



# **A Hardware Implementation of a qEEG-Based Discriminant Function for Brain Injury Detection**

Fotios Kostarelos

Submitted for the award of Master of Engineering

University of Limerick

Supervised by Dr. Brendan Mullane

Joint Supervisor Dr. Ciaran MacNamee

Submitted to the University of Limerick, November 2021



## **Abstract**

### **A Hardware Implementation of a qEEG-Based Discriminant Function for Brain Injury Detection**

This thesis provides a state-of-the-art overview of Electroencephalogram (EEG) and its biomedical applications specific to the development of a biometric signal-processing platform with an important aim for targeting Traumatic Brain Injury (TBI) diagnosis. The outcome of this work is an initial development of a real-time hardware implementation of a system using EEG as a tool for TBI detection.

An injury to the brain tissue is caused by an external force and disrupts the normal function of the brain. The consequences of TBI for society and for each person individually can be immense. A TBI can have physical, cognitive, social emotional and behavioural complications on the patient. Some of these complications might persist for long time after the injury thus making difficult the daily life of people who have experienced it. Notably, 50% of all TBIs are motor vehicle accidents.

Given that motor vehicle accidents occur too often, the importance of a device to be used by first responders is obvious. So, the ultimate goal in terms of broader research is to work towards a multi-modal platform exploiting additional health indicators such as cranial blood pressure, to be used by first responders in cases of emergency, like road accidents and first aid for athletes.

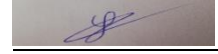
Specifically, this thesis uses the power and cross-power spectrum modalities of EEG signals to extract suitable features relevant to a TBI event. These features which are alternatively called quantitative-EEG (qEEG) variables, are the indicators of an injury in the brain, as they can “expose” even a subtle alteration in the power and the phase content of the signals. The combination of four key qEEG features which constitute the discriminant function are implemented onto hardware and is the key outcome of this work.

A XILINX Zynq UltraScale+ FPGA SoC ZCU104 platform was used as a prototyping board for the hardware implementation of this system. The code for the hardware design was developed using the High-Level-Synthesis (HLS) design flow for each of the four qEEG parameters. Also, this work utilized the Direct Memory Access (DMA) of the ZCU104 board in order to achieve a real-time performance for the discriminant function logic design.



## **Declaration**

I hereby declare that this thesis is entirely my own work and has not been submitted for a degree in part or in full to any other University.

A small rectangular box containing a handwritten signature in blue ink, which appears to be 'F. Kostarelos'.

Fotios Kostarelos



## **Acknowledgements**

I would like to express my heartiest gratitude to my supervisor Dr. Brendan Mullane for giving me the opportunity to pursue this Masters of Engineering at the University of Limerick (UL) and for the continuous support of this research. His motivation, guidance, and immense knowledge helped me to complete my MEng studies. I am also thankful to my joint supervisor Dr Ciaran MacNamee for supporting this work with his insightful comments during the course of this research.

I feel indebted to say thank you to Dr Shantanu Mehta for his overall generous contribution as a colleague but also as a friend too. Needless to say, how gratified I feel for meeting Dr. Vincent O'Brien, his to-the-point advice turned out to be fruitful for my work.

Above all, my sincere and humble appreciation and love to my family for their continuous and unconditional support of my journey in life generally.





## Table of Contents

Abstract.....	ii
Declaration.....	iv
Acknowledgements.....	vi
Table of Contents.....	viii
List of Figures.....	xii
List of Tables .....	xiv
List of Abbreviations .....	xvi

## **Introduction ..... 1**

1.1 Introduction and motivation.....	1
1.2 Project aims and objectives.....	2
1.3 Project outcomes .....	2
1.4 Thesis outline .....	4
1.5 Learning outcomes.....	5

## **EEG and TBI Overview..... 7**

2.1 Introduction.....	7
2.2 What is EEG?.....	7
2.2.1 Biomedical applications of EEG and more.....	8
2.2.2 EEG advantages & disadvantages .....	9
2.2.3 EEG measurement 10-20 System .....	10
2.2.4 EEG Montages.....	13
2.2.5 EEG signal bands.....	15
2.3 Traumatic brain injury symptoms and complications.....	17
2.4 Conclusion .....	18

## **Literature review and qEEG analysis ..... 19**

3.1 Introduction.....	19
3.2 Medical application of EEG-qEEG towards TBI and challenges.....	19
3.3 TBI assessment via qEEG.....	21

3.4 Factors for the evaluation of a medical computer-based method .....	23
3.5 Selecting the discriminant function for TBI .....	28
3.6 Conclusion .....	30

## **Design methodology ..... 33**

4.1 Introduction.....	33
4.2 EEG pre-processing steps .....	33
4.2.1 Bandlimiting .....	34
4.2.2 Artefact rejection .....	38
4.3 Feature extraction methodology .....	43
4.3.1 Fast Fourier Transform vs filter bank .....	43
4.3.2 Welch's method, power and cross power spectrum derivation.....	45
4.4 qEEG features definition.....	48
4.4.1 Relative Power.....	48
4.4.2 Amplitude Asymmetry .....	49
4.4.3 Coherence .....	50
4.4.4 Phase Difference .....	51
4.5 CORDIC algorithm, sine and cosine .....	51
4.6 Conclusion .....	54

## **Implementation..... 55**

5.1 Introduction.....	55
5.2 Data description and data availability issues .....	55
5.3 Design tools .....	56
5.3.1 Vivado HLS .....	56
5.3.2 Vivado IP Integrator .....	57
5.3.3 Vivado SDK.....	58
5.4 XILINX Zynq UltraScale+ MPSoC ZCU 104 .....	59
5.5 AXI protocol .....	61
5.6 DMA .....	63

5.7	MATLAB Code for behavioural modelling .....	65
5.7.1	HLS Code .....	66
5.7.2	SDK code .....	73
5.8	Block design.....	74
5.8.1	Discriminant function design rearrangement .....	75
5.9	Conclusion .....	77

## **Results, conclusions and future work ..... 79**

6.1	Introduction.....	79
6.2	Initial design improvement results.....	79
6.2.1	Baseline FFT code: no improvements .....	80
6.2.2	First improvement: using look-up tables instead of the built-in functions for sine and cosine .....	80
6.2.3	Second improvement: using look-up tables instead of the built-in functions for sine and cosine, and exclude all of the divisions from the code.....	80
6.2.4	Comparison of the baseline vs the improved design.....	80
6.3	Second design improvement results (interfaces, I/F).....	81
6.3.1	1 <sup>st</sup> approach where the AXI4-Stream I/F is used both at the i/p and o/p of each accelerator block. ....	81
6.3.2	2 <sup>nd</sup> approach where the AXI4-Stream I/F is used at the i/p and the AXI4-Lite at the o/p .....	81
6.4	Run-time performance results.....	84
6.5	Experimental validation of the discriminant function results .....	85
6.6	Conclusions and future work .....	88
6.6.1	Evaluation .....	89
6.6.2	Performance .....	89
6.6.3	Classification .....	90
6.6.4	All in one system .....	90

## **References ..... 93**

Appendix 1:	MATLAB code for Relative Power .....	97
Appendix 2:	MATLAB code for Amplitude Asymmetry.....	99

Appendix 3: MATLAB code for Coherence.....	101
Appendix 4: MATLAB code for Phase difference .....	105
Appendix 5: MATLAB code for Discriminant function.....	107
Appendix 6: Note for the HLS C-code .....	109

## List of Figures

Fig. 1-1: A design overview for a TBI detection system exploiting hardware resources solely. ....	4
Fig. 2-1: Neurons image [5]. ....	8
Fig. 2-2: EEG used for the control of a prosthetic limb. ....	9
Fig. 2-3: Main brain parts [11]. ....	10
Fig. 2-4: Brain lobes [12]. ....	11
Fig. 2-5: 10-20 system [15]. ....	12
Fig. 2-6: Bipolar montage example [17]. ....	14
Fig. 2-7: Average reference montage. ....	15
Fig. 3-1: Comparison of DL and traditional ML algorithms [29]. ....	24
Fig. 3-2: Sensitivity-specificity example. ....	25
Fig. 3-3: Block diagram of the classification processor in [33]. ....	28
Fig. 3-4: qEEG variables provided from [24]. ....	29
Fig. 4-1: Typical EEG processing pipeline. ....	33
Fig. 4-2: Power spectrum of EEG data sampled at 250 Hz. ....	35
Fig. 4-3: Power spectrum of EEG data low-pass filtered, up to 50 Hz. ....	36
Fig. 4-4: Power spectrum of EEG data band-stop filtered, from 59 to 61 Hz. ....	36
Fig. 4-5: Low-pass filter specifications from MATLAB FDA tool for $f_s = 1000$ Hz. ....	37
Fig. 4-6: Low-pass filter specifications from MATLAB FDA tool for $f_s = 250$ Hz. ....	38
Fig. 4-7: Experimental EEG showing alpha rhythm reaction to eye-opening [40]. ....	39
Fig. 4-8: EEG data before ICA. ....	42
Fig. 4-9: EEG data before (blue) and after (red) ICA. ....	42
Fig. 4-10: EEG data after ICA. ....	42
Fig. 4-11: Filter bank schematic. ....	45
Fig. 4-12: Welch's method schematic representation. ....	46
Fig. 4-13: Example spectrum derived with Welch's method vs. ....	47
Fig. 4-14 : The Discriminant Function which was implemented on hardware. ....	48
Fig. 4-15: Sine and cosine in unit circle. ....	52
Fig. 4-16: CORDIC algorithm illustration for sine and cosine [47]. ....	53
Fig. 5-1: Vivado HLS design flow [57]. ....	57
Fig. 5-2: RTL to Bitstream flow. ....	58
Fig. 5-3: ZCU MPSoC system architecture [58]. ....	60

Fig. 5-4: ZCU104 Evaluation Kit [60].....	61
Fig. 5-5: AXI Master & AXI Slave components. ....	62
Fig. 5-6: AXI4-Stream interface. ....	63
Fig. 5-7: A computer sytem containing a DMA. ....	64
Fig. 5-8: Hardware design verification process. ....	66
Fig. 5-9: Algorithm flow diagram for HLS Accelerators. ....	68
Fig. 5-10: FFT baseline C-code HLS implementation.....	69
Fig. 5-11: Modified C code of Fig. 5-10.....	70
Fig. 5-12: HLS IP control signals. ....	72
Fig. 5-13: List of qEEG variables from Fig. 4-14 grouped by category.....	73
Fig. 5-14: Block design before rearrangement showing insuffienct HP port resources. ....	76
Fig. 5-15: Block design after rearrangement. ....	76
Fig. 6-1: 1 <sup>st</sup> approach DMA IP vs 2 <sup>nd</sup> approach DMA IP.....	84
Fig. 6-2: Hardware deisgn results vs software model results. ....	87
Fig. 6-3: A complete EEG processing pipeline for hardware implementation.....	91

## List of Tables

Table 2-1: Electrodes & naming convention [16].....	13
Table 2-2: Brain signals frequency bands [14]. .....	16
Table 3-1: Brain injury severity clinical indices [28]. .....	22
Table 3-2: Confusion matrix (cancer example). .....	26
Table 5-1: Design tool list.....	56
Table 5-2: Vivado HLS utilization report for ZCU104. ....	60
Table 5-3: Vivado IPI utilization report for ZCU104.....	61
Table 5-4: AXI interfaces setting for the accelerators. ....	72
Table 6-1 : Baseline FFT design, HLS utilization report.....	80
Table 6-2: Baseline FFT design after first improvement, HLS utilization report. ....	80
Table 6-3: Baseline FFT design after second round of improvements, HLS utilization report. ....	80
Table 6-4: Comparison of the baseline FFT deisgn with the improved version.....	81
Table 6-5 : Discriminant function Report Utilization for first approach of interfaces. ....	83
Table 6-6 : Discriminant function Report Utilization for second approach of interfaces. ....	83
Table 6-7 : Comparison of the two inerface approaches. ....	83
Table 6-8: Individual accelerators execution time comparison software vs hardware.....	85
Table 6-9: Complete Discriminant Function execution time comparison software vs hardware.....	85





## List of Abbreviations

ASIC	Application-Specific Integrated Circuit
API	Application Programming Interface
AXI	Advanced eXtensible Interface
BCI	Brain Computer Interface
BSP	Board Support Package
CAT	Computerized Axial Tomography
CNN	Convolutional Neural Network
CNS	Central Nervous System
CPU	Central Processing Unit
CT	Computed Tomography
DFT	Discrete Fourier Transform
DL	Deep Learning
DMA	Direct Memory Access
ECG	Electrocardiography
EEG	Electroencephalogram
EMG	Electromyography
EOG	Electrooculogram
ERP	Event-Related Potentials
FFT	Fast Fourier Transform
FE	Feature Extraction
FN	False Negatives
FNR	False Negative Rate
FPR	False Positive Rate
FP	False Positives
FPGA	Field Programmable Gate Array
GCS	Glasgow Coma Scale
HP	High-Performance Port
HLS	High Level Synthesis
ICA	Independent Component Analysis
IP	Intellectual Property
LOC	Loss of Consciousness
MEG	Magneto-Encephalography
ML	Machine Learning
MRI	Magnetic Resonance Imaging
NN	Neural Network
OS	Operating System
PET	Positron Emission Tomography
PL	Programmable Logic
PS	Processing System
PTA	Post-Traumatic Amnesia
REM	Rapid Eye Movement
RTL	Register Transfer Level
SDK	Software Development Kit
SNR	Signal to Noise Ratio
SSP	Signal Space Projection
SVM	Support Vector Machine

SoC	System on Chip
TN	True Negatives
TP	True Positives
TBI	Traumatic Brain Injury
VHDL	Very High-speed integrated circuit hardware Description Language
fMRI	functional Magnetic Resonance Imaging
qEEG	quantitative EEG

# 1

## Introduction

---

### 1.1 Introduction and motivation

Traumatic-Brain-Injury (TBI) occurs when an external force like a hard blow, a sudden acceleration-deceleration, or a penetration by an object causes internal damage to the brain. The result of the injury can be tearing of brain tissue, bruising, bleeding and in more severe cases hematoma. Every year in the European Union, 2.5 million people endure a TBI, 1.5 million of whom are admitted to hospitals and a reported 57,000 deaths occur [1]. The economic burden stemming from the consequences of TBI amounts to \$400bn worldwide. From a patient's perspective the implications of a TBI can have an impact on the daily life of those who have experienced it. This is especially problematic because the damage that has been caused may be unknown, making it hard to predict the extent of the damage and its short term and long-term effects. In some cases, TBI may result in permanent and long-term impairments and even death. People who have experienced a TBI often have problems returning to work, even a long time after the injury, suffer sleep disorders and other problems relating to daily life. Thus, TBI is a major problem for both individuals and for society in general.

To conclude, TBI is a major problem, and a late or misdiagnosed case can lead to great consequences. Therefore, early and fast detection of an injury especially in emergency settings will help prognoses and prevent discomfort sequelae. Many effective techniques to facilitate assessment and mapping of TBI effects are available in a clinical or hospital environment but these involve expensive instrumentation and equipment which are not always suitable for use in an emergency setting. The project described in this thesis is a contribution towards a low cost even portable TBI assessment instrument. Prior TBI research carried out in clinical settings has established the viability of these techniques in theory, but at the time that research was carried out, the necessary computing power to analyse the relevant measurement signals in near real time was not available. This project applies recent advances in signal processing, and the availability of parallel computing solutions provided by modern System-on-chip (SoC)

boards to establish the feasibility of a comparatively low cost and portable instrument capable of detecting the incidence and possible extent of a TBI.

## **1.2 Project aims and objectives**

The aim of this project is to investigate techniques that could facilitate the development of relatively low cost TBI assessment instrumentation, using state of the art software techniques, and hardware based on parallel computation by using task-specific hardware accelerators Intellectual Property (IP). It is intended that newly available analysis and inference techniques applying powerful compute intensive platforms with parallel processing capabilities can be applied to allow a practical implementation of prior clinical research outcomes.

The objective of this thesis is to investigate the use of Electroencephalogram as a tool to address TBI. Specifically, this work considered the utilization of power and cross-power spectrum of EEG signals as a source in order to extract features so as to indicate a brain dysfunction caused by an injury. The examination of the underlying frequencies that compose an EEG signal, and the mathematically interpreted characteristics derived from the spectrum of EEG signals was studied in this work. Finally, this work aims to use Field Programmable Gate Array (FPGA) logic in order to implement compute intensive tasks. FPGAs contain circuitry which can be configured to integrate custom IP hardware designs specific to an application as opposed to CPUs which are general purpose circuits.

## **1.3 Project outcomes**

The author has surveyed the research literature relating to TBI, especially relating to EEG and has developed a key component of a TBI analysis system that could be used in clinical or emergency situations. In particular, Fig. 1-1 shows the proposed design (which was inspired by the literature review) for a software-independent system comprised completely of hardware resources which can produce a TBI prediction based on EEG input data. The hardware implementation of the four accelerators Relative Power, Amplitude Asymmetry, Coherence, and Phase Difference, shown in this image, is the significant outcome of this thesis. The key components shown in Fig. 1-1 contribute to the process of inferring a TBI event and are explained in more detail as the thesis progresses. The deployment of this design onto a modern SoC constitutes a potential TBI assessment instrument. So, the main contributions of this thesis are:

1. Identification of a qEEG-based technique, defined as a discriminant function, which could be used to indicate a possible TBI event.
2. Creation of a set of MATLAB code blocks for the accelerators mentioned previously using computationally efficient implementations capable of being implemented on a FPGA.
3. Development of an FPGA system design that combines hardware accelerators or IP components which are equivalent to these MATLAB blocks. These hardware blocks are deployed onto a multicore CPU SoC platform (a XILINX ZCU104 FPGA board) and exploit the DMA-based streaming capabilities of this board in order to calculate a discriminant function in real-time.
4. **The work from this thesis was published and presented at the IEEE Biomedical Circuits and Systems Conference, held virtually in October 2021 [2].**

The third contribution is a notable outcome, since the real-time implementation of the discriminant function based on EEG feature extraction using DMA engines and dedicated hardware accelerators can be used as part of a potential instrument prototype as illustrated in Fig. 1-1, and help towards the real-time implementation of the remaining components of this instrument. A complete system like the one in Fig. 1-1, which carries out the required EEG signal processing tasks for a TBI inference, is suggested as a starting point which can be further extended in terms of area and speed improvement optimizations, as part of a multimodal TBI instrument.

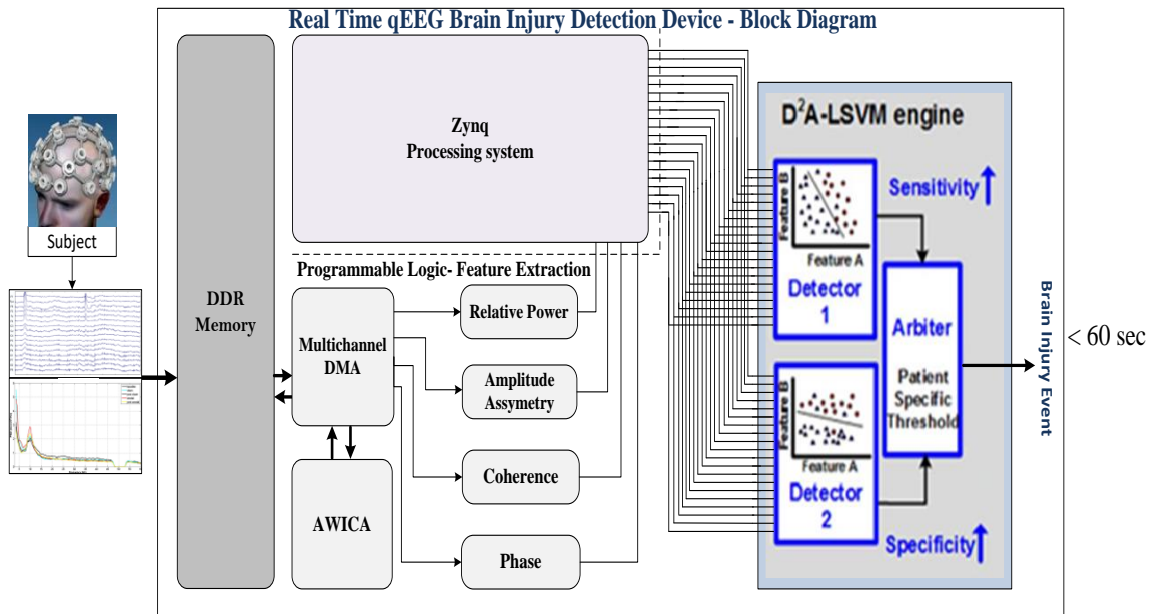


Fig. 1-1: A design overview for a TBI detection system exploiting hardware resources solely.

## 1.4 Thesis outline

The chapters of this thesis are structured as follows.

Chapter 2 covers the introductory theory about EEG and TBI. It goes through the definition of both terms and in the most part analyses the EEG properties, the advantages and disadvantages as a medical tool, and describes the utilization of it by medical or Brain-Computer Interface applications.

Chapter 3 provides the literature review i.e., the research findings that helped select the subject of this work and also helped in the understanding of the practical aspects for the EEG signal processing procedure. This chapter also covers the qEEG term analysis, in particular explains what is qEEG, how it is used for TBI detection, what is the relation of qEEG with the discriminant function, and how a discriminant function is selected which is one of the fundamental outcomes of this work.

Chapter 4 discusses the methodology followed to define and generate the discriminant function presented in this work. The chapter starts with a review of some pre-processing steps of the EEG processing pipeline, and then focuses on the methods used for the conversion of the signals from time to frequency domain, the spectral density method estimation, and finally goes through some techniques for the trigonometric function calculations.

Chapter 5 describes the implementation part of this work which is the most critical part of this project. This chapter first explains the tools and the equipment that is used for the hardware implementation of the discriminant function, and then covers the crucial pieces such as the DMA explanation and the AXI protocol, including the code design description too.

Chapter 6 finally presents the real-time performance evaluation results of the hardware implementation of the discriminant function deployed on a System on Chip (SoC) board, along with the improvement outcomes that were applied to the components of the system in the context of this thesis. In addition, this chapter discusses an overall conclusion from this research, providing details on hurdles faced along the way and the pros and cons of using EEG as a medical tool leading to a review of recommended steps for future work.

## 1.5 Learning outcomes

In the context of this work, some things that contributed to the personal development of the researcher as a professional are the following.

### 1. Rules of research conduct

During this Masters project, a part of the work was to acquire appropriate knowledge about research ethics. For that reason, I attended a course provided by Oxford university on the topic of research integrity for engineers. This course is intended to prepare researchers with the principles and responsibilities required throughout the research process. This course goes through good practises that must be followed by researchers including planning and managing of the research, data selection and analysis amongst others. This course also covered areas like intellectual property of the work that is being done and conflicts of interest so that researchers can be aware of their role and what are the limits of this role.

### 2. Practical experience

This work includes a software model of a system for TBI detection in MATLAB and the equivalent hardware implementation of this system using the XILINX tools and a XILINX SoC device. So, the first learning and hands-on outcome, is the experience gained with writing MATLAB code and working with plug-ins related to EEG signal processing. Gaining expertise in the EEG area/field is probably the most important and beneficial learning outcome. While undergoing this research on EEG signal processing, the background knowledge on this technology opens a new world of interesting job opportunities where BCI applications are

prevalent. Finally, the hardware experience is also a skill gained from this work. Specifically, working with the XILINX's HLS coding tools to build hardware designs from scratch fully deployed on SoC boards is a notable outcome.



# 2

## EEG and TBI Overview

---

### 2.1 Introduction

This chapter defines and describes the two terms that are the starting points of this thesis, EEG and TBI. An introduction and background knowledge for both of these terms is covered. The chapter explores the Electroencephalogram as a tool describing the details behind the measurement of EEG signals, and provides a brief inspection of the underlying structure of an EEG signal.

### 2.2 What is EEG?

Electroencephalogram (EEG) is a measurement of the brain's electrical activity. Based on the electrophysiological properties of neurons, EEG maps the electrical response from various parts of the brain due to a stimulus by placing electrodes onto the scalp. Our brain's functionality and all processes that require our consciousness or not, are carried out from the nervous system. The nervous system is a large network of neurons which may be seen as nodes, which are interconnected and communicate through crossroads called synapses. At this point via electrochemical processes, a neuron (resting or postsynaptic neuron) is triggered from another neuron (active or presynaptic neuron) and under some conditions, an electric signal is propagated from one end of the neuron to the other (Fig. 2-1). When a neuron is stimulated enough, this creates an action potential, which travels along its body and arrives at the synapse with the adjacent neuron. There it causes, through neurotransmitters, ions to flow to the membrane of the resting neuron. Ions alter the membrane's potential and this alteration is the post-synaptic potential [3, 4] which is actually what is measured from the EEG. Since the activity of a single neuron creates a potential that is too small to be captured using a non-invasive measurement, an EEG is the aggregate electric potential generated from thousands or millions of neurons firing at the same time and which are similarly oriented perpendicular to the surface of the brain's cortex.

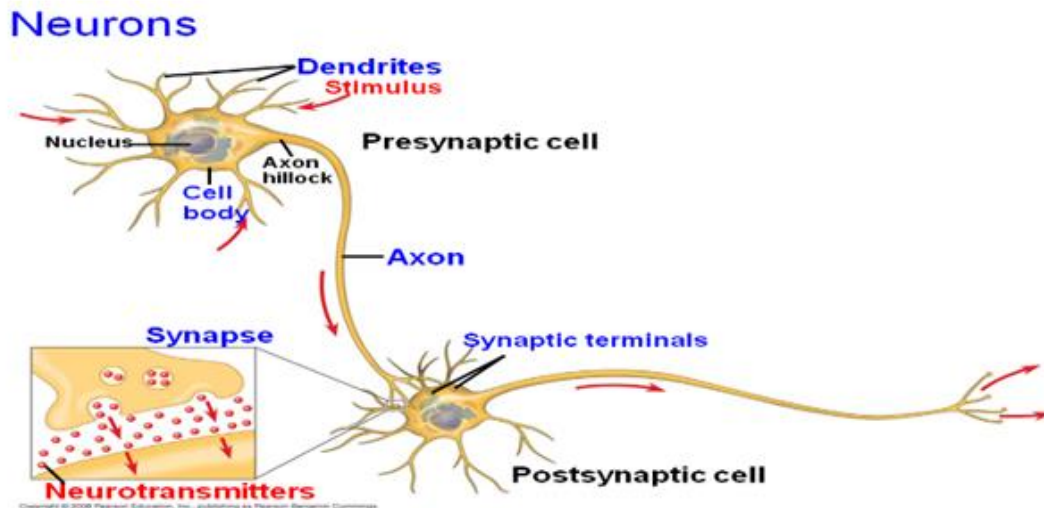


Fig. 2-1: Neurons image [5].

### 2.2.1 Biomedical applications of EEG and more

EEG is mainly used as a diagnosis tool but with technology advancement, it is increasingly used to facilitate the improvement of lives of people with special needs. The first human EEG was recorded in 1924. Hans Berger, a German psychiatrist, made a step forward from previous works conducted on animals and measured the electric potential produced by human brain directly from the scalp of a patient [6, 7]. Berger's discovery provided a new neurologic and psychiatric diagnostic tool that was initially used to detect epilepsy and is still used today. Additional usage of EEG in the clinical domain includes, distinguishing organic disorders from psychiatric, for example to differentiate organic encephalopathy or delirium from primary psychiatric syndromes such as catatonia. Furthermore, EEG is a tool to predict the outcome in patients experiencing coma, to investigate sleep disorders, to diagnose brain death and other neurological abnormalities. Another application of EEG is associated with Brain-Computer Interfaces (BCI) which in part helps facilitating lives of people with special needs. A BCI is a system consisting of a subject i.e. human and a computer-based processing unit, which translates signals, acquired from the brain into commands or tasks. EEG signals enable the illustration of brain functions and mental states to a computer. Therefore, the combination of a BCI and a user who is trained to execute a specific experiment, like the imagination of a movement, can lead to the control of a prosthetic limb. The repeated imagination of a movement can give identifiable and repeated patterns of signals from the brain, which under

proper processing will eventually be translated as the movement of the prosthetic limb as illustrated in Fig. 2-2.

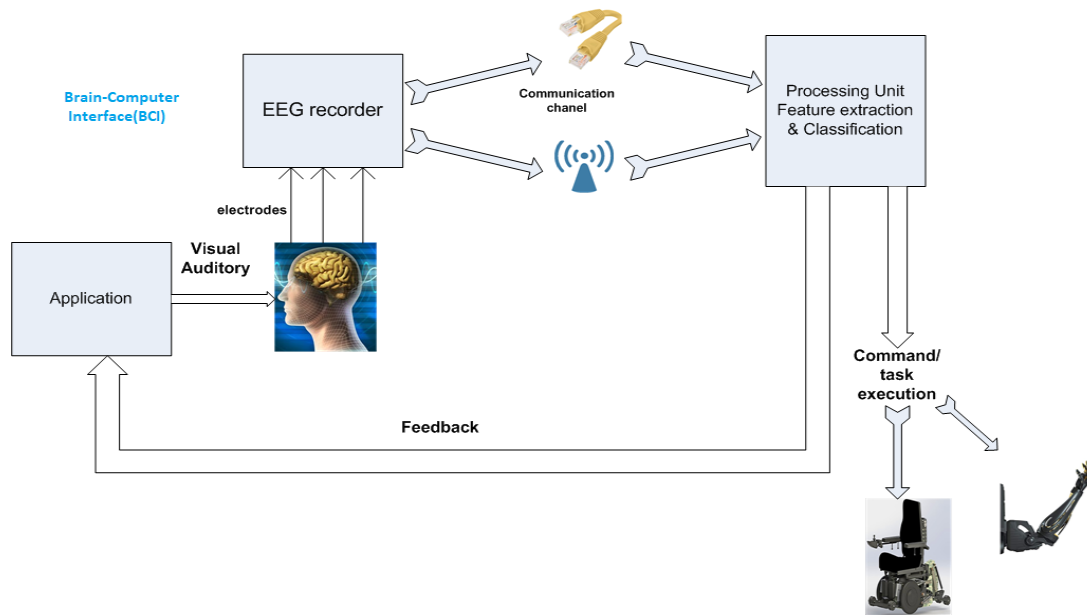


Fig. 2-2: EEG used for the control of a prosthetic limb.

