PAPER • OPEN ACCESS

Integrate-and-fire circuit for converting analog signals to spikes using phase encoding[*]

To cite this article: Javier Lopez-Randulfe *et al* 2023 *Neuromorph. Comput. Eng.* **3** 044002

View the article online for updates and enhancements.

NEUROMORPHIC
Computing and Engineering

CrossMark

PAPER

Integrate-and-fire circuit for converting analog signals to spikes using phase encoding[*]

Javier Lopez-Randulfe[**] [iD], Nico Reeb [iD] and Alois Knoll [iD]

TUM School of Computation, Information and Technology, Technical University of Munich, Munich, Germany
[**] Author to whom any correspondence should be addressed.

E-mail: lopez.randulfe@tum.de

Abstract

Processing sensor data with spiking neural networks on digital neuromorphic chips requires converting continuous analog signals into spike pulses. Two strategies are promising for achieving low energy consumption and fast processing speeds in end-to-end neuromorphic applications. First, to directly encode analog signals to spikes to bypass the need for an analog-to-digital converter. Second, to use temporal encoding techniques to maximize the spike sparsity, which is a crucial parameter for fast and efficient neuromorphic processing. In this work, we propose an adaptive control of the refractory period of the leaky integrate-and-fire (LIF) neuron model for encoding continuous analog signals into a train of time-coded spikes. The LIF-based encoder generates phase-encoded spikes that are compatible with digital hardware. We implemented the neuron model on a physical circuit and tested it with different electric signals. A digital neuromorphic chip processed the generated spike trains and computed the signal's frequency spectrum using a spiking version of the Fourier transform. We tested the prototype circuit on electric signals up to 1 kHz. Thus, we provide an end-to-end neuromorphic application that generates the frequency spectrum of an electric signal without the need for an ADC or a digital signal processing algorithm.

1. Introduction

The human brain processes a vast amount of sensory information quickly and efficiently. This efficient and fast processing is currently unmatched by any artificial system. Hence, ongoing research aims to understand and mimic the brain's sensory processing. An essential part of this research is encoding incoming information into spikes [1]. Coding schemes are commonly classified into rate coding and temporal coding, that assume information is stored in the rate of spike trains for a given time window or in the actual timing of the spikes, respectively. There is evidence that points toward the existence of time coding schemes in the brain [2–4], as well as for rate coding schemes [5–7]. These two paradigms can also be generalized for neuron populations, where redundant information in a group of neurons allows to reduce the inference time or the uncertainty in the signal [8]. From a conceptual point of view, rate encoding offers robustness and can be easily generalized for large populations [7]. In contrast, time coding is faster and more energy efficient, as it requires few spikes. It also explains the fast reaction times, especially for sensory pathways, that are crucial for survival [9–11]. One hypothesis of how neurons implement time coding is by carrying information in the inter-spike interval, i.e. the time between two consecutive spikes [12]. Alternatively, time coding schemes may rely on reference points in time to convey information. These references include the onset of a stimulus, spikes from other neurons, or periodic waves within the brain activity [13, 14].

One of the aims of neuromorphic research is to represent data with discrete spikes, mimicking the efficient operation of the brain. Therefore, spike encoding is a crucial step in the design of neuromorphic algorithms. When developing spiking neural networks (SNNs), the choice of the encoding scheme depends on aspects like the nature of the input data, the network architecture, or the neuron dynamics model used by the SNN. Neuromorphic computing research often focuses on high-end tasks that use processed data, e.g. classification [15], object detection [16], tracking [17], or motor control [17, 18]. These applications typically use rate encoding, as the main motivation is to validate neural models in terms of accuracy, rather than the benefits of sparsity and efficiency provided by temporal encoding schemes [19]. However, energy and time efficiency becomes more relevant for low-level neuromorphic applications that directly process sensor data. This is the case of embedded systems where the pool of energy is limited, such as automotive applications [20, 21]. There are recent examples of neuromorphic computing algorithms that deal with low-end tasks and are applied to sensor data, such as LiDAR [20], event-based cameras [22], FMCW radar [21], electrocardiogram signals [23], or microphones [24].
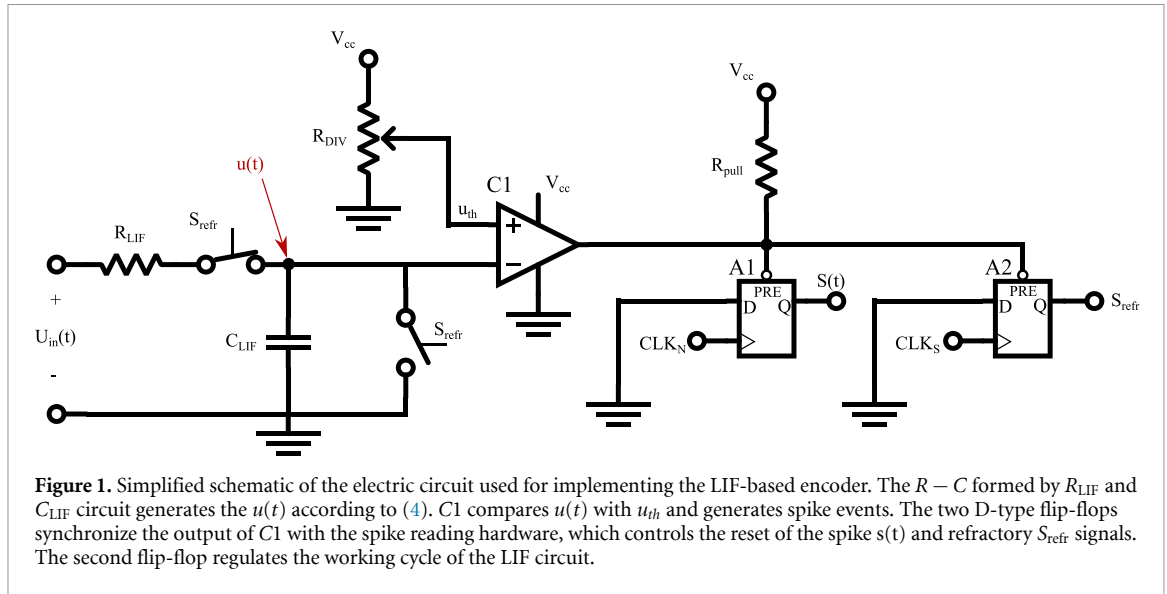
The interface between the sensor and the neuromorphic chip is crucial for obtaining a high energy efficiency and processing speed. When dealing with analog data, the most efficient approaches directly operate on the sensor signals by using ad-hoc circuits that encode them to spike trains. The signal-to-spike encoding mechanisms are typically application-specific, as they need to be compatible with the SNNs afterward. The idea of encoding analog signals to spikes for obtaining efficient implementations is not new. In [23, 25], authors use temporal contrast to identify positive and negative gradients, which discriminate high and low frequencies in the analog signal. Zhao *et al* [26] introduces a VLSI circuit for implementing the interspike-interval (ISI) mechanism for encoding analog signals. The authors in [27, 28] introduce silicon neurons for generating different spike bursting behaviours, similar to those observed in biological neurons. We refer the interested reader to [29] for getting an overview of the different analog circuits used in neuromorphic systems for real-time applications, and to [30] for a comparison of the different encoding approaches.

