Sistema de Predicción Epileptogenica en Lazo Cerrado Basado en Matrices Sub-Durales

James Brian Romaine

Memoria presentada para optar al grado de Doctor por la Universidad de Sevilla

Director,Tutor: **Manuel Delgado Restituto**

Director: **Ángel Rodríguez Vázquez**

Departamento de Electrónica y Electromagnetismo
Universidad de Sevilla

May 2019

I would like to dedicate this thesis to my family, friends and co-workers ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

James Brian Romaine
May 2019

Acknowledgements

Firstly, I would like to thank Manuel Delgado Restituto and Ángel Rodríguez Vázquez for their constant collaboration and intuitive guidance throughout my PhD.

Secondly, I would like to thank Miguel Angel Lagos Florido and Rafaella Fiorelli Martegani for their design guidence and help with test diagnostics and technical designs.

Last but not least, I would also like to thank Verónica Siles Santana and Marco Trevisi for their constant support, friendship and patients, throughout the course of this PhD.

This work has been supported by the Spanish Ministry of Science and Innovation under grant TEC2016-80923-P and the FEDER Program.

Abstract

The human brain is the most complex organ in the human body, which consists of approximately 100 billion neurons. These cells effortlessly communicate over multiple hemispheres to deliver our everyday sensorimotor and cognitive abilities.

Although the underlying principles of neuronal communication are not well understood, there is evidence to suggest precise synchronisation and/or de-synchronisation of neuronal clusters could play an important role. Furthermore, new evidence suggests that these patterns of synchronisation could be used as an identifier for the detection of a variety of neurological disorders including, Alzheimers (AD), Schizophrenia (SZ) and Epilepsy (EP), where neural degradation or hyper synchronous networks have been detected.

Over the years many different techniques have been proposed for the detection of synchronisation patterns, in the form of spectral analysis, transform approaches and statistical based studies. Nonetheless, most are confined to software based implementations as opposed to hardware realisations due to their complexity. Furthermore, the few hardware implementations which do exist, suffer from a lack of scalability, in terms of brain area coverage, throughput and power consumption.

Here we introduce the design and implementation of a hardware efficient algorithm, named Delay Difference Analysis (DDA), for the identification of patient specific synchronisation patterns. The design is remarkably hardware friendly when compared with other algorithms. In fact, we can reduce hardware requirements by as much as 80% and power consumption as much as 90%, when compared with the most common techniques. In terms of absolute sensitivity the DDA produces an average sensitivity of more than 80% for a false positive rate of 0.75 FP/h and indeed up to a maximum of 90% for confidence levels of 95%.

This thesis presents two integer-based digital processors for the calculation of phase synchronisation between neural signals. It is based on the measurement of time periods between two consecutive minima. The simplicity of the approach allows for the use of elementary digital blocks, such as registers, counters or adders. In fact, the first introduced processor was fabricated in a 0.18μm CMOS process and only occupies 0.05mm^2 and consumes 15nW from a 0.5V supply voltage at a signal input rate of 1024S/s. These low-area and low-power features make the proposed circuit a valuable computing element in closed-loop neural prosthesis for the treatment of neural disorders, such as epilepsy, or for measuring functional connectivity maps between different recording sites in the brain.

A second VLSI implementation was designed and integrated as a mass integrated 16-channel design. Incorporated into the design were 16 individual synchronisation processors (15 on-line processors and 1 test processor) each with a dedicated *training and calculation module*, used to build a specialised epileptic detection system based on patient specific synchrony thresholds. Each of the main processors are capable of calculating the phase synchrony between 9 independent electroencephalography (EEG) signals over 8 epochs of time totalling 120 EEG combinations. Remarkably, the entire circuit occupies a total area of only 3.64 mm^2.

This design was implemented with a multi-purpose focus in mind. Firstly, as a clinical aid to help physicians detect pathological brain states, where the small area would allow the patient to wear the device for home trials. Moreover, the small power consumption would allow to run from standard batteries for long periods. The trials could produce important patient specific information which could be processed using mathematical tools such as graph theory. Secondly, the design was focused towards the use as an *in-vivo* device to detect phase synchrony in real time for patients who suffer with such neurological disorders as EP, which need constant monitoring and feedback. In future developments this synchronisation device would make an good contribution to a full system on chip device for detection and stimulation.

El cerebro humano es el órgano más complejo del cuerpo humano, que consta de aproximadamente 100 mil millones de neuronas. Estas células se comunican sin esfuerzo a través de ambos hemisferios para favorecer nuestras habilidades sensoriales y cognitivas diarias.

Si bien los principios subyacentes de la comunicación neuronal no se comprenden bien, existen pruebas que sugieren que la sincronización precisa y/o la desincronización de los grupos neuronales podrían desempeñar un papel importante. Además, nuevas evidencias sugieren que estos patrones de sincronización podrían usarse como un identificador para la detección de una gran variedad de trastornos neurológicos incluyendo la enfermedad de Alzheimer(AD), la esquizofrenia(SZ) y la epilepsia(EP), donde se ha detectado la degradación neural o las redes hiper sincrónicas.

A lo largo de los años, se han propuesto muchas técnicas diferentes para la detección de patrones de sincronización en forma de análisis espectral, enfoques de transformación y análisis estadísticos. No obstante, la mayoría se limita a implementaciones basadas en software en lugar de realizaciones de hardware debido a su complejidad. Además, las pocas implementaciones de hardware que existen, sufren una falta de escalabilidad, en términos de cobertura del área del cerebro, rendimiento y consumo de energía.

Aquí presentamos el diseño y la implementación de un algoritmo eficiente de hardware llamado "Delay Difference Aproximation" (DDA) para la identificación de patrones de sincronización específicos del paciente. El diseño es notablemente compatible con el hardware en comparación con otros algoritmos. De hecho, podemos reducir los requisitos de hardware hasta en un 80% y el consumo de energía hasta en un 90%, en comparación con las técnicas más comunes. En términos de sensibilidad absoluta, la DDA produce una sensibilidad promedio de más del 80% para una tasa de falsos positivos de 0,75 PF / hr y hasta un máximo del 90% para niveles de confianza del 95%.

Esta tesis presenta dos procesadores digitales para el cálculo de la sincronización de fase entre señales neuronales. Se basa en la medición de los períodos de tiempo entre dos mínimos consecutivos. La simplicidad del enfoque permite el uso de bloques digitales elementales, como registros, contadores o sumadores. De hecho, el primer procesador introducido se fabricó en un proceso CMOS de 0.18μm y solo ocupa 0.05mm^2 y consume 15nW de un voltaje de suministro de 0.5V a una tasa de entrada de señal de 1024S/s.

Estas características de baja área y baja potencia hacen que el procesador propuesto sea un valioso elemento informático en prótesis neurales de circuito cerrado para el tratamiento de trastornos neuronales, como la epilepsia, o para medir mapas de conectividad funcional entre diferentes sitios de registro en el cerebro.

Además, se diseñó una segunda implementación VLSI que se integró como un diseño de 16 canales integrado en masa. Se incorporaron al diseño 16 procesadores de sincronización individuales (15 procesadores en línea y 1 procesador de prueba), cada uno con un *módulo de entrenamiento y cálculo* dedicado, utilizado para construir un sistema de detección epiléptico especializado basado en umbrales de sincronía específicos del paciente. Cada uno de los procesadores principales es capaz de calcular la sincronización de fase entre 9 señales de electroencefalografía (EEG) independientes en 8 épocas de tiempo que totalizan 120 combinaciones de EEG. Cabe destacar que todo el circuito ocupa un área total de solo 3.64 mm^2.

Este diseño fue implementado teniendo en mente varios propósitos. En primer lugar, como ayuda clínica para ayudar a los médicos a detectar estados cerebrales patológicos, donde el área pequeña permitiría al paciente usar el dispositivo para las pruebas caseras. Además, el pequeño consumo de energía permitiría una carga cero del dispositivo, lo que le permitiría funcionar con baterías estándar durante largos períodos. Los ensayos podrían producir información importante específica para el paciente que podría procesarse utilizando herramientas matemáticas como la teoría de grafos. En segundo lugar, el diseño se centró en el uso como un dispositivo *in-vivo* para detectar la sincronización de fase en tiempo real para pacientes que sufren trastornos neurológicos como el EP, que necesitan supervisión y retroalimentación constantes. En desarrollos futuros, este dispositivo de sincronización sería una buena base para desarrollar un sistema completo de un dispositivo chip para detección de trastornos neurológicos.

Table of contents

List of figures

List of tables

Preface

Organisation

1 Introduction **5**

1.1 The larger brain 6

1.2 Connectivity and the smaller brain 8

1.2.1 Neurons 8

1.2.2 Neural oscillations 10

1.3 Phase and Synchronisation 13

1.3.1 Basic concept 13

1.3.2 Synchronisation and phase locking 14

1.3.3 Synchronisation in the brain 16

1.4 Synchronisation in pathological brain states 17

1.4.1 Synchrony in neurological disorders 19

1.5 Phase synchronisation methods 20

1.5.1 Analytical measures 20

1.5.2 Numerical measures 25

1.6 VLSI implementations of PS estimators 26
1.7 Thesis Objectives 28

2 Delay Difference Analysis for Phase Synchronisation Computation 31
2.1 Proposed algorithm 32
2.2 Parameters setting 37
2.2.1 Quantisation 37
2.2.2 Sample distribution 37
2.2.3 Window length 39
2.3 Verification 40
2.3.1 DDA-PLV correlation 43
2.3.2 SNR 44
2.3.3 Sensitivity 46
2.4 Conclusions 51

3 Synchronisation Processor 53
3.1 Overview 54
3.2 Design 55
3.2.1 Minimum detection 55
3.2.2 Time stamp 57
3.2.3 Calculation 59
3.2.4 Filter 60
3.2.5 Detect 62
3.3 Methods 63
3.3.1 Design flow 63
3.4 Discussion 68
3.4.1 Indexing 68
3.4.2 Minimum detection scheme 68

3.4.3 Filtering . . . 68
3.4.4 Power/Accuracy trade-off . . . 69
3.4.5 Resource allocation . . . 70
3.5 Experimental results . . . 72
3.5.1 Experimental Setup with Neural Recordings . . . 74
3.5.2 Epilepsy Detection Results . . . 77
3.5.3 Functional Connectivity Results . . . 78
3.6 Conclusion . . . 84

4 Multi-channel synchronisation processor 85
4.1 Overview . . . 86
4.2 Design . . . 88
4.2.1 Input conversion SPI . . . 88
4.2.2 Output conversion PSI . . . 89
4.2.3 Epoch selection . . . 91
4.2.4 Window, epoch and memory addressing logic . . . 92
4.2.5 Flow . . . 93
4.2.6 Neural multiplexing . . . 94
4.2.7 Post processing . . . 97
4.2.8 Training and Thresholds overview . . . 98
4.3 FPGA Implementation . . . 104
4.4 Conclusions . . . 110

5 Conclusions 111
5.1 Conclusions . . . 112

References 115

Appendix A Graph Theory concepts 123

Appendix B Code listings **129**

B.1 Matlab Code DDA 129

B.2 Matlab Code Threshold 132

B.3 VHDL Code DDA 133

B.4 RTL-compiler 148

B.5 MBED-C++ test platform code 149

Appendix C Pseudo codes **155**

C.1 Pseudo code DDA hardware flow 155

C.2 Pseudo code for training and calculation hardware flow 157

Appendix D Control signals for future 16-channel testing platform **159**

Appendix E James Romaine, Publication list and curriculum **163**

List of figures

1.1 Main lobes of the larger brain 7

1.2 Action potential and anatomy of the neuronal cell 10

1.3 Overview of common neuronal oscillatory bands 12

1.4 Example of phase rotations around a stationary wave 15

1.5 Graph theory, application to real data 18

2.1 Transition periods and approximated phases of two signals 34

2.2 Synchronisation indexes obtained through the DDA algorithm 35

2.3 Results from DDA and Hilbert transform approaches 36

2.4 Effects of quantisation on algorithm 38

2.5 Effects of sample distribution on algorithm 39

2.6 Effects of window size on algorithm 41

2.7 $\mathrm{FR}_1084_B lock_0161, electrodepositionsM5$ 43

2.8 $\mathrm{FR}_1084_B lock_0161, electrodepositionsGA2$ 44

2.9 SNR effects 45

2.10 patient #7, SI results from sensitivity test. 48

2.11 Zoom synchronisation indexes for patient #7 50

2.12 Average Sensitivity against FPR for DDA + SI and PLV + HT 50

3.1 2 channel synchronisation processor block diagram 56

3.2 Minama detection logic and application 58

3.3 Exponential filter block diagram 62
3.4 VHDL simulation of parameters and verification 65
3.5 VHDL simulation of minima detection 65
3.6 VHDL simulation showing the results for phi and detection 65
3.7 VIVADO schematic of the prototype synchronisation processor 66
3.8 Detailed VIVADO schematic of the prototype synchronisation processor 66
3.9 CADENCE layout synchronisation processor 67
3.10 2 channel synchronisation processor correlation and power trade-off's . 70
3.11 Bar chart showing the FPGA resource allocation for the designed DDA algorithm 71
3.12 AMS-0.18-micro microscopic view 72
3.13 test-setup 72
3.14 spimeasurement 73
3.15 mbed-setup 76
3.16 $\mathrm{FR}_1084_B block_0161, electrodepositionsM5$ 77
3.17 EEG connectivity maps for patient #7 81
3.18 Graph theory results for patient #7 82
3.19 Graph theory contour map results for patient #7 83

4.1 16-Channel ASIC implementation top level block diagram overview . . 87
4.2 Microscopic view of 16-Channel ASIC 88
4.3 1 to 160 SPI for neural input data 90
4.4 160 to 1 SPI for captured synchronisation values 90
4.5 Control logic for 16-channel device 93
4.6 16-channel synchronisation processor flow 94
4.7 Neural input multiplexing scheme and dedicated control logic 96
4.8 Modified exponential filter block diagram 98
4.9 Training and calculation module block diagram 103

4.10 Dedicated control logic overview for memory latching 104
4.11 Array of 15 synchronisation processors synthesised using VHDL code . 106
4.12 Single synchronisation processor VHDL synthesis 106
4.13 Training logic VHDL simulation 107
4.14 Training logic VHDL simulation2 107
4.15 Output results from VHDL simulation of 16-channel processor 109

A.1 graphtheory basic concepts 127

List of tables

1.1 Basic overview of brain functions for specific lobes 7

1.2 Synchrony in different neurological disorders. 20

1.3 State of the art comparison table 28

2.1 Records used in the verification of the phase synchronisation processor 42

2.2 DDA and PLV results sensitivity test 49

3.1 Power breakdown of the synchronisation processor based on the blocks in Fig.3.1 75

3.2 Normalised performance summary 75

4.1 Table showing which neural data streams are connected to specific processor during each of the 8 epochs of time 92

4.2 Table showing the memory allocation during calculation phase of the patient specific thresholds 101

4.3 Table showing the signals for processor 1 during VHDL simulation of 16-channel design 108

4.4 Table showing all design resource allocation for the DDA hardware implementation. 108

Preface

This thesis was written to give a broad overview of the work conducted during the doctoral scholarship over a period of approximately 5 years. The main goal during the life of the project was the design and implementation of low powered and lower resource consuming VLSI hardware, for the efficient and reliable detection of synchronisation between neural assemblies.

The work was conducted in several key stages:

Firstly, through the research of state of the art in the field of neurological disorders and synchronisation, we gathered a thorough understanding of the current limitations and design challenges faced in this field of study. Using this information, we then targeted the design of a new mathematical algorithm, which focused on solving the most commonly faced problems. Furthermore, to test the algorithmic design a MATLAB implementation was introduced which was compared against the state of the art in terms of comparable features and accuracy. At this point, important aspects are tested such as the signal-to-noise ratio, algorithmic sensitivity, sliding window size and sampling rate distribution to name a few.

To follow, the algorithm was implemented into a field programmable gate array (FPGA) using a hardware description language (VHDL), which allowed us to verify the algorithm running at the logic gate level. This also provided us with prototype metrics which could then be used to validate the scalability of the device compared to the state of the art.

The FPGA implementation of the design was later mapped into CADENCE as a register technology level (RTL) circuit. This design, operating at 1.2V, gave us a good estimation as to the real life functionality of the circuit and algorithmic design. Once the 1.2V design was verified at the RTL level in CADENCE, a 0.5V sub-threshold biased design was implemented using custom layout techniques. Afterwards, the ASIC design was tested in the laboratory using a mixture of test apparatus such as a logic analyser for basic variable verification and micro-controllers for longer in-depth neural signal simulations.

Later, a 16-channel mass VLSI implementation was integrated. This design incorporated 15 main synchronisation processors based on the previously integrated 2-channel processor. In order to integrate this design a similar method to that of the previous design was adopted. This included the FPGA implementation and verification before a CADENCE based custom layout and verification.

NOTE: Due to a change in the MPW policy of the selected foundry, AMS AG, regarding its 0.18µm CMOS process (MPWs were finally cancelled in 2018), the 16-channel VLSI DDA processor was received 13 months after the tape-out. The limited time that was available for the completion of the thesis meant that this design could not be tested in the laboratory. Nevertheless, FPGA simulations have been incorporated into the thesis as a verification of functionality.

Organisation

This thesis has been separated into five chapters.

Chapter 1 describes the most relevant principles needed for the rest of the Thesis. Starting with a brief overview of the anatomy of the brain at the system level ("the larger brain"), including its distinct regions and basic functions, we later go on to describe the anatomy of the brain at the cell level ("the smaller brain") and the electrical signals, such as action potentials (APs) and local field potentials (LFPs), which are involved.

Moving forward, we introduce the concept of phase and synchronisation in both stationary signals and neural signals. This includes the introduction of the most common methods of detection and how synchronisation is being treated as one of the most fundamental underlying principles for the encoding and transfer of information between neuronal clusters.

Afterwards, we give a small overview of some of the most common neurological disorders which suggests that synchronisation could be used as a tool for the detection of pathological states.

