Design of Time-Encoded Spiking Neural Networks in 7-nm CMOS Technology

Sandro Widmer, Marcel Kossel, *Senior Member, IEEE*, Giovanni Cherubini, *Life Fellow, IEEE*, Stanisław Woźniak, *Member, IEEE*, Pier Andrea Francese, *Senior Member, IEEE*, Ana Stanojevic, *Member, IEEE*, Matthias Brändli, *Member, IEEE*, Klaus Frick, and Angeliki Pantazi, *Senior Member, IEEE*

***Abstract*—In biologically inspired spiking neural networks (SNNs) neurons communicate by short pulses, called spikes. SNNs have the potential to be more power efficient than artificial neural networks (ANNs), thanks to the fewer computational steps required by the spike transmission and processing, as compared to the multiply-and-accumulate (MAC) operations with wide bit-vectors usually adopted in ANNs. We present the design of two types of SNNs with integrate-and-fire dynamics and single-spike per neuron operation, where neural communication is based on synchronous time-to-first-spike (sTTFS) and time-to-first-spike (TTFS) encoding schemes. In the considered time-encoded SNNs, the information is carried by the timing of the spikes with respect to a reference time. In 7nm CMOS technology both designs are synthesized as VHDL-based random-logic-macros (RLMs) and compared to an equivalent ANN design in terms of power consumption, latency and silicon area, using the Iris data set for inference. A cost function expressed as a product of energy consumption and silicon area is introduced to compare the three network designs. With respect to this cost function, it turns out that the SNN-TTFS implemented for the considered classification task outperforms the ANN used as baseline model.**

***Index Terms*—Spiking neural networks (SNNs), artificial neural networks (ANNs), synchronous time-to-first-spike (sTTFS) encoding, time-to-first-spike (TTFS) encoding.**

I. Introduction

THE OBJECTIVE of spiking neural networks (SNNs) is to reproduce a behavioral model of the sophisticated chemical processes occurring in our brain. This is achieved by means of electrical signal processing and neuronal transmission based on encoding information as sequences of short pulses, called spikes, and by applying neuronal dynamics, here modeled as integrate-and-fire dynamics. Fig. 1 illustrates a simplified diagram of a biological neuron and of the events associated with neuronal activity. The neuronal characteristics serve as inspiration for how SNNs are setup and operate, as the incoming synaptic signals are received by dendrites building up a membrane potential that triggers the emission of spikes once a threshold value is reached. Motivated by neuroscientific experiments suggesting that neuronal information is conveyed by the temporal position of spikes [1], [2], [3], recent research on SNNs focused on temporal coding schemes for neuronal transmission [4], [5], [6], [7].

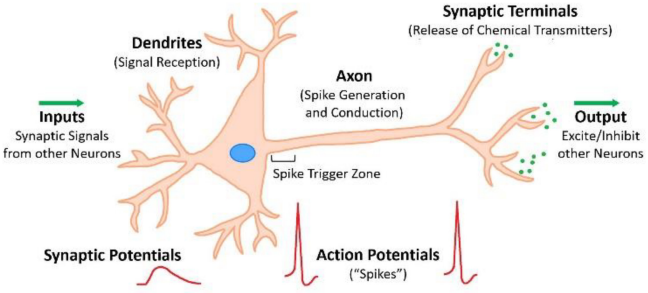

Fig. 1. Basic functionality of a neuron cell applying spike transmission.

Temporal coding in SNNs enables faster inference and requires fewer spikes than other coding techniques such as, e.g., rate coding, thus leading to efficient network operation.

Prior work on the hardware implementation of SNNs, where the transmitted neural information is encoded in a temporal manner, is found in [8], [9], [10]. However, no attempt has been made to compare VLSI designs of ANNs and time-encoded SNNs in term of area, latency, and power consumption.

