential of a broad area of ON-OFF subunits in its receptive field. The nonlinearity takes the form of an expansive function with a finite threshold. The thresholded responses of the subunits are summed into the AC (see Fig. 1). Because of this nonlinearity, weak global inhibitions will not cancel out local strong excitations, because the latter have stronger impact. The synaptic input to the AC is calculated as a weighted subtraction of the total ON and OFF units as in 1:

$$I_{net} = G_{off} * \sum V_{off} - G_{on} * \sum V_{on} \tag{1}$$

The membrane potential of the AC is calculated as 2:

$$dV_{mem} = I_{net} * dT \tag{2}$$

[1] jAER Open Source Project for real time sensory-motor processing for event-based sensors and systems. http://www.jaerproject.org/.

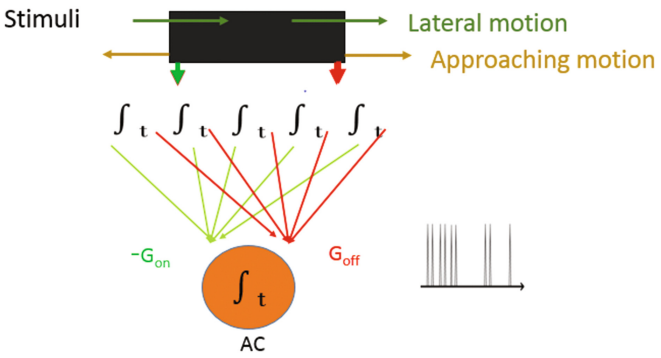


Fig. 1. Biological model of the Approach Sensitivity cell.

where dT is the inter-spike interval. To calculate the input of each subunit, the potential of each subunit is half rectified to perform non-linearity as described in [2].

3 The AC Software Implementation

Figure 2 shows the software model of the AC. One AC has been designed to have 8×8 subunits that process the input events in their receptive field in parallel. Each subunit has ON and OFF parts received from 16×16 pixels for a DVS128 sensor that has 128×128 pixels. Whenever an event with certain polarity (ON or OFF) is received in the receptive field of one subunit, the membrane potential of the ON or OFF subunit is updated. A threshold-linear nonlinearity is implemented for each subunit. All the subunits simultaneously and periodically decay. The decay time constant can be set to be tuned to a particular motion speed. The membrane potentials of all the OFF subunits are summed to provide the

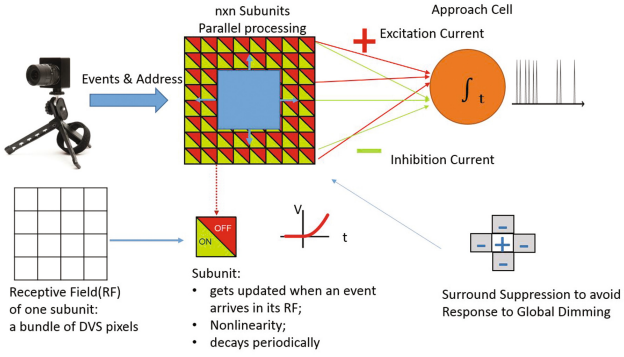


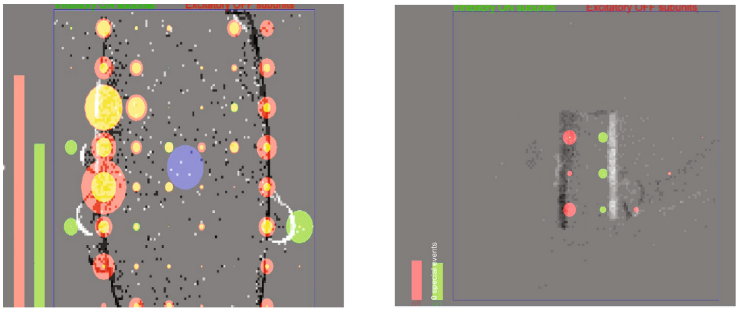
Fig. 2. Software implementation result of the Approach Sensitivity Cell.

excitation current to the AC ganglion cell; the potentials of all the ON subunits are summed to provide the inhibition current to the AC. An ON center - OFF surround subtract scheme is implemented to avoid the AC firing from global dimming of the environment. The potential of the center cell is calculated as in Eq. 3.

$$V_{centertoAC} = V_{center} - \frac{\sum V_{surround}}{n} \tag{3}$$

where n is the number of surrounding cells of the center cell; n can be 0, 2, 3, 4 depending on whether it is on the center, boarder or corner of the receptive field.