Finally, we introduce the most recent advances in hardware implementations for synchronisation detection algorithms.

Chapter 2 presents the proposed Delay Difference Analysis (DDA) algorithm for phase synchronisation computation. The chapter discusses the mathematical premises behind the algorithm and illustrates its functionality in real world scenarios. It also shows how parameters such as quantisation, sample distribution or window evaluation length, affect synchrony estimations.

Next the algorithm is verified in the Matlab environment using neural recordings from 10 epilepsy patients. These recordings sum in total over 1800 hours of neural data

and contain over 60 annotated seizures. The noise tolerance and sensitivity performance of the DDA algorithm is evaluated and compared against more conventional techniques to give a true impression of the algorithms capabilities.

Chapter 3 introduce the design of a VLSI phase synchronisation processor based on the algorithm described in chapter 2. The chapter firstly gives a broad overview of the design including its strengths and weaknesses. Next, the steps followed towards the implementation of the VLSI design are presented. They included the design in an FPGA, followed by the synthesis with an RTL compiler and finally the custom layout in the CADENCE environment.

Then, the experimental results from the prototype are introduced. After describing the laboratory test setup, a breakdown of the design's power consumption is presented. Then, several measurements using neural recording data from the mentioned epilepsy dataset are presented. They include functional connectivity and graph theoretical results.

Chapter 4 includes the design and implementation of a multi-channel phase synchronisation processor based on the initial design in chapter 3. This section is laid out similar to the previous section in that firstly, all of the individual digital blocks and digital flow functionality is introduced. This is accompanied by many illustrations of the digital blocks as well an illustration of the overall flow and the implemented ASIC micro-photograph.

Next, a FPGA implementation of the VLSI circuit is introduced and system simulations are presented.

Finally, **Chapter 5** provides some final concluding comments and draw possible future work which will be conducted using the designed algorithm and VLSI implementations.

Chapter 1

Introduction

1.1 The larger brain

The human brain weighs approximately 1.5kg and accounts for only 2% the total weight of the average human. Nevertheless, it is the most diverse and complex organism known in the universe today. The brain is responsible for everything that we do in our everyday lives from basic motor functions such as, walking, eating or talking to sensory functions such as seeing, listening and touch. It is also responsible for our underlying cognitive abilities in the form of thinking, memories, attention, logical processing and reasoning. It also gives rise to consciousness, which is the state of being aware of one's surroundings and the ability to comprehend them. The brain also has some impressive attributes. It has been theorised that the brain has a storage capacity of upwards of a quadrillion bytes and can perform upwards of 10 quadrillion calculations per second. This complex organ has been extensively studied over the past centuries with new knowledge and understanding being discovered every day.

The larger brain is made up of the cerebrum, cerebellum and brain stem. The cerebrum itself is made up of two hemispheres, the left and right which make up the cerebral cortex. Although it is the largest and most distinguishable part of the brain it only consists of 16 billion of the estimated 100 billion total neurons.

The cerebral cortex can be further broken down into four main lobes, namely, the frontal, parietal, temporal and occipital lobes.

The frontal lobe is the largest of the four main lobes and maintains connections to almost all other parts of the brain. Its main responsibilities include personality and decision making, as well as motor control. This lobe makes sense of the environment around us as well as making sense of memories and emotions. It also provides us with working memory which allows us to hold relative information for short periods in order to drive the attention span.

The parietal lobe interprets our sense of touch (somatosensation) and monitors the position of the body as well as the relative position of the limbs. It integrates information from senses in order to focus our attention on important tasks. It also receives information from the occipital lobe in terms of location, size and speed of objects and incorporates them into the sense of touch in order to interoperate our surroundings.

The temporal lobe is where memories, emotions and language comprehension are stored. It is also key to the recognition of objects, places and people.

Frontal	Parietal	Temporal
Regulation of personality	Interoperates language and words	Understanding of Language
Behaviour and emotions	Touch and pain	Long term Memory
Judgement	vision	Hearing
Planning and problem solving	Hearing	Sequence and organisation
Speech	motor	Facial recognition
Writing	sensory	Emotions
Smell	memory	Emotions
Intelligence		
Concentration		
Self awareness		
Emotional reactions		

Table 1.1 Table showing the basic control functions of the frontal, parietal and temporal lobes [1]

The occipital lobe is located at the far back of the cerebral cortex. It decodes information arriving from the retina. It mainly informs us of where, how and what we are seeing, via the detection of high contrast edges and their orientation and motion. This information is then passed to the parietal lobe for response mechanisms or to the temporal lobe to see if we recognise the object.

Fig.1.1 shows the relative positions and surface coverage of the four main lobes.[1], and Table 1.1 summarizes some of the basic functions for the frontal, parietal and temporal lobes.

The cerebellum is tucked underneath the occipital lobe and is actually small compared to the four main lobes. Nevertheless, the cerebellum contains approximately 70 billion neurons. This part of the brain is responsible for our talents from playing musical instruments to athletic agility. It does this by controlling complex movements, it detects errors and creates adjustments to compensate, which in turn strengthens neural circuits throughout the brain.

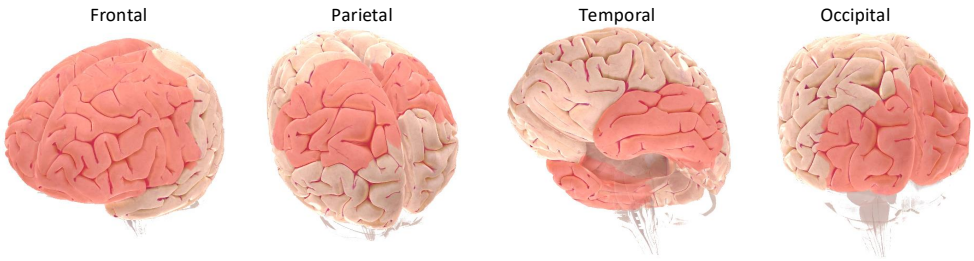

Fig. 1.1 Overview of the human brain including its four major regions of the cerebrum [1]

1.2 Connectivity and the smaller brain

1.2.1 Neurons

Neurons are the individual cells which make up the central nervous system. There exist many different types of neurons all of which are packed into small networks of upwards of 10000 neurons per network and are interconnected by kilometres of wiring per cubic millimetre. These are called individual neural circuits. Their main purpose is the transfer of information, which allows for the specific and complex movements, associated with human motor skills as well as allowing for cognitive perception and memory [2].

The basic neuron structure can be seen in Fig. 1.2. Consisting of three main parts. Firstly, the dendrites, which are the main collectors of information (they are analogous, to that of an input to a system).

The soma is the neuronal structure which is responsible for the central processing of the incoming information (similar to that of a systems function $f(x)$). The soma performs a non-linear threshold based comparison of the incoming information from the dendrites.

The axon, which is the final main component of the neuronal structure, is considered the main delivery system of information. This sends the soma generated information to other neurons within the network. At the end of each axon, there are axon terminals which connect to dendrites via synapses. In addition, there is a layer of fat which surrounds the axon known as the myelin sheath, which is an electrically insulating layer. The sheath regulates the proper transmission of electrical signals, helping speed up electrical transmission down the axon.

The transmitted information is referred to as an action potential (AP). The AP is a product of concentration gradients, which refers to the difference in ion concentrations inside and outside of the neuronal cell.

In general an AP is generated as follows. Firstly, neuronal cells are surrounded by an extracellular fluid (EF) which is full of positively charged sodium and potassium ions. While the inside of the neuronal cell there are also positively charged sodium and potassium ions. Nevertheless, the EF has many more positively charged ions than the inside of the neuronal cell. This leads to a high concentration gradient which causes a negative potential inside of the cell, known as a resting membrane potential (RMP).

The RMP is the state in which neurons spend most of their time, it is a negative concentration gradient of approximately -70 millivolts.

The membrane of the neuronal cell is very permeable to potassium ions, meaning that they leak out of the cell towards the EF via leakage channels. The membrane of the neuronal cell is also partially permeable to sodium ions which also flow out of the cell via sodium leakage channels. This creates instability within the cell. Nevertheless, a cell is constantly trying to to maintain its resting potential so it has tiny ion pumps which pump potassium back into the cell and sodium out.

In addition to the ion leakage channels and pumps, the neuronal cell also has two voltage controlled ion gates, one for sodium and the other for potassium. during RMP, these gates are always closed. When an event occurs at the dendrites of the cell, such as an AP sent from another neuron or a nerve input it causes the voltage-gated sodium channels to open, allowing sodium ions to flow into the cell which in turn causes a depolarisation in the current which tries to push the resting membrane potential towards 0 millivolts. If the membrane potential reaches a critical threshold, roughly speaking the threshold of approximately -55 millivolts an AP is formed. The AP itself is a positive electrical impulse which can push the membrane potential to +30 mV. Potassium channels then open and the sodium channels close in order to repolarise the cell. The basic overview of the AP can be seen in Fig. 1.2, which marks the opening and closing times of each voltage controlled gate. The potassium gate, however, stays open slightly too long creating a hyperpolarisation after the AP which then slowly moves back towards the resting potential [3, 4].

The interconnection between two neurons occurs at the synapse, which is an electro or chemical junction. The human brain contains anywhere between 100 and 500 trillion synapses of different varieties. Synapses are located at the dendrites (input terminals) and the axon terminals (output terminals).

In a chemical synapse, once the pre-synaptic (output terminal) AP or nerve impulse reaches the junction, a neurotransmitter chemical molecule is transmitted and diffused across a small space known as the synaptic cleft. These molecules then bind with neurotransmitter receptors on the postsynaptic (input terminal) neuron which opens or closes ion channels in the receiving neuron. This leads to a change in the membrane potential of the receiving neuron which can either lead to an AP or not. If the cell does fire it is what is known as an excitatory synapse, otherwise it is an inhibitory synapse. Since a neuronal cell has multiple synapse connections the firing of an AP is actually the combination or summation of all of the excitatory and inhibitory activity

at the input of the cell. If there are more excitatory signals, the membrane potential increase towards the AP threshold. For the signal to end, the synaptic cleft must then be cleaned of neurotransmitter.

Electrical synapse have a direct connection in the form of a gap junction, which allows current to flow extremely fast from one neuron to the next. This has the advantage of allowing neural circuits to synchronise rapidly. One of the benefits of an electrical synapse is the ability to allow current flow in both directions, meaning a bi-directional flow of information. It is believed that these types of synapses are used in fast response actions such as the 'fight or flight mode' in humans [5].

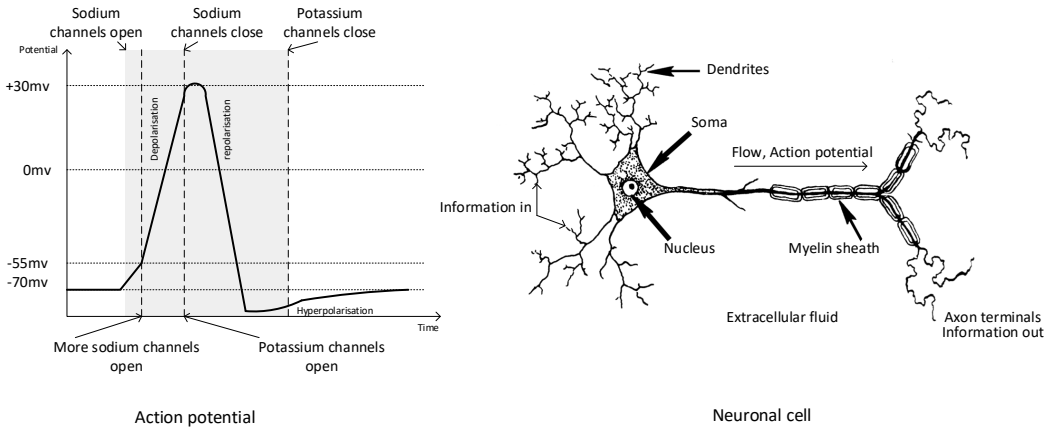

Fig. 1.2 Overview of neural connectivity, left, shows an example of an AP including approximate channel opening and closing times, right, shows the basic structure of a single neuron

1.2.2 Neural oscillations

The term neuronal oscillation refers to periodic variations in neural activity. These variations can be measured at different scales such as the microscopic scale, which could consist of the measuring of direct synaptic currents which produce spike trains of AP's, the mesoscopic scale, which consists of the summation of many synaptic currents from a Local network, named a local field potential (LFP), or the macroscopic scale, in the form of electroencephalography (EEG) data, which is the measure of multiple local networks [6].

The history of neural oscillations started with the first ever recordings of individual microscopic recordings. First recorded by Richard Caton in 1875, a physician from

Liverpool, England, who used electrodes to probe the exposed surface of the brain in animals. Using a reflecting galvanometer, he was successful in monitoring signals in the microamperes range [7].

Nevertheless, it was the work of Berger in 1929, who first reported that he had identified oscillations of approximately 8-12Hz coming from the brain. Berger named this the alpha band since it was the first oscillation detected from such a source. This was early on in his pioneering work on the study of EEG. During one of his experiments Berger noticed that while the patients eyes were closed the prominent oscillations of the alpha band were visible. In the power spectrum the alpha waves produced a lone prominent peak of at approximately 12 Hz. However, when the patient opened their eyes the oscillations seemed to diminish in amplitude and become faster and more erratic, in a way the signals seemed to desynchronise and the prominent peak disappeared. This is known as the 'Berger effect'. Noting this, it was apparent that the alpha oscillations were attenuated by other more complex frequencies due to the visual stimuli originating from the occipital lobe.

Further observations of Hans Bergers works were carried out by W. Grey Walter in 1934, who devised an experiment entitled 'Thought and Brain: A Cambridge experiment', in Which one set of results was quoted as:

'When the eyes of a subject were open the lines were irregular, but when the subject's eyes were closed they showed as regular sinusoidal type signals occurring approximately every 10 seconds. Interestingly, When the subject closed their eyes and was asked to perform a simple arithmetic problem, the regular patterns disappeared, similar to that of when the patient's eyes were open. The lines continued to appear irregular until the simple problem was solved at which time the regular patterns reappeared'.

The fundamental observation by these pioneers revealed that the electrical fields recorded, must have been the combination of thousands of neurons firing in synchrony as stated by [8].

Since then, further advances in technology have led to the identification of multiple active frequency bands in which oscillations tend to occur. Namely,

- Infraslow ($< 0.2Hz$) and $\delta = 0.2 - 3.5Hz$. These wave patterns are usually seen during sleep of newborn infants or adults with serious brain diseases.

- $\theta = 4-7.5Hz$. These waves have been identified to mainly oscillate in the parietal and temporal regions of children's brains, and in healthy adults. Nonetheless, these slow varying oscillations are absent or indistinguishable under normal

conditions. Nevertheless, these oscillations do appear in healthy adults who show signs of emotional stress or frustration.

- $\alpha, \mu = 8 - 13Hz$. These oscillations tend to occur during the period when a person is awake but in a very relaxed state, mainly with the eyes closed. The introduction of complex thought was proven to also diminish alpha waves causing alpha blocking due to complex thought.

- $\beta = 14 - 30Hz$. The lower spectrum of the beta band usually falls into the same category as the alpha band, in that it can be disturbed by small amounts of complex activity or tension. The upper spectrum of the beta band is only affected by extreme thought processes, such as complex arithmetic. The beta band is usually considered as the awake and alert band, as we constantly process visual and auditory senses, as well as perform cognitive tasks throughout the day.

Other oscillatory bands include, $\gamma = 30 - 90Hz$ and the upper limit of High Frequency Oscillations $HFO > 90Hz$. Recent research is, now starting to identify the gamma band as a communication layer between individual neural circuits [9]. Fig. 1.3, shows the decomposition of an original neural signal measured using EEG, into some common respective oscillatory bands.

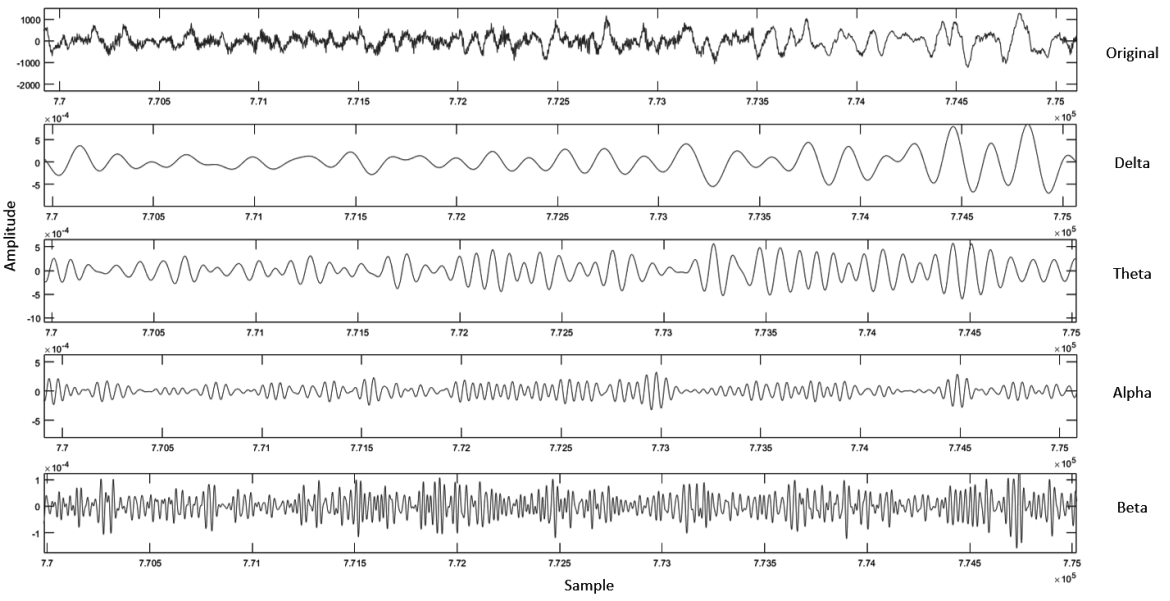

Fig. 1.3 Example of the most common neuronal oscillatory bands

The interconnection of neurons forms several states of connectivity, three main types are as follows [10] :

1. *structural connectivity*, which refers to the anatomical connectivity via the structure of major tracts (i.e, white matter) connecting local and remote cortical regions of the brain.