Some neuromorphic applications focus on generating the frequency spectrum of the incoming signal [24, 31, 32]. In these examples, an analog-to-digital converter (ADC) samples the sensor data and an SNN computes a higher-level algorithm on a digital neuromorphic chip afterward. To maximize the efficiency of these applications, the ADC should be removed by directly encoding the input signal into a spike train compatible with the digital neuromorphic chip. For the specific case of an SNN that implements the Fourier transform (FT) [32], the network needs spikes referenced to a periodic signal, as the FT algorithm works with data sampled uniformly over time. Thus, the best-suited temporal approach for these applications is phase encoding, as ISI and temporal contrast lack a signal that serves as reference over the time dimension. Additionally, the results in [30] show the benefits of phase encoding in terms of accuracy and efficiency. The benefits are more clear when comparing with rate-coded approaches, as phase coding needs down to 6.5 times less spike operations for processing data [33], resulting in a lower energy footprint [19].

The work in [32] proposed an SNN that computes a mathematically equivalent version of the FT by using time encoding. The authors implemented the spiking FT (S-FT) on a digital neuromorphic chip and validated it on automotive radar data. Here, we propose adapting the leaky integrate-and-fire (LIF) neuron model for directly encoding analog signals to phase-encoded spikes, which the S-FT can use on a digital neuromorphic chip. The proposed approach leads to the implementation of an end-to-end pipeline that does not require an ADC nor a digital processing algorithm that encodes digitized signals into spikes. The proposed analog-to-spike encoder (ASE) employs phase encoding, i.e. each sample is represented by the time difference between the spike and a reference pulse. We implemented the ASE on a physical circuit and validated the system with real data by comparing the reconstructed data from the spikes with the original data. We modeled the nature of the error of the ASE for different scenarios. Moreover, we implemented the S-FT on the neuromorphic chip SpiNNaker 2 [34] and ran it with the ASE output spikes. The results show that the chip can use the spike train that results from encoding a periodic wave. To our knowledge, we provide the first end-to-end neuromorphic pipeline for generating the frequency spectrum of analog signals without employing an ADC.

2. Analog-to-spike encoder

The LIF is a century-old single-compartment neuron model that mimics the behavior of biological neurons [35]. Its simplicity makes it a widely used artificial neuron in neuromorphic engineering. The model is based on an $R - C$ circuit that charges with the incoming current and discharges in the absence of input, resembling the flux of ions in a biological neuron through its dendrites and the constant leak of them over time through its membrane. We can express the dynamics of the LIF as

Figure 1. Simplified schematic of the electric circuit used for implementing the LIF-based encoder. The $R - C$ formed by R_{LIF} and C_{LIF} circuit generates the $u(t)$ according to (4). C1 compares $u(t)$ with u_{th} and generates spike events. The two D-type flip-flops synchronize the output of C1 with the spike reading hardware, which controls the reset of the spike s(t) and refractory S_{refr} signals. The second flip-flop regulates the working cycle of the LIF circuit.

$$C\frac{\mathrm{d}u}{\mathrm{d}t} + \frac{u(t) - u_{\text{rest}}}{R} = i_{\text{in}}(t), \tag{1}$$

where $u(t)$ is the neuron's membrane potential, u_{rest} is the resting potential, and $i_{\text{in}}(t)$ is the input current to the neuron. The LIF model does not include the depolarization and hyperpolarization mechanisms on a biological neuron when it generates a spike. Instead, it is necessary to manually add the behavior of the spike generation and refractory period. A spike is triggered at time t_s when $u(t)$ crosses the threshold voltage u_{th}. After the spike generation, $u(t)$ is reset and fixed to u_{rest} for the neuron's refractory period t_{refr},

$$t_s : u(t_s) \geqslant u_{\text{th}} \quad \text{and} \tag{2}$$

$$u(t) = u_{\text{rest}} \quad \text{for} \quad t_s < t < t_s + t_{\text{refr}}. \tag{3}$$

As artificial sensors often provide information in the form of a voltage, we modify (1) so the evolution of the membrane voltage depends on an input voltage u_{in}. We also set $u_{\text{rest}} = 0$ for simplicity,

$$RC\frac{\mathrm{d}u}{\mathrm{d}t} + u(t) = u_{\text{in}}(t). \tag{4}$$

In the remainder of the paper, we use the time constant $\tau = RC$. By integrating (4) and setting $u(t_s) = u_{\text{th}}$, we obtain an encoding function $f(u_{\text{in}})$ that maps the input voltage u_{in} to a spike time t_s after onset,

$$t_s = f(u_{\text{in}}) = -\tau log\left(1 - \frac{u_{\text{th}}}{u_{\text{in}}}\right). \tag{5}$$

Accordingly, the ideal decoding function is given by the inverse of $f(u_{\text{in}})$,

$$u_{\text{in}} = f^{-1}(t_s) = \frac{u_{\text{th}}}{1 - e^{-t_s/\tau}}. \tag{6}$$

We implemented the LIF neuron model (4) for generating a spike train from input analog signals. We encoded the information into precise spike times using (5) and controlling the refractory period according to a reference signal.

Figure 1 shows a simplified schematic of the electric circuit we used to implement the encoder. This circuit served as a proof-of-concept for implementing an end-to-end pipeline. For implementing a high-performance circuit, we refer the interested reader to literature focused on the VLSI design of silicon neurons [29].

2.1. Encoding analog voltages with temporal spikes
By default, the LIF dynamics described in (4) generates a spike rate proportional to the input voltage $u_{\text{in}}(t)$. In our approach, we sample the continuous signal $u_{\text{in}}(t)$ with a constant sampling time T_S by generating a
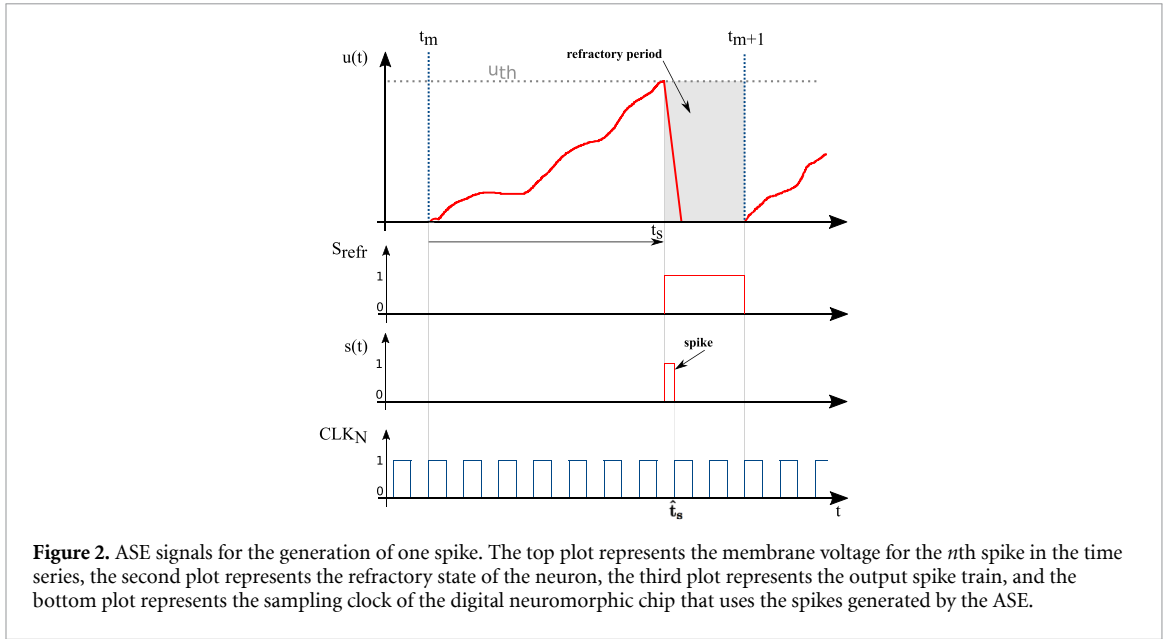
Figure 2. ASE signals for the generation of one spike. The top plot represents the membrane voltage for the *n*th spike in the time series, the second plot represents the refractory state of the neuron, the third plot represents the output spike train, and the bottom plot represents the sampling clock of the digital neuromorphic chip that uses the spikes generated by the ASE.

