Master's Thesis - Electrical and Electronic Section

Feature ranking for Seizure Detection on FPGA

Chatagny Salma

July 15, 2022

Supervisor

Alexandre Schmid
alexandre.schmid@epfl.ch

Laboratory

Biomedical and Neuromorphic Microelectronic Systems Research Group (BNMS)

EPFL

Contents

Acronyms

CFS Correlation Feature Selection. 5

DMA Direct Memory Access. 31

DWT Discrete Wavelet Transform. 2, 6, 7, 10, 24, 26, 28, 29

EEG Electroencephalogram. 5, 13

FD Fractal Dimension. 10

FFT Fast Fourier Transform. 9, 24, 29

FOM figure of merits. 3, 4, 12, 14, 20, 22, 23, 25, 28

FP False Positive. 15, 20

FSM Finite State Machines. 31–35

HLS High Level Synthesis. 5

IQR Interquartile Range . 8, 9

LASSO Least Absolute Shrinkage and Selection Operator. 1, 5, 6, 17, 19, 21, 23, 24, 27, 37, 43

LDA Linear Discriminant Analysis. 5

MI Mutual Information. 5, 6, 16, 17

MIQ Mutual Information Quotient. 17

MRMR Maximum Relevance Minimum Redundancy. 1, 5–7, 16, 18, 19, 23, 27, 37, 43

PSD Power Spectral Density. 5, 6, 9, 17, 24

RF Random Forest. 1, 5, 6, 18, 19, 23, 24, 27, 37, 43

RMS Root Mean Square. 7

SVM Support Vector Machine. 5, 18

1 Abstract

Automatic seizure detection is a largely studied subject and shows promising results to ease the life of epileptic patients, in particular those for which drug treatments are not efficient. As of today, no complete implantable or wearable system is ready for clinical use to monitor and control seizure events. In this work, a user-oriented selection method based on three selection algorithms namely Least Absolute Shrinkage and Selection Operator (LASSO), Maximum Relevance Minimum Redundancy (MRMR), and Random Forest (RF) was developed to sustain the need for patient-specific solutions. It was validated on a large feature pool composed of time domain, frequency domain, time-frequency domain, and non-linear features with a threshold classifier. The results were assessed with the short-term iEEG recordings of the SWEC-ETHZ iEEG Database of 16 patients. The detection algorithm demonstrated good results with up to 100% accuracy and latency below 10s, comparable to other existing seizure detection algorithms. A selection method, developed to assist the neurologist in the selection of an optimal subset of features for a patient, was shown to be helpful and ready to support future improvements. Additionally, an architecture for hardware implementation was proposed on a Xilinx Virtex-7 FPGA (VC707).

Keywords - Epilepsy, Feature Ranking, Selection Method, FPGA, Feature extraction, Mutual Information, Lasso, Random Forest

2 Acknowledgement

I would like to express my special thanks to Professor Alexandre Schmid for his continuous support and advises during the realization of my Master's Thesis.

3 Introduction

Nowadays about 50 million people worldwide suffer from epilepsy, a chronic neurological disorder. A third of them have drug-resistant epilepsy and efforts are made to find alternative treatments to ease their life. These unexpected seizures may lead to physical injury and generate constant anxiety for the patient and their relatives. Several alternatives to medication exist such as palliative procedures or surgical intervention which are heavy treatments without a guarantee of a successful outcome. Alternatively, automated real-time seizure detection systems based on EEG signals have emerged over the past years, offering the opportunity to provide immediate treatment, for example, in the form of electrical stimulation [1] to stop the seizure. In the future, the hope is to prevent the seizures proactively.

Seizure detection based on EEG signals requires, in the first place, recording through electrodes the electrical activity of the brain. Data set of EEG signals, usually labeled between ictal and inter-ictal periods by well-trained specialists, are now available to train algorithms to recognize seizure event. Actually, each patient will show different EEG characteristics and the available amount of data per patient are usually very limited. It is indeed very time-consuming to record the EEG signals and requires hours of work by highly qualified physicians to label the recordings. To perform classification of the recorded signals, one should first extract specific features that characterize seizure events more accurately and more specifically than a raw signal. Possibilities are numerous but usual features are basically divided into four different domains: time domain, frequency domain, time-frequency domain, and non-linear features [2]. Although the available features are well-documented, no exhaustive and reduced list of the ideal extracted features for seizure detection exists. In fact, choosing the best feature set is a very problematic issue and many works mainly focus on that matter[2] [3]. This selection is critical because it will affect the detection accuracy, reactivity, and power consumption of the system. The solution is usually not trivial and a trade-off will be required. Indeed, a minimal latency may lead to an increased false alarm rate while large subsets of features will lead to increased power consumption. Extracted features are then fed into a classifier. The classifier is able to sort the EEG signals to detect seizure events after being trained with labeled data samples. It allows for evaluating the performance of the features to detect seizures. Ultimately, the combination of an EEG signal acquisition Unit (Electrodes, Filters, ADC, ...), a feature extractor, and a classifier constitute the foundation of any detection system that should generate an alarm when a seizure event is detected. Today, there are no automated closed-loop systems ready for clinical purposes and that is why this problem requires further research.

In this master's thesis, the goal was to, first, develop software as a selection method to optimize seizure detection and offer patient-specific solutions. This task falls within the global project of designing a complete closed-loop system [4] that would detect seizure events with minimal latency and set an alarm that is able to generate electrical stimulation feedback to stop the seizure. In short, the goal is to minimize detection delay while keeping a very low false alarm rate. The second task of this project is to implement on FPGA some scenarios of seizure detection developed during the first part to validate the results and offer an applied hardware architecture. This implementation covers feature extraction and classification with some of the feature subsets selected with the software feature selection method.

This report is organized according to the following structure: The first introductory part of the global project is covered in Sections 4 to 6. In Section 4, related works on seizure detection are discussed and some relevant values of figure of merits (FOM) are mentioned.

Note that FOM such as Accuracy, Sensitivity, Specificity, and False Alarm Rate are defined in the subsection 6.6 for those who would not be familiar with the topic. In Section 5, features that can be extracted from EEG signals are documented and finally in Section 6 the context of this project is specified with the relevant FOM that will allow a good analysis of the results and technical details about the data set are given. In the second part, Section 7 and 8 describe the seizure detection algorithm developed in this work with application examples. Some theoretical elements concerning the three chosen feature selection methods are given in the Subsection 7.3. Then the user-oriented selection method is defined and analyzed to end the software implementation. The Section 8 is dedicated to the hardware implementation of the detection device.

4 State-of-the-art

The first descriptions of epileptic seizures date back to Antiquity and scientists were already trying to identify the symptoms of this disease. Over the centuries, the understanding of this disorder slowly grew. At the end of the 19th century, with the introduction of Electroencephalogram (EEG), knowledge about epilepsy made a huge advancement [5]. EEG was a key element for the development of classification and the will to replace or help neurologists with automatic seizure detection systems. The automatic seizure detection issue is now very popular and the available papers on the subject are numerous. Still, this topic can be divided into a non-exhaustive list of the main covered thematic: Signal pre-processing, Feature extraction, Feature Selection, Classification, and Hardware Implementation. It is important to note that although the final goal is usually to implement a portable or implantable device, most of the papers do not propose the hardware implementation of the established algorithmic work.
For example, Jiang and Zhao [2] constructed a feature pool of 24 features commonly used that cover the four feature domains[1]. They used a Maximum Relevance Minimum Redundancy (MRMR) algorithm based on Mutual Information (MI) to rank the features and generate feature subsets specific to each patient. Optimal subsets were selected by evaluation of the performances through an Support Vector Machine (SVM) classifier. In that way, they were able to propose seizure detection with high accuracy[2] (up to 98.33% accuracy with only one feature) with only a few personalized features per patient. Their results were assessed with the UBonn and the CHB-MIT[3] datasets. It is worth noting that in their work, features such as the Lyapunov exponent and the average power of the α band showed very good ranking results and happened to lead to the best accuracy when used alone for seizure detection with some patients. Finally, they demonstrated that MRMR offered a better ranking solution (with smaller subsets) than commonly used selection methods, namely, Correlation Feature Selection (CFS), RelieF, and f_classif.
On the other hand, Peng et al. [6] demonstrated a two-step feature selection method based on Linear Discriminant Analysis (LDA) and Least Absolute Shrinkage and Selection Operator (LASSO). LDA helped to detect the most informative channels while LASSO allowed to build the subsets step by step. The initial feature pool was restrained to normalized Power Spectral Density (PSD) from six different subbands thus this paper does not demonstrate the developed selection method on a more broad range of features type. They used the CHB-MIT dataset that they segmented in 10s windows and an SVM classifier.
Although the previously summarized research papers show the desire to offer personalized feature selection for epileptic patients, no wearable platform was proposed and the work was only developed on software from now on. Wang et al. [3] also provided a two-step selection approach, implemented off-line, with MI for channel selection and Random Forest (RF) for feature selection. The average false alarm rate provided by their method is about 8.5 daily false detection with a mean latency of 6s. Sensitivity for the seizure onset was 98.4% while sensitivity by sample reached 74.2%. Additionally, they implemented, by the mean of High Level Synthesis (HLS), on a Xilinx FPGA the signal processing based on FFT to extract spectral energy and a SVM classifier. EEG signals were treated as half-overlapped windows of 4s. They proposed two different hardware implementations that fulfilled real-time requirements with an execution latency below 0.5 ms.
Burrelo et al. [7] proposed a very well documented work using the SWEC-ETHZ iEEG

[1] Time domain, Frequency Domain, Time-frequency Domain, and Non-linear features
[2] In their case, accuracy is defined per window, thus this value is relatively high even if lower than what is achieved in this work
[3] This dataset is based on scalp EEG signals

Database [8] that will also be used in this work. For that reason, their work is good basis for comparison purposes. Only three features were extracted: Line Length[4], Mean Amplitude and Local Binary Pattern (LBP). These features were then fed into various classifiers using HD computing. Results are very promising and offer reduced latency and the possibility for specific hardware implementation. Over the 16 patients that compose the database, an average latency of 8.81s was obtained while achieving a specificity of 97.31% and a sensitivity of 96.38%.
Wijesinghe et al. [9] depicted a hardware implementation of feature extraction for EEG signals on a Virtex 7 FPGA that is similar to the one implemented in this work. A control unit allows to activate the extraction of features selected by the user. The EEG signals are stored in 24-bit data words. Multiple features, such as DWT, PSD, Band energies, or zero crossing histogram are extracted simultaneously for each channel while the 14 channels are fed serially to the extractors. All channels can be processed[5] in less than one window duration (2s) so that the real-time condition is fulfilled. They achieved an overall specificity of 99.06% and sensitivity per sample of 88.43% with the dataset of Bonn University.
To sum up, these automated closed-loop systems are still in the development phase and suffer from many limitations when dealing with hardware implementation. These kinds of portable or implantable devices undergo some hardware constraints such as computational complexity, power consumption, or the availability of the resources and require a lot of attention.
Based on the study of these research papers this Master's Thesis was organized and defined as follows. The project is divided into two main parts: a software implementation in MATLAB and a hardware implementation on FPGA. In the first part of this work, a selection methodology was developed based on three different selection algorithms, namely Maximum Relevance Minimum Redundancy (MRMR) (based on Mutual Information), Least Absolute Shrinkage and Selection Operator (LASSO), and Random Forest (RF) to rank the features. Note that no channel selection was applied and that channels were simply averaged for the sake of simplicity. In the future, it would be appreciated to also dig in that direction. Indeed, the number of channels can be very large (up to 100 in the database) thus processing each channel is not an option. Some channels may also contain noise artifacts that could degrade the detection while some channels may be redundant [3]. A large feature set is extracted in MATLAB (offline) and fed to the different selection algorithms to generate three independent rankings. Then, the detection results based on a simple threshold classification are organized in EXCEL to expose the results of the collected data and highlight the feature subsets that would fulfill the targeted specifications. The goal is to provide the material for simpler trade-off analysis of the optimal generated subsets and assist the neurologist in her/his decision. Although it is not to be proven that classification also plays an important role in the detection performances, it is out of the scope of this project and the main work will be directed towards the feature selection. The novelty of this method is the direct implication of the specialist in the subset choice to provide personalized seizure detection for each patient. The goal of the second part of this project is to implement and validate on a Xilinx FPGA some scenarios that the neurologist could select with this method.

[4]Equivalent to Coastline
[5]The sampling frequency is 128 Hz and the FPGA runs at 50Mhz

5 Feature Pool

In this chapter, a non-exhaustive list of features that can be extracted from the EEG signals is proposed. A reduced feature pool (Section 7.2), that does not contain all the listed features here below, was fed to the selection method to ease the analysis and the exposition of the results. The goal was also to limit the number of features to implement on hardware for the second part of the work.

5.1 Time Domain Features

Time domain features are usually simple and based on statistics. For that reason, they are well suited to hardware implementation because they can offer a low computational cost.

5.1.1 Mean

Even if it usually does not perform well for seizure detection when applied to the raw signal, the mean value is an unavoidable statistical parameter in signal processing and it will be kept for analysis purposes. It is a simple parameter to extract and should be kept in the feature pool to see how selection methods may or may not discard this feature.

$$\mu = \frac{1}{T}\sum_{i=1}^{T} x_i \tag{1}$$

The mean value[6] of the signal is part of the feature pool of the paper of Jiang and Zhao [2] but was always poorly ranked by the MRMR ranking method.

5.1.2 Root Mean Square (RMS)

$$RMS = \sqrt{\frac{1}{T}\sum_{i=1}^{T} x_i^2} \tag{2}$$

The RMS was used by Frances-Villora et al. [10] and Abbaszadeh, Behrooz et al.[11].

5.1.3 Maximum and Minimum

The maximum and the minimum value of the signal over an epoch are easy parameters to extract that can be very useful for some particular patients. It was used by [2]. These features are often not relevant in the case of the raw signal, however on the DWT coefficients there are very promising for certain ranges of frequency.

5.1.4 Hjorth Parameters [11]

Standard deviation and the three Hjorth parameters are similar possible features to extract. It was chosen to keep only the variance in the reduced feature subset, also referred to as the 1st Hjorth parameter (Activity), for the sake of simplicity and relevancy in the ranking analysis [12].

Variance/Activity 1st Hjorth Parameter)

$$\sigma^2 = \frac{1}{T-1}\sum_{i=1}^{T}(x_i - \mu^2) \tag{3}$$

[6] in absolute value

Mobility (2nd Hjorth Parameter)

$$Mob = \frac{\sigma'}{\sigma} \tag{4}$$

Complexity (3rd Hjorth Parameter)

$$Compl = \frac{\sigma''\sigma'}{\sigma'\sigma} \tag{5}$$

5.1.5 Skewness

Skewness was selected in the reduced feature subset proposed by Frances-Villora et al. [10]. It is the third statistical moment and provides information on the asymmetry of the probability function of the signal around its mean value. [11]

$$Skwe(X) = E((\frac{X-\mu}{\sigma})^3) \tag{6}$$

5.1.6 Kurtosis

Kurtosis is also a statistical feature that is one order higher than Skewness. It shows the flatness of the probability distribution. [11]

$$Kurt(X) = E((\frac{X-\mu}{\sigma})^4) \tag{7}$$

5.1.7 Coastline/ Line Length

Coastline[7] has proved itself to be a very efficient feature to characterize EEG segments (especially with threshold methods) [12] while being easy to extract. It is equivalent to the Line Length by a factor 1/T. In this work, the following definition will be used:

$$Coastline = \sum_{i=2}^{T} |x_i - x_{i-1}| \tag{8}$$

This parameter is very suitable for real-time extraction since it only requires the last two samples.

5.1.8 Interquartile Range (IQR)

The Interquartile Range (IQR) is a measure of the spread of the signal data. It corresponds to the range of values containing half of the data located in the middle as illustrated in Fig. 1[8]. In other words, IQR is defined as the difference between the upper quartile Q3 and the lower quartile Q1. [11][13]

[7] Also called Total Variation

[8] http://sesp.esep.pro/fr/pages_stat-theorie/html_images/envimage6.html

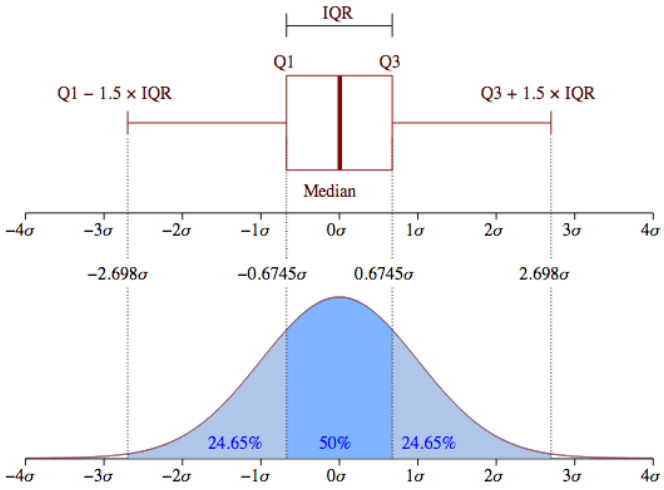

Figure 1: Graphical representation of the IQR for a normal distribution

5.1.9 Zero Crossing Histogram

The zero crossing histogram is the count of the number of times the signal crosses the x-axis during a time window.

5.2 Frequency Domain Features: FFT

Although time domain features offer low complexity extraction and good detection results with a certain type of patients, it is not sufficient. For that reason, it is necessary to include frequency analysis to extract more particularities of EEG signals. EEG signals are known to show five or six specific frequency sub-bands in epilepsy [2][6]. For each band, it is then interesting to extract the average power that corresponds to the integral over the frequency range of the PSD. In this work, the six sub-bands shown in Fig. 1 were considered [10]. [9]

Band Name	δ	θ	α	β	γ_{low}	γ_{high}
Frequency range [Hz]	0.5 - 4	4 - 8	8 - 12	12 - 30	30 - 47	47 - 120

Table 1: Frequency subbands of EEG signal

A commonly used extraction method for frequency components is Fast Fourier Transform (FFT). Other methods that will not be further detailed in this work such as Discrete Cosine Transform, Auto-regression (AR), or other PSD estimation methods also exist [2]. FFT is a strong tool to perform Fourier Transform with reduced computation time and complexity and was largely applied in hardware implementations[3] [14]. The PSD can then be obtained with the square of the magnitude of the FFT. Frequency analysis provides valuable information on EEG signals but is not able to capture the relation between time and frequency.

[9] Frequency range definition may vary across other related works.

5.3 Time-Frequency Domain Features: DWT

EEG provides non-stationary signals which means that the frequency spectrum will vary over time. For that reason, Discrete Wavelet Transform (DWT) is a very useful tool that is able to extract coefficients that reflect the time-frequency relation of the signal. In this work, the Daubechies 4 is used as the mother wavelet which results in one residual signal a5 (0-3Hz), and five detail signals d1(50-100Hz), d2(50-25Hz), d3(25-12Hz), d4(12-6Hz) and d5(6-3Hz) which corresponds the five rhythms of the EEG signal. The previously defined Time Domain features (Section 5.1) can also be defined as statistical features. Some of these statistical features will be extracted for the DWT coefficients to obtain Time-Frequency Domain Features (See Section 7.2).

5.4 Non-Linear Features

5.4.1 Approximate Entropy

Approximate Entropy exposes the complexity of a signal by quantifying the regularity and unpredictability of its fluctuations. It has the advantage to not be affected by low level noise [15]. That is why it is well suited for EEG signal characterization. It can be computed with the two following expressions [16]:
The signal is divided in T-m vectors containing m consecutive sampling points such that $u(i) = [x_i x_{i+1} ... x_{i+m-1}]^T$. The self-similarity can then be calculated as such:

$$C_i^m(r) = \frac{1}{T-m+1} \sum_{j=0}^{T-m} \theta(r - ||u_i - u_j||_\infty) \tag{9}$$

with r the tolerance. Finally, the Approximate Entropy combines all the indices over the 1s window length:

$$ApEn(X, m, r) = \frac{1}{T-m+1} \sum_{i=0}^{T-m} log(C_i^m(r)) - \frac{1}{T-m} \sum_{i=0}^{N-m-1} log(C_i^{m+1}(r)) \tag{10}$$

where θ is the Heaviside step function. During a seizure, this feature shows a steep drop. It was implemented on MATLAB with the function *approximateEntropy* [10] with the default values m = 2 and r = 0.2 std(X)[11].

5.4.2 Higuchi Fractal Dimension

Fractal Dimension (FD) is a measure of the complexity of a pattern. To do so the pattern is measured at different scales. This problem can be imaged with the measure of the coastline: the more precision is used to take the length, the larger the result will be. The FD is calculated as follows with Higuchi algorithm [17]:

$$L_m(k) = \frac{\sum_{i=1}^{\frac{N-m}{k}} \frac{|x(m+ik) - x(m+(i-1)k|}{N-1}}{T-m} \tag{11}$$

$$FD = \sum_{m=1}^{k} \frac{ln(Lm(k))}{ln(1/k)} \tag{12}$$

with k = 5 [17] and m ranging from 1 to k.

[10] Copyright 2017-2018 The MathWorks, Inc.
[11] This value of the tolerance r was shown to be an optimum value for finite data [15]

5.4.3 Non-Linear Energy

The Non-Linear Energy can be defined as follows:

$$NE(X) = \sum_{i=2}^{T-1} (x_i^2 - x_{i+1}x_{i-1}) \tag{13}$$

It has been shown to be a very effective feature for some patients with our method and was shown to be a good parameter when applied to the raw signal by [16].

6 Definitions and Specifications

In this section, the scope of this project will be further specified. Characteristics of the iEEG dataset used to evaluate the performance of the selected features and the classification process will be given. The most important FOM are detailed to allow a better understanding of the assessment of the results.

6.1 Context

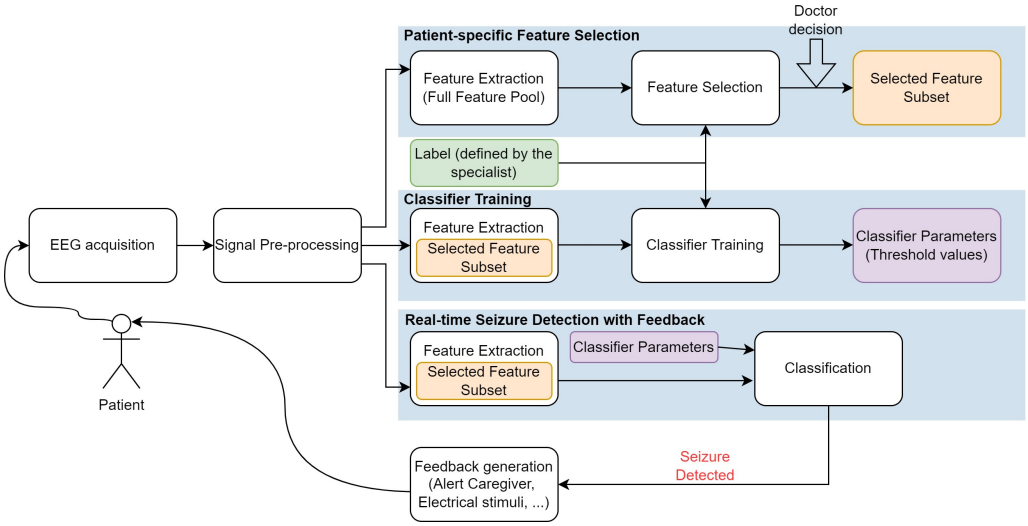

Figure 2: Implantable system device flow

The ultimate purpose of this work development is to design a complete closed-loop implantable system. The device should offer three main functions:

1. Patient-specific Feature Selection
2. Classifier Training
3. Real-time Seizure Detection with Feedback

A simplified flow for each of these functions is shown in Fig. 2. In this work, feature selection was performed offline only. In the future, the hope is to include this data analysis step directly on hardware so that this selection can be performed with the implant. This would offer the possibility for regular updates of the feature subset used for seizure detection without removing the implant. The idea is to have a set of features available for extraction (i.e Full Feature Pool) on the implant and to select only a few of them for seizure detection (i.e Selected Feature Subset). The neurologist has two interactions with the system. First, he/she should label the EEG recordings before the feature selection and the classifier training. Secondly, he/she has to select the optimized subset of features with the assistance of the selection method. In Fig. 3, the model of detection is illustrated. EEG recordings are labeled between two classes, inter-ictal and ictal. The goal of the system is to detect the seizure onset as soon as possible and provide feedback to end the seizure. It is thus assumed that the seizure will not last as illustrated by the red curve. This does not aim to perform a complete EEG recording characterization. That is why the sensitivity is defined per seizure event detected and not per window (see Section 6.6).