The membrane potential of the AC is calculated as in 2. It is compared either to a preset threshold voltage or a randomly generated number when Poisson firing mode is enabled. The AC fires when it is larger than the threshold at integrate-and-fire mode or a random number at Possion-fire mode. Important parameters for the AC Java model are the ON/OFF weight ratio, and the excitation strength. A parameter is also set for the maximum firing rate of the AC. Figure 3 shows the software implementation result of the AC in jAER. The object is a black cell phone. The phone is moved closer to the camera, causing it to apparently expand. The result corresponds to the working principle of the AC. The AC fires when the phone approaches (Fig. 3A), because there are more OFF events generated than ON events as the light intensity decreases on the expanding border. The AC is actively inhibited when the phone recedes, and the ON and OFF activities are balanced when it moves laterally (Fig. 3B).



A **B**

Fig. 3. Results of the Java software model. **A:** AC fires when object approaches **B:** AC doesn't fire when object movement is lateral. The red-yellow disk shows the OFF excitation while the green disk shows the ON inhibition. The blue disk shows the firing AC. The red and green bar on the left shows the total excitation and total inhibition respectively. Black/white dots represent respectively OFF/ON DVS events. (Color figure online)

4 Hardware Implementation

Figure 4 shows the state machine of the AC. The right branch shows the main state transition. Once receiving an event, after the potential of the subunit is updated (*OnEvent* or *OffEvent*), the input current of each subunit is calculated (*ComputeInputtoAC*). Then, the net current of all the subunits is computed (*ComputenetSynapticInput*). After receiving the input current, the AC updates its membrane potential state following the integrate-and-fire neuron model (*ComputeMembranState*). Then the membrane potential of the AC is compared to the threshold (*ComparetoIFThreshold*) to determine whether it should fire or not. The cell then goes back to *idle* state. The left branch of the state machine is the decay scheme. A global counter counts time that has passed. When the counter overflows, all the subunits decay by a shared programmable factor (*Decay*). If there is no event received, the cell stays in *idle* state.

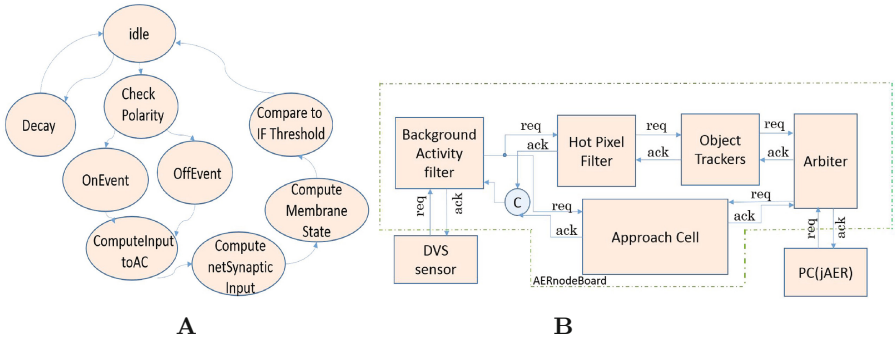


Fig. 4. Approach Sensitivity Cell hardware: **A:** State Machine. **B:** FPGA architecture

In [4], direct hardware integration of architectures for digital post-processing of DVS retina output was presented. The algorithms have the potential of lightweight, power-efficient, and fast retina processing that can be efficiently embedded inside a camera. This work reported the synthesis in a Spartan 6 platform a set of four object trackers - velocity estimation -pattern recognition systems described in VHDL in order to be used for demonstrators. This work was used as a starting point to integrate vision neural models, like [6], with the implementation of the Object Motion Cell functionality of the Retina Ganglion Cells. It is again used here for the AC.

