Real-Time Classification of Epileptiform Activity in the Intrahippocampal Kainic Acid Mouse Model

Master Thesis

Jeroen Vermeulen

Delft University of Technology

TU Delft

Real-Time Classification of Epileptiform Activity in the Intrahippocampal Kainic Acid Mouse Model

Master Thesis

by

Jeroen Vermeulen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday July 10, 2024 at 9:00 AM.

Student number:	4964864	
Thesis Committee:	Prof. dr. ir. Said Hamdioui	TU Delft
	Dr. ir. Matin Jafarian	TU Delft
	Dr. ir. Rajendra Bishnoi	TU Delft
	Dr. Else Tolner	Leiden University Medical Center
Supervisors	Georgii Krivoshein, MD	Leiden University Medical Center
	Dr. ir. Muhammad Ali Siddiqi	TU Delft & Lahore University of Management Sciences
Faculty:	EEMCS	
Degree:	MSc Embedded Systems	

June 30, 2024

TU Delft

Preface

I would like to thank Rajendra Bishnoi for his daily guidance during the project and for always being there to answer any questions. Next to that, he helped by motivating me and encouraging me to think critically about the choices I made during this project. Secondly, I want to thank Georgii Krivoshein for giving me the chance to work on this interdisciplinary project together with him. This project has led to an incredibly enjoyable and valuable experience for my master's thesis. Finally, I want to thank Muhammad Ali Siddiqi and Else Tolner for their assistance in writing a paper next to my thesis and for the outside look they have offered on various parts of my research.

Jeroen Vermeulen
Delft, June 2024

Abstract

One-third of patients suffering from chronic epilepsy, which is caused by abnormal brain activity, is drug-resistant. Animal models are widely used to study the mechanisms leading to epilepsy so better drug treatments can be developed for this disease. In such studies, epileptiform activity, assessed by LFP recordings, can be used as a marker for the development and chronification of disease. However, the analysis of LFP recordings is typically done manually, which is time-consuming, subject to observer bias, error-prone, and lacks consistency and efficiency. Therefore, we present a work which developed a new, automated detection and classification method for epileptiform activity, which was tested in the intrahippocampal kainic acid (IHKA) mouse model, a model of human temporal lobe epilepsy. Our method relies on a spike detector using an improved version of the nonlinear energy operator (NEO) in combination with automatic NEO thresholding (ANT). The detected spikes form the basis of epileptiform event detection and classification. The proposed method is implemented in Python as an automated and time-efficient algorithm that can be used in preclinical studies. Epileptiform event detection accuracy was 93.1% and classification accuracy 95.8%. Moreover, the time for analysis of LFP recordings was reduced by 98.8% compared to manual analysis. Additionally, to demonstrate the potential of the algorithm for application in Brain-Machine Interfaces (BMI), we performed a real-time implementation using both an application-specific integrated circuit (ASIC) and a field programmable gate array (FPGA). The FPGA demonstrated the feasibility of real-time implementation, and the ASIC resulted in an area and power efficient chip using the Taiwan semiconductor manufacturing company (TSMC) 45nm library that constitutes a post-layout area of 9114 μm^2 and a power usage of 6.11 μW.

Contents

1 Introduction

1.1. Motivation & Problem Statement

Epilepsy is a neurological disorder characterized by recurrent seizures caused by abnormal and excessive neuronal activity in the brain, and affects approximately 65 million people worldwide [1]. The consequences of epilepsy can be severe to patients as seizures can involve symptoms such as loss of consciousness, and prolonged failure to control attacks can lead to cognitive decline and, in rare cases, can result in the death of the patient [2]. Approximately 60% of patients suffer from partial (or focal) epilepsy, meaning that seizures originate in only part of the brain. The most common type of partial epilepsy is temporal lobe epilepsy (TLE), which typically originates from the hippocampus, entorhinal cortex, or amygdala [3]. Around one-third of the patients with TLE are resistant to medication, making it one of the most drug-resistant types of epilepsy [4].

Preclinical studies in rodents in which epilepsy is evoked are performed to investigate the underlying disease mechanisms in search of new treatments for epilepsy. A commonly used model for TLE is the intrahippocampal kainic acid (IHKA) mouse model [5]. Epileptiform activity, i.e. excessive, highly synchronized neural network activity [6], is visible in local field potential (LFP) recordings. LFP recordings measure electrical brain activity with an electrode inside the brain tissue and are comparable to electroencephalography (EEG), which is measured outside the scalp. The epileptiform activity found in LFP recordings can be categorized into two main types: isolated spikes and epileptiform events, where the epileptiform events can be classified into multiple subgroups. All forms of epileptiform activity can potentially serve as indicators of the progression of epilepsy. Experts traditionally analyse LFP recordings generated in preclinical studies manually, as shown in recent studies [7, 8]. Preclinical studies generate a lot of LFP recordings. Manual analysis is a time-consuming process and prone to observer bias and error [9].

1.2. State-of-the-Art Solutions

In the past, besides options for spike and/or burst detection available in commercial software packages [10, 11], various (open-source) efforts have been made to implement an epileptiform activity detector that can extract spikes and epileptiform events and classify them into different subgroups [12, 13, 14, 15, 16, 17]. These current state-of-the-art solutions can detect epileptiform activity in LFP recordings but are either not fully automated and thus not time-efficient or are not implemented using the IHKA mouse model. Section 3 gives a more in-depth overview of the state-of-the-art solutions.

1.3. Collaboration Statement

At the Department of Neurology and Human Genetics at the Leiden University Medical Center (LUMC), preclinical EEG studies towards TLE are being conducted. To avoid a time-consuming analysis of all the data, a collaboration has been established between the LUMC and the Quantum and Computer Engineering department of the Delft University of Technology (TU Delft). This collaboration is instan-

tiated to develop an automated and time-efficient tool that can detect and classify epileptiform activity from LFP recordings using the IHKA mouse model. All the data used for this thesis originates from the preclinical study at the LUMC.

1.4. Proposed Solutions

This thesis presents a new methodology for automated detection and classification of epileptiform activity in LFP recordings. It consists of three steps and is developed using data from the preclinical IHKA mouse model of epilepsy. In the first step, a spike detector is introduced using an improved version of the nonlinear energy operator (NEO) and automatic NEO thresholding (ANT) combination. Secondly, an epileptiform event detector is proposed that can detect epileptiform events using a generalized description. Finally, an epileptiform event classifier is proposed to classify epileptiform events into one of four subgroups. The proposed detection and classification methodology is implemented as an algorithm in Python and implemented as an ASIC and FPGA using Hardware Description Languages (HDLs). The implemented algorithm reduces time spent on the analysis of LFP recordings by removing the need for manual analysis. Next, the analysis becomes more consistent and less prone to observer bias when using the proposed methodology.

1.5. Contributions

A summation of the main contributions of this thesis are given below:

- We propose a new and automated method for detecting and classifying epileptiform activity in the Intrahippocampal Mouse Model.
- We implemented the proposed method as an algorithm in Python, decreasing the time spent on LFP recording analysis.
- We implemented the proposed method in real-time as both an ASIC and FPGA to show the working as a Brain-Machine Interface.

1.6. Outline

This thesis is organized as follows: In Chapter 2, the IHKA mouse model is explained by going over the mice and their surgery procedure in the preclinical studies, after which descriptions of the epileptiform activity are given. Chapter 3 goes over related work available on epileptiform activity detection and classification. Chapter 4 introduces the algorithmic implementation of the proposed epileptiform activity detector and classifier, and chapter 5 introduces the proposed hardware implementation. Chapter 6 gives an overview of the experimental setup and chapter 7 goes over the results obtained, the conclusions are given in chapter 8.

2

Intrahippocampal Kainic Acid Model

This chapter introduces the intrahippocampal kainic acid (IHKA) mouse model by first going over the mice used. Then, the surgery procedure followed for generating the mouse model is described. These descriptions come from the Leiden University Medical Center (LUMC, NL). Secondly, all forms of epileptiform activity are described, and examples are given, after which their distinctive features are summarized.

2.1. Animals

Electrographic LFP recordings were obtained from the IHKA mouse model. For the IHKA mouse model, C57BL/6J male mice (Janvier, France) at the age of 10 - 12 weeks were utilized. During experimental procedures, all mice were kept under standard housing conditions (temperature of 22 ± 1.5°C, 12/12 h light/dark cycle) with food and water *ad libitum*. All procedures were approved by local and national ethical committees (project license AVD11600202317073) following recommendations of the European Communities Council Directive (2010/63/EU) and performed in accordance with ARRIVE guidelines.

2.2. Surgery

Animals were anaesthetized with isoflurane (induction 4%; maintenance 1.5%) in pressurized air. Carprofen (5 mg/kg) was injected subcutaneously 15 minutes before surgery as preemptive analgesia. During surgery, mice were placed in a stereotactic device. After the skin scalp was resected, small craniotomies were drilled in the skull using a dentist's bore (Fine Science Tools, USA). Kainic Acid (KA; 200 ng in 50 nL 0.9% NaCl; Sigma-Aldrich, USA) was injected into the CA1 region of the rostral hippocampus (anteroposterior (AP): - 2,0 mm; mediolateral (ML): ± 1,5 mm; dorsoventral (DV): - 1,3 mm; Figure 2.1 A) using a glass 0.5 μL NanoVolume on-column syringe (0.23 mm OD needle; Trajan Scientific and Medical, USA) at a rate of 0.1 μL/min controlled by a UMP2 microinfusion pump (World Precision Instruments, USA). To limit backflow of fluid, the needle was maintained in situ for 2 minutes before and 5 minutes after injection. After surgery, the mice were placed in a recovery box with temperature controlled at 30°C for 30 minutes. The Status Epilepticus, which occurs in the hours after hippocampal KA injection, was monitored using video recording and combined with post-hoc behavioural analysis.

One week later, during the second surgery, custom-made microelectrodes (75-µm platinum/iridium; PT6718, Advent Research Materials, UK) were placed in the same CA1 area and coordinates of the rostral hippocampus (Figure 2.1 B). Additional microelectrodes were placed in the contralateral hippocampus and cortex (bilateral visual cortex V1 and unilateral motor cortox M1 ipsilateral to the hemisphere in which kainic acid was injected) to assess any spread of electrographic epileptiform activity. Reference and ground electrodes were placed in the cerebellum. Then, the microelectrodes were connected to a 7-channel pedestal (Plastic One, USA) and glued to the skull with dental cement (DiaDent Europe, NL). After the experiments, mice were sacrificed for histology procedures to confirm electrode locations and assess cellular damage in the KA-injected hippocampus (Figure 2.1 C).

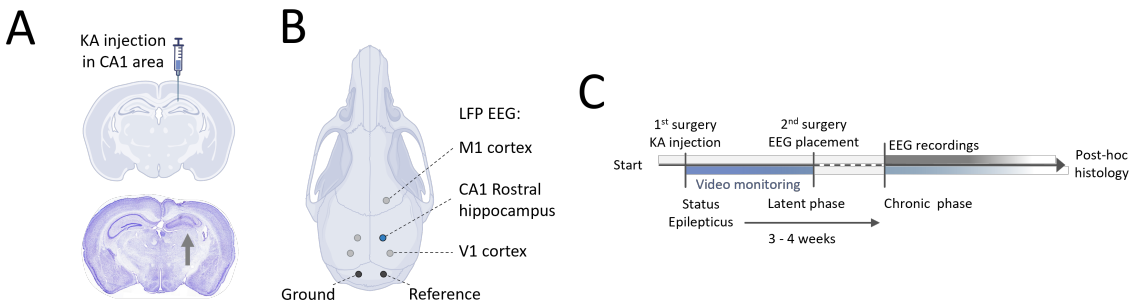

Figure 2.1: LFP recordings in IHKA mouse model. (a) Schematic of the kainic acid (KA) injection site in the CA1 area of the rostral hippocampus assessed by post hoc histology in coronal sections. The arrow indicates the malformed (including dispersion of the normally compact cell layers) and 'damaged' (i.e. gliosis, cell loss) ipsilateral hippocampus, indicating successful KA injection and development of an epileptogenic zone. (b) Schematic of the position of the recording LFP electrodes, whereby the LFP electrode in the right rostral hippocampus at the site of KA injection (i.e. 'ipsilateral') is used for detecting the epileptiform activity in this area. (c) Experimental design of surgeries and video-LFP recordings. Of note, LFP recordings started at 5 weeks post-KA injection/status epilepticus.