In this brief, we present designs of two neuronal communication systems for SNNs that can perform classification tasks and are based on single-spike-per-neuron operation with time coding. Specifically, we consider 1) synchronous time-to-first-spike (sTTFS) encoding, and 2) time-to-first-spike (TTFS) encoding, as illustrated in Fig. 2 for a simplified network topology consisting of three neurons, depicted at the top. In sTTFS the information is encoded by the time duration (e.g., $\Delta t_1 = T - t_1$, see Fig. 2a) occurring from the event of a spike to the end of an observation interval T, which is defined as the

Manuscript received 23 March 2023; accepted 15 May 2023. Date of publication 18 May 2023; date of current version 29 August 2023. This brief was recommended by Associate Editor H. Park. *(Corresponding author: Marcel Kossel.)*

Sandro Widmer was with the Hybrid Cloud Research, IBM Research GmbH, 8803 Rüschlikon, Switzerland, and also with the ICE Institut für Computional Engineering, OST-Eastern Switzerland University of Applied Sciences, 9000 St. Gallen, Switzerland. He is now with the Ultra-Precision Manufacturing Lab, RhySearch, 9471 Buchs, Switzerland (e-mail: sandro.widmer@rhysearch.ch).

Marcel Kossel, Giovanni Cherubini, Stanisław Woźniak, Pier Andrea Francese, Ana Stanojevic, Matthias Brändli, and Angeliki Pantazi are with the Hybrid Cloud Research, IBM Research GmbH, 8803 Rüschlikon, Switzerland (e-mail: mko@zurich.ibm.com; giovanni.cherubini@ibm.com; stw@zurich.ibm.com; pfr@zurich.ibm.com; ans@zurich.ibm.com; mbr@zurich.ibm.com; agp@zurich.ibm.com).

Klaus Frick is with the ICE Institut für Computional Engineering, OST-Eastern Switzerland University of Applied Sciences, 9471 Buchs, Switzerland (e-mail: klaus.frick@ost.ch).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2023.3277784.

Digital Object Identifier 10.1109/TCSII.2023.3277784

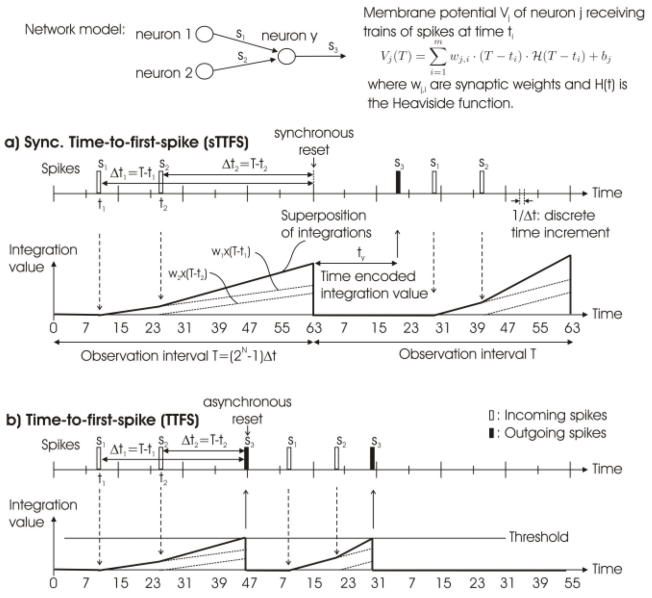

Fig. 2. Temporal encoding schemes for neuronal communications: a) sTTFS and b) TTFS.

product of a time step Δt, given by the inverse of a global clock, and a maximal number of time steps (2^N-1), where N is the resolution of the data signal (e.g., N=6 bits, see Fig. 4).

The multiplication (e.g., $\Delta t_1 w_1$, see Fig. 2a) of the incoming data Δt_1 (expressed as time-encoded information) with the synaptic weight w_1 is performed by an integration, whose value reproduces the evolution of the membrane potential in the biological counterpart. At the end of the observation interval the integration value determines the time instant, at which a spike is transmitted during the next observation interval. In other words, a spike instant represents a time-encoded value, where the transmission delay (e.g., t_y, see Fig. 2a) denotes the integration value obtained at the previous observation interval. The synchronism in sTTFS refers to the encoding process. Each neuron is provided with an initial reference time and a final reference time to determine the observation interval for the evaluation of the time to spike. The reference times are common for all neurons in a layer. The inference evolves synchronously and stops after the observation interval associated with the output layer has elapsed. The latency is thus determined by the observation interval duration and the number of layers.

As opposed to sTTFS, where the evaluation of the integration value occurs synchronously at the end of each observation interval, in TTFS a spike is transmitted whenever the integration value reaches a given threshold (see Fig. 2b). The reference time in TTFS indicates the beginning of the inference, when feature values are presented to the input layer neurons. Afterwards, the inference evolves asynchronously and stops when the first spike at the output layer neurons is observed. The latency is given by the difference between the decision time and the reference time. Because of this behavior, TTFS is characterized by an asynchronous operation that may lead to shorter latency and higher throughput than the sTTFS approach.