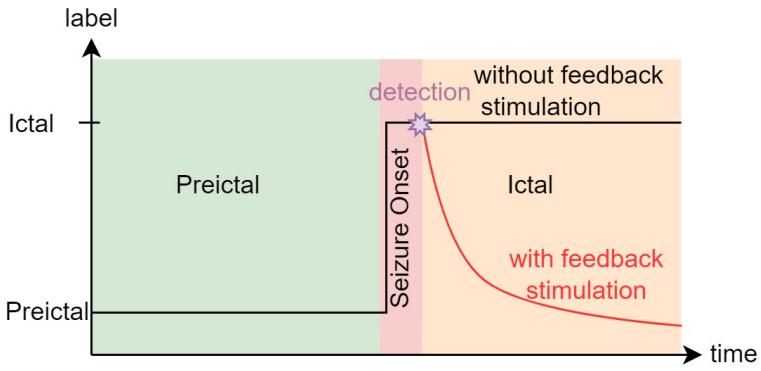

Figure 3: Detection model

6.2 Definitions

- N: number of observations
- M: number of channels/electrodes
- T: number of sampling points per window
- f_s: sampling frequency (512Hz)
- S: feature set, |S|: number of features
- TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative
- EEG: electroencephalography
- iEEG: Intracranial EEG

6.3 iEEG vs scalp EEG

EEG analysis is a strong tool for epilepsy seizure detection. The signals are obtained via several electrodes connected either to the scalp or even implanted inside brain tissues that will collect the signal from the brain. Scalp EEG is a non-invasive diagnostic tool that is often used to locate the source and type of seizures before using a more invasive monitoring method such as iEEG which involves depth electrode implantation in a surgery procedure. Today, iEEG signals have proved to be an important tool to push and complete studies in cognitive neuroscience [18]. In the context of this work, the final goal is to develop an implantable device thus the need to analyze iEEG database to assess the results.

6.4 SWEC-ETHZ iEEG Database

Herein the short-term iEEG of the SWEC-ETHZ iEEG Database [8] was used. The provided data were pre-processed digitally with a Butterworth filter (0.5-150Hz) and sampled at a rate of 512Hz. Data were collected for 16 different patients going through several epileptic crises. These patients are known to be drug-resistant thus the need for real-time seizure detection. Data are stored in a $T \times M$ matrix, with T the number of sampling points and M the number of channels, each provided by one electrode. Each seizure recording starts with 3min pre-ictal segments and continues with an ictal segment ranging from 10s to 1002s that is finally followed by a 3min post-ictal period. The number of recordings

(seizures) per patient is variable (ranging from 2 to 14). The number of recording electrodes per patient may also vary between 36 to 100. Table 7 summarizes the previously mentioned characteristics for each patient.

6.5 Segmentation

First, the data are restructured to be ready for feature extraction. Features will be extracted for each 1s window. This window length was chosen to provide enough sampling points per window to capture the EEG characteristics while keeping an acceptable latency. Further research could be carried out to optimize this parameter. These epochs are non-overlapping and each contain 512 data points. Half of the data of each patient is used for training purposes while the other half is used to test and validate the classification process. All channel outputs are reduced to a single channel output by a simple averaging. This is not ideal and preliminary channel selection could provide better seizure detection results. Indeed, some channels contain more information about epileptic episodes while some may even distort the signal characteristics with noise.

6.6 Figure of merits

To analyze the results obtained with the selected features and the threshold classifier it is important to extract the relevant figure of merits (FOM) that will help to assess the performance of the seizure detection. These FOM will be calculated during the software implementation as a first step and will then be compared with the obtained ones during the hardware implementation. These FOM are also very important to compare the results with the other related work. There are four main FOM that will be used for the comparisons: Sensitivity, Specificity, False Alarm Rate, and Latency. One can also mention accuracy that is easily deduced from the sensitivity and the accuracy. The definitions are as follows [7]:

Sensitivity

Sensitivity is a measure of the correctly detected seizure events. Note that this result is independent of the delay before detection. If one of the 1s epochs that is part of the ictal segment is labeled correctly then the seizure is considered as detected. Thus, the sensitivity will, most of the time, be 100%. The definition of this FOM is directly linked to the context in which this detection algorithm is used as explained in Section 6.1.

$$\frac{\sum_{i=1}^{N} Seiz_i == ictal}{N} \tag{14}$$

Specificity

Specificity is the ratio of correctly classified non-seizure 1s epochs. Specificity must be maximized for the comfort of the patient. Indeed, a low specificity induces a large false alarm rate which leads to possible anxiety for the patients.

$$\frac{TN}{FP + TN} \tag{15}$$

Macro averaging Accuracy

In this work, accuracy is defined in accordance with the number of detected seizures and the correctly labeled intra-ictal 1s window rather than to all the correctly labeled 1s epochs.

$$\frac{sensitivity + specificity}{2} \tag{16}$$

False Alarm Rate

The false alarm rate gives a good indication of the possible disturbance that the patient may experience when False Positive (FP) are recurring. This FOM is often not mentioned in scientific papers. As a matter of fact, it is very difficult to obtain good results with respect to the False Alarm Rate. Indeed, for a short-term database such as ours, the false alarm rate increases drastically for each wrongly labeled window as a seizure (FP)[12].

$$\frac{FP}{SimulationTime}[\#/hour] \tag{17}$$

As a reminder, with the CHB-MIT epilepsy database[13], Wang et al. [3] (MI and RF channel and feature selection + SVM classification) obtained a false alarm rate of only 8.5 per day with a latency of 6s at the cost of a sensitivity of 98.4%.

Seizure Onset Detection Latency

Latency represents the time between the starting point of the seizure defined by the expert and the time at which the classifier detects the seizure. Latency should be minimized while this may increase the false alarm rate. There is therefore a trade-off to be defined between reactivity and accuracy.

$$\frac{\sum_{i=1}^{N}(t_{detection} - t_{start})}{N} \tag{18}$$

[12] For one recording of 6 min without a seizure, a 1s window that is labeled as positive (FP) already lead to a False Alarm Rate of 240/day

[13] 969 Hours of scalp EEG recordings over 23 patients

7 Software Implementation

7.1 Feature extraction

Feature extraction was done in MATLAB. Details about the code[14] can be found in the Appendix A.

7.2 Reduced Feature Pool

The Reduced Feature Pool was constructed with a total of 8 Time domain features, 15 Frequency domain features, 3 Non-Linear features, and, 6x8 Time-Frequency features (See Table 11 in Appendix B). This reduction aimed to focus on an acceptable number of features to ease the analysis of the results provided by the selection method. The frequency domain features referred to as *PSD* correspond to the average power of the associated subband while the one referred to as *NP* contains the normalized band power of these latter with respect to the raw signal [0-120Hz] band power. *RPHFLF* is the ratio between the high (β and γ bands) and the low (δ,θ and α bands) frequency band power.

7.3 Feature ranking and selection

In this work, three different selection methods were used independently to obtain three different rankings of the features. One is a filter-based selection method based on Mutual Information and the two others are embedded type feature selectors.

7.3.1 Mutual Information - Maximum Relevance Minimum Redundancy

The Mutual Information (MI) describes the statistical degree of correlation of two random variables [2]. MI is a powerful tool to minimize redundancy when selecting features but can also be helpful to detect the most relevant features to detect a seizure. That is why MI has been frequently used as a selection method in various fields of studies and more specifically for feature [2] or channel [3] selection in the scope of seizure detection.
There are two ways to define MI mathematically. It can be first defined in terms of joint probability:

$$I(X,Y) = \sum_{x \in X, y \in Y} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{19}$$

where p(x.y) is the joint probability of x and y, and p(x) and p(y) are the respective marginal probability distribution of x and y [2]. MI can also be written in terms of entropy:

$$I(X,Y) = H(X) - H(X|Y) \tag{20}$$

$$= H(X) + H(Y) - H(X,Y) \tag{21}$$

where H(X) and H(Y) are the marginal entropy, H(X|Y) is the conditional entropy and H(x,Y) is the joint entropy of X and Y [2]. As a consequence, I equals the entropy of X when X and Y are the same random variable and equals zero when X and Y are independent [19].
MI is then a very useful mathematical tool to apply a Maximum Relevance Minimum Redundancy (MRMR) Algorithm. The feature-feature MI and the feature-class MI can be further used to apply this algorithm. The class corresponds to the possible patient state: intra-ictal or extra-ictal. Finally, the ranking can be constructed by choosing the features

[14] The complete code is available on request.

offering the best balance between a maximized feature-class MI (maximum relevancy) and a minimized feature-feature MI (minimum redundancy). In this work the Mutual Information Quotient (MIQ) value will be used to rank the features [19] such that:

$$MIQ_x = \frac{V_x}{W_x} \tag{22}$$

where the relevance V_x and the redundancy W_x are defined as follows:

$$V_x = I(X,Z)W_x = \frac{1}{|S|}\sum_{z\in S} I(X,Y) \tag{23}$$

X and Y are the random variables each corresponding to a feature during an epoch and Z is the response variable (ictal/non-ictal). |S| is the number of features in the feature set S. The MRMR ranking method is implemented in the function fscmrmr[15] that is available in the "Statistical and Machine Learning Toolbox" of MATLAB and was used to perform the MI ranking. It has the advantage to perform a binning (256 bins) [20] of the discrete variables when computing the MI values with the definition in Eq. (19).[16]

7.3.2 Least Absolute Shrinkage and Selection Operator (LASSO)

LASSO selection method is an embedded method and was chosen to deeper evaluate the possibilities that it can offer. Peng et al. [6] only applied it to PSD features and showed promising results. LASSO solves a minimization problem that can be described by the following formula:

$$\min_{\beta_0,\beta}\left(\frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - x_i^T\beta)^2 + \lambda\sum_{j=1}^{S}|\beta_j|\right) \tag{24}$$

with N the number of observations, y the label vector, and S the number of features fed into the selector. β_0 is a scalar and β is a vector of length |S| that contains the coefficients. The regularization term $\lambda\sum_{j=1}^{S}|\beta_j|$ will force the coefficients of the eliminated features to zero. The larger the regularization parameter λ the more zero coefficients will appear in the β vector. The nonzero coefficients indicate the features selected for the chosen λ. The limitation of this selection method is its incapacity to discard irrelevant features that may be highly correlated with relevant ones. In this case, LASSO may pick these features to compensate for the over-shrinkage of the nonzero coefficients. e.g if X_1 and X_2 are relevant, X_3 = a X_1 + b X_2 + ϵ would not be relevant even if the correlation is high [21].
In MATLAB, the function lasso(X,y)[17] returns a matrix B[18] containing 100 column-vector that each corresponds to one value of λ[19]. Each vector contains one regression coefficient per feature that will be zero when the feature has been eliminated by the selection method. The input arguments are X, a NxS matrix containing N observations for each of the |S| features. y is the vector containing the class label (seizure/non-seizure) for each observation. To establish a ranking, each feature removed by the algorithm was added to the ranking starting from the end.

[15] Copyright 2019 The MathWorks, Inc.
[16] Other available function implemented in Matlab usually encounter memory issues when using continuous variables.
[17] Copyright 2011-2019 The MathWorks, Inc.
[18] corresponds to β vector from equation 24
[19] Obtained by default by a geometric sequence.

7.3.3 Random Forest (RF)

Random Forest is an ensemble-based machine learning method. Often used as a classifier, it can also provide a ranking for a group of selectors. As its name states it will randomly pick features to grow decision trees but also chose a random part of the training set thanks to bootstrapping. These characteristics enable low correlation between the individual trees as well as low bias and high variance due to the depth of the tree.[22]
There is one main issue with RF, it is reproducibility. There is a different solution to obtain a reproducible selection in MATLAB. One is to force the algorithm to pick all the features to build the decision trees and the other is to set the seed of the random number generator by using *rng*[20]. In this work, it was chosen to use the first option so that all features would be taken into account by the RF algorithm. Note that it is not required to involve all the features in the selection to obtain satisfying results but this does not result in a systematic method and it would have been more difficult to analyze. A disadvantage of RF is that it is unable to eliminate redundant features. It may be more appropriate to use RF on a feature pool already reduced with methods such as MRMR that can remove redundant features.

7.4 Classification

Classification is usually performed with powerful machine learning tools such as SVM or RF. In this work, classification was not the center point of the research and for that reason, the classification will be performed with a simple threshold comparison. This decision will indeed result in lower seizure detection performance but it will ease the hardware implementation and reduce power consumption. In future work, it would be interesting to improve the classification to improve the detection accuracy and reduce the latency.

7.4.1 Threshold algorithm

To perform the classification on the training and the testing data set, it is required to obtain the threshold value for each feature and each patient. This step was done using the training data set and the Algorithm 1 from Wang et al. [23]. If the output range of a feature during the ictal state does not overlap with the range in the inter-ictal state, the threshold will simply be the median value between the minimum value of the higher range and the maximal value of the lower range (l.12-13). If this condition is not fulfilled then the algorithm detects all overlapping values to finally define the threshold as an average of the deviations of each range over the other one (l.15-23). In this way, the threshold is shifted higher if the low values (usually the non-ictal ones) tend to show random peaks of higher amplitude and inversely. Lines 6-8 were added to distinguish the features that show higher values during a seizure and the ones that show lower values.

7.4.2 Seizure Detection

After feature extraction, windows are labelled between ictal (logical '1') and inter-ictal (logical '0') classes. A seizure is detected when all the selected features are characterized as ictal by the threshold classifier during m consecutive windows as illustrated in Fig. 4. In this case $m = 2$ which results in a detection latency of 4s[21]. This decision was made to maximize specificity at all times to avoid a very large false alarm rate.

[20] Allows to control pseudo-random number generator in MATLAB
[21] Best case scenario offers a minimum achievable latency of 2s

Algorithm 1 Threshold algorithm applied to each feature computed with the training set. The label vector is given by the physician

Require: Extracted feature vector X, Label vector L

for i = 1 to N **do**
  if L(i) = Seizure **then**
    $MemSeizure \leftarrow X(i)$
  else
    $MemNonSeizure \leftarrow X(i)$
if Feature is higher during ictal state **then**
  $MemHigh = MemSeizure$
  $MemLow = MemNonSeizure$
else
  $MemHigh = MemNonSeizure$
  $MemLow = MemSeizure$
if min(MemHigh) > max(MemLow **then**
  $threshold \leftarrow \frac{min(MemHigh)+max(MemLow)}{2}$
else
  for each element j of MemLow **do**
    if MemLow(j) > min(MemHigh) **then**
      $SumLow \leftarrow SumLow + MemLow(j)$
      $NumLow \leftarrow NumLow + 1$
  for each element j of MemHigh **do**
    if MemHigh(j) < max(MemLow) **then**
      $SumHigh \leftarrow SumHigh + MemHigh(j)$
      $NumHigh \leftarrow NumHigh + 1$
  $threshold \leftarrow \frac{1}{2}(\frac{SumHigh}{NumHigh} + \frac{SumLow}{NumLow})$

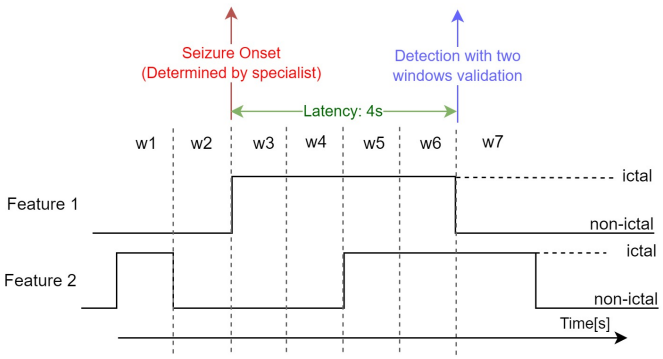

Figure 4: Consecutive windows validation with two features

7.5 Selection Method

When it came to analyzing and selecting manually the optimal feature subset for each patient according to the ranking provided by the three selection methods, the task appeared to be very complex. Although MRMR, LASSO, and, RF provide a satisfying ranking, they do not group the selected features in optimal subsets according to the classification method. Indeed, these methods operate completely independently from the threshold classifier. Thus, post-processing analysis is required.

First, an analysis of subsets generated by taking the k best features (independent for each selection method) was performed. This procedure allowed us to find some good subsets for some patients but the available solutions were very limited. In Table 2 an example of the results obtained for patients 1 and 2 is shown[22]. Each line corresponds to the FOM obtained with the feature subset composed of all the features above. For example, line *top2* of patient ID1 gives the results of the classifier for the subset {7,35}. Figure 5 shows the monotone increase of the specificity as a function of the feature subset size (for 100% sensitivity only) which is a direct result of the logical AND classification performed between all features of the subset (Fig. 4). The more constraints are added, the less FP there will be. As a result, latency can only increase with the size of the subset. Although

	Lasso				MRMR				RF			
ID1	top 10	Sensitivity	Specificity	Latency	top 10	Sensitivity	Specificity	Latency	top 10	Sensitivity	Specificity	Latency
top 1	7	100.0%	88.3%	2.86	7	100.0%	88.3%	2.86	38	100.0%	85.6%	3.57
top 2	35	100.0%	89.9%	2.86	1	100.0%	94.3%	4.14	7	100.0%	90.1%	3.86
top 3	3	85.7%	92.1%	7.17	52	100.0%	96.8%	4.86	27	100.0%	91.6%	7.00
top 4	18	71.4%	96.8%	18.60	60	100.0%	98.5%	9.29	54	100.0%	92.9%	7.14
top 5	50	57.1%	97.0%	18.25	28	71.4%	99.2%	9.80	43	100.0%	93.3%	7.14
top 6	49	42.9%	98.3%	17.67	36	71.4%	99.5%	9.80	51	100.0%	93.9%	7.14
top 7	36	42.9%	99.0%	25.33	65	42.9%	99.8%	5.33	16	71.4%	96.9%	13.20
top 8	30	42.9%	99.0%	25.33	57	14.3%	99.8%	28.00	22	0.0%	0.0%	0.00
top 9	72	42.9%	99.2%	40.33	44	0.0%	0.0%	0.00	11	0.0%	0.0%	0.00
top 10	55	42.9%	99.2%	40.33	49	0.0%	0.0%	0.00	37	0.0%	0.0%	0.00
ID2												
top 1	53	100.0%	70.4%	1.50	7	100.0%	59.7%	2.00	27	100.0%	75.4%	2.50
top 2	45	100.0%	81.8%	3.50	60	100.0%	75.6%	2.00	56	100.0%	86.1%	2.50
top 3	46	100.0%	89.3%	3.50	52	100.0%	86.3%	3.50	53	100.0%	86.1%	2.50
top 4	54	100.0%	92.1%	3.50	36	100.0%	91.7%	5.50	7	100.0%	86.1%	2.50
top 5	3	100.0%	95.1%	3.50	44	100.0%	95.7%	5.50	16	100.0%	90.8%	5.00
top 6	55	100.0%	96.4%	3.50	24	100.0%	98.2%	41.50	48	100.0%	92.6%	5.00
top 7	42	100.0%	96.4%	3.50	28	100.0%	98.8%	47.00	58	100.0%	92.9%	5.00
top 8	19	100.0%	99.2%	5.00	68	100.0%	99.6%	47.00	22	100.0%	99.4%	45.50
top 9	47	100.0%	99.4%	9.00	57	50.0%	99.7%	17.00	60	100.0%	99.7%	45.50
top 10	16	50.0%	99.7%	7.00	65	0.0%	0.0%	0.00	10	100.0%	100.0%	45.50

Table 2: First analysis of subsets generated with the top 10 features of each ranking method for patient 1 and 2. Each line (e.g top 6) corresponds to the results of one subset composed of the features above (e.g top 1 to top 6). A single window validation was used for classification.

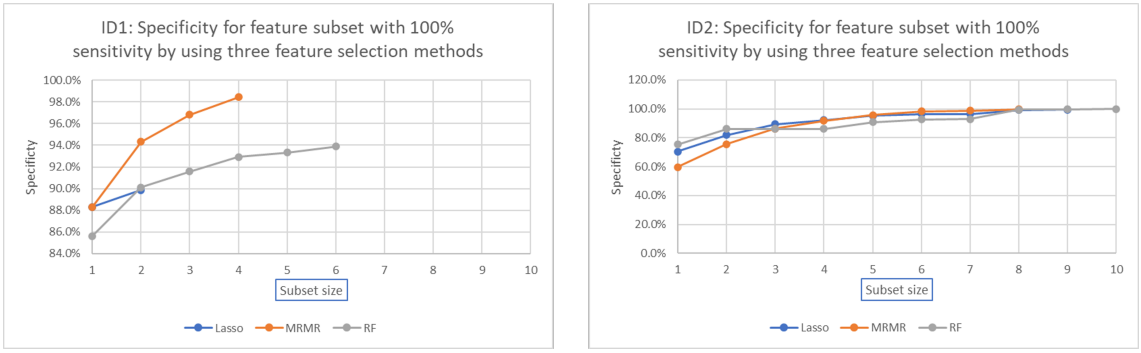

Figure 5: Specificity as a function of the feature subset size of the results from Table 2 showing 100% sensitivity

some acceptable results were obtained with some patients, the subset choice was limited and was lacking flexibility. A deeper analysis was therefore required. For that reason, it was decided to generate more subsets based on the ranking outcome[23]. By choosing a

[22] Note that feature numbering does not correspond to the final reduced feature set of Table 7.2

[23] As well as introducing consecutive windows validation to improve specificity

depth of analysis n corresponding to the n best-ranked features by a selection algorithm it was possible to create N subsets by generating all combinations of features of size 1 to n (combinations of k elements in a given array of size n (Eq. 25)).

$$N = \sum_{k=1}^{n} C_n^k \tag{25}$$

For example, if n = 3, there would be seven generated subsets based on the three top-ranked features: {1},{2},{3},{1,2},{1,3},{2,3},{1,2,3}.
There are now more subsets to analyze and still no systematic method to make the decision. The center point of this problem resides in the fact that all performances depend on each other and that, for example, improving the specificity may as well increase the latency. For example, in Table 2, one can see that for patient 2 with the LASSO method, it is difficult to determine the optimal subset. Indeed, including feature 47 (*top*9) in the subset would increase the specificity but the latency would be degraded and reach 9s. In this case, how can we, engineers, decide whether a reduction in four seconds of the latency should be preferred to ten more false alarms in a day? There is indeed a trade-off to consider and the neurologist will be more qualified than engineers for that task. For these reasons, a selection method that includes the neurologist in the decision-making was developed. The goal is to expose to the neurologist all the information required to choose a subset of features to perform seizure detection for a specific patient.