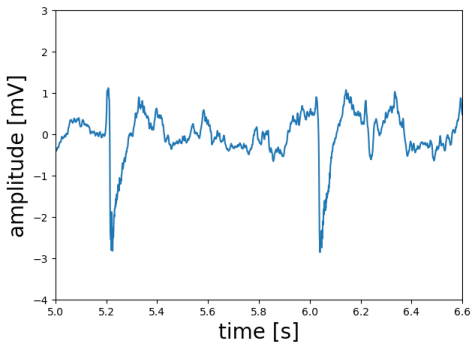

Figure 2.2: Two spikes visible in an LFP recording, both have a clear negative peak and a small positive peak.

2.3. Electrographic epileptiform activity

In this work, the description of epileptiform activity in the IHKA mouse model by Twele *et al.* [5] is used, which consists of two types of epileptiform activity: spikes and epileptiform events. Spikes are visible in LFP recordings as transients that are distinguishable from background activity and have a negative and positive pointed peak. An example of two spikes inside an LFP recording is shown in Figure 2.2. The duration of a spike was defined to range from 40 to 100 milliseconds and the amplitude of one of the pointed peaks is larger than $1.5\times$ the baseline amplitude of the signal. The baseline amplitude was considered to be the average amplitude of a signal segment with no epileptiform activity. Finally, the shape of a spike can vary depending on the neuron that fires, the type of epilepsy, and the stage of epilepsy. Further, spikes can be categorized as ictal or interictal, where ictal spikes are found inside epileptiform events, and interictal spikes are found outside events.

An epileptiform event can be described as a group of spikes that follows a set of requirements with respect to spike frequency, spike amplitude, and the duration of the event. The first type of epileptiform event that occurs in the IHKA model is a "spike train". Spike trains are events which have a duration between 2 and 5 seconds. Spikes were defined to have an amplitude of at least $3\times$ the baseline amplitude, the spike frequency should be at least 2 Hz, and the interval between subsequent events, interevent interval, is minimally 3 seconds.

Secondly, there are high voltage sharp waves (HVSW) that have the same description of spike amplitude, spike frequency, and interevent interval as spike trains. The duration of HVSW is between 5 and 10 seconds. HVSW can show evolution in frequency and pattern, but most of the time, there is no evolution, meaning they are monomorphic. An example of the HVSW is shown in Figure 2.1 D. The HVSW show no clear evolution (monomorphic) over time.

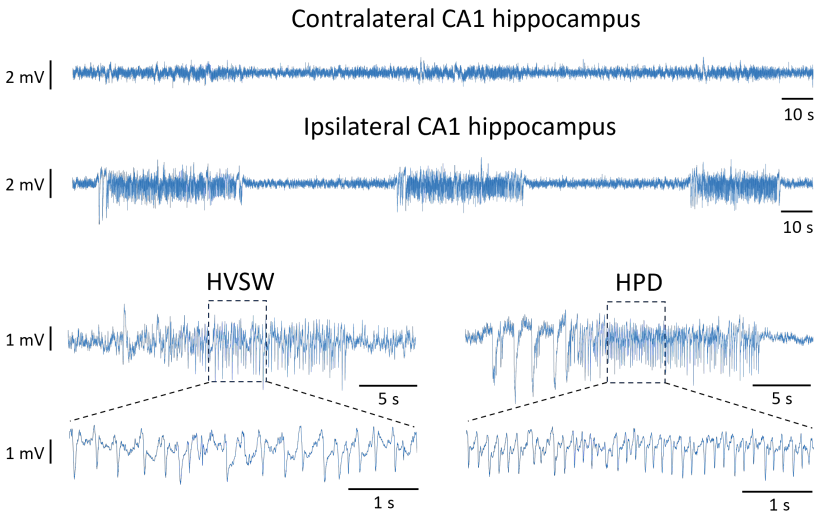

Figure 2.3: Examples of LFP recordings from contralateral and ipsilateral (KA injection) CA1 hippocampi. The ipsilateral side contains spontaneous focal epileptiform events of different electrographic characteristics: high voltage sharp waves (HVSW) and hippocampal paroxysmal discharges (HPD).

Table 2.1: Characteristics of all subgroups of epileptiform events: Spike train, HVSW, sHPD and iHPD.

	Spike train	**HVSW**	**sHPD**	**iHPD**
Spike amplitude ($\times$ Baseline amplitude)	≥ 3	≥ 3	≥ 2	≥ 2
Spike frequency	≥ 2 Hz	≥ 2 Hz	≥ 5 Hz	≥ 5 Hz
Minimum event duration	2 s	5 s	5 s	10 s
Maximum event duration	5 s	20 s	10 s	-
Interevent interval	≥ 3 s	≥ 3 s	≥ 3 s	≥ 3 s

The third type of events are hippocampal paroxysmal discharges (HPD), which typically start with HVSW-like activity and is followed by spikes with a lower amplitude and a higher frequency. The spike amplitude for HPD is at least 2$\times$ the baseline amplitude, the spike frequency is at least 5 Hz and the interevent interval is 3 seconds. Figure 2.1 D shows an example of an HPD event, where the start of the event is HVSW-like activity followed by higher frequency and lower amplitude spikes. This figure clearly shows that HPD are polymorphic and exhibit evolution in pattern and frequency. HPD events are further classified into two types by their duration. Events with a duration of 5 to 10 seconds are short HPD (sHPD) and events with a duration of more than 10 seconds are ictal HPD (iHPD). Moreover, an HVSW event with more than 20 seconds duration is also classified as iHPD. To summarize, the epileptiform activity that needs to be marked are interictal spikes, spike trains, HVSW, sHPD, and iHPD. The characteristics of all types of epileptiform events are shown in Table 2.1.

3

Related Work

As described in the introduction, the goal of this work is to extract all forms of epileptiform activity. One work exists which introduces a method that can detect and classify epileptiform activity. This method will be discussed last. To give a complete overview of related work on this topic, the following chapter is split up into three sections highlighting the individual components that are necessary for epileptiform activity detection and classification. First, spike detection methods are discussed, then epileptiform event detection methods are discussed, and last, one complete work that can also classify epileptiform events is discussed.

3.1. Spike Detection

Spike detection in an LFP recording is an inherently difficult task to automate. This has a few reasons. Firstly, one electrode will pick up data from multiple neurons in the area, which causes distortions to be added to the LFP recordings [18]. Another reason is that differentiating spikes from normal activity is difficult because candidate spikes and actual spikes can look very similar to the naked eye, next to this spike shape can change during events and between patients [19]. Next to finding a suitable algorithm to process the data and highlight spikes a correct thresholding estimation is also important as the algorithm should be unsupervised and thus not rely on a handset threshold.

The algorithms used for spike detection can range from simple to complex methods, arguably the simplest method is Amplitude Threshold (AT) [20]. AT works by manually setting a threshold and every time the signal crosses this threshold a spike is detected. AT is computationally simple, but it lacks robustness as the performance is highly sensitive to noise. This method can be improved by using a sliding window and taking the mean deviation and Root Mean Square (RMS) to determine the noise of the signal [21] and adjusting the threshold accordingly. This improves the method but is computationally heavier and will still not achieve great accuracy.

A matched filter is another method used for spike detection, where one or multiple templates are used to compare to the signal. A spike is detected when such a segment is similar to the signal. The similarity can be measured in Euclidean distance or cross-correlation [22]. The advantages of this method are that it can detect spikes of different morphology and amplitudes, but the drawback is that templates need to be created beforehand. Next to this is the computational complexity high as for every sample, a lot of computations need to be done due to the sliding template(s).

A third method to detect spikes is using the Discrete Wavelet Transform (DWT) [23], which was created to overcome the shortcomings in the fixed time-frequency resolution of the Fourier transform. DWT can better represent time-frequency because of the variable-sized window. DWT is derived from the Continuous Wavelet Transform and is shown in equation 3.1, where ψ is the chosen wavelet, and 2^j and $k2^j$ are the scaling and time localization parameters.

$$DWT(a,b) = \frac{1}{\sqrt{|2^j|}} \sum_{t=-\infty}^{\infty} x(t)\psi(\frac{t-k2^j}{2^j}) \tag{3.1}$$

The DWT is implemented as a filter bank, meaning the wavelet transform functions as a cascade of low- and high-pass filters. Starting with the smallest scale, which corresponds with the highest frequencies, in the second stage, the scale is doubled, and the frequency is thus halved, next to which the signal is down-sampled two times. This can go on until maximum decomposition when the signal has fewer samples than the wavelet. The advantage of DWT is that it has a good representation of time-frequency. The downside is that this method requires a lot of multiplications and is thus computationally intensive.

Next to computational methods, multiple methods use machine learning or deep learning to detect spikes. For example, Prasanth *et al.* [24] propose a method based on a Convolutional Neural Network (CNN). It uses raw EEG data and certain frequency sub-bands as input, which is split up into segments to detect whether it is a spike or a background signal. It was found that using multiple frequency sub-bands helped to prevent false positives and increased the precision for the same sensitivity. This method provides high performance but also has a high cost due to the CNN and the usage of multiple sub-bands.

Finally, energy operator methods are available for spike detection, two of which will be highlighted: Amplitude Slope Operator (ASO) and Nonlinear Energy Operator (NEO). Amplitude Slope Operator (ASO) [25] is described by Equation 3.2. Here $y_n - y_{n-1}$ stands for the slope at a given point, and y_n is the output of a mean subtraction, which is used to remove the temporal drift in a signal.

$$z_n = y_n(y_n - y_{n-1}) \tag{3.2}$$

The ASO threshold is calculated by calculating the average across 64 samples and multiplying that with 40. The ASO's advantage is its computational efficiency, as for every sample, only one multiplication and one division need to be done. The downside is that while the thresholding is dynamic, it is still not robust to noise in different datasets.

Nonlinear Energy Operator (NEO) [26] is another energy operator, which is defined by Equation 3.3.

$$\psi[x(n)] = x^2(n) - x(n+1)x(n-1) \tag{3.3}$$

NEO measures the instantaneous energy from the frequency and amplitude information in the signal. The algorithm is very suitable for real-time detection as it is computationally fast. Conventionally a threshold has been set using $Th = 4\sigma_d$, the advantage is that the threshold is easy to compute but it has been shown to not be accurate [27]. Another method introduces Automatic NEO Thresholding (ANT) [28]. It calculates the threshold Th by first calculating the RMS frequency Ω_{rms} and the standard deviation of the noise σ_n. This threshold calculation is computationally efficient, automatic, and robust against noise.

To summarize, Table 3.1 gives an overview of all spike detection methods discussed together with their advantages and disadvantages.

