A Digital Multiplier-less Neuromorphic Model for Learning a Context-Dependent Task

Hajar Asgari and Babak Mazloom-Nezhad Maybodi
Department of Electrical Engineering
Shahid Beheshti University Tehran, Iran
Email: [h_asgari|b-mazloom]@sbu.ac.ir

Raphaela Kreiser and Yulia Sandamirskaya
Institute of Neuroinformatics
University of Zurich and ETH Zurich, Switzerland
Email: [rakrei|ysandamirskaya]@ini.uzh.ch

***Abstract*—Highly efficient performance-resources trade-off of the biological brain is a motivation for research on neuromorphic computing. Neuromorphic engineers develop event-based spiking neural networks (SNNs) in hardware. Learning in SNNs is a challenging topic of current research. Reinforcement learning (RL) is a particularly promising learning paradigm, important for developing autonomous agents. In this paper, we propose a digital multiplier-less hardware implementation of an SNN with RL capability. The network is able to learn stimulus-response associations in a context-dependent learning task. Validated in a robotic experiment, the proposed model replicates the behavior in animal experiments and the respective computational model.**

***Index Terms*—Neuromorphic engineering, spiking neural networks, reinforcement learning, context-dependent task.**

I. Introduction

Brain-inspired computing systems offer massively parallel, event-driven, dedicated computing substrate for spiking neural networks [1]–[5]. During the past few years, hardware SNNs have received considerable attention in the artificial neural network community because of their performance-resource trade-off [6], [7]. On the other hand, the neuroscience community is also interested in neuromorphic hardware for acceleration of large-scale models of biological neuronal circuits [8], [9].

Similar to conventional neural networks, the capability to learn is an important feature in SNNs. Since conventional supervised learning methods cannot easily be used in SNNs due to the non-differentiable event-based nature of their activation, neuromorphic systems often experiment with unsupervised learning [7], [10]. Reinforcement learning is another powerful learning paradigm, well-suited for neuromorphic realisation, that is particularly important for learning how to make a correct decision in an autonomous agent [11]. In this paper, we present a digital prototype for a neuromorphic implementation of RL. We use the Field-Programmable Gate Arrays (FPGAs) because of their flexibility, reliability, and reconfigurability [12]–[16].
In particular, this paper proposes a digital circuit implementation of an event-driven spiking neural network with RL capability. The SNN is composed of leaky integrate-and-fire (LIF) neurons and spike-timing-dependent plasticity (STDP) based synapses. We use this circuit to demonstrate reinforcement learning of a context-dependent task [17], suitable for online learning on an autonomous agent. We implement the network on an FPGA chip and compare the hardware model results during learning with the computational model developed based on a rat experiment. Moreover, we implement the behavioral experiments using a robot to show that learning in hardware works in a closed sensorimotor loop.

II. System Architecture

The proposed architecture implements spiking neurons connected by plastic and static synapses through a synaptic crossbar. The developed network architecture is parallel and event-driven without using multipliers. All cores use simple basic operations (add, shift, etc.) in their building block. Fig. 1 shows the architecture of the proposed fully digital SNN.

In this network, the synaptic crossbar connects all neurons. Besides neurons, synapses, and synaptic crossbar, several peripheral cores manage the network behavior [18]. Thus, the system consists of four different parts:

Neuron core: In this architecture, all neurons are located in the neuron core. The neuron model used in this hardware is based on a low-complexity LIF. The operation of the LIF is expressed in four basic states: 1. Resting, 2. Waiting, 3. Integrating, 4. Firing. The state diagram of the proposed system is depicted in Fig. 2. Neuron in the Resting state is not active. Based on the Table. I, the neuron's membrane potential (V_m) in this state is equal to reset voltage (V_{reset}). Whenever the neuron is active but there is no input, it stays in the Waiting state. While waiting for new input, there is a constant leakage (V_{leak}) in the neuron's membrane potential. When an input spike arrives, the neuron's state switches to the Integrating-state. In this state all inputs are accumulated according to Table. I and Fig. 2. After integrating all inputs, a barrel shifter produces the total exitatory post synaptic potential (EPSP) for the neuron. Based on the membrane voltage value, the neuron goes to the Firing (if $V_m > V_{th}$) or Resting states (if $V_m < V_{reset}$) or returns to the Waiting state (if $V_m < V_{th}$). In the Firing state, the neuron emits a spike and goes to the Waiting state. Importantly, there is no multiplier in the neuron's building block, which simplifies hardware synthesis of this block.