2. *functional connectivity*, which is a statistical measure of the amount of connectivity in one region vs another and their dependence upon each other.

3. *effective connectivity*, the amount of connectivity in one cortical region which directly affects another cortical region by either activation or depression

1.3 Phase and Synchronisation

1.3.1 Basic concept

Phase is defined as the angular change in a signal over time. This can be represented as the anticlockwise rotation around the unit circle. Each adjustment is measured in terms of degrees, ranging from 0° to 360°, which indicates a full rotation around the unit circle. The rotation can also be measured in the SI unit radians, where $360° = 2\pi$, radians.

A sinusoidal wave can be expressed as:

$$x(t) = A \cdot \sin(\omega t + \theta) \tag{1.1}$$

where, A is the amplitude, θ (this has no relation to the theta oscillatory band described above) is the initial phase of the signal and $\omega = 2\pi f$ denotes the rate of change of the vector rotation around the unit circle. The rate of change of phase increases or decrease as the frequency of the signal. If the frequency remains constant, the rate of change in phase will also remain constant. These types of signals are known as stationary signals.

Non-stationary signals, however, refers to a signal in which frequency changes over time, such as a synthetic sinusoidal chirp in which the rate of change in phase is constantly increasing. Another non-stationary source is that of EEG signals as LFP's tend to incorporate a wide range of frequencies.

Fig. 1.4, shows an example of the change in phase angle from a typical sinusoidal waveform and a cosine waveform. In this case, the first peak in the sine wave represents 0 radians and the first minimum in the cosine wave represents, π radians. As the two signals evolve, by the time the first minimum has appeared in the sine wave the phase angle has increased to π radians and the cosine has evolved to 2π radians.

Since the sine wave and cosine wave are both stationary at 10Hz, the phase of both signals increases at a fixed rate. In comparative situations, it is desirable to take the phase difference between the two waveforms. This can give rise to information on their frequency offsets with respect to one another and can be calculated as:

$$\theta = \phi_{signal1} - \phi_{signal2} \tag{1.2}$$

Using specific markers in each signal such as minima or maxima, if the first derivative $\frac{d(\phi_{signal1} - \phi_{signal2})}{dt}$, is constant it can be said that the signals have a constant phase difference. Since the two signals in Fig. 1.4 are of a constant 10Hz, the phase difference between the two is constant. In this case, the constant phase change is π radians [11].

1.3.2 Synchronisation and phase locking

In general terms, synchronisation is defined as any two-coupled (phase locked) oscillating systems, which have a relationship in both phase and frequency, but at the same time are independent of amplitudes. In non-disturbed and noise free systems synchronisation is usually described as the phase locking between instantaneous phase angles such that $n \cdot \phi_{s1} - m \cdot \phi_{s2} = const$, where ϕ_{s1} and ϕ_{s2} are the instantaneous phase angles of signal 1 and signal 2, respectively. m and n, are integer based weights which need adjusting based on the source of the oscillating system.

In the example in Fig. 1.4, although the two signals are in anti-phase there does exist a relationship between the two, in terms of phase locking.

In the case of non stationary signals, the phase locking condition should be considered as: $|n \cdot \phi_{s1} - m \cdot \phi_{s2} - \delta| < const$. Where δ, is some average phase shift. This condition indicates that although the rate of change is not completely constant, it should reside (fluctuate) around a common constant [12].

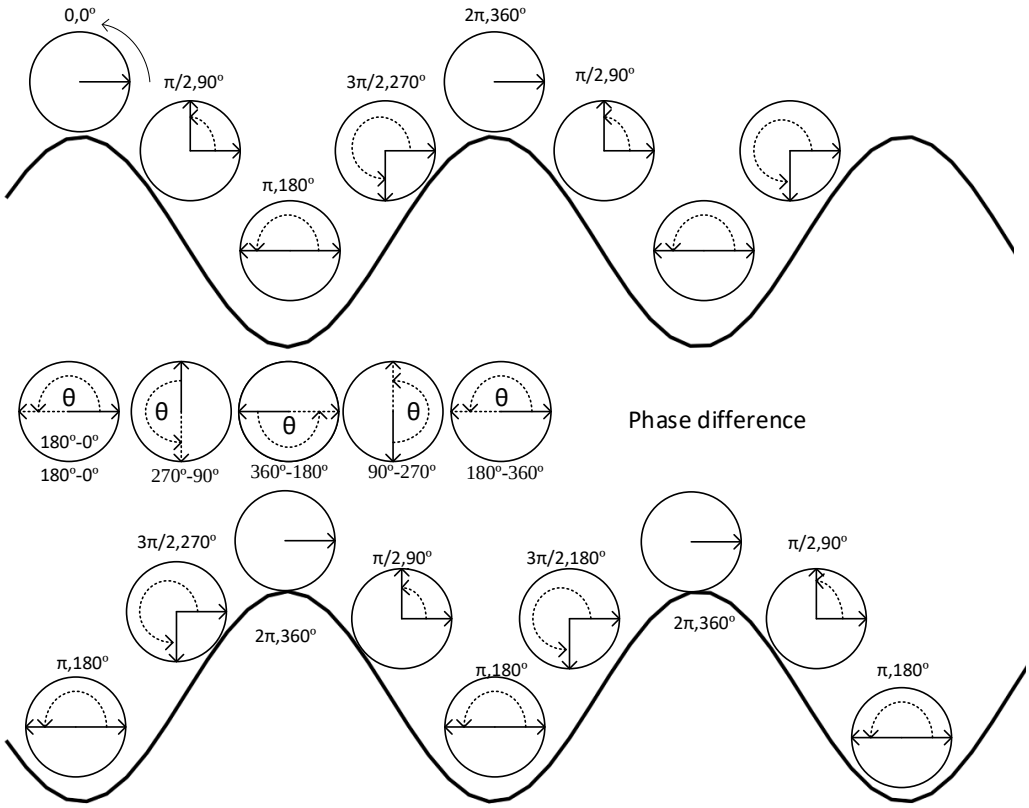

Fig. 1.4 Phase: example of how phase rotates anti-clockwise around the unit circle in a stationary sine wave

In many cases of phase locking, the values m and n are usually set to a fixed 1:1 ratio. This is especially true if the signals originate from the same source such as the brain. Nevertheless, this phase locking attribute also holds true for signals of different frequencies, such as a stationary sine wave of 10Hz and a stationary sine wave of 20Hz. Although the frequencies are not the same there does exist a relationship in phase known as cross-frequency phase locking [13, 14].

1.3.3 Synchronisation in the brain

The human brain seems to effortlessly communicate large quantities of information over multiple regions of the brain. However, one of the questions still unanswered is how neural communication between assembles throughout the larger brain actually works. That‘s to say, what decides what information is accepted or declined at the input of neural assemblies.

For many years it was assumed that general information processing was carried out by changes in firing patterns of so-called 'smart neurons'. which were dedicated to certain tasks and functions. Nevertheless, the work derived from Hans Berger and others has led to the question as to whether synchronisation of neural assemblies is the key to information processing and learning. The fact that neural oscillatory patterns exist, indicates that to some degree, neural synchrony at the given frequency must be intrinsic of some type of neural assembly collaboration.

Over recent years, new evidence is emerging that precise synchronisation and desynchronisation of neural assemblies may, in fact, be the key to the encoding of information transfer within the brain. This can be seen in various reviews and experimental results from [15–17]

Two main hypotheses have been suggested for the efficient transfer of information between neural circuits and brain regions. Namely, coherence, in the form of phase synchronisation and gating by inhibition.

Phase synchronisation could be used as an identifier between neural assemblies, similar to that of two radios set to the same frequency channel. If the presynaptic response from one neural circuit is oscillating at the correct excitability rate of the postsynaptic neurons, then the information is accepted. However, if the presynaptic response is oscillating at a different rate (different frequency), then the postsynaptic inputs of the second neural circuit then the information is blocked or diminished, as with a radios set at different frequencies.

In general terms, it is the phase of an oscillatory neuron population which determines their excitability, hence their spike timings. It is, therefore, the phase relationship between neuron populations which determines routing. This has been proposed to occur in the gamma band [18].

The flow of information is controlled by an increase of alpha oscillations which inhibit the neural population which is not firing in synchrony [19].

1.4 Synchronisation in pathological brain states

The topic of the synchrony between neural circuits for proper functional neuronal communication is a trending topic in neuroscience today.

When comparing brain states we like to consider the differences between normal and pathological brains, in order to identify the connectivity differences which may exist between them. Therefore, the data of pathological brain state patients is usually compared with that of a control subject (a patient with no medical history of neurological disorders). Nevertheless, in the case of neurological disorders such as epilepsy, the patients inter-ictal epileptic period can be associated with a non-pathological brain state, hence used in further analysis.

In order to distinguish between these two brain states, we use a common mathematical tool known as graph theory. This allows us to gather insight into the structural, functional and effective connectivity of the brain. It allows us to visualise the overall connectivity as a function of time.

For an in-depth look into the most common mathematical calculations and an overview of the steps required to calculate complex graph networks. Please see appendix A.

It is widely accepted that the brains neural communication is characteristic of a small world network where neural clusters are well connected and transfer information over small distances. Therefore comparing the small-worldness of a non-pathological brain state with that of pathological states can identify the changes which occur in terms of overall functional connectivity.

Fig. 1.5, shows an overview of the steps required for identifying small-worldness in epilepsy when taken from EEG sources.

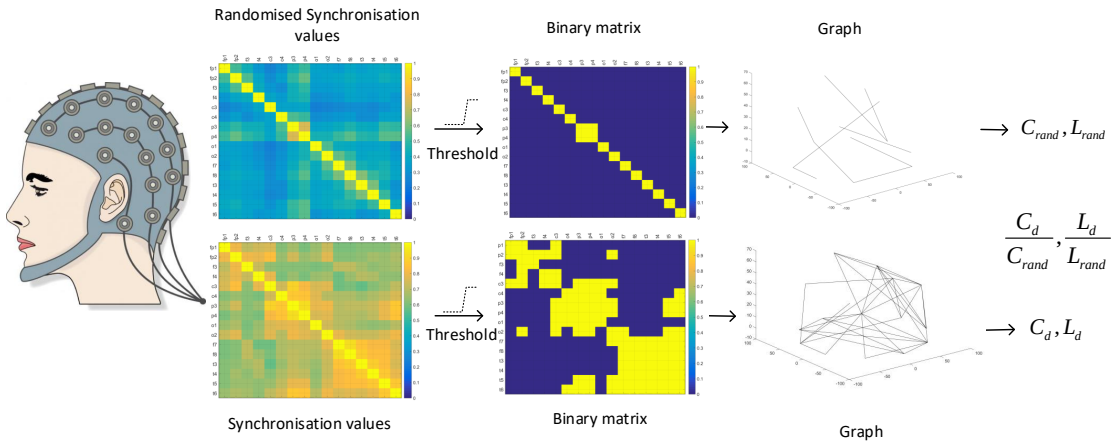

Fig. 1.5 Example of how graph theory methods can be applied to real data. In this case, we compare an epileptic patients ictal period to that of their non ictal period. We then calculate M and A using an binary/undirected graph and calculate metrics as per needed

1. Firstly, all synchronisation values for every possible combination of electrode positions are computed, creating a $O = NXM$ matrix. Where N corresponds to the electrode combination and M corresponds to each synchronisation value for the given electrodes over time.

2. Next, two matrices $G1 = NXN$ For the non ictal period and $G2 = NXN$ for the ictal period are constructed. This is done by creating a single mean representative value for each electrode combination from O such that:

$$G_{x(i,j)} = \frac{1}{M} \sum_{z=0}^{M} O_{i,j} \tag{1.3}$$

3. The next step is to apply thresholds to each matrix G, for the purpose of creating two binary based adjacent matrices A_x. The threshold which is applied is subject to the experimental data and the related subject at hand. In many cases, various thresholds are tested. This allows for the identification of the strongest connections and builds 3d topologies which give rise to information on C, L, k.

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} & \dots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \phi_{23} & \dots & \phi_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{d1} & \phi_{d2} & \phi_{d3} & \dots & \phi_{dn} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & \dots & 0 \\ 1 & 0 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 1 & \dots & 1 \end{bmatrix}$$

4. Finally, we can calculate λ, γ and η, by calculating the mean clustering coefficients and the mean shortest path lengths.to distinguish the small-world characteristics and small-worldness.

1.4.1 Synchrony in neurological disorders

Although each neurological disorder has distinct causes and effects, synchrony has been found to manifest itself as an identifier, to distinguish between the normal and neurologically damaged brain. Table 1.2 [20], shows an overview of the most common neurological disorders as described above and how long and short-range communication is affected by increases or decreases in synchronisation. Other numerical methods such as graph theoretical analysis (please see appendix A) have also been shown to be a useful tool in the identification of both functional and structural connectivity.

In Alzheimer Disease (AD) synchronisation has been observed to increase in the θ and δ bands whilst, conversely, reducing in both the α and β bands at resting state. Using a synchronisation likelihood (SL) measure, it was determined that these bands decreased in synchronisation over both long and short range recording sites [21–23]. Whilst global efficiency measures are higher in AD patients compared to the normal brain, the brain's ability to synchronise decreased significantly over multiple EEG bands in AD patients when their eyes were closed according to [24]. Synchronisation in AD patients has also been analysed in patients during memory working tasks. These include EEG and functional Magnetic Resonance imaging (fMRI) analysis of functional connectivity. Once again it was shown that a decrease in synchronisation was present when compared to control patients specifically in the α and β bands [25].

The way our brain synchronises does not just affect cognitive functions. In fact, there is evidence which suggests that synchronisation in the β band is related to pre-motor movements (i.e preparation for movement). Nevertheless, once the actual movement takes place the β activity is replaced by γ activity [26, 27]. Moreover, there is evidence to suggest that an increase in β activity could be responsible for tremors in Parkinson Disease (PD) [28]. Where in patients who do not suffer from PD a short

Disorder	Neural synchrony
Epilepsy	Increase in local synchrony and evidence for long range reduction
Schizophrenia	Reduction in local and long range synchrony
Alzheimer	Reduced synchrony, evidence for reduced functional connectivity
Parkinson	Increase in synchrony

Table 1.2 Synchrony in different neurological disorders.

period of β synchrony prior to a motor movement is replaced by fast γ bursts during the movement (where the β oscillations are dulled by inhibition), in patients with PD, however, the hypersynchronous β activity cannot be dulled and hence the γ oscillations responsible for the movement cannot take place [29].

It has been proposed that memory, attention and perceptual organisation are associated with synchronisation in the β and γ bands [30]. Various studies of Schizophrenia (SZ) show that a deficit in perceptual organisation is a common symptom and could be related to a reduction in synchrony in the β and γ bands [20].

Epilepsy (EP), on the other hand, is a well studied neurological disorder with overwhelming evidence suggesting that seizures occur due to hyper-synchronous networks. High-frequency oscillations in the γ band have especially been observed in EEG both during the ictal and pre-ictal periods [31]. It has also been observed that neurons synchronise with high precision in the β band pre seizures, however, synchronous activity is reduced during the ictal event [32]. Conversely, a decrease in synchronisation was proposed by Le Van Quyen et al. who examined the synchronous activity of eight patients finding a pre-ictal reduction in the β band in 77% of seizures [33].

1.5 Phase synchronisation methods

1.5.1 Analytical measures

Two main closed-form analytical approaches have been proposed for the estimation of phase angles. One is based on the Hilbert Transform and the other on the Wavelet Transform.

The Hilbert transform (HT) creates a projection of the original signal on to the imaginary plane in such a way that positive frequencies are phase shifted by $-\pi/2$ and negative frequencies by $+\pi/2$, while keeping the amplitude unaffected. Analytically, the

Hilbert transform $\tilde{s}(t)$ can be interpreted as the convolution of $s(t)$ with the function $h(t) = 1/(\pi t)$ and is given by [34]:

$$\tilde{s}(t) = \frac{1}{\pi} \cdot p.v \int_{-\infty}^{+\infty} \frac{s(\tau)}{t-\tau} \cdot d\tau \tag{1.4}$$

Where the integral is defined in terms of the Cauchy principal value $p.v$ and $\tilde{s}(t)$ is a (90°) phase shifted version of the original signal, $s(t)$.

In continuous time signals the HT can be represented as the convolution of a real signal $s(t)$ and a function $h(t) = \frac{1}{\pi t}$, such that $\tilde{s}(t) = s(t) * \frac{1}{\pi t}$ [35].

In the frequency domain the HT can be written as:

$$\tilde{s}(f) = X(f) \times (-jsgn(f)). \tag{1.5}$$

Where $(-jsgn(f))$, is the HT of $X(f)$ [35].

An analytical construct introduced by Gabor in 1946 utilises the HT, to create an analytical signal, given by:

$$s_a(t) = s(t) + j\tilde{s}(t) = A(t)e^{j\phi(t)} \tag{1.6}$$

The resulting complex signal $s_a(t)$, has the property that there exist no spectral components for negative frequencies, that's to say the Discrete Time Fourier transform of $s_a(t)$, is $S_a(e^{j\omega}) = 0, -\pi < \omega < 0$.

The instantaneous phase of the analytical signal $s_a(t)$ can be calculated as:

$$\phi(t) = \tan^{-1}\left[\frac{\tilde{s}(t)}{s(t)}\right] \tag{1.7}$$

Two other important metrics can be extracted from the HT. Firstly, the instantaneous amplitude of the signal, which can be extracted as [36]:

$$A(t) = (s(t)^2 + \tilde{s}(t)^2)^{\frac{1}{2}} \tag{1.8}$$

Secondly, the instantaneous frequency, which is given as the first derivative of the phase,

$$\omega(t) = \frac{d\phi}{dt} \tag{1.9}$$

The HT, however, is a linear function which in retrospect only works on stationary signals. Nevertheless, an adaptation of the HT can be used known as the Hilbert-Huang Transform (HHT).