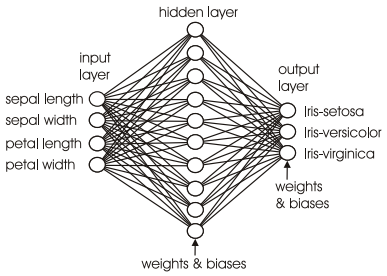

Fig. 3. Implemented network topology. The circles represent the neurons of the network. The edges denote the communication channels between the neurons.

This brief is organized as follows. Section II describes the implemented network topology and the chosen data set for network training and inference. The physical designs of the ANN and of the time-encoded SNNs are presented in Section III. The performance analysis and the comparison of the various designs in terms of latency, silicon area, and power consumption are given in Section IV. The network designs are also compared in terms of a cost function, which is expressed as a product of energy consumption and silicon area. This brief concludes with Section V.

II. Data Set and Network Topology

To keep the implementation effort for a test chip feasible, the Iris data set containing 150 records of Iris flowers is chosen for training and inference of an ANN and two SNNs. For each of these networks the physical design is performed by means of a random-logic-macro (RLM). The Iris data set includes three flower species (Iris-virginica, Iris-versicolor, Iris-setosa) characterized by the four dimensions: sepal width, sepal length, petal width and petal length, all expressed in cm.

The network topology for the Iris data set classification used in this brief is shown in Fig. 3. The input layer consists of four neurons, one for each dimension. This is followed by a hidden layer with 10 neurons. The output layer has three neurons associated with the three species. The neurons in the various layers are fully connected. In total there are 70 weights and 13 biases associated with the hidden layer and the output layer.

In a first step the network topology of Fig. 3 is modelled as a quantized ANN model in TensorFlow to determine the smallest possible model with negligible loss of accuracy. The trained ANN model serves as a baseline model for the comparison with SNNs. The trained weights and biases of the ANN are then used for the modeling and verification of the SNN-sTTFS, which achieves the same accuracy as the ANN if floating-point operations are assumed. The SNN-TTFS is trained using an algorithm tailored to its asynchronous operation [4]. Fig. 4 shows in two tables the accuracy of the sTTFS and TTFS models when sweeping the quantization of the weights and the time-encoded communication signals (referred to as data in the plot). A fixed-point arithmetic is chosen as indicated by the numbers in parenthesis at the labeling of the table axis. The first number in the bracket indicates the total number of bits, the second number refers to the number of bits used

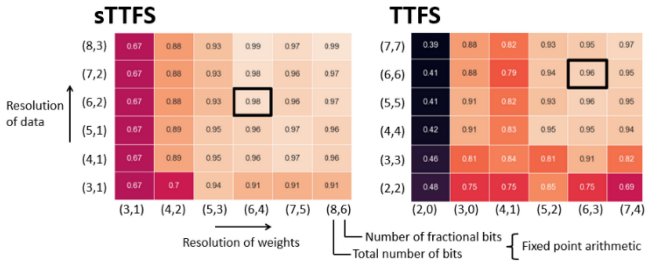

Fig. 4. Accuracy tables versus quantization of data and weights. The color code illustrates different ranges of accuracy.

to represent the fractional part (in short, fractional bits). The encircled accuracy indicates the smallest model with negligible accuracy loss chosen for the physical design. For sTTFS the weights are quantized to 6 bits, whereas 4 of them are fractional bits. The data signal has the same resolution but only 2 fractional bits. TTFS also uses 6-bit resolution, but with 3 fractional bits for the weights and 6 fractional bits for the data. The same 6-bit data and weight resolutions have been chosen for both the sTTFS and TTFS designs. To obtain the smallest possible models with negligible loss of accuracy, different number of fractional bits have been selected for the data and weight representations.

III. Physical Design

Fig. 5 shows the neuron models for the three implemented network types: ANN, SNN-sTTFS, and SNN-TTFS. The depicted neuron models represent the implementation of one of the 10 neurons in the hidden layer shown in Fig. 3. For the input and output layers simplified versions of the neurons of Fig. 5 are used.