Synapse core: In this network there are two types of synapses: inhibitory static synapses and excitatory plastic synapses. For plastic synapses, STDP-based learning rules are

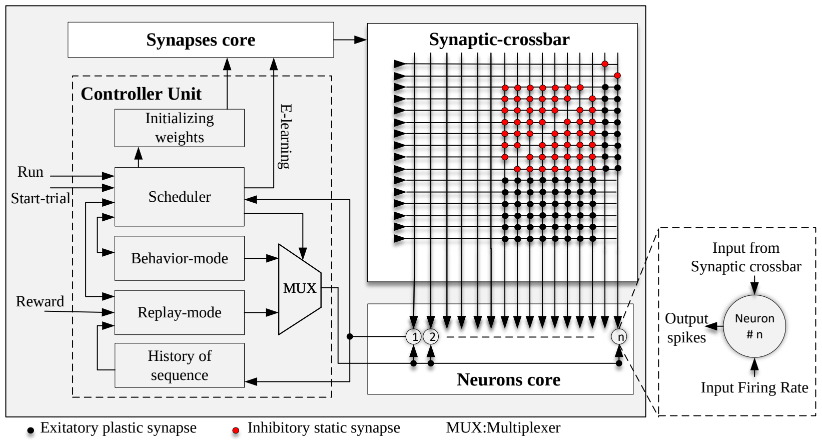


Fig. 1: System level architecture of the digital multiplier-less event-driven SNN with reinforcement learning capability.

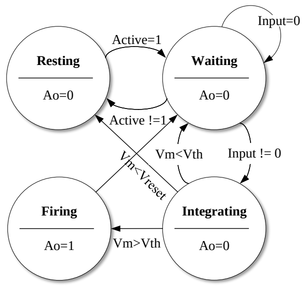


Fig. 2: The state diagram of the LIF neuron. Every circle represents the state in the upper part and the output of the neuron in the lower part. Operation of the LIF neuron model is simplified in four basic states: 1. Resting, 2. Waiting, 3. Integrating, 4. Firing. V_m is given in circuit level.

employed. The STDP algorithm adapts the weight of a synapse according to the timing difference between the pre- and post-synaptic spikes arrival times. In the synapse model of the proposed system, synaptic modifications depend on the previous strength of the synaptic weight. For maximum simplification and to keep the dependence on timing differences and previous

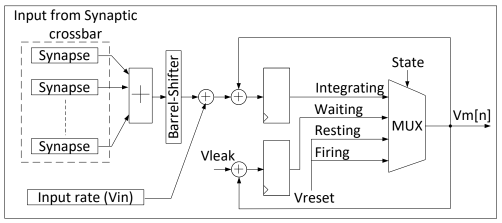


Fig. 3: Circuit level block diagram for calculating the membrane voltage of neuron in each state. V_{th}, V_{reset}, and V_{leak} are neuronal parameters. Based on the neuron state, V_m is running in different ways.

TABLE I: Equations for the membrane potential in each state.