The HHT incorporates an empirical mode decomposition (EMD) to the signals before taking the HT. The EMD consists of breaking down the signal into a finite number of components which are called Intrinsic Mode Functions (IMFs). These are extracted from an arbitrary time series $x(t)$, by means of *sifting*. In order to sift all extrema, both maximum and minimum. The first IMF can be extracted as [37]:

$$h_1(t) = x(t) - m_1(t). \tag{1.10}$$

Where, $x(t)$ is the time series in question and $m_1(t)$, is the mean of the upper and lower envelope derived from the extrema.

However, $h_1(t)$, is usually not sufficient to satisfy the constraints of an IMF and hence the process is repeated such that:

$$h_{11}(t) = h_1(t) - m_{11}(t). \tag{1.11}$$

Where $m_{11}(t)$, is the mean of the upper and lower envelopes of $h_1(t)$.

In general, the process is as follows:

$$h_{1k}(t) = h_{1(k-1)}(t) - m_{1k}(t). \tag{1.12}$$

where k is the number of iterations to find the first IMF $C_1(t) = h_{1k}(t)$.

Furthermore, the residual part the signal should be obtained from such that:

$$h_1(t) = x(t) - c_1(t). \tag{1.13}$$

at the end of the process the original signal can be represented as:

$$x(t) = \sum_{i=1}^{n} c_i(t) + r_n(t). \tag{1.14}$$

The HT is then performed on each IMF [38]:

$$c_{ai}(t) = c_i(t) + j\tilde{c}_i(t) \tag{1.15}$$

The instantaneous phase of the signal can be extracted as:

$$\phi_i(t) = \tan^{-1}\left[\frac{\tilde{c}_i^2(t)}{c_i^2(t)}\right] \tag{1.16}$$

A similar approach for estimating phase angles [39] uses a definition based on the Wavelet Transform (WT). Here, the phase variable is defined as,

$$\varphi_s(t) = \arctan\frac{\mathrm{Im}(W(t))}{\mathrm{Re}(W(t))} \tag{1.17}$$

where Im(·) and Re(·) denote imaginary and real part, respectively, and $W(t)$ is the convolution of the band-limited signal $s(t)$ with a Morlet wavelet $\psi(t)$,

$$\begin{aligned} W(t) &= \int_{-\infty}^{+\infty} \psi(t-\tau)s(\tau) \cdot d\tau \\ \psi(t) &= \exp\left(\frac{-t^2}{2\sigma^2}\right) \cdot \exp\left(j\omega_0 t\right) \end{aligned} \tag{1.18}$$

where σ is the decay rate of the wavelet and ω_0 is the centre frequency of the signal band. Both analytical methods give similar results when applied to neurophysiological data [40].

The Phase Locking Value (PLV) or mean phase coherence usually accompanies the HT or WT methods as an index for quantifying the amount of synchronisation between two signals.

$$PLV = \left| \frac{1}{N} \sum_{K=0}^{N-1} e^{j\phi(t)} \right| = \frac{1}{N} \cdot \sqrt{\left[\sum_{K=0}^{N-1} \cos(\Delta\phi_k) \right]^2 + \sum_{K=0}^{N-1} \sin(\Delta\phi_k) \Bigg]^2} \tag{1.19}$$

The PLV converts instances of the relative phase $\triangle\varphi$ into unit vectors on the complex plane. If N of said vectors all have the same instantaneous phase (i.e the same vector direction) then the PLV will average out to 1 which equates to fully coupled oscillations for that period. However, in the case that the relative phase angles are very different the PLV will result in a value close to 0.

In some cases, a normalised PLV is more desirable than raw values as it gives a better approximation of the difference between normal state PLV and large changes, due to stimulus. The normalised values can be extracted as,

$$PLV_{norm} = \frac{PLV - mean(PLV_{ts})}{\sigma(PLV_{ts})} \tag{1.20}$$

Where ts, is a small baseline selection of samples taken before the episode. In this case, σ, represents the standard deviation of the baseline.

This metric is actually based on the Z-score and provides a zero mean offset.

1.5.2 Numerical measures

Phase synchronisation measures can be alternatively derived from the phase space trajectories of the signals. Using time-delay embedding procedures [41], the phase space trajectory of a time series $\{s(k)\}$ is obtained by constructing the vectors,

$$\vec{S}_i = \{s(j), s(j+d), \ldots, s(j+(m-1)d)\} \tag{1.21}$$

where $i = 1 \ldots N$, $j = i + \text{mod}[N-(m-1)d]$, $m \geq 1$ is the embedding dimension and $d \geq 1$ is an arbitrary but fixed time increment.

Based on this topological representation, the phase synchronisation between two signals can be estimated by computing the cross-correlation between the probabilities of recurrence in their respective phase spaces [42]. This approach has been used recently for quantifying functional connectivity in EEG recordings [43]. The probability of recurrence of a signal measures the likelihood that each phase space vector returns to its neighbourhood after a time delay τ. It is calculated as:

$$p_r(\tau) = \frac{1}{N-\tau} \sum_{i=1}^{N-\tau} \Theta\Big(\epsilon - \|\vec{S}_i - \vec{S}_{i+\tau}\|\Big) \tag{1.22}$$

where ϵ is a pre-defined distance threshold, $\|\cdot\|$ is a norm to calculate the distance between vectors and Θ is the Heaviside function. The cross-correlation CPR of two trajectories is calculated as:

$$CPR = \frac{\Big\langle (p_{r,1}(\tau) - m_1) \cdot (p_{r,2}(\tau) - m_2) \Big\rangle}{\sigma_1 \cdot \sigma_2} \tag{1.23}$$

where m_1 and m_2 are the mean, and σ_1 and σ_2 are the standard deviations of $p_{r,1}(\tau)$ and $p_{r,2}(\tau)$, respectively. If both signals are in synchrony, their probabilities of recurrence will peak at the same time and $CPR \sim 1$. Contrarily, if the signals are not synchronised, low values of CPR can be expected.

Numerical techniques have been also proposed for the quantification of synchrony, as an alternative to the PLV index in 1.19 [44]. Assuming that the phase angles of both signals have been previously estimated, a synchronisation index can be calculated based on the Shannon entropy of the phase difference distribution. Having an estimate $P_{se}(l)$, $l = 1, \ldots, L$ of the relative frequency of finding a phase difference $\triangle\varphi = \varphi_{s1} - \varphi_{s2}$ in a

certain bin l, the index is given by,

$$S_{se} = 1 + \frac{1}{\ln(L)} \sum_{l=1}^{L} P_{se}(l) \cdot \ln[P_{se}(l)] \tag{1.24}$$

where L is the number of bins. Note that S_{se} is comprised in the interval $[0, 1]$, where $S_{se} = 0$ corresponds to a uniform distribution (no synchronisation) and $S_{se} = 1$ corresponds to a distribution localised in one point. The estimate of this index heavily depends on L [44].

Another synchronisation index is based on the conditional probability $P_{cp}(k, l)$ to find the phase of a first signal in a bin k when the phase of the second signal falls in the l-th bin, $l = 1, \ldots, L$. Denoting as M_l the number of hits of the second signal in the l bin, a synchronisation index can be defined as,

$$S_{cp} = \frac{1}{L} \sum_{l=1}^{L} \left| \frac{1}{M_l} \sum_{k=1}^{M_l} \exp[jP_{cp}(k, l)] \right| \tag{1.25}$$

It has been found that this index is particularly suitable for revealing weak interactions between signals [44–47].

1.6 VLSI implementations of PS estimators

As mentioned in Sec.1.5.1, one popular approach for the calculation of the instantaneous phase of a signal relies on the construction of the analytic representation $s_a(t) = s(t) + j\tilde{s}(t)$, where $\tilde{s}(t)$ is the Hilbert Transform of $s(t)$ [see 1.4]. Using time-frequency domain techniques, the Hilbert Transform can be derived by [48] (i) obtaining the Fourier transform of $s(t)$, (ii) nulling the negative frequency components, (iii) calculating the inverse Fourier transform and (iv) taking the imaginary part of the result. Given the non-stationary nature of neural signals, the Fourier transform should be applied over short intervals. This suggests the use of the Short Time Fourier Transform (STFT) algorithm instead of the classical Fast Fourier Transform (FFT) [36]. 1.4 and 1.19

A simpler method for the generation of the analytic signal representation $s_a(t)$ uses Hilbert Transformers based on digital filtering. These transformers can be implemented in complex domain or formed by the combination of two sub-components; a Hilbert filter with ideal transfer function $-jsgn(f)$ for the generation of the imaginary part of $s_a(t)$, and a delay equal to that of the Hilbert filter for the real part. In both cases,

the total group delay of the signal components should be kept constant to not cause phase errors. The vector analyzer presented in [49] follows the second implementation approach and employs 16-tap Hilbert finite impulse response (FIR) filters for obtaining $\tilde{s}(t)$, and 16-tap All-Pass FIR filters for implementing the delay. In a more recent implementation [50], the in-phase and quadrature components are directly provided by the mixed-signal acquisition front-end, and four 64-tap FIR filters are used for band selection.

Another transform-based approach for deriving the instantaneous phase of a signal uses Wavelets (see 1.18). In this case, the on-chip implementation of the method requires Multiply-and-ACcumulate (MAC) processors to correlate the input time series with the wavelet templates. As an example, [51] proposes a charge-recycling mixed-signal MAC processor for epilepsy detection in which time series of 1024 samples are serially loaded and correlated in parallel with all the Morlet wavelet templates (coded in 32 frequency bins with 4-bit coefficient resolution) stored in the on-chip memory.

The trigonometric functions involved in 1.7 and 1.17 as well as the computation of the phase locking value in 1.19 can be readily implemented by CORDIC (COordinate Rotation DIgital Computer) units. This is actually done in [52] where a digital signal processing (DSP) block formed by three 16-bit CORDIC cores and two 32-tap moving average FIR filters is used for computing the phase locking value (PLV) from the analytical signal $s_a(t)$.

Numerical methods for computing PS, such as the one based on probabilities of recurrence, $p_r(\tau)$, described in Sec. 1.5.2, typically use simpler logic than transform-based techniques at the price of increased latency and more memory resources. Thus, for instance, the evaluation of $p_r(\tau)$ in 1.22 requires the collection and combination of phase space trajectories, demanding for a large memory allocation. Nevertheless, it is worth observing that, depending on τ, there are coincident terms in the sum 1.22. This opens doors for reducing the computational complexity of the algorithm by avoiding recalculations. This is actually the approach followed in [53], where a differential scheduling for the computation of the correlation integral of a signal (similar to the calculation of recurrences) allows for a substantial reduction in complexity. Following the calculation of $p_r(\tau)$, a MAC unit should be used for the calculation of the cross-correlation in 1.23 to serve as a synchronization index.

As shown in 1.24 and 1.25, numerical methods can also be used for the computation of synchronization indexes. Although the computational complexity is relaxed as compared to PLV (in particular for the index based on Shannon entropy), long sliding

Table 1.3 Performance summary of proposals related to on-chip PS calculation

	Process	Supply	Core Area	Input rate	Clk Freq.	Power cons.	Computation capability
[49]	0.13μm	1.2 V	1.86 mm^2	7.2kHz (8-bit)	10MHz	400 μW	32 input-pairs multiplexed phase synchronization processor
[50]	0.13μm	1.2 V	1.27 mm^2	NA	NA	260 μW	32 input-pairs multiplexed phase synchronization processor
[51]	0.35μm	1.65 V	16 mm^2	NA	50Hz	NA[a]	128×1024 binary MAC operator
[48]	0.13μm	1.2 V	1.18 mm^2	40kHz (16-bit)	10.24MHz	3.64.4 μW	16-channel multiplexed 16-bit Hilbert Transform block
[52]	0.13μm	0.85 V	0.178 mm^2	1.7kHz (8-bit)	2.5MHz	102 μW	10-bit *PLV* Processor
[53]	90nm	1.0 V	0.143 mm^2	256Hz (9-bit)	4096Hz	2.34 μW	16-channel Correlation Integral Processor
[55]	90nm	1.0 V	1.02 mm^2	256Hz (16-bit)	522.24 kHz	57.3 μW	1-channel EMD processor extracting 5 IMFs and residue

[a]Power consumption mainly due to the 4×128 8-bit $\Sigma\Delta$ algorithmic ADCs. At 15 kHz parallel sample rate, the bank of ADCs dissipates 6.3 mW from a 3.3 V supply [54].

windows are needed to populate the histograms and obtain meaningful distribution of probabilities. Hence, circuit simplicity is counterbalanced with an increase in latency and memory depth.

Table 1.3 illustrates the performance of the different integrated circuits presented in this section. The two first rows are for complete PS processors while the rest corresponds to building elements which can be eventually used depending on the particular PS implementation strategy. The table clearly shows that the algorithms used for PS calculation, although easily programmable in computers, tend to occupy large area and consume significant power when dealing with silicon integration. The problem is even more acute in the context of functional brain connectivity, where multiple phase synchronisation indexes from different sites are needed. For instance, if functional connectivity has to be measured between 16 recording sites, as it is typically done in a standard 10-20 EEG electrode system, 120 different combinations of neural signal pairs should be simultaneously addressed. This means application specific integrated circuits (ASIC's) for functional connectivity quickly become far to big and/or power hungry, even if sophisticated multi-thread sharing DSP techniques are employed.

1.7 Thesis Objectives

This thesis aims to address three major aspects:

- Development of an algorithm for the detection and quantification of phase synchronisation between pairs of neural signals which significantly alleviate the constraints of power consumption and area occupation as compared to state-of-art methods. The algorithm should exhibit little or no degradation

on performance metrics such as seizure detection sensitivity, noise tolerance or functional connectivity computation.

- Implementation of a low-power, low area silicon prototype of the proposed algorithm to confirm its suitability for VLSI, and performance validation with human recordings in a realistic application scenario, namely, the detection of seizures in patients with different epileptic pathologies.

- Demonstration of the scalability of design through the implementation of a multi-channel processor for surface or intracranial EEG signals, and confirmation that the proposal represents a low-cost, low-complexity, programmable and portable solution for computing functional connectivity in wearable/implantable systems.

Chapter 2

Delay Difference Analysis for Phase Synchronisation Computation

In this chapter, an algorithm for phase synchronisation computation, which aims to alleviate the area/power consumption burden of silicon integration of previous approaches such as in Sec.1.6 is proposed. The synchronisation algorithm, called Delay Difference Analysis (DDA), is based on the piece-wise linear approximation in [12], however, it has been modified by us to achieve a more hardware friendly implementation. Furthermore, a new synchronisation index is introduced which allows for a more reliable identification of synchronisation peaks. The two components of the algorithm, were then subjected to tests, including: simulation of nearly two thousand hours of real neural data, noise analysis and sensitivity analysis. Moreover, we include the effects of sampling rate, window length and sample distribution.

All of the following was carried out by the author of this thesis.

2.1 Proposed algorithm

Similar to the transform-based methods, this proposal is also based on the estimation of instantaneous phases in band-limited signals, however, instead of using complex filtering or wavelet convolutions, phases are extracted through the identification of distinct marker events in wave forms. Without loss of generality, in this work, it is assumed that such events correspond to the local minima of the signal at time instants t_n. The time interval between two consecutive minima correspond to one complete cycle and, therefore, the phase increment during this period is exactly 2π. Hence, the signal phase can be approximated for any arbitrary time instant $t_n < t < t_{n+1}$ as,

$$\varphi_s(t) \approx 2\pi \frac{t - t_n}{t_{n+1} - t_n} + 2\pi n \tag{2.1}$$

where the term $2\pi n$ unwraps the phase angle and $T_n^s = t_{n+1} - t_n$ measures the duration of the transition period between the minima, as illustrated in Fig.2.1. Note that this approximation notably simplifies phase estimation as the problem basically reduces to a measure of time intervals. For this reason, the proposed algorithm is denoted as Delay Difference Analysis.

Based on the approximation in (2.1), an index for quantifying the synchronisation level between two signals has been proposed in [56]. It relies on creating histograms on how many times the events in one signal are preceded by events in the other. The approach has a low computational cost, however, similarly to the numerical techniques

described in Sec. 1.5.2, it requires long time series to build meaningful statistics and, in some sense, it wastes the time information between events.

The proposed approach for measuring synchronisation is similar to the PLV technique in (1.19) but avoids the use of trigonometric functions. DDA analyses the time difference between the transition periods of both signals, not their relative phase $\triangle\varphi$, and, remarkably, it operates on-the-fly with no need to store long signal sequences. The procedure simply consists in pairing transition periods from both signals as soon as they are detected. If a pair is formed between the i-th transition period of signal $s_1(t)$ and the j-th transition period of $s_2(t)$, the absolute difference $\triangle T_n = |T_i^1 - T_j^2|$ is calculated, where T_i^1 and T_j^2 denote, respectively, the durations of said transition periods. If two or more transition periods in one signal are detected before a transition period is found in the other, only the last transition period of the former signal is accounted for in the computation of $\triangle T_n$. Fig.2.1 shows an example of the procedure. This, in turn, increases the phase error distribution.