3.2. Epileptiform Event Detection

Multiple methods depend on the increase in power in certain several frequencies [12, 13]. Zeidler *et al.* introduces a method in which first bandpass filters the LFP recording, after which it is smoothed using a Butterworth filter and finally compared to a threshold to indicate possible events. This method uses the IHKA mouse model and has the advantage that it is easy to implement as only two filter operations are done, but the downside is that every mouse needs an individual threshold, and the highlighted sections still need to be reviewed manually. Secondly, Iotchev *et al.* [13] introduce an algorithm that detects events using a combination of increased power in the 5 to 50 Hz frequency range and an increase in asymmetry along the midline. The method first detects peaks by thresholding a filtered and standardized signal using 3$\times$ the standard deviation to find spikes. If two spikes were found inside a

Table 3.1: Overview of spike detection methods and their scoring on features deemed important for this work.

Spike detection method	Complexity	Speed	Robustness to noise	Automatic
AT [20]	+	+	−	−
Matched filter [22]	+	−	+	−
DWT [23]	−	−	+	−
CNN [24]	−	+	+	+
ASO [25]	+	+	−	+
NEO [26] & ANT [28]	+	+	+	+

Min. Amplitude	Max. amplitude	Min. Seizure duration	Max. inter-spike Interval
$3\times$ Baseline	$\leq 1500\mu V$	> 10 s	< 5 s

Table 3.2: Requirements for clustering spikes and determining the presence of an event by Kyle *et al.* [14].

1.5-second window, there was asymmetry in the signal. After this, peaks were found in the frequency domain inside three-time windows of 1 second with 0.5-second overlap. If 2 peaks of $>$ 0.005 or 3 peaks of $>$ 0.0025 (unit is unknown as the signal was standardized), increased power in the frequency domain was found. An event was detected if both the asymmetry and increased power were present. The advantage of this method is the automated implementation, but it is not tailored towards the IHKA model, which makes it unsuitable for this work.

A second method is introduced by Kyle *et al.* [14], where the data is processed by first detecting spikes, after which epileptiform event detection is done. The spike detection is done using amplitude thresholding. The threshold is determined by a value of $2\times$ the baseline amplitude. The baseline amplitude is calculated by taking the 97^{th} percentile of the first hour of the signal. After a spike is detected, it will be clustered as an epileptiform event with other spikes if it fulfils the requirements stated in Table 3.2. This method analyzed three months of EEG recordings and achieved a time efficiency improvement from 4 months of manual work to one hour of code running time. The approach is promising as it both detects spikes and epileptiform events, but it has downsides. Firstly, the baseline calculation and spike detection are not robust to noise or high spiking frequency, which makes the method inaccurate. Next to this, the event detection is oversimplified due to the fact that another mouse model is used. This makes it easy to implement, but for example, spike frequency is not taken into account, which might cause many false positives if used together with the description of Twele *et al.* [29].

A third method by Wei *et al.* is dedicated to designing a machine learning model that works for different mouse models [15]. A model has been based on the XGBoost algorithm and is trained using three different mouse models and a fourth is only used to test the model. The method has shown that it can work on multiple mouse models, including the IHKA mouse model, and can correctly detect epileptiform events. Using machine learning models like the XGBoost algorithm is, however, not possible as no work exists that uses the same mouse model and the same type of EEG recordings. Secondly, there exists no training data to train a model ourselves.

3.3. Epileptiform Event Classification

One work exists that differentiates between multiple types of epileptiform events, and it is introduced by Theilmann *et al.* [16]. The method is based on the spike and event detection introduced by Anjum *et al.* [17], which marks spikes based on amplitude, width, instantaneous energy, and slope. The event detection was modified by Thielmann *et al.* and is based on the description of epileptiform events by Twele *et al.* [5]. This method is thus able to detect and classify epileptiform activity but has two downsides: the spike detection makes use of manual set thresholds, and both the automatic spike and event detection have a visual analysis step, which results in a lot of manual inspections.

4

Proposed Epileptiform Activity Detector and Classifier

This chapter introduces the proposed solution, which is an epileptiform activity detection and classification method and its algorithmic implementation. Figure 4.1 gives an overview of the data flow in the proposed methodology, where the data is first gathered and preprocessed. The preprocessed LFP recordings are used as input to the spike detector, which outputs the detected spikes. The epileptiform event detector uses the detected spikes to detect all types of epileptiform events and will make a distinction between ictal (part of the epileptiform events) and interictal spikes. Finally, the event classifier classifies events and outputs the interictal spikes, detected events and classification of the detected event.

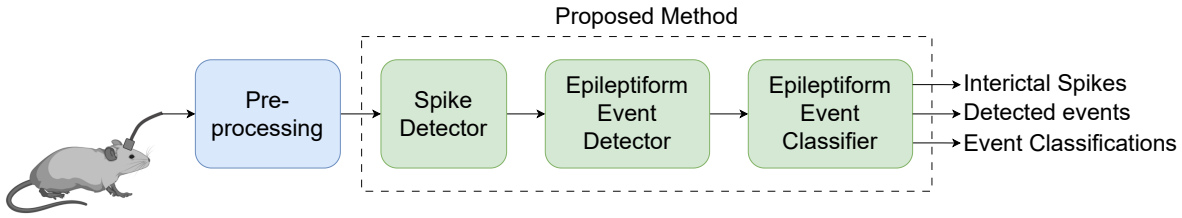

Figure 4.1: Proposed method for detection and classification of epileptiform activity in the IHKA mouse model, divided into spike detector, epileptiform event detector and epileptiform event classifier.

4.1. Pre-processing

The mice used in the preclinical study are connected to a seven-channel commutator in a Faraday shielded recording cage at 5 weeks (i.e. 5 weeks after status epilepticus) after the first surgery for continuous LFP and video recording of the chronic stage of epilepsy. Only recordings of the CA1 area of the rostral hippocampus have been used, which were pre-amplified (3×), filtered (0.05–500 Hz), amplified (200×) using custom-build hardware, and digitized at 5 kHz (Power 1401 and Spike2 software, CED) with at last down-sampling to 1 kHz using MATLAB (version 2022b, MathWorks, USA).

4.2. Spike Detector

In Section 3.1, multiple existing spike detection methods have been discussed, which all have their own advantages and disadvantages. From Section 1, it is clear the proposed solution should be automated and time-efficient. Table 3.1 gives an overview from which it is clear that NEO, in combination with ANT, have the best features, and thus, they are used as the proposed spike detector. A block diagram of the proposed spike detector is shown in Figure 4.2 and illustrates the individual components used. The spike detector consists of three Infinite Impulse Response (IIR) filters, the NEO calculator and the

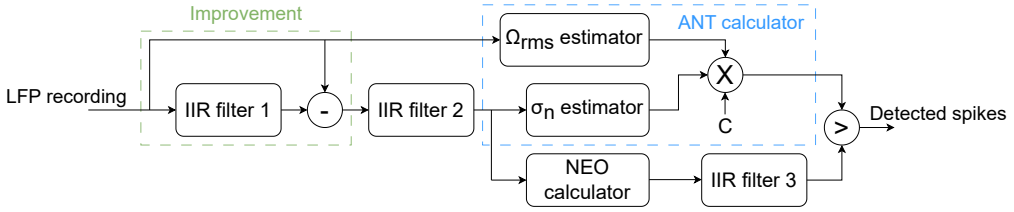

Figure 4.2: Block diagram of the proposed spike detector, consisting of three IIR filters, the NEO calculator and the ANT calculator.

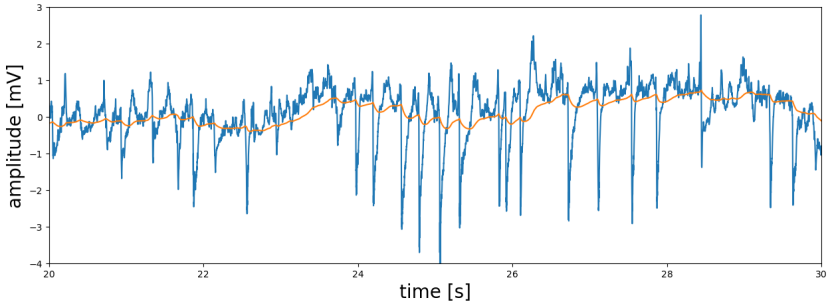

Figure 4.3: Demonstration of IIR filter 1, with the LFP recording in blue and the output result of IIR filter 1 shown in orange.

ANT calculator consisting of the root mean square (RMS) Ω_{rms} frequency estimator and the standard deviation of the background noise σ_n estimator. This section goes over all individual components and their implementation, which leads to the proposed spike detector.

4.2.1. Nonlinear Energy Operator

The NEO calculator implements Equation 4.1, where $x(n)$ is the output of IIR filter 2. As explained in Section 3.1, NEO measures instantaneous energy from amplitude and frequency information. The amplitude information is found in $x^2(n)$, and the frequency is found in $x(n+1)x(n-1)$.

$$\psi[x(n)] = x^2(n) - x(n+1)x(n-1) \tag{4.1}$$

4.2.2. Infinite Impulse Response Filters

The original design proposed by Yang and Mason [28] consists of two exponential filters implemented in the form of IIR filters. Equation 4.2 shows the equation of the IIR filter, where α is the feedback coefficient and $\alpha < 1$. IIR filters 2 and 3 follow the original design and are used to smooth the signal.

$$y(n) = \alpha x(n-1) + (1-\alpha)y(n-1) \tag{4.2}$$

IIR filter 1 is originally implemented to calculate the amplitude of the detected spikes. Figure 4.3 shows the original LFP recording in blue and the output of IIR filter 1 in orange. The positive and negative amplitude is calculated by looking at the amplitude difference of the highest or lowest peak and the output of IIR filter 1.

The result of IIR filter 1, visible in Figure 4.3, also shows slow waves that are visible in the signal. This is known as temporal drift, which is a slow change in LFP recordings over time and affects the reliability of LFP recording analysis. The improvement introduced to the original spike detector, as shown in Figure 4.2, is the removal of the temporal drift. The removal is done by combining IIR filter 1 and subtracting the result from the original LFP recording. This removal results in the removal of amplitude and frequency information, which is not necessary for the NEO calculator and can thus result in a higher spike detection accuracy. Filter 1 is implemented using $\alpha = \frac{1}{300}$, filter 2 using $\alpha = \frac{1}{4}$, and filter 3 uses $\alpha = \frac{3}{32}$.

4.2.3. Automatic NEO Thresholding

ANT is used to calculate a threshold for spike detection. ANT statistically analyzes the signal standard deviation and RMS frequency and is implemented using Equation 4.3 where C is a scalar, σ_n the standard deviation of the noise, and Ω_{rms} the RMS frequency. The standard deviation of the noise and the RMS frequency are both estimated, as calculating them requires too much computational power. For the scalar, a standard value of $C = 14$ is chosen.

$$Th_\psi = C\sigma_n^2\Omega_{rms}^2 \tag{4.3}$$

Estimation of σ_n

Conventionally the standard deviation of the background noise, σ_n, is calculated by calculating the standard deviation of the input signal. This has been shown to be sensitive to spike firing rate and can be better calculated by using the median absolute deviation (MAD) [30]. Equation 4.4 shows the estimation of σ_n, where the median is taken over a window of the absolute value of the output of IIR filter 2.

$$\sigma_n^{MAD} = \frac{median(|x(n)|)}{0.6745} \tag{4.4}$$

Estimation of Ω_{RMS}

Conventionally the RMS frequency is calculated by Equation 4.5, where $X(\Omega)$ is the Fourier transform of the input signal $x(n)$.

$$\Omega_{rms}^2 = \frac{\int_0^\infty \Omega^2 X^2(\Omega) d\Omega}{\int_0^\infty X^2(\Omega) d\Omega} \tag{4.5}$$

This implementation is, however, greatly affected by spike firing rate and requires an expensive computation of the Fourier transform. An alternative method is making use of the zero-cross frequency, which is the number of zero-crossings divided by twice the length of the signal. Equation 4.6 shows the estimation, where n_z is the number of zero-crossings inside the window of size N_z. Following Yang and Mason [28] the input signal should be the output of IIR filter 2, but in the input of this filter, the temporal drift removal is incorporated. The temporal drift removal also removes a lot of zero-crossings which influences the Ω_{RMS} estimator negatively and thus is chosen to take the input signal of the spike detector as input of the Ω_{RMS} estimator.

$$\Omega_{rms} = \frac{n_z}{2N_z}\pi \tag{4.6}$$

As will be explained in Section 7.1.2 the spike detector explained above did not meet the requirement for accuracy. To improve the accuracy the value of C in Equation 3 can be adjusted during analysis. With the ability to adjust C and to keep the analysis time efficient the window for both the estimation of σn and Ωrms is chosen to be the entire dataset as this results in just one threshold for the whole dataset.