Resting	No-spike $V_m^j[n] = V_{reset}$
Integrating	No-spike $V_m^j[n] = V_m^j[n-1] + \sum_{i=1}^{all} A_i W[i][j] + V_{in}^j[n]$
Waiting	No-spike $V_m^j[n] = V_m^j[n-1] - V_{leak}$
Firing	Spike $V_m^j[n] = V_{reset}$

synaptic weights, we propose the following learning rule:

$$\Delta W = \begin{cases} A^{+}(W_{Max} - W) & if\ \Delta t > 0 \text{ and } E_{learning} = 1, \\ A^{-}(W - W_{Min}) & if\ \Delta t < 0 \text{ and } E_{learning} = 1, \end{cases} \quad (1)$$

where the ΔW is the synaptic modification, W is the synaptic weight and W_{Max} and W_{Min} determine the dynamic range of its variation, Δt is the timing difference between arrival time of the pre- and post-synaptic spikes and A^{+} and A^{-} are amplitudes of positive and negative synaptic modifications. By sending a signal to enable the learning phase ($E_{learning} = 1$), the synapse is eligible to change. Taking advantage of these simplifications, the hardware model synapse avoids using multipliers.

Synaptic crossbar: As shown in Fig. 1, in this network all neurons are sorted in a row and are connected through the synaptic crossbar. The value of each node in the synaptic-crossbar is directly determined by initializing weights of plastic synapses core. Black and red circles in Fig. 1 show the excitatory and inhibitory synapses. In this system, it is possible to create arbitrary connectivity schemes, for instance Winner-Take-All (WTA) networks by assigning positive and strong negative synaptic weights.

Controller-unit: The controller unit is responsible for controlling the behavior of the system, sequences, data storage, and preparing initial weights for synapses. This block contains several sub-blocks. The scheduler as the main sub-block decides to enable the other sub-blocks in different sequences. During a network run, at first all synaptic weights are initialized. Each network run includes several trials in order to converge. Each trial consists of a behavioral phase which is a typical network operation and a replay phase which is the learning mode based on the reward. When the network is in the behavior mode, the neurons' states are stored in history and provide the required information for the learning mode. The controller unit manages system operation in different phases.

III. Network model for a Context-dependent task

A context-dependent choice task was designed to study the activity of rat's hippocampal neurons when the animal learns to chose different rewarded items dependent on context [19]. In this experiment, there are two distinguishable contexts –A and B– and two different places in each context. Hence, there are four spatial locations: A1, A2, B1, B2. For example, A1 refers to position 1 in context A. Based on the position of two items X and Y and the contexts, eight different triplets appear. For example, triplet A1X means that item X is located in position 1 in context A. Rats learn to choose item X in context A and item Y in context B, independent of their position. Thus, the rewarded triplets are A1X, A2X, B1Y, and B2Y. The non-rewarded group contains A1Y, A2Y, B1X, and B2X [19].

The spiking neural network model for learning the context-

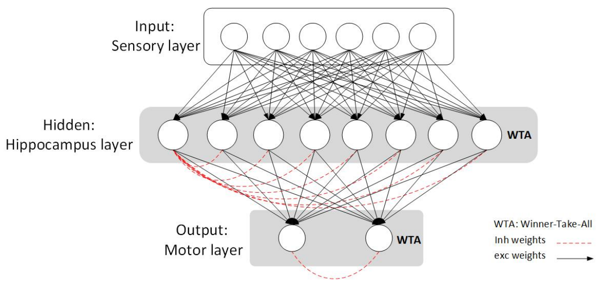


Fig. 4: The spiking network, used to model the reinforcement learning in the context-dependent task. All to all excitatory synapses connect layers in this feed forward network. Inhibitory synaptic connections among all neurons create WTA networks (one neuron's inhibitory synapses are shown in this figure).

dependent task is depicted in Fig. 4. The feed-forward SNN network is composed of input, hidden and output layers. There are excitatory all-to-all connections between layers. Lateral inhibitory synaptic connections in the middle and output layers are strong and construct winner-take-all (WTA) networks. In this model, the resting membrane potential and threshold voltage values of all neurons are -70 and -50 mV respectively. There is a concurrent small leakage in membrane voltage at every time-step. Synaptic weights have different, initially random values between 0 and 1.