single spike per sample. We repeat the spike generation process for successive time windows. Thus, the mth spike represents the signal u_{in} during the time range $[t_m, t_m + T_S]$. This is inspired by the phase encoding technique hypothesized for the encoding of information in certain areas of the brain [14]. A binary variable S_{refr} forces $u(t)$ to stay in a refractory state,

$$u(t) = u_{\text{rest}}, \quad \text{iff } S_{\text{refr}} = 1. \tag{7}$$

We achieve an adaptive refractory period, $t_{\text{refr}} = T_S - t_s$, by controlling the set and reset time of S_{refr} for the *m*th sample,

$$S_{\text{refr}} = \begin{cases} 1 & \text{if } t_m + t_s < t < t_m + T_S \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

This encoding mechanism requires a periodic reference signal CLK_S (see figure 1) that defines the sampling time T_S and the end of the refractory period. We assume the input voltage to be constant during the sampling time of the ASE, $u_{in}(t) = U_{in}$, as the neuron dynamics is faster than the rate of change of the input signal. Figure 2 depicts the increase of the membrane voltage during one sample, the generation of a spike pulse, and the set and reset of the refractory signal.

The sampling time T_N of the hardware component collecting the spikes determines the spike time resolution N,
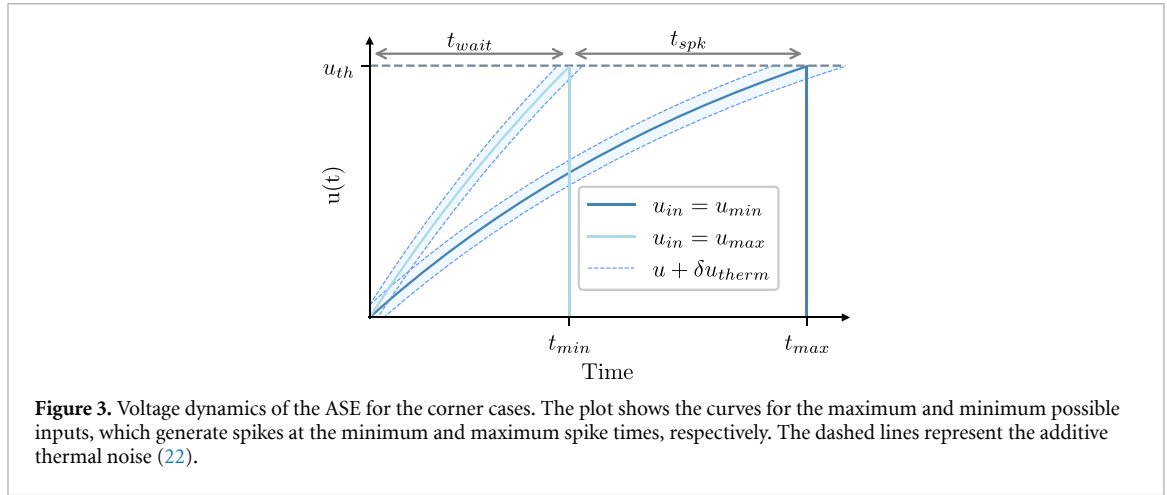
$$N = \frac{T_S}{T_N}. \tag{9}$$

2.2. Parameter tuning

The encoder (5) has two parameters that we can tune in order to map a voltage range $[u_{\min}, u_{\max}]$ onto a time range $[t_{\min}, t_{\max}]$, namely the time constant τ, and the threshold voltage u_{th}. Due to the dynamics of the neuron model, we need to choose $u_{\min} > u_{\text{th}} > 0$ to ensure a finite time range. This leads to $t_{\min} > 0$ (see (5)) and hence introduces a waiting time before the first informative spike can arrive. We can use τ to reduce $t_{\min}$, but this would also lead to a smaller time range, as $(t_{\max} - t_{\min}) \propto \tau$. To optimize the resolution in the encoding time window Δt, we aim for a small waiting time

$$t_{\text{wait}} = t_{\min} = -\tau \ln\left(1 - \frac{u_{\text{th}}}{u_{\max}}\right) \tag{10}$$

and a large spike time range

$$t_{\text{spk}} = t_{\max} - t_{\min} = -\tau \ln\left(\frac{u_{\max}(u_{\min} - u_{\text{th}})}{u_{\min}(u_{\max} - u_{\text{th}})}\right). \tag{11}$$

Figure 3. Voltage dynamics of the ASE for the corner cases. The plot shows the curves for the maximum and minimum possible inputs, which generate spikes at the minimum and maximum spike times, respectively. The dashed lines represent the additive thermal noise (22).

Hence, by maximizing the ratio

$$\mu = \frac{t_{\max} - t_{\min}}{t_{\min}} = \frac{t_{\text{spk}}}{t_{\text{wait}}}, \tag{12}$$

we can ensure to find an optimal value for u_{th} (see figure 3). The ratio (12) is independent of τ. Therefore, we can only use τ as a scaling factor to fit the time range into the encoding window Δt. A voltage threshold $u_{\text{th}} \sim u_{\min}$ maximizes the ratio.