4.3. Epileptiform Event Detector

As Section 3.2 explained, no epileptiform event detector for the IHKA model exists. Because no existing method is available, a new method has been devised for the detection of epileptiform events. The detector uses one general description for all forms of epileptiform events, which is visible in Table 4.1. Next to the description, an algorithm has been developed that checks the signal, using the detected spikes, for events fulfilling the general description.

The output of the epileptiform event detector consists of multiple metrics about the detected event, including start and end sample, spike frequency, number of spikes, maximum number of spikes in five consecutive seconds and the average positive and negative amplitude of the included spikes.

Table 4.1: General description of epileptiform events used for detection.

Spike Amplitude	Spike Frequency	Event Duration	Interevent Interval
$\geq 2\times$ Baseline	≥ 2 Hz	≥ 2 s	≥ 3 s

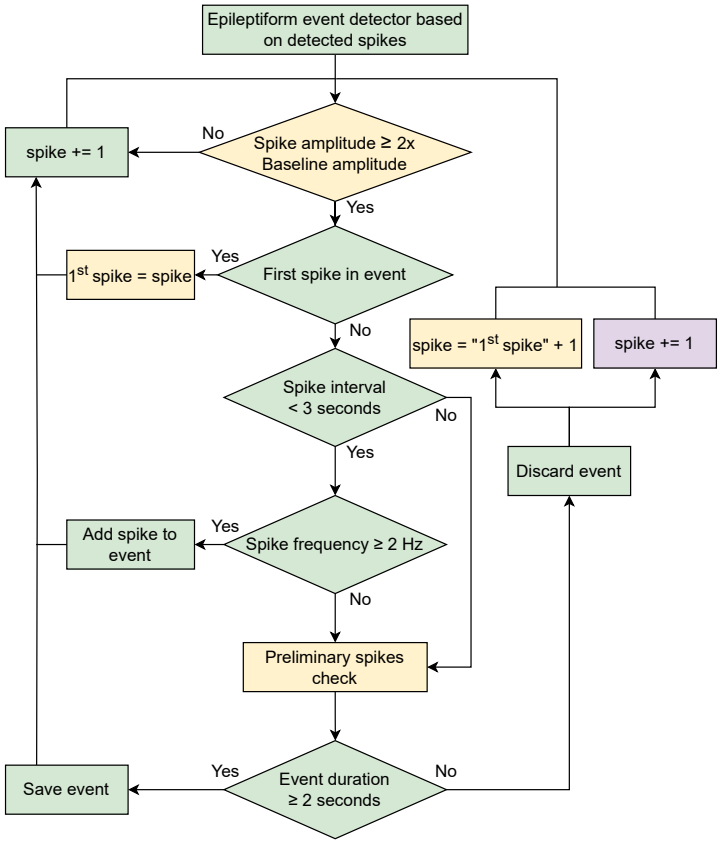

Figure 4.4: Flowchart of the epileptiform event detection algorithm with elements in green implemented in both algorithmic and hardware level. Elements in yellow are only implemented on algorithmic level, and purple elements are only implemented on hardware level.

This section covers the functioning of the epileptiform event detector. For generating the algorithm, first, baseline amplitude calculation is needed for the amplitude requirement, followed by the generation of a detection loop, and at last, a function called $preliminary\ spikes\ check$ is implemented.

4.3.1. Baseline Amplitude Calculation

The first requirement defines that the spike amplitude needs to be larger than two times the baseline amplitude. Jackson *et al.* [14] have implemented a baseline calculation block which determines the baseline amplitude value by stating that 97% of the data points in the first hour of data are lower than the baseline amplitude. The downside of this is that the method depends on spike frequency. To overcome this problem the baseline amplitude calculator determines a signal segment of 30 seconds without any spikes. From this, the middle 20 seconds have a noise level that is not influenced by any nearby spikes and from this segment, the 97^{th} percentile of all samples is taken which is then the baseline amplitude.

Another problem the baseline calculation block of Jackson *et al.* has, is the change of baseline amplitude over time. Only looking at the first hour, or in our case, the first 30 seconds without any spikes, can cause problems with a changed baseline amplitude over time in a 6-hour LFP recording. To overcome this the baseline amplitude is calculated every time a signal segment of 30 seconds without any spikes is found. To not only rely on the new amplitude value the update takes place using an IIR filter with $\alpha = 0.2$.

4.3.2. Detection Loop

As explained before, a general description of epileptiform events has been made to detect all types of events. Detecting events using this general description is done using a detection loop, which is depicted in Figure 4.4 in the form of a flowchart. The elements coloured green are implemented on both algorithmic and hardware level, the yellow elements are only implemented on algorithmic level and the purple elements are only in hardware. The hardware implementation of the epileptiform event detector will be discussed in Section 5.2.

The detection loop starts with the first detected spike and runs until the end of an LFP recording is reached. The detected spike is first checked for its amplitude; if it does not have the required amplitude, the loop skips to the next detected spike. Otherwise, it needs to be determined if this is the first spike in a possible event. If so, the spike is saved as $1^{st}spike$. If the loop is already busy detecting an event, the spike interval is checked, which refers to the interevent interval indicating that events are split if there are at least 3 seconds between them. If the spike interval is more than 3 seconds, the event detection is stopped by first running the $preliminary\ spikes\ check$, which will be explained in the next section. If the interval is less than 3 seconds, the spike frequency is checked. If the frequency is less than 2 Hz, the $preliminary\ spikes\ check$ function is run. If both the spike interval and the spike frequency fulfil the requirements, the spike is added to the event, and the loop will start with the next detected spike. If the $preliminary\ spikes\ check$ function has run, the event duration is checked; if it is at least 2 seconds, the event is saved; otherwise, it is discarded. After an event is saved, the loop will continue with the next detected spike, but when an event is discarded, the loop jumps to the spike detected after the spike saved as $1^{st}spike$. This jump happens to check all available combinations of spikes so that possible events are not missed and to find the best combination of spikes for an event. The downside of this jump is that some computational overhead is added, as some spikes are used twice for the detection loop.

4.3.3. Preliminary Spikes Check

The preliminary spikes check is one of the last steps in the event detection loop. As explained in section 2.3, HPD events can start with HVSW-like activity, which results in the start of HPD events containing higher amplitude but lower frequency spikes than the rest of the event. These lower frequency spikes can reach frequencies below 2 Hz, which causes the detection loop to ignore them. However, if these spikes are within 3 seconds of each other and this group is within 3 seconds of an event, these spikes should be included with the rest of the event if a frequency of at least 2 Hz is reached.

An example of an event where spikes that should be included in the event but are not included is given in Figure 4.5a. The detected event with the standard detection loop reaches a frequency of 3.7 Hz, and all preliminary spikes are within at least 3 seconds of another spike, and the group is within 3 seconds of the event. The preliminary spikes check function runs the detection loop from the beginning of the event over the spikes coming before the event. This is demonstrated in Figure 4.5b. The event detected has a frequency of 2.1 Hz and fulfils all the other requirements.

4.4. Epileptiform Event Classifier

The last step in the epileptiform activity detector and classifier is the classification of epileptiform events. As explained in Section 2.3, there are four different classifications for epileptiform events: spike train, HVSW, sHPD and iHPD. If we compare the characteristics of all events in Table 2.1 to the general event description Table 4.1, it is clear that the general description is most in line with spike train and HVSW. Because of this, events are automatically classified as spike train if they have a duration between 2 and 5 seconds or as HVSW if they have a duration of 5 to 20 seconds.

Events are classified as HPD based on local spike frequency. Twele *et al.* [29] stated that an HPD event has 5 continuous seconds with at least 25 spikes inside. The epileptiform event detector keeps track of the peak number of spikes inside 5 continuous seconds, and an event is classified as HPD if there are at least 25 spikes in those continuous seconds. The distinction between sHPD and iHPD is made based on duration. An event with a duration between 5 and 10 seconds is classified as sHPD, and events longer than 10 seconds are classified as iHPD. Finally, HVSW events that are longer than 20 seconds will also be classified as HPD because the description determines that HVSW cannot reach a length above 20 seconds.

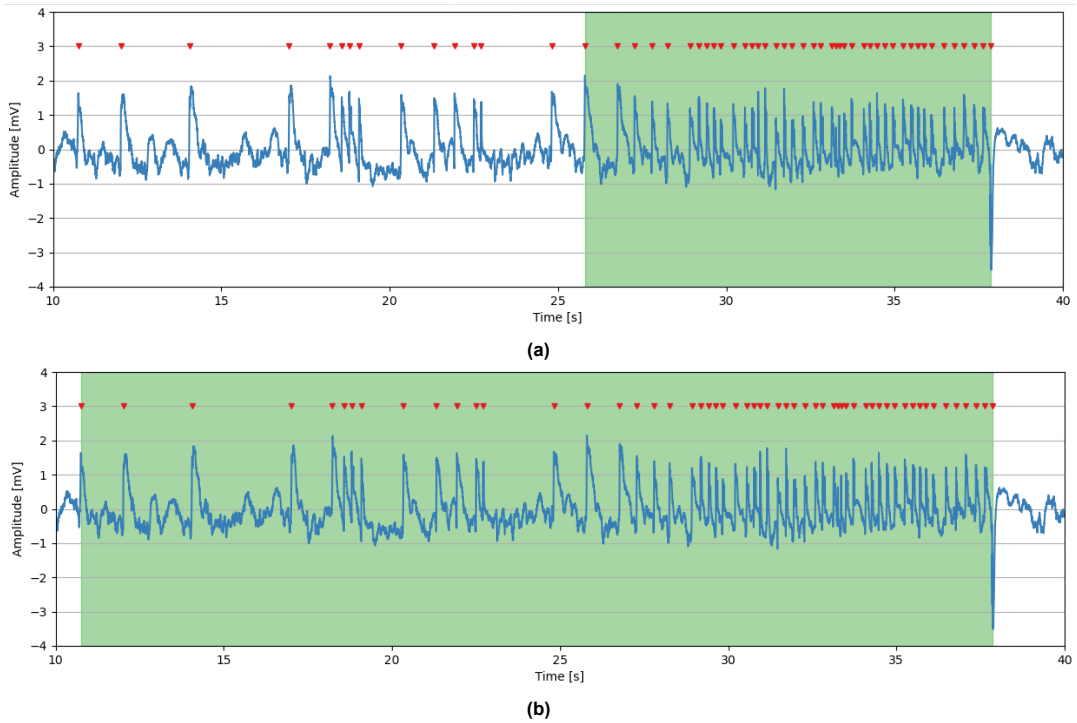

Figure 4.5: Output plot of the system with spike detection and epileptiform event detection. The LFP recording is shown in blue, detected spikes are marked with red markers, and the detected event is highlighted in green.
(a) An event detected by the detection loop without the preliminary spikes check. The event has a frequency of 3.7 Hz
(b) The same event was detected, but now with the preliminary spikes check. This results in an event with a frequency of 2.1 Hz

5

Proposed Hardware Implementation

This chapter goes over the proposed hardware implementation of the epileptiform activity detector and classifier. The hardware implementation follows the same structure as the algorithmic implementation described in the previous chapter. However, it is modified in some parts to support a real-time and, at the same time, an efficient implementation. A block diagram of the system is shown in Figure 5.1, which highlights the interconnects between the different sub-systems of the hardware implementation.