We let the network run for 100 trials. In each trial, the network has random initial weights and starts from a randomly selected input stimulus. Each trial consists of two different modes: behavior and replay. Maximum required time based on the system clock for completing each trial and replaying

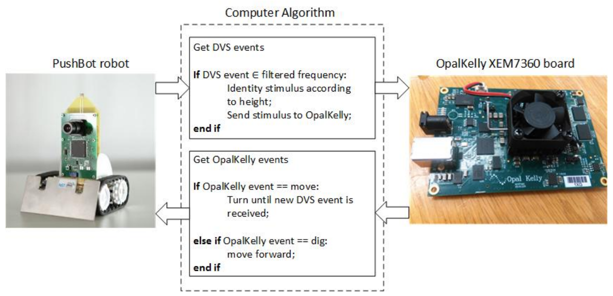
TABLE II: Details of selected parameters. For FPGA implementations numbers are in fixed-point 32 bits digital format.

Design choice		
# Input neurons	N_{Input}	6
# Hidden neurons	N_{Hidden}	8
# Output neurons	N_{Output}	2
Long term potentiation amplitude (LTP)	$A+$	$+2^{-10}$
Long term depression amplitude (LTD)	$A-$	-2^{-11}
Maximum time interval for a trial	T_{trial}	30000 T_{clk}
Maximum time interval for replay	T_{replay}	130 T_{clk}
Input voltage for input neurons	V_{input}	1.28 mV
Input voltage for hidden neurons	V_{hidden}	1.48 mV
Input voltage for output neurons	V_{output}	1.64 mV

phase are reported in Table II. During the behavior-mode in this task, the network starts from a randomly selected binary input triplet. Based on the value of the synaptic weights, the behavior phase can include a few movements between the two triplets (if "move" output neuron is active) and ends up with digging on one of them (whenever "dig" output neuron is active)[1]. After receiving events from the "dig" output neuron, the scheduler switches to the replay mode. During the behavioral phase the states of all neurons for each "move" or "dig" are sampled in the history-of-sequences. In the replay phase, a controlling signal makes the plastic synapses eligible to learn. Furthermore, according to the obtained reward and the sampled action sequences, the controller unit provides suitable inputs for all neurons during replay phase (see Table II).

TABLE III: Overview of FPGA device utilization plus maximum frequency for the implemented SNNs.

Resource	[14]	[20]	[21]	[12]	Proposed
Slice Registers	1023	50228	-	1676	-
Flip flops	-	-	-	-	8906
Slice LUTs	11339	86032	-	6214	19059
DSPs	0	1112	-	32	0
F_{Max}(MHz)	189	63.389	75	25	148.4
Device	Virtex6	Virtex7	Spartan6	Spartan6	Kintex7



```
Get DVS events

If DVS event ∈ filtered frequency:
        Identity stimulus according
        to height;
        Send stimulus to OpalKelly;
end if
```

```
Get OpalKelly events

If OpalKelly event == move:
        Turn until new DVS event is
        received;

else if OpalKelly event == dig:
        move forward;
end if
```

Fig. 5: Overview of the digital event-driven SNN model for learning context-dependent task implemented on the Opalkelly XEM7360 board and connected to the Pushbot robot setup [18].

[1]"Dig" in the experiment is an action that the rat takes to get the hidden reward.

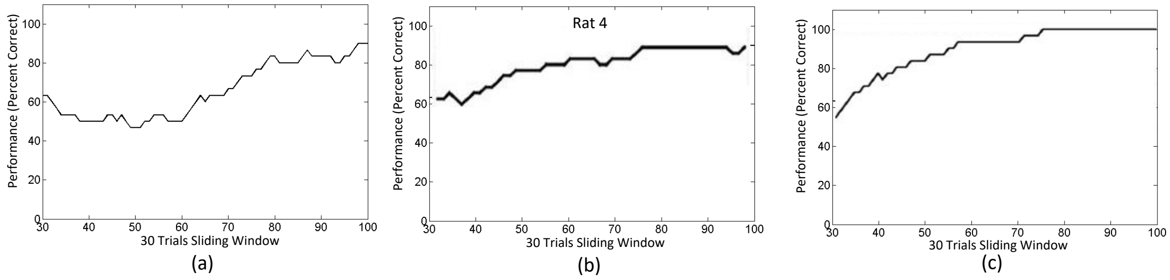


Fig. 6: Behavioral performance during successful learning of the context-dependent task: (a) The proposed digital event-driven hardware model. (b) The animal experiment [19]. (c) The computational model [17].