7.5.1 Implementation and Method flow

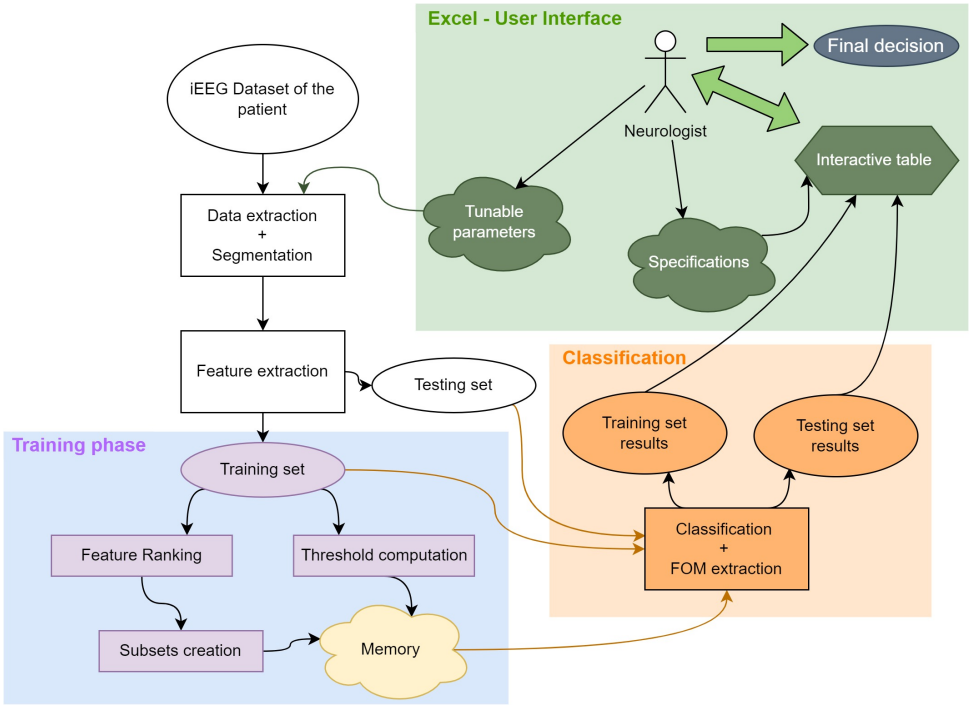

Figure 6: Method flow and interactions: All the signals processing and results calculations are done with MATLAB. Excel is here to offer a synthesized overview to the user with the possibility to tune specifications and filters.

Two software tools were used namely MATLAB and Excel. MATLAB is used to extract the features from the EEG signal, rank the features, train the classifier, generate the subsets

	Subset										Training set: Choice				Testing set: validation				Relative error				
ID1	1	2	3	4	5	6	7	8	9	10	Sensitivity	Specificity	Latency[s]	False Alarm Rate [#/day]	Sensitivity	Specificity	Latency[s]	False Alarm Rate [#/day]	Sensitivity	Specificity	Latency[s]	False Alarm Rate[#/day]	Result
Lasso	31	50									100.00%	99.21%	6.67	522.3	100.00%	98.49%	17.57	1175.9	0.00%	0.73%	163.57%	125.13%	FAILURE
Lasso	50										100.00%	97.73%	6.67	1505.5	100.00%	97.14%	7.86	2228.1	0.00%	0.60%	17.86%	47.99%	SUCCESS
RF	15	32									83.33%	99.68%	6.80	215.1	100.00%	98.77%	15.43	959.3	20.00%	0.91%	126.89%	346.03%	
Lasso	6	32	50								100.00%	99.44%	6.83	368.7	100.00%	98.61%	17.57	1083.1	0.00%	0.84%	157.14%	193.76%	FAILURE
Lasso	6	32	31	50							100.00%	99.44%	6.83	368.7	100.00%	98.61%	17.57	1083.1	0.00%	0.84%	157.14%	193.76%	FAILURE
Lasso	32	31	50								100.00%	99.26%	6.83	491.6	100.00%	98.57%	17.57	1114.0	0.00%	0.69%	157.14%	126.61%	FAILURE
Lasso	32	50									100.00%	99.21%	6.83	522.3	100.00%	98.49%	17.57	1175.9	0.00%	0.73%	157.14%	125.13%	FAILURE
RF	15	32	39								83.33%	99.81%	7.00	122.9	100.00%	98.89%	15.71	866.5	20.00%	0.93%	124.49%	605.01%	
RF	15	39									100.00%	99.81%	8.50	122.9	100.00%	98.69%	15.71	1021.2	0.00%	1.13%	84.87%	730.91%	FAILURE
Lasso	6	45	31	50							100.00%	99.54%	20.67	307.3	100.00%	98.57%	17.57	1114.0	0.00%	0.97%	14.98%	262.58%	

Table 3: Example of the content of the EXCEL results for patient 1

and finally classify the training and the testing set. Excel is used as a tool to expose the results of the training to the user (the neurologist) to help him/her establish his/her subset choices by tuning the specifications. It also displays the testing results for one final analysis. The summarized method flow and interactions between the different blocks are shown in Fig. 6. In the green box, one can observe the interactions between the neurologist and the software. Some tunable parameters such as the depth of analysis and the number of consecutive windows for validation of the seizure onset are given before the MATLAB analysis. Then the analysis can be launched in MATLAB. The MATLAB part is split into two phases: a 1^{st} phase of analysis of the features (Blue box in Fig. 6) using the training set (50% of the database) of the patient, a 2^{nd} phase of classification (Orange box) using both the training and the testing set (the other half of the database) where all the FOM from Section 6.6 are extracted. In the Excel file (see Table 3), the user can find all the subset combinations, up to the specified depth of analysis, and the corresponding FOM related to the classification. In the EXCEL sheet *Specifications*, the user can specify the constraints that the subset should meet to be validated. The available specifications are as follows:

- Minimum Sensitivity [%]
- Minimum Specificity [%]
- Maximum Latency [s]
- Maximum False Alarm Rate [#/day]

Note that this list is not exhaustive and could be enlarged with more specifications to match the requirements expressed by medical professionals (See section 7.7.4. It is then possible to display only the valid subsets by filtering the last column *Results* as shown in Table 15 in Appendix C. The column is either filled by *SUCCESS* when both training and testing set fulfill the specifications, *FAILURE* when testing was not successful, and is blank when the subset was not selected after training. It is then possible to apply filters with respect to the different columns so that only the subsets of interest are displayed. It is also possible to sort the subsets by descending, ascending values, or specific values of a parameter. A case study is detailed in Section 7.6 for a better understanding of the method. Two scenarios of usage of the method are described in the following paragraphs.

7.5.2 Scenario 1: User-oriented Validation

In this case, the neurologist has access to both the training and the testing classification results. In this way, the neurologist can understand if the proposed subsets with the training set are reliable when the outcome matches with the testing set ones. A considerable inconstancy between the two sets' results could show a lack of training data or a deficient feature pool. The neurologist could then determine whether more data are required to analyze the patient or whether the features proposed are not sufficient and cannot characterize

the seizure of this patient. On the other hand, if specifications a met, the neurologist can select a subset to be implemented in the implantable device for real-time seizure detection.

7.5.3 Scenario 2: Theoretical Validation

The testing set results are assumed unknown by the neurologist during the decision-making. The purpose of these validations FOM is to extract an interval of confidence that the training choices offer on the FOM. In this way, the error calculation between training and testing classifications would be a good indicator of the success of the selection method. In that way, the testing set provides a validation of the success rate of the training and selection phase. Although this second scenario would be an interesting continuation for a deeper analysis of the selection methods, the case study in Section 7.6 will be restrained to Scenario 1.

7.6 Case Study

To illustrate the decision flow that the neurologist could follow to pick a feature subset, an example with patient ID6 is detailed step by step. A total of four[24] seizure recordings are provided.

Step 1: Matlab Analysis

Step 2: Specifications

Step 3: Filters

Step 4: Final decision

Step 1 A MATLAB analysis of depth 10 with three consecutive windows validation is performed. This will generate all the possible subsets for each selection method based on the 10 best-ranked features and compute their performances. In this case, there will be 3×1023 (Eq. 25) lines of results.

Step 2 In an Excel sheet dedicated to the specifications, the neurologist can enter the minimum specificity and sensitivity to reach and the maximum acceptable latency. In this example, the following parameters were defined:

Min. Sensitivity	Min. Specificity	Max. Latency [s]	Max. False Alarm Rate [#/day]
1	0.95	15	N/A

Table 4: Specifications for Case Study of Patient 6

Step 3 In this case, there are 69 subsets that fulfill the requirements during training and that were validated during testing as well as shown in Table 15 in the Appendix C. 26 subsets were generated by LASSO, 2 by MRMR, and 41 by RF.[25]. Note that in Table 15 some columns were removed to ease the reading and comprehension of the information[26]. From this first table, it can be already concluded that Patient 6 offers many

[24] Two are used for selection/training and two for testing.

[25] Some subsets may appear twice or three times if selected by different selection methods

[26] These are specified in the notes of Table 15 in the Appendix C

subsets with good specificity results, that often reach 100% and that seizure detection is not problematic. It can be added that 50 subsets that satisfied the specifications during training were discarded due to the testing results that most of the time showed non-acceptable latency (except for two subsets with specificity lower than 95%). However, it is important to note that the number of analyzed seizures is low. On one hand, it implies that only a small amount of recordings (two seizures) is available to determine the threshold for the classification but still, the training seems to be performing well, but on the other hand, the testing set is also small and the validation may not cover all the types of seizure that the patient may experience. Additionally, it can be noted that the feature subsets size is strictly smaller than six which is easily explained by the fact that the classification method is quite strict. Indeed, it requires all the selected features to detect the seizure during the three same consecutive windows to set the alarm. The neurologist can now

	Subset					Training set: Choice			Testing set: validation			Relative error			
ID6	1	2	3	4	5	Specificity	Latency[s]	FalseAlarm Rate[#/day]	Specificity	Latency[s]	FalseAlarm Rate[#/day]	Sensitivity	Specificity	Latency[s]	FalseAlarm Rate[#/day]
Lasso	18					99.31%	8.50	437.2	96.81%	9.00	1920.0	0.00%	2.52%	5.88%	339.11%
RF	18					99.31%	8.50	437.2	96.81%	9.00	1920.0	0.00%	2.52%	5.88%	339.11%
RF	53					98.47%	8.50	961.9	99.44%	11.50	333.9	0.00%	0.99%	35.29%	65.29%
RF	19	18				99.86%	9.00	87.4	99.03%	9.00	584.3	0.00%	0.83%	0.00%	568.21%
Lasso	18	15				100.00%	9.50	0.0	98.47%	9.00	918.3	0.00%	1.53%	5.26%	-
RF	53	18				100.00%	9.50	0.0	100.00%	11.50	0.0	0.00%	0.00%	21.05%	-
Lasso	60					99.58%	9.50	262.3	99.86%	12.00	83.5	0.00%	0.28%	26.32%	68.18%
Lasso	41					99.44%	10.50	349.8	99.72%	9.50	167.0	0.00%	0.28%	9.52%	52.27%
RF	53	6				100.00%	11.00	0.0	100.00%	11.50	0.0	0.00%	0.00%	4.55%	-
RF	6					99.58%	11.00	262.3	99.86%	11.50	83.5	0.00%	0.28%	4.55%	68.18%
Lasso	55					99.86%	12.50	87.4	99.72%	12.50	167.0	0.00%	0.14%	0.00%	90.92%
Lasso	55	41				99.86%	12.50	87.4	100.00%	12.50	0.0	0.00%	0.14%	0.00%	100.00%
Lasso	41	18				100.00%	13.00	0.0	99.86%	11.00	83.5	0.00%	0.14%	15.38%	-
Lasso	41	15				100.00%	13.00	0.0	100.00%	11.00	0.0	0.00%	0.00%	15.38%	-
Lasso	41	17				100.00%	13.00	0.0	100.00%	12.00	0.0	0.00%	0.00%	7.69%	-

Table 5: Best 15 subsets w.r.t specificity that fulfill specifications given in Table 4 for patient 6

proceed to a further reduction of the number of subsets. For example, it is possible to sort by specificity. As said before, in this case, it is not relevant because specificity is very high in all the cases. A second option would be to sort by latency in the training phase (from smaller to larger). The best 15 subsets are shown in Table 4. The normalized PSD of the beta band[27] (18) shows the best latency results, however, it leads to a loss in specificity both during training and testing and more than 3000 false alarm per day. For that reason, this feature may be discarded by the neurologist since it is not acceptable for the patient comfort.

Step 4 Predicting what will be the choice of the neurologist at this point is difficult. Indeed, the trade-off between the different subsets is now very arbitrary and may be led by other parameters. Let's assume the power consumption should be minimized. In this case, the subset size can be limited to a single feature. There are 11 features that can be chosen (Table 6). The only time domain feature is Coastline (6). There are four frequency domain features selected and the others are based on DWT. In this work, it was speculated that the final choice would go towards the coastline because it has the most direct computation model since it does not require pre-processing such as FFT or DWT and still performs well for patient 6.

[27] selected by RF and LASSO

ID6 Feature	Specificity	Latency[s]	FalseAlarm Rate[#/day]	Specificity	Latency[s]	FalseAlarm Rate[#/day]	Sensitivity	Specificity	Latency[s]	FalseAlarm Rate[#/day]
18	99.31%	8.50	437.2	96.81%	9.00	1920.0	0.00%	2.52%	5.88%	339.11%
53	98.47%	8.50	961.9	99.44%	11.50	333.9	0.00%	0.99%	35.29%	65.29%
60	99.58%	9.50	262.3	99.86%	12.00	83.5	0.00%	0.28%	26.32%	68.18%
41	99.44%	10.50	349.8	99.72%	9.50	167.0	0.00%	0.28%	9.52%	52.27%
6	99.58%	11.00	262.3	99.86%	11.50	83.5	0.00%	0.28%	4.55%	68.18%
55	99.86%	12.50	87.4	99.72%	12.50	167.0	0.00%	0.14%	0.00%	90.92%
12	100.00%	13.00	0.0	100.00%	10.00	0.0	0.00%	0.00%	23.08%	-
17	99.58%	13.00	262.3	96.39%	12.00	2170.4	0.00%	3.21%	7.69%	727.31%
13	100.00%	13.50	0.0	100.00%	14.00	0.0	0.00%	0.00%	3.70%	-
66	99.03%	13.50	612.1	99.72%	10.00	167.0	0.00%	0.70%	25.93%	72.73%
52	99.86%	14.50	87.4	100.00%	11.50	0.0	0.00%	0.14%	20.69%	100.00%

First Column Legend: Time domain | Frequency domain

Table 6: Single feature that fulfill specifications for patient 6

7.7 Discussions

In this Section, the global results offered by the combination of the Reduced Feature Pool (Table 11), the Threshold Classifier (Algorithm 1), and the Selection Method (Fig. 6) are discussed.

7.7.1 Patients

		ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8	ID9	ID10	ID11	ID12	ID13	ID14	ID15	ID16
# seizures		13	4	2	14	10	4	2	2	9	5	2	10	7	7	3	6
# channels		47	42	98	62	54	64	36	59	56	100	64	49	92	74	61	92
Seizure duration [s]	Mean	71	223	99	98	99	146	15	57	144	14	124	45	79	587	121	79
	Min	10	96	73	31	80	89	14	52	104	10	60	23	19	154	52	19
	Max	252	301	125	139	154	179	16	61	198	22	192	93	100	1002	184	100
Patient labelling of Burrello et al.		P10	P4	P13	P3	P12	P2	P6	P11	P15	P1	P16	P14	*	P7	P8	P9

Burrello et al. obtained Sensitivity and Specificity of 100% with the patient
Burrello et al. did not reach 100% Sensitivity with the patient

*No correspondance

Table 7: Characteristics of the seizures of the 16 different patients from the short-term SWEC-ETHZ iEEG Database

Beforehand it is interesting to analyze the result obtained by Burrello et al. [7] to learn more about the patients. Indeed, their results and our results corroborate the fact that some patients are more prone to automatic seizure detection and good accuracy is easily obtained while, some other patients, are more difficult to analyze and lead to poor FOM. A first observation is that patients with a large number of recorded seizures are more difficult to analyze. As a matter of fact, the more recordings, the more different types of seizures may appear. Even if the training data are larger, the specificity of each seizure event may degrade the training phase and the disparities within the seizures may not be well characterized. These patients are a good representation of the real problem and point out the difficulties that automatic seizure detection still faces today. Burrello et al. [7] obtained 100% accuracy[28] with eight patients while there were not able to detect all seizures onset for five patients for which 7 to 14 seizure recordings were provided. Patient 4[29] is a very difficult case. The database is large and there do not seem to exist, at least with the Reduced Feature Pool of Section 7.2 and the threshold classifier, individual features that lead to good results with this patient. In Table 13 one can observe the poor specificity results obtained with Patient 4 among all the features (often below 70%).

[28] 100% accuracy implies 100% sensitivity and 100% specificity
[29] P3 in Burrello et al.

Burrello et al. algorithm [7]			
ID	Latency [s]	Spc. [%]	Sens. [%]
P1 (ID10)	3.8	100	100
P2 (ID6)	10.4	100	100
P3 (ID4)	4.1	96.16	86.43
P4 (ID2)	24.9	100	100
P5	19.3	99.42	100
P6 (ID7)	8.0	100	100
P7 (ID14)	0.5	89.27	90.00
P8 (ID15)	12.9	100	100
P9 (ID16)	0	88.33	100
P10 (ID1)	0.8	97.44	94.52
P11 (ID8)	2.0	100	100
P12 (ID5)	0	93.22	95.00
P13 (ID3)	7	100	100
P14 (ID12)	7.9	99.53	76.17
P15 (ID9)	22.6	93.91	100
P16 (ID11)	17.3	100	100
Ave.	**8.81**	**97.31**	**96.38**

Table 8: Recapitulative table of the results of Burrello et al. [7] with the short-term SWEC-ETHZ iEEG Database.

7.7.2 Features

In this section, a brief analysis of the Reduced Feature Pool will be discussed. Fig. 7 shows the number of appearances of each feature in the best 5 ranks of the three selection algorithms across all 16 patients. One can observe how the feature *Coastline* outdoes significantly all the other features with a total count of 18. *Coastline* was selected in the top 5 at least once for each patient except for Patients 10,12 13 and 15[30] (See Table 12 in Appendix B). Moreover, it usually appears in the subsets highlighted by the Selection Method. This a very good feature for seizure detection that offer multiple advantages in addition to the good detection performances. Extraction of the *Coastline* is easy and well suited for real-time. The different feature domains are well represented in the most selected features. After *Coastline*, a time domain feature follows a time-frequency domain feature, the absolute mean value of the d1 coefficient of DWT. In the third position, there is the normalized average Band-power of the band β, a frequency domain feature and in the fourth position a non-linear feature, the Approximate Entropy. Moreover, one can observe that, although time domain features may lead to good results with some patients (e.g ID6, ID1, ...), it is not sufficient for others. For example, Patient 8 has only one time domain feature (*Coastline*) selected (See Table 12 in Appendix B) in rank 5 out of the 10 best-ranked features. For that reason, it is justified to include multiple feature domains in the Feature pool fed to the selection method.
In the first place, the mean value was fed as such in our classifier. It was a very problematic feature since it would often come out as a well-ranked feature for many patients while performing poorly with the threshold classifier. By observing the tendency of the mean value with respect to the seizure label, it is clear that a single threshold algorithm cannot be applied to this feature since it shows large positive and negative peaks during seizures. That is why the absolute average value is more suited for classification purposes. Note that the selection methods rank the mean and the absolute mean values similarly. Fig. 8 illustrates the importance to use the absolute value of the average to perform the classification. It shows the mean value of the db5 coefficient obtained by DWT. The threshold value is also indicated. It can be observed that the feature value is below the threshold during the

[30] Rank 8 with RF

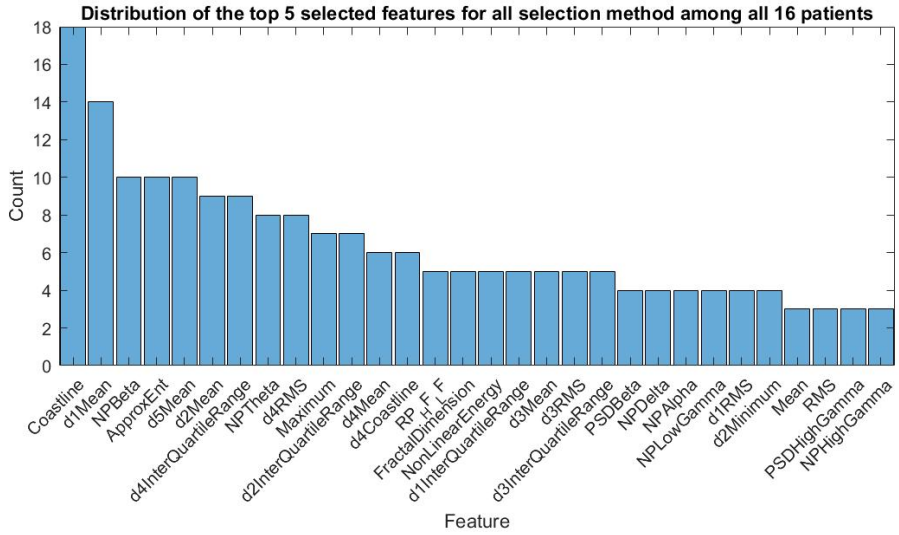

Figure 7: Histogram of the features selected in the top 5 ranking with LASSO,MRMR and RF over the 16 patients

pre-ictal period and that it rapidly increases after the seizure's onset. During the post-ictal period, it seems that the signal still shows some agitation that could lead to false alarms if this feature was used alone with a single window for detection. The specificity could be improved either by consecutive windows validation and/or by combining several features to confirm the seizure on-set. Another possibility would be to increase the threshold but this may increase significantly the latency and this is not a parameter that will be further tuned in this work.

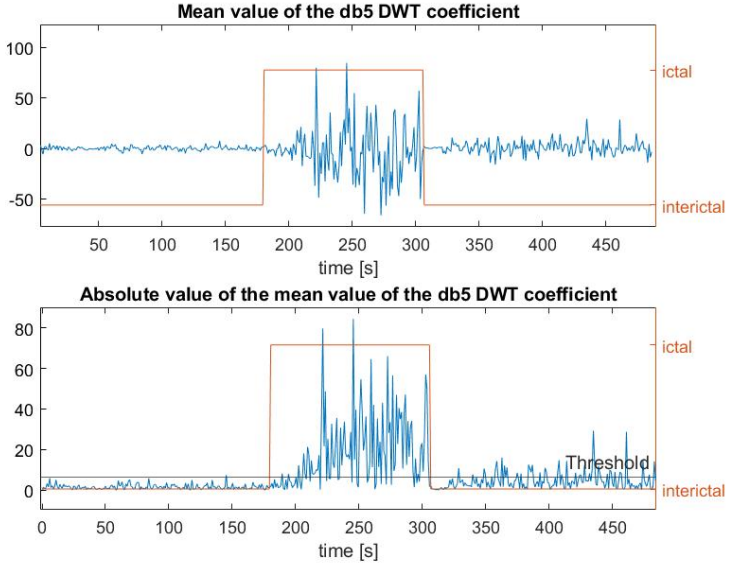

Figure 8: Mean and absolute mean values of the db5 coefficient during seizure 1 of patient 16 recordings. The threshold value for classification is shown in the second plot. The orange curve shows the seizure region marked by the specialist.