In the ANN neuron the incoming data x_0 to x_k are first latched and multiplied by the pertinent weight w_j. The index i of the receiving neuron is omitted for simplicity. The products $w_j x_j$ are then summed up by a carry-save-adder (CSA) tree. CSA has no rippling of the carry signal and hence a smaller power consumption than equivalent ripple-carry-adders (RCAs). The output of the adder tree is fed to another adder that adds the bias b_j followed by a round half up operation to yield the result with the resolution of the comparator, which acts as a limiter if the previously calculated sum is outside the range between 0 and a maximum value. The result is then quantized to 6 bits and latched before being sent out as the output signal y_i.

The sTTFS neuron model of Fig. 5 is explained in conjunction with the timing diagram of Fig. 2. The incoming signals x_0 to x_k represent spikes that occur at respective time instants t_0 to t_k, as illustrated for example in Fig. 2 by s_1 and s_2. In each synaptic receiver a spike detector controls a selector that either outputs zero (if no spike is detected) or the weight w_j (if a spike is detected in x_j). The synaptic receiver outputs are summed up by a CSA tree whose output goes to an RCA. This adder acts as an integrator that produces the time-discrete waveforms of the ramp signals in Fig. 2 (drawn as continuous linear waveforms for simplicity), whose biological counterpart is the membrane potential in the neuron of

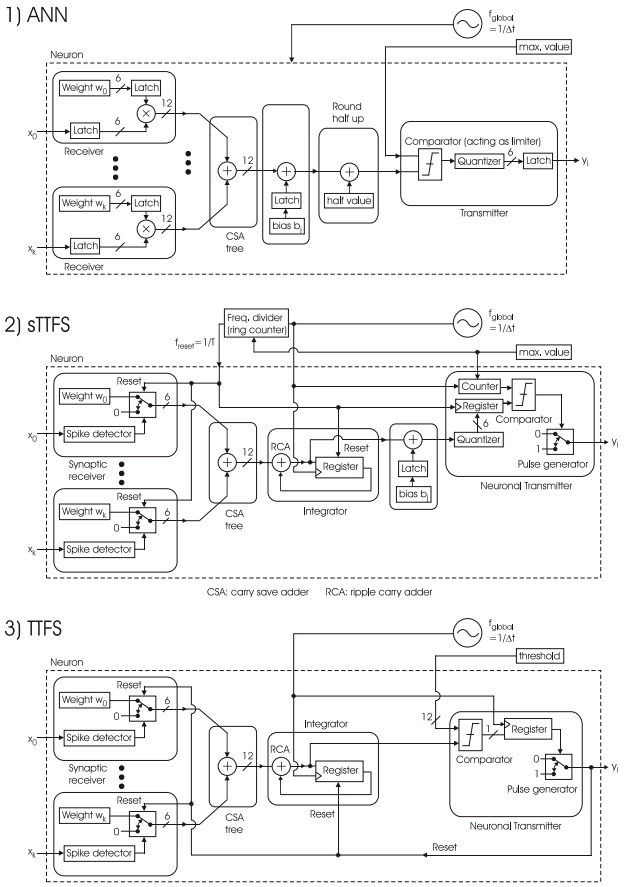

Fig. 5. Neuron models for the implementation of the neural network types: 1) ANN, 2) SNN-sTTFS and 3) SNN-TTFS.

Fig. 1. The integrator clock frequency is $f_{global}=1/\Delta t$, where Δt denotes the time increments on the time axis in Fig. 2. Note that the multiply-and-accumulate (MAC) operation $\Sigma w_j x_j$ of the ANN is equivalent to adding w_j during $T-t_j$ in time steps of Δt to the register with RCA feedback. In other words, the MAC operation of the ANN is mathematically equivalent to the outlined digital integrator in the SNN-sTTFS of Fig. 5. Analogously to the ANN, the integrator output is fed to a successive adder for the addition of the bias b_j before the integration value is quantized to 6 bits. The spike signal y_i to be sent out represents the time-encoded integration value. The conversion from the digital domain (integration value) to the time domain (spike signal) is performed by a counter clocked with f_{global} that counts downwards, i.e., large values lead to early spikes, until the value of the quantized integration value is reached. This is detected by a comparator that controls a pulse generator producing the spike signal y_i. sTTFS encoding pursues a synchronous operation, which means that the weight selectors of the synaptic receivers, the integrator as well as the 6-bit register used to hold the integrator value in the transmitter are reset periodically, i.e., at the end of the observation interval T determined by the frequency divider, which is implemented as a ring counter (see Fig. 2).

