Gyro: A Digital Spiking Neural Network Architecture for Multi-Sensory Data Analytics

FEDERICO CORRADI, Stichting IMEC Nederland, The Netherlands
GUIDO ADRIAANS, Eindhoven University of Technology, The Netherlands
SANDER STUIJK, Eindhoven University of Technology, The Netherlands

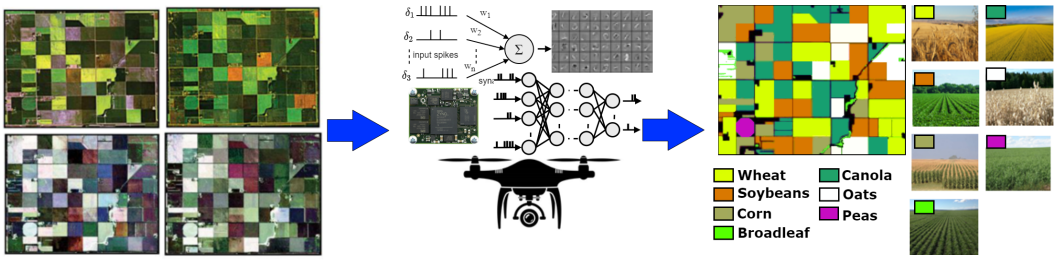

Fig. 1. Cropland monitoring with a spiking neural network embedded in FPGA. Optical-radar data are processed in an hardware implementation of a spiking neural network that outputs pixel-wise classification for seven types of crop.

Unmanned Aerial Vehicles (UAVs) that interact with the physical world in real-time make use of a multitude of sensors and often execute deep neural network workloads for perceiving the state of the environment. To increase UAVs operations, it is required to execute these workloads in the most power efficient manner. Spiking neural networks (SNNs) have been proposed as an alternative solution for the execution of deep neural networks in an energy-efficient way. We introduce Gyro, a digital event-driven architecture capable of executing spiking neural networks. The architecture is tailored towards sensory fusion applications and it is optimized for Field-Programmable-Gate-Arrays (FPGAs). In hardware, we demonstrate the performance of a sensory fusion task using a public dataset of bi-temporal optical-radar data for pixel wise crop classification. We achieve an accuracy of 99, 7%, a peak throughput of 31, 82 GSOPS (Giga Synaptic Operations per Second) while consuming 50 pJ/SO (pico Joule / Synaptic Operation) @ 31, 82 GOPS.

CCS Concepts: • **Computer systems organization → Embedded hardware**.

Additional Key Words and Phrases: optical-radar sensory fusion, spiking neural networks, embedded hardware, fpga, remote sensing

ACM Reference Format:
Federico Corradi, Guido Adriaans, and Sander Stuijk. 2021. Gyro: A Digital Spiking Neural Network Architecture for Multi-Sensory Data Analytics. In *Drone Systems Engineering (DroneSE '21), January 18, 2021, Budapest, Hungary*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3444950.3444951

DroneSE '21, January 18, 2021, Budapest, Hungary

ACM ISBN 978-1-4503-8952-5/21/01…$15.00
https://doi.org/10.1145/3444950.3444951

1 INTRODUCTION

In recent years the development of high-resolution sensing technologies, UAVs, and the advancement of deep learning have opened new application opportunities in the field of robotics, agricultural automation, earth observation, surveillance and inspection [14] [5]. A significant amount of data is already available and can be used to train deep learning models to extract thematic information (e.g. land cover usage, biomass partition, crop types, etc.) using data acquired from various airborne, UAVs and/or satellite systems. In the applications of agricultural automation and earth observation it is important to automatically produce, based on raw-data, semantic maps containing various regions of interest. For example, in land cover mapping, each pixel is assigned to a particular class according to the type of land cover (vegetation, water, road, etc.) or objects (house, airport, farm, etc.) observed at the pixel level, and using a variety of sensor modalities, usually optical-radar data. As the amount of data increases, deep networks have been successfully trained to automatically label and process these data. However, the execution of these deep neural network models directly on UAVs systems poses serious challenges in terms of power consumption and speed of execution. Even though the energy efficiency of DNNs has been improved substantially and will likely keep improving, Spiking Neural Networks (SNNs) have been proposed as an efficient way for the execution of DNNs in embedded hardware [17]. SNNs are more biologically inspired models than DNN. In SNNs the communication among neurons is carried by means of action potentials, also known as spike trains. These individual spikes are sparse in time, and at first approximation, they appear uniform and binary. Thus, the information in SNNs propagates with the exact time of a spike, that includes spike latency, as well as spike rates. SNNs offer a unique opportunity as computation can be carried out only when spikes event occur, and in deep spiking neural networks only a fraction of neurons are active at each time step [15].