Let us denote by K the number of absolute differences $\triangle T_n$ computed in the k-th sliding window of length N, and let us assume with no loss of generality that $N = 2^m$. A synchronisation index between signals $s_1(t)$ and $s_2(t)$ can be defined as,

$$S_{dda}^{(r)}(k) = 1 - \frac{2^r}{N}\min\left(\sum_{n=0}^{K-1} |\triangle T_n| - T_{os}, \frac{N}{2^r}\right) \tag{2.2}$$

where $r \in [0, m]$ is a user-defined selectivity control parameter and T_{os} is a positive integer offset for adjusting the synchronisation index range. Fig.2.2(a) plots $S_{dda}^{(r)}$ for $T_{os} = 0$ and different selectivity configurations assuming that $s_1(t)$ is a sine wave at 20Hz, and $s_2(t)$ is a chirp whose frequency linearly sweeps from 10 to 30Hz. Both signals had a sampling rate of 1024S/s and were quantised at 10-b of resolution. Every calculated index was obtained assuming a non-overlapping sliding window of length $N = 1024$. Clearly, the higher the selectivity parameter, the narrower the band for which the synchronisation index obtains non-null values. No matter the r parameter, the plots in Fig.2.2 are not symmetrical around the tone frequency of $s_1(t)$; they decay linearly for chirp frequencies below said tone but fall away more smoothly for higher frequencies. This is because the number of absolute differences K is limited by the smaller frequency, f_{min}, of both signals as $K = N \cdot f_{min}/f_s$. In Fig.2.2(a), it is also observed that the peak value of the synchronisation index decreases with the selectivity parameter. This can be compensated, without drastically increasing the sampling frequency or the number of samples per observation window, by means of the offset

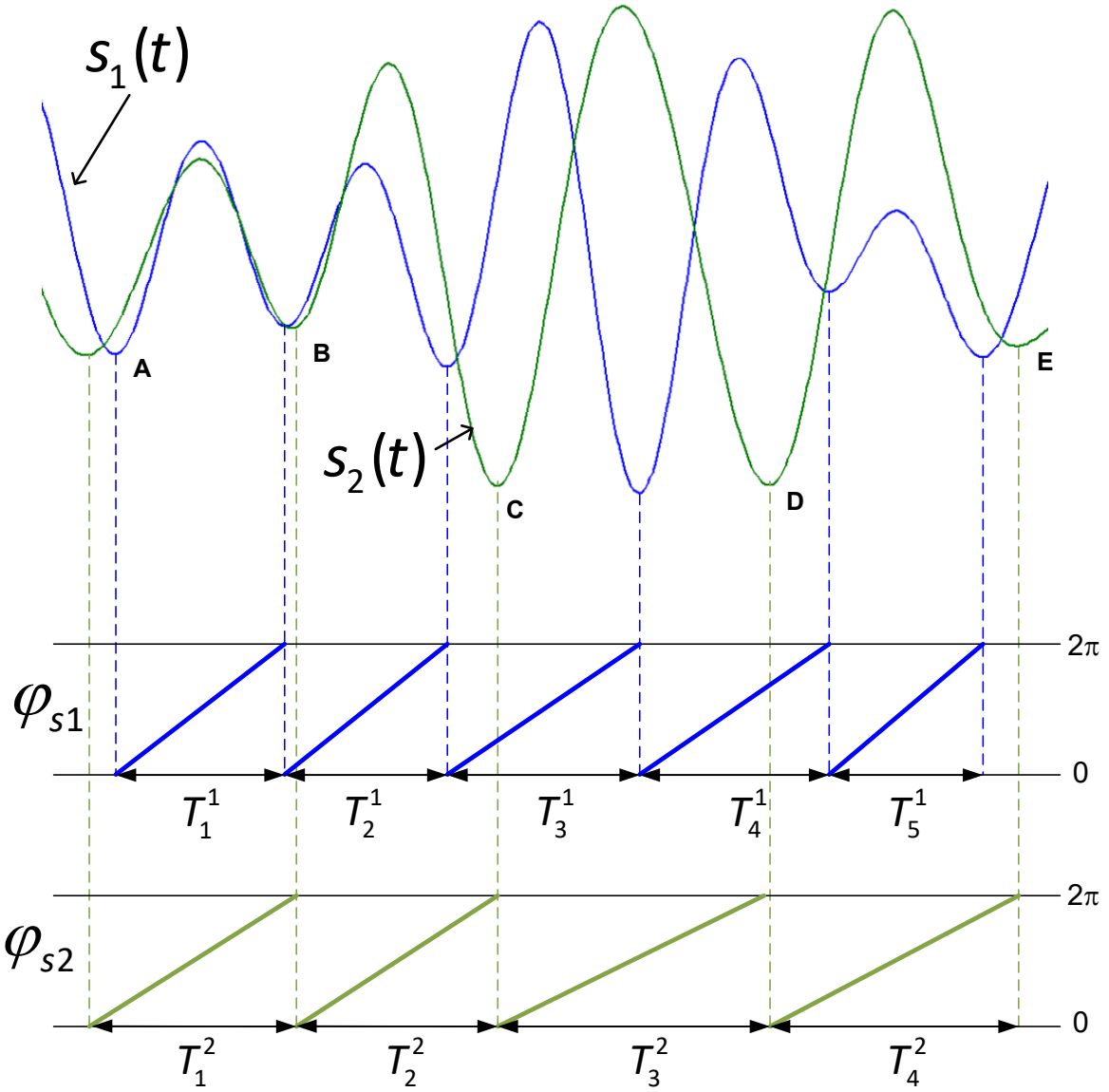

Fig. 2.1 Transition periods and approximated phases of two signals. Starting at point A, the DDA algorithm generates the following absolute differences between transition periods: $|T_1^1 - T_1^2|$, $|T_2^1 - T_2^2|$, $|T_3^1 - T_3^2|$ and $|T_5^1 - T_4^2|$.

parameter T_{os}. This is illustrated in Fig.2.2(b) where $T_{os} = 6$ making the synchronised indexes for all selectivity parameters to mostly cover the range between 0, indicating no coupling, and an approximate 1 for perfect coupling.

Note that too large of an offset may eventually make the synchronisation index larger than 1, hence, a limiter, which can be easily implemented in hardware by an overflow detector, should be used together with (2.2).

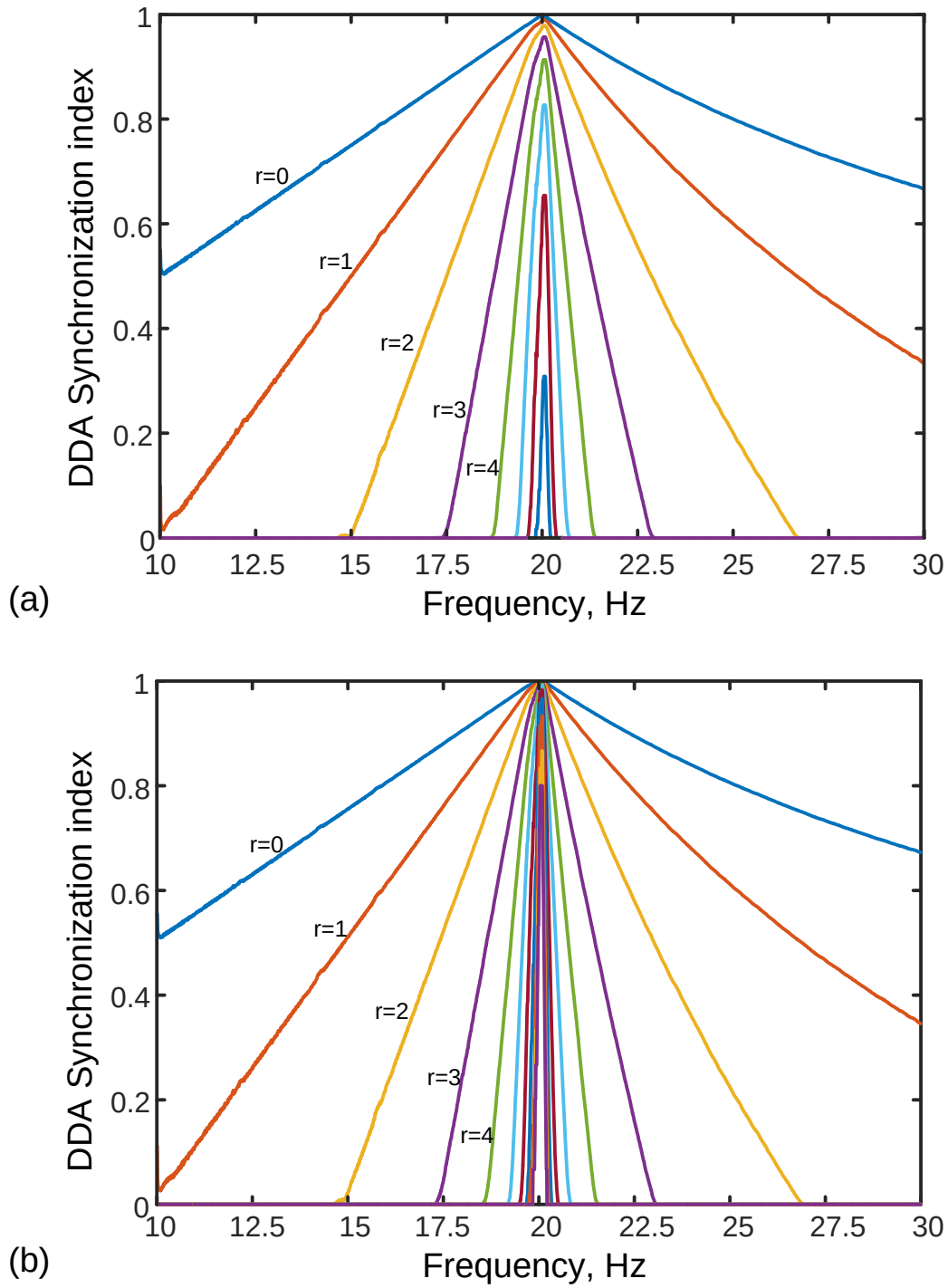

Fig. 2.2 Synchronisation indexes obtained through the DDA algorithm for different selectivity parameters, using a sine wave and a swept-frequency cosine as input signals. (a) $T_{os} = 0$; (b) $T_{os} = 6$.

For comparison purposes, Fig. 2.3 shows the PLV index (dashed line) for the same experiment as in Fig. 2.2 together with the index $S^{(4)}_{dda}$ with no offset. Both approaches give comparable results, with the DDA producing a piecewise linear approximation of the main lobe of the PLV index and eliminating the smaller side lobes. This is not surprising as the PLV index can be approximated as $PLV \simeq 1 + \alpha \sum_{k=0}^{N-1} |\triangle\varphi_k|/N$ for small $\triangle\varphi$ values, where α is a fitting parameter (see equation 1.19). Given the relationship between phases and time delays expressed in (2.1), a clear connection between the phase locking value and the Delay Difference Analysis can be established.

Appendix B.1, shows the MATLAB code used in the design and construction of the DDA algorithm.

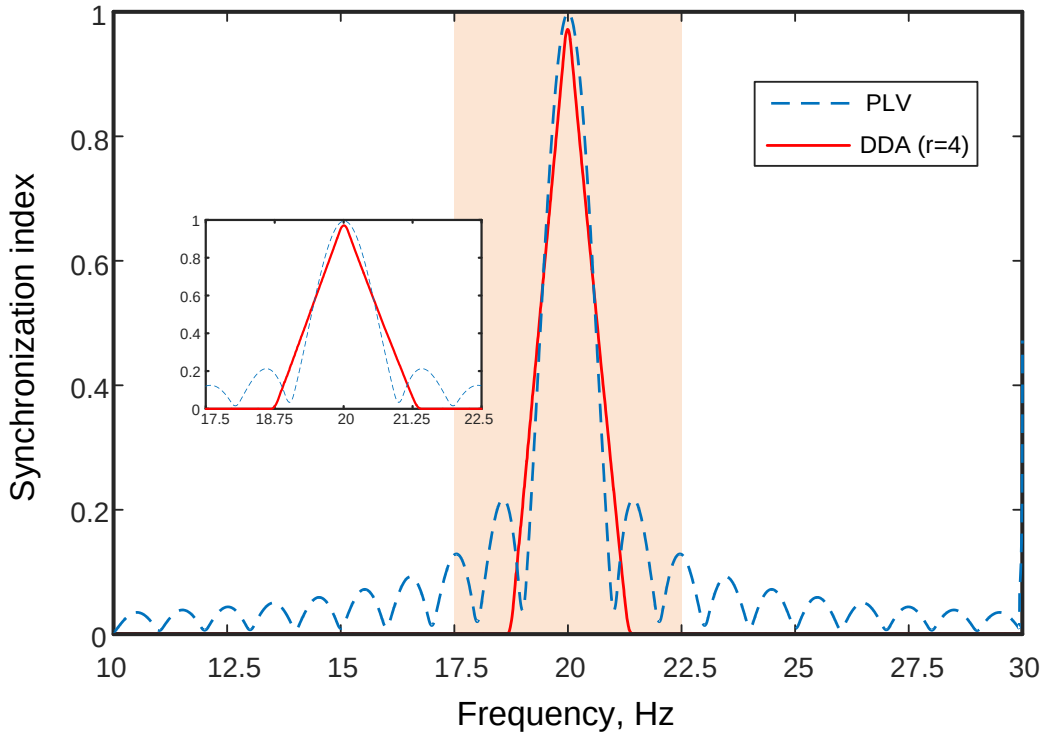

Fig. 2.3 Results from DDA and Hilbert transform approaches for the same experiment in Fig. 2.2. The inset shows a zoom of the shaded area.

2.2 Parameters setting

2.2.1 Quantisation

In this section, we try to explain the effects of quantisation on the output of the algorithm. The test was the same as in Fig. 2.2(a) however, quantising the two input signals 1 and 2, from 6 to 13-b of resolution. The characteristics of the test can be seen below. From the results shown in Fig. 2.4, we can note that 6, 7 and 8-b of resolution cause a noisy output, significantly more for $r = 0$ to $r = 2$, whereas from 9-b onwards the output becomes recognisably consistent with Fig. 2.2(a). In this case, we can see that after 9-b of resolution at a fixed sampling frequency of 1024 S/s the performance of the algorithm no longer increases. Nevertheless, to allow for multiple sampling frequencies a resolution of 10-b is recommended. Mathematically speaking, the number of bits needed is relative to the sampling frequency and signal frequency such that $2^n > 2 \cdot \frac{sampling frequency}{lowest signal frequency}$.

2.2.2 Sample distribution

The sample distribution is another variable which should be analysed for this algorithm, as we consider the phase errors in terms of samples. Fig. 2.5, shows the sample distributions for 3 different sampling rates 1024 S/s, 512 S/s and 256 S/s varied over a frequency range of 1 to 50Hz.

Firstly, in Fig. 2.5, we can note that the general sample distribution at fixed sampling rates tend towards a non-linear exponential distribution at lower frequencies. This means that during lower frequencies the accumulated errors will present much higher values than that of the higher frequencies for a set window length. Nevertheless, if we consider relatively low sampling frequencies such as seen in Fig. 2.5. We can deduce that an approximate linear distribution can be extracted for frequencies ranging from 5Hz upwards. Since the number of samples is defined as $\frac{f_s}{f}$ as f increases the number of samples gets smaller.

It is, however, important to point out that samples at high values of f will present themselves as the same value due to the rounding of samples in the distribution. As an example, at 1024 S/s and 48Hz the number of samples is $\frac{1024}{48} = 21.3$ and at 49Hz $\frac{1024}{49} = 20.89$, therefore 48Hz will obtain 21 samples after rounding and 49Hz will also obtain 21 samples.

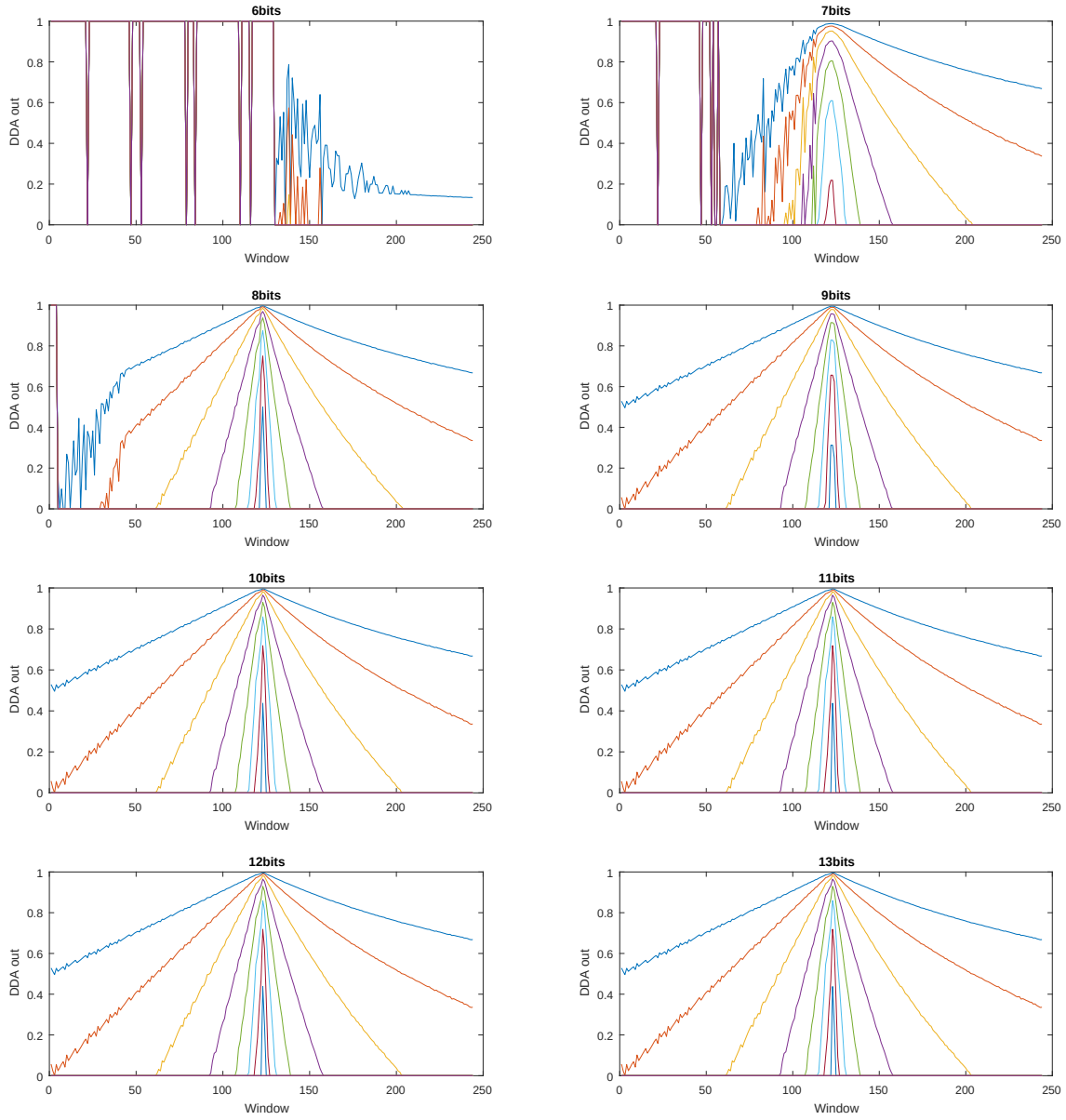

Fig. 2.4 Example of how quantisation of the input signals can effect the results obtained by the algorithm