	Subset					Training set: Choice			Testing set: validation			Relative error			
ID1	1	2	3	4	5	Specificity	Latency[s]	False Alarm Rate[#/day]	Specificity	Latency[s]	False Alarm Rate[#/day]	Specificity	Latency[s]	False Alarm Rate[#/day]	Result
RF	15	39				99.81%	8.50	122.9	98.69%	15.71	1021.2	1.13%	84.87%	730.91%	SUCCESS
Lasso	6	32	45	28		99.49%	5.67	338.0	97.98%	16.29	1578.2	1.52%	187.39%	366.96%	SUCCESS
Lasso	6	45	31	28		99.49%	5.67	338.0	97.94%	16.29	1609.2	1.56%	187.39%	376.11%	SUCCESS
ID2															
Lasso	41					99.31%	14.00	401.5	97.08%	6.50	1443.4	2.24%	53.57%	259.52%	SUCCESS
Lasso	18	15				95.42%	20.00	2649.8	99.03%	18.50	481.1	3.78%	7.50%	81.84%	SUCCESS
Lasso/RF	18					93.75%	10.50	3613.4	98.19%	18.50	893.6	4.74%	76.19%	75.27%	SUCCESS
ID3															
Lasso	27	36				100.00%	10.00	0.0	100.00%	7.00	0.0	0.00%	30.00%	#DIV/0!	SUCCESS
Lasso	41	36				100.00%	10.00	0.0	100.00%	7.00	0.0	0.00%	30.00%	#DIV/0!	SUCCESS
Lasso	48	36				100.00%	10.00	0.0	100.00%	7.00	0.0	0.00%	30.00%	#DIV/0!	SUCCESS
ID4															
RF/Lasso/MRMR	18					93.06%	26.00	4577.7	95.91%	15.43	2867.9	3.07%	40.66%	37.35%	
RF	57					83.61%	25.57	10803.3	75.48%	8.29	17207.6	9.73%	67.60%	59.28%	
Lasso	3	7				80.71%	23.71	12712.8	71.90%	5.71	19713.6	10.91%	75.90%	55.07%	
ID5															
RF	57	6				99.72%	11.40	187.6	99.33%	16.20	454.3	0.39%	42.11%	142.21%	SUCCESS
RF	57					99.33%	6.00	450.2	98.94%	15.60	719.4	0.39%	160.00%	59.79%	SUCCESS
Lasso	22					97.44%	6.20	1725.7	98.78%	13.40	833.0	1.37%	116.13%	51.73%	SUCCESS

Table 9: Subset selection for patients 1 to 5 with sensitivity 100%, Specificity maximized and Latency below 20s. Classification is based on three consecutive windows validation.

7.7.3 Patients results

To provide an overview of the achievable performance with the selection method combined with the threshold classifier, a selection of optimized subsets was done for patients 1 to 5. The only fixed specification is sensitivity. It is set to 100%. On the other hand, specificity will be maximized (with the training results) while an average detection latency of 20s maximum will be accepted. The three best subset[31] proposed by the selection method as exposed in the EXCEL file for each patient are shown in Fig. 9. For each patient, the first subset of this Table has been set as the selected one. FOM of the training are summarized in Fig. 10 with a list of the selected features. Out of the five patients, Patient 3[32] offers the best results with two DWT features. This good selection can be understood by the fact that the database contains only two seizures, thus the algorithm performs better. A latency of 7s was obtained with 100% accuracy which is equivalent to the result of Burrello et al. in Table 8. Patient 4[33] shows a latency below 20s with the testing data set (7 seizures to classify) set although during training no subset was able to provide such a latency. It can also be observed in Table 13 in Appendix B that individual specificity of most of the features hardly reaches 50% of specificity by a single window with this patient. Again, this result proves that the seizure detection algorithm still requires improvement because it is not able to cope with complicated patients with many seizure recordings. Indeed, the false alarm rate remains very high. Note that even a few false alarm per day would still be very uncomfortable for a patient and that is why specificity should, ideally, be more than 99% with such short recordings to be acceptable. Still, it has been demonstrated that the seizure detection algorithm of this work was able to detect all the seizure events for patients 1 to 5 thus offering a sensitivity of 100%. Another advantage of the algorithm is that it tends to select small subsets thanks to the strict classification. This is indeed preferable for low-power applications.

7.7.4 Future Upgrades

The selection method has been developed as a flexible tool that would support different seizure detection algorithms. Indeed, seizure detection being a hot topic, there already exist multiple possibilities regarding the methods for feature extraction and classification and many more are to emerge. The medical professionals may request different specifica-

[31] For patient 3, there were many more subsets offering the same performances
[32] Patient 13 of Burrello et al. (Fig. 7)
[33] Patient 3 of Burrello et al. (Table 7)

Patient	Sensitivty [%]	Specificity [%]	Latency [s]	False Alarm Rate [#/day]	Subset
ID1	100	98.69	15.71	1021.2	{NPDelta(15), d2IQR(39)}
ID2	100	97.08	6.5	1443.4	{d3RMS(41)}
ID3	100	100	7	0	{d1RMS(27), d2Minimum(36)}
ID4	100	95.91	15.43	2867.9	{NPBeta(18)}
ID5	100	99.33	16.2	454.3	{d5Minimum(57), Coastline(6)}

Table 10: Subset selection with FOM (of testing phase) for patients 1 to 5. Sensitivity is 100% and Specificty was maximized for a latency below 20s.

tions to assist their choice. It is therefore very likely that this method will be used with, for example, a more performing classifier. Still, the method flow proposed in this work will remain the same. In this next paragraph, a non-exhaustive list of the possible future upgrades is presented.

First, one can mention the advantages ([3], [6]) that could offer channel selection, similarly to feature selection. Indeed, the number of recording channels can be very large and it is required to reduce this number for the feature extraction. It is indeed too costly to process all channels individually and would not increase detection accuracy. In this work, an average of all channels was performed for the sake of simplicity and allowed good detection. Still, channel selection could offer improvements.

The possible feature selection methods are numerous and only three were implemented in this work. Indefinite further research can be done in that direction. Other algorithms, based on statistics or correlation[16], for example, may offer different rankings while combinations of more than one method could provide a more mature ranking. Ultimately, the ranking method may need to be adapted to the type of classifier. This is here a large lack since the ranking is done by the software without aiming at features that are particularly suited to a fast and accurate threshold detection. In addition to the specifications available in this application, it could be very useful to add specifications regarding resource usage and power consumption. This would include the hardware limitations in the analysis of the most suitable subset and help the neurologist make a choice suited to the hardware implementation. However, these characteristics cannot be computed directly from the MATLAB extraction code since it is not representative of the hardware implementation. A complete hardware platform needs to be first developed to assess the power consumption of each feature and the required resources. A constraint concerning the available processing blocks may also be needed. Features based on FFT and DWT will lead to a higher computational cost and should be limited in number. To conclude there is a large margin for possible improvements and it is not possible to tune, in a realistic amount of time all parameters at the same time. Still, the hope remains that a seizure detection algorithm suited for wearable or implementable devices will soon be ready for medical use.

8 Hardware Implementation

In this Section, the second part of the project, namely the hardware implementation, is described. The role of this part is to validate the feasibility of a physical implementation of the seizure detection system developed in this work and validate the selection method results. As explained in Section 6.1, the feature selection is only performed offline in this work. Here, an architecture for the feature extraction (only Coastline block), the training of the classifier, and the seizure detection for real-time application are developed on FPGA in VHDL language. The VC707 evaluation board for the Virtex®-7 FPGA was used. The system runs at a clock frequency of 200MHz. The Clocking Wizard LogiCORE™ IP was used to generate an output clock frequency of 50MHz to run the detection algorithm.

8.1 Architecture design

In the first place, the input specifications of the application need to be clearly defined to understand the tasks and requirements that the hardware implementation needs to satisfy. The provided iEEG signals were converted from analog to digital on 16-bits[34] and are sampled at a 512 Hz rate. It was decided to segment the recordings in 1s windows and to store the iEEG values on 16-bits to avoid loss of information due to compression. Thus, there will be 512 samples per window that will be processed to obtain one feature value. As a consequence, each window should be processed in less than 1s to deliver a real-time application. A possible latency can be accepted. This constraint is easily fulfilled with a 50MHz clock. A first top-down approach allowed to establish a primary global architecture and identify the main modules. In this first phase of development, iEEG data will be stored directly in the FPGA memory. Thus, the data are not written in real-time in the memory at the sampling rate of 512 Hz but are stored during a memory initialization prior to hardware computing. It was chosen to do so to center the efforts on the development of the datapath and the data processing. By this assumption, the following modules were defined:

- Static Memory Unit
- Control Unit - FSM
- DMA Controller
  - FSM (for control)
  - Address Generator
- Buffer (to transfer data from the memory to the accelerators)
- Accelerators
- Classifier Training (based on Algorithm 1)
- Classifier (Detection)
- Threshold Memory

In Fig. 9 one can observe the relation between the different blocks listed above.

[34]Data available from the SWEC-ETHZ iEEG Database are provided in .mat files with values saved in 64-bits signed floating point type (probably due to the applied digital filtering). The data were reconverted to 16-bits unsigned integer by offset and rescaling

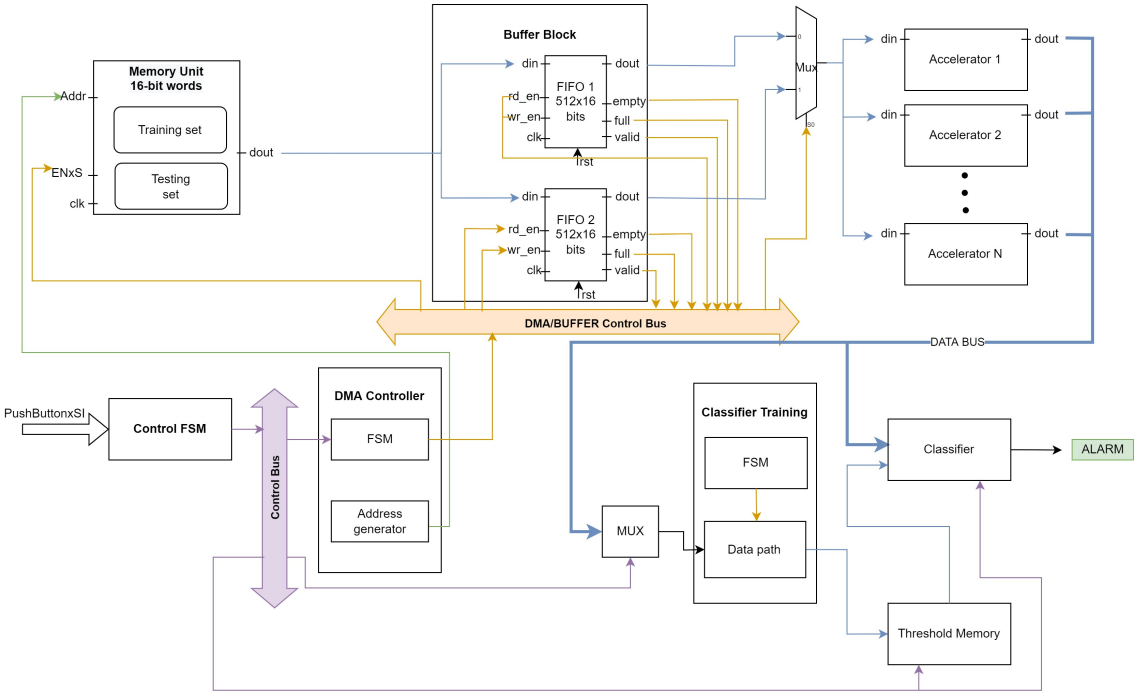

Figure 9: Micro-architecture of the seizure detection algorithm implementation

8.2 Control Unit

The control unit based on a Finite State Machines (FSM) can be triggered by an external input, in this case $PushButtonxS$, and enter either in $TRAINING$ mode or in $TESTING$ mode depending on the previous state as shown in Fig. 10. In the current implementation the dashed lines states are not implemented. They will be required when more than one accelerator will be implemented. $CONFIGURE$ will activate/disable the accelerators blocks chosen during the offline selection. This configuration will control the classification logic but should also disable unused blocks for power savings purpose. One possible solution, although not recommended, would be clock gating. It leads to additional logic but can significantly reduce the power consumption of the system. For a final implementation, the three dashed $TRAIN$ states would replace the single $TRAINING$ state to compute the threshold of each accelerator block sequentially. This allows to reuse et therefore minimize the required hardware resources. In each state, several control signals will be generated and sent to the other blocks. In this implementation the control unit does not communicate with the Memory Unit. This communication is to be implemented in the future for the real-time communication between the PC and the FPGA. The VHDL code can be found in Appendix D.0.2.

8.3 DMA Controller and FIFO Buffer

To transfer and segment EEG signals stored in memory to the other blocks of the architecture, a Direct Memory Access (DMA) controller was designed. It has direct access to the memory unit and generates the addressing. The memory output is driven to the input of the Buffer while the outputs of the two FIFOs are sent to a multiplexer that sends the data ready to be read to the accelerators blocks (Fig. 9). Each FIFO provides state signals that indicate whether the FIFO is empty, full or the output data are valid for reading. When allowed by the control unit, the DMA controls the transfer of data from the memory unit

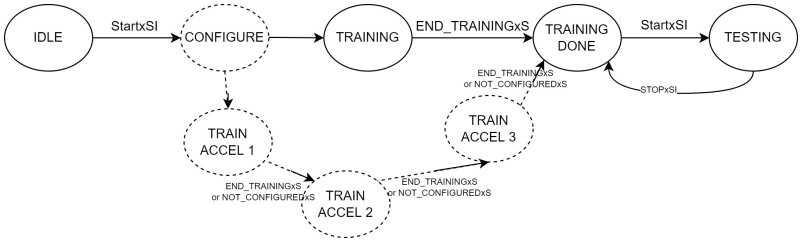

Figure 10: Proposed control FSM - Dashed lines correspond to states that were not yet implemented in the current design

to the buffer block. The FSM shown in Fig. 11 controls the enable/disable signals for reading and writing into the two FIFOs buffers (Moore machine). The state transitions are triggered by the status signals from the two FIFOs: *EmptyxS* and *FullxS* and the process stops either when the memory is empty during training or when the user requests to stop the detection algorithm (*StopxS* during testing). Additionally, the FSM output a high *NewWindowxS* signal during certain transitions (Mealy Machine) to count the number of windows processed and generate the addressing to send to the memory. Additionnal buffer states $WR1\_MEM\_READY$ and $WR1\_MEM\_READY$ were required to synchronize the addressing and the writing into a FIFO. In the current implementation $WR2$, $RD1$, and $RD2$ are states that are never reached because $full\_F1xS$ is true during the same time clock cycle as $empty\_F2xS$ and inversely. Although they are not needed yet, they were kept since they will be required to use certain additional accelerator blocks that require more than 512 cycles to compute. In that case, an additional control signal emitted by the DMA controller will sent the data FSM in these states that pause the buffer until the accelerator is ready again for a new window.

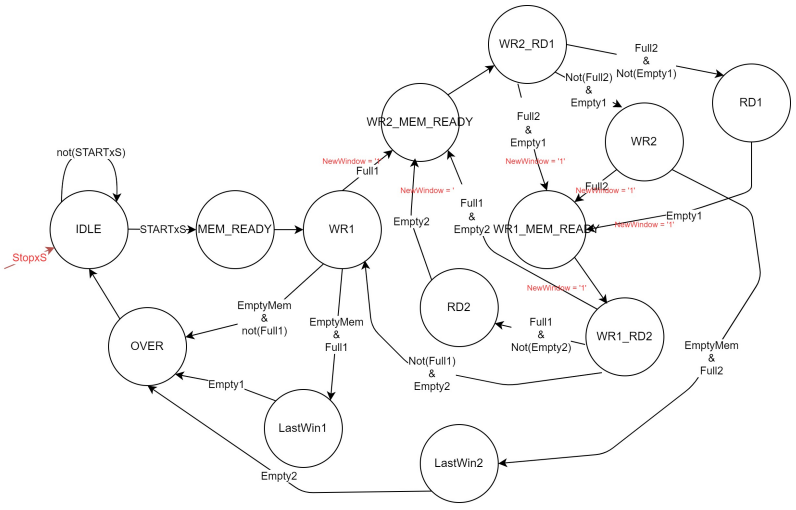

Figure 11: FSM for the control of the data - Read data sample from memory and write windows of 512 samples into FIFO used as a buffer between memory/ADC and accelerators block

8.4 Accelerators

When data are ready to be read from one of the FIFO, the accelerator receives a signal to start processing. Data arrive in the accelerator at the same rate as they were written in

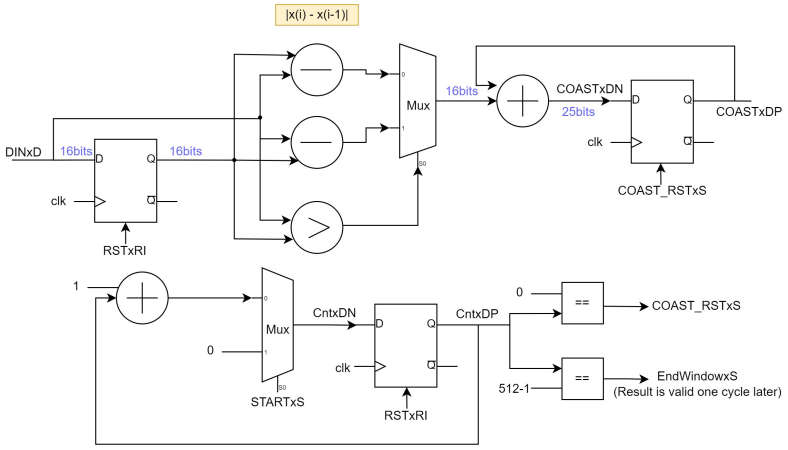

Figure 12: Coastline block

the FIFO (50MHz). It is well suited for the first implemented accelerator *Coastline* that is processed in the same number of cycles required to fill a FIFO. It may be required to use a slower clock for more complex accelerators that cannot be processed in 512 cycles.

8.4.1 Coastline

Coastline is a very suitable feature for hardware implementation. It is the sum over the window of the absolute value of the difference between the current and the previous sample (Eq. 8). The advantage is that the feature can be processed as soon as the samples arrive at the input (once the second sample is read) and that the final value is valid in $CoastxDP$ one cycle after the last sample has been transferred. In Fig. 12 the elements used to compute *Coastline* are presented. A D-Flip-Flop at the entry of the block is used to keep in memory the previous sample. Then the absolute value is obtained by selecting the valid subtraction (positive) with a comparator. The result is added to an accumulator and memorized in another D-Flip-flop until the counter reaches 512-1 and generates a signal $ReadyxS$ to signal that the value will be valid in the next cycle. The counter generates a reset signal that clears the CoastxDP signal[35]. The EEG signal value is stored on 16-bits and the *Coastline* value is stored on 25-bits during the process. In that way, no overflow can occur. The maximum difference between two samples is equivalent to all '1's on 16-bits and a multiplication by 512 corresponds to 9 shifts left thus a maximum of 25-bits (unsigned integer). The $CoastxDP$ is not reduced to 16-bits (by a division by 512 for example) because it could lead to loss of information. The VHDL code is available in Appendix D.0.4.

8.5 Classification

8.5.1 Training

The training phase starts when the user sends a request that is transferred via the control unit to the threshold block. The operations for the threshold algorithm (See Algorithm 1 for pseudocode) are controlled with a Moore FSM shown in Fig. 13. It remains in *IDLE* state until the $training\_startxS$ signal is set high. The training phase is divided into two stages. The first one consists of memory filling and sorting which corresponds to the

[35]The sum is calculated from the second cycle because it should wait for two samples

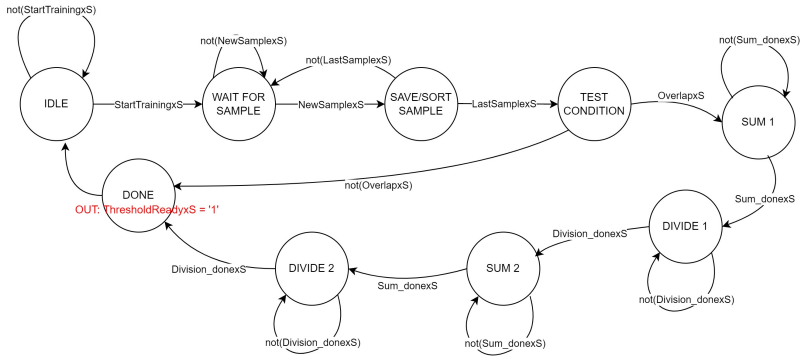

Figure 13: Moore FSM of the threshold algorithm hardware implementation

states *WAIT FOR SAMPLE* and *SAVE/SORT SAMPLE*. For each new value that the accelerator (that is being trained) computes for a 1s window, a new sample can be saved. Indeed, the accelerator requires more than one cycle to process a 1s window (minimum 512 cycles to read the FIFO elements). Once the training values are saved, the test corresponding to line 12 of the Algorithm 1 is evaluated. If the test returns true, the threshold value is known and the training is over, otherwise, the threshold requires several cycles to be determined. For that, four sums ($SumLow$, $NumLow$, $SumHigh$, $NumHigh$) and two divisions (($SumLow/NumLow$, $SumHigh/NumHigh$) must be performed (line 23 of Algorithm 1) and ultimately a shift right (division by 2). It was chosen to perform sequentially the result of the four sums and the two divisions to reuse the blocks since there are nos constraints of time for these operations and it allows to minimize area by using only two counters and one divider as illustrated in the data flow of threshold algorithm in Fig. 14. State $SUM1$ (and $SUM2$) will last a variable number of cycles equal to the number of saved samples. When the transition to the next state is enabled, the data $SUMxDP$ and $CNTxDP$ are sent to the divider inputs, respectively the dividend and the divider. while the division is performed in a fixed number of cycles of 29. This training flow is to be repeated with each selected feature (accelerator) sequentially (Control FSM) with the same training set. Thus, the same memory blocks can be reused to save the ictal and non-ictal values. The size of these two blocks of memory will depend on the size of the training database. For example, a training of the *Coastline* feature with the first seizure of patient 6[36] would require one block of (180x2x512)x25-bits and one block of (89x512)x25-bits which leads to a total of 718 400 KB which is quite consequent already. The proposed algorithm has the disadvantage that it needs to compare all the values of the accelerator (with the training set) to define the $MaxValxDP$ and $MinValxDP$ before switching to state $TEST$ and starting the calculations. To eliminate these two memory blocks during the training phase, it would be possible, once the real-time communication with the PC is implemented, to, first, send these values to the PC and store them. Simultaneously, $MaxValxDP$ and $MinValxDP$ would be determined. And secondly to send back from the PC to the threshold block the stored data. Again, the training is not constrained by time in this case and any delay in the data transmission is acceptable. This classifier is very advantageous with respect to other classifiers in term of resources. The computational cost is low and only a few parameters have to be saved after the training: the threshold values of each selected accelerator.