The system's input is LFP recordings converted to a 16-bit signal using 2's complement and fixed-point representation. The first bit is used as a sign bit, the following four bits are used for the integer, and the last eleven bits are used for the fraction. With this representation, all values between 16.0000 and 15.9995 can be represented in the system with a resolution of 0.0005. The samples that fall outside the range are clipped to either -16.0000 or 15.9995.

The system runs on a clock of 1000 Hz, which is the same frequency as sampling in the dataset. The output of the system are five flags: spike detector threshold calculated, spike detected, event detected, HVSW classification, and HPD classification. The threshold flag indicates if a threshold for the spike detector has been calculated; when this is done, the spike detection can start, and when a spike is detected, the spike detected flag goes high for 100 milliseconds. The flags for event detection show if an event is detected, and the HPD- and HVSW flags show the classification. The last three flags all have a delay of 5 seconds. This is due to the requirement for an HVSW or HPD event to have a duration of at least 5 seconds.

5.1. Spike detector

The spike detector is based on the block diagram visible in Figure 4.2 and is adjusted to suit a hardware-efficient implementation. This section highlights the implementation of the spike detector in hardware and the differences with the algorithmic implementation.

5.1.1. Fixed-Point Multiplication

The NEO and ANT calculators both take a square of their input data and as fixed-point values are used for the system input, their multipliers need a different implementation with respect to normal binary

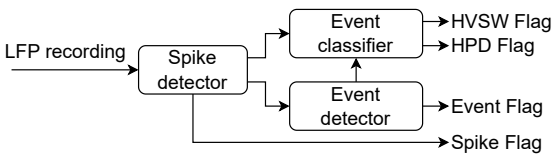

Figure 5.1: Block diagram of the hardware implementation of the proposed epileptiform activity detection and classification method.

Table 5.1: The values of $G1$, $G2$ and S used for the IIR filters implemented in hardware.

	G1	G2	S
Filter 1	2	510	9
Filter 2	3	1	2
Filter 3	3	29	5

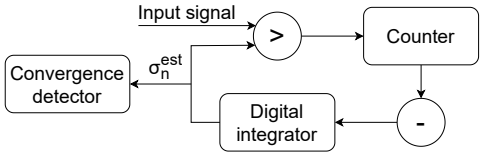

Figure 5.2: Block diagram of the estimation of σ_n implemented in hardware.

multiplications or multiplication of a fixed-point value with a binary value. During the multiplication of two binary values, the number of bits necessary for the result almost doubles ($2 \times n - 1$). When the previously explained 16-bit fixed-point is used, the multiplication results are 1 sign bit, 8 integer bits, and 22 fraction bits. Reducing a 31-bit fixed-point value back to a 16-bit fixed point is done by taking the sign bit, the 4 lowermost bits of the integer, and the 11 uppermost bits of the fraction.

5.1.2. Infinite Impulse Response Filters

The exponential filters implemented as IIR filters on an algorithmic level with Equation 4.2 are modified to be more efficient in hardware. In Equation 4.2, $\alpha < 1$ which means that multiplication is carried out by the numerator of α and a division by the denominator of α. This multiplication and division happen twice, once for α and once for $1 - \alpha$. Divisions are generally considered costly operations in hardware but can be replaced by, for example, a right-shift operation. A right-shift operation can be considered to be a division by 2^S where S is the number of bits shifted. Rewriting Equation 4.2 with right shifts gives Equation 5.1, where G_1 and G_2 are the nominators and $\frac{1}{2^S}$ denotes the right bit shift by S.

$$y(n) = G_1 \frac{x(n-1)}{2^S} + G_2 \frac{y(n-1)}{2^S} \tag{5.1}$$

In hardware, the same three IIR filters are implemented, and to suit the different implementations of the equation, the values of α have been slightly modified. The new values of G_1 and G_2 are the nominators of α and $1 - \alpha$ respectively and $\frac{1}{2^S}$ represents the denominator. The values of G_1, G_2 and S are given in Table 5.1.

5.1.3. Estimation of σ_n

On the algorithmic level, the estimation of σ_n is done by calculating MAD and using Equation 4.4. If implemented in hardware, this is a costly operation as it requires sorting a large array of input samples. Yang and Jason [28] introduce a hardware-efficient implementation for estimating σ_n by following the statistical theory that the probability of Gaussian noise exceeding σ_n is known to be 0.159. Figure 5.2 shows a block diagram of the σ_n estimator. The input signal is compared with the estimated σ_n, σ_n^{est}, and outputs a $'1'$ if the input signal is greater. The amount of $'1'$s inside a window of size M is counted and subtracted from $0.159 \times M$. This result is fed into a digital integrator and is then used to update σnest every M clock cycles. The loop keeps updating until a convergence is detected, after which the system outputs the converged value σ_n.

5.1.4. Estimation of Ω_{RMS}

The implementation of the Ω_{RMS} in hardware is, like on algorithmic level, based on the zero-cross frequency. Figure 5.3 shows a block diagram of the counter, which calculates the number of zero-crossing n_z in a window of N_z. The sign bit is compared to the previous sign bit, and when it differs, the XOR-gate will output a $'1'$, which is added to the current value of n_z. After the counter reaches the value of N_z, the system will be reset.

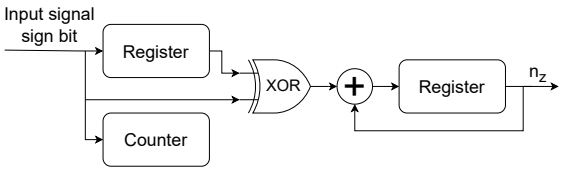

Figure 5.3: Block diagram of the estimation of Ω_{RMS} implemented in hardware.

5.2. Epileptiform Event Detection

Section 4.3 has described the working of the proposed epileptiform event detector, which is visualized by the flowchart in Figure 4.4, where the green and purple elements are implemented in hardware. The yellow elements of the detector have been left out of the hardware implementation as the implementation should be real-time and efficient. The changes that have been made to the working of the event detector will be highlighted in this section.

5.2.1. Amplitude Calculation and Check

The spike detector proposed in Section 4.2 does amplitude calculation for both the positive and negative peaks of a spike. This is done by finding the highest and lowest value in the 100 milliseconds after a spike is detected. Hardware would require two comparators and registers that compare 100 consecutive samples and save the highest and lowest values. This in itself is not a costly operation, but the spike amplitude is only used for the amplitude check done by the event detector.

The amplitude check of the event detector requires 20 seconds, thus 20.000 continuous samples, from which the 97^{th} percentile is calculated. This would be a costly operation if implemented in hardware as 20.000 samples need to be saved and sorted to find the 97^{th} percentile. This, in combination with the fact that from analysis, it was found that almost all detected spikes fulfil the amplitude requirement, which caused the amplitude calculation and check to be left out of the hardware implementation.

5.2.2. Nested Loop

The nested detection loop is highlighted in Figure 4.4 by the yellow element containing $spike = 1^{st}spike + 1$. This implementation mostly impacts the real-time working of the system but also requires extra memory. The real-time working of the system is impacted greatly by the nested loop as it starts the event detection using a spike that was detected a few seconds prior. This helps with the correct detection of the start and end of events but will cause extra and variable delays if added, which is undesirable for a real-time system, and thus, it was chosen to be left out of the hardware implementation.

5.2.3. Priliminary Spikes Check

The preliminary spikes check was introduced, as explained in section IV, to correctly detect the start of HPD events. A hardware implementation of this function is not viable as the algorithmic, non-real-time implementation checks the spikes in front of a detected event to correct the detection. If this were to be implemented in hardware a delay of the longest detectable event needs to be added to be able to show a detected event with preliminary spikes, which would result in the system not running in real-time. The effect of leaving out the preliminary spikes check is that the beginning of some HPD events will not be detected correctly, but in the end, the design is able to run in real-time.

5.3. Epileptiform Event Classification

The epileptiform event classifier is also simplified with respect to the presented version in Section 4.4. The events are classified into only HVSW and HPD as these are the most important factors in the development of epilepsy. As explained in section 4.4, an event will be classified automatically as an HVSW event unless five continuous seconds with at least 25 spikes are detected.

The classifier is implemented using a ring buffer, which saves the moment in time when a spike is detected. Two variables keep track of the first and last spike added to the buffer. The value in the ring buffer of the spike is compared to the current time to see if the first added spike needs to be removed, if so the variable for the first spike is incremented. Using the values of the first and last spike added

the number of spikes in the previous 5 seconds is checked. This is done by subtracting the value for the first spike added from the value of the last spike added and taking the modulo using the size of the ring buffer. If the number of spikes in the previous 5 seconds is tracked, when the event detection flag goes high, a check on the amount of spikes determines the classification. During the time the event detection flag is high, the classification can change from HVSW to HPD if the number of spikes in the buffer reaches 25. The classification relies on event detection and thus has a delay of at least 5 seconds.

6

Experimental Setup

6.1. Performance Metrics

The performance results of the spike detector, event detector, and event classifier are represented using one or more of the following metrics. (1) Accuracy, which indicates the rate of correct detections or classifications, illustrated in Equation 6.1 and given in percentage. (2) Sensitivity, as part of Equation 6.2, showing the rate of positive detections or classifications, which are correctly identified. (3) Precision, as part of Equation 6.3, indicates the number of true detections or classifications over the total number of true events. These metrics use True Positive (TP), False Positive (FP), and False Negative (FN) detection. True Negative (TN) is not considered, as this results in a large bias by all samples not being a spike or event and not being detected.

$$Accuracy = \frac{TP}{TP + FP + FN} \tag{6.1}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{6.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{6.3}$$

6.2. Spike Detector

The spike detector is evaluated using a synthetic dataset, which consists of 8 recordings with noise levels ranging from 0.05 dB to 0.2 dB at an interval of 0.05 dB [27]. The naming structure used for the different datasets starts with E (easy) or D (difficult), which refers to the difficulty level of clustering the spikes (not important for this work) and ends with the value of the noise level. With these datasets, the proposed spike detector can be evaluated and compared to three different methods: the standard implementation of NEO and ANT, ASO [25], and DWT [27]. As the implementations of ASO and DWT only have results reporting accuracy, this is the only metric used to evaluate performance.

6.3. Epileptiform Event Detector and Classifier

The performance results of the epileptiform event detector and classifier are gathered by an animal epilepsy expert who reviewed the output of the epileptiform event detector and classifier. For the event detector, 48 hours and for the classifier, 108 hours of input LFP recordings from the IHKA model are used, originating from 8 and 9 different mice, respectively. After the evaluation of the event detector, the algorithm was adjusted, and with the new version, the event classifier was evaluated. During the analysis, the TP, FP, and FN event detections and classifications were counted, from which the performance metrics were calculated.

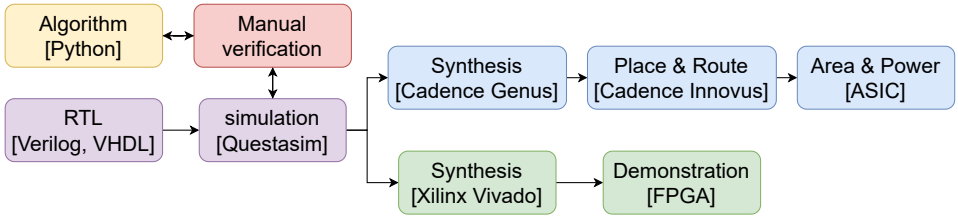

Figure 6.1: Development flow of the hardware implementation as an ASIC and in an FPGA.