The overall encoding performance also depends on the decoding and the subsequent algorithms. By tuning the encoding/decoding parameters, we optimize the process. We establish three pipelines to evaluate the accuracy of the coding process. First, a coding scheme using the proposed encoding $f(y)$ (5) and its inverse as decoding $f^{-1}(t)$,

$$y \xrightarrow{f} t \xrightarrow{f^{-1}} \hat{y} = y. \tag{13}$$

Second, a coding scheme using the proposed encoding $f(y)$ (5) and a linear decoding $g^{-1}(t)$,

$$y \xrightarrow{f} t \xrightarrow{g^{-1}} \hat{y}. \tag{14}$$

Finally, we assess the encoder performance using a spiking algorithm A_s that takes linearly encoded spike times as input and produces linearly encoded spike times as output. We apply the algorithm on spike times encoded as $t = f(x)$ (5). To evaluate the pipeline, we compare the decoded spiking output $g^{-1}(A_s(t)) = \hat{y}$ with the output of a non-spiking version of the algorithm $A(x) = y$,

$$x \xrightarrow{f} t \mapsto A_s(t) \xrightarrow{g^{-1}} \hat{y}. \tag{15}$$

For (13), the decoding does not need tuning because it is the inverse of the encoding (5). For (14) and (15), we tune the parameters of the encoding and decoding to minimize the error

$$\varepsilon = \int_{y_{\min}}^{y_{\max}} |\varepsilon(y)| \, \mathrm{d}y = \int_{y_{\min}}^{y_{\max}} |y - \hat{y}| \, \mathrm{d}y. \tag{16}$$

We evaluated the encoding on the S-FT algorithm [32]. The S-FT network relies on linear coding and employs current-based LIF neurons without leak. The model assumes that the input consists in phase-encoded spikes that follow the encoding

$$t_s = g(y) = t_{\text{lin, min}} + \frac{t_{\text{lin, max}} - t_{\text{lin, min}}}{y_{\max} - y_{\min}} (y_{\max} - y) \tag{17}$$

and decoding

$$y = g^{-1}(t_s) = y_{\max} - \frac{y_{\max} - y_{\min}}{t_{\text{lin, max}} - t_{\text{lin, min}}} (t_s - t_{\text{lin, min}}). \tag{18}$$

By calculating the error

$$\varepsilon_{\text{lin}} = \int_{y_{\text{min}}}^{y_{\text{max}}} |y - \hat{y}| \, \mathrm{d}y = \int_{y_{\text{min}}}^{y_{\text{max}}} |y - g^{-1}(f(y))| \, \mathrm{d}y \tag{19}$$

with the encoding $f(y)$ and decoding $g^{-1}(t)$, we see that the multiplicative time constant τ cancels out. The error depends on the decoder's time range $[t_{\text{lin, min}}, t_{\text{lin, max}}]$, the voltage threshold of the encoder u_{th} and the input voltage range $[u_{\text{min}}, u_{\text{max}}]$, where the latter is fixed by the given setup. Due to the complexity of the relationship between the tunable parameters u_{th}, $t_{\text{lin, min}}$ and $t_{\text{lin, max}}$, we propose a loss function

$$\mathcal{L} = \alpha \varepsilon_{\text{lin}} - \mu, \tag{20}$$

that we can use to determine the optimal parameters, where α defines the weight of the linear error w.r.t. the time ratio, i.e. a small α puts focus on obtaining good time ratios, whereas big α favors small linear errors. By setting a voltage threshold u_{th} that optimizes the time ratio (12), the decoding parameters $t_{\text{lin, min}}$ and $t_{\text{lin, max}}$ can be determined by directly minimizing the error ε_{lin}.

3. Results

This section covers the experiment results for the proposed ASE. We implemented a prototype on an electric board with standard components. We tested the three scenarios introduced in section 2.2 with signals obtained from a function generator. We collected the spikes and generated the reset signals for the encoder with a microcontroller from the dsPIC33CK family. The minimum sampling time of the microcontroller is 1.48 μs, which determines the resolution T_N of the spike times. We finally implemented the S-FT algorithm on the SpiNNaker 2 chip [34] and ran it with the spikes collected by the microcontroller.

We powered the circuit with 5 V, and the input voltage signal was in the range of $[1, 5]$ V. Figure 4 shows a measurement of the ASE on an oscilloscope for two consecutive spikes.

3.1. Decoding of flat voltages with an ideal decoder

We tested the ASE functionality using constant voltage signals as input. We collected the generated spikes and applied the ideal decoding function (6) to the recorded spike times. In this case, the absolute error in the measurement is mainly due to the quantization error and the circuit's thermal noise.

The spike quantization introduces a time shift δt_{q} to the actual spike time t_s due to the finite resolution of the digital hardware reading the spike events. Assuming that the spike time bins are uniformly distributed with a period of T_N, we can theoretically model this error with an average sampling error

$$\delta t_{\text{q}} = \frac{T_N}{2}. \tag{21}$$

On the other hand, voltage dynamics are noisy and voltage fluctuations can alter the threshold crossing time. The thermal error $\delta t_{\text{thermal}}$ on the spike time highly depends on the voltage dynamics. For simplification, we model the fluctuations as a constant added to the actual membrane voltage, $u(t) + \delta u_{\text{thermal}}$, and we estimate its impact on the spike time as

$$t_s - \delta t_{\text{thermal}} = -\tau \ln\left(1 - \frac{u_{\text{th}} - \delta u_{\text{thermal}}}{u_{\text{in}}}\right). \tag{22}$$

From (22), we can conclude that the thermal error on the spike times increases exponentially for small input voltages u_{in}. Figure 3 shows that thermal noise typically anticipates the spike time.

The overall measured spike time is thus obtained by combining the the quantization and the thermal noise time shifts

$$\hat{t}_s = t_s + \delta t_{\text{q}} - \delta t_{\text{thermal}}, \tag{23}$$

that leads to the decoding error

$$\varepsilon(u) = u - f^{-1}(\hat{t}_s) \tag{24}$$

$$= u - \frac{u_{\text{th}}}{1 - \left(1 - \frac{u_{\text{th}} \mp \delta u_{\text{thermal}}}{u_{\text{in}}}\right) e^{-\delta t_{\text{q}}/\tau}}. \tag{25}$$

Figure 4. Measurement on a oscilloscope of the ASE membrane voltage $u(t)$, the threshold voltage u_{th}, and the spike event signal $s(t)$ in red, yellow, and blue, respectively. The measurements show the signals for two consecutive samples.

Figure 5. Measurement errors of the ASE for encoding constant voltages $u_{in} \in [1,5]$ V with varying $u_{th} \in [0.1, 0.5, 0.75, 0.9]$ V. Top: In dark blue, decoding error $\varepsilon_u = |u_{in} - f^{-1}(\hat{t}_s)|$ for measured spike times $\hat{t}_s$. In light blue, modelled quantization error $\varepsilon_u = |u_{in} - f^{-1}(t_s + \delta t_q)|$ with ideal spike time $t_s = f(u_{in})$. Bottom: normalized error $\varepsilon_{t_s} = |t_s - \hat{t}_s|/T_N$ between the measured spike times $\hat{t}_s$.

This experiment converted constant voltages in the working range $[1,5]$ V to spikes. We ran the experiment for four different setups with different threshold voltage: $u_{th} = [0.1, 0.5, 0.75, 0.9]$ V (see figure 5).

We measured the absolute decoding error as the difference between the input voltage u_{in} and the voltage $\hat{u}_{in}$ that results from decoding the spike times,

$$\varepsilon_u = |u_{in} - \hat{u}_{in}|. \tag{26}$$

Moreover, we also measured the error in the spike times by comparing the obtained spike times $\hat{t}_s$ and the ideal spike times $t_s = f(u_{in})$ calculated with (5),

$$\varepsilon_{t_s} = |t_s - \hat{t}_s|/T_N, \tag{27}$$

where we use T_N as a normalization coefficient for making the error comparable across experiments. Figure 5 depicts ε_u and ε_{t_s} for the ASE when using the values of u_{th} from the previous experiment. We can see that the quantization error is high for high input voltages, whereas the thermal noise is low. On the other hand, low input voltages increase the probability of earlier spikes due to thermal noise, but low quantization errors in this area make spike times more precise. These errors are inherent to the used hardware and limit the accuracy of the encoding regardless of the approach followed for decoding the spikes.