Current hardware platforms that target the execution of SNN workloads make use of neuromorphic ASICs, such as INTEL loihi [1], IBM TrueNorth [3], multi-core processors, or graphical processing units, and offer various degree of programmability (parameters as well as instruction level control). While ASICs offer high-performance and low-power consumption, they are fixed at fabrication. On the other hand, multi-core and GPU systems offer very good programmability and flexibility but they tend to have larger power consumption. FPGA systems have been proposed as accelerators for SNNs, in these systems conductance based models as well as leaky-integrate-and-fire neurons have been implemented [16], [18], [11], [10], [13], [4]. Despite very recent progress, these FPGAs-based accelerators lack design automation tools, and support only a small range of models, thus resulting in low flexibility. We address these challenges with the development of Gyro, a digital-event driven SNN architecture. Our main contributions are the following: i) an efficient mapping strategy that makes use of on-chip memory ii) a digital event-driven design that exploits sparse information processing to reduce the number of operations compared to time-based computation iii) a simplified leaky-integrate-and-fire neuron model that exploits bit-shift operations to approximate the decay of the membrane potential, thus reducing the use of resources. In addition, Gyro aims to ease its integration in drone platforms by providing a parametric interface at compile time, and automating the development of applications by using high-level python software for training and instantiating SNNs in FPGA.

2 SPIKING NEURAL NETWORKS

A SNN is formed by interconnecting spiking neurons. Neurons in a spiking neural network are state full, this means that they store the weighted history of the recent input spikes in their membrane potential. When the membrane potential exceeds its threshold value it generates an output spike, i.e. a binary event in time. The neurons are connected by means of weighted synapses. A biologically

plausible neuron model is the Leaky-Integrate-and-Fire (LIF) model [2]. This simplified neuron model has been widely used to emulate the dynamics of SNNs. The dynamics of the sub-threshold membrane potential is described by the following differential equation

$$\tau_m \frac{dV_{mem}}{dt} = R \cdot I(t) - V_{mem}(t) \tag{1}$$

where τ_m is the membrane time constant, V_{mem} the membrane potential, I is the total input current, and R is the membrane resistance. As the model name implies, the neuron integrates the input current in its membrane potential V_{mem}. Once the membrane potential exceeds its threshold potential V_{thr}, an output spike is generated and the membrane potential is reset to V_{rst} and remains V_{rst} during the refractory period t_{refrac}. The neuron model has a time-dependent memory as the membrane potential leaks away exponentially over time. An exemplary LIF membrane potential behavior is shown in Figure 2.

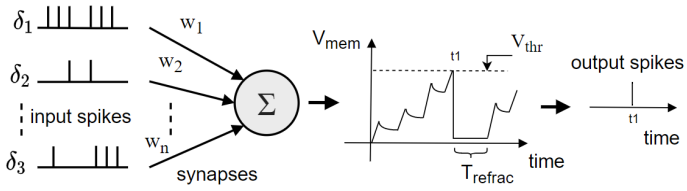

Fig. 2. Leaky-integrate-and-fire neuron's membrane potential. Input and output spikes are depicted as instantaneous events.

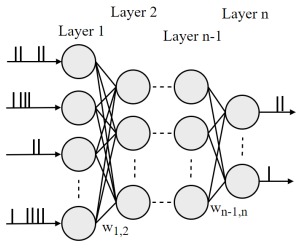

Fig. 3. A fully connected SNN

The input current $I(t)$ is defined as the weighted summation of the input spikes as

$$I(t) = \sum_{i=0}^{n} (w_{i,j} \sum_{k} \delta_i(t - t_k)) \tag{2}$$

in which n indicates the number of synaptic contacts, $w_{i,j}$ is the synaptic weight that connects the ith neuron to the jth target neuron. The $\delta_i(t - t_k)$ represents a Kronecker delta function that is used to indicate a spike event from ith neuron to the target neuron.

$$\delta_i(t - t_k) = \begin{cases} 1, & \text{if } t = t_k, \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Where t_k is the time at which the spike event occurs. This means that LIF neurons implement instantaneous dynamics. When neuron i spikes to neuron j, the membrane potential of neuron j is increased or decreased with a step of size $w_{i,j}$.