IV. Robotic experiment

In order to verify the proposed system, we interfaced the digital neuromorphic system on FPGA to a neuromorphic dynamic vision sensor (DVS) mounted on a robotic vehicle and developed an autonomous neuromorphic agent that is able to learn the context-dependent task. As shown in Fig. 5, the experimental setup used in this work consists of the Pushbot robotic vehicle with an embedded DVS camera (eDVS [22]) and the Opallkelly XEM7360 board. Furthermore, a computer is used to direct the flow of events between the robot and the SNN on FPGA. The computer runs a simple program that manages the stream of events between the neuromorphic system on FPGA and the robot. The supplementary movie shows the robot's behavior during the first trials and also after complete training[2].

V. Results and Discussion

The proposed digital spike-based neural network with RL capability is described using the standard top-bottom digital ASIC design flow with separate fully-synthesizable synchronous modules. As a multiplier is a high-cost core in terms of area utilization and power consumption, the proposed architecture does not contain any multipliers. In order to achieve a fair balance between accuracy and cost, the proposed spiking neural network model performs data transformation using 32-bit fixed-point numbers with 31 fractional bits. To minimize silicon area utilization, neuron and synapse blocks are designed as simple as possible.
As a proof of concept we implement the network on Kintex-7 XC7kt160t FPGA. This FPGA device contains 101400 slice LUTs and 202800 slice Flip Flops. Table III reports the FPGA device utilization of the proposed SNN and a collection of previous studies on implementing different SNN models, in which spiking neural networks are implemented on similar hardware platforms. The proposed SNN uses a total of 4.39 and 18.80 percent of the available Slice FFs and slice LUTs, respectively. Please note that other networks in Table III are not applied in the same task. Furthermore, different FPGA devices and synthesizer versions have been used for implementations. Therefore, the device utilization results presented in this table should not be used for comparison but act as an overview of similar research.

We propose an event-driven hardware model which is able to learn the same task as in the biological experiments and previous computational studies, within 100 trials. The proposed network with randomly initialized synaptic weights is simulated using the Xilinx Vivado Design suite. Performance within 80 % to 90 % was reached (see Fig. 6(a)). As shown in Fig. 6(b), the experimental study of the context-dependant task was learned in about 100 trials where rats reached about 80% to 90% correct behavioral response [19]. The computational model was able to learn the task in 100 trials with a mean of 80% to 90% correct detection rate [17] (Fig. 6(c)). Thus, the proposed digital event-driven network shows a performance that is more similar to animal behavior in the experiment.

VI. Conclusion

This paper proposes a digital, event-driven spiking neural network with the capability of reinforcement learning. Due to hardships in multiplier implementations in hardware, both neurons and synapses avoid using multiplier cores in their building blocks. As a proof of concept, we implement the proposed model on the Xilinx Kintex-7 FPGA device. For validating the network ability of online learning on hardware, we connect the proposed network to a robotic vehicle in a closed sensory-network-motors loop. The proposed model, both on hardware and in the robotic experiment successfully learns a context-dependent task and shows performance close to the computational model and also to the animal experiment of previous studies. This work facilitates research to employ reinforcement learning for autonomous agents using neuromorphic systems.

Aknowledgement

This work was supported by SNSF grant Ambizione (grant No. PZOOP2_168183), and Ministry of Science, Research and Technology (MSRT), Islamic republic of Iran.

References

[1] A. D. Brown *et al.*, "SpiNNaker - Programming Modele," *IEEE Trans. Computers*, vol. 64, no. 6, pp. 1769–1782, 2015.