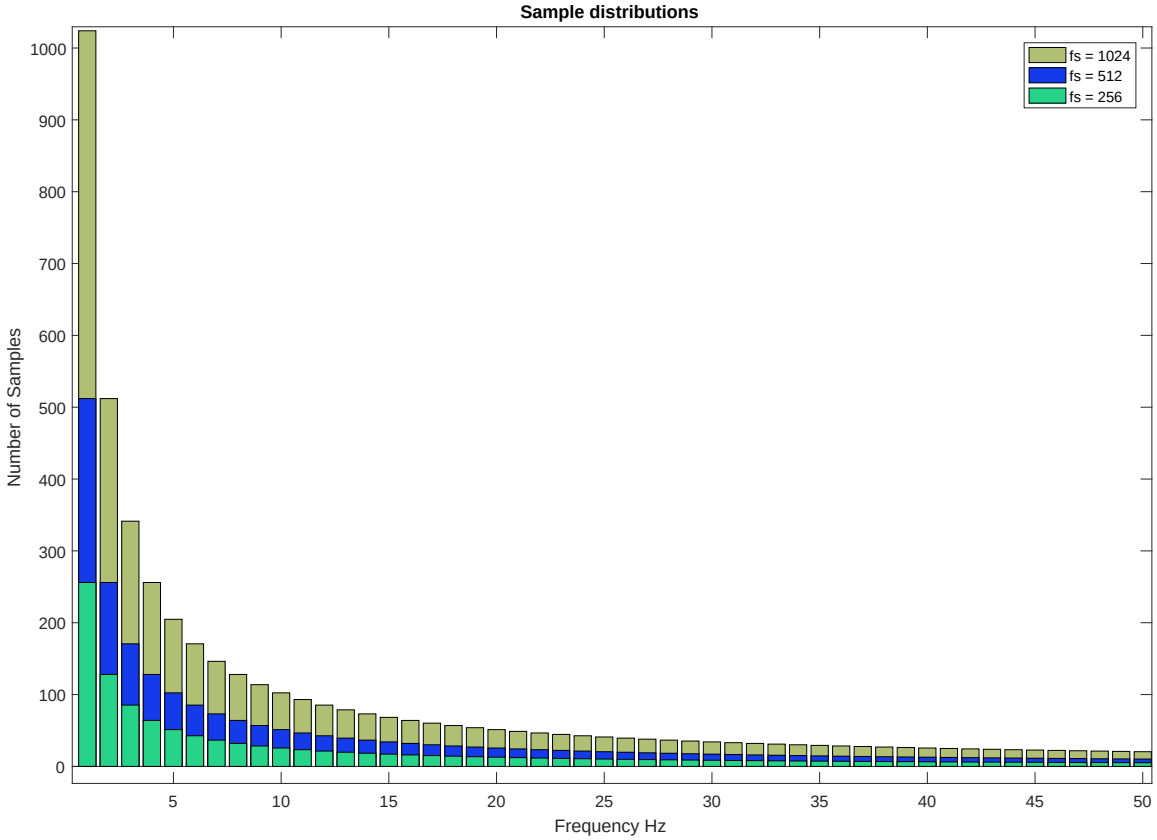

Fig. 2.5 Example of the non-linearity of a sample distribution at different sampling frequencies

Therefore the most accurate results will occur using a 512 or 1024 bit sampling rate from 5Hz upwards towards 30Hz. That said the sampling frequency can always vary based on the specific frequency band of interest. As an example, if we would like to monitor signals in the 80Hz range we could always increase the sampling frequency above 1024 as the distribution will be approximately linear at that frequency given the higher sampling rate. If we want to monitor the 10Hz to 20 Hz range it makes more sense to use a lower sampling frequency of 256 S/s, where the linearity is highest and the effects of quantisation are not prominent.

2.2.3 Window length

The window length is a common factor which can greatly vary the outcome of the results as well as the detection of specific biomarkers. In Fig. 2.6, the effect of various window

lengths on the DDA algorithm can be seen, once again using the same experiment as in Fig. 2.2(a). The variables for the test can be seen below.

1. sampling frequency fixed 1024
2. window size variable $32 \cdot 2^i$, where i is varied from 1 to 10
3. Signal 1: sine frequency fixed 20Hz
4. Signal 2: chirp frequency 10Hz-30Hz

The results show that for very small time windows, the output of the DDA suffers. This occurs since at the lowest frequency of 10Hz and the fixed sampling rate of 1024, a single period of the waveform is composed of approximately 124 samples. However, the window length is only 64 samples, this indicates that DDA is unable to collect a sufficient amount of data for accurate results, instead, creating noise perturbations. At the other extreme, when the window length is very long such as 32 seconds, the DDA over collects samples meaning the central 20Hz of the fixed sine wave never coincides solely with the 20Hz of the chirp signal. Instead, the error accumulations from the chirp consist of more widespread frequencies. Therefore, the DDA never reaches one and has broader angles. In the case of the medium length windows of approximately 0.5 to 1 second, we see an optimal performance. As the window is sufficiently short, that the two tone frequencies can exist solely together within 1 window and is long enough to collect a reliable amount of errors.

It is important to point out that the signals captured during EEG, posses what is known as the 3 N's, non-stationary, non-linear and noisy. This can make identification of specific markers difficult using linear time-based approaches. Nevertheless, it has been observed that well filtered EEG signals posses quasi-stationary attributes for periods lasting up to 1 second.

Therefore, with these two factors in mind, it remains clear that the optimal window length should be the same as or similar to that of the sampling frequency of the device. Once again it is still possible to vary the sampling frequency as long as the window length adjusts accordingly.

2.3 Verification

The algorithm has been verified and validated using neural recording data available in the European Epilepsy database (http://epilepsy-database.eu/) [57]. This database

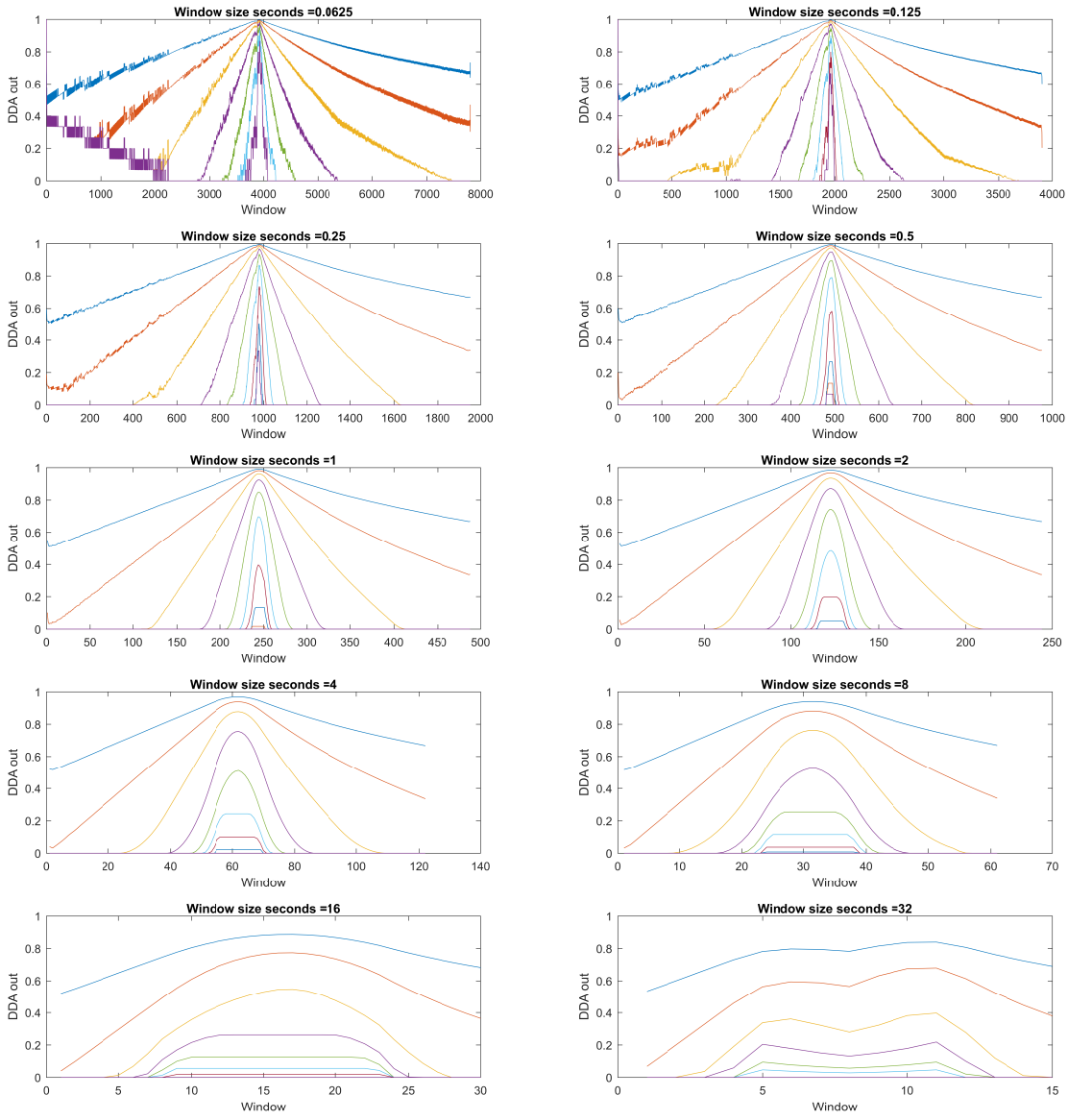

Fig. 2.6 Example of the effects of different window sizes on the algorithm

Table 2.1 Records used in the verification of the phase synchronisation processor

Id	Patient	Dominant pattern	Electrodes [a]	Sync. Pairs	Sample Rate	Seizures	Hours
#1	male 22y	rhythmic sharp waves	1/114/0/0	SCL7 - {SCL8, SCR9, TPL1}	1024 Hz	26	113.20
#2	female 29y	rhythmic beta waves	0/121/0/0	HAR1 - {HPR12, HPR1, HPR2}	1024 Hz	9	183.10
#3	female 16y	low amplitude fast activity	21/0/12/48	IHB2 - {IHA1, IHA2, GB6}	256 Hz	9	229.30
#4	male 18y	repetitive spiking	21/20/46/32	BLB1 - {BLB3, TRB1, FP1}	1024 Hz	13	245.20
#5	male 35y	rhythmic beta waves	21/10/46/0	TBLB1 - {TBLB3, TBRB1, FP1}	256 Hz	26	180.00
#6	female 53y	rhythmic beta waves	19/25/16/0	HL2 - {HL1, HL4, TBB1}	256-512 Hz	6	164.40
#7	male 5y	low amplitude fast activity	21/20/50/0	TBA1 - {TBA2, HRA4, HRA5}	1024 Hz	22	170.60
#8	male 15y	rhythmic beta waves	51/72/52/58	GG3 - {TL2, GG4, GH2}	256 Hz	19	199.8
#9	female 32y	rhythmic beta waves	36/34/35/37	HL4 - {HL2, HL3, HL5}	1024 Hz	9	162.6
#10	female 11y	rhythmic alpha waves	60/57/59/4	HR12 - {HR9, HR11, TBA4}	1024 Hz	14	155.0

[a] (S/D/M/G) for surface, depth, strip and grid electrodes.

contains recordings of multiple patients including 225 scalp recordings, 50 intracranial recordings and over 100 annotated sets using both intracranial and surface electrodes which were organised into a standard 10-20 format.

In this work, simulation experiments including the DDA and the PLV with 10 recording sets included in the European Epilepsy database have been conducted. In total, 1803.2 hours of recordings and 61 annotated seizures have been analysed. Simulations were carried out in Matlab using mathematical models of the DDA and the PLV processors. In the DDA case, the model closely follows the block diagram of Fig. 3.1 for $N = 1024$, $r = 1$, $M = 10$ and $Q = 2$ (see chapter 3, subsection 3.2.1 for definition of M and Q) and the Matlab code can be seen in Appendix B.1 while, in the PLV case, the model uses the architecture presented in [50]. In both cases, an input signal resolution of 10-b has been assumed.

Table 2.1 shows the records used for the verification of the proposed phase synchronisation calculation approach. For each patient the table shows the electrode configuration (both EEG and invasive), sampling rate, number of seizures, recording duration and dominant seizure pattern. The EEG data were acquired using a Neurofile NT digital video EEG system, while the invasive measurements were obtained with depth, strip or grid electrodes from Ad-Tech. For each patient, three different phase synchronisation indexes, involving four intracranial electrodes, were calculated. In all cases, the three contact pairs have one electrode in common, located in the proximity of the epileptic focus, while the other three electrodes, not necessarily from the same array, were used as reference. These electrodes are identified in Table 2.1 as #F-{#R1, #R2, #R3}, where #F denotes the electrode at the seizure focus and #Rx denote the references. Prior to the calculation of the synchronisation indexes, records were

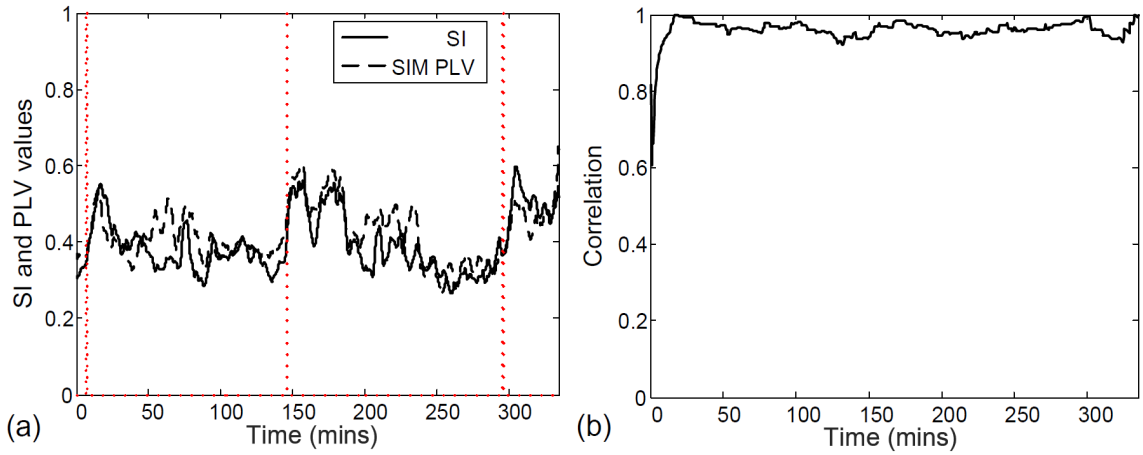

Fig. 2.7 (a) Simulated DDA algorithm index and simulated PLV value from recordings at positions M5 and M8 of a 1×8 EcoG strip implanted in patient FR_1084. Signals were pre-filtered in the β-band. The input rate of the chip was adjusted to the sample rate of the recordings, i.e., 1024S/s. (b) Correlation between both plots.

band-pass filtered in the frequency bands where higher activities were observed. This gave the selection of β-band for all patients but patient #4, for which the α-band was most insightful. In the case of patient #10, the β- and θ bands offered similar results.

2.3.1 DDA-PLV correlation

For illustration purposes, 2 examples of the correlation results between the DDA SI index and the PLV can be seen in Figs. 2.7 and 2.8, respectively, the synchronisation results obtained from two recording blocks measured in patient FR_1084 (48 year old female). In both blocks (6h long) three low amplitude fast activity (lafa) seizures were annotated (red dotted lines). For each figure, the plots on the left show the simulated synchronisation index (labeled as "SI") and the calculated PLV value (labeled as "SIM PLV"), while the plots on the right represent the correlation between both measures calculated at the time instants in which the experimental synchronisation indexes were recorded. The DDA algorithm detects all major changes in synchrony and indeed follows a similar trend as the PLV algorithm. This is verified by the high correlation results of approximately 90% on average.

Figs. 2.7 and 2.8 also show that seizures manifest with a remarkable increase in the synchronisation measures. This feature is in fact on the basis of the detection

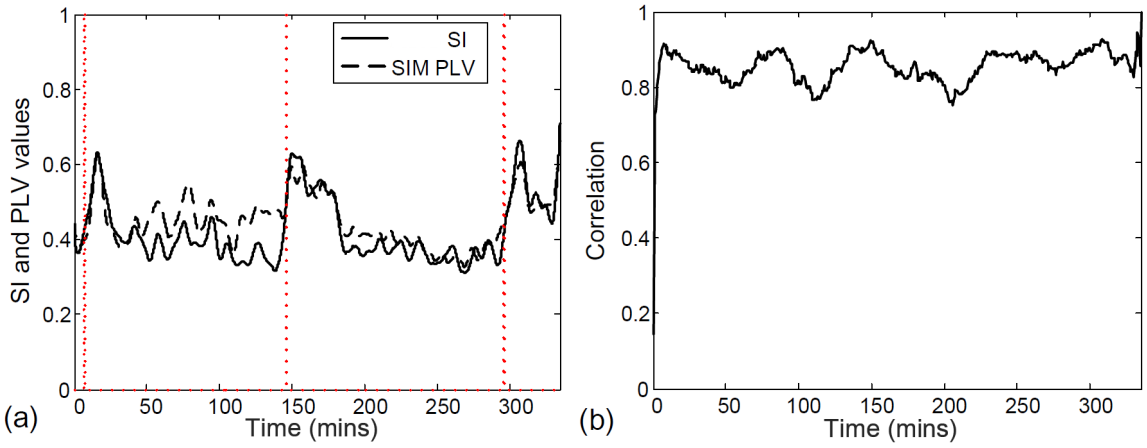

Fig. 2.8 (a) Similar as for Fig. 2.7 using the electrodes GA2 and GA5 of a 8×8 subdural grid implanted in patient FR_1084. (b) Correlation between both plots.

mechanism presented in [52] and [50] in which seizures are detected by applying thresholds on the PLV indicators calculated between pairs of channels.