Neural network architectures are achieved by connecting spiking neurons in a layered structure. In this way, a full mesh of spiking neuron process input spike trains and output spike trains. In a fully connected structure, as shown in Figure 3, input features are encoded in spikes by using a mean rate or a spike time interval approach. Outputs of the networks are also spikes, a common way to interpret these outputs is by using first-time to spike or mean rate activity over a time interval.

3 GYRO: AN EVENT-BASED ARCHITECTURE

The design of our architecture is based on a layered structure of fully-connected neurons, this means that each spike is sent to all neurons of another layer and therefore all neurons in a layer are always updated simultaneously. Our architecture supports recurrent fully-connected layers. The assumption that all layers are fully-connected has two advantages. Firstly, a fully-connected network is the worst-case scenario in terms of connection number, thus any connection scheme that requires less connections is inherently supported. Secondly, by updating all neurons in a layer simultaneously upon an input spike, the implementation and behavior is predictable. Apart from interfacing modules, the design consists of two main modules: the layer and the weight memory. There is a layer module for each layer in the SNN. Between every two layers there is a weight memory module that stores synaptic weights used to interconnect the two layers. Hence, for a SNN of n layers, the design consists of n layers and n weight memories. The layer module consists of a spike queue that receives one or two streams of spikes. The weight controller receives spikes from the spike queue and retrieves the corresponding weights from the weight memories depending on the spike source. The neuron wrapper contains the LIF neurons along with spike buffering and arbitration. In addition, a timer module creates a notion of time which is required for the calculations in the neurons (decay, and refractory updates). A set of registers is used to program network's parameters via an AXI4-lite interface. The main inputs of the network are spikes that

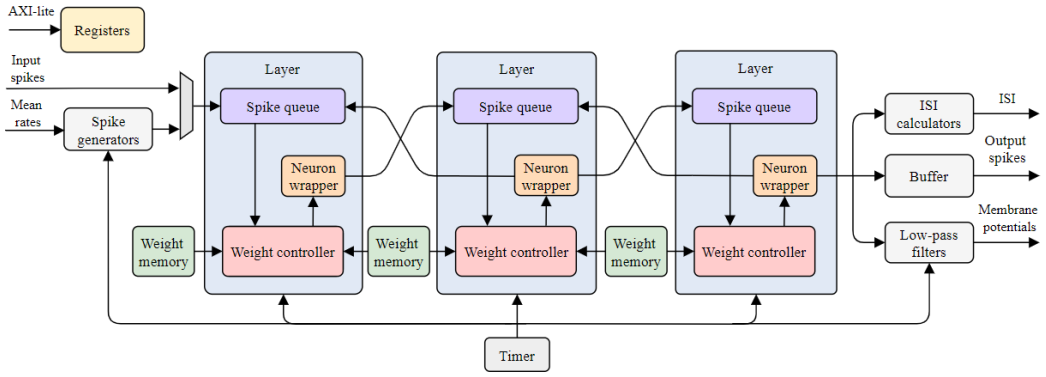

Fig. 4. Spiking Neural Network Architecture. Layers support up to fully-connected forward and backwards synaptic connections.

stream into the spike queue at the input of the first layer module, as shown in the top-left of Figure 4. The spikes can either be streamed in directly or they can be generated using mean-rate spike generators. The output of the network are the spikes at the output layer. There are three ways to monitor the behavior of the output layer. The literal spikes can be streamed through a buffer and the Inter-Spike Interval (ISI), and a low-pass filtered membrane potential is computed for each neuron at the output layer.

In our design, each weight memory contains one or more true dual-port on-chip memories. Being true dual-port ensures that two layers can access the same weight memory simultaneously, so each layer has independent access to weights. Furthermore, weights can be updated after synthesis at run time since the memories are writable. This avoids a potentially long synthesis and implementation time.

3.1 Digital Spiking Neuron

An adapted version of the event-driven neuron update algorithm from [11] is used as a basis for our implementation. Since the layers are fully-connected, all neurons in one layer are updated simultaneously and the previous spike times are always identical for all neurons in a layer. To save storage, the algorithm is adapted by replacing the individual previous spike times by a single previous spike time that is shared over all neurons in a layer. Besides, we implemented a statements that avoid negative values for the membrane potential. While a time-driven algorithm loops through all time steps, our event-driven algorithm loops through spike events. This ensure that our computations only happen on-demand, i.e. upon spiking activity.