### 2.2.2 EEG advantages & disadvantages

Although EEG has many applications, other techniques can be more effective - for example, in the diagnosis of tumour or stroke. Generally, brain mapping methods can be split into two categories, those that depict the brain structure in the form of images, which are static methods like Computed Tomography or Computerized Axial Tomography (CT/CAT) and magnetic resonance imaging (MRI), and those, which depict brains on-going function like EEG and magneto-encephalography (MEG), in the form of signals or waves. Structural or functional information about certain brain parts can be very useful, considering the needs of the case which is studied. That is, each method comes with advantages and disadvantages, and so EEG has its own. Two important factors for every method are temporal and spatial resolution. Temporal resolution refers to how often a measurement can be obtained and that is one of the key advantages of EEG. A typical 100-200 Hz sampling rate for EEG allows to capture the on-going activity in the brain. Some of the combined methods like functional Magnetic Resonance Imaging (fMRI) which has a temporal resolution of 1-4 seconds has been challenged for its reliability and reproducibility [8]. On the other hand EEG has bad spatial resolution [9, 10]. That is, for every electrical activity that is captured from EEG on the scalp of the brain there

are many different possible originating sources for this activity. Another practical disadvantage of EEG is noise induced by any other source than the brain, which is a challenging field for engineers. Since there is a variety of artefacts which produce the noise, many techniques and algorithms are implemented in order to reduce the effect on the signal and potentially improve the SNR (signal to noise ratio). Nevertheless, EEG is a completely safe method, as it does not affect the human at all, without exposure to radiation for example. EEG is a non-invasive method<sup>1</sup> which measures the potential that is generated from the brain. Additionally, the low cost of the equipment and the time required to set up an experiment compared to other methods makes EEG a legitimate solution for brain study.

### 2.2.3 EEG measurement 10-20 System

The human brain consists of the cerebrum, cerebellum and the brainstem (Fig. 2-3).

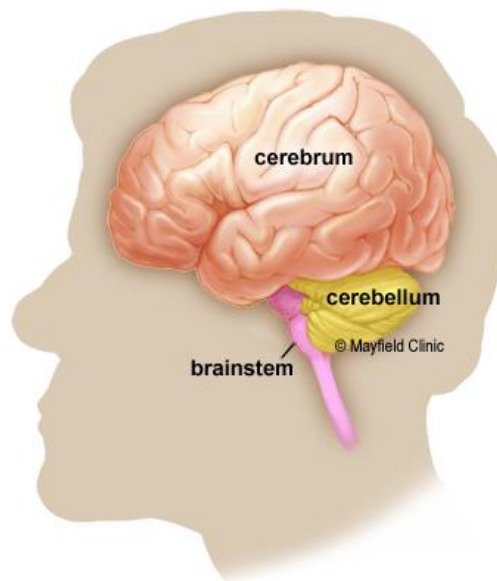


Fig. 2-3: Main brain parts [11].

- The brainstem connects the spinal cord with the cerebrum and cerebellum, and controls many automatic and essential processes like breath, heart rate, eye movement and body temperature.

---

<sup>1</sup> Although EEG can also be an invasive technique, this is out of the scope of this work as this study focuses on non-invasive measurements from the brain.

- The cerebellum is responsible for regulation and control of muscle movements and for maintaining posture and balance. Almost 80% all of the brain neurons are located there. Cerebellar damage produces disorders in fine movement, equilibrium, posture and motor learning.
- The cerebrum or cortex is the largest part of the brain and is divided into hemispheres, the left and the right. Each hemisphere controls the opposite side of the body. The cerebrum is associated with higher brain functions like thoughts, imagination, emotions, interpreting touch, vision and hearing and decision making [11, 12].

Now, the cerebrum is separated into 4 lobes, specifically, the frontal, temporal, parietal and occipital lobe as shown in Fig. 2-4.

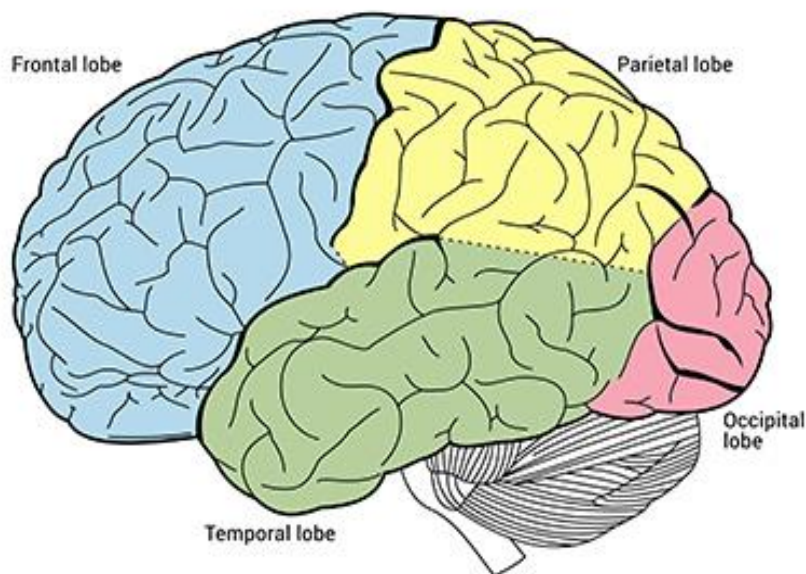


Fig. 2-4: Brain lobes [12].

**Frontal lobe** - Is the part of the brain which is responsible for conscious thoughts, decision making, voluntary movement, and controls speech (speaking and writing), personality and concentration and problem solving tasks.

**Temporal lobe** - Is associated with memory, making memories and retrieving them, understanding language, emotional control and hearing.

**Parietal lobe** - Is the place where somatosensory events are perceived and processed like touch and pain.

**Occipital lobe** - Is the section of the brain where vision is processed and interpreted, including orientation, colour differentiation, light intensity and motion and spatial perception.

Keeping in mind this basic knowledge, what follows is a review of the electrodes placement on the scalp, which is according to the rules of the international 10-20 system [13, 14]. The 10-20 system is a standard method for electrodes placement on the head. This ensures a consistent protocol for studies around the world. Fig. 2-5 illustrates the placement of electrodes according to the 10-20 system.

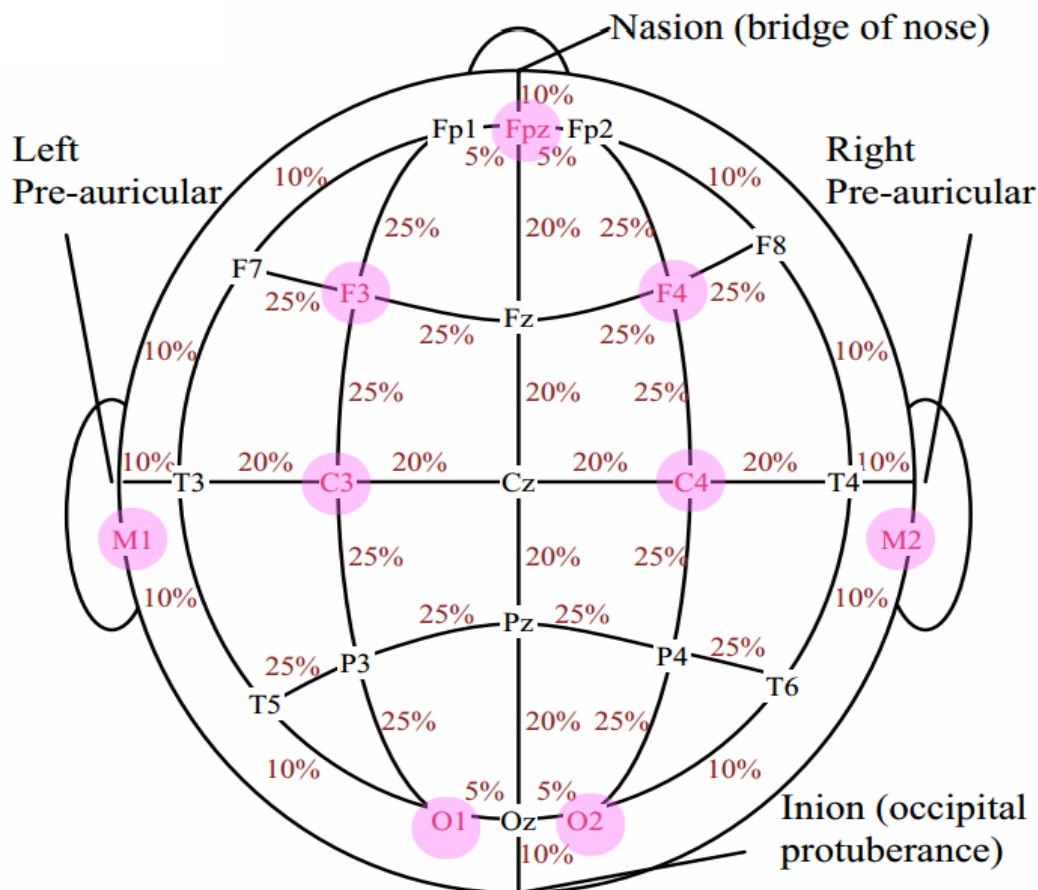


Fig. 2-5: 10-20 system [15].

The concept is the division of the head into equal spaces between the electrodes. The first step is the connection of four main positions of the head, the nasion with the inion, and the left with the right preauricular point. Then the vertical line connecting nasion and inion and the

horizontal connecting the left and right pre-auricular are divided into 10 and 20 percent portions. The points where these portions are enclosed are marked for the electrode's placement. The circumference of the head is divided into 10 percent portions as well, with the corresponding marks to each division. Then Fp1 point is connected with O1 and is divided by 25 percent portions and similarly the connection between Fp2-O2 is the same. Lastly, the lines connecting T5-T6 and F7-F8 are marked to their intersections with the aforementioned lines. The naming of the electrodes refers to the section of the brain region that is targeted (Table 2-1).

Table 2-1: Electrodes & naming convention [16].

<i>Electrode naming</i>	<i>F</i>	<i>P</i>	<i>T</i>	<i>O</i>	<i>C</i>	<i>Fp</i>	<i>Fz</i>	<i>Cz</i>	<i>Pz</i>
<i>Brain region</i>	Frontal	Parietal	Temporal	Occipital	Central	Frontal Polar	Midline Frontal	Midline Central	Midline Parietal

The numbering format is according to the side of the head at which the electrode is recording the activity. Odd numbers indicate the left side, even numbers the right side, and zero the midline.

#### 2.2.4 EEG Montages

The recording of an EEG signal is carried out using a differential amplifier. The amplifier takes two inputs and outputs the difference between them. So, an EEG recording is actually the difference between the voltages of two electrodes. A pair of electrodes constitutes a channel that is displayed as the outcome of the measurement. The display of the channels can be achieved in several ways, called montages.

**Bipolar or sequential montage** (Fig. 2-6) is the most common type of the montages. Each channel consists of two adjacent electrodes and shows their potential difference.

**Common reference or referential montage** (Fig. 2-7) represents the difference of each electrode with a common reference electrode that is usually one of the middle electrodes. A channel in this case is the electrode under examination and the common reference electrode.



**Common average reference or average reference montage** is the comparison of the signal from one electrode and the average of the rest electrodes of the head. A channel in this case is the electrode under examination and the summation of the remaining electrodes divided by the number of electrodes.

**Laplacian montage** represents the difference of a single electrode and the average of the surrounding electrodes. A channel in this case is the electrode under examination and the average of its nearest neighbours.

#### a) Bipolar montage example



Fig. 2-6: Bipolar montage example [17].



## b) Average reference montage example








Fig. 2-7: Average reference montage.

### 2.2.5 EEG signal bands

A recorded EEG signal is a set of several underlying frequencies, which are considered to have a certain distribution over the scalp and a certain biological significance. In this way, a classification into bands is applied to them for better analysis and further feature extraction. A typical EEG signal is composed of waves inside the 0–100 Hz frequency range (approximately), and the amplitude range of EEG signals is from 10 $\mu$ V to 100  $\mu$ V (approximately). The amplitude which is recorded on the scalp is of this magnitude-10 $\mu$ V to 100  $\mu$ V- and it increases to about 10-20 mV when measured from invasive methods closest to the electrical source [18]. The main frequency bands are the delta, theta, alpha, beta and gamma bands as labelled in Table 2-2. As always the data presented here are not 100% accurate and there is a margin for error because different resources might provide different information, and these are dynamic information (i.e. change over time).

Table 2-2: Brain signals frequency bands [14].

Name	Frequency (Hz)	Location	Example of filtered signal
Delta	0 – 4 Hz	Frontally in adults posteriorly in children and are stronger in the right hemisphere.	
Theta	4 – 8 Hz	Observed almost in every lobe.	
Alpha	8 – 15 Hz	Found more on posterior regions of the head including parietal occipital and the back side of temporal lobe.	
Beta	16 – 30 Hz	Generated at both regions frontal and posterior, but most evident frontally.	
Gamma	Above 30 Hz	Somatosensory cortex.	

Depending on the activity that takes place on the brain with time, the frequency of the signals changes proportionally and the amplitude changes conversely. The greater the alertness of the brain, the higher the frequency of the signal and the lower the amplitude.

**Delta waves** are associated with the less activity on the brain and are present during deep non-Rapid Eye Movement (REM) or dreamless sleep, are also known as slow-wave sleep.

**Theta waves** are noticed during the REM or dreaming sleep, drowsiness, idling or deep meditation.

**Alpha waves** are associated with relaxing, reflecting, and situations of absence of mental activity, no engagement and attenuated attention.

**Beta waves** are associated with situations of active thinking, anxiety fear and stress, high alert, concentration and learning.

**Gamma waves** occur on higher and more complex mental states like perception that combines two different senses, such as sound and sight.

## 2.3 Traumatic brain injury symptoms and complications

An injury to the brain is in essence an internal damage to the brain's matter which is caused by a jolt, a sudden acceleration or deceleration, or a penetration by an object or a combination of them. Brain injury is classified based on the severity of the damage, as **mild**, **moderate** or **severe**. Some prominent symptoms that are observed after a brain injury are:

- Loss of consciousness which lasts from a few seconds to hours depending on the severity of the injury.
- Headache that worsens or is persistent depending on the severity of the injury.
- Dizziness.
- Nausea.
- Loss of balance and coordination.
- Blurred vision.
- Ringing in the ears.
- Difficulty in speaking [19].

Some of the symptoms stated above may resolve themselves within a few hours after the injury occurs, while others may last for longer and affect the everyday life of the person. The impact of a brain injury can result in physical, but also in cognitive, social, emotional, and behavioural complications as well [19]. Regarding those symptoms that occur following the injury:

1. Physical complications include seizures, blood vessel damage, and paralysis of facial muscles or in some cases paralysis to the limbs of the opposite side of the body to which the injury occurred to the brain.
2. Cognitive complications include memory loss, impaired attention and lack of concentration, reduced processing speed. Generally, problems with executive functions like problem-solving, planning, multitasking and decision-making are experienced after a TBI event.

3. Social complications include inability to control anger, lack of initiative, troubles with communication skills and social withdrawal.
4. Emotional complications include depression, anxiety, mood swings, irritability and mania.
5. Behavioural complications include lack of self-control, apathy, and risky behaviour.

## **2.4 Conclusion**

The information provided in this chapter about EEG and TBI are not an in-depth investigation but rather an introduction to the field of this work. It is important for example to define the concept of the 10-20 system and the EEG montages but also the EEG frequency bands (Delta, Theta etc.) as they play a significant role in every study involving EEG signals. For the next chapter, the utility of EEG-qEEG as a tool for TBI detection based on current literature and studies is reviewed.

# 3

---

## Literature review and qEEG analysis

---

### 3.1 Introduction

This chapter covers any literature findings that helped this work, and analyses the term Quantitative-EEG (qEEG) which shows potential as a future tool relative to TBI detection. Specifically, the sections start with an overview of EEG-qEEG towards TBI, followed by the debate that exists around it as a detection tool and the challenges that are faced during a study incorporating EEG-qEEG. In addition the chapter provides the guidelines needed in order to establish the background knowledge for the discriminant function developed in this work, i.e. what is a discriminant function and what is the relation to qEEG variables. The chapter also explains how the discriminant function can help towards TBI detection and how it is selected.

### 3.2 Medical application of EEG-qEEG towards TBI and challenges

From a medical perspective, a brain injury is not considered as a well-defined disease. An injury to the brain is in fact a very complex issue even for experts, because the damage which is caused to the brain is associated with many factors. The location, depth and how violent the force of the trauma are some critical factors that will form the injury prognosis. It is obvious how the location of the trauma affects the damage if one considers that different parts of the brain will influence different functionalities like vision, hearing, speaking. Equally the force of the trauma may lead from temporal paralysis of certain muscles to disability. The consequences of a brain injury vary according to the severity. In all cases, even a mildly injured patient merits attention because some complications might persist for months, even years.

For TBI, no symptom or group of symptoms can constitute a sufficient sign for diagnosis by medical personnel. If no prior information exists, i.e. if it is not known that a brain injury has occurred then symptoms like dizziness, headache or nausea are common for many pathological conditions, and is not possible to have a comprehensive diagnosis based just on symptoms like these. So, for the detection of an injury to the brain, more sophisticated tools need to be utilized. Those tools must be able to detect the injury first of all, then manifest the

severity and maybe predict the extent and the duration of the injury. A tool that looks promising in this direction is EEG and specifically qEEG which is the mathematically processed EEG signal.

qEEG refers to various mathematical properties that can be obtained from processing the EEG signals i.e. amplitude, power, coherence and phase. Using EEG and qEEG in TBI examination is a relatively new approach to the field, therefore there are not many research studies in terms of quantity, but also quality, so a debate about the efficacy and accuracy of the approach as a diagnostic tool used to trouble the scientific community. While there are many studies [20] demonstrating the correct usage of EEG-qEEG as a clinical tool for the detection of TBI, the absence of a common experimental standard, and the different conditions under which each experiment was conducted, means that the field needs more research that follow the same experimental guidelines and provide reliable results.

In a TBI study involving people, firstly, TBI patients must be compared to a matched-healthy control group, to verify the reliability and the results of the study. Also, a test-retest approach is the best way to evaluate the reliability of a study since the implications of a TBI vary with time, some implications may resolve, and some others may not. The various stages after TBI has occurred are described as **acute** (hours to weeks), **subacute** (weeks to months) and **chronic** (more than 6 months). Those stages are of importance for research as EEG-qEEG changes that are observed during those stages following brain injury are used to facilitate and improve the study for future detection and prediction of the injury outcome. This can potentially provide valuable information about the therapy that patients must receive. One notable feature for those stages of TBI, are the variations on the frequency power fluctuations among the bands that is a key finding and merits attention. In addition, a patient's health history must be examined carefully, and possible current medication must be considered too. Neuropsychiatric diseases like bipolar disorder have the same effect [20] as medicines on the EEG records of the patient, and makes it difficult to discriminate between TBI or bipolar disorder or a patient taking such medications.

With regard to brain injury and using EEG-qEEG as a detection tool, there was a big debate about the clinical use of EEG-qEEG of a patient due to the misdiagnosis or the erroneous interpretation of an injury with other abnormalities. As this retired report [21] stated more than 20 years ago, qEEG-based techniques are prone to falsely identify normal patterns as abnormal, and that's why its clinical use was controversial. However, the claims of this report are no longer considered valid after a court conflict too, and EEG-qEEG remains a powerful and

promising tool in the disposal of researchers and clinicians. Neuropsychiatric conditions or medicine effects may be misdiagnosed due to the alterations and damage which they cause to the normal functionality of the Central Nervous System (CNS). Nevertheless, experts in the field are aware of the sensitivity and specificity problems that must be taken into account when they decide to target a certain problem (brain injury, epilepsy etc).

Although not as strong as standalone tool, EEG has advantages (see Chapter 1) which make it a good solution for first aid situations. In addition, the advantages provided from qEEG make this technology suitable for more widespread use in the future as a TBI detection tool. An injury to the brain can cause changes to the electrophysiology of the brain that are not interpretable by conventional EEG records. Without using qEEG measurements, it is not possible to detect changes like the phase of the signals. Using this mathematical analysis, useful conclusions can be found. This is discussed in the next section.

### **3.3 TBI assessment via qEEG**

qEEG provides helpful metrics through the mathematical interpretation of raw EEG signals. There are two promising directions which have already been investigated towards TBI detection via qEEG analysis;

1. Spectral analysis.
2. Functional connectivity.

Spectral analysis refers to the investigation of individual signal's spectrum whereas functional connectivity involves the investigation of the cross-power spectrum between two signals. Both spectrum and cross-spectrum analysis enable signal inspection using the frequency domain. This allows signals inspection over all the frequency range of the brain and consequently the sub-bands (Theta, Alpha etc.). According to [20], there are some common observations, which are deduced after quantitative interpretation of the signals of an impaired brain via spectral analysis. Increased theta and delta activity (power and amplitude) and a reduction to the alpha activity (power and amplitude) has been confirmed from many research outcomes. A reduction in power of some frequency bands might be due to the impact of the injury to the brain in the area of the brain that has been affected. Conversely, increased power might indicate the inability of the brain to cover the needs for executing some tasks and so it uses resources from other areas with a consequent increase in overall power consumption [22].

Similarly, measuring the coherence and the phase of a signal at different sites, the functional connectivity of the signals can be estimated and the synchronization of signals correspondingly [23]. Synchronization of signals is in fact a significant index of a healthy brain and a measure of neuronal network connectivity. Every task of the brain cognitive, sensory etc. requires a fully connected and effective neural network communication. An alteration in normal execution of such tasks is a manifestation of a damaged neural network, and the synchronization of signals can indicate this alteration.

qEEG features such as phase and power, are referred to in the literature as variables. Single qEEG variables cannot be used standalone as detection tools. Rather a joint usage of qEEG variables is a powerful tool for the detection of abnormalities following TBI. Those variables are combined together to constitute a discriminant function for the detection of a TBI [24]. Furthermore, the severity of the TBI can be evaluated-assessed as well [25] by a discriminant function. Some common clinical indexes for the classification of an injury are Glasgow Coma Scale (GCS), Loss of Consciousness (LOC) and Post-Traumatic Amnesia (PTA) [26]. GCS [27] is a calibration scale for the severity of head injury. GCS provides a level of consciousness by measuring eye, verbal and motor responses to the stimuli of a patient. Each category, eye, verbal, motor is labelled with a number from 1 to 4, 1 to 5, and 1 to 6 correspondingly. Their aggregation indicates the level of the severity of the injury. LOC and PTA are measurements of the duration of unconsciousness and amnesia respectively after TBI. Table 3-1 below lists some indices and their correlation with the severity of TBI.

Table 3-1: Brain injury severity clinical indices [28].

Criteria	Mild	Moderate	Severe
Structural imaging	Normal	Normal or abnormal	Normal or abnormal
Loss of Consciousness	0–30 min	> 30 min and < 24 hrs	> 24 hrs
Alteration of consciousness/mental state (AOC) any alteration like confusion, disorientation, slowed thinking etc	a moment up to 24 hrs	> 24 hours. Severity based on other criteria	



Post-traumatic amnesia	0-1 (day)	> 1 and < 7 (days)	> 7 (days)
Glasgow Coma Scale (best available score in first 24 hours)	13-15	9-12	< 9

The distinction between mild, moderate and severe TBI relies on the extent of the symptoms. In other words, pretty much the same symptoms occur for each form of the TBI. Headache, dizziness, loss of consciousness, blurred vision and ringing in the ears are some consistent symptoms for any case of brain injury severity. The impact of a headache on the patient is often the difference. For example, if the headache is persistent and gets worse over time then this indicates severe TBI. On the other hand, if the headache lasts for a short amount of time, then the case is mild. Furthermore, the outcome of a TBI can be a classification index too. For instance, if a TBI leads to death then it is automatically classified as a severe case. According to [25] a discriminant function is not only able to accurately detect a brain injury, it can also classify the injury as mild, moderate or severe similar to the results that show the indices above in Table 3-1. However, the accuracy of a method is not the only thing which is of concern for a health care study. There are also the terms of sensitivity and specificity, which will be covered next.

### 3.4 Factors for the evaluation of a medical computer-based method

Based on the field in which a study is conducted, the researchers must adapt to any issues or limitations associated with it. That is, in the medical field when a research includes the use of a Machine Learning (ML) technique, then this is classified as a-priori known that it is likely to face data availability issues. ML refers to the utilization of sophisticated algorithms like linear regression, Support Vector Machines (SVM), decision trees and many others, to execute various tasks that were not explicitly written for, based on training data. These algorithms are trained and evolve over time so that they can gain knowledge on a particular task. A subset within the Machine Learning realm is Deep Learning (DL). DL is the term used to describe all types of neural networks (NN) like convolutional neural networks, recurrent neural networks etc. NN is a network of artificial neurons designed to be an analogous model to the human brain neural network. Currently NN is a state of the art technique used in the context of learning algorithms. One major requirement and the most important for a NN to perform adequately is the amount of training data available. The more the data, the better the performance of the

network. Otherwise the performance of NN can be poor. Fig. 3-1 is a simple but concise diagram showing the superiority of NNs compared to other ML algorithms.

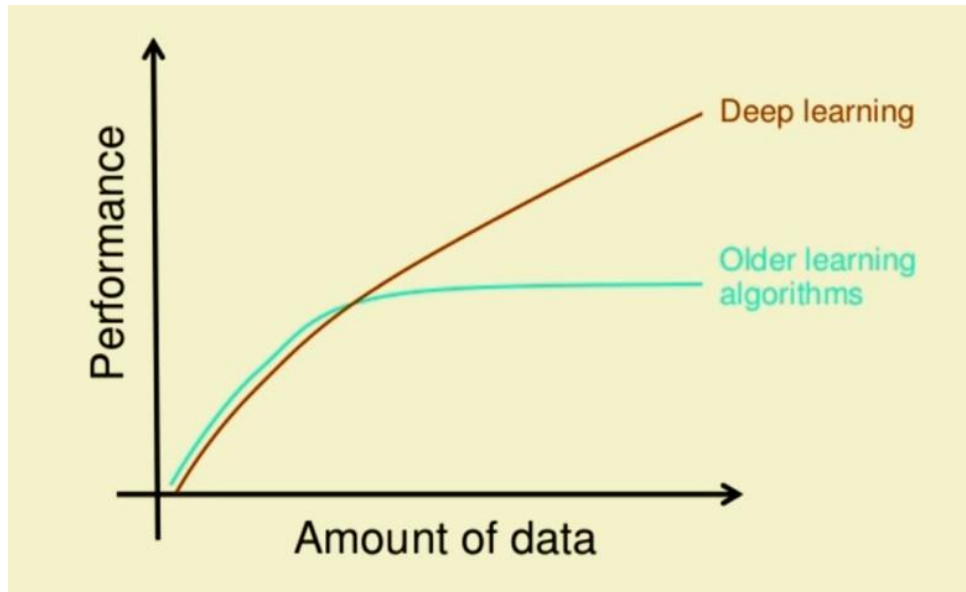


Fig. 3-1: Comparison of DL and traditional ML algorithms [29].

Given this superiority of NNs and the amount of data that can be found in a BCI application, the authors in [30] developed a wearable real-time system able to recognize human emotions which included a Convolutional Neural Network (CNN). Although this work uses a CNN for the decision-making part, the architecture still implements the feature extraction part separately. To be more specific, NNs can automate the procedure of feature engineering, i.e. feature extraction and feature selection. The first layers of the network carry out the feature extraction and selection while the rest of the layers perform the classification. This “detour” of feature extraction and selection process does not allow any control on the features from the designer [31]. In any way, the availability of large data sets, is what will enhance the ability of the network to detect the most suitable features, so this automation may be more promising than traditional feature extraction methods when there is a plethora of data available. By exploiting complex maths, neural networks are able to achieve classification accuracies over 99%. Nonetheless, accuracy is not the only factor to consider in a medical study. Sensitivity and specificity are critical factors that determine the efficiency of a method in the medical field. Sensitivity is how well a method can detect a disease or disorder to a subject who has it, and specificity is the ability of this method to detect when the subject does not have it. This ambiguity on the definition of these terms can be illustrated with the following example, which helps with the comprehension of the terms.

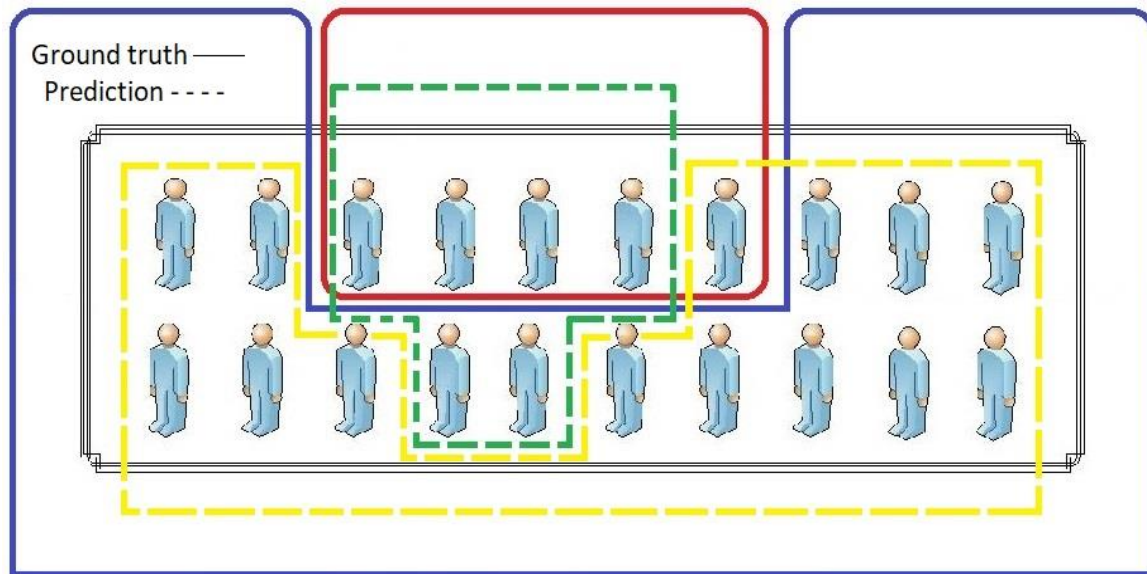


Fig. 3-2: Sensitivity-specificity example.