We offer in figure 6(a) visual comparison between the decoded voltages $\hat{u}_{in}$ and the input voltages u_{in} for the described experiment.

3.2. Decoding of flat voltages with a linear decoder

We tested the ASE setup from the previous section with a linear decoder (18). We used the optimization strategy introduced in section 2.2 to find the parameters from (18) that best fit the curve from the ASE. We
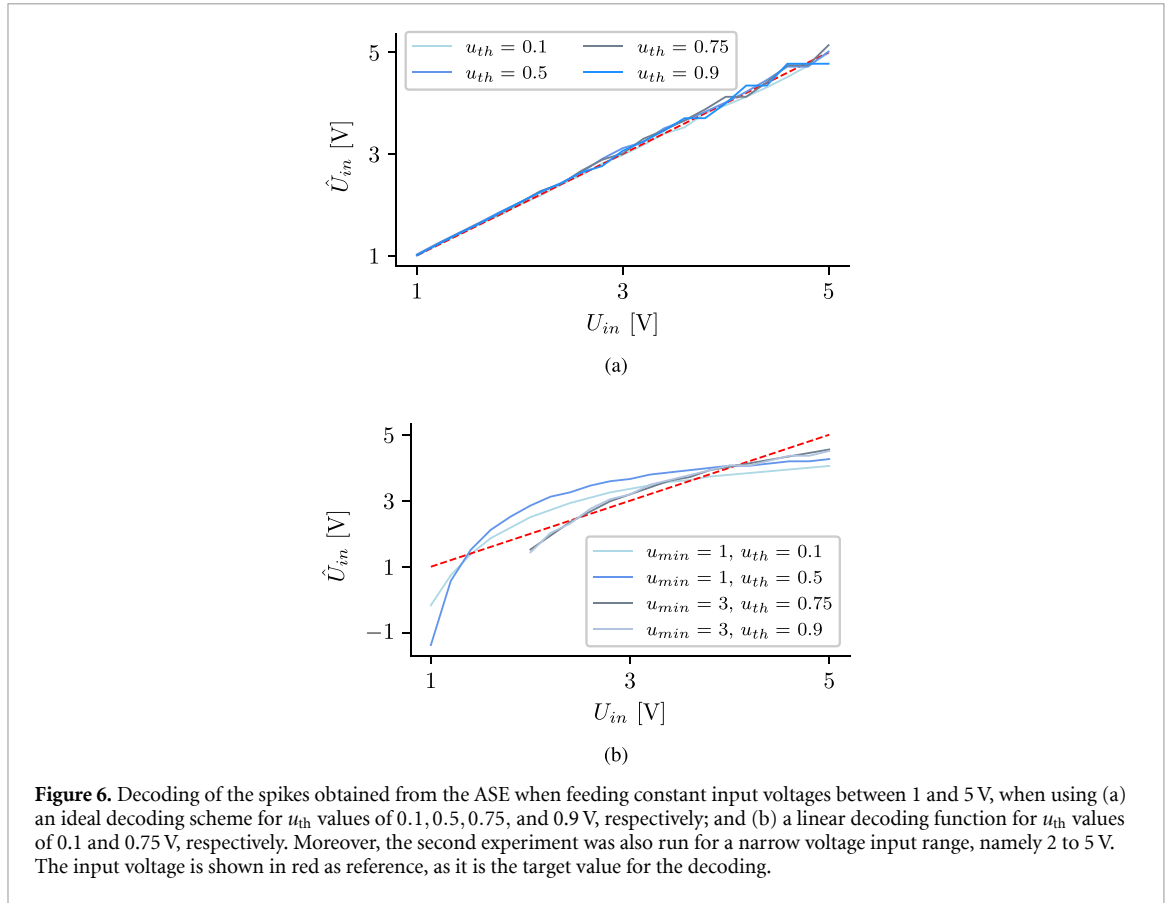
Figure 6. Decoding of the spikes obtained from the ASE when feeding constant input voltages between 1 and 5 V, when using (a) an ideal decoding scheme for u_{th} values of $0.1, 0.5, 0.75,$ and 0.9 V, respectively; and (b) a linear decoding function for u_{th} values of 0.1 and 0.75 V, respectively. Moreover, the second experiment was also run for a narrow voltage input range, namely 2 to 5 V. The input voltage is shown in red as reference, as it is the target value for the decoding.

used a differential evolution method[1] that minimizes $\mathcal{L}$ (20) by adjusting k_1 and k_2 parameters that fix the decoding time limits

$$t_{\text{lin, min}} = t_{\min} \left(1 + k_1 \right),\tag{28}$$

and

$$t_{\text{lin, max}} = t_{\max} \left(1 + k_2 \right).\tag{29}$$

As the spike times can only be real-valued, the hard boundaries for k_1 and k_2 are $[-1, \infty]$. In our implementation, we chose the boundaries $k_1, k_2 \in [-1, 2]$.

For testing the decoding, we reconstructed the original signal $\hat{u}_{\text{in}} = g^{-1}(\hat{t}_s)$ from the obtained spike times $\hat{t}_s$ with the inverse of the linear encoding function g^{-1} (18). We evaluated the reconstruction error in terms of the root mean squared error between the original signal $u_{\text{in}}(t)$ and the reconstructed signal $\hat{u}_{\text{in}}(t)$
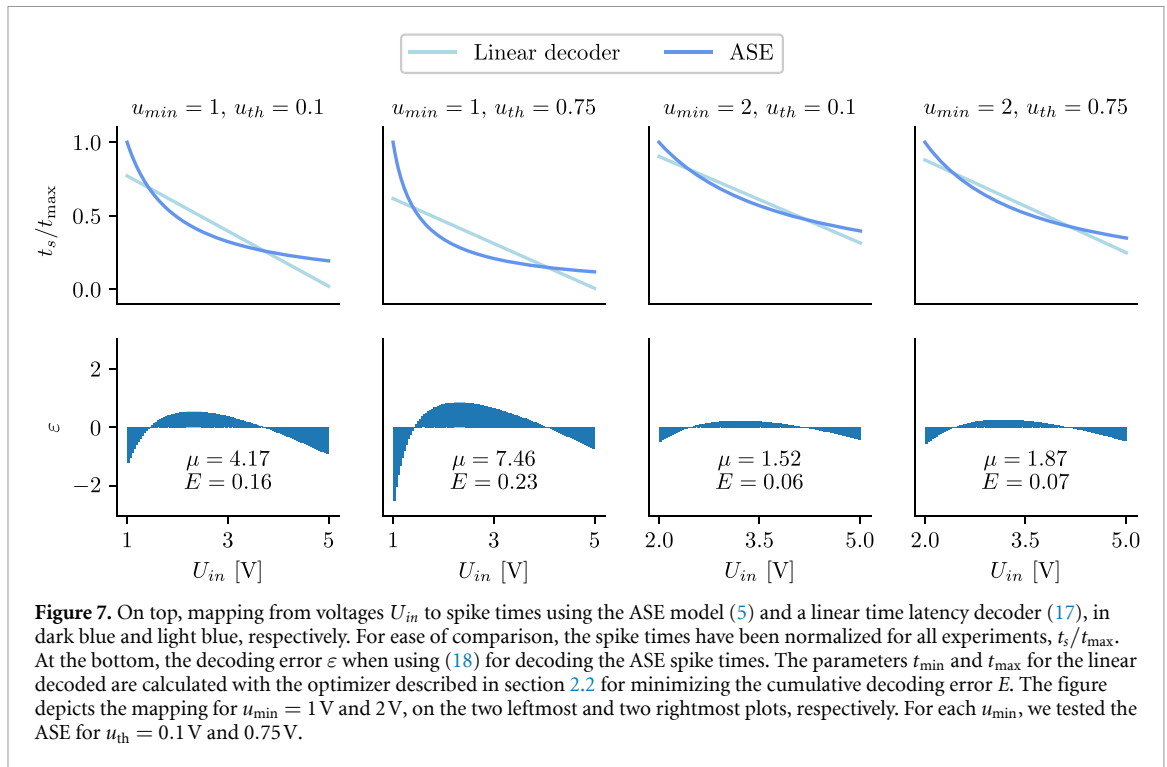
$$E = \sqrt{\frac{\sum_{m}^{M} \left(u_{\text{in},m} - \hat{u}_{\text{in},m} \right)^2}{M}},\tag{30}$$