The TTFS neuron depicted at the bottom of Fig. 5 has the same synaptic receivers as explained above for the sTTFS neuron. Also, the successive CSA tree and integrator are identical

TABLE I
PERFORMANCE METRICS OF RLMS IMPLEMENTED

Model	Clock speed [GHz]	Latency [ns]	Power [mW]	Energy [pJ]	Width [um]	Height [um]	Area [um^2]	Usage	Area used [um^2]	Cost [10^3] (proposed FOM)
ANN	1.25	4.8	2.38	11.4	101.76	101.088	10'286	39.2%	4032	46.1
SNN-sTTFS	1.25	153.6	1.30	199.8	49.92	50.544	2'523	42.1%	1062	212.1
SNN-TTFS	1.25	42.4 (50.4)	1.21	51.4 (61.1)	44.16	45.36	2'003	41.4%	828	42.5 (50.6)
SNN-sTTFS	2.667	72.0	2.42	174.8	49.92	50.544	2'523	42.1%	1062	185.6
SNN-TTFS	2.667	21.0 (24.8)	2.62	55.1 (65.1)	44.16	45.36	2'003	41.4%	828	45.7 (54.0)
(…) maximum possible value in asynchronous TTFS operation										

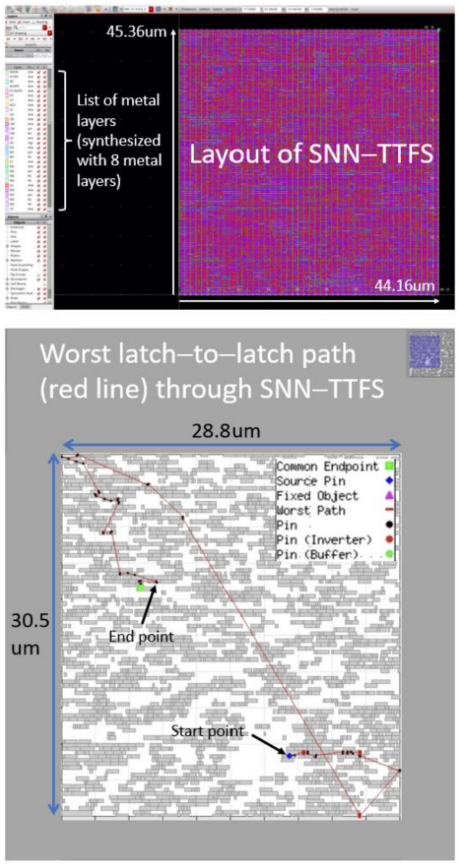

Fig. 6. Layout of SNN-TTFS in 7nm CMOS technology (top) and pertinent worst latch-to-latch path analysis (bottom).

to those of the sTTFS neuron. The main difference between sTTFS and TTFS is related to the implementation of the neural transmitter, because TTFS pursues an asynchronous operation where a spike is emitted immediately once the integration value reaches a given threshold value. This is implemented by a comparator whose output is retimed via a 1-bit register to f_{global} and controls the pulse generator that emits the outgoing spike signal y_i. The transmitter output is being used as reset signal for the weight selector of the synaptic receiver and the integrator.

The neural network topology shown in Fig. 3 is coded in VHDL with the three different neuron implementations depicted in Fig. 5. The physical design is performed in 7nm CMOS technology. As an example, the layout of the SNN-TTFS as well as its worst latch-to-latch path are shown in Fig. 6. The same synthesis and physical design flow is also applied to ANN and SNN-sTTFS. A performance analysis (latency, power consumption) is performed in Cadence with the Spectre simulator using the RC extracted netlists of the individual RLMs with the trained weights and biases taken from the TensorFlow network simulations.

IV. PERFORMANCE ANALYSIS

One performance metric is the area of the neural network implementation, which is listed in Table I for a target utilization of roughly 40% (see columns Area, Usage and Area used where the latter is Area times Usage). The two SNNs consume 4x less area than the ANN, which is to be ascribed to the very fast but area-consuming multipliers in the ANN.