The membrane potential of each neuron i decays with

$$V_m^i = V_m^i \cdot e^{-(t-t_{prev}^i)/\tau_m} \tag{4}$$

which requires the computation of the exponential function. Calculating the exponential function is resource and time expensive. To simplify this computation, the exponential function is approximated using bit-shifts. By shifting the membrane potential by n bits to the right, its value is divided by 2^n and the least significant bits are eliminated. This is a resource efficient operation in digital hardware. Figure 5 shows the dataflow for one neuron, except for subtraction $t - t_{prev}$ which is shared over all neurons in one layer.

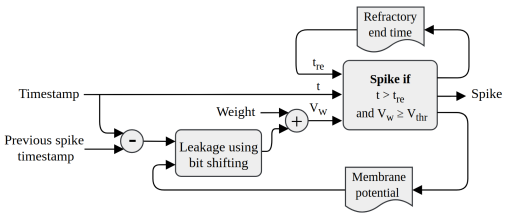

Fig. 5. LIF neuron behavior

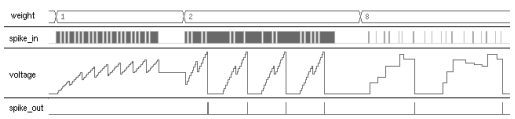

Fig. 6. LIF RTL Simulation

By moving the computation of the previous spike time outside the neuron and simplifying the computation of the exponential function, the neuron is implemented such that it processes an input spike in two clock cycles without the use of a pipeline. Figure 6 shows the neuron behavior using RTL simulation waveforms. The event-driven approach appears from the observation that the membrane potential, indicated as voltage, remains unchanged when input spikes are absent. The refractory period causes the membrane potential to remain zero after an output spike even though new input spikes arrive. Furthermore, every now and then the membrane potential leaks away by a noticeable amount. In the period where the weight has value 1, the membrane potential gets into an equilibrium as the potential increases as much as it leaks away.

3.2 Weight memory mapping

Each of the weight memory modules stores a weight matrix that represents all the weight values of the connections between two layers (as shown in Fig. 3). The major challenge of implementing the SNNs on digital hardware is to efficiently map the connection matrices to the on-chip memories. The way the weights are distributed over the memories determines the memory bandwidth and ultimately the system throughput. Additionally, this mapping determines the implementation of the weight controller.

To visualize the impact of a weight mapping on memory bandwidth, an example weight set for a four by four network is shown in Figure 7 (right). The corresponding weight matrix is shown in

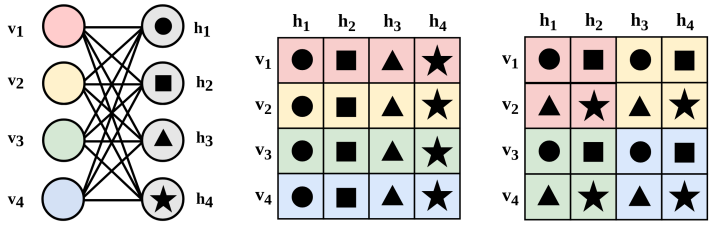

Fig. 7. Left: Example a two layer network. Middle: Weight mapping requiring either one or four memory accesses. Right: Weight mapping always requiring two memory accesses

Figure 7. When this structure is mapped to memory, one row can be read per clock cycle. When a neuron v_n spikes, each of the four neurons h_m require to be updated and the corresponding, equally colored, weights can be read in one clock cycle. The resulting throughput is four neuron updates per clock cycle. On the contrary, when a neuron h_n spikes, each of the four neurons v_m require to be updated and the corresponding weights, which contain the same shape, can be read in four consecutive clock cycles. As it effectively reads one weight per clock cycle, the resulting throughput is only one neuron update per clock cycle.

To make a more fair trade-off between forward and backward weight accesses, the weights can be mapped differently to the memory as shown in Figure 7. In this case, the weights corresponding to any neuron v_n or h_n are on two different lines. Thus, both a forward and backward spike require two consecutive reads and have a throughput of two neuron updates per clock cycle.