[36] 2x3min of non-ictal samples et 89s of ictal samples

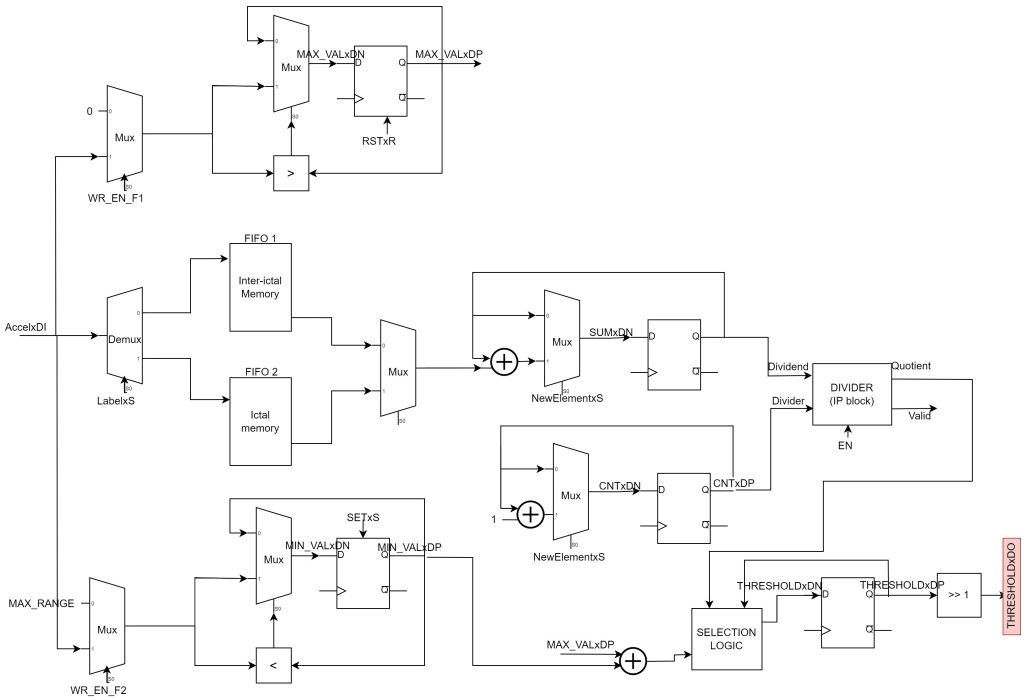

Figure 14: Data flow of the training block (with simplifications)

8.5.2 Detection

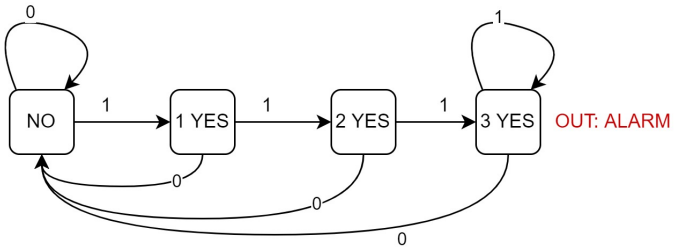

Figure 15: FSM for seizure detection with three consecutive windows validation

The multiple consecutive windows validation for classification was simply implemented with a FSM with four states. If the label obtained from the comparison between the accelerator block output value and the threshold computed during the training is '1' (ictal) the FSM goes from state NO to $YES1$. When state $3YES$ is reached the alarm is set.

8.6 Discussions

In this section, a global hardware architecture for training and the testing phase was proposed. Efforts were made to minimize the resources by reusing blocks when it was possible. This is for example well suited for the threshold algorithm where it is not required to process all the feature values at the same time, the training data set being accessible at all times[37]. The value of the Coastline block is stored on 25-bits. By a simple division by a constant (a multiple of 2), it is possible to reduce the number of bits. This would allow reducing the memory blocks required during the training. An analysis of the accuracy with

[37] not real-time acquisition since it requires the labels from the neurologist

respect to the bit reduction would allow finding the optimal number of bits (between 25 and 16) that do not lead to critical loss of information. Due to technical difficulties and time constraints, it was not possible to propose a demonstration on the FPGA and collect the detection results obtained with the implemented *Coastline* accelerator. The data transfer through FIFOs from the memory unit to the accelerator block and the *Coastline* training phase were validated with simulations. The division used with threshold classifier is not optimal but still requires fewer resources than a machine learning based classifier. The next step for this hardware implementation will first be to develop a real-time PC to FPGA data transmission to simulate real-time data acquisition. Secondly, additional accelerator blocks should be implemented with appropriate control management regarding the selected features and system to disable non-selected blocks for power savings.

9 Conclusion

A state-of-the-art study of the existing seizure detection systems leads to the identification of possible areas of improvement. Hence, it was proposed to develop a detection algorithm from software to hardware implementation, the final goal being an implantable device for closed-loop seizure detection system with patient-customized solutions. This implementation aimed to optimize the seizure on-set detection latency without degrading the detection accuracy. For that purpose, a large Feature Pool was built with features of different domains that are commonly found in the literature. The following selection algorithms were chosen to generate small subsets of features optimized for each patient: Least Absolute Shrinkage and Selection Operator (LASSO), Maximum Relevance Minimum Redundancy (MRMR) and Random Forest (RF). A classification procedure based on threshold comparison was used. It was then suggested to complete the software processing by an intervention of the neurologist in the process because their expertise would allow finalizing the decision process of optimal subsets. The developed selection method with MATLAB and EXCEL demonstrated promising results. It was designed such that it offers flexibility to support further needs, specifications, or requirements for future development. It was possible to identify good subsets for patients with a small number of seizure recordings that could reach 100% sensitivity and accuracy for both the training and the testing set. Results in latency were acceptable (below 20s) but there is a place for further improvements. Parameters such as the Feature Pool, the window length (with possible overlap), the threshold value, the number of consecutive windows for seizure detection, or the logical function applied within the selected features can be further tuned. It was demonstrated that, as observed in the work of Burrello et al. [7], patients with large data sets are more illustrative of reality and more complex to characterize. These patients showed lower detection accuracy or long detection latency. One can conclude that the seizure detection algorithm is still not robust enough and a more adaptive methodology for training and classification may solve this issue. This work was able to show that the Coastline is a very interesting feature that was often chosen by the different selection methods and demonstrated good performance for seizure detection with several patients. The hardware implementation also points out that this feature is well-suited for real-time applications since it can be processed at the same rate as the data inputs without any delay and with only one D-Flip to save the previous data sample. A hardware architecture was proposed to implement the seizure detection algorithm.

In future works, the hardware implementation can be further extended. The system should propose a set of available accelerator blocks that can be activated/disabled according to the neurologist feature subset selection. Ultimately, this block should be implemented in a complete closed-loop seizure detection system that will consist of a data acquisition block, the block proposed in this work (feature extraction, training, detection), and a feedback generation block and aim to stop the seizure in real-time.

Bibliography

[1] Patrick Kwan, Steven C. Schachter, and Martin J. Brodie. "Drug-Resistant Epilepsy". In: *New England Journal of Medicine* 365.10 (2011). PMID: 21899452, pp. 919–926. DOI: 10.1056/NEJMra1004418. eprint: https://doi.org/10.1056/NEJMra1004418. URL: https://doi.org/10.1056/NEJMra1004418.

[2] Zhen Jiang and Wenshan Zhao. "Optimal Selection of Customized Features for Implementing Seizure Detection in Wearable Electroencephalography Sensor". In: *IEEE Sensors Journal* 20.21 (2020), pp. 12941–12949. DOI: 10.1109/JSEN.2020.3003733.

[3] Hongda Wang, Weiwei Shi, and Chiu-Sing Choy. "Hardware Design of Real Time Epileptic Seizure Detection Based on STFT and SVM". In: *IEEE Access* 6 (2018), pp. 67277–67290. DOI: 10.1109/ACCESS.2018.2870883.

[4] Keyvan Farhang Razi et al. "System-Level Modeling of a Safe Autonomous Closed-loop Epileptic Seizure Control Implant". In: *2021 4th International Conference on Bio-Engineering for Smart Technologies (BioSMART)*. 2021, pp. 1–4. DOI: 10.1109/BioSMART54244.2021.9677729.

[5] Puja Patel and Solomon L Moshé. "The evolution of the concepts of seizures and epilepsy: What's in a name?" en. In: *Epilepsia Open* 5.1 (Jan. 2020), pp. 22–35.

[6] Genchang Peng et al. "Personalized Feature Selection for Wearable EEG Monitoring Platform". In: *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*. 2020, pp. 380–386. DOI: 10.1109/BIBE50027.2020.00069.

[7] Alessio Burrello et al. "An Ensemble of Hyperdimensional Classifiers: Hardware-Friendly Short-Latency Seizure Detection With Automatic iEEG Electrode Selection". In: *IEEE Journal of Biomedical and Health Informatics* 25.4 (2021), pp. 935–946. DOI: 10.1109/JBHI.2020.3022211.

[8] Alessio Burrello et al. "Hyperdimensional Computing with Local Binary Patterns: One-shot Learning of Seizure Onset and Identification of Ictogenic Brain Regions using Short-time iEEG Recordings". en. In: *IEEE Transactions on Biomedical Engineering* 67.2 (2020-02), pp. 601–613. ISSN: 0018-9294. DOI: 10.3929/ethz-b-000350002.

[9] L.P Wijesinghe, D.S Wickramasuriya, and Ajith A. Pasqual. "A generalized preprocessing and feature extraction platform for scalp EEG signals on FPGA". In: *2014 IEEE Conference on Biomedical Engineering and Sciences (IECBES)*. 2014, pp. 137–142. DOI: 10.1109/IECBES.2014.7047472.

[10] Jose V. Frances-Villora et al. "Real-Time Localization of Epileptogenic Foci EEG Signals: An FPGA-Based Implementation". In: *Applied Sciences* 10.3 (2020). ISSN: 2076-3417. DOI: 10.3390/app10030827. URL: https://www.mdpi.com/2076-3417/10/3/827.

[11] Behrooz Abbaszadeh, César Teixeira, and Mustapha Yagoub. "Feature Selection Techniques for the Analysis of Discriminative Features in Temporal and Frontal Lobe Epilepsy: A Comparative Study". In: *The Open Biomedical Engineering Journal* 15 (June 2021), pp. 1–15. DOI: 10.2174/1874120702115010001.

[12] M. Niknazar et al. "A unified approach for detection of induced epileptic seizures in rats using ECoG signals". In: *Epilepsy Behavior* 27.2 (2013), pp. 355–364. ISSN: 1525-5050. DOI: https://doi.org/10.1016/j.yebeh.2013.01.028. URL: https://www.sciencedirect.com/science/article/pii/S1525505013000474.

[13] M. Bedeeuzzaman, Omar Farooq, and Yusuf Khan. "Automatic Seizure Detection using Inter Quartile Range". In: *International Journal of Computer Applications* 44 (Apr. 2012), pp. 1–5. DOI: 10.5120/6304-8614.

[14] Dakila Serasinghe, Duvindu Piyasena, and Ajith Pasqual. "A Novel Low-Complexity VLSI Architecture for an EEG Feature Extraction Platform". In: *2018 21st Euromicro Conference on Digital System Design (DSD)*. 2018, pp. 472–478. DOI: 10.1109/DSD.2018.00084.

[15] U. Rajendra Acharya et al. "Automated diagnosis of epileptic EEG using entropies". In: *Biomedical Signal Processing and Control* 7.4 (2012), pp. 401–408. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2011.07.007. URL: https://www.sciencedirect.com/science/article/pii/S1746809411000838.

[16] Poomipat Boonyakitanont et al. "A review of feature extraction and performance evaluation in epileptic seizure detection using EEG". In: *Biomedical Signal Processing and Control* 57 (2020), p. 101702. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2019.101702. URL: https://www.sciencedirect.com/science/article/pii/S1746809419302836.

[17] Heba Elhosary et al. "Low-Power Hardware Implementation of a Support Vector Machine Training and Classification for Neural Seizure Detection". In: *IEEE Transactions on Biomedical Circuits and Systems* 13.6 (2019), pp. 1324–1337. DOI: 10.1109/TBCAS.2019.2947044.

[18] Josef Parvizi and Sabine Kastner. "Promises and limitations of human intracranial electroencephalography". eng. In: *Nature neuroscience* 21.4 (Apr. 2018), pp. 474–483. ISSN: 1546-1726. DOI: 10.1038/s41593-018-0108-2. URL: https://doi.org/10.1038/s41593-018-0108-2.

[19] Inc. Copyright 2019 The MathWorks. *fscmrmr: Rank features for classification using minimum redundancy maximum relevance (MRMR) algorithm*. URL: https://ch.mathworks.com/help/stats/fscmrmr.html. (accessed: 05.05.2022).

[20] G.A. Darbellay and I. Vajda. "Estimation of the information by an adaptive partitioning of the observation space". In: *IEEE Transactions on Information Theory* 45.4 (1999), pp. 1315–1321. DOI: 10.1109/18.761290.

[21] Peng Zhao and B. Yu. "On model selection consistency of LASSO". In: *Journal of Machine Learning Research* 7 (Dec. 2006), pp. 2541–2563.

[22] Kai-Quan Shen et al. "A Feature Selection Method for Multilevel Mental Fatigue EEG Classification". In: *IEEE Transactions on Biomedical Engineering* 54.7 (2007), pp. 1231–1237. DOI: 10.1109/TBME.2007.890733.

[23] Yuanfa Wang et al. "Hardware Design of Seizure Detection Based on Wavelet Transform and Sample Entropy". In: *Journal of Circuits, Systems and Computers* 25.09 (2016), p. 1650101. DOI: 10.1142/S0218126616501012. eprint: https://doi.org/10.1142/S0218126616501012. URL: https://doi.org/10.1142/S0218126616501012.

Appendix A MATLAB code

The code used for the software implementation is available on request. It has been written according to the format of the short-term SWEC-ETHZ iEEG Database. The main file can be run without any input arguments. In this case, the ID of the patient should be specified in the main script. Other parameters can be tuned such as the depth of analysis for the selection method, the number of consecutive positively labelled windows required for a seizure detection and the window length in seconds. If one need to extract the selection method results, for one or more patients, in an excel file, the script RunAllPatients.m can be launched without arguments. To use this code with other datasets, the path to the file in read_data.m should be updated and the reading method may need changes to correspond to the data format of the recordings. In the same way, the segmentation of the windows and creation of the label vector may need updates.
Features are stored as structures with 4 fields: Name, ID, type and threshold. The name and the type are read (*get_feature_info.m*) from an excel file where all information are stored. The threshold is computed during the training phase and is used for the classification of the training and the testing pool.
The following MATLAB Toolbox are required to execute the code:

- Signal processing Toolbox
- Statistics and Machine learning Toolbox
- Wavelet Toolbox
- Predictive Maintenance Toolbox
- System Identification Toolbox
- Data Acquisition Toolbox

The Higuchi Fractal Dimension was computed with the script from Jesús Monge Álvarez[38]

Appendix B Selection method: Excel results

ID	Name	Domain
1	Mean∗	Time
2	RMS	Time
3	Maximum	Time
4	Minimum	Time
5	Variance	Time
6	Coastline	Time
7	IQR	Time
8	PSDRaw	Frequency
9	PSDDelta	Frequency
10	PSDTheta	Frequency
11	PSDAlpha	Frequency
12	PSDBeta	Frequency
13	PSDLowGamma	Frequency

ID	Name	Domain
35	d2Maximum	Time-freq.
36	d2Minimum	Time-freq.
37	d2Variance	Time-freq.
38	d2Coastline	Time-freq.
39	d2IQR	Time-freq.
40	d3Mean∗	Time-freq.
41	d3RMS	Time-freq.
42	d3Maximum	Time-freq.
43	d3Minimum	Time-freq.
44	d3Variance	Time-freq.
45	d3Coastline	Time-freq.
46	d3IQR	Time-freq.
47	d4Mean∗	Time-freq.

[38] https://github.com/malkhodari/COVID19_breathing_machine_learning/blob/main/Higuchi_FD.m

ID	Name	Domain
14	PSDHighGamma	Frequency
15	NPDelta	Frequency
16	NPTheta	Frequency
17	NPAlpha	Frequency
18	NPBeta	Frequency
19	NPLowGamma	Frequency
20	NPHighGamma	Frequency
21	RP_HF_LF	Frequency
22	RPBetaGamma	Frequency
23	ApproxEnt	NonLinear
24	FractalDimension	NonLinear
25	NonLinearEnergy	NonLinear
26	d1Mean*	Time-freq.
27	d1RMS	Time-freq.
28	d1Maximum	Time-freq.
29	d1Minimum	Time-freq.
30	d1Variance	Time-freq.
31	d1Coastline	Time-freq.
32	d1IQR	Time-freq.
33	d2Mean*	Time-freq.
34	d2RMS	Time-freq.

ID	Name	Domain
48	d4RMS	Time-freq.
49	d4Maximum	Time-freq.
50	d4Minimum	Time-freq.
51	d4Variance	Time-freq.
52	d4Coastline	Time-freq.
53	d4IQR	Time-freq.
54	d5Mean*	Time-freq.
55	d5RMS	Time-freq.
56	d5Maximum	Time-freq.
57	d5Minimum	Time-freq.
58	d5Variance	Time-freq.
59	d5Coastline	Time-freq.
60	d5IQR	Time-freq.
61	a5Mean*	Time-freq.
62	a5RMS	Time-freq.
63	a5Maximum	Time-freq.
64	a5Minimum	Time-freq.
65	a5Variance	Time-freq.
66	a5Coastline	Time-freq.
67	a5IQR	Time-freq.

Table 11: Reduced feature pool used with the selection algorithms

*The absolute value of this feature is considered.

Rank	1	2	3	4	5	6	7	8	9	10
ID1										
Lasso	6	32	3	17	45	1	31	54	50	28
MRMR	6	54	17	47	26	33	61	40	35	1
RF	15	6	32	40	1	35	17	18	39	13
Specificity	25	30	37	14	27	34	44	12	58	6
ID2										
Lasso	48	41	42	49	3	50	38	18	43	15
MRMR	6	23	54	26	33	47	50	22	40	13
RF	48	54	9	6	32	56	18	57	25	35
Specificity	14	12	37	58	51	56	25	13	35	34
ID3										
Lasso	27	6	41	48	36	49	29	18	54	50
MRMR	27	18	33	26	54	40	47	36	53	17
RF	27	36	26	50	17	1	6	42	47	16
Specificity	13	14	42	43	10	44	30	28	40	58
ID4										
Lasso	21	18	28	30	3	1	17	7	20	12
MRMR	18	26	33	25	1	16	61	59	28	15
RF	53	18	12	6	57	23	15	39	14	38
Specificity	21	20	19	18	13	17	61	54	10	14
ID5										
Lasso	6	66	59	23	46	39	20	32	19	22

Rank	1	2	3	4	5	6	7	8	9	10
MRMR	2	33	40	47	46	22	59	10	39	1
RF	2	39	4	32	46	3	57	6	8	14
Specificity	58	11	10	12	8	9	5	51	65	7
ID6										
Lasso	55	48	66	41	18	15	42	60	23	17
MRMR	52	23	26	40	61	47	17	33	54	39
RF	53	54	61	6	52	16	19	13	12	18
Specificity	51	48	52	13	50	12	58	49	11	25
ID7										
Lasso	44	30	53	63	33	19	26	56	42	13
MRMR	44	40	16	33	1	61	54	15	47	26
RF	14	6	28	27	45	40	37	46	15	31
Specificity	6	14	27	30	39	25	38	41	44	45
ID8										
Lasso	55	48	66	41	18	15	42	60	23	17
MRMR	52	23	26	40	61	47	17	33	54	39
RF	53	54	61	6	52	16	19	13	12	18
Specificity	51	48	52	13	50	12	58	49	11	25
ID9										
Lasso	48	55	50	41	49	3	15	56	45	5
MRMR	6	15	47	26	54	33	40	50	18	42
RF	50	19	26	6	3	49	23	42	13	41
Specificity	29	36	43	13	28	12	42	35	11	50
ID10										
Lasso	37	14	21	19	39	11	10	58	66	5
MRMR	39	16	40	47	26	19	46	35	32	17
RF	39	2	38	36	64	16	3	32	26	14
Specificity	13	46	14	44	45	41	43	37	42	12
ID11										
Lasso	6	15	21	53	19	29	56	8	20	17
MRMR	25	15	47	26	33	40	22	39	57	18
RF	6	36	23	19	64	7	2	26	54	16
Specificity	29	5	21	8	10	17	18	50	66	22
ID12										
Lasso	52	51	21	12	53	58	8	16	9	19
MRMR	52	16	18	33	54	26	24	30	61	53
RF	21	24	23	52	3	18	53	2	17	27
Specificity	51	12	44	25	13	14	58	37	48	43
ID13										
Lasso	32	53	23	18	39	16	13	35	65	52
MRMR	48	18	46	53	13	12	16	50	39	1
RF	53	23	48	3	18	12	10	2	14	50
Specificity	51	12	13	58	8	11	10	25	44	48
ID14										
Lasso	24	6	23	20	39	31	38	16	45	41
MRMR	25	17	16	22	26	47	40	61	60	33
RF	12	3	66	25	49	1	11	48	29	18
Specificity	5	8	65	51	9	10	25	58	3	7

Rank	1	2	3	4	5	6	7	8	9	10
ID15										
Lasso	55	52	24	53	16	20	18	22	57	19
MRMR	10	22	54	23	16	60	24	18	51	67
RF	24	20	22	12	16	3	19	6	53	1
Specificity	11	50	53	24	12	22	51	57	56	64
ID16										
Lasso	48	41	4	18	57	50	21	35	15	66
MRMR	6	20	26	54	33	47	40	61	18	60
RF	39	23	43	46	13	15	45	18	41	28
Specificity	10	42	14	13	36	35	43	28	44	5

Time Domain
Frequency Domain
Non Linear Features
DWT

Table 12: Top 10 ranking of the three selection methods (LASSO, MRMR and RF) and Top 10 specificity ranking obtained with the threshold classification on the training set

	ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8
1	75.14%	65.97%	73.61%	58.25%	85.39%	61.39%	75.56%	79.17%
2	74.35%	75.97%	80.00%	42.94%	87.44%	87.08%	98.33%	85.83%
3	76.76%	82.36%	86.11%	43.33%	85.89%	90.28%	98.06%	87.50%
4	74.63%	77.78%	83.33%	42.78%	86.89%	92.50%	96.11%	84.44%
5	82.45%	83.19%	85.00%	51.51%	88.83%	91.67%	98.33%	87.22%
6	88.15%	90.42%	75.83%	45.32%	83.78%	89.44%	99.44%	94.17%
7	75.74%	78.89%	81.94%	45.04%	87.94%	85.14%	98.33%	87.78%
8	82.59%	87.92%	89.17%	58.29%	90.28%	90.00%	97.78%	89.72%
9	80.88%	77.64%	83.33%	61.07%	90.06%	81.81%	86.94%	88.06%
10	88.15%	90.42%	91.94%	61.79%	91.72%	91.67%	87.50%	88.33%
11	86.62%	88.89%	88.06%	61.51%	91.78%	93.89%	89.44%	80.28%
12	88.47%	95.83%	88.06%	60.36%	90.83%	94.17%	96.94%	82.78%
13	86.34%	92.64%	97.22%	65.79%	85.39%	95.97%	98.89%	88.33%
14	92.13%	96.67%	94.44%	61.71%	84.00%	92.08%	99.44%	91.94%
15	44.21%	65.97%	60.00%	56.03%	32.94%	76.81%	54.17%	72.50%
16	56.44%	66.94%	56.94%	48.10%	51.61%	69.17%	46.39%	58.61%
17	53.56%	70.69%	69.72%	63.93%	51.22%	78.33%	58.06%	46.94%
18	48.98%	71.67%	72.78%	69.29%	39.83%	82.36%	70.83%	38.33%
19	53.52%	65.69%	63.06%	71.75%	44.56%	76.53%	65.83%	44.17%
20	52.69%	61.81%	60.56%	75.16%	42.61%	66.67%	68.89%	42.22%
21	59.40%	72.08%	72.22%	75.83%	47.22%	83.89%	81.39%	43.61%
22	63.01%	82.22%	71.94%	60.00%	74.00%	81.53%	72.78%	60.56%
23	58.80%	47.36%	53.33%	34.56%	76.72%	41.94%	56.67%	69.72%
24	53.33%	63.89%	59.72%	39.60%	66.50%	61.53%	53.61%	61.94%
25	94.91%	93.06%	80.83%	50.52%	87.83%	93.75%	99.17%	93.89%
26	66.85%	76.67%	72.50%	61.35%	63.83%	79.17%	81.94%	75.83%
27	90.05%	87.78%	89.44%	52.14%	74.78%	88.75%	99.44%	96.39%
28	85.00%	89.86%	90.56%	52.78%	74.89%	88.61%	98.06%	91.94%

	ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8
29	85.83%	86.11%	89.72%	53.65%	73.72%	88.89%	96.67%	90.56%
30	94.77%	90.97%	91.39%	59.05%	80.50%	91.94%	99.44%	96.67%
31	87.50%	84.86%	85.83%	52.10%	72.72%	90.00%	98.89%	95.28%
32	85.42%	78.33%	75.56%	55.75%	71.11%	91.94%	98.61%	91.67%
33	64.21%	75.14%	78.33%	59.60%	61.78%	80.28%	72.78%	63.06%
34	89.95%	92.22%	80.56%	51.51%	78.78%	86.67%	98.89%	91.94%
35	85.60%	92.50%	86.39%	56.35%	75.78%	88.33%	97.78%	95.28%
36	86.06%	88.61%	90.28%	54.05%	72.17%	86.39%	97.50%	92.22%
37	94.72%	94.58%	84.17%	58.25%	83.78%	89.86%	98.89%	94.72%
38	87.78%	89.58%	78.89%	53.17%	77.44%	86.81%	99.17%	93.06%
39	84.81%	81.39%	73.06%	55.99%	77.17%	92.78%	99.44%	93.33%
40	65.28%	77.36%	90.56%	59.72%	63.78%	81.94%	84.44%	61.67%
41	83.94%	90.42%	85.83%	51.15%	81.78%	90.28%	99.17%	84.17%
42	81.57%	91.94%	94.17%	54.72%	80.06%	91.39%	98.06%	89.44%
43	83.33%	91.11%	93.61%	59.25%	77.28%	89.17%	98.61%	86.67%
44	89.72%	91.94%	91.67%	60.32%	85.06%	93.75%	99.17%	86.11%
45	82.59%	89.31%	82.50%	50.67%	81.17%	91.11%	99.17%	85.28%
46	81.57%	85.83%	76.94%	53.49%	78.78%	90.00%	99.17%	92.22%
47	66.30%	83.47%	83.06%	61.03%	65.83%	83.89%	87.50%	67.22%
48	79.72%	89.58%	82.50%	48.33%	83.50%	97.36%	98.06%	77.78%
49	78.70%	90.56%	85.56%	51.03%	84.83%	94.03%	94.72%	83.61%
50	79.35%	90.14%	90.28%	55.16%	83.22%	95.28%	97.22%	81.94%
51	87.45%	93.89%	87.78%	56.71%	88.33%	98.89%	98.61%	83.61%
52	78.43%	89.31%	86.67%	49.60%	83.28%	96.11%	96.94%	78.89%
53	74.95%	85.56%	88.89%	53.06%	78.22%	82.50%	93.33%	78.61%
54	70.46%	81.53%	80.56%	62.18%	67.17%	81.81%	81.11%	66.39%
55	78.70%	90.00%	85.56%	47.98%	86.50%	90.69%	95.83%	80.00%
56	76.62%	93.61%	88.89%	50.40%	84.67%	93.06%	87.22%	76.39%
57	78.24%	85.28%	90.00%	51.43%	84.28%	92.78%	91.11%	77.50%
58	88.19%	94.17%	90.56%	57.78%	92.67%	94.17%	96.67%	84.44%
59	76.53%	89.72%	81.11%	48.57%	86.33%	87.08%	95.28%	76.67%
60	72.87%	85.42%	80.83%	52.78%	80.28%	85.28%	90.83%	76.94%
61	73.47%	73.06%	78.06%	63.21%	78.06%	70.00%	80.56%	72.22%
62	71.71%	73.33%	78.89%	44.13%	85.67%	79.44%	89.72%	88.33%
63	73.01%	77.64%	86.94%	44.01%	85.39%	79.17%	89.17%	78.89%
64	70.05%	75.14%	80.83%	47.14%	86.28%	80.97%	88.61%	84.72%
65	80.14%	80.97%	83.89%	54.33%	88.17%	87.22%	92.50%	90.28%
66	73.56%	75.00%	76.39%	44.09%	85.61%	83.47%	89.72%	85.28%
67	72.36%	74.31%	83.06%	57.30%	86.00%	77.92%	85.28%	82.50%

Table 13: Individual specificity of each feature with single window validation (Patient 1 to 8)

	ID9	ID10	ID11	ID12	ID13	ID14	ID15	ID16
1	73.26%	77.08%	65.83%	65.33%	65.74%	74.72%	78.06%	80.28%
2	80.49%	85.97%	83.89%	71.67%	66.94%	94.54%	61.94%	95.93%
3	90.49%	88.33%	81.11%	73.11%	78.33%	94.63%	59.44%	97.96%
4	87.43%	87.78%	79.72%	73.44%	67.78%	92.31%	76.11%	95.09%

	ID9	ID10	ID11	ID12	ID13	ID14	ID15	ID16
5	90.63%	90.14%	90.83%	80.50%	86.20%	98.52%	68.33%	98.15%
6	73.89%	91.94%	46.67%	79.00%	71.39%	84.17%	64.72%	86.48%
7	80.76%	84.44%	76.11%	71.44%	67.50%	94.63%	72.50%	95.09%
8	91.88%	89.58%	90.00%	81.22%	88.52%	97.50%	66.11%	98.15%
9	80.49%	88.47%	78.33%	79.17%	86.30%	96.76%	82.50%	93.89%
10	95.56%	87.22%	89.44%	79.44%	87.87%	96.76%	86.39%	99.54%
11	95.83%	85.83%	78.33%	80.67%	88.06%	90.93%	94.17%	97.69%
12	96.46%	93.61%	66.11%	89.61%	93.70%	91.39%	91.11%	96.48%
13	97.01%	97.92%	67.78%	87.00%	93.06%	76.11%	81.94%	99.17%
14	92.36%	96.67%	81.94%	85.44%	81.20%	79.17%	65.83%	99.35%
15	72.01%	58.61%	85.00%	46.11%	48.15%	38.06%	55.00%	72.59%
16	67.29%	41.81%	83.89%	46.28%	53.06%	67.04%	79.44%	63.15%
17	76.88%	45.42%	87.50%	50.94%	61.02%	59.63%	81.11%	72.96%
18	77.43%	72.22%	87.50%	65.50%	63.24%	43.06%	77.78%	82.41%
19	70.35%	80.00%	82.50%	63.94%	64.07%	38.24%	55.28%	78.89%
20	66.88%	76.67%	76.39%	61.00%	69.81%	33.24%	54.72%	75.19%
21	79.38%	81.81%	90.28%	80.00%	73.43%	45.09%	62.78%	85.28%
22	79.51%	55.00%	86.11%	73.50%	55.46%	84.35%	90.83%	70.93%
23	46.88%	40.69%	32.22%	56.11%	46.67%	81.48%	79.44%	48.89%
24	59.79%	48.33%	71.94%	61.78%	35.46%	81.39%	91.39%	63.98%
25	89.24%	93.33%	55.83%	87.17%	87.69%	96.02%	69.17%	95.28%
26	80.21%	85.69%	77.22%	63.89%	71.30%	63.33%	73.89%	81.20%
27	82.85%	88.61%	58.06%	75.28%	66.85%	78.24%	58.89%	91.20%
28	96.60%	92.22%	84.44%	74.94%	68.70%	77.59%	60.83%	98.70%
29	97.29%	92.78%	91.11%	74.89%	67.59%	77.41%	61.94%	97.96%
30	89.72%	91.67%	65.28%	79.39%	83.80%	82.22%	68.89%	93.89%
31	79.58%	87.92%	55.56%	73.61%	67.59%	76.30%	63.06%	87.96%
32	78.82%	88.61%	50.56%	71.61%	69.72%	70.93%	66.39%	94.81%
33	71.11%	80.97%	64.72%	64.61%	69.17%	62.87%	77.22%	84.35%
34	81.25%	91.94%	62.22%	78.17%	68.98%	84.26%	57.78%	94.54%
35	96.11%	92.50%	76.67%	77.89%	70.65%	81.20%	60.00%	98.89%
36	97.22%	92.92%	72.50%	79.67%	71.39%	84.07%	62.78%	98.98%
37	87.29%	93.89%	68.61%	84.28%	83.80%	92.04%	62.78%	96.67%
38	76.81%	90.69%	63.33%	76.94%	69.44%	80.09%	60.83%	85.37%
39	75.28%	90.83%	58.33%	73.44%	71.30%	76.57%	64.44%	86.39%
40	80.07%	85.28%	68.06%	64.94%	66.76%	62.96%	76.11%	83.61%
41	83.96%	95.14%	57.22%	81.00%	72.41%	88.33%	62.22%	96.02%
42	96.32%	93.75%	85.00%	82.28%	74.26%	84.63%	61.67%	99.54%
43	97.08%	94.72%	77.22%	83.17%	73.70%	83.43%	60.28%	98.80%
44	91.32%	96.11%	63.33%	87.56%	87.50%	91.76%	68.06%	98.24%
45	77.71%	95.97%	64.44%	78.11%	73.15%	79.72%	66.39%	92.41%
46	70.35%	96.81%	65.56%	74.39%	74.63%	75.37%	65.83%	81.76%
47	74.86%	88.75%	68.06%	66.61%	67.04%	63.43%	82.78%	90.19%
48	78.75%	88.89%	58.89%	83.94%	87.13%	94.35%	80.28%	85.65%
49	90.49%	90.14%	68.06%	81.67%	83.61%	91.11%	81.94%	91.67%
50	95.83%	90.28%	86.67%	82.89%	84.35%	91.20%	93.33%	96.85%
51	89.17%	91.25%	68.61%	89.83%	94.91%	97.13%	90.83%	91.76%
52	75.56%	88.06%	54.17%	78.78%	84.07%	87.69%	79.44%	78.80%

	ID9	ID10	ID11	ID12	ID13	ID14	ID15	ID16
53	69.03%	89.31%	54.44%	76.50%	80.56%	72.31%	93.06%	77.87%
54	84.03%	77.22%	73.33%	68.67%	65.46%	70.65%	87.50%	84.07%
55	75.56%	84.58%	64.44%	74.00%	76.76%	90.65%	79.44%	86.30%
56	90.69%	83.33%	83.61%	72.17%	75.28%	87.87%	89.17%	93.98%
57	93.82%	83.61%	68.89%	70.89%	76.30%	87.13%	90.00%	96.57%
58	87.22%	88.19%	76.39%	85.33%	93.06%	94.81%	86.94%	92.87%
59	76.74%	83.06%	69.44%	67.17%	73.24%	89.26%	79.17%	77.04%
60	79.31%	83.47%	75.83%	71.39%	71.39%	85.83%	85.83%	83.15%
61	77.99%	75.56%	74.72%	70.33%	66.67%	81.11%	78.89%	84.54%
62	75.83%	84.72%	74.44%	68.89%	63.80%	92.78%	83.89%	94.07%
63	76.88%	86.53%	76.39%	70.33%	70.19%	91.67%	83.89%	94.44%
64	78.13%	82.92%	78.33%	68.17%	62.69%	92.50%	88.61%	94.07%
65	86.32%	88.61%	85.56%	78.00%	82.22%	97.41%	87.22%	96.94%
66	69.93%	82.50%	86.67%	67.83%	63.98%	93.61%	76.67%	82.96%
67	80.07%	81.25%	69.17%	68.61%	66.57%	89.81%	81.11%	92.22%

Table 14: Individual specificity of each feature with single window validation (Patient 9 to 16)

Appendix C Case Study Table

	Subset					Training set: choice			Testing set: validation			Relative error			
ID6	1	2	3	4	5	Specificity	Latency[s]	FalseAlarm Rate[#/day]	Specificity	Latency[s]	False Alarm Rate[#/day]	Specificity	Latency[s]	False Alarm Rate[#/day]	Result
Lasso	55					99.86%	12.50	87.4	99.72%	12.50	167.0	0.14%	0.00%	90.92%	SUCCESS
Lasso	66					99.03%	13.50	612.1	99.72%	10.00	167.0	0.70%	25.93%	72.73%	SUCCESS
Lasso	41					99.44%	10.50	349.8	99.72%	9.50	167.0	0.28%	9.52%	52.27%	SUCCESS
Lasso	18					99.31%	8.50	437.2	96.81%	9.00	1920.0	2.52%	5.88%	339.11%	SUCCESS
Lasso	60					99.58%	9.50	262.3	99.86%	12.00	83.5	0.28%	26.32%	68.18%	SUCCESS
Lasso	17					99.58%	13.00	262.3	96.39%	12.00	2170.4	3.21%	7.69%	727.31%	SUCCESS
Lasso	55	41				99.86%	12.50	87.4	100.00%	12.50	0.0	0.14%	0.00%	100.00%	SUCCESS
Lasso	55	18				100.00%	14.50	0.0	100.00%	12.50	0.0	0.00%	13.79%	-	SUCCESS
Lasso	55	15				100.00%	14.50	0.0	100.00%	12.50	0.0	0.00%	13.79%	-	SUCCESS
Lasso	55	60				99.86%	13.50	87.4	99.86%	12.50	83.5	0.00%	7.41%	4.54%	SUCCESS
Lasso	41	18				100.00%	13.00	0.0	99.86%	11.00	83.5	0.14%	15.38%	-	SUCCESS
Lasso	41	15				100.00%	13.00	0.0	100.00%	11.00	0.0	0.00%	15.38%	-	SUCCESS
Lasso	41	60				99.86%	15.00	87.4	100.00%	12.00	0.0	0.14%	20.00%	100.00%	SUCCESS
Lasso	41	17				100.00%	13.00	0.0	100.00%	12.00	0.0	0.00%	7.69%	-	SUCCESS
Lasso	18	15				100.00%	9.50	0.0	98.47%	9.00	918.3	1.53%	5.26%	-	SUCCESS
Lasso	18	17				100.00%	13.00	0.0	99.03%	12.00	584.3	0.97%	7.69%	-	SUCCESS
Lasso	15	17				99.86%	13.00	87.4	98.19%	12.00	1085.2	1.67%	7.69%	1140.97%	SUCCESS
Lasso	55	41	18			100.00%	14.50	0.0	100.00%	12.50	0.0	0.00%	13.79%	-	SUCCESS
Lasso	55	41	15			100.00%	14.50	0.0	100.00%	12.50	0.0	0.00%	13.79%	-	SUCCESS
Lasso	55	18	15			100.00%	14.50	0.0	100.00%	12.50	0.0	0.00%	13.79%	-	SUCCESS
Lasso	41	18	15			100.00%	13.00	0.0	100.00%	11.00	0.0	0.00%	15.38%	-	SUCCESS
Lasso	41	18	17			100.00%	13.00	0.0	100.00%	12.00	0.0	0.00%	7.69%	-	SUCCESS
Lasso	41	15	17			100.00%	13.00	0.0	100.00%	12.00	0.0	0.00%	7.69%	-	SUCCESS
Lasso	18	15	17			100.00%	13.00	0.0	99.17%	12.00	500.9	0.83%	7.69%	-	SUCCESS
Lasso	55	41	18	15		100.00%	14.50	0.0	100.00%	12.50	0.0	0.00%	13.79%	-	SUCCESS
Lasso	41	18	15	17		100.00%	13.00	0.0	100.00%	12.00	0.0	0.00%	7.69%	-	SUCCESS
MRMR	52					99.86%	14.50	87.4	100.00%	11.50	0.0	0.14%	20.69%	100.00%	SUCCESS
MRMR	17					99.58%	13.00	262.3	96.39%	12.00	2170.4	3.21%	7.69%	727.31%	SUCCESS
RF	53					98.47%	8.50	961.9	99.44%	11.50	333.9	0.99%	35.29%	65.29%	SUCCESS
RF	6					99.58%	11.00	262.3	99.86%	11.50	83.5	0.28%	4.55%	68.18%	SUCCESS
RF	52					99.86%	14.50	87.4	100.00%	11.50	0.0	0.14%	20.69%	100.00%	SUCCESS
RF	13					100.00%	13.50	0.0	100.00%	14.00	0.0	0.00%	3.70%	-	SUCCESS
RF	12					100.00%	13.00	0.0	100.00%	10.00	0.0	0.00%	23.08%	-	SUCCESS
RF	18					99.31%	8.50	437.2	96.81%	9.00	1920.0	2.52%	5.88%	339.11%	SUCCESS
RF	53	6				100.00%	11.00	0.0	100.00%	11.50	0.0	0.00%	4.55%	-	SUCCESS
RF	53	52				100.00%	14.50	0.0	100.00%	11.50	0.0	0.00%	20.69%	-	SUCCESS
RF	53	13				100.00%	13.50	0.0	100.00%	14.00	0.0	0.00%	3.70%	-	SUCCESS
RF	53	12				100.00%	13.00	0.0	100.00%	11.50	0.0	0.00%	11.54%	-	SUCCESS
RF	53	18				100.00%	9.50	0.0	100.00%	11.50	0.0	0.00%	21.05%	-	SUCCESS
RF	6	52				99.86%	14.50	87.4	100.00%	11.50	0.0	0.14%	20.69%	100.00%	SUCCESS
RF	6	13				100.00%	13.50	0.0	100.00%	14.00	0.0	0.00%	3.70%	-	SUCCESS
RF	6	12				100.00%	13.00	0.0	100.00%	11.50	0.0	0.00%	11.54%	-	SUCCESS
RF	6	18				100.00%	13.50	0.0	100.00%	11.50	0.0	0.00%	14.81%	-	SUCCESS
RF	52	13				100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	52	18				100.00%	14.50	0.0	100.00%	11.50	0.0	0.00%	20.69%	-	SUCCESS
RF	19	12				100.00%	15.00	0.0	100.00%	10.00	0.0	0.00%	33.33%	-	SUCCESS
RF	19	18				99.86%	9.00	87.4	99.03%	9.00	584.3	0.83%	0.00%	568.21%	SUCCESS
RF	13	12				100.00%	14.00	0.0	100.00%	14.00	0.0	0.00%	0.00%	-	SUCCESS
RF	13	18				100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	53	6	52			100.00%	14.50	0.0	100.00%	11.50	0.0	0.00%	20.69%	-	SUCCESS
RF	53	6	13			100.00%	13.50	0.0	100.00%	14.00	0.0	0.00%	3.70%	-	SUCCESS
RF	53	6	12			100.00%	13.00	0.0	100.00%	11.50	0.0	0.00%	11.54%	-	SUCCESS
RF	53	6	18			100.00%	13.50	0.0	100.00%	11.50	0.0	0.00%	14.81%	-	SUCCESS
RF	53	52	13			100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	53	52	18			100.00%	14.50	0.0	100.00%	11.50	0.0	0.00%	20.69%	-	SUCCESS
RF	53	13	12			100.00%	14.00	0.0	100.00%	14.00	0.0	0.00%	0.00%	-	SUCCESS
RF	53	13	18			100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	6	52	13			100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	6	52	18			100.00%	14.50	0.0	100.00%	11.50	0.0	0.00%	20.69%	-	SUCCESS
RF	6	13	12			100.00%	14.00	0.0	100.00%	14.00	0.0	0.00%	0.00%	-	SUCCESS
RF	6	13	18			100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	52	13	18			100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	53	6	52	13		100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	53	6	52	18		100.00%	14.50	0.0	100.00%	11.50	0.0	0.00%	20.69%	-	SUCCESS
RF	53	6	13	12		100.00%	14.00	0.0	100.00%	14.00	0.0	0.00%	0.00%	-	SUCCESS
RF	53	6	13	18		100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	53	52	13	18		100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	6	52	13	18		100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS
RF	53	6	52	13	18	100.00%	15.00	0.0	100.00%	14.00	0.0	0.00%	6.67%	-	SUCCESS

Table 15: Step 3: Filter specifications - 69 selected subsets that fulfill specifications *Columns 6 to 10 where removed because all the selected subsets have a maximal size of 5. Sensitivity columns are removed because all 100%

Appendix D VHDL code

D.0.1 Top level

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 09.06.2022 11:24:27
-- Design Name:
-- Module Name: top_level - rtl
-- Project Name: Seizure Detection
-- Target Devices:  Virtex VC707 Evaluation Kit
-- Tool Versions: Vivado v2020.2
-- Description: top level of seizure detection project
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity top_level is
  Port (
    clk_pin_p     : in  STD_LOGIC;
    clk_pin_n     : in  STD_LOGIC;
    RSTxRI : in std_logic;
    LCD_RWxSO : out std_logic;
    LCD_ExSO : out std_logic;
    LCD_RSxSO : out std_logic;
    LCD_dataxSO : out std_logic_vector(3 downto 0);
    PushButtonxSI : in std_logic
    );
end top_level;

architecture rtl of top_level is

component clk_wiz_0 is
        Port ( clk_in1_p          : in  std_logic;
               clk_in1_n          : in  std_logic;
               clk_out1           : out std_logic
        );
end component clk_wiz_0;

component detection_top
      port(
    CLKxCI     : in  STD_LOGIC;
    RSTxRI : in std_logic;
    STARTxSI : in std_logic;
```

```
    STOPxSI : in std_logic;
    LCD_RWxSO : out std_logic;
    LCD_ExSO : out std_logic;
    LCD_RSxSO : out std_logic;
    LCD_dataxSO : out std_logic_vector(3 downto 0)
    );
end component;