The diagram in Fig. 3-2 shows a sample of 20 subjects, 5 of each are patients having a certain type of cancer highlighted within the red colour box. The remaining 15 subjects delineated by the blue colour box are healthy. Now, suppose that a new method is developed to detect cancer and is applied to the 20 subjects in order to tell who has the cancer and who has not. Subsequently, the classification results will evaluate the method. According to the method, 6 out of 20 subjects have cancer which are delineated with the green colour box, and 14 out of 20 subjects are healthy, delineated with a yellow colour box. Solid lines denote the ground truth and dashed lines denote the outcome of the classification method.

Some performance metrics can be derived that will help evaluate the method in the aftermath, given the description above. These metrics are termed, accuracy, sensitivity and specificity. Before proceeding to the calculation, it must also be defined what true & false positives and true & false negatives are. As shown in Fig. 3-2, 4 out of 5 positives were correctly identified by the method, and this number is referred to as the True Positives (TP). In other words, TP is the intersection of the red solid lined rectangle and the green dashed lined shape. Another correct result is termed the True Negatives (TN). There are 13 out of 15 healthy subjects that were correctly identified as negatives. TN is the intersection of the blue solid lined shape and the yellow dashed lined shape. Furthermore, False Positives (FP) are classified as 2 out of 15 healthy subjects that were falsely classified as positives. This is the intersection of the blue solid lined shape and the green dashed lined shape. Finally, the False Negatives (FN) metric is specified as 1 out of 5 patients that falsely classified as negative, or the intersection

of the red solid lined rectangle and the yellow dashed lined shape. At this point the accuracy sensitivity and specificity of the method can be calculated. These are given by the following equations (3-1) ~ (3-3) [32].

$$accuracy = \frac{TN + TP}{P + N} \quad (3-1)$$

$$sensitivity = \frac{TP}{TP + FN} \quad (3-2)$$

$$specificity = \frac{TN}{TN + FP} \quad (3-3)$$

where TP, FP, TN, FN are true & false positives and true & false negatives respectively, as described above, and P and N are the actual positives (P) and negatives (N). With P=5, N=15, TP=4, FP=2, TN=13, FN=1, the accuracy of our method is calculated as  $accuracy = \frac{4+13}{5+15} = 17/20$  or **85%**, the  $sensitivity = \frac{4}{4+1} = 4/5$  or **80%** and  $specificity$  is  $= \frac{13}{13+2} = 13/15$  or **86.6%**. The calculations above can be derived also from the so-called confusion matrix shown in Table 3-2. Confusion matrices are widely used in Machine Learning applications as a depiction of the result versus the ground truth, to extract useful information like the metrics calculated above. Using this matrix, we can extract additional useful metrics but in the context of this work only accuracy sensitivity and specificity are considered.

Table 3-2: Confusion matrix (cancer example).

		Actual	
		P(5)	N(15)
Predicted	P(6)	4 (TP)	2 (FP)
	N(14)	1 (FN)	13 (TN)

The example described here was demonstrated as cancer is a serious illness. Thus, it is crucial for a patient who is positive not to be misdiagnosed as negative. Conversely, a healthy human with no cancer should not be misdiagnosed as positive. The importance is equal in both

cases, however if a person positive to cancer is misdiagnosed, this can be fatal. On the other hand, if a human negative to cancer is misdiagnosed, then he or she will go through the pain and discomfort of believing he/she has contracted the cancer illness. So, a comment about the evaluation of this example's made up method is that although it is a good method-it has high scores in all 3 metrics ( $>80\%$ )-nevertheless it needs improvement. This method performed better on detecting negatives (specificity) than positives (sensitivity) which means that it has a greater false negative rate (FNR) than a false positive rate (FPR). Particularly for 2 healthy subjects out of 15, the  $FPR=2/15=0.133$  that are misclassified as positives, and 1 patient (positive) out of 5  $FNR=1/5=0.2$  misclassified as negative. This interpretation is important because as already mentioned, if someone who has cancer cannot be detected then this can be fatal, therefore the method must increase the ability for not misclassifying positives, i.e. reduce the false negative rate (FNR).

Brain injury cannot be compared with cancer illness, however in a study, both the sensitivity and specificity must also be analysed, and not just the accuracy. As mentioned previously, a debate existed in the field about the use of EEG-qEEG as a detection tool due to potentially incorrect interpretations. That is, if another disease or the effect of medication is interpreted as TBI, this is a false positive, and thus the specificity needs improvement. Using a system technology similar to the one in [33] with two integrated ML engines (Fig. 3-3), where each one controls sensitivity and specificity correspondingly, then maybe a more robust technique resilient to false positives is achieved. This can potentially help to overcome to some extent the controversy that exists in the field.

This work does not use neural networks for the realization of the ML engines but rather SVMs [34]. This is important due to the small amount of data available in the medical field compared to the countless datasets at hand in BCI applications. As a result of this constraint, the utilization of a neural network is not suitable for TBI applications. An illness is not as frequent an event as an emotion or a movement of the limbs for example.

Another factor to consider is the features' derivation method. For spectral-based features there are two ways to derive the features, one is via Fourier Transform and second is filter banking. The work described in [30] developed Feature Extraction (FE) methods for an emotion recognition application by converting EEG signals from time domain to the frequency domain via Fourier Transform. In contrast, the authors in [33] present a FE engine design for epilepsy seizure detection that uses a filter banking technique for the conversion of EEG signals to the frequency domain based on windowed band-pass filters to produce feature vectors. In

Fig. 3-3 the complete dual detection classification processor architecture is shown which incorporates the FE engine design with a SVM-based machine learning algorithm to support patient-to-patient and age-to-age variations to increase the sensitivity and specificity. Two SVM seizure detectors are used in conjunction with an arbitration engine to improve the accuracy of results for both sensitivity and specificity.

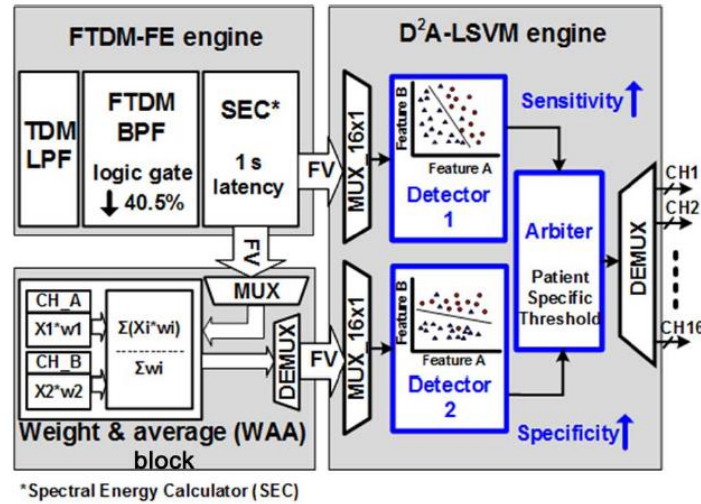


Fig. 3-3: Block diagram of the classification processor in [33].

Both aforementioned studies are real-time applications. These designs implemented the hardware EEG signal processing methods differently according to the application needs. The proposed architecture in each case must analyse multi-channel EEG data in real-time and be suitable for hardware FPGA or chip implementation. Sometimes filters are more suitable as building blocks for the features and sometimes the Fourier Transform is the better option, but that depends for example on the number of the features, the number of electrodes, the spectrum of frequencies being examined and other relative details which lie in each study's needs. So, the available data, the classification technique (ML algorithm), the metrics to evaluate it, and the implementation of both the features and the ML algorithm are the factors that will determine the efficiency of a method and as a consequence this will decide if it is suitable for a real-time application or not.

### 3.5 Selecting the discriminant function for TBI

A discriminant function constitutes the set of features that will efficiently reveal the existence of a disease or a disorder, in this case a brain injury. In [24] the power spectrum and

cross-power spectrum were used as a source for the individual features or variables which included spectral variables such as Relative Power and Amplitude Asymmetry and cross-power spectral variables like Phase Difference and Coherence. The details for these features are described in the next chapter. As reported in [24] the discriminant function developed which is shown in Fig. 3-4 can achieve high percentages of overall sensitivity (95.2%), specificity (93.2%) and accuracy (94.8 %). In addition, the authors of [24] developed another discriminant function in [25] but the sample size of patients in the latter study is only 108 compared to 608 of the former study. Therefore, the discriminant function in [24], was implemented on hardware in this work, due to the large sample of patients included in the study and the high percentages of overall classification.

TABLE III

List of EEG power spectral variables entered into the training-set discriminant function and the pooled-within-groups correlation between each variable and the canonical discriminant function.

Variables	Correlations
Theta frequency coherence between Fp1 and F3	0.3366
Beta frequency coherence between T3 and T5	0.2598
Beta frequency coherence between C3 and P3	0.3915
Beta frequency phase between Fp2 and F4	-0.4658
Beta frequency phase between F3 and F4	-0.4537
Alpha frequency amplitude asymmetry between F4 and T6	0.3298
Alpha frequency amplitude asymmetry between F8 and T6	0.3129
Beta frequency amplitude asymmetry between F4 and T6	0.2886
Beta frequency amplitude asymmetry between F8 and T6	0.2921
Alpha frequency amplitude asymmetry between F3 and O1	0.2939
Alpha frequency amplitude asymmetry between F4 and O2	0.3241
Alpha frequency amplitude asymmetry between O1 and F7	-0.2944
Beta frequency amplitude asymmetry between F4 and O2	0.2722
Alpha frequency relative power for P3	-0.2612
Alpha frequency relative power for P4	-0.2544
Alpha frequency relative power for O1	-0.3532
Alpha frequency relative power for O2	-0.3529
Alpha frequency relative power for T4	-0.2390
Alpha frequency relative power for T5	-0.2851
Alpha frequency relative power for T6	-0.2832

Fig. 3-4: qEEG variables provided from [24].

Although there are plenty of options for features in the frequency domain, the combination of spectrum and cross-power spectrum features remains the more promising approach. Once the set of features is determined, the next step is to select the most appropriate ones. In the beginning, a stack of features can be exceptionally large depending on the number of EEG channels. Furthermore, the optimal combination of features is probably the most crucial part

of this step. Some methods for this purpose include analysis of variance and genetic algorithms. As reported by the authors in [35], the analysis of variance can perform better than genetic algorithms for this task.

In [24], a multivariate analysis of variance is conducted from the initial large set of features-variables. The ensemble of qEEG variables, which are used as discriminant features for TBI are selected after multiple such statistical tests. In the beginning, numerous variables are available for analysis. This is because all of the features (relative power, amplitude asymmetry, coherence, phase) are measured for every electrode or pair of electrodes across the frequency bands. Therefore, sifting through those large groups of measurements was critical in order to retain the most appropriate variables. Variables are initially divided into groups with respect to the severity score indicated by GCS. Particularly, three groups of variables were set, with the corresponding score of each group being 13, 14 and 15 for the injured subjects. A comparison was made on those groups of the variables against the variables obtained from normal subjects for homogeneity of variance. Only those variables, which exhibited equal variances, were analysed further for the next steps. This is important because variables with equal variances are of the same kind. This means that those variables are representative of a new neurological pattern or a new state of the brain following the injury which is discussed in this study. That is, these variables that exhibit equal variances are features of this new state in the brain. After the stabilization of this state in the brain then the variables present a homogeneity of variance i.e. the new range of their potential values is not random. Let's take as an example the case of having a fever. Temperature measurements for people who are not sick will be distributed around 36.4°C. On the other hand, temperature measurements for sick people will be distributed around higher values 38°C, 39°C or so. That means the transition from one state to another, affected the mean of the values of the fever or feature-variable for sickness. So, from those statistical tests, the aim is to obtain the most discriminant features for a brain injury.

### **3.6 Conclusion**

This chapter has identified a promising technique that can be used to facilitate possible instances of a TBI, using EEG measurement, and qEEG calculations to define a discriminant function. Based on the study outcomes in [24], and given the large samples size of the subjects compared to the number of subjects in study [25], and the high scores of overall classification, the discriminant function of the first study was selected for implementation in the context of



this work. Literature suggests that the combination of power and cross-power spectrum analysis is the best way to build a discriminant function for TBI detection. The discriminant function in [24] contains 2 power spectral features that are termed Relative Power and Amplitude Asymmetry, and 2 cross-power spectral features, which are called Coherence and Phase Difference. In the next chapters, the step by step development of those features is discussed. In particular, Chapter 4 covers the theory and methods behind the realization of these features and furthermore the design implementation of these features is presented in Chapter 5 which give form to the Discriminant Function.



## 4

# Design methodology

## 4.1 Introduction

The goal of this chapter is to cover the methodology behind the derivation of the features that constitute the discriminant function in [24]. Features extraction, which is a key EEG signal processing step as outlined in Fig. 4-1 is the essential part of this work, so the signal processing methods used for their calculation are described in detail. The selection of the features that best describe the target, i.e. TBI detection, has already been explained in Chapter 3, section 3.5. For this work, the demanding task of artefacts rejection is not considered as part of this thesis implementation. The focus of this chapter is about the theoretical background for the features extraction step leading to a hardware implementation and it is assumed that clean data is fed into the system.

## 4.2 EEG pre-processing steps

A typical pipeline for EEG signal processing comprises of the following steps as shown in Fig. 4-1

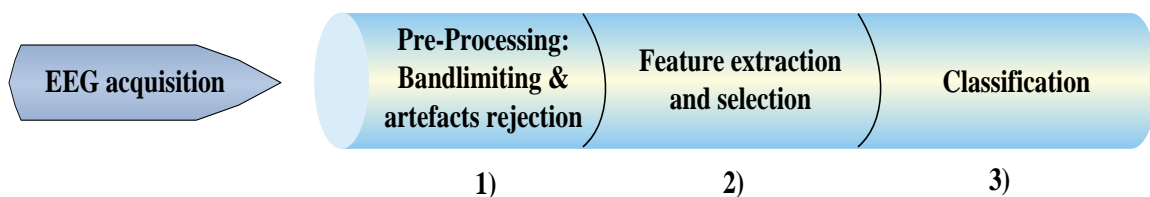


Fig. 4-1: Typical EEG processing pipeline.

This chapter describes the pre-processing steps, however, these steps and the classification stage are not part of the hardware implementation. The classification step is briefly discussed in 5.6 & 6.6.

### 4.2.1 Bandlimiting

Once the EEG acquisition stage is completed, and the data is ready for the next stage of processing, a low-pass filter or a bandpass filter is used to cancel out dc components i.e. frequencies below 1 Hz. This filter is applied to the signals from each EEG channel to limit signal frequencies to the desirable range. At this point, it is important to note an optional step preceding band-limitation. Sometimes when the sampling rate is greater than the standard rate for EEG applications-which is around 250 Hz, then data reduction is an optional, but rather important step. Suppose that a sampling device has a fixed sampling rate which is not adjustable, at 400 Hz for example. Due to the fact that this sampling rate is higher than the standard for EEG applications, it results in redundancy of data. Therefore, downsampling those data is a common technique used to reduce the amount of data captured and consequently the memory required for their storage. The outcome of downsampling is a narrower spectrum of frequencies as well.

Although narrowing the frequency range is done with filtering, downsampling will result in a narrower spectrum too. However, digital filters provide more flexibility to narrow down the spectrum, whereas downsampling will reduce the frequencies of the spectrum, to half of the sampling rate according to the Nyquist theorem  $f_s \geq 2f_{max}$ , where  $f_s$  is the sampling frequency and  $f_{max}$  is the highest frequency that can be recorded with the given sampling frequency. According to the Nyquist theorem, the sampling frequency has to be at least double or greater of the value of the highest analogue frequency component of the signal [36, 37]. So, the sampling frequency is what will determine the limits of the examined spectrum without using filtering. Using filtering methods, the limits of the desired spectrum are defined freely.

Consider the following example; assume a sampling frequency equal to 1 kHz, so the maximum frequency that can be measured is 500 Hz. In order to apply downsampling, this sampling frequency is reduced to 250 Hz, so the maximum frequency component that can be detected now is at 125 Hz (narrower spectrum). Then the sampling is repeated again with the new sampling frequency (250Hz) to the formerly acquired data with the higher sampling frequency (500Hz). With respect to EEG signal processing, this reduction in the sampling frequency is advantageous, provided signals from the brain are transmitted at a frequency range from 0.5 Hz to a value slightly greater than 100 Hz (approximately). Keeping the higher sampling frequency for an experiment with EEG signals results in no valuable information above ~100 Hz, so the data can be considered redundant. Selection of an appropriate sampling frequency, which covers the spectrum of interest, is optimal as a first step. So, conceptualizing

downsampling, a) essentially, is a second sampling from the already captured data, at a lower rate. b) it can reduce the frequency range of a signal, according to the sampling rate of the front-end device. If it's necessary to use downsampling, then, unavoidably the frequency spectrum content is reduced.

Following downsampling (if applied), conventional filtering is the subsequent necessary step to adjust the frequency spectrum in a suitable form for each case studied. An advantage of filters is that they can be realised in digital and analogue format, if required by the design needs. Analog filters outperform digital in terms of speed. However, digital filters are more flexible in terms of design and implementation that make analog filters used only if it is demanded from system requirements. There are various types of digital filters implementations and these types are low-pass, high-pass, band-pass and band-stop. A low-pass filter rejects all frequencies from a signal above a threshold called the cut-off frequency. Conversely, a high-pass filter eliminates all frequencies below the cut-off frequency. Band-pass and band-stop filters allow and reject, respectively, a pre-defined frequency band, by adapting low-pass and high-pass filters for this purpose. The images in Fig. 4-2~Fig. 4-4 show examples of EEG signals being filtered using the EEGLAB [38] plug-in, a MATLAB<sup>2</sup>-based toolbox for EEG signal processing. Fig. 4-2 shows the power spectrum of EEG sample data provided by OpenBCI [39], an open-source software encompassing EEG signal monitoring and processing.

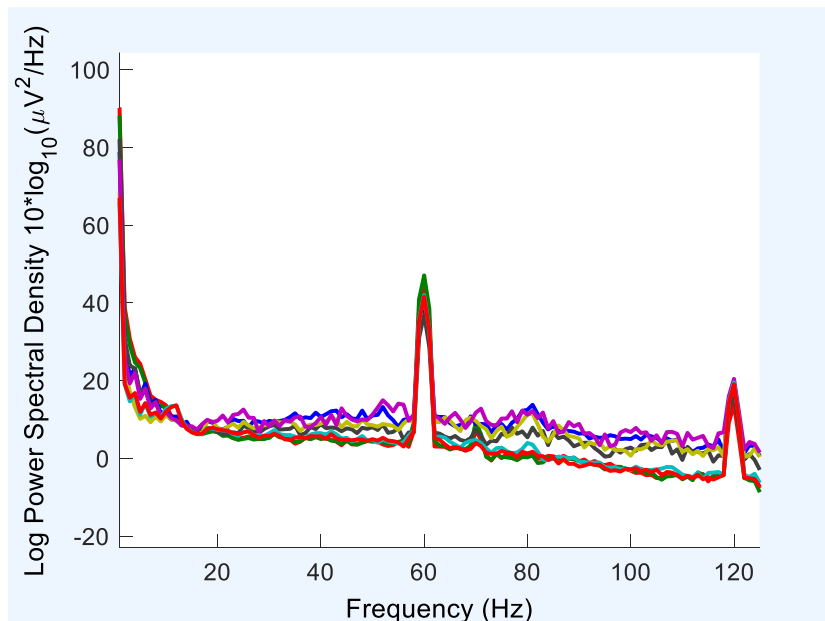


Fig. 4-2: Power spectrum of EEG data sampled at 250 Hz.

<sup>2</sup> <https://uk.mathworks.com/>

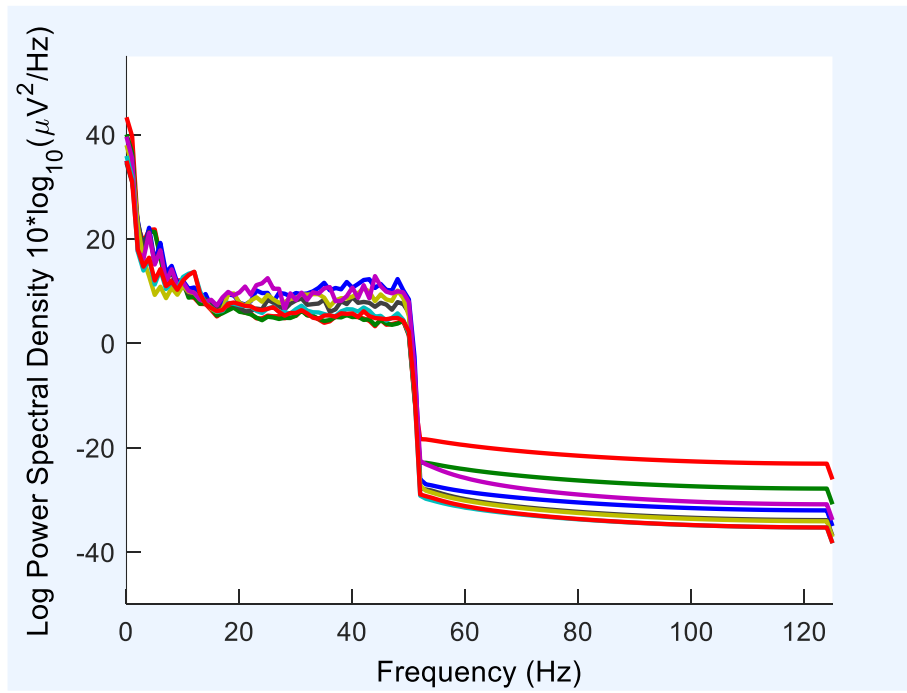


Fig. 4-3: Power spectrum of EEG data low-pass filtered, up to 50 Hz.

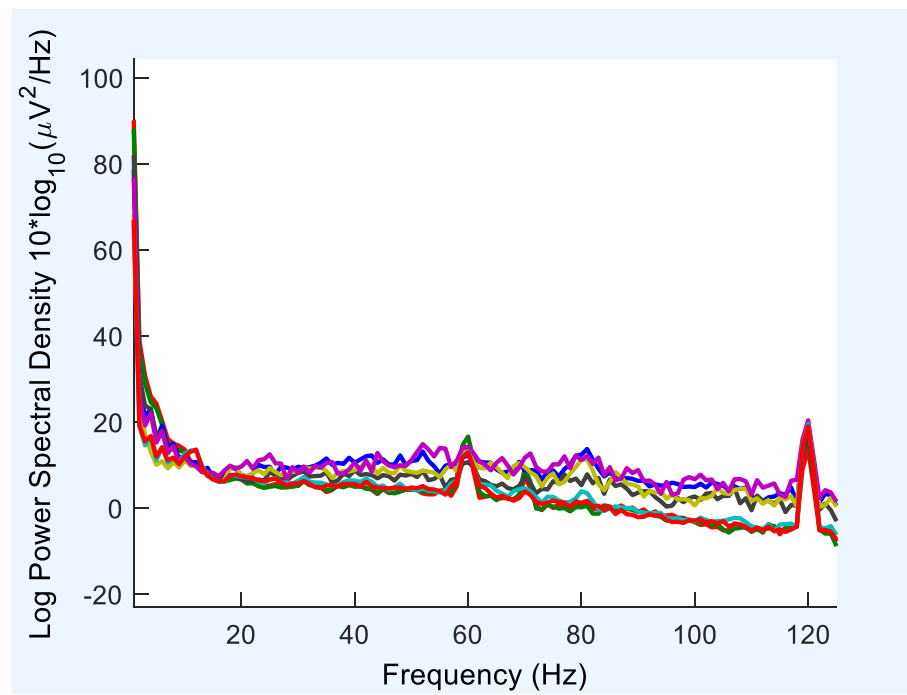


Fig. 4-4: Power spectrum of EEG data band-stop filtered, from 59 to 61 Hz.

The sampling rate is at 250 Hz, hence the spectrum contains frequencies up to 125 Hz. At 60 Hz, this peak in the diagram is noise originating from the power line, which in the USA is at 60 Hz while in Europe, this is at 50 Hz. Fig. 4-3 illustrates the same EEG data from Fig. 4-2 that is low-pass filtered with a cut-off frequency of 50 Hz. Fig. 4-4 shows the EEG spectrum that is band-stop filtered with a lower cut-off frequency of 59 Hz and a higher cut-off frequency of 61 Hz. It should be noted that for the process of analysis, frequencies below 1 Hz (dc terms) are not taken into account. Therefore, digital filters provide the ability to target specific frequency bands where the noise is prominent and to nullify only this part without affecting the rest of the spectrum. Power line noise removal is an example demonstrating the flexibility of digital filters. Finally, the selection of an appropriate sampling frequency for each case can have benefits for the filter design as well. The wider the frequency range i.e. the greater the sampling frequency ( $f_s$ ), the more coefficients will be needed for the filter implementation, which results in more resource consumption in terms of operations, execution time and hardware primitives (registers, adders, multiplexers etc.).

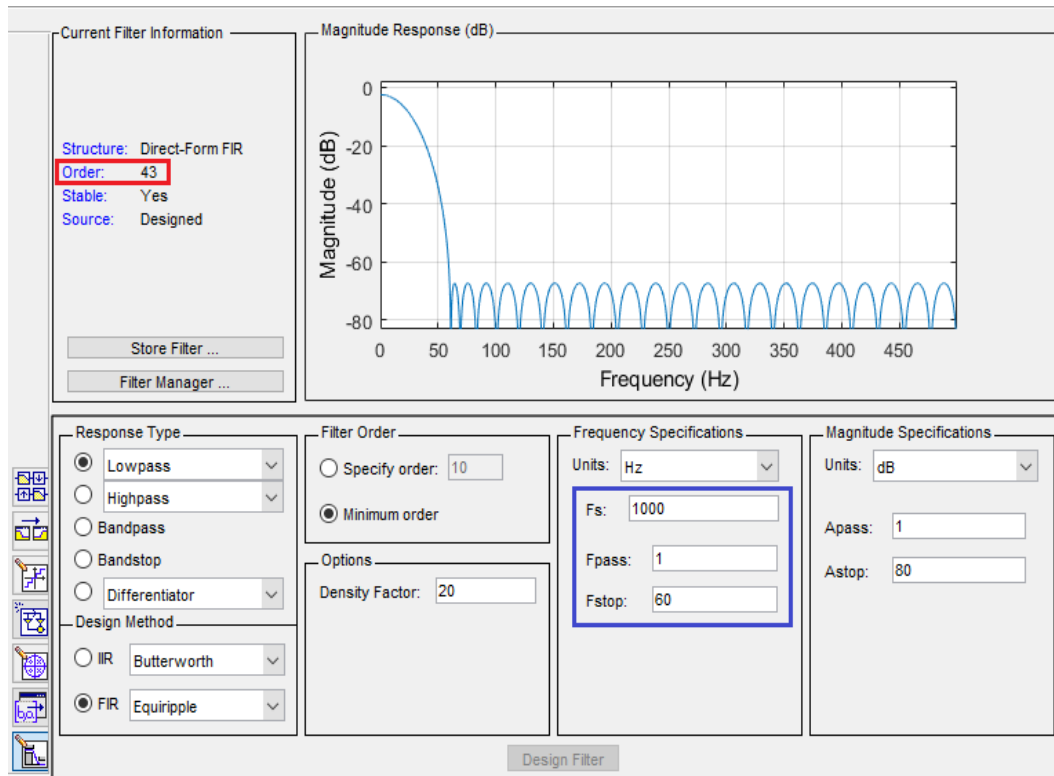


Fig. 4-5: Low-pass filter specifications from MATLAB FDA tool for  $f_s = 1000$  Hz.

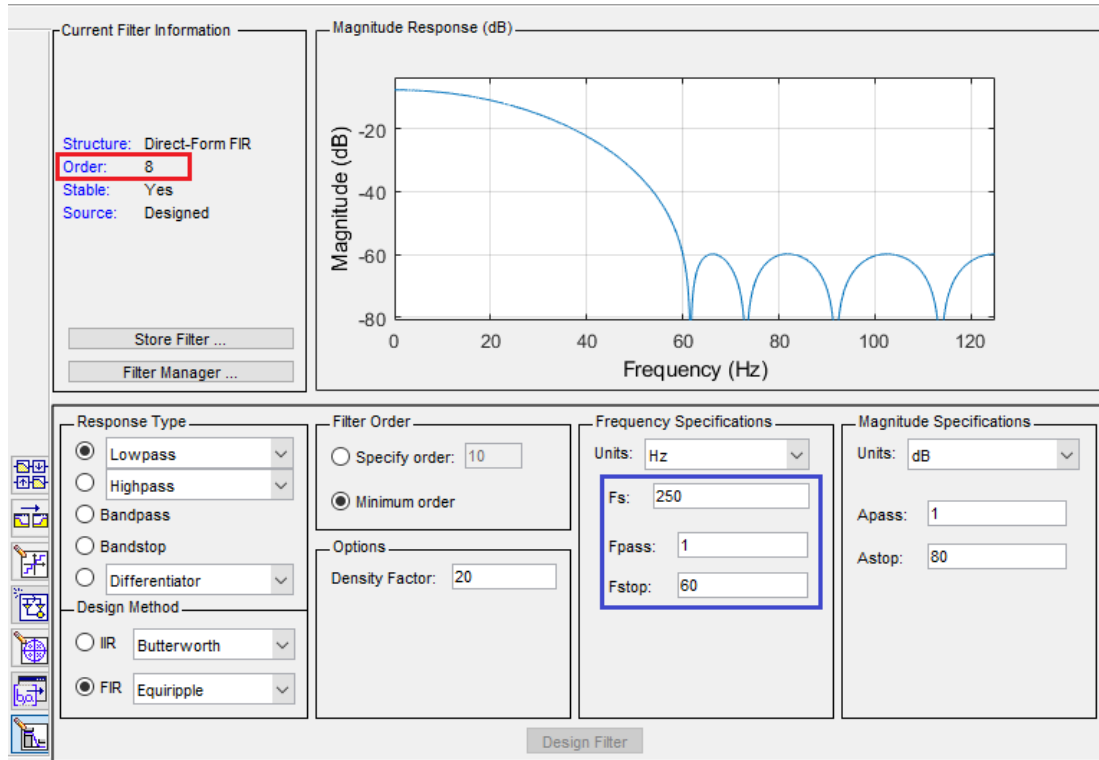


Fig. 4-6: Low-pass filter specifications from MATLAB FDA tool for  $f_s = 250$  Hz.