where M is the number of samples collected during the experiment.

Figure 7 depicts the decoding results for four scenarios with different $u_{\min}$ and u_{th}. A higher $u_{\min}$ leads to a narrower input range, which directly affects the output dynamic range of the encoder, i.e. narrow voltage ranges increase the waiting time t_{wait} until a spike can take place. On the other hand, the mapping of the ASE has a smaller curvature, which diminishes the error when using a linear decoder (18). These two effects are counterbalanced with the threshold voltage, as a higher u_{th} leads to a higher dynamic range, and lower values reduce the decoding error.

After validating the optimizer on the theoretical scenarios, we applied the fitted curves to data from the experiments with real data in section 3.1. Figure 6(b) shows the obtained decoded voltages to four scenarios, with $u_{\text{th}} = [0.1, 0.75]$ V, and $u_{\min} = [1, 2]$ V. Results show that narrowing down the input voltage range is the

---

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html.

Figure 7. On top, mapping from voltages U_{in} to spike times using the ASE model (5) and a linear time latency decoder (17), in dark blue and light blue, respectively. For ease of comparison, the spike times have been normalized for all experiments, t_s/t_{max}. At the bottom, the decoding error ε when using (18) for decoding the ASE spike times. The parameters t_{min} and t_{max} for the linear decoded are calculated with the optimizer described in section 2.2 for minimizing the cumulative decoding error E. The figure depicts the mapping for $u_{min} = 1\,\text{V}$ and $2\,\text{V}$, on the two leftmost and two rightmost plots, respectively. For each u_{min}, we tested the ASE for $u_{th} = 0.1\,\text{V}$ and $0.75\,\text{V}$.

best way of improving the decoding error at the expense of reducing the output dynamic range. The results for $u_{min} = 1\,\text{V}$ also show that a higher u_{th} leads to a larger curvature and thus increases the output dynamic range. This effect is reduced for the results for $u_{min} = 2\,\text{V}$ because the change in the ratio u_{th}/u_{min} is smaller for the shown experiments. Increasing u_{th} to values closer to u_{min} would bring more significant improvements to the output dynamic range.

3.3. Computing the S-FT with the ASE output

We tested the performance of the ASE for an end-to-end pipeline that computes the frequency spectrum of the input signal. For the input, we used a function generator that creates signals consisting of one sinusoidal component. These signals can be defined as

$$h(t) = A \sin 2\pi \nu t + B, \tag{31}$$

where A, ν, and B are the amplitude, frequency, and offset of the sinusoidal wave, respectively. We connected the function generator to the ASE, generating one spike per sampling period T_S. Finally, we ran the S-FT algorithm on a SpiNNaker 2 board with the spikes generated from the ASE.

We tested the implementation for frequencies in the range $\nu \in [25\,\text{Hz}, 1\,\text{kHz}]$. Same as for the experiment in section 3.2, we created experiments with two different setups for the input voltage range. Namely, a working range of $[1, 5]$ V, and a working range of $[2, 5]$ V, i.e. we used the parameters $A = 2$ and $B = 3$, and $A = 1.5$ and $B = 3.5$, respectively. We fixed $u_{th} = 0.1\,\text{V}$ and $\tau = 3\,\text{ms}$, and the resolution per spike to $N = 100$ time steps. These parameters lead to maximum spiking times of 315 and 155 μs for the wide and narrow input voltage ranges, respectively.

In figure 8, we compare the result of decoding the spikes of the ASE using an ideal decoder and a linear decoder for a wide and narrow input voltage range, respectively. We also show on the right the result of the S-FT for both input ranges. The results correspond to a setup with a frequency of $\nu = 500\,\text{Hz}$. We repeated this experiment for frequencies $\nu \in [25, 50, 75, 100, 250, 500, 750, 1000]$ Hz and calculated the root mean squared error (30) between the obtained frequency spectrum and an FFT computed with a synthetic sine wave sampled with an ideal ADC. We show the error for the different frequencies in figure 9.