To generalize the mapping of any weight matrix between V visible neurons and H hidden neurons to a set of memories, a three dimensional x, y, z coordinate system is introduced. Variables x and y represent the memory width and depth respectively and z represents the distinctive memories. All variables x, y and z are integers. The memory width is represented as a number of weights, so the memory width in bits is obtained by multiplying x by the weight bit width. The core idea of the memory mapping is that the set of H weights that correspond to a visible neuron v_n are grouped into a cube of size x_1 by y_1 by z_1. The volume of this cube equals the amount of hidden neurons, so $x_1 \cdot y_1 \cdot z_1 = H$. This cube is repeated x_2 times in the x dimension, y_2 times in the y dimension and z_2 times in the z dimension. As there is a cube for each visible neuron, the amount of repetitions equals the amount of visible neurons, so $x_2 \cdot y_2 \cdot z_2 = V$. To update the set of hidden neurons upon a forward spike, x_1 by y_1 weights are read from z_1 memories.

This is visualized in Figure 8 (left) where the weights belonging to a hidden neuron are colored pink. Similarly, to update the set of visible neurons upon a backward spike, x_2 by y_2 weights are read from z_2 memories. This is visualized in Figure 8 (right) where the weights belonging to a visible neuron are colored green.

4 RESULTS

4.1 Implementation

The design is implemented in the hardware description language VHDL and evaluated on Xilinx FPGAs. One of the important design metrics is the amount of FPGA resources that are required to implement a given architecture. The resource utilization in terms of Lookup Tables (LUTs), Flip-Flops (FFs), BRAMs and DSPs is shown in Table 1 for different network sizes. The numbers only include the resources for the SNN, so external logic such as reset and AXI infrastructure logic is excluded. The networks use DSP blocks because of the IIR filter of each neuron at the output

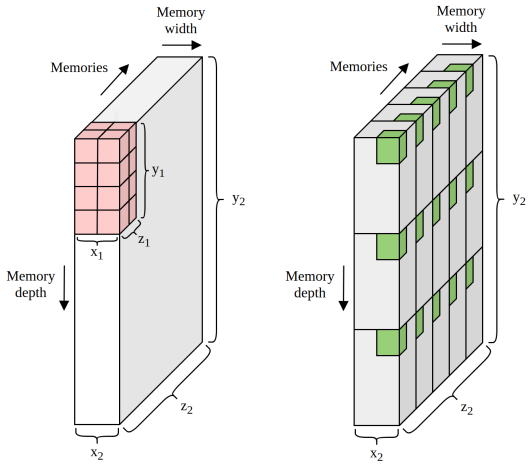

Fig. 8. Left: Set of $x_1 \cdot y_1 \cdot z_1$ pink colored weights for a forward spike. Right: Set of $x_2 \cdot y_2 \cdot z_2$ green colored weights for a backward spike.

layer. The IIR filters are used to monitor a low-pass version of membrane potential of all the output neurons.

Network	LUTs	FFs	BRAMs	DSPs
(a) 784-330-330-10	53,466	48,310	100	10
(b) 784-720-720-720-10	140,206	131,977	306	10
(c) 102-600-600-600-7	101,583	104,738	170	7

Table 1. Resource utilization for networks implemented in two FPGAs (a,b) Trenz TE0820-03-4DE21FA, (c) Trenz TE0808-04-09EG-1EE. Networks are fully connected with 4 (a) and 5 (b,c) layers.

4.2 Power

For a measure of the power dissipation of the design, the estimation from Vivado is used. The toggle rate is 12,5% indicating a toggle every 8 clock cycles, which is in agreement with the sparsity of SNNs. The static probability is 50%. The distribution of the power consumption is listed for different network sizes in Table 2. The design is implemented on a SoC that includes an embedded PS. However, because the SNN is implemented in PL while the PS is only used to interface with the SNN, the power consumption of the PS is excluded. The power consumption positively correlates with the network size and the amount of resources used as shown in Table 1.

Network size	(a)	(b)	(c)
FPGA	ZU4EV	ZU9EG	ZU9EG
Clock freq. (MHz)	250	250	250
Clocks (mW)	233	216	174
Signals (mW)	542	357	214
Logic (mW)	272	209	144
BRAM (mW)	473	505	318
DSP (mW)	16	6	5
Static (mW)	472	746	740
Total (mW)	2008	2039	1595

Table 2. Estimated PL power dissipation for three feed-forward networks.

4.3 Energy efficiency

The absolute resource utilization, throughput or power dissipation are impractical for comparison with other implementations since they highly depend on the used network size. Therefore, the peak throughput and power dissipation are combined to create a measure of the energy efficiency in unit nanoJoules per synaptic operation (nJ/SO). The energy efficiency is listed for different network sizes and clock frequencies in Table 3.