6.4. Hardware Implementation

The development flow of the hardware implementation is shown in Figure 6.1. The algorithm first developed in Python has been transferred to hardware using Verilog and VHDL from which simulations have been done in Questasim. The simulations are compared to the Python implementation to verify its working. The next step splits up the process into the workflow of synthesizing an ASIC and the development of an FPGA demonstration.

6.4.1. Application Specific Integrated Circuit (ASIC)

The workflow of the ASIC implementation is highlighted in blue in Figure 6.1. Using Cadence Genus a worst corner synthesis of the design is done using the TSMC 40nm library. The system clock frequency is set to a frequency of 5000 Hz, which is the lowest clock frequency allowed by Genus. From the worst corner analysis, results on area and power are generated for the spike detector, event detector, event classifier, and the full design. Using the synthesized design, a place and route is done using Cadence Innovus using a density of 0.7. From the place and route results on area and power of the ASIC design are reported.

6.4.2. Field Programmable Gate Array (FPGA)

A demonstration of the design is shown using a PYNQ-Z1 board containing a ZYNQ-7000 series FPGA. The system's clock is set to 1000 Hz, and 300.000 16-bit values in 2's complement and fixed point representation are loaded into the BRAM and used as the input signal. The FPGA synthesis, done using Xilinx Vivado, gives results on power usage and area, which are reported in the available logic blocks on the FPGA.

7 Results

In this chapter, the results of the proposed algorithmic and hardware implementations are presented. First, the algorithmic results are gathered for each individual component, and then the results for the complete algorithm are shown. Second, the hardware implementation results cover the ASIC and FPGA implementation.

7.1. Algorithm

This section covers the results gathered on the algorithmic implementation in Python. This section first covers the spike detector, epileptiform event detector, epileptiform event classifier, and, finally, the complete algorithm.

7.1.1. Spike Detector

The average accuracy results of the four spike detection methods are compared using a boxplot illustrated in Figure 7.1. The Wavelet transform has an average accuracy of 89.5% and ASO of 91.1%. Both the standard and improved version of the NEO and ANT implementation achieve a higher accuracy, the standard version achieves an accuracy of 92.9%, while the improved version achieves an accuracy of 93.3%. This is a marginal improvement that is achieved by the removal of the temporal drift. This improvement is done using an IIR filter that was initially implemented to do spike amplitude calculations, and because of this, no overhead is added to the spike detection. Next to the fact that the improved version achieves the highest accuracy, it is also visible that the minimum accuracy value is higher for the improved version. This indicates that this method is more robust to the noise present in the different datasets than the other methods. An overview of the individual results on all datasets is given in Figure A.1.

7.1.2. Epileptiform Event Detector

Figure 7.2 shows the average accuracy, sensitivity and precision retrieved from the manual analysis of the output of 48 hours of input data to the epileptiform event detector. The event detector has an average accuracy of 93.6%, with a precision of 99.0% and a sensitivity of 94.0%. The accuracy, sensitivity and precision results on all the individual datasets are visible in Figure A.2. In Figure 7.2, it is visible that there is one outlier for both accuracy and sensitivity results. This outlier is dataset 5, and the low sensitivity indicates that a lot of events are missed in the detection. During the manual analysis of this dataset, it was found that the spike detection accuracy is not high enough to support an accurate event detection. The spike detector is improved as the event detection entirely relies on spike detection. The improvement introduced is the ability to adjust the scalar C in the ANT shown in Equation 4.3. This results in a small portion of manual work during analysis but will give better overall results on sensitivity and accuracy.

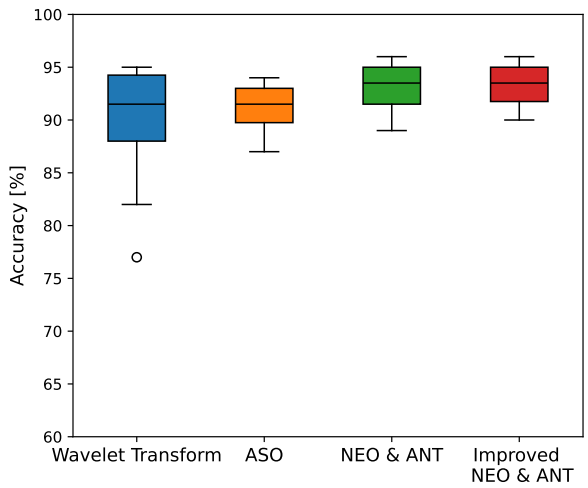

Figure 7.1: Accuracy of the spike detector, compared to three different methods using a synthetic dataset.

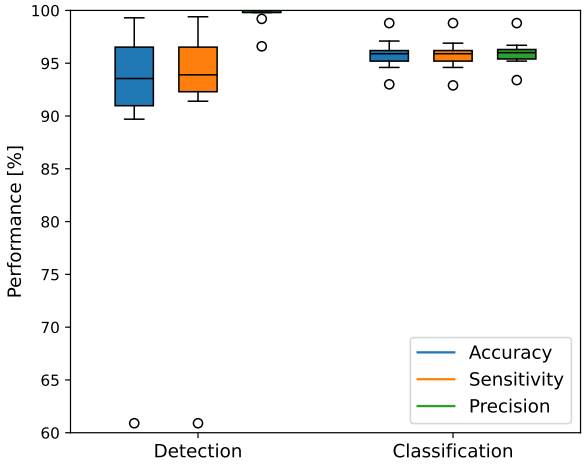

Figure 7.2: Accuracy, Sensitivity and Precision results from both the epileptiform event detector (on the left) and classifier (on the right) extracted from the manual analysis of the algorithm.

7.1.3. Epileptiform Event Classifier

The accuracy, sensitivity and precision results from the manual analysis of 108 hours of input data for the event classifier are shown in Figure 7.2. The individual accuracy, sensitivity, and precision results per dataset are visible in Figure A.3. In the analysis of the event classifier, the ability to adjust the scalar C in the equation of ANT for the spike detector has been included. The event classifier reaches an accuracy of 95.8% with a sensitivity of 95.8% and a precision of 95.8%. When comparing the results on detection and classification in Figure 7.2, we can see that the classifier achieves a higher consistency over the three metrics on all the datasets. The consistency is visible in the small spread indicated by the quartile and maximum data values for classification. The results are directly correlated as both the detector and classifier fully rely on the spike detector. However, we cannot directly compare both results and give an impact value to the added ability to adjust scalar C. Still, this stability in the classifier results shows that the adjustment of scalar C positively impacts the results with little manual work added.

7.1.4. Complete System

An overview of the output of the algorithmic implementation is shown in Figure 7.3 B, where the detected spikes, the detected event and the classification are shown. The coloured area indicates a detected event, and each classification has a dedicated colour (spike train: purple, HVSW: green, sHPD: orange, iHPD: yellow). Examples of the four different classifications are shown in Appendix A.2.

Table 7.1 shows the time reduction achieved by the automated detection and classification of epileptiform activity implemented using Python. The manual analysis of 108 hours of input data takes 27 hours for an expert to analyze. The algorithm has reduced this to 20 minutes of computations and thus achieves a time reduction of 98.8%.

Table 7.1: Manual analysis time compared to computational analysis time of LFP recordings used in preclinical studies.

Input data	108 hours
Manual analysis	27 hours
Computation time	20 minutes
Time reduction	98.8%

7.2. Hardware

A simulation result of the hardware implementation is visible in Figure 7.3 C, where the detected spikes, the detected event and the HVSW and HPD classification output signals are shown. The figure shows

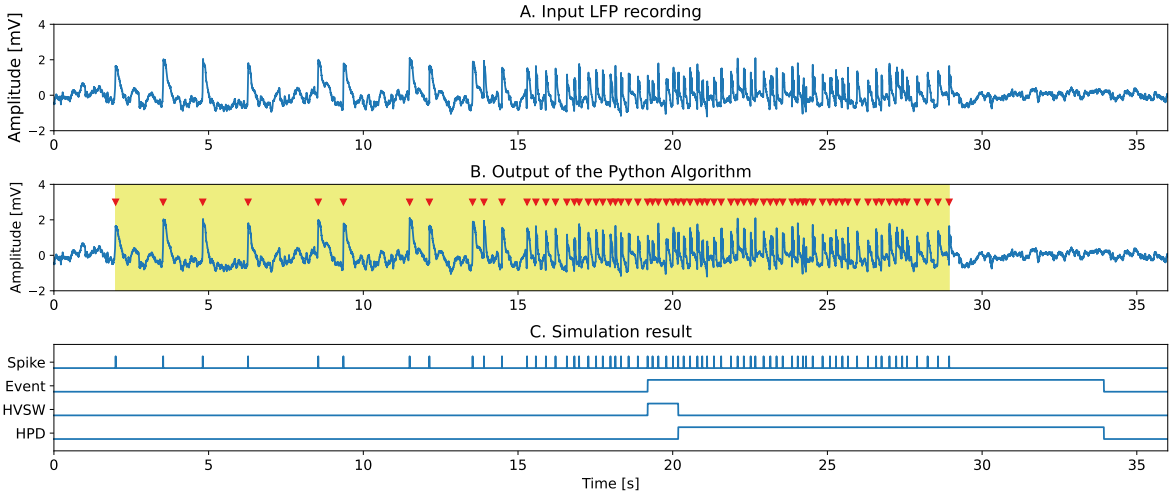

Figure 7.3: Output results of both the algorithmic and hardware implementation. (a) Input LFP recording. (b) The output of the algorithmic implementation in Python: red markers indicate the detected spikes, the coloured area indicates the detected event, and the colour indicates the classification. (c) Output signals of the hardware simulation where $Spike$ indicates a detected spike, $Event$ a detected event and $HVSW$ and HPD represent the classification of a detected event.

the detected spikes in the same moment in time as the algorithmic implementation and, as discussed in Section 5, the event detection and both classification signals have a delay of 5 seconds. Next to this, it is clear from this figure that the preliminary spikes check function is not implemented in hardware as the event's start is missed in detection. Nevertheless, Figure 7.3 C shows a real-time implementation of the proposed detection and classification method.

During synthesis, chip area and power results are generated for both the ASIC and FPGA, which are discussed in this section, but timing analysis results are also gathered. For both the ASIC and FPGA, there are no timing constraints resulting from the low clock frequency used in the design.

7.2.1. Application Specific Integrated Circuit (ASIC)

This section will first introduce results gathered on the synthesis of the ASIC, after which the results of the place and route of the design are discussed.

Synthesis

The area and power results gathered from the synthesis are shown in Figure 7.4. It is clear from the figure that the spike detector, together with the interconnect, takes up the most significant part of the area and uses the most power. Next, it is visible that the distribution in the area and power usage is correlated as the percentage per part of the design is almost equal. The area of the spike detector could be reduced by decreasing the input bit size, which results in a lower resolution throughout the whole system, as the size of the input signal is used everywhere. From testing, it was found that the accuracy reaches an undesirably low level when the input size is decreased slightly, and thus, this was not a viable option.

Place & Route

The area and power results of the place and route are shown in Table 7.2 from which is visible that the ASIC design reaches an area of $9114\ \mu m^2$, which is 13.6% larger than the synthesis achieves. This is because an ASIC gives a more realistic result on the area as the netlist resulting from the synthesis is realistically routed in the design. The dynamic and static power is also shown in Table 7.2; the static power is the result of the leakage power, and the dynamic power is the result of the internal and switching power. The dynamic power is low compared to the static power due to the low clock frequency used and the efficient implementation of the design. Figure 7.6 shows the layout gathered from the place and route.