[2]Shared video: https://www.dropbox.com/s/vokdma10j5z763m/experiment-clip.mp4?dl=0

[2] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014. [Online]. Available: https://science.sciencemag.org/content/345/6197/668

[3] C. Frenkel *et al.*, "A 0.086-mm212.7-pj/sop 64k-synapse256-neuron online-learning digital spiking neuromorphic processor in 28nm cmos system," *IEEE Trans. Biomedical Syst*, vol. 13, no. 1, pp. 145 – 158, 2019.

[4] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, January 2018.

[5] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, Oct 2015.

[6] B. V. Benjamin *et al.*, "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[7] N. Qiao *et al.*, "A Reconfigurable On-Line Learning Spiking Neuromorphic Processor Comprising 256 Neurons and 128K Synapses," *frontiers in Neuroscience*, vol. 9, no. 141, pp. 1–17, 2015.

[8] E. Izhikevich, "Simple Model of Spiking Neurons," *IEEE Trans. Neural Netw*, vol. 14, no. 6, pp. 1569–1572, 2003.

[9] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, 2004.

[10] C. Frenkel, G. Indiveri, J. Legat, and D. Bol, "A fully-synthesized 20-gate digital spike-based synapse with embedded online learning," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning:An Introduction*, 2nd ed. Massachusetts,US: The MIT Press Cambridge, 2018.

[12] D. Ma *et al.*, "Darwin: A Neuromorphic Hardware Co-processor Based on Spiking Neural Networks," *Journal of Computational Electronics*, vol. 77, pp. 43–51, 2017.

[13] C. Lammie, T. Hamilton, and M. R. Azghadi, "Unsupervised Character Recognition with a Simplified FPGA Neuromorphic System," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[14] E. Z. Farsa *et al.*, "A Low-Cost High-Speed Neuromorphic Hardware Based on Spiking Neural Network," *IEEE Trans. Circuits Syst. II*, vol. 66, no. 9, pp. 1582–1586, 2019.

[15] C. Lammie, T. J. Hamilton, A. van Schaik, and M. R. Azghadi, "Efficient FPGA Implementations of Pair and Triplet-Based STDP for Neuromorphic Architectures," *IEEE Trans. on Circuits and Systems I*, vol. 66, no. 4, pp. 1558–1570, 2019.

[16] H. Soleimani and E. M. Drakakis, "An Efficient and Reconfigurable Synchronous Neuron Model," *IEEE Trans. Circuits Syst. II*, vol. 65, no. 1, pp. 91–95, 2018.

[17] F. Raudies and M. E. Hasselmo, "A Model of Hippocampal Spiking Responses to Items During Learning of a Context-Dependent Task," *frontiers in System Neuroscience*, vol. 23, no. 8, pp. 1–12, 2014.

[18] H. Asgari *et al.*, "Digital Multiplier-less Event-Driven Spiking Neural Network Architecture for Learning a Context-Dependent Task ," *arXiv:1906.09835*, 2019.

[19] R. W. Komorowski, J. R. Manns, and H. Eichenbaum, "Robust Conjunctive Item-place Coding by Hippocampal Neurons Parallels Learning What Happens Where," *Neural Comput.*, vol. 29, no. 31, pp. 9918–9929, 2009.

[20] S. Y. Bonabi *et al.*, "FPGA Implementation of A Biological Neural Network Based on The Hodgkin-Huxley Neuron Model," *Frontiers in Neuroscience*, vol. 8, no. 379, pp. 1–12, 2014.

[21] D. Neil and S. Liu, "Minitaur, An Event-Driven FPGA-Based Spiking Network Accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2621–2628, Dec 2014.

[22] J. Conradt, R. Berner, M. Cook, and T. Delbruck, "An Embedded AER Dynamic Vision Sensor For Low-Latency Pole Balancing," *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pp. 780–785, 2009. [Online]. Available: http://citeseerx.ist.psu.edu