Network size	(a)	(b)	(c)
Clock frequency (MHz)	250	250	250
Peak throughput (GSOPS)	13,32	40,71	31,82
Power dissipation (mW)	2008	2039	1595
Energy Efficiency (GSOPS/W)	6,63	20,45	19,95
Energy efficiency (nJ/SO)	0,151	0,050	0,050

Table 3. Energy efficiency at peak throughput.

4.4 Cropland mapping using optical-radar sensory fusion

We use a public dataset for cropland classification using optical-radar data [6]. The dataset contains a large set of numerical features derived from remote sensing information from two primary sources, i) an optical information (images) was collected by RapidEye satellites ii) radar-based information was collected by an Uninhabited Aerial Vehicle Synthetic Aperture Radar (UAVSAR) system over an agricultural region near Winnipeg, Manitoba, Canada on 2012. There are 2x49 radar features and 2x38 polarimetric radar features for two dates: 5th and 14th July 2012. The dataset contains labels of seven crop types: Corn, Peas, Canola, Soybeans, Oats, Wheat, and Broadleaf. Class distribution in the cropland dataset is highly unbalanced and it is as follow: corn 12.02%, Peas 1.10%, Canola 23.22%, Soybeans 22.73%, Oats 14, 46%, Wheat 26.12%, and Broadleaf 0.35%.

The dataset contains 174 features. However, after a feature correlation analysis we filtered 72 features as they show high correlations ($> 0,95$) with other features. The resulting number of optical-radar features is 102. In FPGA, we have used network **c** in Table 1. The dataset has been split in 80% training and 20% testing maintaining the same class distribution as in the dataset. We used 260667 pixels for training and 65167 pixels for testing.

Training SNNs is linked to some difficulties that have to deal with the gradient calculation and the network's temporal dynamics. This is mainly caused by the discontinuous nature of the neuron spiking function (i.e. the Kronecker delta function). To overcome this challenge, we have

Table 4. Confusion matrix on testing set

	Corn	Pea	Can.	Soy	Oat	Whe.	Bro.
Corn	7855	2	2	17	1	3	1
Pea	1	698	3	1	0	2	0
Canola	5	0	15225	10	4	1	1
Soybeans	24	3	18	14806	15	18	0
Oat	5	0	9	25	9067	86	0
Wheat	5	0	15	26	234	16798	0
Broadleaf	6	1	3	3	11	11	209

trained an equivalent analog neural network and converted it in a spiking neural network that uses mean rate to encode neuron's activation, as in [12]. During training, we also performed quantization of the weights to match the limited on-chip memory bit width. We used the PyTorch deep-learning framework to obtain weight values compatible with the 6-bits precision of our FPGA implementation. In FPGA, we used 10 bits in fixed point format to represent the V_{mem} of each neuron. The weights are signed with 6-bit resolution. During training, we clip the weight in the range $[-1, 1]$. After training,the weights are scaled so that the range $[-1, 1]$ maps to the integer range $[-31, 31]$. Training boosts the output neuron encoding the correct class to spike as much and as early as possible, which often happens before the majority of the hidden layer neurons have even spiked. The output of the network is evaluated using a single measure of inter-spike-interval (ISI). ISI is defined as the difference in time among two consecutive spikes. The output neuron that has the smallest ISI is considered to be the correct output class, and the network can proceed to compute the next input. Table 4 shows the results of the classification on the test set. Previous classification of the same dataset using a random forest classifier obtained 99,03% accuracy [7], [8]. Our hardware neural network approach improves over previous results by achieving an overall accuracy of 99.7%.