2.3.2 SNR

The simplicity of the DDA circuit comes at the expense of higher susceptibility to noise on the recorded signals as compared to the PLV approach. This is illustrated in Fig. 2.9 which shows the correlation between the ideal PLV values obtained with two noise-free 60 minutes long neural signals, and the synchronisation indexes, derived from the DDA approach, when the same signals are contaminated with white Gaussian noise. The correlation is plotted against the spot Signal-to-Noise Ratio (SNR), assuming 1Hz bandwidth, at the maximum frequency of the β band, i.e. 30Hz. PS estimations are derived on that frequency band as well. Spot $SNRs$ in the range between 10dB and 70dB at 5dB steps have been considered and, for each SNR, a Montecarlo analysis (100 instances) has been carried out. The error bars show both the mean value and the $\pm 3\sigma$ of the synchronisation indexes. It is worth observing that for spot $SNRs$ above 30dB the average correlation between the DDA approach and the noise-free PLV technique is larger than 90%. Also note that for large $SNRs$, i.e. when the noise contamination is negligible, the correlation is in the order of 95%. This "intrinsic" deviation can be attributed to the structural differences between the DDA and the PLV algorithms. Fig. 2.9 also shows the correlation between the noise-free PLV values and those obtained with a 16-tap Hilbert transformer with passband ripple of

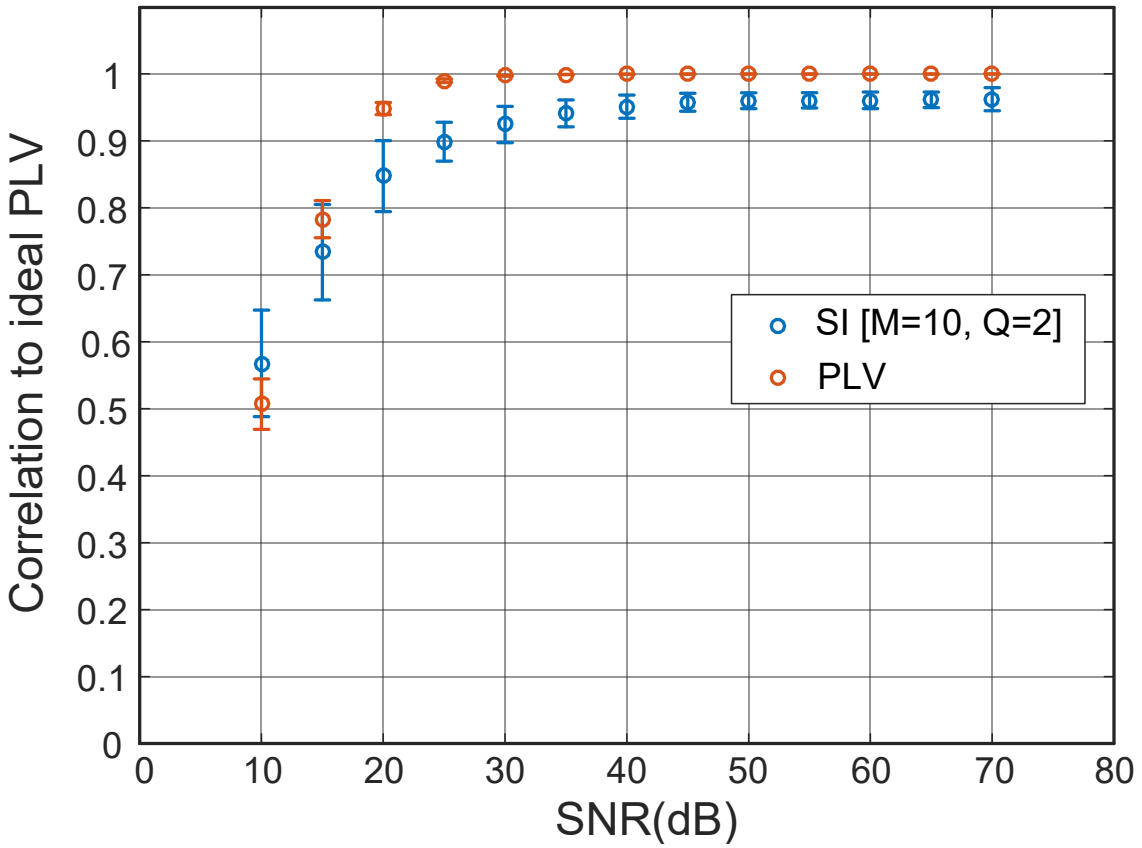

Fig. 2.9 Correlation of the DDA synchronisation index and the PLV processor in [52] with an ideal noise-free PLV calculation when the input signals are contaminated by Gaussian noise.

0.01dB when input signals are corrupted by noise, following the implementation in [52]. Similar as for the DDA case, performance worsens for $SNRs$ below 20dB.

To assess the impact of the noise behaviour shown in Fig. 2.9, it must be emphasised that surface and intracranial EEG signals exhibit $1/f^x$-like power spectra at low frequencies [58, 59]. This makes low frequency bands more tolerant to the aggregated noise contributed by biological tissue, electrodes and recording front-ends [60, 61]. Indeed, it has been shown that even with modest amplifier noise specifications, without suppressing flicker contributions, SNRs larger than 30dB can be obtained at low frequencies [62]. This suggests that PS calculations can be relevant as a neurological biomarker in the β and lower frequency bands although the reliability decreases for the high-γ band and higher frequencies. This observation is indeed confirmed in Section 2.3.1, where it is also demonstrated that DDA deviations compared to PLV can be tolerated without sensibly degrading the quality of the results while the DDA approach

offers light and modular hardware implementations, easily scalable to large neural recording arrays.

2.3.3 Sensitivity

In order to estimate the sensitivity (number of seizures which are correctly detected divided by the total number of real positive cases) of the DDA and the PLV approaches in terms of the False Positive Rate, FPR (number of wrong positive detections divided by the total number of actual negative events), different thresholds per contact pair and patient have been applied over the corresponding synchronisation indexes. Thresholds for the DDA and the PLV algorithms do not necessarily coincide even if applied on the same electrode pair. Thresholds are obtained by scanning the synchronisation results and identifying the index value leading to a specific FPR performance. To identify each of the 5 thresholds for each individual contact pair and patient the following procedure is enforced. Firstly, by identifying the total length of the resulting SI and PLV values (the length is in hours and does not include the pre, post and ictal periods). Secondly, by multiplying the total hours by the desired FPR we can extract the exact amount of false positives allowed during the resulting SI and PLV values. Since we are looking for peaks (false positives), we then search the SI and PLV values for the top x peaks in descending order. From there, using the last peak from the x peaks selected we set a threshold. The Matlab code for the thresholding can be seen in Appendix B.2.

In our experiment, a detection is produced whenever *one out of three* synchronisation indexes (or PLV values) rises above their corresponding thresholds. Hence, decisions are taken based on the outcomes from three contact pairs, not just one. Additionally, the following considerations have been adopted [63]. An observation window of 15 min before the annotated seizure has been defined for the detection of true positives. Hence, if a crossing is detected any time within this window, which can be associated to a preictal state, a true seizure detection is accounted for. Further, in order to exclude effects from postictal states, recording periods within 30min after the onset of a seizure were discarded for threshold calculations. If two successive seizures are separated by less than 45min, the preictal window before the onset of the following seizure is preserved in the analysis. Fig. 2.11 illustrates a true positive detection identified in patient #7. In this case, the synchronisation indexes calculated with DDA from the three contact pairs pass their respective thresholds during the preictal period.

Fig. 2.10, shows an example of the full SI index results obtained from patient #7. The 5 horizontal red lines indicate the different thresholds for each FPR and the vertical red boxes indicate the pre/post and ictal periods. All though the results can be somewhat noisy, it is apparent that there do exist very Strong prominent peaks in synchronisation.

Fig. 2.12 shows the sensitivity variation with respect to FPR, averaged over all the 10 patients in Table 2.1, according to the *one out of three* detection rule described before. Five different FPR values, in units of number of events per hour, have been considered. Both DDA and PLV detectors were analysed. Error bars indicate the 95% confidence interval limits per method and FPR value. The average detection threshold variation along the FPR range (from 0.15 to 0.75 FP/hr) was 8.8% for DDA and 14.5% for PLV. Table 2.2, shows all of the true positives, false negatives and sensitivity results which where used for the sensitivity analysis.

As can be seen in Fig. 2.12, there are little differences between the two approaches (only 4.6% sensitivity variation averaged over all FPR), with DDA slightly outperforming PLV detection, excepting at 0.45 FP/hr for which both methods obtain similar results. As the constraint is relaxed in terms of FPR, it is observed that the sensitivity rises and the confidence levels get tighter. In fact, at 0.75 FP/hr, DDA achieves an 84% sensitivity whilst its upper confidence level reaches as much as 92%. These results are comparable to those in [50].

Although results can be hardly generalised, it was observed in the analysis that certain seizure patterns tend to offer better sensitivities than others. In particular, those patients showing low-amplitude, fast activity or rhythmic β waves, obtained some 5% better sensitivity in β band. This was observed both for DDA and PLV PS detection.

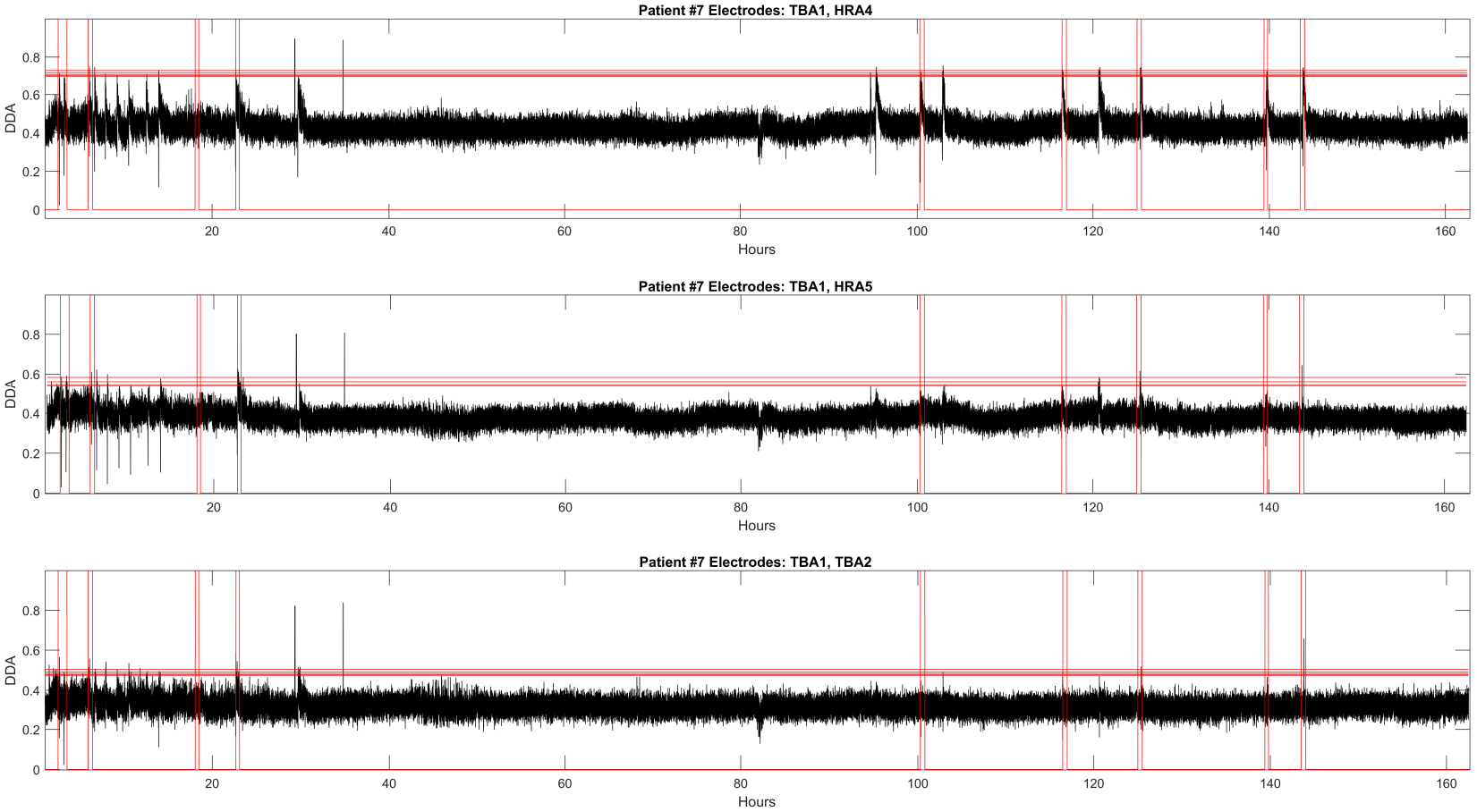

Fig. 2.10 patient #7, SI results from sensitivity test. The horizontal red lines represent the threshold levels and the vertical red boxes represent the pre/post and ictal periods.

Table 2.2 DDA and PLV results showing the number of true positives (TP), false negatives (FN) and sensitivity for each of the 10 patients shown above. The FPR rate ranges from 0.15 through to 0.75

Patient	Metric			DDA					PLV		
	FPR	0.15	0.3	0.45	0.6	0.75	0.15	0.3	0.45	0.6	0.75
	TP	4	5	5	6	6	4	4	5	5	5
#1	FN	3	2	2	1	1	3	3	2	2	2
	Sensitivity	0.57	0.71	0.71	0.86	0.86	0.57	0.57	0.71	0.71	0.71
	TP	7	8	9	9	10	9	10	10	10	10
#2	FN	3	2	1	1	0	1	0	0	0	0
	Sensitivity	0.70	0.80	0.90	0.90	1.00	0.90	1.00	1.00	1.00	1.00
	TP	2	3	3	3	3	3	3	3	3	3
#3	FN	1	0	0	0	0	0	0	0	0	0
	Sensitivity	0.67	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	TP	2	2	2	2	2	2	2	2	2	2
#4	FN	1	1	1	1	1	1	1	1	1	1
	Sensitivity	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.67
	TP	1	1	1	1	1	1	1	1	1	1
#5	FN	0	0	0	0	0	0	0	0	0	0
	Sensitivity	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	TP	4	4	4	4	4	3	3	3	3	3
#6	FN	0	0	0	0	0	1	1	1	1	1
	Sensitivity	1.00	1.00	1.00	1.00	1.00	0.75	0.75	0.75	0.75	0.75
	TP	1	6	6	7	10	0	3	7	8	10
#7	FN	11	6	6	5	2	12	9	5	4	2
	Sensitivity	0.08	0.50	0.50	0.58	0.83	0.00	0.25	0.58	0.67	0.83
	TP	3	3	3	3	3	2	3	3	3	3
#8	FN	4	4	4	4	4	5	4	4	4	4
	Sensitivity	0.43	0.43	0.43	0.43	0.43	0.29	0.43	0.43	0.43	0.43
	TP	3	3	3	3	4	1	3	3	3	4
#9	FN	1	1	1	1	0	3	1	1	1	0
	Sensitivity	0.75	0.75	0.75	0.75	1.00	0.25	0.75	0.75	0.75	1.00
	TP	5	5	6	7	7	1	6	7	7	7
#10	FN	5	5	4	3	3	9	4	3	3	3
	Sensitivity	0.50	0.50	0.60	0.70	0.70	0.10	0.60	0.70	0.70	0.70

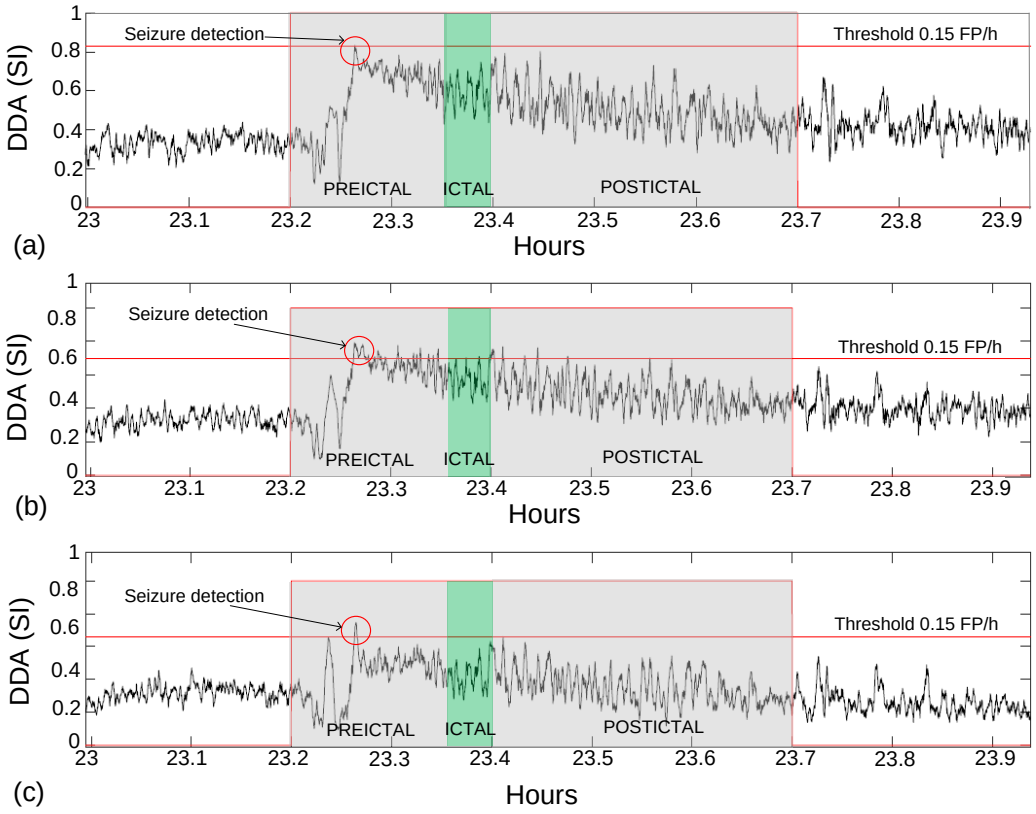

Fig. 2.11 Zoom synchronisation indexes for patient #7 during a clinically annotated seizure and pre/post ictal padding at a false positive rate of 0.15 per hour. (a) Pair TBA1-HRA4, (b) pair TBA1-HRA5, (c) pair TBA1-TBA2.