The AC is implemented using the VHDL hardware description language in the Spartan6 1500FXT FPGA for the AERNode platform [3]. This AC implementation requires 4.5 K slice registers (2% of the FPGA), 11.2 K slice LUTs (12%) and 2 DSP blocks (1%). The AERNode platform allows multi-board communication with conventional parallel handshake AER chips, serial Low-Voltage Differential Signaling (LVDS) connections or robots control with the adequate motor interfaces. A daughter board based on an OpalKelly module,

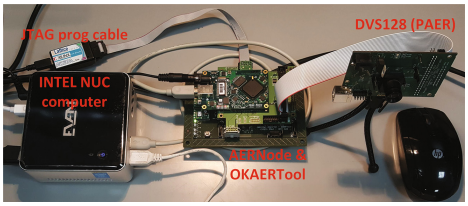


Fig. 5. The hardware system setup.

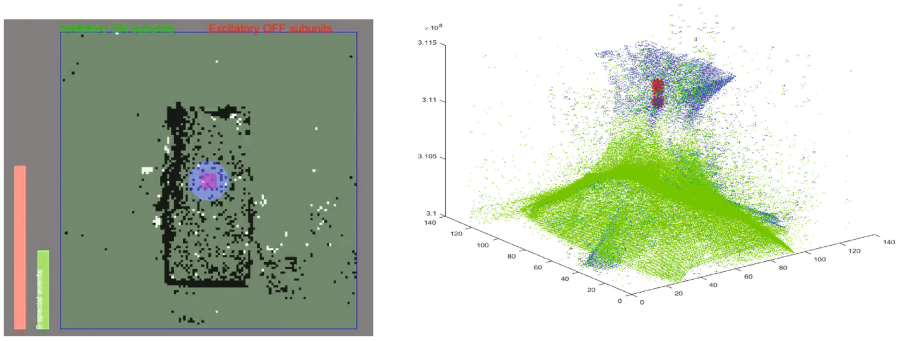
called OKAERTool [5], is used for monitoring, sequencing, logging or playing events from and to the AERNode board. It is able to sequence events from its on-board DDR2 128MB SDRAM to the AERNode board and to monitor its output through USB2.0 in jAER. OKAERTool is used for debugging.

In this work we have implemented one AC in the FPGA to prove functionality. There are available resources in the FPGA to implement several ACs spread in different regions of the visual field if particular applications require that. The implementation is structured in two levels. The first level is called Mother Cell (**AC-MC**), which is ready to host one or many AC in different visual fields, together with the circuit to receive the corresponding parameters for each cell through an Serial Peripheral Interface (SPI) to the USB interface microcontroller on the OKAERTool. Figure 5 shows the testing scenario, where a DVS128 retina is connected to the OKAERTool merger. The OKAERtool is plugged to the AERNode, which is running the system showed in Fig. 4B. OKAERTool is configured to send to the AERNode the DVS128 output and to collect the output of the AERNode and send it to jAER through USB2.0. The output of the AERNode is configured to be the merge of the DVS128 after a Background-Activity-Filter in the FPGA, and the output of the AC. The USB2.0 output of the OKAERTool is connected to the NUC embedded computer, which is running jAER. This jAER is processing live incoming events and it is executing two processing filters: (1) the AC software model and (2) a simple algorithm to highlight the AC hw output in the screen.

5 Results

Figure 6A shows a snapshot of the combined results of the software and hardware implementations. The blue circle in the middle shows the software AC firing, while the red square shows the hardware AC firing when the object is approaching. Red and green bars on the left show an accumulated view of OFF and ON events respectively. Figure 6B shows a space-time view of the event data. The vertical dimension is the increasing timestamp while the horizontal dimensions are the (x, y) address of the pixel array. As time increases, the area of the blue dots (events) first decreases and then increases, indicating that the object first recedes (shrinks in the view) and then approaches (expands in the view). We can see the AC fires when the object approaches and does not fire

when it recedes. Given this slow stimulation, the minimum inter-spike-interval (**ISI**) for each AC firing signal is 16.6 ms and the maximum is 75.7 ms. The average ISI is 31.2 ms. The latency to process one incoming event depends on the state machine. The clock of the FPGA runs at 50 MHz. The latency of the AC is 160 ns. It is the time difference between the input request to the AC and the output of the AC, measured using ChipScope embedded in the FPGA. Table 1 shows a comparison of the software and hardware performance in terms of latency and power. The software performance is measured in two types of PCs with very different resources. The software latency is measured by using the "measure filter performance" option in jAER.