5 CONCLUSIONS

We presented an event-driven architecture for implementing spiking neural networks on digital hardware. On-chip memories are used to store the synaptic weights to ensure a high memory bandwidth as opposed to off-chip memory. A method is presented to map the synaptic weights to on-chip memories facilitating both feed-forward and feed-back networks. Furthermore, a hardware efficient LIF neuron model has been implemented to avoid the necessity of neuron time-multiplexing. The architecture is implemented and evaluated on various Xilinx FPGAs and it has been characterized using theoretical and empirical analysis. With a feed-forward network of 2954 neurons and 1,608,440 synapses we achieved a peak throughput of 40,71 GSOPS. The energy efficiency of Gyro varies depending on network size, mapping parameters, and clock frequencies, from 0,050 nJ/SO to 0,151 nJ/SO. These results, to our knowledge, show a >10x times more efficient implementation than the most related works [4, 9, 18] in which the best efficiency reported is > 2 nJ/SO with a best peak throughput of about 20 GSOPS. The effectiveness of Gyro has been demonstrated in a sensory fusion cropland classification task. We achieved an accuracy of 99,7%. This result is superior to a random forest algorithm previously presented in [7] and executed in software. Gyro has a parametric interface that is compatible with standard digital workflow and can be easily ported to different FPGAs. Given our analysis and results, we believe that Gyro is a first step towards the application of event-based information processing systems for drone data analytics.

ACKNOWLEDGMENTS

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826610. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Austria, Belgium, Czech Republic, France, Italy, Latvia, Netherlands.

REFERENCES

[1] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 1 (2018), 82–99.

[2] Peter Dayan and Laurence F Abbott. 2001. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series.

[3] Michael V DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K Nayak, Rathinakumar Appuswamy, et al. 2019. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer* 52, 5 (2019), 20–29.

[4] J. Han, Z. Li, W. Zheng, and Y. Zhang. 2020. Hardware implementation of spiking neural networks on FPGA. *Tsinghua Science and Technology* 25, 4 (2020), 479–486.

[5] Mahmoud Hussein, Réda Nouacer, Yassine Ouhammou, Eugenio Villar, Federico Corradi, Carlo Tieri, and Rodrigo Castiñeira. 2020. Key Enabling Technologies for Drones. In *2020 23rd Euromicro Conference on Digital System Design (DSD)*. IEEE, 489–496.

[6] Iman Khosravi. 2018. *Crop mapping using fused optical-radar data set Data Set.* https://archive.ics.uci.edu/ml/datasets/Crop+mapping+using+fused+optical-radar+data+set

[7] Iman Khosravi and Seyed Kazem Alavipanah. 2019. A random forest-based framework for crop mapping using temporal, spectral, textural and polarimetric observations. *International Journal of Remote Sensing* 40, 18 (2019), 7221–7251.

[8] Iman Khosravi, Abdolreza Safari, and Saeid Homayouni. 2018. MSMD: maximum separability and minimum dependency feature selection for cropland classification from optical and radar data. *International Journal of Remote Sensing* 39, 8 (2018), 2159–2176.

[9] I. Kiselev, D. Neil, and S. Liu. 2016. Event-driven deep neural network hardware system for sensor fusion. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2495–2498. https://doi.org/10.1109/ISCAS.2016.7539099

[10] Hesham Mostafa, Bruno U Pedroni, Sadique Sheik, and Gert Cauwenberghs. 2017. Fast classification using sparsely active spiking networks. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–4.

[11] D. Neil and S. Liu. 2014. Minitaur, an Event-Driven FPGA-Based Spiking Network Accelerator. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22, 12 (Dec 2014), 2621–2628. https://doi.org/10.1109/TVLSI.2013.2294916

[12] Peter O'Connor, Daniel Neil, Shih-Chii Liu, Tobi Delbruck, and Michael Pfeiffer. 2013. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience* 7 (2013), 178.

[13] Danilo Pani, Paolo Meloni, Giuseppe Tuveri, Francesca Palumbo, Paolo Massobrio, and Luigi Raffo. 2017. An FPGA Platform for Real-Time Simulation of Spiking Neuronal Networks. *Frontiers in Neuroscience* 11 (2017), 90. https://doi.org/10.3389/fnins.2017.00090

[14] Harry A Pierson and Michael S Gashler. 2017. Deep learning in robotics: a review of recent research. *Advanced Robotics* 31, 16 (2017), 821–835.

[15] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. 2019. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience* 13 (2019), 95.

[16] Georgios Smaragdos, Sebastian Isaza, Martijn F van Eijk, Ioannis Sourdis, and Christos Strydis. 2014. FPGA-based biophysically-meaningful modeling of olivocerebellar neurons. In *Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays*. 89–98.

[17] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. 2019. Deep learning in spiking neural networks. *Neural Networks* 111 (2019), 47–63.

[18] Runchun M Wang, Chetan S Thakur, and André van Schaik. 2018. An FPGA-based massively parallel neuromorphic cortex simulator. *Frontiers in neuroscience* 12 (2018), 213.