Some other electrical performance metrics depicted in Table I are the latency as well as the power and energy consumption obtained by Spectre simulations in Cadence with the RC extracted netlist of the RLMs. Since the MAC operation of the ANN is replaced by integrators in the SNNs, the latency of the SNNs is much larger than that of the ANN because the observation interval needs to be resolved into finer time steps $\Delta t = 1/f_{global}$. Conversely, the power consumption of the SNNs is smaller than that of the ANN, as the MAC operation of the ANN exhibits arithmetic ripples due to the ripple carry operations, a phenomenon that is less pronounced in the case of the digital integrator of the SNNs. However, the smaller power consumption of the SNN does not compensate the larger latency, which means that the energy $E = P \cdot t$ consumed by SNNs is higher than that of the ANN.

Note that the SNN-TTFS rows include values written in parenthesis. They represent the maximum latency of the asynchronous TTFS operation (obtained by a statistical analysis), whereas the other numbers denote the average values of the specific test case run.

Focusing only on the area or on the electrical performance metrics alone would not lead to a meaningful figure of merit (FOM), as they are all equally important. To obtain a fair comparison we therefore introduce the following cost function

$$FOM = Energy \cdot Area \tag{1}$$

Table I shows in the last column the proposed FOM to compare the different neural network implementations. The SNN-TTFS outperforms the SNN-sTTFS, which is negatively impacted by the larger latency of the synchronous operation. It turns out that the FOM of the SNN-TTFS is also slightly better than that of the ANN. One important aspect is that the speed

of both SNNs is much higher than that of the ANN, which is limited to 1.25 GHz, owing to the larger size that leads to timing violations at higher clock frequencies. The higher speed of 2.667 GHz (derived from double-data-rate (DDR) generation 5 (DDR5) data rates for memory links) for the SNNs is set for project specific reasons and does not represent the upper speed limit of these RLMs. For a fair comparison with the ANN, the SNNs are also synthesized at 1.25 GHz.

V. Conclusion and Outlook

A design study on the implementation of neural networks for the classification of the Iris flower data set in 7nm CMOS technology is presented. An ANN implementation is used as baseline model for comparison with the biologically inspired SNNs that use either synchronous (sTTFS) or asynchronous (TTFS) operation. With respect to the proposed FOM that takes the energy consumption and the area into consideration, it turns out that the SNN-TTFS outperforms the other neural network implementations.

SNNs are appealing because of their smaller size and reduced power consumption compared to the baseline ANN. However, the replacement of the MAC operation by a digital integrator increases both the latency and the energy consumption. Further work on the optimization of SNNs needs to focus on the latency reduction. One approach would be to reduce the timing granularity, leading to a decimation in time, which would yield a reduced latency. Another avenue would rely on the introduction of analog components to implement the neural integrate-and-fire dynamics, leading to a hybrid design with the potential for reduced latency and power consumption. However, an analog-circuitry based approach would be more challenging in terms of linearity and process variations than the fully digital approach presented in this brief.

References

[1] M. F. Kubke, D. P. Massoglia, and C. E. Carr, "Developmental changes underlying the formation of the specialized time coding circuits in barn owls (Tyto alba)," *J. Neurosci.*, vol. 22, no. 17, pp. 7671–7679, 2002.

[2] T. Gollisch and M. Meister, "Rapid neural coding in the retina with relative spike latencies," *Science*, vol. 319, no. 5866, pp. 1108–1111, 2008.

[3] R. S. Johansson and I. Birznieks, "First spikes in ensembles of human tactile afferents code complex spatial fingertip events," *Nat. Neurosci.*, vol. 7, no. 2, pp. 170–177, 2004.

[4] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3227–3235, Jul. 2018.

[5] B. Rueckauer and S.-C. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–5.

[6] I. M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, "Temporal coding in spiking neural networks with alpha synaptic function," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2020, pp. 8529–8533.

[7] A. Stanojevic, E. Eleftheriou, G. Cherubini, S. Woźniak, A. Pantazi, and W. Gerstner, "Approximating ReLU networks by single-spike computation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2022, pp. 1901–1905.

[8] M. Bouvier et al., "Spiking neural networks hardware implementations and challenges: A survey," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 2, pp. 1–35, 2019.

[9] C.-H. Chien, S.-C. Liu, and A. Steimer, "A neuromorphic VLSI circuit for spike-based random sampling," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 135–144, Jan.–Mar. 2018.

[10] T. Mesquida, A. Valentian, D. Bol, and E. Beigne, "Impact of the AER-induced timing distortion on spiking neural networks implementing DSP," in *Proc. 12th Conf. Ph.D. Res. Microelectron. Electron. (PRIME)*, 2016, pp. 1–4.