The diagrams in Fig. 4-5 and Fig. 4-6 illustrate the difference in a low-pass filter design using the Filter Designer application in MATLAB. This filter could be a potential implementation for an EEG application. Both designs have the same lower and upper limit for the frequency range, but differ in the sampling frequency as shown in the blue-highlighted areas. In the first case  $f_s = 1000$  Hz and in the second case  $f_s = 250$  Hz. As can be seen from the red-highlighted area in both figures, the second design has a lower order, which means fewer hardware resources and power consumption.

#### 4.2.2 Artefact rejection

EEG *pre-processing* is the term used to describe the steps required in order to prepare the signals for feature extraction. These steps are filtering and artefact rejection. This term is called pre-processing because signals are not ready yet for analysis in the frequency domain, so some processing is required before the conversion. Signals not being ready means that signals contain non useful information that must be removed. Signals containing non useful information are considered non clean data, and the “stains” are called artefacts. An artefact is anything that contaminates the signal and gives no valuable information in the feature extraction step. Signals



are contaminated from various sources of noise like the power line noise presented in Fig. 4-2. For the artefacts rejection phase, signals are filtered out and “cleaned”, occasionally several times, to produce data with as much suppressed noise as possible. That noise, if not cancelled out, is transformed and is present in the signal spectrum, and causes troubles to the feature extraction step, by obstructing identification of known patterns.

Each study of the brain using EEG signals is based on observations of the signal’s alteration in terms of amplitude, frequency and phase. Alterations occur from external events, for example the sight of an image, listening to a sound, touch, and generally whatever triggers one of the senses. Also, tasks taking place in the brain e.g. feelings, thoughts, memories etc. cause alterations to the normal EEG activity of the brain. If those alterations are not distinguishable enough over noise, then no useful conclusions can be drawn. An example of EEG signal alteration, for instance, is to observe the alpha rhythm blocking when eye-opening activity occurs, in other words vision activation (Fig. 4-7).

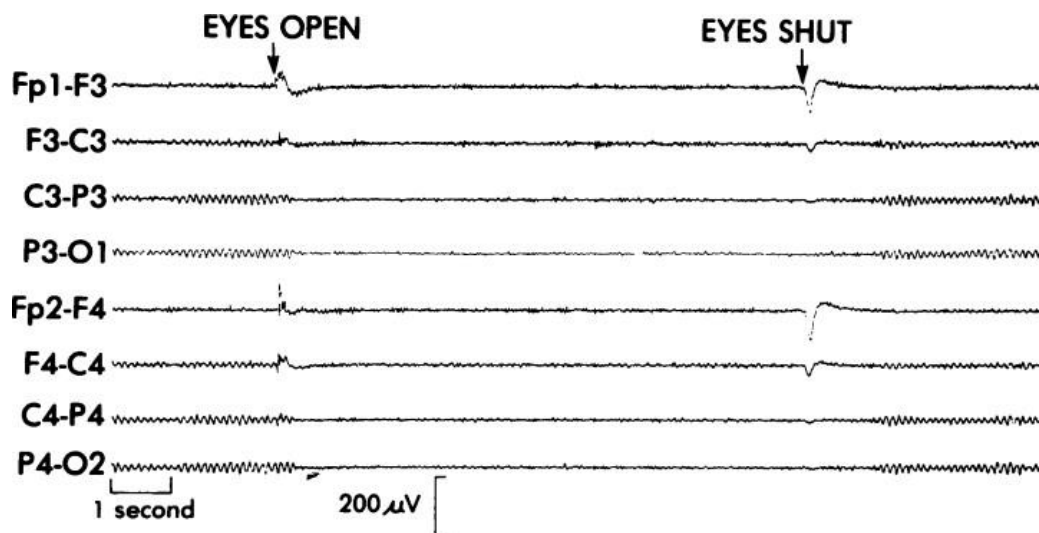


Fig. 4-7: Experimental EEG showing alpha rhythm reaction to eye-opening [40].

#### 4.2.2.1 Types of artefacts

EEG artefacts are induced either intrinsically or extrinsically. Intrinsic artefacts are caused from any other part of the body rather than the brain. Extrinsic artefacts are generated from the surrounding environment and the EEG equipment.

- 1) **Intrinsic Artefacts:** Include every muscle movement as in moving eyebrows, jaw clenching, cheeks movement (e.g. smiling), generally electromyographic (EMG) activities which can affect the EEG recordings. The further the muscle is located from

the electrodes, the less is the impact on the EEG signal. For example a movement of toes has minute impact on EEG recordings (from the brain) whereas a shoulder movement has huge impact. Another activity, which is a dominant contaminant of EEG readings, are eye-generated potentials. Eye blinks, or moving eyes up down and left right, caused reflexively or intentionally, produce electrical potential detectable from EEG electrodes because the human eye functions as a dipole. The human eye has a potential difference between the anterior side (Cornea) and the posterior side (Retina). The Cornea has a positive polarity with respect to Retina, which has a negative polarity in turn. Thus, when a movement of the eyeball occurs, the polarity of the back or the front of the eye, affects the adjacent electrodes.

The significance of eye blinks and eye movements lies in the fact that they exhibit a relatively large amplitude compared to brain signals. Another factor is the frequency of each; eye blinks occur multiple times per minute whereas eye movements occur several times per second. Potentials stemming from eye movements are known as electrooculogram (EOG) events. Cardiac events i.e. heartbeat, also affect the normal cerebral signals on the EEG. Specifically, electrodes that are placed close to an artery or a vein, are more vulnerable to this artefact, which is easy to misinterpret because its shape is quite similar to a spike. Such events are termed cardiac events and recorded by Electrocardiography (ECG). Again, the constant appearance of this artefact with a low frequency though, makes it a target for artefact rejection techniques.

- 2) **Extrinsic Artefacts:** These are artefacts introduced from any source outside the body. Displacement of the electrodes due to head movement causes big spikes on the EEG signals. Furthermore, poor grounding affects the EEG with the aforementioned 60 Hz or 50 Hz power line noise. In addition to that, poor contact of the electrode with the skin, and electromagnetic interference from surrounding devices are some examples of extrinsic artefacts.

#### **4.2.2.2 Artefact rejection techniques**

- 1) **Technical approaches**
  - “Raw” rejection: The first way to deal with artefacts is based on visual inspection of EEG recordings. If artefacts are dominant over normal EEG signals, then even entire segments of time series recordings are severely affected from noise and are removed completely from the waveform file. But this will

likely result in erasing data that otherwise would be valuable. For this method, a lot of experience is required in order to detect and distinguish specific artefacts from EEG patterns. In many cases, this task is assigned to skilled professionals in the field. So, this method of artefact rejection is considered an experience-based method.

- Montage modification: Different ways of displaying signals recorded from the electrodes i.e. different montages, or even modifying the setup of a montage, can avoid displaying certain artefacts or even reduce the impact of the artefact [41].

## 2) Computational approaches

There are two algorithms that were investigated in the context of this work, Signal-Space projection (SSP) utilized by Brainstorm [42] but more closely Independent Component Analysis (ICA) utilized by EEGLAB. ICA [43] is a data or source separation method, which was tested following the official tutorials<sup>3</sup> of the authors, in order to include it in a future version of a complete system as in Fig. 1-1. The separation of data is based on the assumption that the different components or sources that a signal consists of, are statistically independent and that their distribution is not Gaussian. Provided the distribution of data is non Gaussian, somewhere in the signal, the variance of the data from the different components is higher or lower, but most importantly different, with respect to the other components' variances. So, given that these variances emerge from independent components (statistically independent components have independent variances) then ICA tries to separate those independent components, using weighting matrices, which are constructed with the goal of finding when those components are maximally independent i.e. having the highest entropy. Below are some pictures illustrating the application of this algorithm on actual data using the EEGLAB tool.

---

<sup>3</sup> [https://eeglab.org/tutorials/06\\_RejectArtifacts/RunICA.html](https://eeglab.org/tutorials/06_RejectArtifacts/RunICA.html)

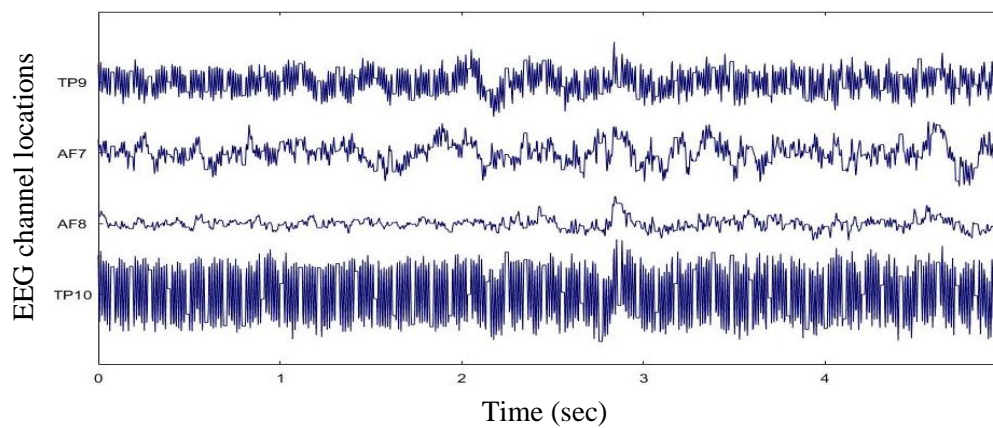


Fig. 4-8: EEG data before ICA.

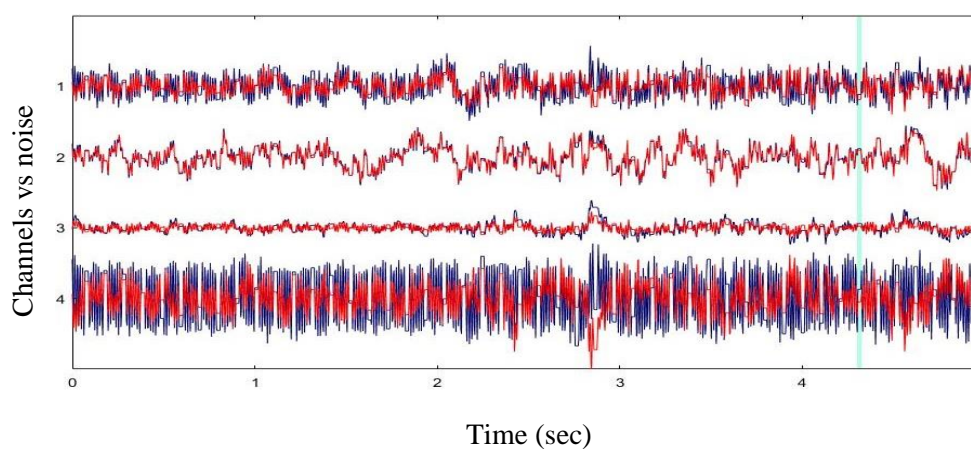


Fig. 4-9: EEG data before (blue) and after (red) ICA.

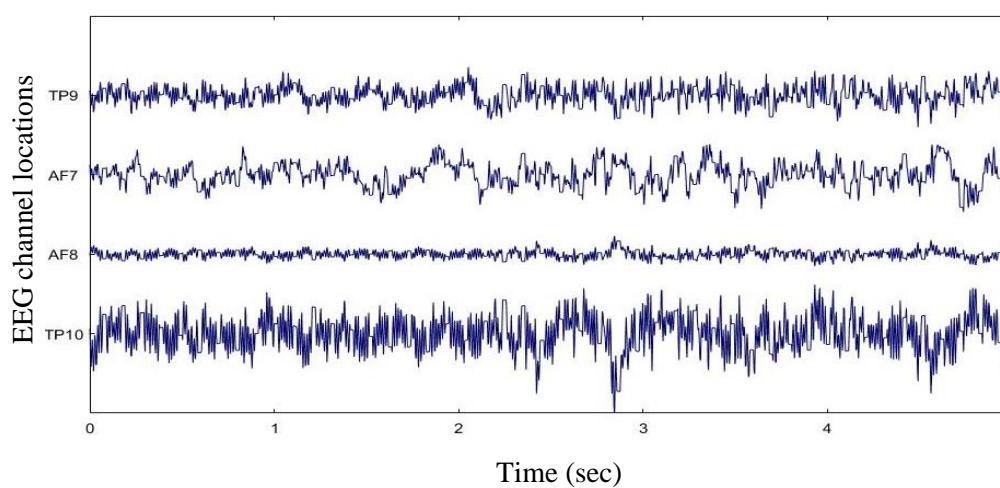


Fig. 4-10: EEG data after ICA.

Fig. 4-8 depicts the data of four EEG channels (TP9, AF7, AF8, TP10) versus time. Applying the ICA algorithm to these channels, the effect of the component removal is shown in Fig. 4-9 presenting the data before (blue colour) and after (red colour). As can be seen, the first and the last channel were the most affected channels by noise, and the EEG channel data is cleaner after one pass of ICA. Fig. 4-10 shows the clean data after running the ICA once and removing one component. Essentially, Fig. 4-9 is the combination of Fig. 4-8 and Fig. 4-10, but the figures are provided separately for a clearer viewing.

Although ICA is an efficient algorithm, the time needed for its execution is a critical challenge for a hardware implementation. However, a variation of the algorithm, Automatic Wavelet Independent Component Analysis (AWICA) described in [44] can achieve high percentages of artefact removal automatically, thus making it a promising technique. An in-depth review of algorithms for artefacts rejection can be found in this review [45].

### 4.3 Feature extraction methodology

#### 4.3.1 Fast Fourier Transform vs filter bank

Provided the signals are clean, then the next step involves building up the feature extraction part of the system. A way for examining the underlying frequencies encapsulated in a signal must be defined. The method opted for this work is the Fast Fourier Transform (FFT) algorithm. An improvement of the Discrete Fourier Transform (DFT) is the FFT which was designed by Cooley-Tukey [46] to perform the same task more efficiently. The very first thing which must be specified when an FFT is applied to a signal is the number of points or the frequencies that will be examined. That is to say, a 64-point FFT calculates the individual frequencies from 0 up to 64 levels, if the sampling frequency is equal to the number of points (see next paragraph). The number of FFT points one can calculate is not arbitrary in contrast with the number of DFT points which is arbitrarily defined. While the number of DFT points can be any integer number, the number of FFT sample points is restricted to a sequence length that is a power of two (2, 4, 8, 16 etc.). This restriction is due to the recursive structure of the algorithm, which uses lower order DFTs to compute higher order DFTs. For example, an 8-point FFT is calculated using recursively two 4-point and four 2-point DFTs. More details on the algorithm can be found in [47].

An issue that arises from the values of sampling frequency and the number of FFT points is the frequency resolution. When the number of FFT points is higher than the sampling

frequency, then the frequency resolution is finer. That is, with finer frequency resolution, not just integer values of frequencies can be studied e.g. 1 Hz, 5 Hz but also intermediate values like 1.5 Hz, 4.37 Hz etc. The formula for frequency resolution is given by the following equation and is measured in Hz/bin

$$f_{res} = \frac{f_s}{N}, \quad (4-1)$$

where  $f_s$  is the sampling frequency and  $N$  the number of FFT points. For this work, a 128-point FFT is used so as to be as close as possible to the sampling frequency namely 125 Hz, since no special value for the frequency resolution is required.

Compared to the widely used FFT algorithm, another way to isolate individual frequencies from a signal is to use a filter bank, which is applied by Thatcher et al in [24]. This is a set of bandpass filters as shown in Fig. 4-11, where each filter delivers an output according to the transfer function  $H_k(z)$ , which is a specific sub-band part of the frequencies that constitute the original signal's spectrum. In this way, the sub-bands are isolated and examined independently in the time or frequency domain if desired, after this filtering. It is also worth stressing, that although some features like phase require the conversion of a signal into the frequency domain, other features like power are calculated in the time domain too. So, this method of processing individual frequencies can be useful when a small number of bands and certain features are analysed.

In order to reduce the complexity of using large FFTs to compute the frequency spectrum, smaller FFT sizes were used for the conversion of EEG signals to the frequency domain. In the case of this thesis, multiple shortened FFTs of 128 points were applied to overlapping windowed data segments and summed. The method produces a smooth spectrum which allows efficient calculation of the qEEG variables. This approach is termed Welch's periodogram and is discussed in the next section.

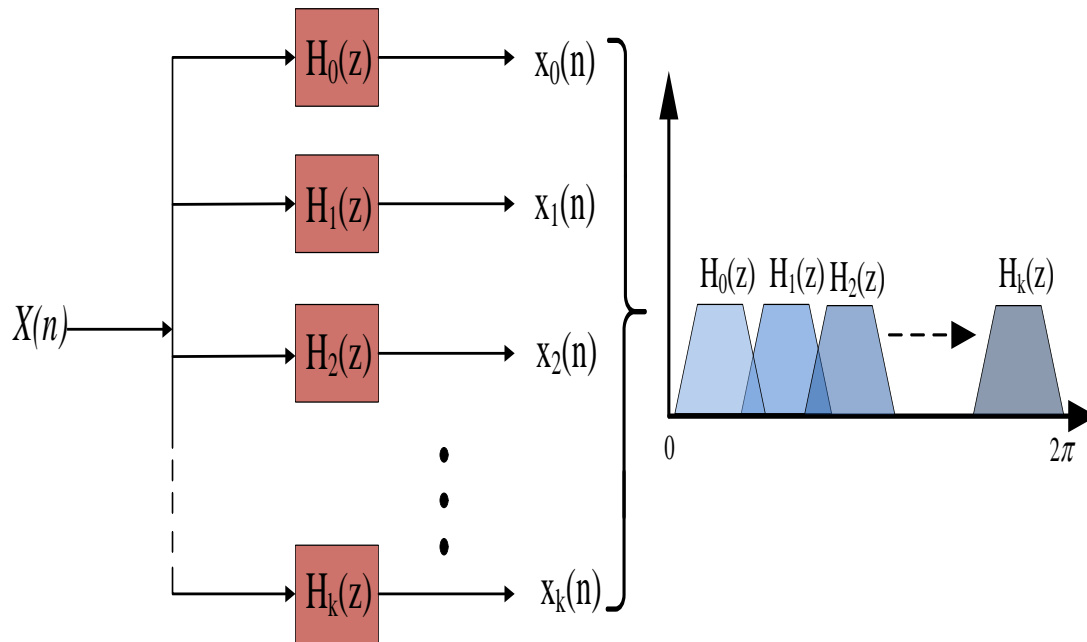


Fig. 4-11: Filter bank schematic.

#### 4.3.2 Welch's method, power and cross power spectrum derivation

As mentioned in the previous section, the FFT is used for the transformation of signals from the time domain to the frequency domain. The length of the input time domain sequence or sequences, regardless of the examined variable, was defined as 8192 ( $2^{13}$ ) data points. With a sampling frequency of  $F_s=125$  Hz, the corresponding period is at  $T=0.008$  sec which means there is a 0.008 sec time interval between each data point. Multiplying the number of data points with the sample time interval period yields 65.536 secs of data. It is a good practice when looking at EEG studies, to work with at least one minute of EEG measurements. For these measurements, the EEG data are free of artefacts. The FFT number of points is defined at 128 and also an equal length (128) Hamming window is used to further smooth the data before applying the FFT.

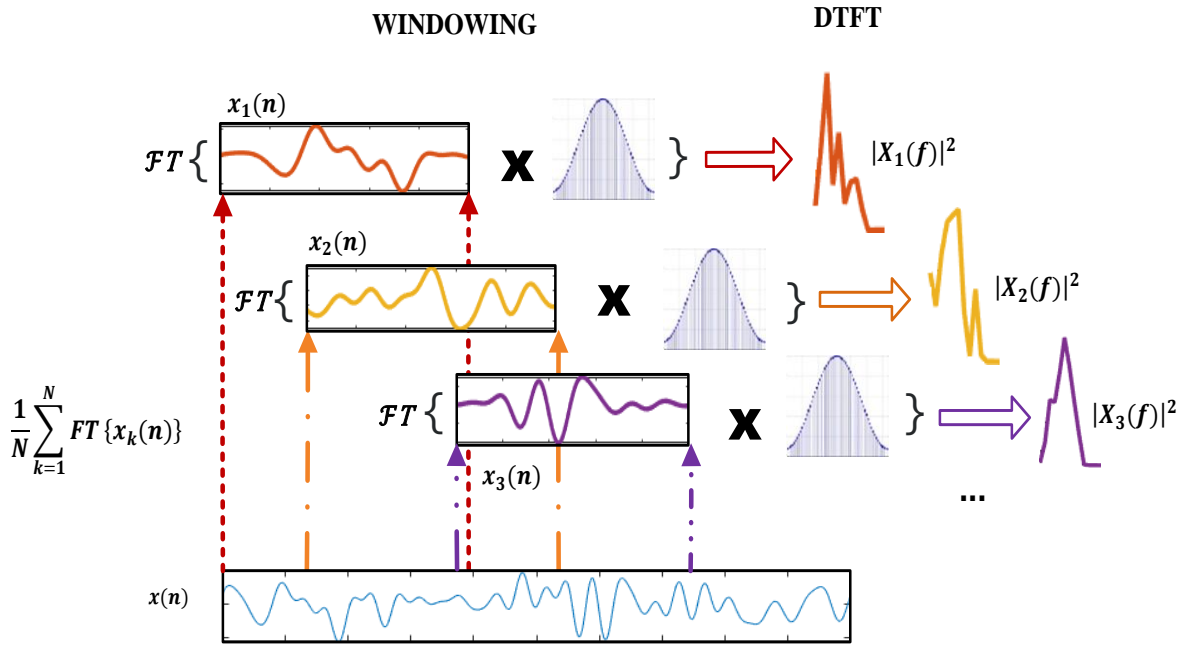


Fig. 4-12: Welch's method schematic representation.

In order to calculate the spectrum estimation or spectral density, Welch's method [48] is used. Since power spectrum is used to determine where the power of the signal is concentrated, then using Welch's method for power and cross power spectrum estimation, is an advantageous option because this method is intended to give an alternative spectrum which is more easily interpretable. This method is actually an averaging of successive periodograms with an optional overlap on the data. In essence, a periodogram is a shorter FFT section of the original signal as illustrated in Fig. 4-12, part of the original signal. Also, to be more specific, a segment-part of the original signal which is multiplied with a window function first, is called a modified periodogram. This procedure helps to remove noise, by averaging multiple shorter spectrums (periodograms) which results in a spectrum with reduced variance. Thus, the resulting spectrum is a smoother version of the full-length FFT spectrum (Fig. 4-13). The overlapping sections of the data is necessary so as to include the parts of the segments which are cancelled out by windowing, since the windows applied to the segments decay to zero at the extremities.



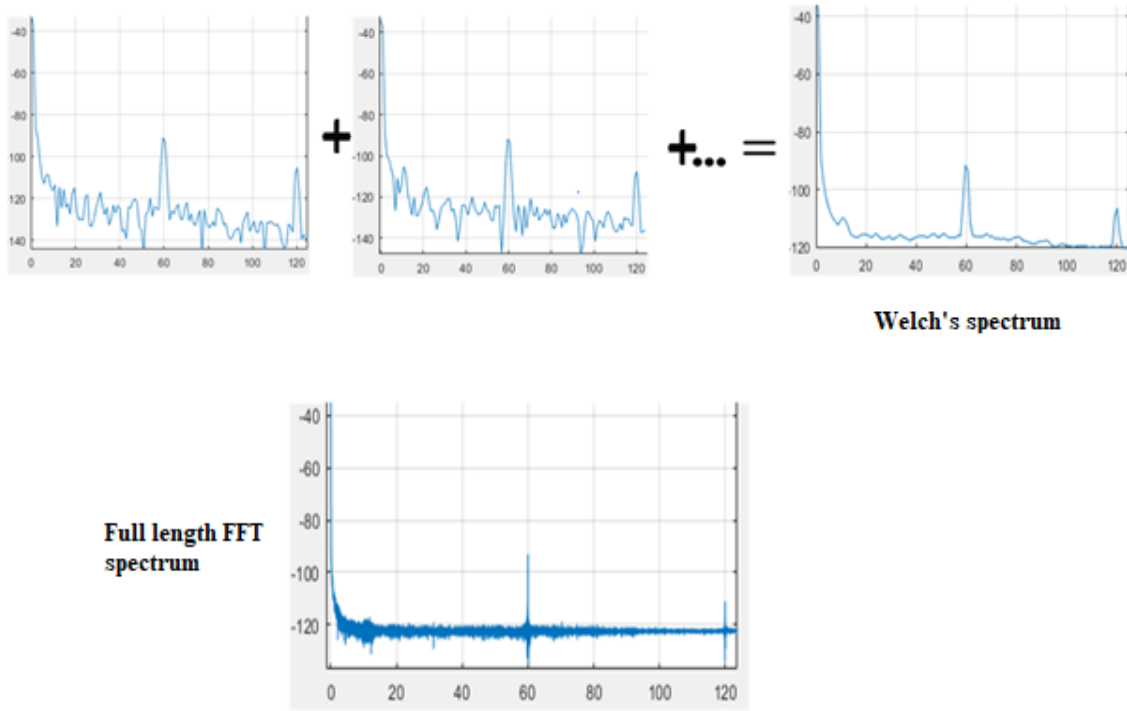


Fig. 4-13: Example spectrum derived with Welch's method vs full-length FFT derived spectrum.

Finally, the cross power spectrum is the shared power between two signals as the word 'cross' implies. With regard to its derivation, it is based on the same concept as the power spectrum, which is the multiplication of the FFT of the signals. Suppose we have two time-domain signals  $x$  and  $y$ , the formula to calculate the cross-power spectrum between  $x$  and  $y$  is given by (4-3) where the Fourier Transform of signal  $x$  is multiplied with signal's  $y$  complex conjugate Fourier Transform. Similar to this, the power spectrum is the multiplication of the Fourier Transform of the signal  $x$  with the complex conjugate of itself, and the formula is given by (4-2).

$$P_x = FT_x * conj\{FT_x\}, \quad (4-2)$$

$$P_{xy} = FT_x * conj\{FT_y\}, \quad (4-3)$$

## 4.4 qEEG features definition

As outlined in the beginning of this chapter, the frequency range of EEG signals spans approximately from 0.5 Hz to just over 100 Hz. In seminal studies of TBI, the complete spectrum of EEG signals hadn't been investigated as it is today. Thus, the area of the spectrum discussed in [24] is limited in frequencies, namely from 0.5 Hz up to 22 Hz. Consequently, the frequency bands are limited to the delta (0.5-3.5 Hz), theta (3.5-7.0 Hz), alpha (7.0-13.0 Hz), and beta (13-22 Hz) frequencies, the gamma frequency band (> 30Hz) was not included in that study. The aim of that study was to assess the dynamics of power spectral analysis to reveal distinctive qEEG features, and the effectiveness of distinguishing injured from normal subjects. A list of twenty such features or variables is provided in the study and is shown in Fig. 4-14.

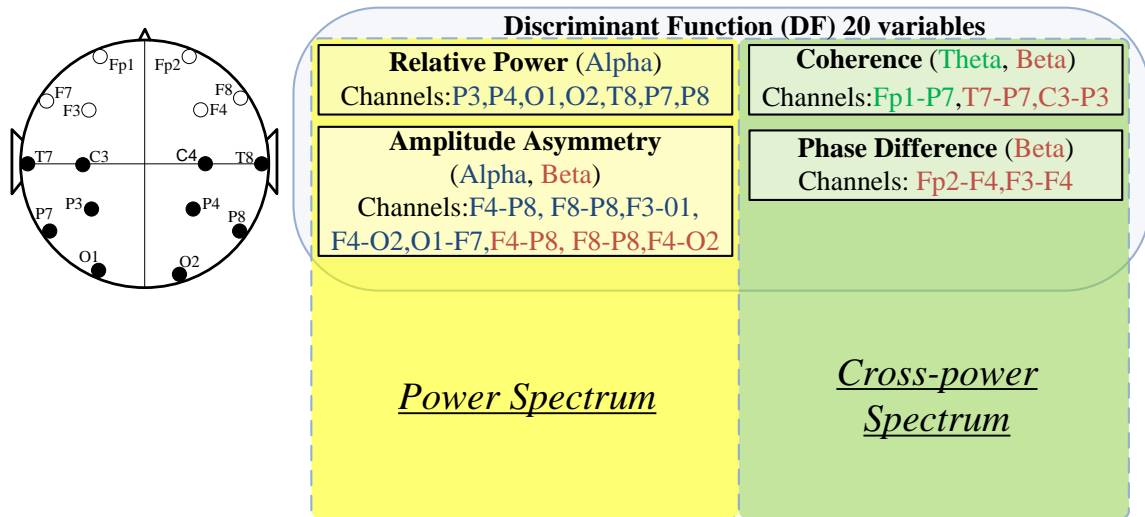


Fig. 4-14 : The Discriminant Function which was implemented on hardware.

These variables that are listed in Fig. 4-14, Relative Power, Amplitude Asymmetry, Coherence and Phase Difference, are described next.

### 4.4.1 Relative Power

The formula for calculating the relative power variable is described by (4-4) and restated as mentioned in [49]. Relative Power (RP) is the percentage of power in a specific frequency band, across the whole frequency range. That is, how much or what is the contribution of a specific frequency band to the total power.

$$RP(f_1, f_2) = \frac{P(f_1, f_2)}{P(1, 22)} * 100\%, \quad (4-4)$$

where  $P()$  indicates the power,  $RP()$  indicates the relative power and  $f_1, f_2$  indicate the low and high frequency of the targeted frequency band respectively. The  $P(1, 22)$  in the denominator is the power calculated across the whole frequency range which is examined, where in this case spans from 1 to 22 Hz.

This work computes the Relative Power for the needs of the Discriminant Function seven times, and specifically for the channels P3, P4, O1, O2, T8, P7, and P8, only in the Alpha band.

The procedure is executed in two steps.

1. Calculation of the spectral density as described in section 4.3.2.
2. Summing up the power from the frequency bins of the examined frequency band and dividing this value with the summation of power over the whole frequency range.

A major and relatively reliable finding associated with TBI is a reduction in the power of the alpha rhythm and increased theta and delta at the same time [50], [51].

A detailed MATLAB implementation for the relative power measurement can be found in Appendix 1.