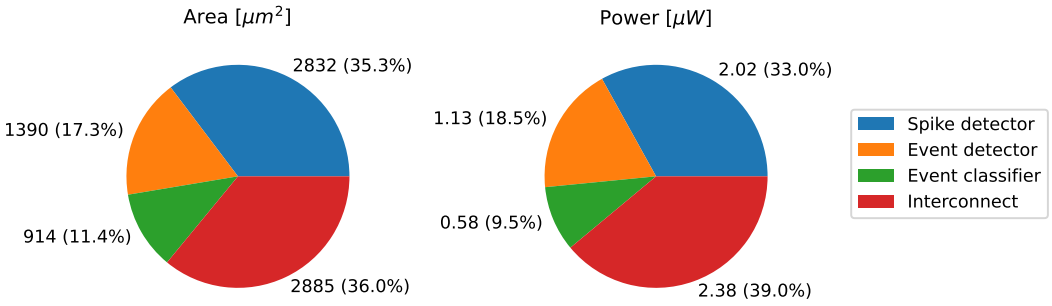

Figure 7.4: Chip area and power results from the synthesis of the design using Genus and the TSMC 40nm library.

Table 7.2: Chip area and power results of the placed and routed design and the FPGA implementation.

	ASIC	**FPGA**
Tool	Cadence Genus	Xilinx Vivado
Technology	TSMC 45nm	PYNQ-Z1 board with ZYNQ-7000 FPGA
Supply voltage	0.99 V	1.00 V
Clock	5000 Hz	1000 Hz
Area	9114μm^2	815 LUTs 427 Registers 8 DSPs
Static power	6.659 μW	114 mW
Dynamic power	0.065 μW	58 mW

7.2.2. FPGA

Figure 7.5 shows the FPGA during a demonstration of the algorithm; all the system outputs are routed to LEDs, the input data is stored on the BRAM, and the reset is linked to a switch. It is shown that the algorithm is successfully implemented on an FPGA to show that it works as a Brain-Machine Interface (BMI). The area and power results are reported in Table VI next to the ASIC results; a comparison is impossible as both implementations use different technologies. In the power results of the FPGA, we can also see here that the static power consumption is higher than the dynamic power consumption. Next, the total power consumption is orders of magnitude larger than the ASIC due to the logic blocks used on the FPGA and the extensive interconnect available.

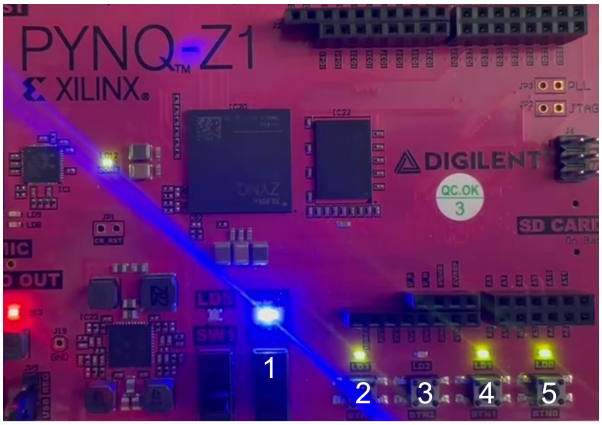

Figure 7.5: FPGA demonstration with all the outputs linked to the LEDs on the PYNQ-Z1 board. LED 1 the threshold calculated flag, 2 the HPD classification flag, 3 HVSW the classification flag, 4 the event detected flag and 5 the spike detected flag.

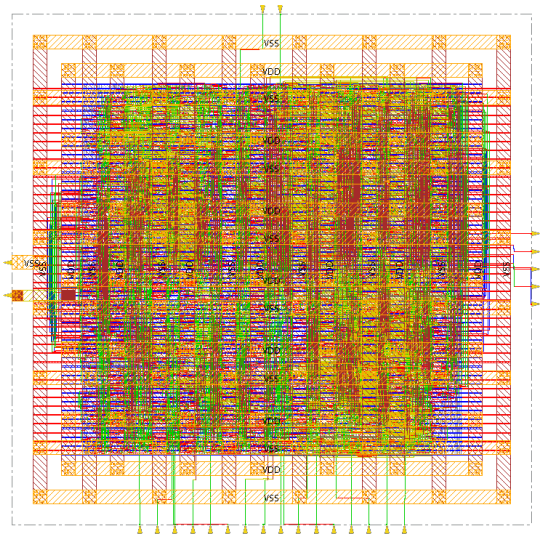

Figure 7.6: Layout retrieved from the place and route of the synthesized design

8

Conclusion

Recent preclinical studies using the IHKA mouse models to research TLE have used manual analysis of the LFP recordings originating from these studies. In this manual analysis, epileptiform activity, which indicates the presence and evolution of epilepsy, is analyzed and reported. Manual analysis is a time-consuming process, and it is also prone to observer bias and error. An automated tool to detect and classify epileptiform activity would streamline research, improve accuracy, and accelerate advancements in epilepsy studies.

In this thesis, a new method is proposed for the detection and classification of epileptiform activity using LFP recordings gathered from a preclinical study on the IHKA mouse model. The proposed method is based on a spike detection stage, which is followed by an epileptiform event detector, which uses a general description of all types of epileptiform events. The last stage is the epileptiform event classifier, which classifies events based on duration and local spike frequency into one of four groups. The proposed detection and classification method is implemented as an algorithm in Python and reaches a general epileptiform event detection accuracy of 93.1%. Epileptiform events are classified with an accuracy of 95.8%, and the implemented algorithm reaches a time reduction of 98.8% over manual analysis.

This work achieves superior sensitivity, 94.0% for detection and 95.8% for classification, over the existing method ($86 - 90\%$) introduced by Thielmann *et al.* which is the one comparable work available. Next to this, a superior time reduction of 98.8% is achieved over the $> 80\%$ by Theilmann *et al.* This is due to the automated level of the algorithm that has not been seen before because there is almost no need for manual input. A real-time hardware implementation has been developed for the algorithm using HDL. Power and area results have been gathered using the TSMC 40nm library and a Zynq 7000-series FPGA. The work was demonstrated using the FPGA, which shows the potential of a brain-machine interface (BMI). This work streamlines research and accelerates advancements in epilepsy studies due to the complete package of a spike detector, epileptiform event detector and epileptiform event classifier which is introduced, the superior accuracy that is achieved, and the time reduction that has been realised possible.

Future Work

If this research were to be continued, the following research areas are recommended to be pursued:

- **Introducing machine learning:** The current method does not use machine learning as no training set is available. With the proposed algorithmic implementation, this problem is solved, and the extracted events and their classification can be used as a training set. Using machine learning will have the advantage that it is robust and can extrapolate patterns beyond the training set, which leads to better performance over varying datasets. Next to this, a real-time implementation in hardware will benefit from machine learning as detecting events does not have to rely on a minimum duration of 5 seconds.

- **Extending the analysis:** Next to the extracted epileptiform activity, more information can be extracted from the LFP recordings. The form of spikes and their development over time can be analyzed by clustering the interictal spikes. Generalized seizures can be detected and predicted, and finally, the vigilance state of a mouse can be detected. Extracting this kind of information can further automate and streamline preclinical studies.

Bibliography

[1] T. A. Milligan, “Epilepsy: a clinical overview,” *The American Journal of Medicine*, vol. 134, no. 7, pp. 840–847, 2021.

[2] K. D. Laxer, E. Trinka, L. J. Hirsch, F. Cendes, J. Langfitt, N. Delanty, T. Resnick, and S. R. Benbadis, “The consequences of refractory epilepsy and its treatment,” *Epilepsy & behavior*, vol. 37, pp. 59–70, 2014.

[3] M. Lévesque and M. Avoli, “The kainic acid model of temporal lobe epilepsy,” *Neuroscience & Biobehavioral Reviews*, vol. 37, no. 10, pp. 2887–2899, 2013.

[4] P. Kwan and M. J. Brodie, “Early identification of refractory epilepsy,” *New England Journal of Medicine*, vol. 342, no. 5, pp. 314–319, 2000.

[5] L. Welzel, A. Schidlitzki, F. Twele, M. Anjum, and W. Löscher, “A face-to-face comparison of the intra-amygdala and intrahippocampal kainate mouse models of mesial temporal lobe epilepsy and their utility for testing novel therapies,” *Epilepsia*, vol. 61, no. 1, pp. 157–170, 2020.

[6] C. E. Stafstrom and L. Carmant, “Seizures and epilepsy: an overview for neuroscientists,” *Cold Spring Harbor perspectives in medicine*, vol. 5, no. 6, p. a022426, 2015.

[7] L. Welzel, D. H. Bergin, A. Schidlitzki, F. Twele, M. Johne, P. Klein, and W. Löscher, “Systematic evaluation of rationally chosen multitargeted drug combinations: a combination of low doses of levetiracetam, atorvastatin and ceftriaxone exerts antiepileptogenic effects in a mouse model of acquired epilepsy,” *Neurobiology of disease*, vol. 149, p. 105227, 2021.

[8] M. D. Mardones, K. D. Rostam, M. C. Nickerson, and K. Gupta, “Canonical wnt activator chir99021 prevents epileptogenesis in the intrahippocampal kainate mouse model of temporal lobe epilepsy,” *Experimental Neurology*, vol. 376, p. 114767, 2024.

[9] R. A. Bergstrom, J. H. Choi, A. Manduca, H.-S. Shin, G. A. Worrell, and C. L. Howe, “Automated identification of multiple seizure-related and interictal epileptiform event types in the eeg of mice,” *Scientific reports*, vol. 3, no. 1, p. 1483, 2013.

[10] “NeuroScore — datasci.com,” https://www.datasci.com/products/software/neuroscore, [Accessed 29-06-2024].

[11] “EEG Software - Natus — natus.com,” https://natus.com/neuro/eeg-aeeg/eeg-software/, [Accessed 29-06-2024].

[12] Z. Zeidler, M. Brandt-Fontaine, C. Leintz, C. Krook-Magnuson, T. Netoff, and E. Krook-Magnuson, “Targeting the mouse ventral hippocampus in the intrahippocampal kainic acid model of temporal lobe epilepsy,” *eneuro*, vol. 5, no. 4, 2018.

[13] I. B. Iotchev, D. A. Perevozniuk, I. Lazarenko, M. F. Perescis, E. Sitnikova, and G. van Luijtelaar, “The “twin peaks” method of automated spike-wave detection: A two-step, two-criteria matlab application,” *Journal of Neuroscience Methods*, p. 110199, 2024.

[14] J. J. Kyle, S. Sharma, G. Tiarks, S. Rodriguez, and A. G. Bassuk, “Fast detection and quantification of interictal spikes and seizures in a rodent model of epilepsy using an automated algorithm,” *Bio-protocol*, vol. 13, no. 6, 2023.

[15] L. Wei, H. Boutouil, R. R. Gerbatin, O. Mamad, M. Heiland, C. R. Reschke, F. Del Gallo, P. F. Fabene, D. C. Henshall, M. Lowery *et al.*, “Detection of spontaneous seizures in eegs in multiple experimental mouse models of epilepsy,” *Journal of Neural Engineering*, vol. 18, no. 5, p. 056060, 2021.

[16] W. Theilmann, B. Gericke, A. Schidlitzki, S. M. M. Anjum, S. Borsdorf, T. Harries, S. L. Roberds, D. J. Aguiar, D. Brunner, S. C. Leiser *et al.*, “Novel brain permeant mtorc1/2 inhibitors are as efficacious as rapamycin or everolimus in mouse models of acquired partial epilepsy and tuberous sclerosis complex,” *Neuropharmacology*, vol. 180, p. 108297, 2020.

[17] S. M. M. Anjum, C. Käufer, R. Hopfengärtner, I. Waltl, S. Bröer, and W. Löscher, “Automated quantification of eeg spikes and spike clusters as a new read out in theiler’s virus mouse model of encephalitis-induced epilepsy,” *Epilepsy & Behavior*, vol. 88, pp. 189–204, 2018.