4. Discussion

We modelled the expected output of the proposed ASE from a theoretical perspective and validated its behaviour with experiments on analog signals. In section 3.1 we modelled the error in the spike encoding as a combination of a quantization error and the thermal error present in analog circuits. We proved that the quantization error grows exponentially with the value of the input voltage, and the experiments show
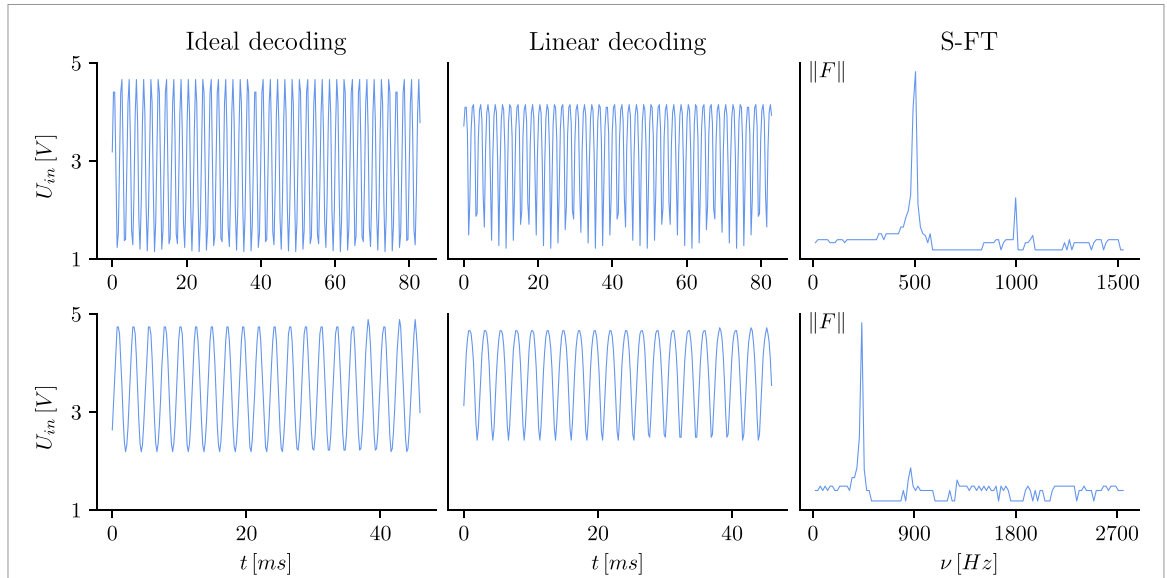
Figure 8. Output of the ASE for a sinusoidal wave of 500 Hz. The top plots correspond to a wave with amplitudes ranging between 1 and 5 V, and the bottom plots correspond to a wave with amplitudes ranging between 2 and 4 V. From left to right, the plots correspond to the ideal decoding of the spikes, the decoding using a linear function, and the result of applying the S-FT on the SpiNNaker 2 board to the output spikes, respectively. The ASE was tuned with $\tau = 3\,\text{ms}$ and $u_{\text{th}} = 0.1\,\text{V}$. The sampling frequency was 3 and 5.5 kHz, respectively, and each sample had a resolution of 100 time steps.
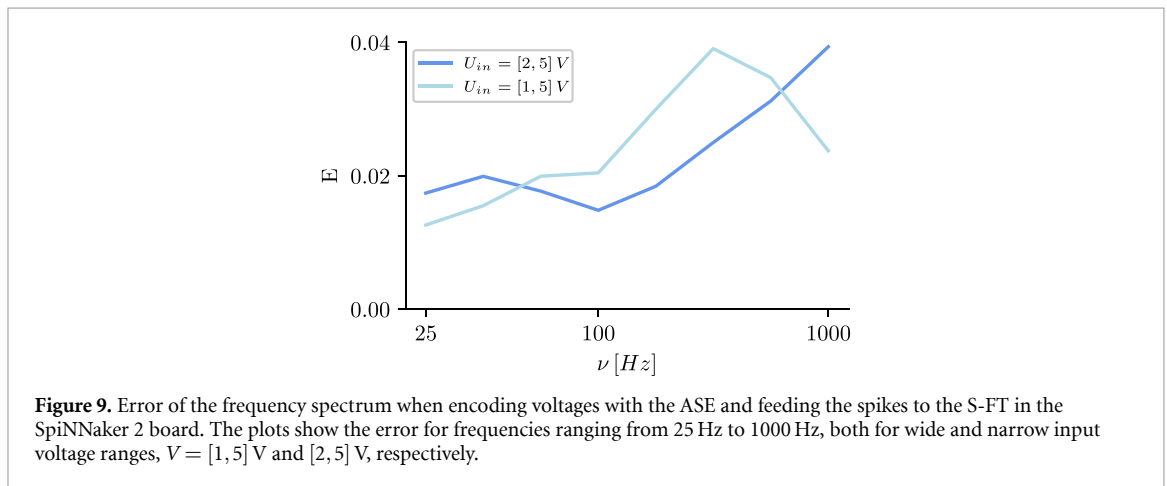

Figure 9. Error of the frequency spectrum when encoding voltages with the ASE and feeding the spikes to the S-FT in the SpiNNaker 2 board. The plots show the error for frequencies ranging from 25 Hz to 1000 Hz, both for wide and narrow input voltage ranges, $V = [1, 5]\,\text{V}$ and $[2, 5]\,\text{V}$, respectively.

evidence that this is the dominating error in the output spikes. The experiment in section 3.1 tested this hypothesis by converting input voltage signals in the range $[1, 5]\,\text{V}$ to spikes and decoding them back into voltages. We show the measured error in the decoding on the top row of figure 5 and compare it with the quantization error equivalent to half of a time step in the spike times, i.e. the difference between the input voltage and the voltage that the corresponding spike actually represents. We also show the errors in the spike times on the bottom row of figure 5. The results indicate that errors in the spike times, mostly due to thermal error, are larger for low input voltages. However, these errors do not have a big effect on the decoding error, as it is mostly dominated by the quantization error. Finally, figure 6 shows the relationship between input and decoded voltages for the considered voltage range and for four different values of u_{th}. We observe that the decoding stays close to the target value, and that errors are more significant towards high values of u_{in}.

The experiment in in section 3.2 fit a linear decoder to the mapping curve of different setups of the ASE. This step is necessary before implementing the ASE together with the S-FT, as the latter assumes phase encoded input spike times. We fit a linear decoder for four different ASE setups, where we vary u_{th} and u_{min}, and we evaluated the result in terms of the decoding error E and the dynamic output range μ. We can observe that both parameters have a big influence on both indicators. Large parameter values lead to small errors and narrow dynamic ranges. In general, the actual choice of parameters will depend on the requirements and limitations of the application. If the SNN that processes the spikes is not limited to a linear encoding function the design should aim to maximize μ, as the decoding error will stay close to zero (see top and bottom plots in figure 6).

The experiment described in section 3.3 tested the ASE together with a digital neuromorphic chip for computing the frequency spectrum of the input analog signal. As we aimed to minimize the error in the generated spectrum, we chose $u_{th} \approx 0.1\,\text{V}$. We run experiments for input signal frequencies up to 1 kHz and $u_{min} = 1$ and 2 V, and fit a linear decoding function for each setup. Figure 8 depicts the results for $\nu = 500\,\text{Hz}$. The non-linear behaviour of the ASE (5) can be approximated by a hyperbolic function assuming small ratios $u_{th}/u_{in} \ll 1$, which leads to a compression of the data for low values of u_{in}, and an expansion of the encoding range for high values of u_{in}. This translates into the appearance of harmonics in the resulting S-FT, which is specially noticeable for larger input dynamic ranges (see top right plot of figure 8). We also plot the total error of the generated frequency spectrum with that of an FFT applied on a synthetic signal sampled with an ideal ADC, i.e. simulated on a PC without any added noise. The error stays relatively low, and its mostly due to the usage of a decoding function different to the encoding one. We observe bigger errors for higher frequencies and higher input voltage ranges. For $u_{in} = [2, 5]$, this trend stops at $\nu = 500\,\text{Hz}$, as after this frequency the second harmonic of the main signal is out of the range of the generated FT. This source of noise is easy to model, and this error could be mitigated with a post-processing algorithm.

5. Conclusion

We proposed adapting the widely used LIF neuron model for mapping analog signals into spike trains using phase encoding. By using an adaptive refractory period, the output spike train keeps a notion of the sampling rate, which is a fundamental requirement for applying FT-based frequency spectrum analysis techniques.

We constructed a circuit prototype and ran it on simple periodic waves to validate the model. Our experiments show that the proposed approach can encode voltages with high accuracy. Moreover, we used the output of the ASE for running a spike-based FT on the digital neuromorphic chip SpiNNaker 2. This implementation is more efficient and simpler than previous approaches, as it does not require an ADC between the sensor and the digital chip.

One necessary step to develop real-time neuromorphic applications for signal processing is to enable digital neuromorphic chips with low-level interfaces for reading single-source spikes in real-time. Such improvement would lead to neuromorphic sensor data processing, which is crucial for efficient real-world applications, like radar-based collision avoidance.