#### 4.4.2 Amplitude Asymmetry

The formula for amplitude asymmetry is represented by (4-5),

$$AA(f_i) = \frac{A(f_i) - B(f_i)}{A(f_i) + B(f_i)}, \quad (4-5)$$

where AA is the amplitude asymmetry, A indicates the amplitude value at a specific frequency at a specific frequency location, and B is the amplitude value at the same frequency but different site. The amplitude of a signal for a given frequency is the square root of the power spectral magnitude. For interhemisphere site comparisons, the fraction goes (left – right)/(left+right) and for intrahemisphere site comparisons (posterior derivation– anterior derivation)/(posterior derivation + anterior derivation) or (anterior derivation – posterior derivation)/(anterior derivation + posterior derivation).

This work computes the Amplitude Asymmetry for the needs of the Discriminant Function eight times. Specifically, the AA is computed for five pairs of channels in the Alpha band

namely F4-P8, F8-P8, F3-O1, F4-O2, O1-F7, and for three pairs of channels in the Beta band namely F4-P8, F8-P8, F4-O2.

The procedure for the amplitude asymmetry is executed in two steps;

1. Calculation of the spectral density as described in section 4.3.2.
2. Calculation of AA for each frequency bin of the examined frequency band and then take the mean of those values.

Previous research suggests that Alpha Amplitude Asymmetry is associated with cognitive tasks and task difficulty [52], [53].

A detailed MATLAB implementation for calculating Amplitude Asymmetry can be found in Appendix 2.

#### 4.4.3 Coherence

The formula for Coherence between two EEG channels is defined by (4-6),

$$C_{xy}(f) = \frac{|G_{xy}(f)|^2}{G_{xx}(f) * G_{yy}(f)}, \quad (4-6)$$

where  $C_{xy}(f)$  is the coherence value for channel signals  $x$  and  $y$  at frequency  $f$ ,  $G_{xy}$  indicates the cross-spectral density between  $x$  and  $y$ ,  $G_{xx}$  is the spectral density for  $x$ , and  $G_{yy}$  is the spectral density for  $y$ . Coherence measurements, are able to determine the similarity of the signals on a frequency basis. Coherence or magnitude-square coherence takes on a value always between zero and one. i.e.  $0 \leq C_{xy}(f) \leq 1$ . A coherence value close to one indicates highly related signals (a value of 1 indicates that  $x$  and  $y$  are the same signal), and in contrast values close to zero indicate unrelated signals. In other words, coherence implies whether or not a signal is affected by another signal and the extent to which this happens.

This work computes the Coherence for the needs of the Discriminant Function three times. Specifically, Coherence is computed in the Theta band for one pair of channels namely Fp1-P7, and in the Beta band for two pairs of channels namely T7-P7 and C3-P3.

A detailed MATLAB implementation for calculating Coherence can be found in Appendix 3.

#### 4.4.4 Phase Difference

The formula for Phase Difference is given by the standard formula for phase computation, divided by the centre frequency of each band as shown in (4-7).

$$P(f) = \frac{\arctan\left(\frac{q_{xy}(f)}{c_{xy}(f)}\right)}{SC}, \quad (4-7)$$

where  $P(f)$  is the value of phase difference or phase shift between signals  $x$  and  $y$ ,  $q_{xy}(f)$  is the imaginary part of the Cross spectrum density number at frequency  $f$  and  $c_{xy}(f)$  is the real part of the cross spectrum density number at frequency  $f$  and  $SC$  is the centre frequency of the examined frequency band as provided by the author in [24]. So, similar to Coherence variable, this involves the computation of Cross Spectrum density. The domain range of the phase difference is  $-\pi/2$  to  $\pi/2$ . Coherent signals do not have random phase difference [54]. Since frequency coherence shows the similarity of signals in the frequency domain, then coherent signals are either in perfect synchronization, which means zero phase difference or their phase difference remains the same for some time.

This work computes the Phase Difference for the needs of the Discriminant Function two times only in the Beta band for two pairs of channels namely Fp2-F4, F3-F4.

A detailed MATLAB implementation for calculating Phase Difference can be found in Appendix 4.

### 4.5 CORDIC algorithm, sine and cosine

A difficult decision engineers have to make when using FFT, is the method for calculating sine and cosine values. Especially a hardware design for these functions requires a good knowledge of the underlying mathematics. There is no option of calling such a function when writing hardware description languages like VHDL or Verilog. There is this option though when writing C or C++ code. An in-depth understanding of the maths helps to decide a more efficient way to realise sine and cosine functions in hardware. Starting from the definition of these functions - in order to take the value of any arbitrary angle, one must use the length of the sides of the triangle that this angle forms Fig. 4-15. However, knowing a-priori the length

of the triangle sides formed for a given angle is impossible since there is an infinite number of them.

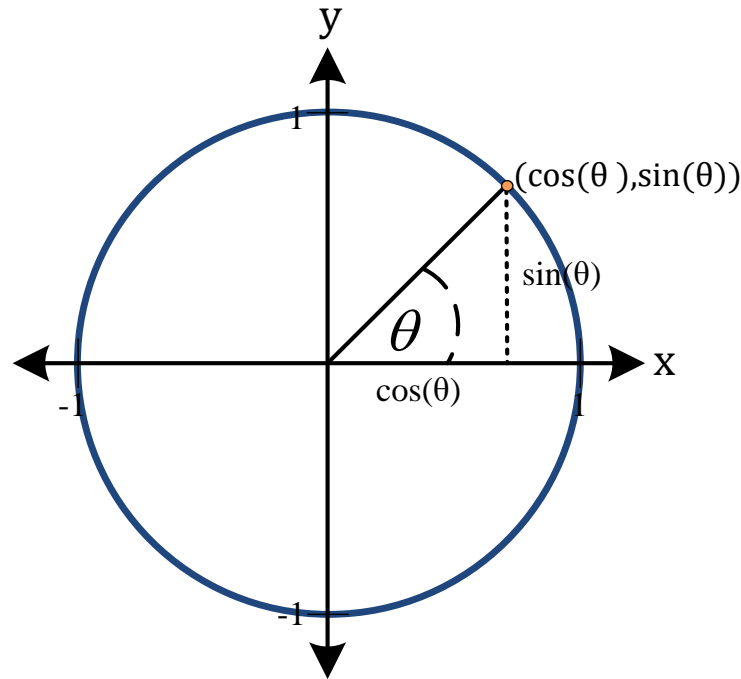


Fig. 4-15: Sine and cosine in unit circle.

Another approach for calculating sine and cosine values at any point is expressing them as a Taylor series (4-8), (4-9).

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots, \quad (4-8)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots, \quad (4-9)$$

This approach has also an infinite set of terms added and subtracted indefinitely. Consequently, for sine and cosine there is no way to have a precise method for software or hardware implementation of any arbitrary angle. Nevertheless, a good approximation of the values is what is used in practice, and for this purpose the CORDIC algorithm, which is a hardware-friendly technique was developed in 1950 [55]. The CORDIC algorithm is used to calculate a variety of functions, such as, trigonometric, hyperbolic, square roots, exponentials and

logarithms. Specifically for trigonometric functions the concept is to make fixed and computationally efficient rotations around the unit circle in order to approximate the sine and cosine value of any angle. A fixed rotation is a rotation for which is known a-priori the sine and cosine value of this angle. Computationally efficient rotations are transitions from one rotation to another using only additions subtractions and shifts i.e. hardware primitive operations, which are less expensive in terms of resources and execution time. So, if the tangent of the angles of the rotations performed by the algorithm around the circle is a power of two, then going from the current rotation (or position) to the next, becomes a shift operation. For example,  $\tan(45^\circ) = 2^0$ ,  $\tan(26.565^\circ) = 2^{-1}$ ,  $\tan(14.036^\circ) = 2^{-2}$  and so forth.

An example of how the CORDIC works can be shown in Fig. 4-16. The target angle is  $60^\circ$ . The first rotation is set at a  $45^\circ$  angle. This is less than  $60^\circ$  thus the next rotation has a positive sign, just as the conventional direction, which results in a rotation of  $45^\circ + 26.565^\circ = 71.565^\circ$ . Furthermore, since  $71.565^\circ > 60^\circ$ , the next rotation has a negative sign, therefore  $71.565^\circ - 14.036^\circ = 57.529^\circ$ . This procedure continues for a certain number of rotations. For this demonstration-example the rotations are five but in practice the number of rotations varies from thirty two to sixty four. For a more in depth understanding of the algorithm refer to [47].

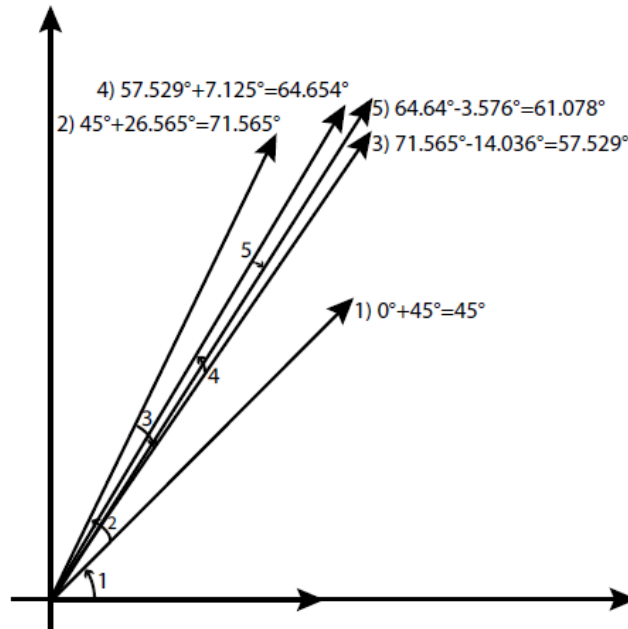


Fig. 4-16: CORDIC algorithm illustration for sine and cosine [47].

If the application requirements allow for a fixed number of FFT points, then simply storing an array for the sine and cosine values attached to a certain number of FFT points (128-256 etc.) is another way to carry out the sine and cosine value calculations. For example, a 128

point FFT needs two arrays equal in size i.e. 128 each one, with pre-stored values for sine and cosine correspondingly. This method is used to boost the time performance of the application when execution time is a priority. Although this way is helpful with regard to time execution, it's not a general case. Depending on the application needs, if the number of FFT points is not fixed, then this is not an applicable method. However, for this work, arrays with pre-stored values of sine and cosine are utilised since the execution time of the discriminant function is critical and the number of FFT points is fixed.

## **4.6 Conclusion**

This chapter has presented the set of signal processing calculations that can be combined to produce a discriminant function that may be used to indicate a possible TBI event. The emphasis has been on computationally efficient techniques that could be applied in a low power but high performance design. Using shorter FFTs, the objective was to achieve a frequency resolution close to 1 Hz, and the Welch's method was selected as it computes the power spectrum based on shorter FFTs. Furthermore, the realisation of the sine and cosine function with lookup tables is utilised due to the real-time needs of the discriminant function with emphasis given to the speed of the design. In summary, having defined the signal processing aspects and the formulas for the qEEG features, the hardware implementation of them is described in the next chapter along with the architectural interventions to implement a discriminant function design. Starting with the baseline methods, an attempt to improve the performance both in terms of resources consumption and execution time is also covered.



# 5

## Implementation

---

### 5.1 Introduction

The goal of this chapter is to have a hardware implementation of the discriminant function described previously, that can operate in real-time. This discriminant function comprises the four features-variables which are Relative Power, Amplitude Asymmetry, Coherence and Phase Difference developed as hardware accelerators in the FPGA design. In essence, the hardware implementation is the realisation of these variables. Starting with the design tools used for this realisation, this chapter presents the steps towards an efficient hardware design with real-time requirements and the sequential improvements as the design progressed.

### 5.2 Data description and data availability issues

This work is intended to use EEG readings from the brain of healthy or injured subjects in order to detect a TBI. This requires a classification algorithm to be trained so it can make such predictions. By the time this thesis was written no publicly TBI data were available for training a ML algorithm. In this work, a dataset was used with EEG recordings from sample data provided by the OPENBCI GUI. This file format is a simple text file. These data samples are not TBI recordings but rather recordings taken from people performing various tasks, like blinking and jaw clenching. Since there was no TBI data publicly available, then the aforementioned data was used for run-time performance and testing purposes. This set of data is considered to be free of artefacts, and is used for testing the hardware design implementation.

### 5.3 Design tools

Three basic tools were used for the hardware implementation of this work that are all part of the XILINX Vivado Design Suite<sup>4</sup>. This is a software suite produced by XILINX for synthesis and analysis of HDL (Hardware Description Language) designs. Vivado is a design environment for FPGA products, and has three main tools, Vivado HLS, Vivado IP Integrator and Vivado SDK, each fulfilling the needs for the different stages of the development process. In addition to the XILINX tools, MATLAB was used for the modelling of the discriminant function, where the corresponding code can be found in Appendices 1~5. The full list of tools that were used in the development of this work is detailed in Table 5-1.

Table 5-1: Design tool list.

Tool name	Tool description	Tool version
MATLAB	A programming and numeric computing environment developed by Mathworks.	R2019a
XILINX Vivado HLS	Allows high level programming languages to be synthesized to RTL implementation.	2019.1
XILINX Vivado IPI	A block design environment for FPGA IP products from XILINX.	2019.1
XILINX SDK	A tool for creating software embedded applications for XILINX's devices.	2019.1

#### 5.3.1 Vivado HLS

Vivado High Level Synthesis (HLS) converts C, C++, or SystemC source code to a Register Transfer Level (RTL) implementation, with both Verilog and VHDL codes generated automatically. Vivado HLS's compiler enables C, C++ or SystemC programs to be directly targeted onto XILINX devices without the need to manually create RTL design code. The tool allows developers to take advantage of the capabilities of high-level programming languages. An advantage of the HLS tool is that it enables fast code development and more areas to be investigated by the developers, like implementing neural networks on hardware which would otherwise be a fairly difficult task. Developers working on a higher level of abstraction benefit from the simplicity of writing code like C or C++ and the tool features which are directives.

Directives are special commands for the HLS compiler that carry out the interface definition and optimizations of the produced RTL code or hardware accelerator. The directives are specified as #pragmas [56] inside the code and interpreted accordingly from the compiler. Interface definition directives have to do with the accelerator's communication with the

<sup>4</sup> <https://www.xilinx.com/products/design-tools/vivado.html>

external world, i.e. what protocol is used for the initialization of the accelerator and what protocol is used for its data transactions. On the other hand, optimization directives have to do with the latency, the throughput, and the resource utilization of the accelerator. Since, the HLS tool generates a HDL code, there are some restrictions about writing hardware oriented C or C++ code. System calls like `printf` or `scanf` functions are not allowed, and in addition, no recursion or dynamic memory allocation (`malloc`) can be used in the code since these cannot be synthesized. Once the C code has been written and tested, then the Synthesis step within the HLS flow is the generation of the RTL code (VHDL and Verilog). Finally, the RTL code (synthesized C code) is then packaged as an IP (Intellectual Property) block ready to be used by other XILINX tools like System Generator and Vivado IP Integrator. The design flow for the HLS tool is illustrated in Fig. 5-1.

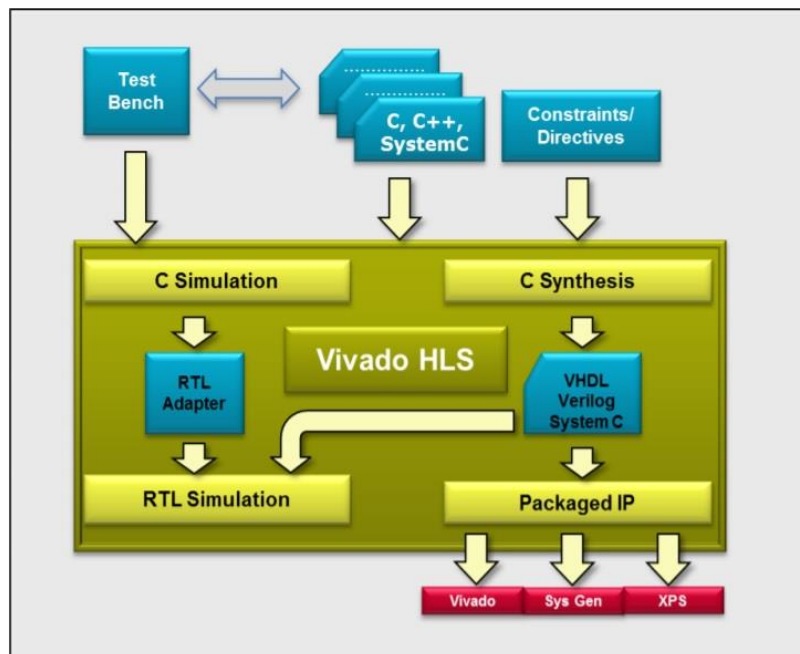


Fig. 5-1: Vivado HLS design flow [57].

### 5.3.2 Vivado IP Integrator

Vivado IP Integrator (IPI) is the main design tool in the XILINX Vivado Design Suite. Vivado IPI is a design environment for synthesis and analysis of HDL designs for FPGA products from XILINX. Vivado IPI allows hardware developers to create complex system designs, by instantiating and interconnecting IP cores through its design canvas GUI, in the case of this thesis. IP in hardware terminology stands for Intellectual Property, and is a block

of logic or data, that is used in making FPGA or ASIC (Application-Specific Integrated Circuit) applications. Ideally an IP core should be entirely portable, which means that it should be easily integrated into any vendor technology or design methodology. Every block design created with this tool will eventually be translated to hardware primitives, through the processes in Fig. 5-2. The output of this tool is a configuration file targeting a specific FPGA board, called a bitstream. First the RTL design is synthesized, which in the case of Vivado IPI is the process of creating a netlist file out of a specific RTL design. This netlist file is an electronic depiction of the RTL design, or a set of instructions that describe the electronic circuit connectivity which corresponds to the RTL design. After synthesis, the implementation step will map the synthesized design to physical device FPGA resources through place and route, and then the final step is the bitstream generation. After completing these steps, the bitstream is ready for downloading onto an FPGA board, but, if the design incorporates the CPU i.e. the Processing System (PS) of a modern System on Chip (SoC) then it must first be exported to the the Vivado Software Development Kit (SDK) in order to write the required driver code.

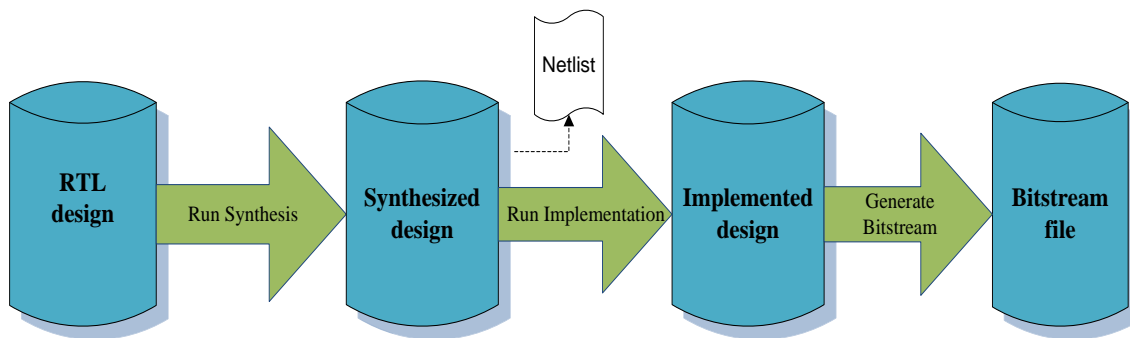


Fig. 5-2: RTL to Bitstream flow.

### 5.3.3 Vivado SDK

Vivado Software Development Kit (SDK) is an Integrated Design Environment (IDE) for creating embedded software applications. The Vivado SDK targets XILINX devices which integrate the Processing System (CPU & peripherals), and the Programmable Logic (PL) onto the same FPGA die fabric such as the Zynq Ultra Scale SoCs<sup>5</sup>. The SDK is an Eclipse-based IDE with a C/C++ editor and compiler, comprising all application build tools along with a debugging environment. SDK supports two types of applications. The first is standalone or

<sup>5</sup> <https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html>

bare metal, which means, applications that do not depend on an Operating System (OS) and run directly on the processor. While the second are OS-based applications that usually run on Linux. In the context of this thesis, a bare metal application was developed, in C code. This C driver code, contains all of the methods necessary for the operation of the hardware that has been created.

Every component that is described in the hardware specifications which is imported to the SDK tool, has an Application Programming Interface (API) attached to it. For this work, the HLS tool automatically creates the API for each accelerator when the IP is packaged. All necessary functions for interaction between the components are provided by the API. For this purpose, a project that is created with the SDK and targets a specific hardware design, must incorporate a Board Support Package (BSP). Inside the BSP there is a collection of libraries and driver functions (API) that are available for use from the driver application under-construction. These helper functions for the processor enable the initialization of the IP cores and the memories contained in the design. The helper functions also control the starting and halting of them, and the coordination of data transactions between them.

For every module of the hardware design there is an address space in memory assigned to it. By simply reading and writing to the registers in this address space of the module, the processor is able to identify the instance of the module, and then to begin storing and fetching information required for its operation. The assigned address space for every IP core begins with a base address and every register (control, status etc.) is accessed by providing the offset of the register from the base address. This process is more or less automated by the functions provided by the API, thus making the development of a driver code even easier for developers. Lastly, the SDK tool will provide an Input/Output (I/O) interface for communication with the hardware on the board. This I/O communication is done through the serial ports that are used for transferring data to and from the board and the host PC.

## 5.4 XILINX Zynq UltraScale+ MPSoC ZCU 104

The XILINX Zynq® UltraScale+™ MPSoC ZCU104 Evaluation Kit (Fig. 5-4) is a flexible prototyping platform with high-speed DDR4 memory interfaces, an FMC expansion port, multi-gigabit per second serial transceivers, a variety of peripheral interfaces, and a FPGA fabric for customized designs. The evaluation kit is based on a XCZU7EV MPSoC that pairs Programmable Logic (PL) with a Processing System (PS) based on a quad-core ARM®

Cortex™-A53 applications processor and a dual-core Cortex-R5 real-time processor. The architecture for this system [58] is illustrated in Fig. 5-3, but for an in depth description of the architecture refer to [59]. The tables below show the available on-chip resources for the PL as they appear in the tools reports as shown for Vivado HLS (Table 5-2) and Vivado IPI (Table 5-3) respectively. These table reports are useful in the context of this work, and specifically for the improvement results which are discussed in the next chapter.

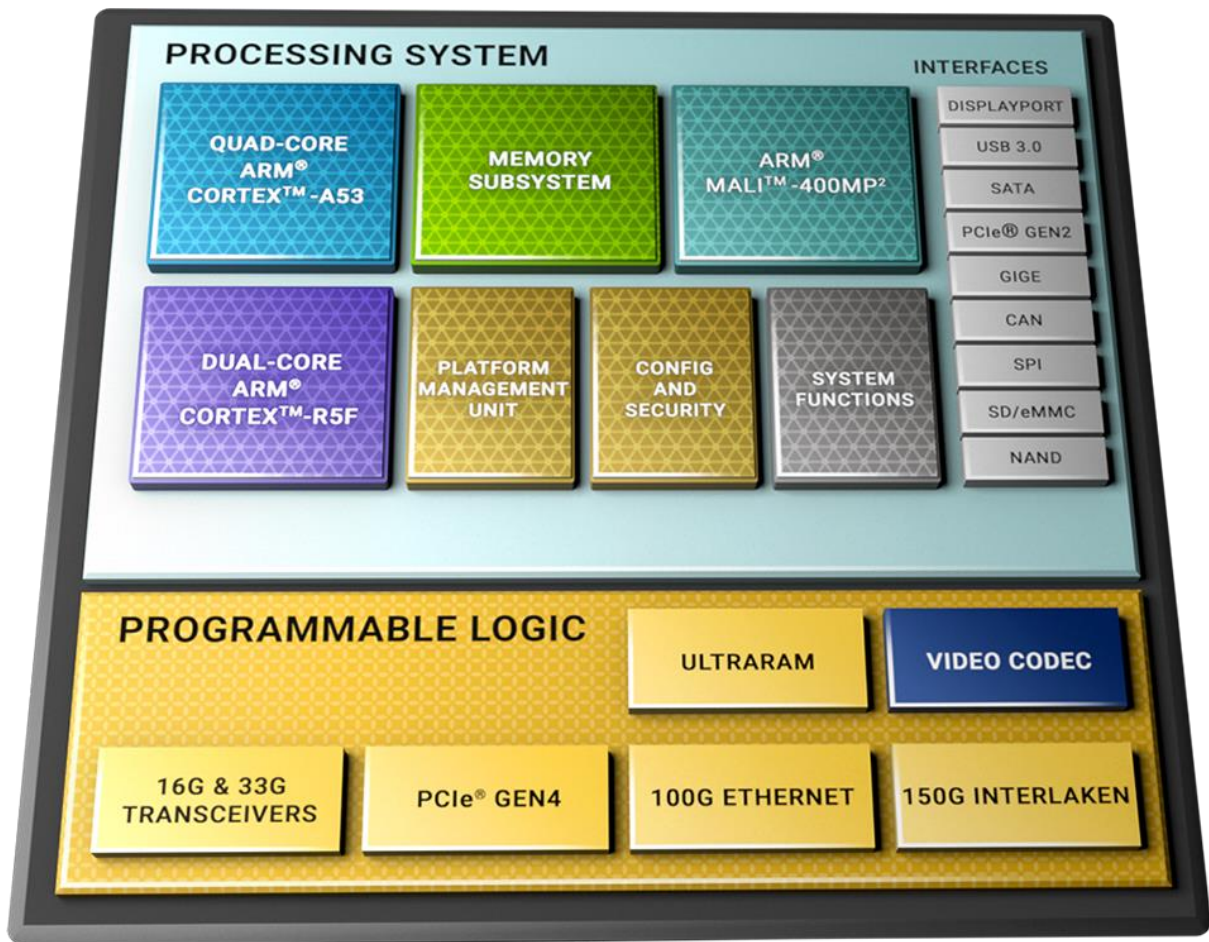


Fig. 5-3: ZCU MPSoC system architecture [58].

Table 5-2: Vivado HLS utilization report for ZCU104.

BRAM_18K	DSP48E	FF	LUT
624	1728	460800	230400



Table 5-3: Vivado IPI utilization report for ZCU104

CLB LUTs	CLB registers	CARRY 8	F7 MUXES	F8 MUXES	CLB	LUT as Logic	LUT as Memory	Block RAM Tile
230400	460800	28800	115200	57600	28800	230400	101760	312

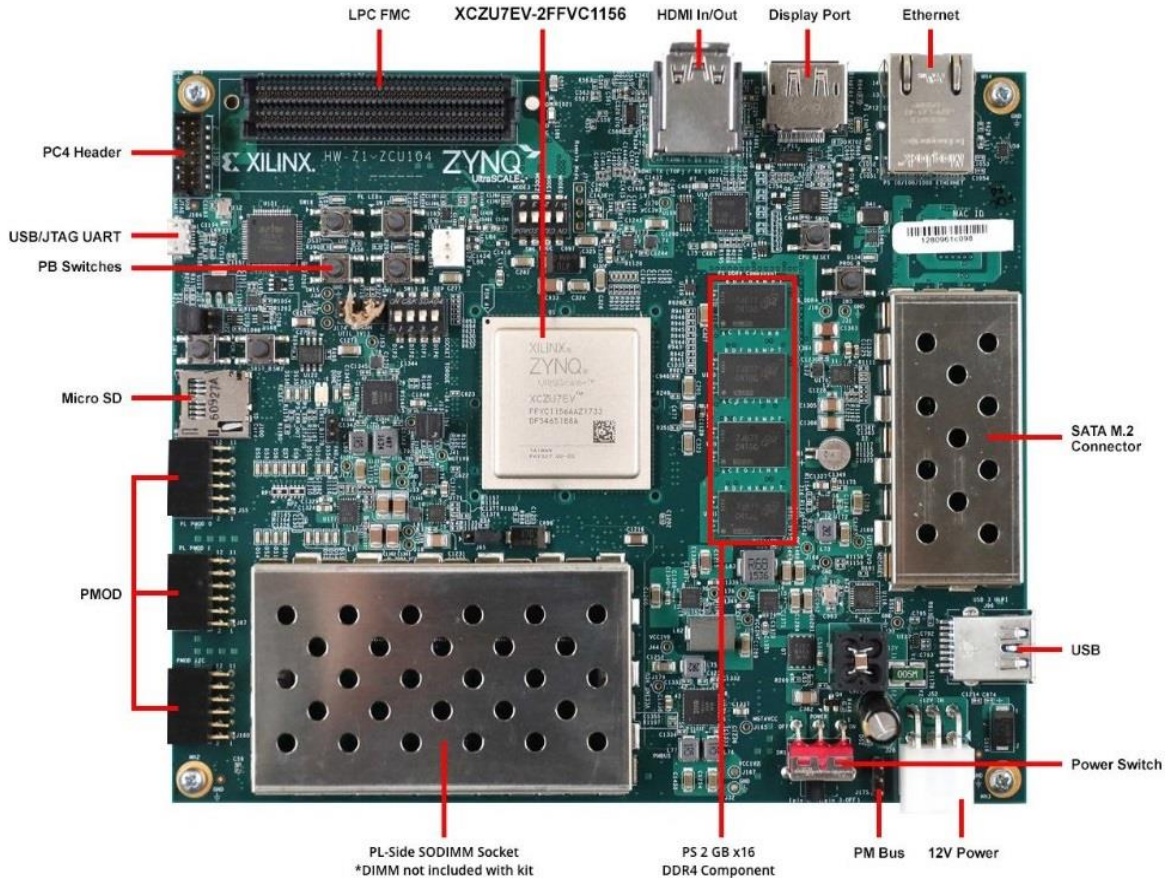


Fig. 5-4: ZCU104 Evaluation Kit [60].

## 5.5 AXI protocol

The Advanced eXtensible Interface (AXI) is a communication protocol between hardware IP modules. This protocol simply sets up the rules for how different modules on a chip communicate with each other, requiring a handshake-like procedure before all data transmissions. AXI is a standard through which, the modules can interface to the outside world. When a whole product family follows the same rules for the communication of the components of their system, then maintenance, portability and debugging of the design becomes very easy. The latest protocol that is adopted by XILINX is termed AXI4.

There are three types of AXI4 interfaces:

- AXI4, for high-performance memory mapped transactions.
- AXI4-Lite, for simple, low-throughput memory-mapped communication (for example to and from control and status registers).
- AXI4-Stream, for high-speed streaming data transactions.

The concept of communication between modules is formed by the AXI Master and the AXI Slave. The AXI Master is the module that initiates a transaction, and the AXI Slave is the module which responds to the initiated transaction (Fig. 5-5). A transaction can be either a read or write transaction.