--clk wiz
signal CLKxC : STD_LOGIC;
signal stopxS : std_logic;
--signal TRAINING_DONExS, END_TRAININGxS : std_logic;

begin

i_clk_wiz_0 : clk_wiz_0
    port map (
        clk_in1_p => clk_pin_p,
        clk_in1_n => clk_pin_n,
        clk_out1 => CLKxC
    );
i_detection_top : detection_top
PORT MAP(
    CLKxCI   => CLKxC,
    RSTxRI  => RSTxRI,
    STARTxSI => PushButtonxSI,
    STOPxSI => stopxS,
    LCD_RWxSO => LCD_RWxSO,
    LCD_ExSO => LCD_ExSO,
    LCD_RSxSO => LCD_RSxSO,
    LCD_dataxSO => LCD_dataxSO
    );
end rtl;
```

D.0.2 Control Unit

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/29/2022 09:48:44 AM
-- Design Name:
-- Module Name: detection_top - rtl
-- Project Name: Seizure Detection
-- Target Devices:  Virtex VC707 Evaluation Kit
-- Tool Versions: Vivado v2020.2
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library work;
use work.pdm_prj_pkg.all;

entity detection_top is
Port(
    CLKxCI    : in  STD_LOGIC;
    RSTxRI : in std_logic;
    STARTxSI : in std_logic;
    STOPxSI : in std_logic;
    LCD_RWxSO : out std_logic;
    LCD_ExSO : out std_logic;
    LCD_RSxSO : out std_logic;
    LCD_dataxSO : out std_logic_vector(3 downto 0)
    );
end detection_top;

architecture rtl of detection_top is
--================================================================================
--SIGNAL DECLARATION
--================================================================================
--control
signal stopxS, stop_dataxS : std_logic;
signal startxS : std_logic;
--clk wiz
signal CLKxC : STD_LOGIC;
```

```
signal NEW_DATAxS : std_logic;
signal DATAxD : STD_LOGIC_VECTOR (16-1 downto 0);
signal ACCELxD : STD_LOGIC_VECTOR (25-1 downto 0);

signal labelxD :std_Logic;
signal new_labelxS : std_logic;
signal coast_readyxS :std_logic;
signal RLEDxS : std_logic;
signal GLEDxS : std_logic;

signal THRESHOLD_COASTxDP, THRESHOLD_COASTxDN, THRESHOLDxD :
        STD_LOGIC_VECTOR(25-1 DOWNTO 0);
signal THRESHOLD_READYxS, START_TRAININGxS : std_logic;
signal last_samplexS : std_logic;
signal label_trainxS : std_logic;
signal CNT_TRAININGxDP, CNT_TRAININGxDN : unsigned(10-1 downto 0);
signal MemLabelxDP, MemLabelxDN : std_logic_vector(3 downto 0);

--CONTROL FSM
  -- Declare Types for FSM
  type Control_fsm_Type is (
    IDLE, TRAINING, TRAINING_DONE, TESTING
  );
  signal STATExDP, STATExDN  : Control_fsm_Type;

--============================================================================
--COMPONENT DECLARATION
--============================================================================

component lcd
      port( clk  : in std_logic;
        rst    : in std_logic;
        SF_D   : out std_logic_vector(11 downto 8);
        LCD_E  : out std_logic;
        LCD_RS : out std_logic;
        LCD_RW : out std_logic;
        labelxDI : in std_logic_vector(3 downto 0));
end component;

component detection_fsm is
  port (
    CLKxCI : in std_logic;
    RSTxRI : in std_logic;

    new_labelxSI : in std_logic;
    labelxDI : in std_logic;
    RLEDxSO : out std_logic;
    GLEDxSO : out std_logic
  );
```

```
end component;

component data_fsm is
Port (
        CLKxCI: in std_logic;
        RSTxRI : in std_logic;
        STARTxSI : in std_logic;
        STOPxSI : in std_logic;
        DATAxDO : out STD_LOGIC_VECTOR(15 DOWNTO 0); --FIFO MUX output
        RD_VALIDxSO : out STD_LOGIC
        );
end component data_fsm;

component coastline is
Port (
       CLKxCI : in STD_LOGIC;
       RSTxRI : in STD_LOGIC;
       COASTxDO : out STD_LOGIC_VECTOR (25-1 downto 0);
       COAST_READYxSO :out std_logic;
       FIFOxDI : in STD_LOGIC_VECTOR (16-1 downto 0);
       ENxSI : in STD_LOGIC --when reading fifo in data_fsm
       );
end component coastline;

component threshold_fsm is
  Port (
    CLKxCI: in std_logic;
    RSTxRI : in std_logic;
    SAMPLExDI : in std_logic_vector (25-1 downto 0);
    NEW_SAMPLExSI :in std_logic;
    LAST_SAMPLExSI : in std_logic;
    START_TRAININGxSI: in std_logic;
    THRESHOLDxDO : out std_logic_vector (25-1 downto 0);
    THRESHOLD_READYxSO: out std_logic;
    LABELXSI: in std_logic
    );
end component threshold_fsm;

begin
--START_TRAININGxS <= '1' when (STARTxS = '1' and TRAINING_DONExS = '0') else '0';
CLKxC <= CLKxCI;
--to modify when several accelerators with a mux
THRESHOLD_COASTxDN <= THRESHOLDxD when STATExDP = TRAINING
else THRESHOLD_COASTxDP;
--================================================================================
--COMPONENT INSTANTIATION
--================================================================================
i_lcd : lcd
PORT MAP(
```

```
    clk  => CLKxC,
    rst => RSTxRI,

    LCD_RW => LCD_RWxSO,
    LCD_E => LCD_ExSO,
    LCD_RS => LCD_RSxSO,
    SF_D => LCD_dataxSO,
    labelxDI => MemlabelxDP
    );

i_detection_fsm : detection_fsm
PORT MAP(
    CLKxCI  => CLKxC,
    RSTxRI => RSTxRI,
    new_labelxSI => new_labelxS,
    labelxDI => labelxD,
    RLEDxSO => RLEDxS,
    GLEDxSO => GLEDxS
);

i_data_fsm : data_fsm port map ( CLKxCI => CLKxCI,
                                 RSTxRI => RSTXRI,
                                 STARTxSI => startxS,
                                 STOPxSI => stop_dataxS,
                                 RD_VALIDxSO => NEW_DATAxS,
                                 DATAxDO => DATAxD
                                 );
i_coastline : coastline port map ( CLKxCI => CLKxCI,
                                   RSTxRI => RSTXRI,
                                   COASTxDO => ACCELxD,
                                   COAST_READYxSO => coast_readyxS,
                                   FIFOxDI =>DATAxD,
                                   ENxSI => NEW_DATAxS
                                   );

i_threshold_fsm : threshold_fsm port map(
        CLKxCI => CLKxCI,
        RSTxRI => RSTXRI,
        SAMPLExDI => ACCELxD,
        NEW_SAMPLExSI => coast_readyxS,
        LAST_SAMPLExSI => LAST_SAMPLExS,
        START_TRAININGxSI =>  START_TRAININGxS,
        THRESHOLDxDO => THRESHOLDxD,
        THRESHOLD_READYxSO => THRESHOLD_READYxS,
        LABELXSI => label_trainxS
    );

-- Clck process
PROCESS (CLKxC, RSTxRI)
```

```
BEGIN
IF RSTXRI = '1' THEN
    MemLabelxDP <= (others => '0');
ELSIF CLKxC'EVENT AND CLKxC = '1' THEN
    MemLabelxDP <= MemLabelxDN;
END IF;
END PROCESS;

-- CNT windows used for training to generate stop when reached max
PROCESS (CLKxC, RSTxRI)
BEGIN
IF RSTXRI = '1' THEN
    CNT_TRAININGxDP <= (others => '0');
    THRESHOLD_COASTxDP <= (others => '0');
ELSIF CLKxC'EVENT AND CLKxC = '1' THEN
    CNT_TRAININGxDP <= CNT_TRAININGxDN;
    THRESHOLD_COASTxDP <= THRESHOLD_COASTxDN;
END IF;
END PROCESS;
--First window is numbered as 1 because coast_readyxS is
--high one cycle before label is sent to threshold
CNT_TRAININGxDN <= (others => '0') when startxS = '1' else CNT_TRAININGxDP + 1
                            when coast_readyxS = '1' else CNT_TRAININGxDP;

LAST_SAMPLExS <= '1' when (CNT_TRAININGxDP = (TRAINING_WIDTH - 1)) else '0'; --TO TUNE

--GENRERATE LABEL TRAIN - TO UPDATE WITH TRAINING SET INFORMATIONS
label_trainxS <= '0' when CNT_TRAININGxDP < 31 or CNT_TRAININGxDP > 31 + 89 - 1
else '1';
--Updtate memory with label
MemLabelxDN <= (labelxD & MemLabelxDP(3 downto 1)) when new_labelxS = '1'
else MemLabelxDP; --TO UPDATE
--detection_fsm signals input assignement
labelxD <= '1' when (ACCELxD > THRESHOLD_COASTxDP) else '0';
--new_labelxS <= coast_readyxS when TRAINING_DONExS ='1' else '0';
--training done depends on State of controller
new_labelxS <= coast_readyxS when STATExDP = TESTING else '0';
startxS <= STARTxSI;
stopxS <= STOPxSI;


--CONTROL FSM
PROCESS (CLKxC, RSTxRI)
BEGIN
IF RSTXRI = '1' THEN
    STATExDP <= IDLE;
ELSIF CLKxC'EVENT AND CLKxC = '1' THEN
    STATExDP <= STATExDN;
END IF;
```

```
END PROCESS;

-- FSM
p_FSMControl: process (all) is
  begin
    STATExDN   <= STATExDP;
    START_TRAININGxS <= '0';
    stop_dataxS <= '0';
case STATExDP is
      -- Initial state
    when IDLE =>
        if STARTxS = '1' then
            STATExDN <= TRAINING;
            START_TRAININGxS <= '1';
        end if;
    when TRAINING =>
        if THRESHOLD_READYxS = '1' then
            STATExDN <= TRAINING_DONE;
        end if;
    when TRAINING_DONE =>
        stop_dataxS <= '1';
        if STARTxS = '1' then
            STATExDN <= TESTING;
        end if;
    when TESTING =>
        if stopxS = '1' then
            STATExDN <= TRAINING_DONE;
        end if;
     when others => NULL;
    end case;
  end process p_FSMControl;
end rtl;
```

D.0.3 data_fsm.vhd

```
----------------------------------------------------------------------------------
-- Engineer: Salma Chatagny
--
-- Create Date: 06/20/2022 10:23:06 PM
-- Design Name: Buffer with 2 FIFOS
-- Module Name: data_fsm - rtl
-- Project Name: eizure Detection
-- Target Devices:  Virtex VC707 Evaluation Kit
-- Tool Versions: Vivado v2020.2
-- Description: This block controls in alterance the Writing and the Reading
-- of 2 FIFOS blocks of size 512x16. When one FIFO is full, it is then read while the
-- second FIFO is written into and inversely. Written samples are directly extracted
-- from the memory.
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library work;
use work.pdm_prj_pkg.all;

entity data_fsm is
  Port (
    CLKxCI: in std_logic;
    RSTxRI : in std_logic;
    STARTxSI : in std_logic;
    STOPxSI : in std_logic;
    DATAxDO : out STD_LOGIC_VECTOR(15 DOWNTO 0); --FIFO MUX output
    RD_VALIDxSO : out STD_LOGIC
    );
end data_fsm;

architecture rtl of data_fsm is
--================================================================================
--SIGNAL DECLARATION
--================================================================================
signal ADDRAxD : STD_LOGIC_VECTOR(ADDR_SIZE -1 DOWNTO 0); --TO TUNE
signal DINAxD : STD_LOGIC_VECTOR(N_BITS - 1 DOWNTO 0);
signal DOUTAxD : STD_LOGIC_VECTOR(N_BITS - 1 DOWNTO 0);
signal ENAxS : std_logic;
signal WEAxS : STD_LOGIC_VECTOR(0 DOWNTO 0);

signal CNTxDP, CNTxDN : unsigned(ADDR_SIZE -1 downto 0);
```

```
--FIFO
signal RST_FIFOxR: STD_LOGIC;
signal DIN_F1xD, DIN_F2xD: STD_LOGIC_VECTOR(N_BITS-1 DOWNTO 0);
signal WR_EN_F1xS, WR_EN_F2xS: STD_LOGIC;
signal RD_EN_F1xS, RD_EN_F2xS: STD_LOGIC;
signal DOUT_F1xD,DOUT_F2xD: STD_LOGIC_VECTOR(15 DOWNTO 0);
signal FULL_F1xS, FULL_F2xS:  STD_LOGIC;
signal EMPTY_F1xS, EMPTY_F2xS, valid_F1xS, valid_F2xS:  STD_LOGIC;

--Data Count
signal NewWindowxS : STD_LOGIC;
signal WindowCountxDP, WindowCountxDN :unsigned(ADDR_SIZE -1 downto 0); --5?

--FIFO FSM
TYPE state_type IS(IDLE, INIT_MEM_READY, WR_F1_MEM_READY, WR_F2_MEM_READY, WR1,
                   WR2, WR1_RD2, WR2_RD1,RD1, RD2, LAST_F1, LAST_F2,
                   OVER,WAIT_BEFORE_LAST_F1, WAIT_BEFORE_LAST_F2);
signal STATExDP, STATExDN : state_type;

--Algorithm control signals
signal startxS, stopxS : std_logic;
signal EmptyMemxS: STD_LOGIC;

--================================================================================
--COMPONENT DECLARATION
--================================================================================

    component blk_mem_gen_0 --single port memory
      PORT (
        clka : IN STD_LOGIC;
        ena : IN STD_LOGIC;
        wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
        addra : IN STD_LOGIC_VECTOR(ADDR_SIZE-1 DOWNTO 0);
        dina : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
        douta : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
      );
    end component blk_mem_gen_0;

COMPONENT fifo_generator_0
  PORT (
    clk : IN STD_LOGIC;
    srst : IN STD_LOGIC;
    din : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
    wr_en : IN STD_LOGIC;
    rd_en : IN STD_LOGIC;
    dout : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
    full : OUT STD_LOGIC;
    empty : OUT STD_LOGIC;
```

```
    valid : OUT STD_LOGIC
  );
END COMPONENT;
COMPONENT fifo_generator_1
  PORT (
    clk : IN STD_LOGIC;
    srst : IN STD_LOGIC;
    din : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
    wr_en : IN STD_LOGIC;
    rd_en : IN STD_LOGIC;
    dout : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
    full : OUT STD_LOGIC;
    empty : OUT STD_LOGIC;
    valid : OUT STD_LOGIC
  );
END COMPONENT;

begin

--============================================================================
--COMPONENT INSTANTIATION
--============================================================================

i_blk_mem_gen_0 : blk_mem_gen_0
  PORT MAP (
    clka => CLKxCI,
    ena => ENAxS,
    wea => WEAxS,
    addra => ADDRAxD,
    dina => DINAxD,
    douta => DOUTAxD
  );
i_fifo_generator_0 : fifo_generator_0
  PORT MAP (
    srst => RST_FIFOxR, --asynchronous active low reset
    clk => CLKxCI,
    din => DIN_F1xD,
    wr_en => WR_EN_F1xS,
    rd_en => RD_EN_F1xS,
    dout => DOUT_F1xD,
    full => FULL_F1xS,
    empty => EMPTY_F1xS,
    valid => valid_F1xS
  );

  i_fifo_generator_1 : fifo_generator_1
  PORT MAP (
    srst => RST_FIFOxR, --asynchronous active low reset
    clk => CLKxCI,
```

```
    din => DIN_F2xD,
    wr_en => WR_EN_F2xS,
    rd_en => RD_EN_F2xS,
    dout => DOUT_F2xD,
    full => FULL_F2xS,
    empty => EMPTY_F2xS,
    valid => valid_F2xS
  );
--============================================================================
--ARCHITECTURE
--============================================================================
--Control signals
WEAxS <= "0"; --The memory is read only in this case
RST_FIFOxR <=  '1' when (RSTxRI = '1' or STOPxS = '1') else '0';
startxS <= STARTxSI;
stopxS <= STOPxSI;

-- No value to write into memory (avoid unkonwn state)
DINAxD <= (OTHERS => '0');

-- Clck process
PROCESS (CLKxCI, RSTxRI, STATExDP)
BEGIN
IF RSTXRI = '1' or STATExDP = IDLE THEN
    CNTxDP <= (others => '0');
ELSIF CLKxCI'EVENT AND CLKxCI = '1' THEN
    if NewWindowxS = '1' then
    CNTxDP <= WindowCountxDN;
    else
    CNTxDP <= CNTxDN;
    end if;
END IF;
END PROCESS;

-- WindowCount (for address generation)
PROCESS (CLKxCI, RSTxRI, STATExDP)
BEGIN
IF RSTXRI = '1' or STATExDP = IDLE THEN
    WindowCountxDP <= to_unsigned(0, WindowCountxDP'length);
ELSIF CLKxCI'EVENT AND CLKxCI = '1' THEN
    WindowCountxDP <= WindowCountxDN;
END IF;
END PROCESS;

--FSM
PROCESS (CLKxCI, RSTxRI, stopxS)
BEGIN
IF RSTXRI = '1' or stopxS = '1' THEN
    STATExDP <= IDLE;
```

```
ELSIF CLKxCI'EVENT AND CLKxCI = '1' THEN
    STATExDP <= STATExDN;
END IF;
END PROCESS;

process(all) is
begin
NewWindowxS <= '0';
STATExDN <= STATExDP;
case STATExDP is
--Initial state
    when IDLE =>
        if startxS = '1' then
            STATExDN <= INIT_MEM_READY;
        end if;
    when INIT_MEM_READY =>
        STATExDN <= WR1;
    when WR1 =>
        if Full_F1xS = '1' then
            STATExDN <= WR_F2_MEM_READY;
            NewWindowxS <= '1';
        end if;
    when WR_F2_MEM_READY =>
        STATExDN <= WR2_RD1;
    when WR2_RD1 =>
        if EmptyMemxS = '1' then
            STATExDN <= WAIT_BEFORE_LAST_F2;
            NewWindowxS <= '1';
        elsif Empty_F1xS = '1' then
            if Full_F2xS = '1' then
                STATExDN <= WR_F1_MEM_READY;
                NewWindowxS <= '1';
            else
                STATExDN <= WR2;
            end if;
        elsif Full_F2xS = '1' then
               STATExDN <= RD1;
        end if;
    when RD1 =>
        if Empty_F1xS = '1' then
            STATExDN <= WR_F1_MEM_READY;
            NewWindowxS <= '1';
        end if;
    when WR_F1_MEM_READY =>
        STATExDN <= WR1_RD2;
    when WR1_RD2 =>
        if EmptyMemxS = '1' then
            STATExDN <= WAIT_BEFORE_LAST_F1;
            NewWindowxS <= '1';
```

```
        elsif Full_F1xS = '1' then
            if Empty_F2xS = '1' then
                STATExDN <= WR_F2_MEM_READY;
                NewWindowxS <= '1';
            else
                STATExDN <= RD2;
            end if;
        elsif Empty_F2xS = '1' then
               STATExDN <= WR1;
        end if;
    when RD2 =>
        if Empty_F2xS = '1' then
            STATExDN <= WR_F2_MEM_READY;
            NewWindowxS <= '1';
        end if;
    when WAIT_BEFORE_LAST_F1 =>
        STATExDN <= LAST_F1;
    when WAIT_BEFORE_LAST_F2 =>
        STATExDN <= LAST_F2;
    when LAST_F1 =>
        if Empty_F1xS = '1' then
            STATExDN <= OVER;
         end if;
    when LAST_F2 =>
        if Empty_F2xS = '1' then
            STATExDN <= OVER;
         end if;
    when OVER =>
        STATExDN <= IDLE;
    when OTHERS =>
        STATExDN <= IDLE;
end case;
end process;

--Memory control signals update: Enable reading from memory
ENAxS <= '1' when STATExDP = INIT_MEM_READY or STATExDP = WR_F1_MEM_READY
             or STATExDP = WR_F2_MEM_READY or STATExDP = WR1 or
             STATExDP = WR2 or STATExDP = WR1_RD2 or STATExDP = WR2_RD1
             or STATExDP = LAST_F1 or STATExDP = LAST_F2 else '0';

--Counter for memory adressing
CNTxDN <= CNTxDP + 1;
ADDRAxD <= std_logic_vector(CNTxDP);

WindowCountxDN <= WindowCountxDP + SAMPLE_NUM when NewWindowxS = '1'
                  else WindowCountxDP; --TO TUNE

--FIFO control signals update
WR_EN_F1xS <= '1' when STATExDP = WR1_RD2 or STATExDP = WR1 else '0';
```

```
RD_EN_F1xS <= '1' when STATExDP = WR2_RD1 or STATExDP = RD1
                or STATExDP = LAST_F1 else '0';
WR_EN_F2xS <= '1' when STATExDP = WR2_RD1 or STATExDP = WR2 else '0';
RD_EN_F2xS <= '1' when STATExDP = WR1_RD2 or STATExDP = RD2
                or STATExDP = LAST_F2 else '0';

--Write memory output into FIFO input
DIN_F1xD <=  DOUTAxD;
DIN_F2xD <=  DOUTAxD;

--Flag update -TO UPDATE
EmptyMemxS <= '1' when CNTxDP = 0 and (STATExDP /= (IDLE)
                                  and STATExDP /= (INIT_MEM_READY)) else '0';

--Output assignement
DATAxDO <=  DOUT_F1xD when STATExDP = RD1 or STATExDP = WR2_RD1 or STATExDP = LAST_F1
       else DOUT_F2xD when STATExDP = RD2 or STATExDP = WR1_RD2 or STATExDP = LAST_F2
       else (OTHERS => '0');
RD_VALIDxSO <= valid_F1xS or valid_F2xS; --output value of the FIFO is valid

end rtl;
```

D.0.4 Coastline.vhd

```
----------------------------------------------------------------------------------
-- Engineer: Salma Chatagny
--
-- Create Date: 13.06.2022 13:59:52
-- Design Name: coastline - rtl
-- Module Name: coastline_accelerator
-- Project Name: Seizure Detection
-- Target Devices: Virtex VC707 Evaluation Kit
-- Tool Versions: Vivado v2020.2
-- Description: This block (when enabled) computes the coastline value of the 512
-- consecutive samples provided at the input at the system clck frequency (50Mhz)
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
library work;
use work.pdm_prj_pkg.all;

entity coastline is
    Port ( CLKxCI : in STD_LOGIC;
           RSTxRI : in STD_LOGIC;
           COASTxDO : out STD_LOGIC_VECTOR (N_BITS_COAST -1 downto 0);
           COAST_READYxSO :out std_logic;
           FIFOxDI : in STD_LOGIC_VECTOR (N_BITS - 1 downto 0);
           ENxSI : in STD_LOGIC);
end coastline;

--============================================================================
-- ARCHITECTURE DECLARATION
--============================================================================
```

```
architecture rtl of coastline is
--================================================================================
-- SIGNALS DECLARATION
--================================================================================
signal DataxDN, DataxDP, SubxS: unsigned(N_BITS - 1 downto 0);
signal SumxS, CoastxDN, CoastxDP: unsigned(N_BITS_COAST -1 downto 0);
-- CNT should reach N_sample (16)(one extra bit) -->count to 512 later TO TUNE
signal CNTxDP, CNTxDN : unsigned(CNT_SAMPLE downto 0);
signal CoastRSTxS, EndWindowxS, WindowReadyxS: std_logic;

begin

--Input assginements
WindowReadyxS <= ENxSI;
DataxDN <= unsigned(FIFOxDI);

--Control assignement
EndWindowxS <= '1' when CNTxDP = SAMPLE_NUM - 1 else '0'; --TO TUNE
CoastRSTxS <= '1' when CNTxDP = 0 else '0';

-- Clck process
PROCESS (CLKxCI, RSTxRI, CoastRSTxS)
BEGIN
IF RSTXRI = '1' THEN
    CNTxDP <= (others => '0');
    DataxDP <= (others => '0');
    CoastxDP <= (others => '0');
ELSIF CLKxCI'EVENT AND CLKxCI = '1' THEN
    CNTxDP <= CNTxDN;
    DataxDP <= DataxDN;
    CoastxDP <= CoastxDN when CoastRSTxS = '0' else (OTHERS => '0');
END IF;
END PROCESS;

--Counter: Counts when the accelerator block is enabled
CNTxDN <= (OTHERS => '0') when WindowReadyxS  = '0' or CNTxDP = SAMPLE_NUM
                          else CNTxDP + 1; --TO TUNE

--Substraction and absolute value
PROCESS(all)
BEGIN
IF DataxDP > DataxDN THEN
    SubxS <= DataxDP - DataxDN;
ELSE
    SubxS <= DataxDN - DataxDP;
END IF;
END PROCESS;

-- Addition
```

```
SumxS <= resize(SubxS,25) + CoastxDP;

CoastxDN <= SumxS;
--=============================================================================
-- OUTPUT ASSIGNEMENT
--=============================================================================
COASTxDO <= std_logic_vector(CoastxDP);

COAST_READYxSO <= EndWindowxS;
end rtl;
```