[18] X. Liu, X. Yang, and N. Zheng, “Automatic extracellular spike detection with piecewise optimal morphological filter,” *Neurocomputing*, vol. 79, pp. 132–139, 2012.

[19] S. B. Wilson and R. Emerson, “Spike detection: a review and comparison of algorithms,” *Clinical Neurophysiology*, vol. 113, no. 12, pp. 1873–1881, 2002.

[20] R. Harrison, “A low-power integrated circuit for adaptive detection of action potentials in noisy signals,” in *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No.03CH37439)*, vol. 4, 2003, pp. 3325–3328 Vol.4.

[21] M. Rizk and P. D. Wolf, “Optimizing the automatic selection of spike detection thresholds using a multiple of the noise level,” *Medical & biological engineering & computing*, vol. 47, pp. 955–966, 2009.

[22] S. Kim and J. McNames, “Automatic spike detection based on adaptive template matching for extracellular neural recordings,” *Journal of neuroscience methods*, vol. 165, no. 2, pp. 165–174, 2007.

[23] H. Ocak, “Automatic detection of epileptic seizures in eeg using discrete wavelet transform and approximate entropy,” *Expert Systems with Applications*, vol. 36, no. 2, Part 1, pp. 2027–2036, 2009.

[24] T. Prasanth, J. Thomas, R. Yuvaraj, J. Jing, S. S. Cash, R. Chaudhari, T. Y. Leng, R. Rathakrishnan, S. Rohit, V. Saini, B. M. Westover, and J. Dauwels, “Deep learning for interictal epileptiform spike detection from scalp eeg frequency sub bands,” in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 2020, pp. 3703–3706.

[25] Z. Zhang and T. G. Constandinou, “Adaptive spike detection and hardware optimization towards autonomous, high-channel-count bmis,” *Journal of Neuroscience Methods*, vol. 354, p. 109103, 2021.

[26] S. Mukhopadhyay and G. Ray, “A new interpretation of nonlinear energy operator and its efficacy in spike detection,” *IEEE Transactions on Biomedical Engineering*, vol. 45, no. 2, pp. 180–187, 1998.

[27] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, “Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering,” *Neural computation*, vol. 16, no. 8, pp. 1661–1687, 2004.

[28] Y. Yang and A. J. Mason, “Hardware efficient automatic thresholding for neo-based neural spike detection,” *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 4, pp. 826–833, 2017.

[29] F. Twele, K. Töllner, M. Bankstahl, and W. Löscher, “The effects of carbamazepine in the intrahippocampal kainate model of temporal lobe epilepsy depend on seizure definition and mouse strain,” *Epilepsia Open*, vol. 1, no. 1-2, pp. 45–60, 2016.

[30] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, “Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering,” *Neural computation*, vol. 16, no. 8, pp. 1661–1687, 2004.

A

Figures

A.1. Spike detection, event detection, event classification results per dataset

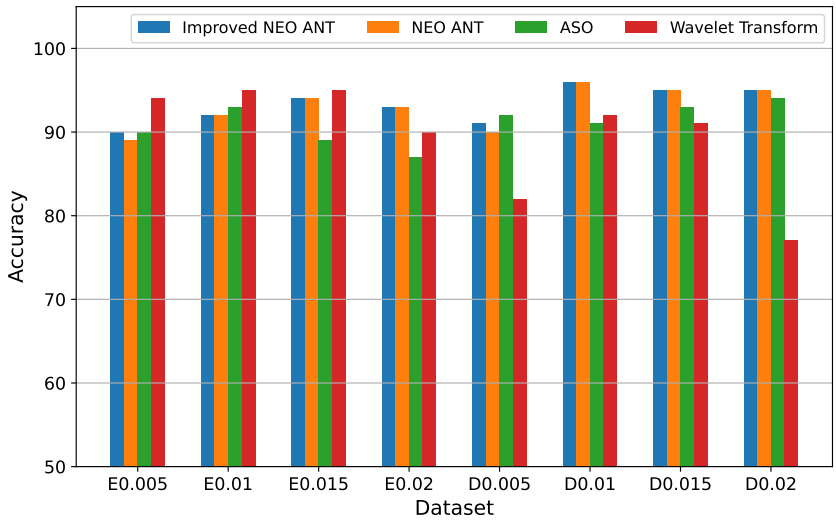

Figure A.1: Accuracy of the spike detector compared to three different methods using a synthetic dataset

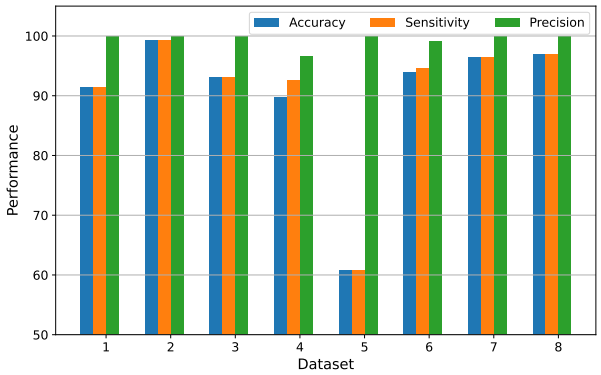

Figure A.2: Accuracy, Sensitivity and Precision results of the epileptiform event detector from manual analysis of the output

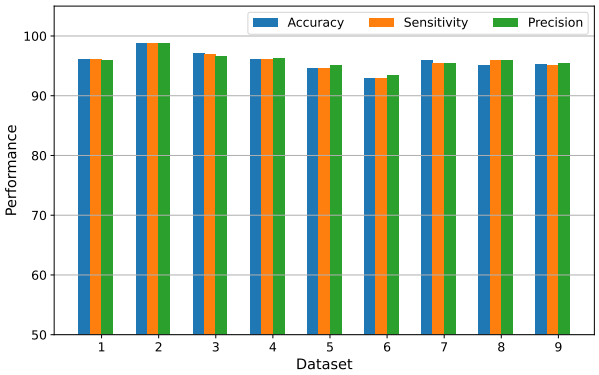

Figure A.3: Accuracy, Sensitivity and Precision results of the epileptiform event classifier from manual analysis of the output

A.2. Event detection and classification examples

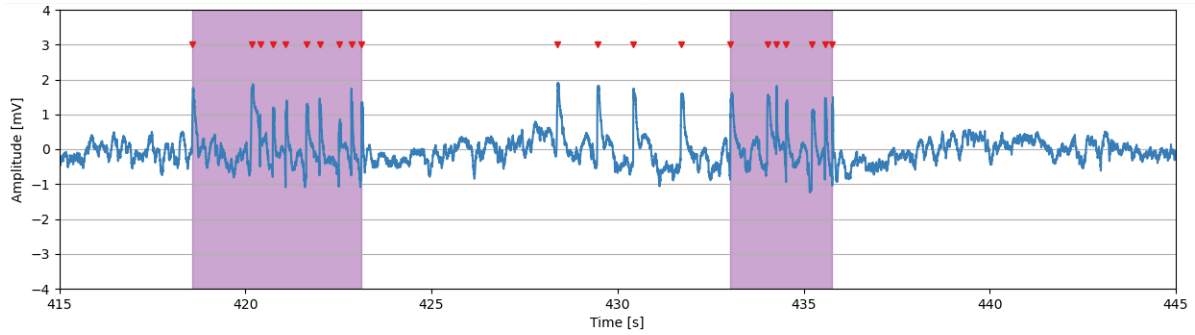

Figure A.4: Example of events classified as spike trains. Spikes are marked with red markings and the events are marked in purple.

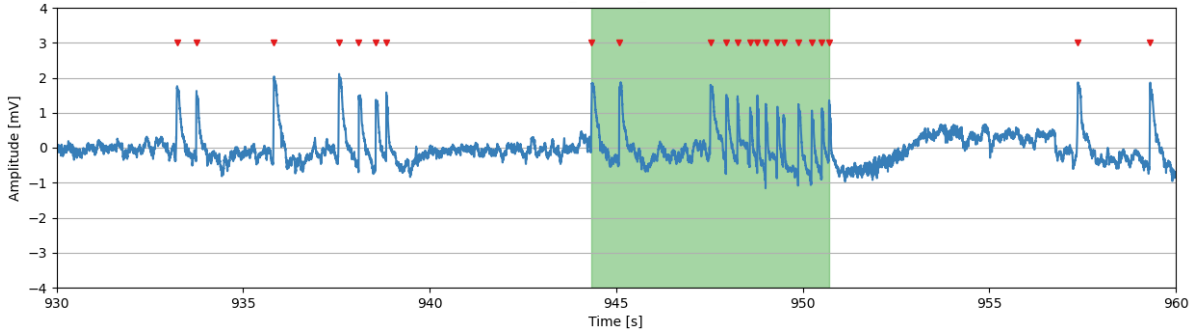

Figure A.5: Example of events classified as HVSW. Spikes are marked with red markings and the events are marked in green.

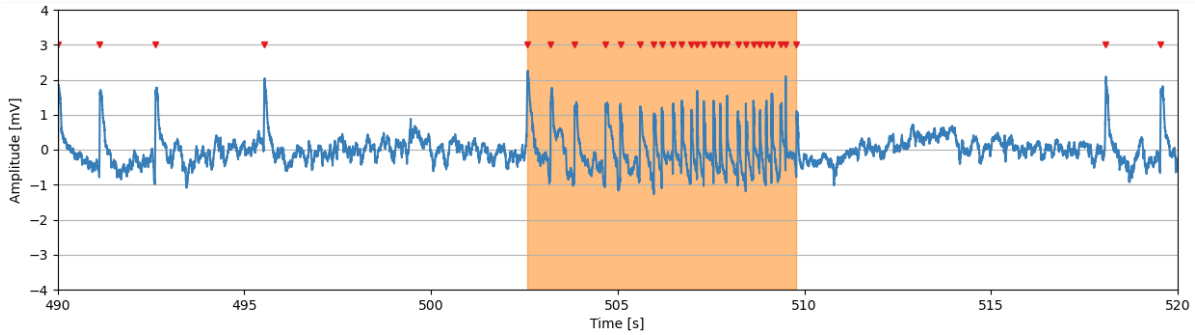

Figure A.6: Example of events classified as sHPD. Spikes are marked with red markings and the events are marked in orange.

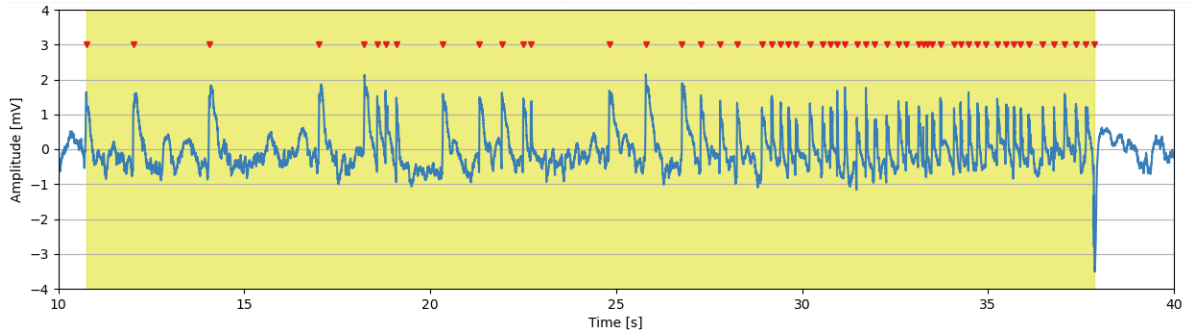

Figure A.7: Example of events classified as iHPD. Spikes are marked with red markings and the events are marked in yellow.