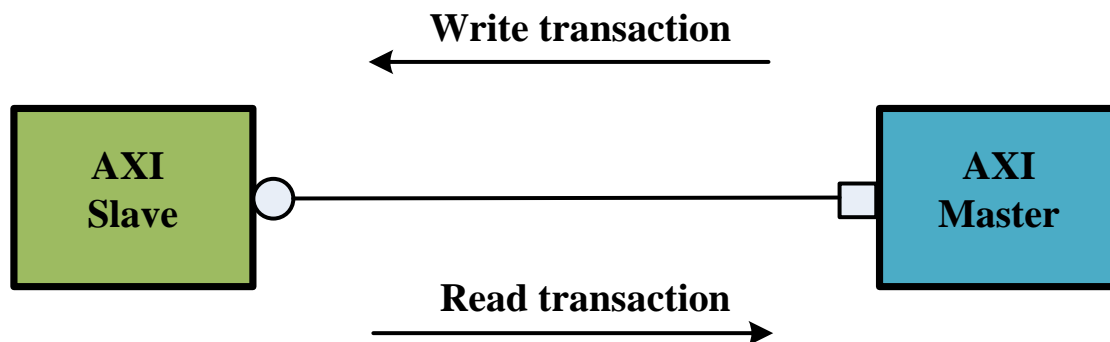


Fig. 5-5: AXI Master & AXI Slave components.

Let's consider the example of when the AXI Slave is a block of memory and the AXI Master is a DSP module. Then in the case of memory-mapped transactions, the AXI Master has to specify the address from where it should read the data in the memory, or where it should write the data respectively. This is not the case for streaming transactions, where the data movement between modules is a flow, and no address is required to be specified. There are cases, that there is no read or write operation, just a data transmitter and a receiver. Receiving data from the network or from a HDMI cable are examples of streaming transactions. Therefore, the implementation of a streaming interface is much simpler and faster than the memory-mapped interfaces. In general, an interface is a grouping of signals that share a common function. So, the above model Fig. 5-5 of the AXI Master and AXI Slave components containing signals is as follows.



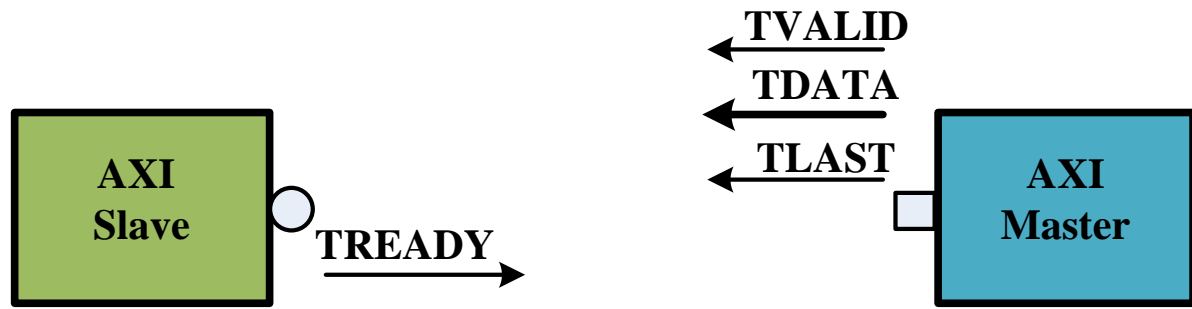


Fig. 5-6: AXI4-Stream interface.

For example, the AXI4-Stream interface and its functioning signals are shown in Fig. 5-6. These signals are the basic signals that all AXI Stream Slaves and AXI Stream Masters incorporate. There are also more optional signals that are not considered in the context of this work. The valid and ready signals compose the handshake mechanism, required for a transmission. When the AXI Slave is ready for a transaction, it informs the AXI Master with the TREADY signal and the AXI Master sends the data on TDATA bus when the TVALID and TREADY signals are high in the same clock cycle. Finally the TLAST signal goes high when the last piece of data has been transmitted, so the Slave knows that this is the end of a transaction. In this thesis work, the AXI4-Stream interface plays a significant role.

## 5.6 DMA

Direct Memory Access (DMA) IP is a dedicated hardware component of modern computer systems that allows data transfers between the main memory of the system and the peripherals without the involvement of the CPU (Fig. 5-7). DMAs are used as a “detour” around the CPU, for read or write data transactions to and from the main memory, without having to interrupt the CPU throughout the operation. This can save significant time for the execution of other tasks until a transaction completes. If the CPU is occupied throughout a transaction with the main memory then valuable computation power is stolen from other resources. Using DMAs, the involvement of a CPU is restricted to the start and the end of the transaction, thus making the DMAs an effective solution for intensive data transactions, such as real-time streaming applications as is the case for this work. For a real-time application, the DMA blocks offer a faster implementation, and low-cost resource consumption, compared to using the CPU.

In order for a DMA to perform high-speed transactions to and from the main memory, the address of the data in memory, the destination of the transferred data, and the length of the transaction is all that is required. This is the only information the CPU needs to process as it configures the DMA with an initial address for the data, the destination, the length of the transaction, and then it receives an interrupt signal when the transaction is completed.

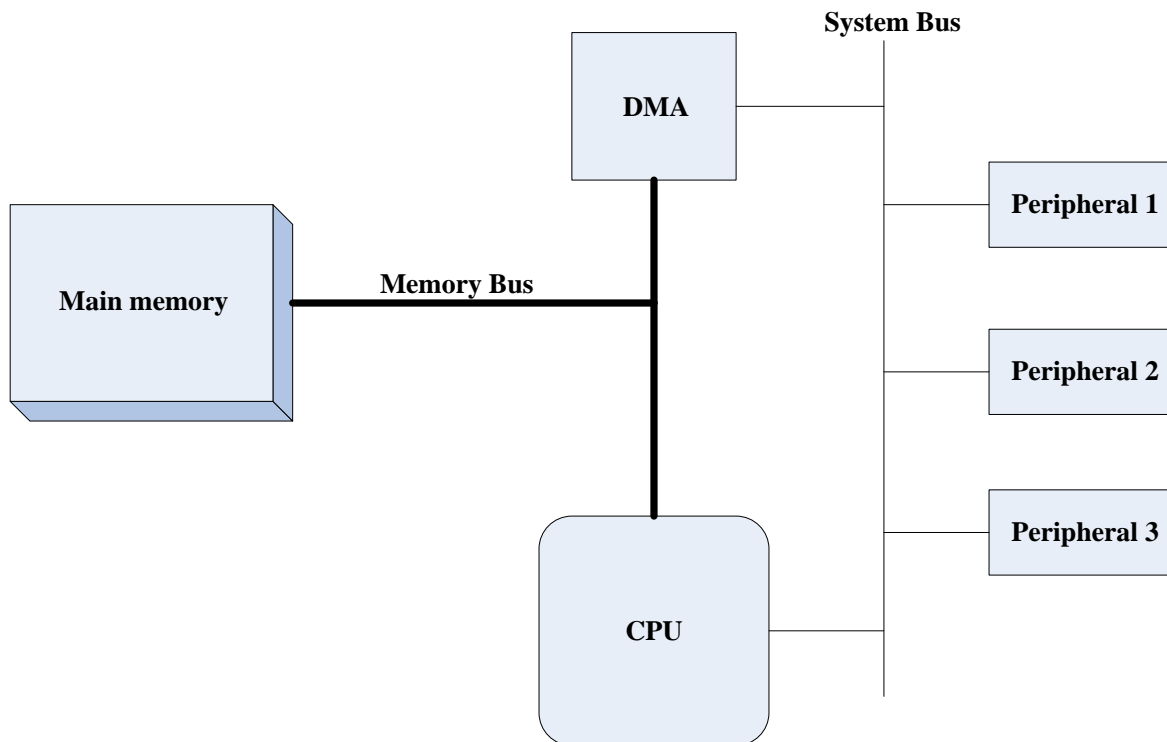


Fig. 5-7: A computer system containing a DMA.

XILINX provides an AXI DMA IP for its products, and this IP was used in the context of this work. This IP has AXI memory-mapped channels to interface the memory, which is the DRAM of the ZCU104 board, and some AXI4-Stream channels to interface to the HLS IP accelerators created for the discriminant function. So, when EEG data are streamed into the ZCU104 board from an external acquiring device and stored in the memory, then the DMAs take over and transfer the data to the accelerators for processing, thus giving the discriminant function the real-time form and performance. It must be noted that using an EEG headset and streaming the data to the ZCU104 board has not been performed as part of this work. Instead EEG channel data (see 5.2) was stored directly into the DRAM memory of the ZCU104 for testing the discriminant function design.

## 5.7 MATLAB Code for behavioural modelling

In order to develop the code for hardware, it is good practice to have a behavioural model in software (Fig. 5-8). Here, the MATLAB code presented in Appendices 1~5 is the software model used for this work. MATLAB script functions 1~4 are the guides for the HLS code and script 5 is the guide for the SDK code. The fifth script in essence is the container of the discriminant function, which invokes the other four scripts that have been implemented as MATLAB functions. Thus, scripts 1~4 are the counterparts for the hardware accelerators that will be implemented as FPGA components for the PL on the board, while script 5 is the base for the standalone driver code that will run on the ARM processor of the processing system on the board. So, the outcome of this thesis is the conversion of these MATLAB functions to a hardware design suitable for FPGA integration. The hardware design is implemented as HLS IP block designs that make up the discriminant function and are managed by C software code running on an ARM processor. In the next section, both the HLS and SDK code analysis are discussed.

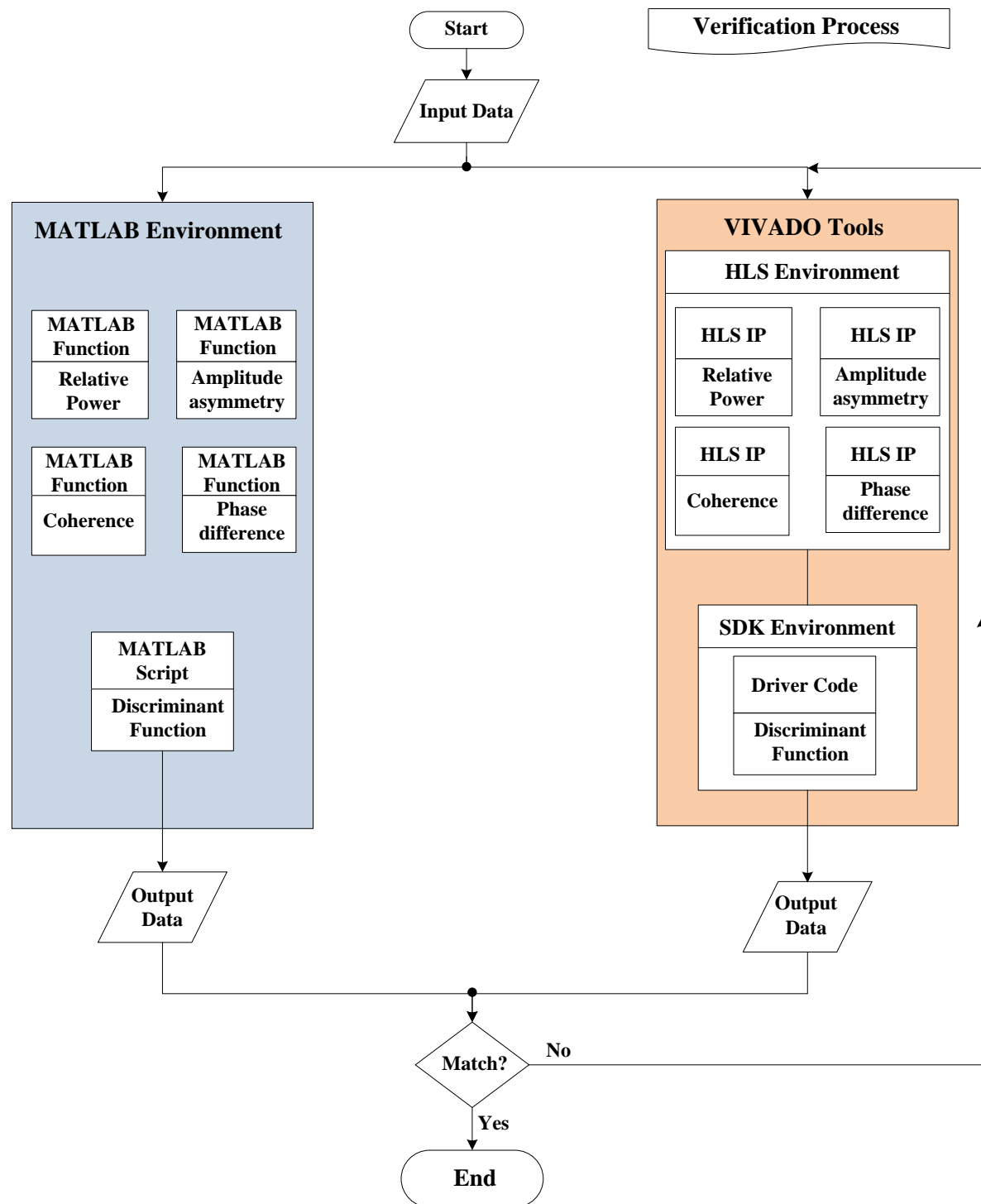


Fig. 5-8: Hardware design verification process.

### 5.7.1 HLS Code

Looking at the MATLAB scripts in Appendices 1~4, some notable common characteristics can be described with the flow chart diagram in Fig. 5-9. All of the scripts start with variable initializations which have to do with the length of the input data, the sampling frequency etc.

followed by arrays initializations. Both of these initializations in HLS C++ code are defined as global macros in the case of variables, and global arrays in the case of vectors. Since not all of them are used within the same scope, and their values do not change no matter how many times the accelerator is used, they are constants. The initializations follows the spectral calculations inside a big for-loop where the number of iterations is 127 times defined by (5-1).

$$N = A * \left( \frac{\text{input}_{\text{length}}}{\text{FFT}_{\text{points}}} \right) - 1 = 127, \quad (5-1)$$

where  $\text{input}_{\text{length}} = 8192$ ,  $\text{FFT}_{\text{points}} = 128$  and  $A = 2$  due to 50% overlap according to the Welch's method (see section 4.3 Feature extraction methodology). Inside the for-loop, some key operations are taking place in each iteration:

**Key operations:**

Step 1: Read a data length of 128 EEG samples (equal to FFT size) from the input stream.

Step 2: Windowing data samples using Hamming function.

Step 3: FFT of the windowed data.

Step 4: Spectra computation which breaks down to

- a. Spectrum folding. i.e. conversion from two-sided to single-sided spectrum.
- b. Squaring FFT complex numbers.

Step 5: Feature calculations – Relative Power, Amplitude Asymmetry, etc.

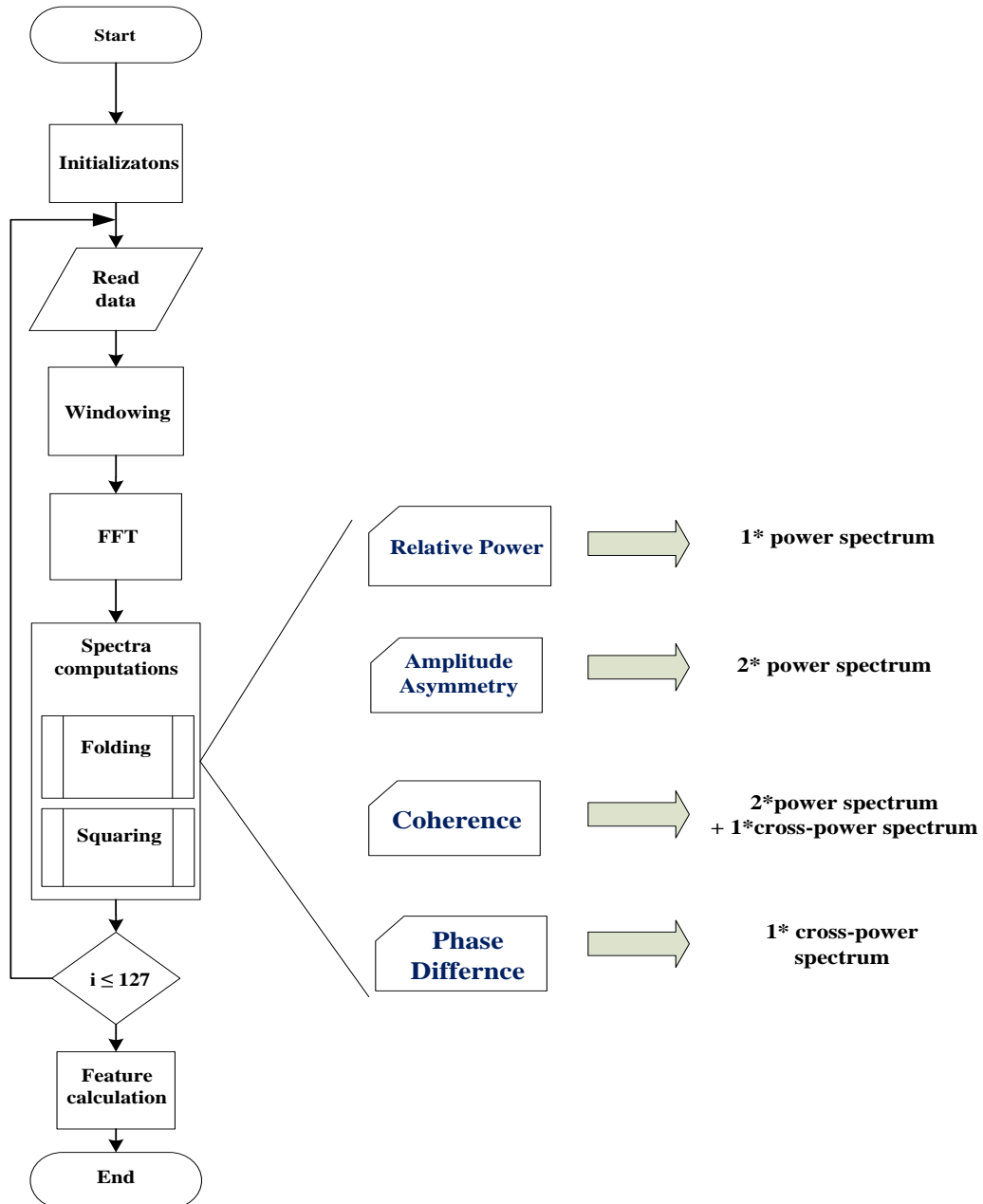


Fig. 5-9: Algorithm flow diagram for HLS Accelerators.

Although this is a generic illustration of the procedure for each accelerator, the fourth step is somewhat different. This can be seen in the right part of Fig. 5-9. Relative Power computes the power spectrum for a single channel, Amplitude Asymmetry computes the power spectrum for two channels, while Coherence computes the cross-power spectrum between two channels and the corresponding power spectrum of each one. Finally, Phase Difference computes only the cross power spectrum between two channels. The Steps 4.a and 4.b are the same in the case of cross-power or power spectrum calculations; what changes is the multiplication of the complex numbers as in (4-2) and (4-3). The code completes the feature calculation step, by

applying the equations (4-4) ~ (4-7). Each equation is applied to every frequency bin of the targeted frequency band and then their average is calculated and propagated to the output.

From the HLS code, the most complex part is the FFT operation. A baseline code for the FFT obtained from [47] is shown in Fig. 5-10. This code was then modified and Fig. 5-11 shows how it was converted after the modifications.

```

void fft(DTYPE X_R[SIZE], DTYPE X_I[SIZE]) {
    DTYPE temp_R;    // Temporary storage complex variable
    DTYPE temp_I;    // Temporary storage complex variable
    int i, j, k;      // Loop indexes
    int i_lower;      // Index of lower point in butterfly
    int step, stage, DFTpts;
    int numBF;        // Butterfly Width
    int N2 = SIZE2;   // N2=N>>1

    bit_reverse(X_R, X_I); // Bit-reverse function

    step = N2;
    DTYPE a, e, c, s;

stage_loop:
    for (stage = 1; stage <= M; stage++) { // Do M butterflies stages
        DFTpts = 1 << stage; // DFT = 2^stage = points in sub DFT
        numBF = DFTpts / 2; // Butterfly WIDTHS in sub-DFT
        k = 0;
        e = -6.283185307178 / DFTpts; // -2π/128
        a = 0.0;
        // Perform butterflies for the j-th stage
        butterfly_loop:
        for (j = 0; j < numBF; j++) {
            c = cos(a); // ←←
            s = sin(a); // ←←
            a = a + e;
            // Compute butterflies that use same W**k
            dft_loop:
            for (i = j; i < SIZE; i += DFTpts) {
                i_lower = i + numBF; // index of lower point in butterfly
                temp_R = X_R[i_lower] * c - X_I[i_lower] * s;
                temp_I = X_I[i_lower] * c + X_R[i_lower] * s;
                X_R[i_lower] = X_R[i] - temp_R;
                X_I[i_lower] = X_I[i] - temp_I;
                X_R[i] = X_R[i] + temp_R;
                X_I[i] = X_I[i] + temp_I;
            }
            k += step;
        }
        step = step / 2;
    }
}

```

Fig. 5-10: FFT baseline C-code HLS implementation.

```

void fft(float X_R[SIZE], float X_I[SIZE]) {

    .
    .
    .

    for (stage = 1; stage <= M; stage++) { // Do M stages of butterflies
        DFTpts = 1 << stage; // DFT = 2^stage = points in sub DFT
        numBF = DFTpts >> 1; // Butterfly WIDTHS in sub-DFT (numBF = DFTpts/2;)
        initposition=(SIZE>>1)>>stage-1;
        position=0;
        // Perform butterflies for j-th stage
        butterfly_loop:
            for (j = 0; j < numBF; j++) {

                if(position==0){
                    c = cosinearray[0];
                    s = sinearray[0];
                }
                else{
                    c = cosinearray[position];
                    s = sinearray[position];
                }

                position = position + initposition;

                .
                .
                .

            }
        }
    }
}

```

Fig. 5-11: Modified C code of Fig. 5-10.

#### 5.7.1.1 Initial design improvement

The first improvement on the baseline FFT C-code implementation, was to change the built-in functions for sine and cosine (highlighted with blue colour box in Fig. 5-10) of the HLS <math.h> library with lookup tables. This library uses a CORDIC core to calculate trigonometric functions, but it turns out to be costly in terms of resource consumption. An attempt was made to use a CORDIC implementation from [47] for sine and cosine, with fixed-point variables. In this case, the problem occurred with the precision of the values and the hardware resource consumption. First, using fixed-point variables the loss in precision was more than two decimal places, and the number of bits needed for the custom-defined variables exceeded 32-bits. As shown in Fig. 5-9 the input data is first windowed, then a FFT is applied



to them and lastly the values are squared. Each one of these 3 steps includes a multiplication for the input data, so in total there are 3 different phases of multiplications for the input data.

At every multiplication, a fixed-point variable has a loss in precision due to rounding, so after 3 phases of multiplications (where one of these phases includes the FFT), the loss in precision was significant. In order to have a good approximation of the result that could be obtained using floating-point variables, the number of bits that would be needed for the fixed-point variables was over 50 (for the integer and fraction part together), which results in considerable area overhead. Therefore, floating-point variables were finally used for this work since the cost in area was less than using fixed-point variables over 50-bits long. The execution time overhead that is required for floating-point operations is alleviated using lookup tables with precomputed variables for sine and cosine. These lookup tables were defined as global arrays within the HLS code. All global arrays are implemented as ROM memories by default using the HLS tool as they have constant values. Precision and execution time are vital for a design which is intended to be deployed in portable devices for a fast and in-field examination.

A second intervention on the baseline implementation is related to the yellow highlighted code in Fig. 5-10. Where possible, the divisions were replaced with shift operations. Operations such as shift, are considered hardware primitive operations and are a fast and hardware effective operation. It is worth noting, how these two basic interventions made a reduction in both the resource consumption and the execution time which is shown in results Table 6-4. In future, this work could add more optimization techniques provided by HLS tool directives – for example loop unroll techniques for improving speed of operation [61]. In the context of this work, the Vivado HLS directives were not used for optimization.

#### **5.7.1.2 Second design improvement (Interfaces)**

Within the HLS code, the accelerators interfaces are defined using directives, in the form of #pragmas. These directives are defined in the same scope of the code or in a separate file dedicated for directives. The HLS tool allows the developer to simply select what protocol must be used for the control of the exported accelerator and the interface for input and output. There is no need for manually writing and managing the interface signals as is the case of HDL languages. Once the interface is selected then the tool will automatically generate the underlying behavioural code for the function of the signals. For this thesis, two protocols were used, the AXI4-Lite and the AXI4-Stream in two different approaches (Table 5-4). Note that the AXI4-Lite protocol is selected for controlling the HLS IPs connected to the ARM processor

of the ZCU104 subsystem as this is the only AXI4 protocol that can be applied for the control. A HLS IP block without any control protocol has 4 signals as shown in Fig. 5-12 for controlling operations.



Fig. 5-12: HLS IP control signals.

These are the “ap\_start”, “ap\_done”, “ap\_idle” and “ap\_ready” signals. These signals are activated when the accelerator starts, when it is done with a task, when it is idle, and when it is ready for new data. By the time the AXI4-Lite protocol is applied to the accelerator for control, then the interaction with these signals is carried out with API functions and exported with the packaged IP. So, one approach for using the protocols was to use the AXI4-Lite for the control, and the AXI4-Stream protocol for the input and the output transactions. The second approach was to use the AXI4-Lite interface for the control and output operations, while the AXI4-Stream was used for input data operations. In the first approach, the idea was to keep consistent the protocol for the accelerator input and output transactions. Since the output of the accelerators is a single value, the AXI4-Lite was applied to the output (single transaction). As it turns out, the second approach is more efficient in terms of resource utilization [Table 6-7].

Table 5-4: AXI interfaces setting for the accelerators.

<b>Interfaces Combinations</b>	<b>Control</b>	<b>Input</b>	<b>Output</b>
1 <sup>st</sup> approach	AXI4-Lite	AXI4-Stream	AXI4-Stream
2 <sup>nd</sup> approach	AXI4-Lite	AXI4-Stream	AXI4-Lite

### 5.7.2 SDK code

The discriminant function takes form via the driver code developed within the SDK. The discriminant function consists of multiple variables which are based on computations using the features explained by (4-4) ~ (4-7) such as Relative Power. The difference between a variable and a feature is based on using the features in particular ways. For example Relative Power for channel P3 is one variable, while Relative Power for channel P4 is another variable. Also, the Amplitude Asymmetry between channels (F4-P8) for alpha band is one variable and the Amplitude Asymmetry between channels (F4-P8) for beta band is another variable.

**Discriminant function –  
List of qEEG variables**

<p><b>a) Relative Power (only alpha band)</b> channel locs: P3,P4,O1,O2,T8,P7,P8</p>
<p><b>b) Amplitude Asymmetry</b> channel locs for alpha band: (F4-P8),(F8-P8),(F3-O1),(F4-O2),(O1-F7) channel locs for beta band (F4-P8),(F8-P8),(F4-O2)</p>
<p><b>c) Coherence</b> channel locs for theta band: (Fp1-P7) channel locs for beta band: (T7-P7),(C3-P3)</p>
<p><b>d) Phase Difference (only beta band)</b> channel locs: (Fp2-F4),(F3-F4)</p>

Fig. 5-13: List of qEEG variables from Fig. 4-14 grouped by category.

The twenty variables that constitute the discriminant function implemented in this work, are shown in Fig. 5-13. The variables in this figure are the variables from Fig. 4-14 grouped by category.

The MATLAB code in Appendix 5 is the model for this driver code. Inside this script, the length of the input is first defined, and then for each of the variables above (Fig. 5-13), the EEG data samples are read from a repository, and then passed to the corresponding function (Appendices 1-4). The repository that the data is read from is a structure which is automatically created using the EEGLAB tool. This is a dedicated tool for EEG signal processing, and it gives the ability to process several EEG file formats from different vendors. Lastly, the output of each function is held in a variable with the appropriate naming for debugging purposes.

In the SDK driver code, the previously described procedure is somewhat different. A difference is that there are no longer MATLAB functions, but rather hardware accelerators (actual logic circuits) which accept data in a streaming manner, and this is carried out using DMA IPs. So, the driver code feeds the DMAs with data from the main memory and then the DMAs in turn are streaming the data to the accelerator components. The address of the data located in memory is given to the DMAs in the form of pointers, which in the case of this thesis are large arrays with data values. In this case too, the output of the accelerators is held in a variable with the appropriate naming style for debugging purposes.

One more thing that is taking place inside the driver code is the identification and initialization of the hardware components. The hardware specifications that are imported to the SDK tool, describe the hardware design which will be managed by the driver code. The hardware design comprises custom hardware IPs as are the HLS IPs, and ready-made IPs such as the DMAs. Also, the Processing Subsystem (PS), is included in the hardware specifications. A hardware design might contain multiple instances of the same IP, so, there must be a distinction between the several instances of the same IP, and between different IPs. Refer to Fig. 5-15, in this work where five instances of the DMA IP are used. It is necessary then to have an id for every hardware component in order to distinguish between different instances of the same IP, and an address to identify the IP components. An important header file which holds all of the information needed for the management of the IPs (id, base address etc.) is the `<parameters.h>`. This file is contained within the Board Support Package (BSP), which a SDK project incorporates. This file also contains information about the clock frequency and the address for the standard input (stdin) and standard output (stdout) amongst others.

## 5.8 Block design

The block hardware design for this work was built using the GUI canvas of the XILINX's Vivado IPI tool. Using this canvas one can select the components of the FPGA or SoC board that will be included in the design, by simply dragging and dropping block IPs from a list, called the IP Catalog. The block design for this work, comprises of the HLS IP blocks for the features, DMA IPs and the Zynq UltraScale+ PS. The tool automatically adds IPs for the interconnection of the IPs and the system reset. Through the canvas GUI, a user is able to customize the IPs that will be used in the design. For example, in this work's design, the DMAs and the Zynq PS must be customized. By customization, it is implied that this is a pre-synthesis

configuration of the IP blocks. In this work the ZCU PS IP must be customized, to activate the High-Performance Ports (HP) that will interface the DDR controller with the DMAs. Furthermore, the DMAs must be configured for the length of the transaction. Particularly, it must be defined during pre-synthesis, the maximum length of a transaction that the DMA IP can carry out. This is a value that is written to one of the registers of the address space for the DMA, called Buffer Length Register. For this work this value must be at least  $2^{13}$  which is 8192 samples for 1 minute of EEG recordings.