Future work shall focus on generating a comprehensive benchmark of the ASE by creating a VLSI version of the circuit and assessing its performance with complex data obtained from actual sensors. Those benchmarks shall include critical parameters like energy efficiency and the robustness to the noise of the ASE, and compare them with the equivalent parameters of ADCs.

Data availability statement

The data that support the findings of this study will be openly available following an embargo at the following URL/DOI: https://github.com/KI-ASIC-TUM/time-coded-SFT. Data will be available from 1 November 2023.

ORCID iDs

Javier Lopez-Randulfe ● https://orcid.org/0000-0002-6922-6951
Nico Reeb ● https://orcid.org/0000-0003-4617-6215
Alois Knoll ● https://orcid.org/0000-0003-4840-076X

References

[1] Brette R 2015 Philosophy of the spike: rate-based vs. spike-based theories of the brain *Front. Syst. Neurosci.* **9** 151
[2] Bair W and Koch C 1996 Temporal precision of spike trains in extrastriate cortex of the behaving macaque monkey *Neural Comput.* **8** 1185–202
[3] Reich D S, Mechler F, Purpura K P and Victor J D 2000 Interspike intervals, receptive fields and information encoding in primary visual cortex *J. Neurosci.* **20** 1964–74
[4] Brasselet R, Panzeri S, Logothetis N K and Kayser C 2012 Neurons with stereotyped and rapid responses provide a reference frame for relative temporal coding in primate auditory cortex *J. Neurosci.* **32** 2998–3008
[5] Hubel D H and Wiesel T N 1968 Receptive fields and functional architecture of monkey striate cortex *J. Physiol.* **195** 215–43
[6] Heeger D J 1992 Normalization of cell responses in cat striate cortex *Visual Neurosci.* **9** 181–97
[7] London M, Roth A, Beeren L, Häusser M and Latham P E 2010 Sensitivity to perturbations *in vivo* implies high noise and suggests rate coding in cortex *Nature* **466** 123–7
[8] Pouget A, Dayan P and Zemel R S 2003 Inference and computation with population codes *Annu. Rev. Neurosci.* **26** 381–410
[9] Heil P 2004 First-spike latency of auditory neurons revisited *Curr. Opin. Neurobiol.* **14** 461–7
[10] Gollisch T 2009 Throwing a glance at the neural code: rapid information transmission in the visual system *HFSP J.* **3** 36–46

[11] Martin J G, Davis C E, Riesenhuber M and Thorpe S J 2018 Zapping 500 faces in less than 100 seconds: evidence for extremely fast and sustained continuous visual search *Sci. Rep.* **8** 12482

[12] Oswald A-M M, Doiron B and Maler L 2007 Interval coding. I. Burst interspike intervals as indicators of stimulus intensity *J. Neurophysiol.* **97** 2731–43

[13] Buzsáki G and Moser E I 2013 Memory, navigation and theta rhythm in the hippocampal-entorhinal system *Nat. Neurosci.* **16** 130–8

[14] Hakim N and Vogel E K 2018 Phase-coding memories in mind *PLoS Biol.* **16** e3000012

[15] Rueckauer B, Lungu I-A, Hu Y, Pfeiffer M and Liu S-C 2017 Conversion of continuous-valued deep networks to efficient event-driven networks for image classification *Front. Neurosci.* **11** 682

[16] López-Randulfe J, Duswald T, Bing Z and Knoll A 2021 Spiking neural network for Fourier transform and object detection for automotive radar *Front. Neurorobot.* **15** 688344

[17] Cao Z, Cheng L, Zhou C, Gu N, Wang X and Tan M 2015 Spiking neural network-based target tracking control for autonomous mobile robots *Neural Comput. Appl.* **26** 1839–47

[18] Stagsted R *et al* 2020 Towards neuromorphic control: a spiking neural network based PID controller for UAV *Robotics: Science and Systems 2020 (RSS)*

[19] Davidson S and Furber S B 2021 Comparison of artificial and spiking neural networks on digital hardware *Front. Neurosci.* **15** 651141

[20] Zhou S, Chen Y, Li X and Sanyal A 2020 Deep SCNN-based real-time object detection for self-driving vehicles using lidar temporal data *IEEE Access* **8** 76903–12

[21] Vogginger B *et al* 2022 Automotive radar processing with spiking neural networks: concepts and challenges *Front. Neurosci.* **16** 414

[22] Rançon U, Cuadrado-Anibarro J, Cottereau B R and Masquelier T 2022 Stereospike: depth learning with a spiking neural network *IEEE Access* **10** 127428–39

[23] Gerber S *et al* 2022 Neuromorphic implementation of ecg anomaly detection using delay chains *2022 IEEE Biomedical Circuits and Systems Conf. (BioCAS)* (IEEE) pp 369–73

[24] Auge D, Hille J, Kreutz F, Mueller E and Knoll A 2021 End-to-end spiking neural network for speech recognition using resonating input neurons *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th Int. Conf. on Artificial Neural Networks (Proceedings, Part V) (Bratislava, Slovakia, 14–17 September 2021)* vol 30 (Springer) pp 245–56

[25] Burelo K, Ramantani G, Indiveri G and Sarnthein J 2022 A neuromorphic spiking neural network detects epileptic high frequency oscillations in the scalp EEG *Sci. Rep.* **12** 1798

[26] Zhao C, Yi Y, Li J, Fu X and Liu L 2017 Interspike-interval-based analog spike-time-dependent encoder for neuromorphic processors *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **25** 2193–205

[27] Livi P and Indiveri G 2009 A current-mode conductance-based silicon neuron for address-event neuromorphic systems *2009 IEEE Int. Symp. on Circuits and Systems* (IEEE) pp 2898–901

[28] Wijekoon J H and Dudek P 2008 Compact silicon neuron circuit with spiking and bursting behaviour *Neural Netw.* **21** 524–34

[29] Chicca E, Stefanini F, Bartolozzi C and Indiveri G 2014 Neuromorphic electronic circuits for building autonomous cognitive systems *Proc. IEEE* **102** 1367–88

[30] Forno E, Fra V, Pignari R, Macii E and Urgese G 2022 Spike encoding techniques for IoT time-varying signals benchmarked on a neuromorphic classification task *Front. Neurosci.* **16** 999029

[31] Orchard G, Frady E P, Rubin D B D, Sanborn S, Shrestha S B, Sommer F T and Davies M 2021 Efficient neuromorphic signal processing with loihi 2 *2021 IEEE Workshop on Signal Processing Systems (SiPS)* (IEEE) pp 254–9

[32] López-Randulfe J, Reeb N, Karimi N, Liu C, Gonzalez H A, Dietrich R, Vogginger B, Mayr C and Knoll A 2022 Time-coded spiking Fourier transform in neuromorphic hardware *IEEE Trans. Comput.* **71** 2792–802

[33] Guo W, Fouda M E, Eltawil A M and Salama K N 2021 Neural coding in spiking neural networks: a comparative study for robust neuromorphic systems *Front. Neurosci.* **15** 638474

[34] Höppner S *et al* 2021 The spinnaker 2 processing element architecture for hybrid digital neuromorphic computing (arXiv:2103.08392) (https://doi.org/10.48550/arXiv.2103.08392)

[35] Abbott L F 1999 Lapicque's introduction of the integrate-and-fire model neuron (1907) *Brain Res. Bull.* **50** 303–4