A **B**

Fig. 6. A: Result of AC software and hardware implementation displayed in jAER viewer. **B:** 3-dimensional visualization of address-event and the AC firing, red stars shows the hardware AC firing, blue and green dots represent the excitatory (OFF) and inhibitory (ON) events respectively. (Color figure online)

Table 1. Performance table

	jAER (64 bit Intel NUC, 4 GB RAM, i54250U, 1.30 GHz)	jAER(64 bit PC, 16 GB RAM, i74770K, 3.50 GHz)	FPGA Xilinx Spartan6, 50 MHz
Latency	370 ns/ev at 0.2 Mev/s, at CPU load 5%	55 ns/ev at 0.2 Mev/s, at CPU load 3%	160 ns/ev at any event rate
Power	6.2 W static 6.2 W for running jAER 2.5 W	160 W	0.775 W static 0.05 W

6 Conclusions

This paper offers software and FPGA implementation of the AC model for real time detection of expanding dark objects. The FPGA implementation requires less than 0.8 W power and has a latency of 180 ns, which is smaller than that of the software approach, 370 ns on the embedded "Next Unit of Computing" Intel NUC computer. The software model running on a more powerful desktop PC takes 55 ns to process one event (on average) with a power consumption higher than 100 W. Future work will be focused on expanding from one AC to multiple ACs in order to infer the relative location of the approaching object. This will also solve the issue that multiple objects moving in opposite directions cancel out their effect to the AC. It would be interesting as well to integrate the AC with the OMC [6] or other algorithms where the AC serves as an attention mechanism.

Acknowledgments. This work has been partially supported by the Spanish government grant (with support from the European Regional Development Fund) COFNET (TEC2016-77785-P) and the European Project VISUALISE (FP7-ICT-600954). We thank Prof. Francisco Gomez-Rodriguez for his support.

References

1. Lichtsteiner, P., Posch, C., Delbrck, T.: A 128 x 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. IEEE J. Solid- State Circ. **43**(2), 566–576 (2008)
2. Münch, T.A., et al.: Approach sensitivity in the retina processed by a multifunctional neural circuit. Nat. Neurosci. **12**(10), 1308–1316 (2009)
3. Iakymchuk, T., et al.: An AER handshake-less modular infrastructure PCB with x8 2.5 Gbps LVDS serial links. In: 2014 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE (2014)
4. Linares-Barranco, A., et al.: A USB3.0 FPGA event-based filtering and tracking framework for dynamic vision sensors. In: 2015 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2417–2420 (2015)
5. Rios-Navarro, A., et al.: A 20 Mevps/32 Mev event-based USB framework for neuromorphic systems debugging. In: 2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP). IEEE (2016)
6. Moeys, D.P., et al.: Retinal ganglion cell software and FPGA model implementation for object detection and tracking. In: 2016 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE (2016)
7. Clady, X., et al.: Asynchronous visual event-based time-to-contact. In: Neuromorphic Engineering Systems and Applications 51. APA (2015)
8. Serrano-Gotarredona, R., 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. IEEE Trans. Neural Netw. **20**(9), 1417–1438 (2009)

9. Denk, C., Llobet-Blandino, F., Galluppi, F., Plana, L.A., Furber, S., Conradt, J.: Real-time interface board for closed-loop robotic tasks on the SpiNNaker neural computing system. In: Mladenov, V., Koprinkova-Hristova, P., Palm, G., Villa, A.E.P., Appollini, B., Kasabov, N. (eds.) ICANN 2013. LNCS, vol. 8131, pp. 467–474. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40728-4_59
10. Khan, M.M., et al.: SpiNNaker: mapping neural networks onto a massively-parallel chip multiprocessor. In: IEEE International Joint Conference on Neural Networks, 2008, IJCNN 2008. IEEE World Congress on Computational Intelligence. IEEE (2008)
11. Delbck, T., et al.: Activity-driven, event-based vision sensors. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE (2010)