Since there are four different features to calculate, the initial design approach, was to implement four separate accelerators, according to the list of features in Fig. 5-13, and the initial design as shown in Fig. 5-14. However, due to a limitation with the number of High-Performance Ports (HP) supported by the PS system, it was necessary to modify the design architecture approach and to rebuild the discriminant function according to the limitation.

### **5.8.1 Discriminant function design rearrangement**

One problem that was faced as the design progressed was a limitation in the number of High-Performance ports that can be used. The ZCU104 is equipped with 6 Slave High-Performance ports, which means that only 6 DMA instances can be utilized using these HP ports. This work follows the official XILINX tutorial which uses the HP ports to access the DRAM memory [62]. For the initial design approach 7 DMAs are required to support 4 accelerators, but this implementation is short one HP port (Fig. 5-14). This limitation imposes architectural constraints for the discriminant function logic and consequently the accelerators designs. To address this limitation and to reduce the number of DMA instances, a rearrangement in the accelerators design was made, and then merged as shown in Fig. 5-15.

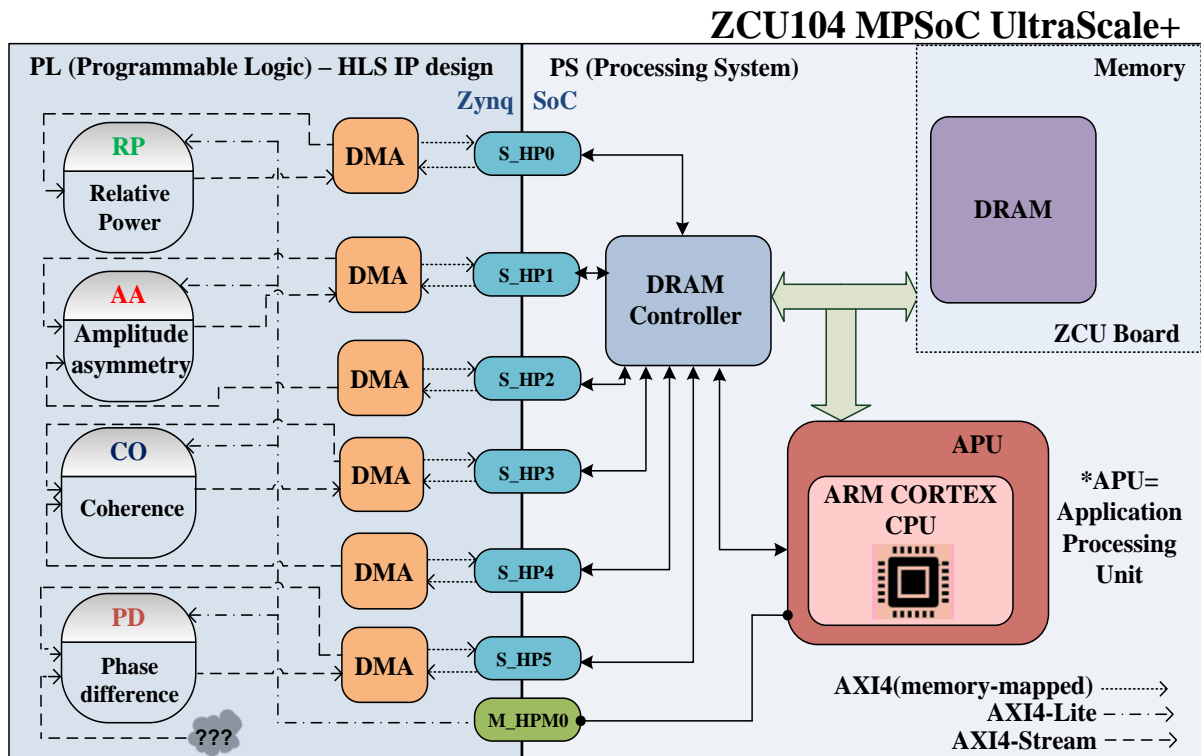


Fig. 5-14: Block design before rearrangement showing insufficient HP port resources.

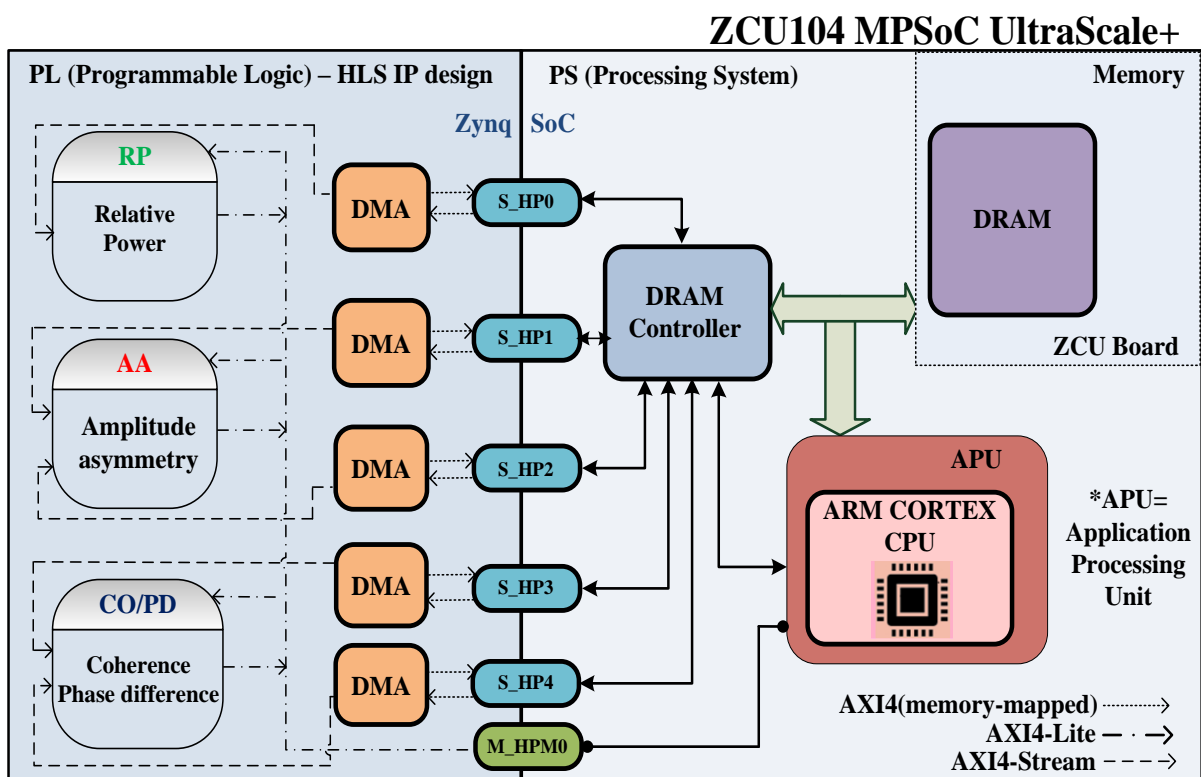


Fig. 5-15: Block design after rearrangement.

Using three accelerators compared to the initial four, reduces the number of DMA instances to five. Based on this, the RP and the AA computations are performed as separate accelerators, while PD and CO computations are merged into a single accelerator design. As part of future investigation, it is possible that the accelerators could be merged even more, for example the AA and the CO\_PD accelerators could be combined, but this has not been investigated fully at this time. To distinguish between the separate tasks of the CO\_PD accelerator, an extra parameter was used as input, which acts as a “mode” selection. This parameter is a single value (an integer in this case), which is set from the PS system via the AXI4-Lite interface.

## 5.9 Conclusion

In this chapter the first steps towards a real-time hardware design of a discriminant function that is able to predict a TBI were explored. Utilizing DMAs for quick data transfers to and from the main memory, independent of the CPU of the system, is the key for streaming applications as is the case for this work. Improvements were made where possible to boost further the performance of this hardware design. In the context of this thesis, some improvements were applied to the hardware design. However, there are other options for optimizations, like using the HLS directives for loop unrolling and pipelining. Finding a good set of optimizations though for a certain HLS hardware design can be a time-consuming task, requiring a lot of simulations and comparisons, and estimates of the trade-offs between the optimizations. Therefore, due to time constraints, emphasis was given to the development of a system realizing the discriminant function and its deployment on a SoC board for actual verification and testing. This is important as it gives confidence for having a system that works in real-life, to which developers can then apply more optimizations for future work. In the next chapter, the results of testing the discriminant function design downloaded onto a ZCU104 board are presented along with the results of the improvements described in this chapter. Finally, although the evaluation of the discriminant function with TBI data is a necessary step to assess its accuracy as a TBI detection tool, this is considered as future work including the important classification step as further described in Chapter 6.





# 6

## Results, conclusions and future work

---

### 6.1 Introduction

In this chapter, the on-board testing results and the improvement outcomes of the hardware system described in the previous chapter are presented. Then, an overall conclusion is discussed leading to a review of ideas for future work. Primary testing for the hardware designs involves the comparison of the hardware design against a software version of the same design, in order to assess the improvement in execution time. Based on this approach, the results that follow include a comparison of a hardware vs. software implementation of the discriminant function. Beginning with the results that came from the design improvements, this chapter demonstrates the results of the discriminant function deployment onto a ZCU104 board and analyses the potential future steps for this research.

### 6.2 Initial design improvement results

The code improvements described in section 5.7.1.1 are about the HLS FFT implementation. The FFT operation is the building block for the accelerators of this work, so the testing and the improvements of the code for the FFT precedes all the other implementations and improvements. Firstly, the resource utilization for the FFT code as it was obtained from [47] is presented. The tables below are a repeat of the information contained in Table 5-2, which is the utilization report of the HLS tool for the ZCU104 PL contained on the FPGA device.

### 6.2.1 Baseline FFT code: no improvements

The HLS tool for a 128-points FFT, reports the following results targeting the ZCU104 PL.

Table 6-1 : Baseline FFT design, HLS utilization report.

FFT Design				
Resources	BRAM_18K	DSP48E	FF	LUT
Available	624	1728	460800	230400
Utilized	16	204	8201	17574
Utilized (%)	2	11	1	7

### 6.2.2 First improvement: using look-up tables instead of the built-in functions for sine and cosine

The HLS tool for a 128-points FFT, reports the following results targeting a ZCU104

Table 6-2: Baseline FFT design after first improvement, HLS utilization report.

FFT Design				
Resources	BRAM_18K	DSP48E	FF	LUT
Available	624	1728	460800	230400
Utilized	2	16	4060	5851
Utilized (%)	~0	~0	~0	2

### 6.2.3 Second improvement: using look-up tables instead of the built-in functions for sine and cosine, and exclude all of the divisions from the code

The HLS tool for a 128-points FFT, reports the following results targeting a ZCU104

Table 6-3: Baseline FFT design after second round of improvements, HLS utilization report.

FFT Design				
Resources	BRAM_18K	DSP48E	FF	LUT
Available	624	1728	460800	230400
Utilized	2	16	1646	4868
Utilized (%)	~0	~0	~0	~0

### 6.2.4 Comparison of the baseline vs the improved design

The table below compares the results of the baseline design vs. the improved design where the values denoted in bold show the considerable performance gains.

Table 6-4: Comparison of the baseline FFT design with the improved version.

FFT Design (no improvements) / <b>FFT Design (with improvements)</b>				
Resources	BRAM_18K	DSP48E	FF	LUT
Available	624	1728	460800	230400
Utilized	16 / <b>2</b>	204 / <b>16</b>	8201 / <b>4060</b>	17574 / <b>5851</b>
Utilized (%)	2 / <b>~0</b>	11 / <b>~0</b>	1 / <b>~0</b>	7 / <b>~0</b>

It is obvious that even with some basic improvements for the FFT code, the design logic area overhead is decreased significantly and it can now be deployed in all of the accelerators. The idea was to use a FFT engine for each one of the accelerators rather than having only one FFT engine shared among the accelerators. So, it was necessary to make these improvements in order to achieve an optimal design with the least possible resource consumption since 3 instances of the FFT engine were used in the final implementation.

### 6.3 Second design improvement results (interfaces, I/F)

The results presented here have to do with the interface of the accelerators. As described in 5.7.1.2, two design approaches were investigated. The tables below are a repeat of the information contained in Table 5-3. Here the design is the discriminant function block design after the rearrangement improvement as shown in Fig. 5-15, which contains the three HLS accelerators blocks within the design architecture.

#### 6.3.1 1<sup>st</sup> approach where the AXI4-Stream I/F is used both at the i/p and o/p of each accelerator block.

**AXI4-Stream → Input, AXI4-Stream → Output**

The Vivado IPI Report Utilization of the implemented design targeting a ZCU104, gives the following results as reported by Table 6-5.

#### 6.3.2 2<sup>nd</sup> approach where the AXI4-Stream I/F is used at the i/p and the AXI4-Lite at the o/p

**AXI4-Stream → Input, AXI4-Lite → Output**

The Vivado IPI Report Utilization of the implemented design targeting a ZCU104, gives the following results as reported by Table 6-6.

Finally, the Table 6-7 gives the comparison of the two approaches and the gain percentage in resource consumption.

Table 6-5 : Discriminant function Report Utilization for first approach of interfaces.

Discriminant Function design									
On-chip resources	CLB LUTs	CLB registers	CARRY 8	F7 MUXES	F8 MUXES	CLB	LUT as Logic	LUT as Memory	Block RAM Tile
Available	230400	460800	28800	115200	57600	28800	230400	101760	312
Utilized	40858	35081	2476	63	3	7623	39717	1141	77.5
Utilized (%)	17.7	7.4	8.5	0.054	0.0052	26.4	17.2	1.1	24.8

Table 6-6 : Discriminant function Report Utilization for second approach of interfaces.

Discriminant Function design									
On-chip resources	CLB LUTs	CLB registers	CARRY 8	F7 MUXES	F8 MUXES	CLB	LUT as Logic	LUT as Memory	Block RAM Tile
Available	230400	460800	28800	115200	57600	28800	230400	101760	312
Utilized	35331	27509	2367	31	3	6236	34534	787	70
Utilized (%)	15.3	5.9	8.2	0.026	0.0052	21.6	14.9	0.7	22.4

Table 6-7 : Comparison of the two interface approaches.

Discriminant Function design 1 <sup>st</sup> approach/ Discriminant Function design 2 <sup>nd</sup> approach									
On-chip resources	CLB LUTs	CLB registers	CARRY 8	F7 MUXES	F8 MUXES	CLB	LUT as Logic	LUT as Memory	Block RAM Tile
Utilized (%)	17.7/ 15.3	7.4/ 5.9	8.5/ 8.2	0.054/ 0.026	0.0052/ 0.0052	26.4/ 21.6	17.2/ 14.9	1.1/ 0.7	24.8/ 22.4
Gain (%)	2.4	1.5	0.3	0.03	0	4.8	2.3	0.4	2.4

The gain percentages in resource consumption might not seem to be significant as absolute numbers. However, the fact that they result from the different accelerator interface approach is an important improvement. It becomes even clearer why the interface has an impact on the resource consumption by referring to Fig. 6-1. The AXI4-Stream interface utilizes the DMA channels to stream the data to the input of the accelerator and to return the data from its output (1<sup>st</sup> approach). While in the second approach, the data from the accelerator output is read via the AXI4-Lite interface and not the DMA streaming channel. That means more FPGA resources are consumed for the 1<sup>st</sup> approach, because the DMA IPs are part of the PL.

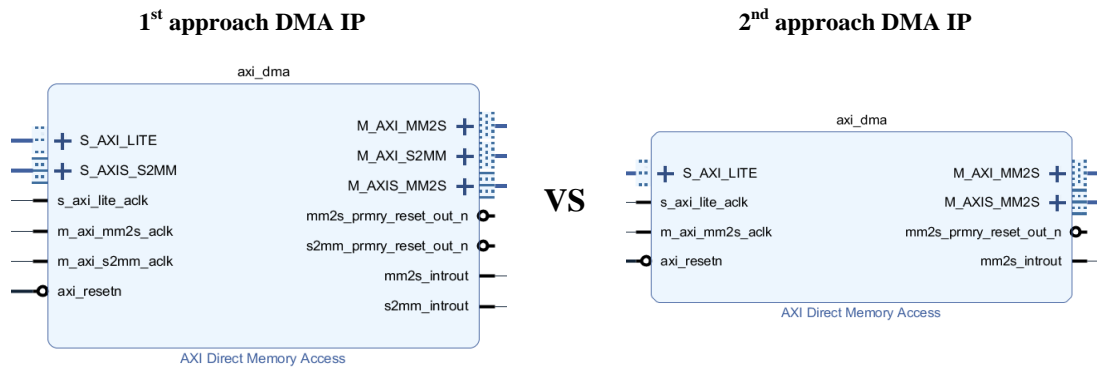


Fig. 6-1: 1<sup>st</sup> approach DMA IP vs 2<sup>nd</sup> approach DMA IP.

## 6.4 Run-time performance results

As part of the design evaluation, a second design for the discriminant function was fully executed from the Cortex ARM CPU in a software-only C-code implementation. The software runs on the PS system on the same board with the accelerators, under the same clock frequency conditions, which is an important factor to consider in order to make a fair comparison. The software contains all the accelerator operations to execute the feature extraction step, these include the FFT computations and power spectral calculations for the qEEG variables that make up the discriminant function. The C-code contained in Appendix 6 is compiled to operate on the Cortex ARM CPU without the use of hardware accelerators. From these setups, timing experiments were conducted to compare hardware vs. software implementations and to evaluate the hardware design. Timing measurements yielded the results as shown in Table 6-8 and Table 6-9.

Table 6-8: Individual accelerators execution time comparison software vs hardware.

Implementation	Timing comparison (millisecond)			
	<i>RP</i>	<i>AA</i>	<i>CO</i>	<i>PD</i>
Software	15.390	30.564	33.036	31.673
Hardware	16.416	17.896	21.672	19.174

Table 6-9: Complete Discriminant Function execution time comparison software vs hardware.

Implementation	Timing comparison (millisecond)	
	<i>Discriminant Function</i>	
Software	514.957	
Hardware	285.706	

The input to each accelerator in the software and hardware tests from Table 6-8 and Table 6-9 consisted of 8192 floating-point values for each EEG channel input. From these tables, a conclusion is that the benefits of a hardware implementation are not obvious when comparing the designs individually but rather when running the system as a whole design the difference in execution time is apparent. In fact, in the individual testing, the software outperforms hardware in the case of RP, but in terms of the complete comparison, the hardware system is almost 2x faster. Therefore, the benefits of a hardware streaming application are obvious particularly in the case of multiple transactions.

## 6.5 Experimental validation of the discriminant function results

For the timing experiments previously discussed, the software C-code runs on the Cortex ARM CPU of the ZCU104 system, whereas the hardware logic design is contained within the PL section of the FPGA logic on the same die. In order to validate the hardware design accelerators that constitute the discriminant function, the hardware design results are compared to the output of the MATLAB model as described in section 5.7 and illustrated in Fig. 5-8. This comparison is a sufficient indication of the correctness of the system. As shown in Fig. 6-2 the output of the hardware design matches the output of the software in MATLAB with high accuracy and tested with the same EEG test data as discussed in the previous paragraph. Red boxes in the left side of Fig. 6-2 show the output of the hardware design using the debugging environment of the Vivado SDK tool, while the red boxes in the right side illustrate the outputs of the MATLAB script in Appendix 5. The small differences in the results are in part due to the fact that MATLAB is using variables of type double for all of its calculations whereas the

IP hardware accelerators blocks are implemented with variables of type float. Note also that MATLAB discriminant function values are shown rounded to 4 decimal places.



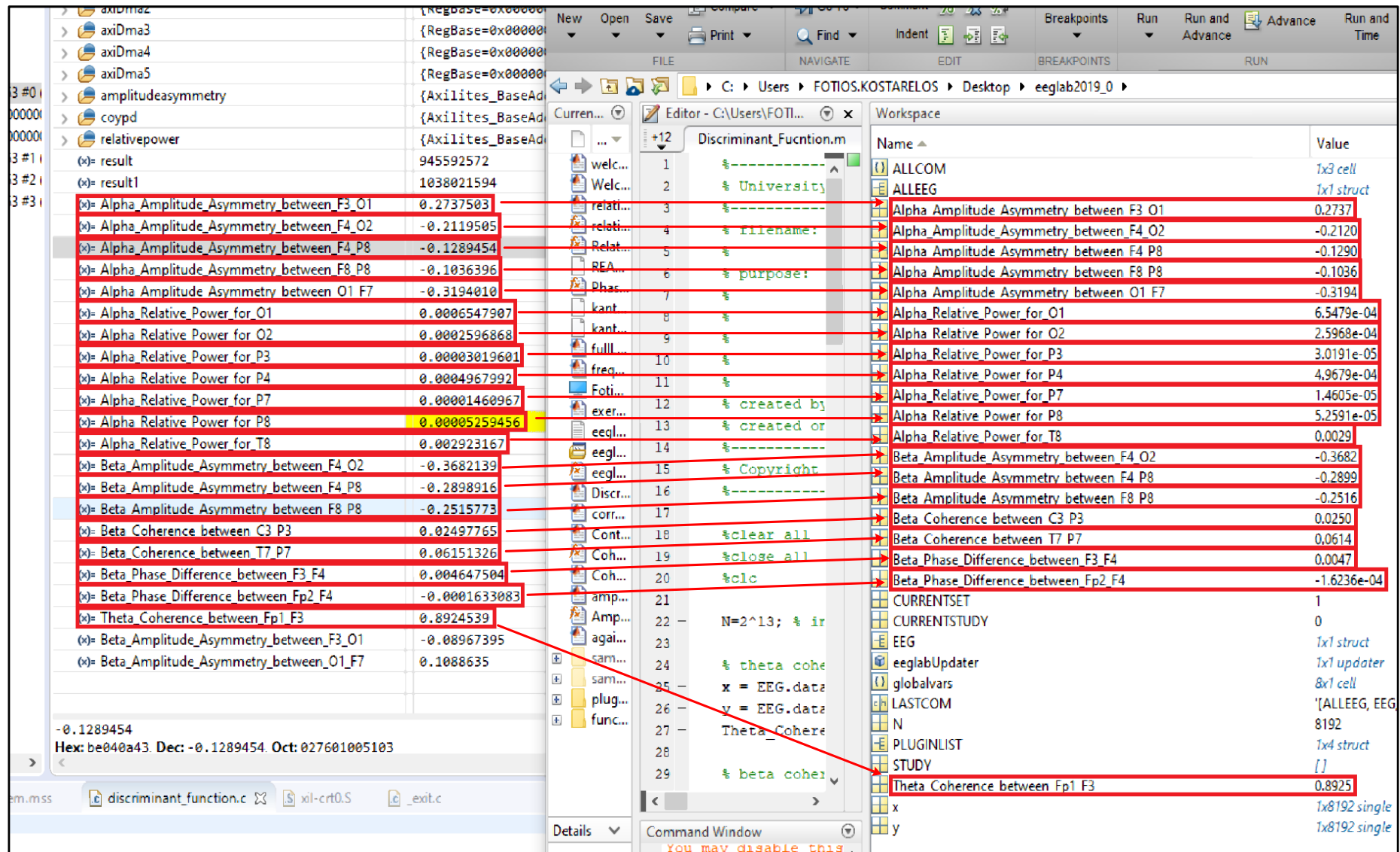


Fig. 6-2: Hardware design results vs software model results.

## 6.6 Conclusions and future work

The goal of this master's thesis was to investigate the use of EEG for TBI detection. A long-term goal of the research which this work is part of, is to have a portable device to be used by first-responders in cases of emergency that will enable a fast and reliable TBI estimation. This is planned to be a multi-modal platform which is based on multiple health factors like, brain EEG signals and Event-Related Potentials (ERPs), along with cranial blood pressure. Exploiting EEG and other health factors too, can result in a more complete system for the detection and classification of a TBI, giving more confidence for the validity of the result of an examination.

As an essential step towards the realisation of this goal, the outcome of this work was a hardware implementation of a discriminant function for TBI detection. Based on power spectrum features like Relative power and Amplitude Asymmetry, and cross-power spectrum features like Coherence and Phase Difference, this discriminant function has been reported to achieve high scores of accuracy, specificity and sensitivity [24]. The authors of this work follow good practice for the development of this discriminant function, i.e. test-retest, independent cross-validations, along with a large sample of healthy patients vs. TBI patients, make this work an excellent example for the capabilities of a discriminant function for TBI detection. To date a discriminant function seems to be the most promising tool for the detection of TBI. Many works have been conducted based on single spectral or cross-power spectral features but as reported by this systematic review [20], this isn't enough for a more comprehensive examination. Although spectral-based techniques like the discriminant function method can give very good results for the detection of TBI, other advanced signal processing techniques like information dynamics and analysis of causal relationship and more, are yet to be investigated. These tools also show promise as feature derivation methods [20].

The hardware implementation prototype was tested on a SoC board, specifically the XILINX Zynq UltraScale+ FPGA SoC ZCU104. This device provides the processing capabilities of an ARM-based CPU system and a FPGA system on the same die. Utilizing hardware accelerators for carrying out compute intensive tasks (like the FFT), is the key idea for this work. Also, using a device like this, allows the real-time implementation of a system with the use of DMAs that enable fast data transfers to and from the main memory. A SoC system or an ASIC would constitute a very good solution for the complete system for future work. XILINX provides other series of UltraScale and UltraScale+ FPGA devices such as the Kintex and Virtex series which have larger design capabilities.

Some thoughts and recommendations as future steps are the following that might help the course of this research.

### **6.6.1 Evaluation**

A significant challenge when EEG is used in clinical studies is the lack of sufficient data, which need to be experimental as well. The absence of publicly accessible repositories with data from studies already conducted is counterproductive to research. This can harm research as ideas cannot be explored by researchers and consequently, they cannot come with new ones. In the context of this TBI study only a single database was found [63] and gaining access to that database requires ethical approval, which is a process that can take a considerable amount of time to get. In addition, the unexpected pandemic events of 2020 made the quest of data access impossible. So, the very first step as future work is to obtain data in order to evaluate the potential of this discriminant function in practice.

### **6.6.2 Performance**

The performance of the hardware design is split between time execution and power consumption metrics.

- a) For time execution, the HLS directives for loop unrolling and pipelining are the most suitable as a first attempt to boost the performance in terms of time. The HLS code for the discriminant function accelerators has some parts such as large ‘for loops’ that could apply these two directives. However, it is wise first to have a complete system as described in 6.6.4 below and then apply directives since some directives can improve the execution time but with a significant impact on the area overhead, for example the loop unroll directive.
- b) For the power consumption, reducing the clock frequency from the default (100 MHz) which is the main clock that operates the discriminant function logic is the first step. This procedure can reveal the golden ratio between the execution time and power consumption. Also, another way to reduce power consumption is to use the Low-Performance ports of the ZCU104 for accessing the DRAM memory with the DMAs.

### **6.6.3 Classification**

As literature suggests from this research, EEG signals cannot stand alone as detection tools [20]. One reason is the high risk of misinterpretation of a brain injury with other diseases like autism or depressive disorder. This is a specificity problem (see Chapter 3 Section 3.4). From this work, another question that remains to be answered is, to what extent Machine Learning can facilitate the TBI detection process. Since, the problem of misinterpretation is a specificity problem, then one promising future work is to apply this Machine Learning technique [33] which renders the specificity and sensitivity separately. This technique probably can make a difference and help to overcome specificity problems which are an obstacle to the use of EEG signals for TBI detection since misinterpretation is often a problem.

### **6.6.4 All in one system**

Finally, the hardware implementation of the Automatic Wavelet Independent Component Analysis (AWICA) technique [44], is an interesting topic. The ICA algorithm for artefact rejection is an extremely compute-intensive task requiring successive matrix multiplications, which sometimes must be run several times in order to achieve a good signal-noise-to-ratio outcome. By combining the three steps of Fig. 6-3 into a hardware system, which performs artefact rejection, feature extraction and classification, then it is plausible for this system to be completely implemented in hardware and to operate software-independently. It is a big challenge to see if this system can operate within a time window of 1 minute to process multiple EEG records.

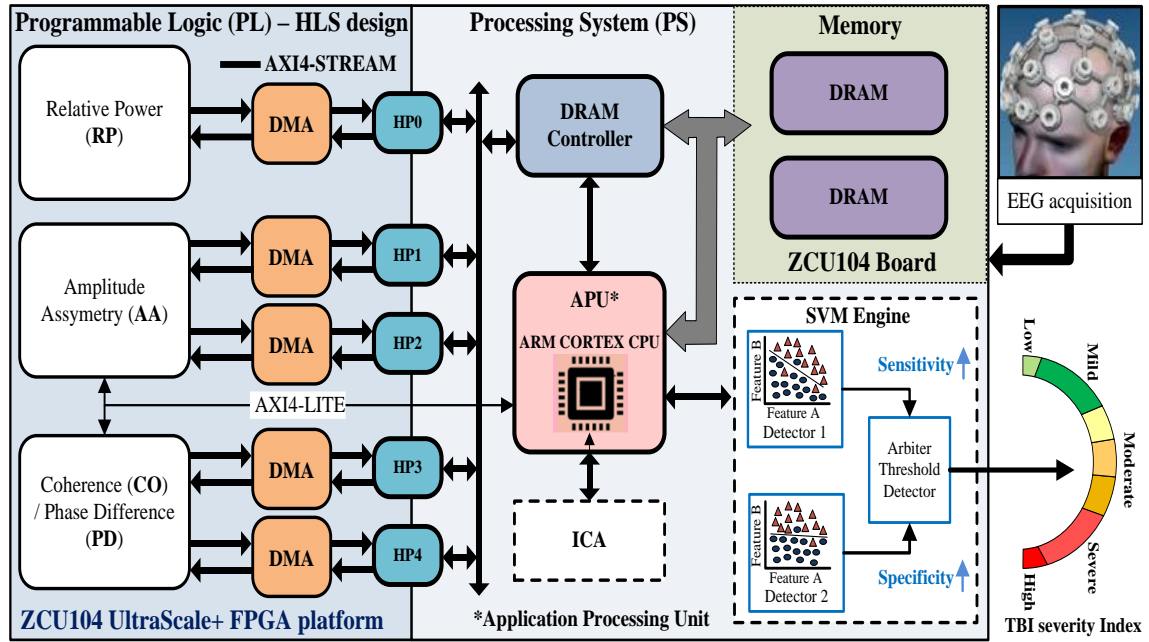


Fig. 6-3: A complete EEG processing pipeline for hardware implementation.