D.0.5 Classifier Training

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 08.07.2022 14:56:33
-- Design Name:
-- Module Name: threshold_fsm - rtl
-- Project Name: Seizure Detection
-- Target Devices: Virtex VC707 Evaluation Kit
-- Tool Versions: Vivado v2020.2
-- Description: This code computes the threshold for an accelerator block
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity threshold_fsm is
  Port (
    CLKxCI: in std_logic;
    RSTxRI : in std_logic;
    SAMPLExDI : in std_logic_vector (25-1 downto 0);
    NEW_SAMPLExSI :in std_logic;
    LAST_SAMPLExSI : in std_logic;
    START_TRAININGxSI: in std_logic;
    THRESHOLDxDO : out std_logic_vector (25-1 downto 0);
    THRESHOLD_READYxSO: out std_logic;
    LABELXSI: in std_logic
    );
end threshold_fsm;

architecture rtl of threshold_fsm is

--============================================================================
--SIGNAL DECLARATION
--============================================================================
  ---------------------------------------------------------------------------
  -- Divider input signals
  ---------------------------------------------------------------------------

  -- Slave channel DIVIDEND inputs
```

```
  signal s_axis_dividend_tvalid    : std_logic := '0';  -- TVALID for channel A
  signal s_axis_dividend_tdata     : std_logic_vector(31 downto 0)
                                   := (others => 'X');  -- TDATA for channel A

  -- Slave channel DIVISOR inputs
  signal s_axis_divisor_tvalid    : std_logic := '0';  -- TVALID for channel B
  signal s_axis_divisor_tdata     : std_logic_vector(15 downto 0)
                                  := (others => 'X');  -- TDATA for channel B
  signal quotient : std_logic_vector(24 downto 0) := (others => '0');
  ------------------------------------------------------------------------
  -- Divider output signals
  ------------------------------------------------------------------------

  -- Master channel DOUT outputs
  signal m_axis_dout_tvalid : std_logic := '0';  -- TVALID for channel DOUT
  signal m_axis_dout_tdata  : std_logic_vector(47 downto 0) := (others => '0');
                              -- TDATA for channel DOUT
------------------------------------------------------------------------
signal MIN_VALxDP : STD_LOGIC_VECTOR(25-1 DOWNTO 0):= (others => '1');
signal MIN_VALxDN : STD_LOGIC_VECTOR(25-1 DOWNTO 0);
signal MAX_VALxDN : STD_LOGIC_VECTOR(25-1 DOWNTO 0);
signal MAX_VALxDP : STD_LOGIC_VECTOR(25-1 DOWNTO 0):= (others => '0');
signal THRESHOLDxDP, THRESHOLDxDN : STD_LOGIC_VECTOR(26-1 DOWNTO 0);

signal CNTxDP, CNTxDN : unsigned(9-1 downto 0); --depends of number of window
signal SUMxDP, SUMxDN : unsigned(32-1 downto 0);
signal NEW_SUMxS : unsigned(32-1 downto 0);

signal NEW_ELEMENTxS: STD_LOGIC;

--Control signal
signal start_trainingxS, new_samplexS, last_samplexS,overlapxS,
       Threshold_readyxS, labelxS: std_logic;
--FSM
TYPE state_type is (IDLE, WAIT_FOR_SAMPLE, SAVE_SAMPLE, TEST, SUM1, SUM2,
                    DIVIDE1, DIVIDE2,THRESHOLD_DONE);
signal STATExDP, STATExDN : state_type;

--FIFO
signal RST_FIFOxR: STD_LOGIC;
signal DIN_F1xD, DIN_F2xD: STD_LOGIC_VECTOR(25-1 DOWNTO 0);
signal WR_EN_F1xS, WR_EN_F2xS: STD_LOGIC;
signal RD_EN_F1xS, RD_EN_F2xS: STD_LOGIC;
signal DOUT_F1xD,DOUT_F2xD: STD_LOGIC_VECTOR(25-1 DOWNTO 0);
signal FULL_F1xS, FULL_F2xS:  STD_LOGIC;
signal EMPTY_F1xS, EMPTY_F2xS:  STD_LOGIC;
signal valid_F1xS,valid_F2xS:  STD_LOGIC;

--================================================================================
```

```
--COMPONENT DECLARATION
--============================================================================
COMPONENT div_gen_0
  PORT (
    aclk : IN STD_LOGIC;
    s_axis_divisor_tvalid : IN STD_LOGIC;
    s_axis_divisor_tdata : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
    s_axis_dividend_tvalid : IN STD_LOGIC;
    s_axis_dividend_tdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    m_axis_dout_tvalid : OUT STD_LOGIC;
    m_axis_dout_tdata : OUT STD_LOGIC_VECTOR(47 DOWNTO 0)
  );
END COMPONENT;
COMPONENT fifo_generator_2
  PORT (
    clk : IN STD_LOGIC;
    srst : IN STD_LOGIC;
    din : IN STD_LOGIC_VECTOR(25-1 DOWNTO 0);
    wr_en : IN STD_LOGIC;
    rd_en : IN STD_LOGIC;
    dout : OUT STD_LOGIC_VECTOR(25-1 DOWNTO 0);
    full : OUT STD_LOGIC;
    empty : OUT STD_LOGIC;
    valid : OUT STD_LOGIC
  );
END COMPONENT;
COMPONENT fifo_generator_3
  PORT (
    clk : IN STD_LOGIC;
    srst : IN STD_LOGIC;
    din : IN STD_LOGIC_VECTOR(25-1 DOWNTO 0);
    wr_en : IN STD_LOGIC;
    rd_en : IN STD_LOGIC;
    dout : OUT STD_LOGIC_VECTOR(25-1 DOWNTO 0);
    full : OUT STD_LOGIC;
    empty : OUT STD_LOGIC;
    valid : OUT STD_LOGIC
  );
END COMPONENT;
begin

--============================================================================
--COMPONENT INSTANTIATION
--============================================================================

i_div_gen_0 : div_gen_0
  PORT MAP (
    aclk => CLKxCI,
    s_axis_divisor_tvalid => s_axis_divisor_tvalid,
```

```
    s_axis_divisor_tdata => s_axis_divisor_tdata,
    s_axis_dividend_tvalid => s_axis_dividend_tvalid,
    s_axis_dividend_tdata => s_axis_dividend_tdata,
    m_axis_dout_tvalid => m_axis_dout_tvalid,
    m_axis_dout_tdata => m_axis_dout_tdata
    );
i_fifo_generator_2 : fifo_generator_2
  PORT MAP (
    srst => RST_FIFOxR, --asynchronous active low reset
    clk => CLKxCI,
    din => DIN_F1xD,
    wr_en => WR_EN_F1xS,
    rd_en => RD_EN_F1xS,
    dout => DOUT_F1xD,
    full => FULL_F1xS,
    empty => EMPTY_F1xS,
    valid => valid_F1xS
  );

  i_fifo_generator_3 : fifo_generator_3
  PORT MAP (
    srst => RST_FIFOxR, --asynchronous active low reset
    clk => CLKxCI,
    din => DIN_F2xD,
    wr_en => WR_EN_F2xS,
    rd_en => RD_EN_F2xS,
    dout => DOUT_F2xD,
    full => FULL_F2xS,
    empty => EMPTY_F2xS,
    valid => valid_F2xS
  );
--Assign input signals
start_trainingxS <= START_TRAININGxSI;
labelxS <= LABELxSI;
new_samplexS <= NEW_SAMPLExSI;
last_samplexS <= LAST_SAMPLExSI;
--Control signals
overlapxS <= '1' when (MIN_VALxDP < MAX_VALxDP) else '0';

-- Clck process
PROCESS (CLKxCI, RSTxRI)
BEGIN
IF RSTXRI = '1' THEN
    STATExDP <= IDLE;
ELSIF CLKxCI'EVENT AND CLKxCI = '1' THEN
    STATExDP <= STATExDN;
END IF;
END PROCESS;
```

```
process(all) is
begin
Threshold_readyxS <= '0';
s_axis_divisor_tvalid <= '0';
s_axis_dividend_tvalid <= '0';
WR_EN_F1xS <= '0';
WR_EN_F2xS <= '0';
RD_EN_F1xS <= '0';
RD_EN_F2xS <= '0';
STATExDN <= STATExDP;
case STATExDP is
--Initial state
    when IDLE =>
        if start_trainingxS = '1' then
            STATExDN <= WAIT_FOR_SAMPLE;
        end if;
    when WAIT_FOR_SAMPLE=>
        if new_samplexS = '1' then
            STATExDN <= SAVE_SAMPLE;
        end if;
    when SAVE_SAMPLE =>
        if labelxS = '0' then
            WR_EN_F1xS <= '1';
        else
            WR_EN_F2xS <= '1';
        end if;
        if last_samplexS = '1' then
            STATExDN <= TEST;
        else
            STATExDN <= WAIT_FOR_SAMPLE;
        end if;
    when TEST =>
        if overlapxS = '1' then
            STATExDN <= SUM1;
        else
            STATExDN <= THRESHOLD_DONE;
        end if;
    when SUM1 =>
    RD_EN_F1xS <= '1';
        if empty_F1xS = '1' then
            s_axis_divisor_tvalid <= '1';
            s_axis_dividend_tvalid <= '1';
            STATExDN <= DIVIDE1;
        end if;
    when SUM2 =>
    RD_EN_F2xS <= '1';
        if empty_F2xS = '1' then
            s_axis_divisor_tvalid <= '1';
            s_axis_dividend_tvalid <= '1';
```

```
            STATExDN <= DIVIDE2;
        end if;
    when DIVIDE1 =>
        --s_axis_divisor_tvalid <= '1';
        --s_axis_dividend_tvalid <= '1';
        if m_axis_dout_tvalid  = '1' then
            STATExDN <= SUM2;
        end if;
    when DIVIDE2 =>
        --s_axis_divisor_tvalid <= '1';
        --s_axis_dividend_tvalid <= '1';
         if m_axis_dout_tvalid  = '1' then --division is done
            STATExDN <= THRESHOLD_DONE;
          end if;
    when THRESHOLD_DONE =>
        Threshold_readyxS <= '1';
        STATExDN <= IDLE;
    when OTHERS =>
        STATExDN <= IDLE;
end case;
end process;

--SUM
NEW_SUMxS <= SUMxDP + unsigned(DOUT_F1xD) when STATExDP = SUM1
        else SUMxDP + unsigned(DOUT_F2xD);

--MUX (use the same for sum1 and sum2
SUMxDN <= NEW_SUMxS when NEW_ELEMENTxS = '1' else (others => '0')
                    when STATExDP = DIVIDE1 or STATExDP = IDLE else SUMxDP;
CNTxDN <= CNTxDP + 1 when NEW_ELEMENTxS = '1' else (others => '0')
                    when STATExDP = DIVIDE1 or STATExDP = IDLE  else CNTxDP;

NEW_ELEMENTxS <= '1' when (unsigned(DOUT_F1xD) > unsigned(MIN_VALxDP)
                                    and STATExDP = SUM1 and valid_F1xS = '1')
                                    or (unsigned(DOUT_F2xD) < unsigned(MAX_VALxDP)
                                    and STATExDP = SUM2 and valid_F2xS = '1')
                                    else '0';
s_axis_divisor_tdata(8 downto 0) <= std_logic_vector(CNTxDP);
s_axis_divisor_tdata(15 downto 9) <= (others => '0');
s_axis_dividend_tdata(31 downto 0) <= std_logic_vector(SUMxDP);
quotient  <= m_axis_dout_tdata(40 downto 16);

--THRESHOLD VALUE
THRESHOLDxDN <= std_logic_vector(unsigned('0' & MAX_VALxDP(25-1 downto 0)) +
                unsigned('0' & MIN_VALxDP (25-1 downto 0))) when STATExDP = TEST
                else '0' & quotient (25-1 downto 0) when
                (STATExDP = DIVIDE1 and m_axis_dout_tvalid  = '1')
                else  std_logic_vector(unsigned('0' & quotient(25-1 downto 0))
                + unsigned(THRESHOLDxDP)) when (STATExDP = DIVIDE2 and
```

```
                m_axis_dout_tvalid  = '1')
                else (others => '0') when STATExDP = IDLE
                        else THRESHOLDxDP;

--do division by 2 when assigning threshold to output(shift right)
THRESHOLDxDO <= THRESHOLDxDP(26-1 downto 1);
THRESHOLD_READYxSO <= Threshold_readyxS;
--FIFO
DIN_F1xD <= SAMPLExDI;
DIN_F2xD <= SAMPLExDI;

MAX_VALxDN <= (others => '0') when STATExDP = IDLE else SAMPLExDI
                              when (SAMPLExDI > MAX_VALxDP and WR_EN_F1xS = '1')
                              else MAX_VALxDP;
MIN_VALxDN <= (others => '1') when STATExDP = IDLE else SAMPLExDI
                              when (SAMPLExDI < MIN_VALxDP and WR_EN_F2xS = '1')
                              else MIN_VALxDP;
RST_FIFOxR <= '1' when STATExDP = IDLE else '0';
-- Clck process
PROCESS (CLKxCI, RSTxRI)
BEGIN
IF RSTXRI = '1' then
    MIN_VALxDP <= (others => '1');
    MAX_VALxDP <= (others => '0');
    SUMxDP <= (others => '0');
    CNTxDP <= (others => '0');
    THRESHOLDxDP <= (others => '0');
ELSIF CLKxCI'EVENT AND CLKxCI = '1' THEN
    MIN_VALxDP <= MIN_VALxDN;
    MAX_VALxDP <= MAX_VALxDN;
    SUMxDP <= SUMxDN;
    CNTxDP <= CNTxDN;
    THRESHOLDxDP <= THRESHOLDxDN;
END IF;
END PROCESS;
end rtl;
```

D.0.6 Classifier

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06.06.2022 11:35:25
-- Design Name:
-- Module Name: detection_fsm - rtl
-- Project Name: Seizure Detection
-- Target Devices:  Virtex VC707 Evaluation Kit
-- Tool Versions: Vivado v2020.2
-- Description: Perform classification by comparing the --last three windows comparison res
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity detection_fsm is
  port (
    CLKxCI : in std_logic;
    RSTxRI : in std_logic;

    new_labelxSI : in std_logic;
    labelxDI : in std_logic;
    RLEDxSO : out std_logic;
    GLEDxSO : out std_logic
  );
end detection_fsm;

architecture rtl of detection_fsm is

  -- Declare Types for FSM
  type Detection_fsm_Type is (
    No, Yes1, Yes2, Yes3
  );

  -- FSM
  signal STATExDN, STATExDP : Detection_fsm_Type;

  signal CountCLRxS     : std_logic;
  signal CountENxS      : std_logic;
  signal new_labelxS :std_logic;
```

```
  signal labelxD :std_logic;

begin
new_labelxS <= new_labelxSI;
labelxD <= labelxDI;
-- FSM
p_FSMComb: process (all) is
  begin
    STATExDN   <= STATExDP;
    GLEDxSO    <= '0';  -- default : LED OFF
    RLEDxSO    <= '0';  -- default : LED OFF
    CountCLRxS <= '0';
    CountENxS  <= '0';
case STATExDP is
      -- Initial state
    when No =>
            RLEDxSO <= '1';
            if labelxD = '1'  then
                STATExDN <= Yes1;
            else
                STATExDN <= No;
            end if;
    when Yes1 =>
            RLEDxSO <= '1';
            if labelxDI = '1' then
                STATExDN <= Yes2;
            else
                STATExDN <= No;
            end if;
    when Yes2 =>
            RLEDxSO <= '1';
            if labelxDI = '1' then
                STATExDN <= Yes3;
            else
                STATExDN <= No;
            end if;
    when Yes3 =>
            GLEDxSO <= '1';
            if labelxDI = '1' then
                STATExDN <= Yes3;
            else
                STATExDN <= No;
            end if;
     when others => NULL;
    end case;
  end process p_FSMComb;

  -- FSM Sequential Process
  p_FSMSeq: process (CLKxCI, RSTxRI, new_labelxS) is
```

```
  begin
    if (RSTxRI = '1') then
      STATExDP <= No;
    elsif (CLKxCI'event and CLKxCI = '1' and new_labelxS = '1') then
      STATExDP <= STATExDN;
    end if;
  end process p_FSMSeq;
end architecture rtl;
```

D.0.7 Testbench for detection top

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 29.06.2022 14:13:23
-- Design Name:
-- Module Name: detection_top_tb - tb
-- Project Name: Seizure Detection
-- Target Devices:  Virtex VC707 Evaluation Kit
-- Tool Versions: Vivado v2020.2
-- Description: Testbench for detection top.vhd
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity detection_top_tb is
--  Port ( );
end detection_top_tb;

architecture tb of detection_top_tb is

--================================================================================
-- TYPE AND CONSTANT DECLARATIONS
--================================================================================
  constant CLK_HIGH   : time := 4 ns;
  constant CLK_LOW    : time := 4 ns;
  constant CLK_PERIOD : time := CLK_LOW + CLK_HIGH;
  constant CLK_STIM   : time := 1 ns;
  constant CLK_RESP   : time := CLK_PERIOD - 1 ns;

--================================================================================
-- COMPONENT DECLARATIONS
--================================================================================
  component detection_top is
    Port (
    CLKxCI    : in  STD_LOGIC;
    RSTxRI : in std_logic;
    STARTxSI : in std_logic;
    STOPxSI : in std_logic;
    LCD_RWxSO : out std_logic;
```

```
    LCD_ExSO : out std_logic;
    LCD_RSxSO : out std_logic;
    LCD_dataxSO : out std_logic_vector(3 downto 0)
          );
  end component detection_top ;
--================================================================================
-- SIGNAL DECLARATIONS
--================================================================================
  signal CLKxC : std_logic;
  signal RSTxR : std_logic;
  signal STARTxS, STOPxS : std_logic;
  signal LCD_RWxS, LCD_ExS , LCD_RSxS: std_logic;
  signal LCD_dataxS : std_logic_vector(3 downto 0);

begin

--================================================================================
-- COMPONENT INSTANTIATIONS
--================================================================================

  -- Instantiate dut
  mem_to_coast_1: entity work.detection_top
    port map (
      CLKxCI   => CLKxC,
      RSTxRI   => RSTxR,
      STARTxSI => STARTxS,
      STOPxSI => STOPxS,
      LCD_RWxSO => LCD_RWxS,
      LCD_ExSO => LCD_ExS,
      LCD_RSxSO => LCD_RSxS ,
      LCD_dataxSO => LCD_dataxS
    );

--================================================================================
-- CLOCK PROCESS
-- Process for generating the clock signal
--================================================================================
  p_CLK: process is
  begin
    CLKxC <= '0';
    wait for CLK_LOW;
    CLKxC <= '1';
    wait for CLK_HIGH;
  end process p_CLK;
    --================================================================================
-- RESET PROCESS
-- Process for generating the reset signal
--================================================================================
  p_RST: process is
```

```
  begin
    RSTxR <= '1';
    wait until CLKxC'event and CLKxC = '1'; -- Align to rising-edge
    wait for (2*CLK_PERIOD + CLK_STIM);     -- Wait 2 CC and a little after edge
    RSTxR <= '0';
    wait;
  end process p_RST;
    --================================================================================
-- TEST PROCESSS
--================================================================================
  p_STIM: process is
  begin
  --reset
    STARTxS <= '0';
    STOPxS <= '0';
    wait until CLKxC'event and CLKxC = '1' and RSTxR = '0';
  --test
    wait for 0.5ms;
    wait until CLKxC'event and CLKxC = '1';
    STARTxS <= '1';
    wait for CLK_STIM;
    wait until CLKxC'event and CLKxC = '1';
    STARTxS <= '0';
    --stopxS <= '1';
    wait for 1us;
    wait until CLKxC'event and CLKxC = '1';
    stopxS <= '0';
    STARTxS <= '1';
    wait until CLKxC'event and CLKxC = '1';
    STARTxS <= '0';
    wait for 1 ms;
  end process p_STIM;
  end tb;
```

D.0.8 Constraints file

```
##############################################################################
# CLOCKS
##############################################################################

# SYSCLK 200MHz
set_property IOSTANDARD LVDS [get_ports clk_pin_p]
set_property PACKAGE_PIN E19 [get_ports clk_pin_p]
set_property PACKAGE_PIN E18 [get_ports clk_pin_n]
set_property IOSTANDARD LVDS [get_ports clk_pin_n]

##############################################################################
# LCD Display (2x15 5x8 Dot display) (DisplayTech 162D) (ST7066U Driver)
##############################################################################

set_property PACKAGE_PIN AT42 [get_ports {LCD_dataxSO[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {LCD_dataxSO[0]}]
set_property PACKAGE_PIN AR38 [get_ports {LCD_dataxSO[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {LCD_dataxSO[1]}]
set_property PACKAGE_PIN AR39 [get_ports {LCD_dataxSO[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {LCD_dataxSO[2]}]
set_property PACKAGE_PIN AN40 [get_ports {LCD_dataxSO[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {LCD_dataxSO[3]}]
set_property PACKAGE_PIN AN41 [get_ports LCD_RSxSO]
set_property IOSTANDARD LVCMOS18 [get_ports LCD_RSxSO]
set_property PACKAGE_PIN AR42 [get_ports LCD_RWxSO]
set_property IOSTANDARD LVCMOS18 [get_ports LCD_RWxSO]
set_property PACKAGE_PIN AT40 [get_ports LCD_ExSO]
set_property IOSTANDARD LVCMOS18 [get_ports LCD_ExSO]

# Pushbuttons - center button
set_property PACKAGE_PIN AV39 [get_ports PushButtonxSI]
set_property IOSTANDARD LVCMOS18 [get_ports PushButtonxSI]

set_property IOSTANDARD LVCMOS18 [get_ports RSTxRI]
set_property PACKAGE_PIN AV40 [get_ports RSTxRI]
set_property CONFIG_VOLTAGE 1.8 [current_design]
set_property CFGBVS GND [current_design]
```