As it turns out there are a few things to work out in this research for the future. Due to the limitations described above in 6.6.1, this work focused on the hardware implementation of the accelerators and the real-time needs of the discriminant function. The work described in this thesis has established that it is possible to compute a discriminant function using computationally efficient FPGA technology and proven MATLAB signal processing techniques. The work was successfully carried out during the Covid-19 pandemic, which placed some restrictions on the data that could be used, but the system developed during the project can be further tested on real TBI datasets when these become available.



## References

- [1] CENTER-TBI, "Collaborative European NeuroTrauma Effectiveness Research in TBI," 16 June 2020.
- [2] F. Kostarelos, C. MacNamee, and B. Mullane, "A Hardware Implementation of a qEEG-Based Discriminant Function for Brain Injury Detection," presented at the IEEE Biomedical Circuits and Systems Conference (BIOCAS), Berlin, Germany., October 2021.
- [3] J. S. Kumar and P. Bhuvaneswari, "Analysis of Electroencephalography (EEG) Signals and Its Categorization—A Study," *Procedia Engineering*, vol. 38, pp. 2525-2536, 2012/01/01/ 2012.
- [4] C. Henley, "Foundations of Neuroscience," Michigan State University, September 2020.
- [5] SliderBase. *Neurons, Synapses, and Signaling*. Available: <http://www.sliderbase.com/spitem-1448-2.html>
- [6] D. Millet, "Hans Berger: From Psychic Energy to the EEG," *Perspectives in Biology and Medicine*, vol. 44, no. 4, pp. 522-542, Autumn 2001.
- [7] P. D. Bryn Farnsworth, "What is EEG (Electroencephalography) and How Does it Work?," *iMotions Blog*,
- [8] A. Eklund, T. E. Nichols, and H. Knutsson, "Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates," *Proceedings of the National Academy of Sciences*, vol. 113, no. 28, p. 7900, 2016.
- [9] P. D. Bryn Farnsworth, "Making Waves – The Past, Present, and Future of EEG Research," *iMotions Blog*, Accessed on: October 18th
- [10] B. Burle, L. Spieser, C. Roger, L. Casini, T. Hasbroucq, and F. Vidal, "Spatial and temporal resolution of EEG: Is it really black and white? A Scalp Current Density view," *International Journal of Psychophysiology*, vol. 8, 05/12 2015.
- [11] T. Hines. (2018). *Anatomy of the Brain*. Available: <https://mayfieldclinic.com/pe-anatbrain.htm>
- [12] The University Of Queensland, Australia -Brain Institute. (2018). *Lobes of the brain*. Available: <https://qbi.uq.edu.au/brain/brain-anatomy/lobes-brain>
- [13] G. H. Klem, H. Lüders, H. H. Jasper, and C. E. Elger, "The ten-twenty electrode system of the International Federation. The International Federation of Clinical Neurophysiology," *Electroencephalography and clinical neurophysiology. Supplement*, vol. 52, pp. 3-6, 1999.
- [14] IMOTIONS, "EEG POCKET GUIDE," Available: <https://imotions.com/guides/electroencephalography-eeeg/>
- [15] P. S. Guide. (2013). *The International 10/20 System of Electrode Placement*. Available: <https://sleeptechstudy.wordpress.com/category/1020-system/>
- [16] J. Moeller. (2014, July 13). *Introduction to EEG*. Available: <https://www.youtube.com/watch?v=XMizSSOejg0>

- [17] *CHILDHOOD EPILEPSY WITH CENTROTEMPORAL SPIKES*. Available: <https://www.epilepsydiagnosis.org/syndrome/ects-eeg.html>
- [18] J. Malmivuo and R. Plonsey, "Bioelectromagnetism. 13. Electroencephalography," 1995, pp. 247-264.
- [19] The US Dept. of Veteran Affairs, "Traumatic brain injury: A Guide For Patients ", ed.
- [20] P. E. Rapp *et al.*, "Traumatic Brain Injury Detection Using Electrophysiological Methods," (in English), *Frontiers in Human Neuroscience*, Review vol. 9, no. 11, 2015-February-04 2015.
- [21] M. Nuwer, "Assessment of digital EEG, quantitative EEG, and EEG brain mapping: Report of the American Academy of Neurology and the American Clinical Neurophysiology Society\* [RETIRED]," *Neurology*, vol. 49, no. 1, p. 277, 1997.
- [22] T. T. K. Munia, A. Haider, and R. Fazel-Rezai, "Evidence of brain functional deficits following sport-related mild traumatic brain injury," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, pp. 3212-3215.
- [23] C. Cao and S. Slobounov, "Alteration of Cortical Functional Connectivity as a Result of Traumatic Brain Injury Revealed by Graph Theory, ICA, and sLORETA Analyses of EEG Signals," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 1, pp. 11-19, 2010.
- [24] R. W. Thatcher, R. A. Walker, I. Gerson, and F. H. Geisler, "EEG discriminant analyses of mild head trauma," *Electroencephalography and Clinical Neurophysiology*, vol. 73, pp. 94-106, 1989.
- [25] R. Thatcher *et al.*, "An EEG Severity Index of Traumatic Brain Injury," *The Journal of neuropsychiatry and clinical neurosciences*, vol. 13, pp. 77-87, 02/01 2001.
- [26] M. Sherer, M. Struchen, S. Yablon, Y. Wang, and T. Nick, "Comparison of indices of traumatic brain injury severity: Glasgow Coma Scale, length of coma and post-traumatic amnesia," *Journal of neurology, neurosurgery, and psychiatry*, vol. 79, pp. 678-85, 07/01 2008.
- [27] G. Institute of Neurological Sciences. (2019). *What is the Glasgow Coma Scale?* Available: <https://www.glasgowcomascale.org/what-is-gcs/>
- [28] U. S. N. C. f. B. Information and S. database. (2013, January). *Complications of Mild Traumatic Brain Injury in Veterans and Military Personnel: A Systematic Review [Internet]*. Available: <https://www.ncbi.nlm.nih.gov/books/NBK189784/table/appc.t1/>
- [29] t. d. science. (2018, March 21). *Why Deep Learning over Traditional Machine Learning?* Available: <https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063>
- [30] W. Fang, K. Wang, N. Fahier, Y. Ho, and Y. Huang, "Development and Validation of an EEG-Based Real-Time Emotion Recognition System Using Edge AI Computing Platform With Convolutional Neural Network System-on-Chip Design," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 645-657, 2019.
- [31] F. Shaheen, B. Verma, and M. Asafuddoula, "Impact of Automatic Feature Extraction in Deep Learning Architecture," in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2016, pp. 1-8.
- [32] A. Baratloo, M. Hosseini, A. Negida, and G. El Ashal, "Part 1: Simple Definition and Calculation of Accuracy, Sensitivity and Specificity," (in eng), *Emergency (Tehran, Iran)*, vol. 3, no. 2, pp. 48-49, Spring 2015.
- [33] M. A. B. Altaf, C. Zhang, and J. Yoo, "A 16-Channel Patient-Specific Seizure Onset and Termination Detection SoC With Impedance-Adaptive Transcranial Electrical Stimulator," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 11, pp. 2728-2740, 2015.



- [34] A. Ng. (1 Jan 2017). *Lecture 12.1 — Support Vector Machines / Optimization Objective — [ Machine Learning / Andrew Ng]*. Available: <https://www.youtube.com/watch?v=hCOIMkcsmg>
- [35] M. Heyden, "Classification of EEG data using machine learning techniques," MSc, LUND University, 2016.
- [36] H. Nyquist, "Certain Topics in Telegraph Transmission Theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617-644, 1928.
- [37] C. E. Shannon, "Communication In The Presence Of Noise," *Proceedings of the IEEE*, vol. 86, no. 2, pp. 447-457, 1998.
- [38] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," *Journal of Neuroscience Methods*, vol. 134, no. 1, pp. 9-21, 2004/03/15/ 2004.
- [39] (6/20/2020). *The OpenBCI GUI*. Available: <https://docs.openbci.com/docs/06Software/01-OpenBCISoftware/GUIDocs>
- [40] N. Key. (Aug 29, 2019). *Electroencephalography: General Principles and Clinical Applications*. Available: <https://neupsykey.com/electroencephalography-general-principles-and-clinical-applications/>
- [41] A. J. H. Jayant N. Acharya, † Partha D. Thirumala,‡ and Tammy N. Tsuchida§, "A Proposal for Standard Montages to Be Used in Clinical EEG," *Journal of Clinical Neurophysiology* vol. Volume 33, Number 4, August 2016.
- [42] S. Baillet, J. Mosher, R. M. Leahy, and D. W. Shattuck. (1999). *Brainstorm: A matlab toolbox for the processing of MEG and EEG signals*. Available: <https://neuroimage.usc.edu/brainstorm/>
- [43] N. Kutz. (31 December 2018). *Independent Component Analysis 1*. Available: <https://www.youtube.com/watch?v=e4SN4TWlgY>
- [44] B. Albert *et al.*, "Automatic EEG processing for the early diagnosis of Traumatic Brain Injury," in *2016 World Automation Congress (WAC)*, 2016, pp. 1-6.
- [45] X. Jiang, G.-B. Bian, and Z. Tian, "Removal of Artifacts from EEG Signals: A Review," (in eng), *Sensors (Basel, Switzerland)*, vol. 19, no. 5, p. 987, 2019.
- [46] J. W. C. a. J. W. Tukey, " "An algorithm for the machine calculation of complex Fourier series", " 1965.
- [47] R. Kastner, J. Matai, and S. Neuendorffer, *Parallel Programming for FPGAs* (arXiv:1805.03648 [cs]). 2018.
- [48] P. Welch, "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms," *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 70-73, 1967.
- [49] Z. Bian, Q. Li, L. Wang, C. Lu, S. Yin, and X. Li, "Relative power and coherence of EEG series are related to amnesic mild cognitive impairment in diabetes," (in English), *Frontiers in Aging Neuroscience*, Original Research vol. 6, no. 11, 2014-February-04 2014.
- [50] J. N. Ianof and R. Anghinah, "Traumatic brain injury: An EEG point of view," (in eng), *Dementia & neuropsychologia*, vol. 11, no. 1, pp. 3-5, Jan-Mar 2017.
- [51] Z. Haneef, H. S. Levin, J. D. Frost, Jr., and E. M. Mizrahi, "Electroencephalography and quantitative electroencephalography in mild traumatic brain injury," (in eng), *Journal of neurotrauma*, vol. 30, no. 8, pp. 653-656, 2013.
- [52] J. B. Earle, "Task Difficulty and EEG Alpha Asymmetry: An Amplitude and Frequency Analysis," *Neuropsychobiology*, vol. 20, no. 2, pp. 96-112, 1988.
- [53] S. Butler, "Alpha Asymmetry, Hemispheric Specialization and the Problem of Cognitive Dynamics," 1988.

- 
- [54] *Fourier analysis of neuronal oscillations and synchronization*. Available: <https://www.fieldtriptoolbox.org/tutorial/fourier/>
- [55] J. E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330-334, 1959.
- [56] G. team. (2019-11-28). 7 *Pragmas*. Available: <https://gcc.gnu.org/onlinedocs/cpp/Pragmas.html>
- [57] Xilinx, Vivado Design Suite User Guide, High-Level Synthesis, UG902, May 30, 2014. [Online]. Available: [www.xilinx.com](http://www.xilinx.com).
- [58] XILINX. (2021). *Zynq UltraScale+ MPSoC*. Available: <https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html>
- [59] Xilinx, ZCU104 Evaluation Board, User Guide, UG1267 (v1.1) October 9, 2018. [Online]. Available: [https://www.xilinx.com/support/documentation/boards\\_and\\_kits/zcu104/ug1267-zcu104-eval-bd.pdf](https://www.xilinx.com/support/documentation/boards_and_kits/zcu104/ug1267-zcu104-eval-bd.pdf).
- [60] Xilinx. (2021). *Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit*. Available: <https://www.xilinx.com/products/boards-and-kits/zcu104.html>
- [61] XILINX. (20 August 2018). *HLS Pragmas*. Available: [https://www.xilinx.com/html\\_docs/xilinx2017\\_4/sdaccel\\_doc/okr1504034364623.html](https://www.xilinx.com/html_docs/xilinx2017_4/sdaccel_doc/okr1504034364623.html)
- [62] Xilinx, Vivado Design Suite Tutorial, High-Level Synthesis, May 22, 2019. [Online]. Available: [https://www.xilinx.com/support/documentation/sw\\_manuals/xilinx2019\\_1/ug871-vivado-high-level-synthesis-tutorial.pdf](https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug871-vivado-high-level-synthesis-tutorial.pdf).
- [63] Federal Interagency Traumatic Brain Injury Research Informatic System. Available: <https://fitbir.nih.gov/>

## Appendix 1: MATLAB code for Relative Power

```
%-----
% University of Limerick - Dept. of Electronic and Computer Engineering
%-----
% filename:      Relative_Power.m
%
% purpose:       This function calculates the frequency Relative Power
%               for a signal by applying the equation (3-4)
%               using the Welch's method as described in section 3.3.2.
%
% created by:    Fotios Kostarelos
% created on:    9th March 2021
%-----
---
% Copyright 2021 University of Limerick
%-----
--

function Relative_Power = Relative_Power(x)

% initializations
Fs=125; %% sampling rate

Navg = 127; %% Welch's method iterations for averaging
nfft = 128; %% FFT points
win=hamming(nfft); %% window function
window=win.*sqrt(nfft/sum(win.^2));
samplesread = nfft-1;

magsqmpax= zeros(nfft/2,1);

alphabandStart = 7;
alphabandEnd = 13;

TotalbandStart = 1;
TotalbandEnd = 22;

x = x*10^-2;
startingpointx=1;
endpointx=0;

% spectral calculations
for i=1:Navg
    endpointx=startingpointx+samplesread;
    x_column1=x(startingpointx:endpointx)'; %% Reading data from input
    equal in size with FFT points
    startingpointx=startingpointx+(samplesread+1)/2;
    xw = x_column1.*window;                % apply window of length nfft
    X= fft(xw,nfft);                        % DFT
    X= X(1:nfft/2);                         % retain samples from 0 to fs/2
    magsqx= real(X).^2 + imag(X).^2;         % DFT magnitude squared
    magsqmpax(:,1) = magsqmpax(:,1) + magsqx;% sum of DFT mag squared
end
```

```
magsqmpax = magsqmpax/Navg; % average of DFT mag squared
P_bin1x= 2/nfft.^2 *magsqmpax; % W/bin power spectrum
P_Hz1x= P_bin1x*nfft/Fs; % W/Hz power spectrum
P_Hz1x(1)=P_Hz1x(1)/2;

% feature calculations
overallPower = sum(P_Hz1x(TotalbandStart:TotalbandEnd));
alphaPower = sum(P_Hz1x(alphabandStart:alphabandEnd));

Relative_Power = (alphaPower/overallPower) * 100;

end
```

## Appendix 2: MATLAB code for Amplitude Asymmetry

```
%-----
% University of Limerick - Dept. of Electronic and Computer Engineering
%-----
% filename:      Amplitude_Asymmetry.m
%
% purpose:       This function calculates the frequency Amplitude
%               Asymmetry between two signals by applying the equation
%               (3-5) using the Welch's method as described in section
%               3.3.2.
% created by:    Fotios Kostarelos
% created on:    9th March 2021
%-----
% Copyright 2021 University of Limerick
%-----

function Amplitude_Asymmetry = Amplitude_Asymmetry(x,y,z)

% variables initializations
Fs = 125;
Navg = 127;
nfft = 128;
samplesread = nfft-1;
alphabandStart = 7;
alphabandEnd = 13;
betabandStart = 12;
betabandEnd = 22;
startingpointx=1;
endpointx=0;

% vectors initializations
win=hamming(nfft);
window=win.*sqrt(nfft/sum(win.^2));

magsqmpax= zeros(nfft/2,1);
magsqmpay= zeros(nfft/2,1);

betaAmplitudeAsymmetry=zeros(1,betabandEnd-betabandStart);
alphaAmplitudeAsymmetry=zeros(1,alphabandEnd-alphabandStart);

% scaling(not part of the HLS code)
x = double(x);
y = double(y);
x = x*10^-2;
y = y*10^-2;

% spectral calculations
for k=1:Navg
    endpointx=startingpointx+samplesread;
    x_column1=x(startingpointx:endpointx)';
    startingpointx=startingpointx+(samplesread+1)/2;
    xw = x_column1.*window;                % apply window of length nfft
    X= fft(xw);                            % DFT
    X= X(1:nfft/2);                        % retain samples from 0 to fs/2
    magsqx= real(X).^2 + imag(X).^2;        % DFT magnitude squared
    magsqmpax(:,1) = magsqmpax(:,1) + magsqx;% sum of DFT mag squared
end
```

```

magsqmpax = magsqmpax/Navg; % average of DFT mag squared
P_bin1x= 2/nfft.^2 *magsqmpax; % W/bin power spectrum
P_Hz1x= P_bin1x*nfft/Fs; % W/Hz power spectrum
P_Hz1x(1)=P_Hz1x(1)/2;

startingpointy=1;
endpointy=0;

for l= 1:Navg
    endpointy=startingpointy+samplesread;
    y_column1=y(startingpointy:endpointy)';
    startingpointy=startingpointy+(samplesread+1)/2;
    yw = y_column1.*window; % apply window of length nfft
    Y= fft(yw); % DFT
    Y= Y(1:nfft/2); % retain samples from 0 to fs/2
    magsqy= real(Y).^2 + imag(Y).^2; % DFT magnitude squared
    magsqmpay(:,1) = magsqmpay(:,1) + magsqy;% sum of DFT mag squared
end

magsqmpay = magsqmpay/Navg; % average of DFT mag squared
P_bin1y= 2/nfft.^2 *magsqmpay; % W/bin power spectrum
P_Hz1y= P_bin1y*nfft/Fs; % W/Hz power spectrum
P_Hz1y(1)=P_Hz1y(1)/2;

% feature calculations
n=1;
for m=betabandStart:betabandEnd
    n = n + 1;
    betaAmplitudeAsymmetry(n)=(sqrt(P_Hz1x(m))-
sqrt(P_Hz1y(m)))/(sqrt(P_Hz1x(m))+sqrt(P_Hz1y(m)));
end
sumbetaAA = sum(betaAmplitudeAsymmetry);
betaAAWelch = sumbetaAA/(betabandEnd-betabandStart+1);

n = 1;
for m=alphabandStart:alphabandEnd
    n = n + 1;
    alphaAmplitudeAsymmetry(n)=(sqrt(P_Hz1x(m))-
sqrt(P_Hz1y(m)))/(sqrt(P_Hz1x(m))+sqrt(P_Hz1y(m)));
end
sumalphaAA = sum(alphaAmplitudeAsymmetry);
alphaAAWelch = sumalphaAA/(alphabandEnd-alphabandStart+1);

% output
if z==1
    Amplitude_Asymmetry = alphaAAWelch;
else
    Amplitude_Asymmetry = betaAAWelch;
end
end

```

### Appendix 3: MATLAB code for Coherence

```

%-----
% University of Limerick - Dept. of Electronic and Computer Engineering
%-----
% filename:      Coherence_.m
%
% purpose:       This function calculates the frequency Coherence
%                between two signals by applying the equation (3-6)
%                using the Welch's method as described in section 3.3.2.
%                **It gives the same result to the MATLAB call
%                mscohere().**
% created by:    Fotios Kostarelos
% created on:     9th March 2021
%-----
% Copyright 2021 University of Limerick
%-----
% initializations
Fs = 125;
x = x*10^-2;
y = y*10^-2;
x = double(x);
y = double(y);

thetabandStart = 4;
thetabandEnd = 6;

betabandStart = 12;
betabandEnd = 22;

mybetaCo=zeros(1,betabandEnd-betabandStart);
betaCo=zeros(1,betabandEnd-betabandStart);

mythetaCo=zeros(1,thetabandEnd-thetabandStart);
thetaCo=zeros(1,thetabandEnd-thetabandStart);

Navg = 127;
nfft = 128;
win=hamming(nfft);
window=win.*sqrt(nfft/sum(win.^2));

samplesread = nfft-1;
startingpoint=1;
endpoint=0;
xmulconjyfft = zeros(nfft/2+1,Navg);

magsqmpax= zeros(nfft/2,1);
magsqmpay= zeros(nfft/2,1);

```

```

% spectral calculations
N = length(x);
for i=1:Navg
    endpoint=startingpoint+samplesread;
    x1=x(startingpoint:endpoint)'.*window;
    y1=y(startingpoint:endpoint)'.*window;
    xdft1 = fft(x1,nfft); %<outputs two-sided complex amplitude spectra
    ydft1 = fft(y1,nfft);
    startingpoint=startingpoint+(samplesread+1)/2;
    Sxyla=ydft1.*conj(xdft1);
    Sxylc=Sxyla(1:nfft/2+1);
    Sxylc(2:end-1)=2*Sxylc(2:end-1);
    Sxylc = Sxylc';
    xmulconjyfft(:,i)=Sxylc;
end

avgxyfft = mean(xmulconjyfft,2);
avgxyfft= avgxyfft/(nfft*Fs);

startingpointx=1;
endpointx=0;

for i=1:Navg
    endpointx=startingpointx+samplesread;
    x_column1=x(startingpointx:endpointx)';
    startingpointx=startingpointx+(samplesread+1)/2;
    xw = x_column1.*window; % apply window of length nfft
    X= fft(xw); % DFT
    X= X(1:nfft/2); % retain samples from 0 to fs/2
    magsqx= real(X).^2 + imag(X).^2; % DFT magnitude squared
    magsqmpax(:,1) = magsqmpax(:,1) + magsqx;% sum of DFT mag squared
end

magsqmpax = magsqmpax/Navg; % average of DFT mag squared
P_bin1x= 2/nfft.^2 *magsqmpax; % W/bin power spectrum
P_Hz1x= P_bin1x*nfft/Fs; % W/Hz power spectrum
P_Hz1x(1)=P_Hz1x(1)/2;

startingpointy=1;
endpointy=0;

for i= 1:Navg
    endpointy=startingpointy+samplesread;
    y_column1=y(startingpointy:endpointy)';
    startingpointy=startingpointy+(samplesread+1)/2;
    yw = y_column1.*window; % apply window of length nfft
    Y= fft(yw); % DFT
    Y= Y(1:nfft/2); % retain samples from 0 to fs/2
    magsqy= real(Y).^2 + imag(Y).^2; % DFT magnitude squared
    magsqmpay(:,1) = magsqmpay(:,1) + magsqy;% sum of DFT mag squared
end

magsqmpay = magsqmpay/Navg; % average of DFT mag squared
P_bin1y= 2/nfft.^2 *magsqmpay; % W/bin power spectrum
P_Hz1y= P_bin1y*nfft/Fs; % W/Hz power spectrum
P_Hz1y(1)=P_Hz1y(1)/2;

```



```

% feature calculations
n = 1;
for m=betabandStart:betabandEnd
    n = n + 1;

    mybetaCo(n) = ((real(avgxyfft(m))*real(avgxyfft(m)))+(imag(avgxyfft(m))*i
mag(avgxyfft(m))))/(P_Hz1x(m)*P_Hz1y(m));
end
summybetaCo = sum(mybetaCo);
mybetaCoh = summybetaCo/(betabandEnd-betabandStart+1);
n = 1;
for m=thetabandStart:thetabandEnd
    n = n + 1;

    mythetaCo(n) = ((real(avgxyfft(m))*real(avgxyfft(m)))+(imag(avgxyfft(m))*i
mag(avgxyfft(m))))/(P_Hz1x(m)*P_Hz1y(m));
end
summythetaCo = sum(mythetaCo);
mythetaCoh = summythetaCo/(thetabandEnd-thetabandStart+1);

if z==1
    Coherence_ = mybetaCoh;
else
    Coherence_ = mythetaCoh;
end

end

```



## Appendix 4: MATLAB code for Phase difference

```

%-----
% University of Limerick - Dept. of Electronic and Computer Engineering
%-----
% filename:      Phase_Difference.m
%
% purpose:       This function calculates the frequency Phase difference
%                between two signals by applying the equation (3-7)
%                using the Welch's method as described in section 3.3.2.
%
% created by:    Fotios Kostarelos
% created on:    9th March 2021
%-----
% Copyright 2021 University of Limerick
%-----
function Phase_Difference = Phase_Difference(x,y)

% initializations
Fs = 125;

x = x*10^-2;
y = y*10^-2;
N = length(x);
Navg = 127;
nfft = 128;
win=hamming(nfft);
window=win.*sqrt(nfft/sum(win.^2));

betabandStart = 12;
betabandEnd = 22;

betaPD1=zeros(1,betabandEnd-betabandStart);

samplesread = nfft-1;
startingpoint=1;
endpoint=0;
xmulconjyfft = zeros(nfft/2+1,Navg);

% spectral calculations
for i=1:Navg
    endpoint=startingpoint+samplesread;
    x1=x(startingpoint:endpoint)'.*window;
    y1=y(startingpoint:endpoint)'.*window;
    xdft1 = fft(x1,nfft); %<outputs two-sided complex amplitude spectra
    ydft1 = fft(y1,nfft);
    startingpoint=startingpoint+(samplesread+1)/2;
    Sxyla=ydft1.*conj(xdft1);
    Sxylb=Sxyla./N^2;
    Sxylc=Sxylb(1:nfft/2+1);
    Sxylc(2:end-1)=2*Sxylc(2:end-1);
    Sxylc = Sxylc';
    xmulconjyfft(:,i)=Sxylc;
end

avgxyfft = mean(xmulconjyfft,2);
avgxyfft= avgxyfft*((length(x).^2)/(nfft*Fs));

```

```
% feature calculations
n=1;
for m=betabandStart:betabandEnd
    n = n + 1;
    betaPD1(n)=atan(imag(avgxyfft(m))/real(avgxyfft(m)))/19; %% Reference
    thatcher's paper & theory
end
sumbetaPD1 = sum(betaPD1);

Phase_Difference = sumbetaPD1/(betabandEnd-betabandStart+1);

end
```

## Appendix 5: MATLAB code for Discriminant function

```
%-----
% University of Limerick - Dept. of Electronic and Computer Engineering
%-----
% filename:      Discriminant_Function.m
%
% purpose:       Discriminant Function realisation as described in thesis.
%               Uses the MATLAB functions which are the features for
%               Relative Power, Amplitude_Asymmetry, Coherence_ and
%               Phase_Difference, to calculate the 20 variables from
%               Fig 4-13.
%
% created by:    Fotios Kostarelos
% created on:    9th March 2021
%-----
% Copyright 2021 University of Limerick
%-----
clear all
close all
clc

N=2^13; % input length

% theta coherence channels Fp1 F3
x = EEG.data(2,N+1:2*N);
y = EEG.data(2,1:N);
Coherence_between_Fp1_F3 = Coherence_(x,y,2);

% beta coherence channels T7 P7
x = EEG.data(3,N+1:2*N);
y = EEG.data(1,1:N);
Coherence_between_T7_P7 = Coherence_(x,y,1);

% beta coherence channels C3 P3
x = EEG.data(4,N+1:2*N);
y = EEG.data(4,1:N);
Coherence_between_C3_P3 = Coherence_(x,y,1);

% beta phase difference channels Fp2 F4
x = EEG.data(5,N+1:2*N);
y = EEG.data(6,N+1:2*N);
Phase_Difference_between_Fp2_F4 = Phase_Difference(x,y);

% beta phase difference channels F3 F4
x = EEG.data(2,1:N);
y = EEG.data(6,N+1:2*N);
Phase_Difference_between_F3_F4 = Phase_Difference(x,y);

% alpha,beta amplitude asymmetry channels F4 P8
x = EEG.data(6,N+1:2*N);
y = EEG.data(3,1:N);
alpha_Amplitude_Asymmetry_between_F4_P8 = Amplitude_Asymmetry(x,y,1);
beta_Amplitude_Asymmetry_between_F4_P8 = Amplitude_Asymmetry(x,y,2);
```

```

% alpha,beta amplitude asymmetry channels F8 P8
x = EEG.data(8,N+1:2*N);
y = EEG.data(3,1:N);
alpha_Amplitude_Asymmetry_between_F8_P8 = Amplitude_Asymmetry(x,y,1);
beta_Amplitude_Asymmetry_between_F8_P8 = Amplitude_Asymmetry(x,y,2);

% alpha,beta amplitude asymmetry channels F4 O2
x = EEG.data(6,N+1:2*N);
y = EEG.data(6,1:N);
alpha_Amplitude_Asymmetry_between_F4_O2 = Amplitude_Asymmetry(x,y,1);
beta_Amplitude_Asymmetry_between_F4_O2 = Amplitude_Asymmetry(x,y,2);

% alpha amplitude asymmetry channels F3 O1
x = EEG.data(2,1:N);
y = EEG.data(5,1:N);
alpha_Amplitude_Asymmetry_between_F3_O1 = Amplitude_Asymmetry(x,y,1);

% alpha amplitude asymmetry channels O1 F7
x = EEG.data(5,1:N);
y = EEG.data(1,N+1:2*N);
alpha_Amplitude_Asymmetry_between_O1_F7 = Amplitude_Asymmetry(x,y,1);

% alpha relative power channel P3
x = EEG.data(4,1:N);
alpha_Relative_Power_P3 = Relative_Power(x);

% alpha relative power channel P4
x = EEG.data(7,1:N);
alpha_Relative_Power_P4 = Relative_Power(x);

% alpha relative power channel O1
x = EEG.data(5,1:N);
alpha_Relative_Power_O1 = Relative_Power(x);

% alpha relative power channel O2
x = EEG.data(6,1:N);
alpha_Relative_Power_O2 = Relative_Power(x);

% alpha relative power channel T8
x = EEG.data(8,1:N);
alpha_Relative_Power_T8 = Relative_Power(x);

% alpha relative power channel P7
x = EEG.data(1,1:N);
alpha_Relative_Power_P7 = Relative_Power(x);

% alpha relative power channel P8
x = EEG.data(3,1:N);
alpha_Relative_Power_P8 = Relative_Power(x);

```

## **Appendix 6: Note for the HLS C-code**

*\*The contents of the HLS C-code and the stand-alone ARM CPU C-code is contained in a zip file attached with this thesis.\**