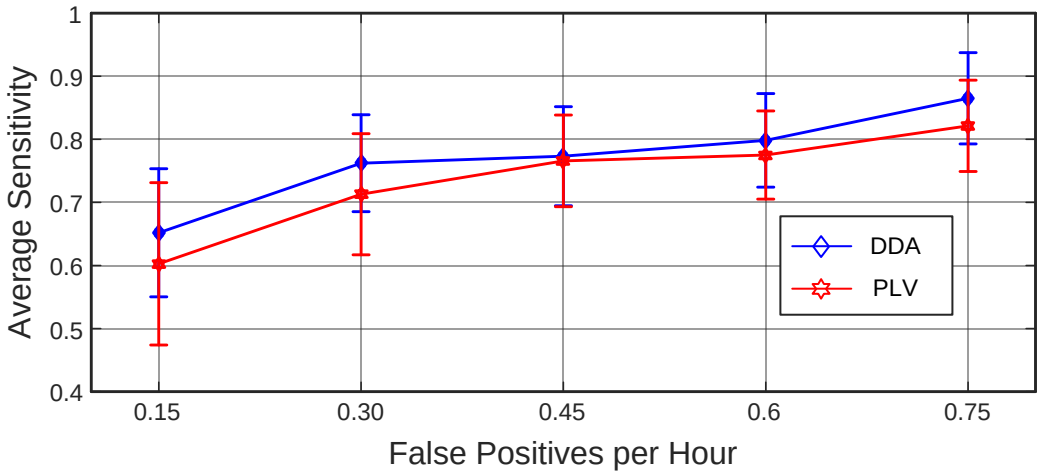

Fig. 2.12 Average Sensitivity against FPR for DDA + SI and PLV + HT including 95% confidence analysis obtained via 1803.2 hours of recordings and 61 annotated seizures.

2.4 Conclusions

In this chapter, a hardware friendly algorithm for the detection of phase synchronisation has been introduced. The algorithm achieves a very high absolute sensitivity of approximately 80% for an FPR of 0.75 FP/h. Indeed, it produces an absolute sensitivity of close to 90% at 0.75 FP/h at the upper bound of the 95% confidence levels. The effect of noise has also been assesed which shows for SNR values greater than 30dB the algorithm performs almost perfectly with over 90% correlation to an ideal PLV. The correlation between other proven methods of phase detection has been assessed giving on average a 90% correlation to the PLV.

Chapter 3

Synchronisation Processor

The contributions in this chapter includes the design, implementation and verification of a sub-threshold phase synchronisation processor capable of calculating 'phase synchronisation between two neural signals. The processors design and verification process was firstly implemented in an FPGA environment and RTL compiler, later the design was implemented manually in the CADENCE layout environment.

All of the design and verification processes were carried out by the author of this thesis.

3.1 Overview

The synchronisation processor is based on the measurement of time periods between two consecutive minima as described in section 2. The simplicity of the approach allows for the use of elementary digital blocks, such as registers, counters or adders. In fact, the processor, fabricated in a 0.18μm CMOS process, only occupies 0.05mm^2 and consumes 15nW from a 0.5V supply voltage at a signal input rate of 1024S/s. These low-area and low-power features make the proposed processor a valuable computing element in closed-loop neural prosthesis for the treatment of neural disorders, such as epilepsy, or for measuring functional connectivity maps between different recording sites in the brain.

The ASIC uses circuit elements from a full-custom digital library for 0.5V power supply operation, based on low-leakage transistors in weak inversion. The library blocks were generally designed using conventional CMOS logic, with device dimensions close to minimum size, as they were found to offer an acceptable trade-off between power dissipation and operating frequency [64]. Indeed, the library was exhaustively verified and characterised under PVT deviations assuming a maximum clock frequency of 25kHz, i.e., about twice the operation frequency needed by the I/O ports of the ASIC. In spite of the reduced noise margin and the slow switching transitions, post-layout simulations of the chip showed complete functionality with no data loss for 0.5V operation.

The implemented ASIC is competitive when compared to currently available FDA approved devices and other state-of-the-art devices such as described in chapter 1.6. The small size and power consumption allows for increased scalability, allowing for more EEG data streams to be incorporated, leading to an increase in sensitivity

when compared to more complex algorithmic structures such as the HT and PLV combination.

Based on the algorithmic design from chapter 2, An overview of the system can be seen in Fig. 3.1. The design consists of 5 main layers. Firstly the minimum detection layer, which is used for the identification of the quasi-periodic local minima as denoted by t_n from equation 2.1, in each given neural signal denoted s1 and s2 for neural signal 1 and neural signal 2, respectively. The Timestamp layer is the practical identification of the phase/frequency increase between two consecutive local minima within a given signal i.e T_n^s from equation 2.1. The third and most complex layer is the calculation layer, which is responsible for calculating the difference in samples between consecutive minima in the two signals, such that we get the relative error $\triangle T_n = |T_i^1 - T_j^2|$. The calculation block is also responsible for the accumulation of these errors such that $Sdda(k) = \sum_{n=0}^{K-1} \triangle T_n$. The filtering layer is used for the smoothing of $Sdda(k)$, leading to more interoperable data and easier to set thresholds at the output of the device. The final, detect layer, is a test platform for setting epileptic alarm thresholds.

3.2 Design

The following section describes each of the 5 layers in-depth including examples. Appendix C.1, provides a pseudo code of the hardware flow for the design.

3.2.1 Minimum detection

From Fig. 3.1, The minimum detection architecture consists of two identical subsystems, one for neural data stream $S1$ and the second for neural data stream $S2$.

In order to reduce hardware utilisation and minimise power consumption, the minimum detection scheme introduces a simple yet robust toggling architecture.

To begin, each subsystem is comprised of a 10-b most significant bit comparator which, compares two samples, $S(i)$ and $S(i-1)$, corresponding to the signals current sample and previous sample respectively. Where $S(i-1)$ is delayed 1 clock cycle via a 10-b register. If $S(i)$ is smaller than $S(i-1)$, then, a logic 0 is produced such that the result from the comparison is, $C = 0$. Contrarily, if the opposite is true then a logic 1 is produced such that $C = 1$. In the final case that $S(i) = S(i-1)$, then $C = C$, i.e there is no change in the output. Nevertheless, the final scenario is unlikely to occur

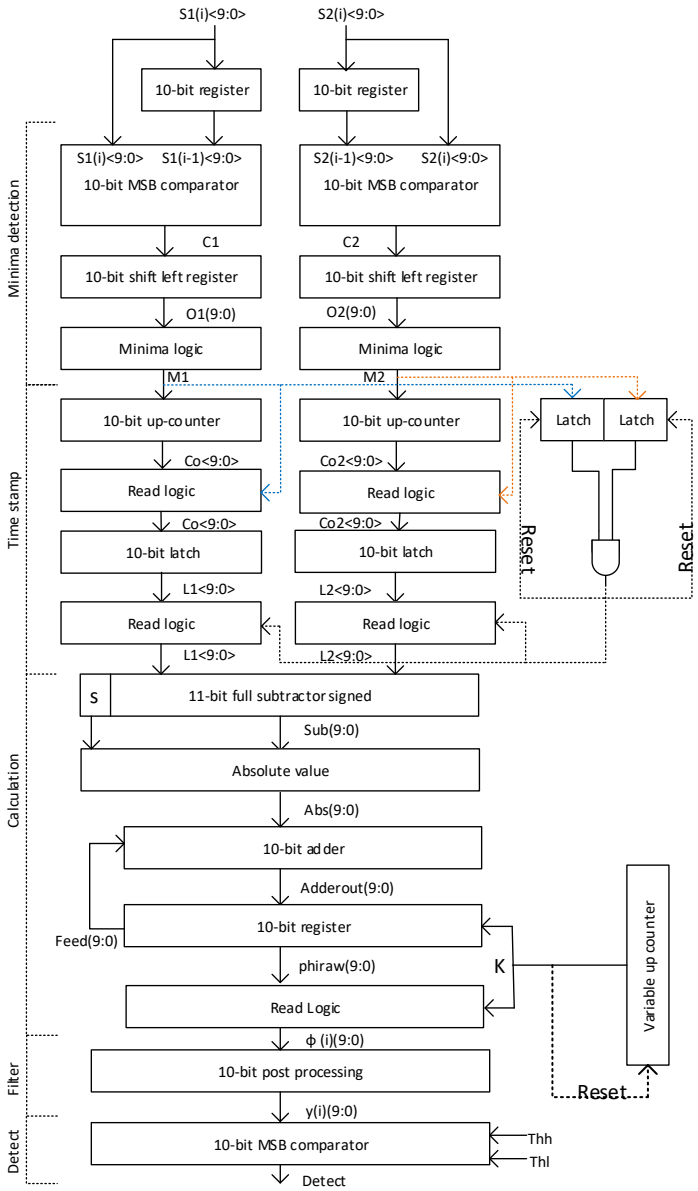

Fig. 3.1 Synchronisation processor: Block diagram of the main blocks used for the design of the synchronisation processor. This design consists of 5 main layers and incorporates a non complex architecture structure

if the sampling rate of the system is sparse enough. To reduce hardware utilisation and power consumption, the MSB comparator first compares the 5 most significant bits of $S(i)$ and $S(i-1)$. If a solution is found then the result C is produced else the comparator moves on to the lower 5 bits for comparison. In general C is given as:

$$C = \begin{cases} 0 & \text{for} \quad S(i) < S(i-1) \\ 1 & \text{for} \quad S(i) > S(i-1) \\ C & \text{for} \quad S(i) = S(i-1) \end{cases}$$

The value of C gives an indication as to whether, at the current sample in time, the signal is traversing in a downwards fashion or in an upwards fashion. Additionally, to achieve a more robust system, a 10-b M shift left register was implemented which records the signals history over the past 10 values of C. The result from this register O, is outputted every clock cycle with the newly updated word.

In the case that the shift left registers output is $O = 0000011111$, it becomes evident that at some point during the signals history, it moved in a downward trend followed by an upward trend. This indicates that at points $O(6) = 0$ and $O(5) = 1$, there was a local minimum t_n. Fig. 3.2, shows this concept.

This simple architecture is hardware friendly, nevertheless, as described in section 2 subsection 2.3.2 the noisy nature of neural data and the possibility of random fluctuations can produce false positives in detection out the output.

Therefore, to increase robustness of the minimum detection system, bubble detection logic, as seen in Fig 3.2, is used. This small piece of logic relaxes the strict rule of O, by allowing for one outlier (Q) either side of the values $O(6) = 0$ and $O(5) = 1$, such that a detection $M1 = 1$, of a local minimum is confirmed if any single value in $O(9:6) = 1$ or any single value in $O(4:0) = 0$. However, the values $O(6) = 0$ and $O(5) = 1$ must be true in any of the cases.

3.2.2 Time stamp

The time stamp layer relies heavily on a single 10-b up counter. Whose, main purpose is to count upwards with time between two consecutive minima within a signal, hence discovering T_n^s. This gives a good understanding of the instantaneous frequency of the signal at that given point in time.

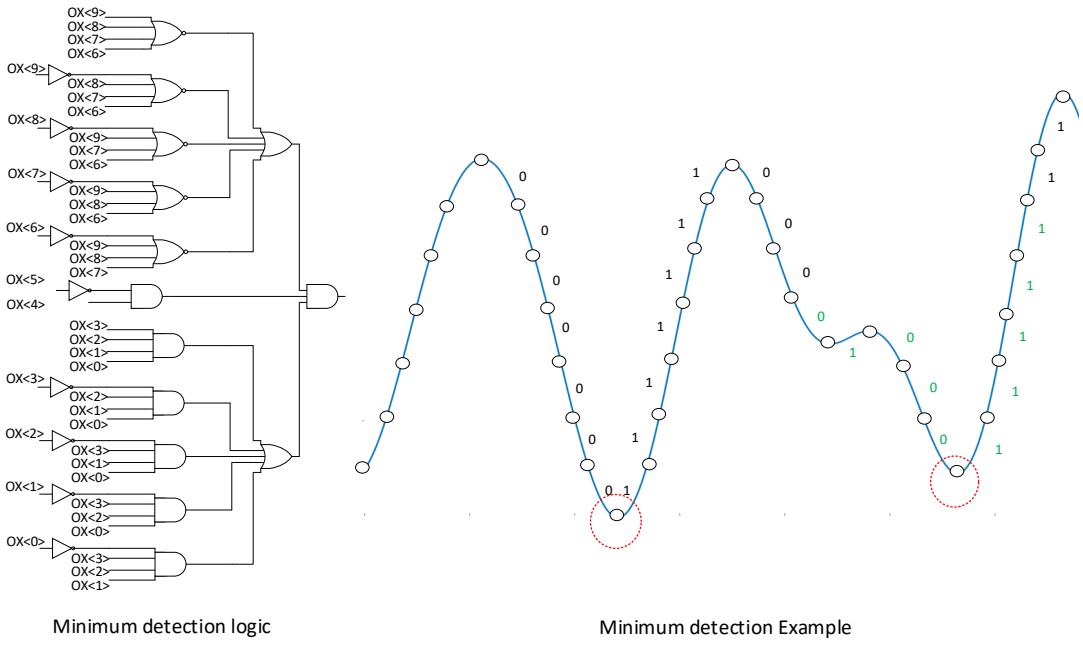

Fig. 3.2 Minima detection logic (left) and example of bubble detector(right). The example shows an example of a real EEG data stream pre filtered into the beta band. Here we can note that the first minimum in the signal is detected as $O = 0000011111$. As for the second minimum it is also detected due to the relaxed outlier condition such that $O = 0100011111$

In the case that a minimum is detected, M1 activates a 10-b array of reading logic gates which are implemented as standard 2 input AND gates and the current value in the up counter $Co(9:0)$ is passed into a 10-b latch and stored. simultaneously, since a new minimum has been detected the counter is reset and it once again begins its count towards the next minimum in the signal.

3.2.3 Calculation

The architecture of the calculation layer starts with a signed 10-b subtracter with one-bit overflow. This is used to calculate the difference $\triangle T_n = |T_i^1 - T_j^2|$, between the two signals detected minima. The subtracter runs continuously subtracting values every cycle. However, since the algorithm simplifies the calculations to a minima detection basis only, rather than a sample by sample basis, a small piece of extra logic is needed.

This logic comes in the form of two 1 bit latches and a logic AND gate. Once T_n^1 has been detected in $S1$ the logic value of $M1$ gets latched. This indicates that $S1$, is ready for the calculation stage, Similarly, $M2$, is latched if T_n^2 is detected in $S2$.

If $M1 = M2 = 1$, then a signal is sent to read logic block situated at the output of the previous latching stage. This allows the values from both of the counter modules to be subtracted. Simultaneously the small piece of logic resets the latches holding $M1$ and $M2$ such that the output is now of logic state 0. In terms of the read logic if they are inactive then the output of the read logic is a constant logic 0, for all 10-b.

Therefore, although the subtracter is in continuous operation, when no relevant data is ready for calculation it will simply subtract $0 - 0$. Since non-switching is necessary for this specific case, the dynamic power consumption of the subtracter during non-crucial data is almost null and only consumes static power.

The initial subtraction can create two different outputs at $Sub(9:0)$, firstly, if $L1(9:0) > L2(9:0)$, then the result will be a positive unsigned 10-b result. However, in the reverse case that $L(9:0) < L2(9:0)$, a 2's complement negative result is produced. In this case, the MSB (the eleventh bit of the subtracter(S)), is set to 1 indicating the result is negative. Since the algorithm desires $|\triangle T_n|$, the 2's complement result needs converting back to standard unsigned notation. To achieve this the design incorporates an array of 10 bit XOR gates which perform the logic conversion to 1's complement and a full adder adds 1 to $Sub(9:0)$ producing $Abs(9:0)$.

The next stage requires the accumulation of errors for the value of $Sdda(k)$, This is achieved using a 10-b add and accumulate block, where the window size K, consists of a variable up counter with a maximum size of 2048 samples. When the window counter is less than its maximum value. $Adderout(9:0)$, consistently calculates $Abs(9:0) + feed(9:0)$, However, once again, since $Abs(9:0)$, is only non zero when relevant data is available the resulting sum is $0 + feed(9:0)$, which requires no switching.

Once the counter reaches its maximum value. A pulse is generated which resets the feedback register in the accumulator creating a zero-sum, at which point read logic at the output of the accumulator captures $Sdda(k) = \phi(i)$ and is passed to the post-processing stage for smoothing. This happens every K samples and avoids irrelevant data being passed into the next stage.

3.2.4 Filter

Due to fast fluctuations of $\phi(i)$ at the processors output, the detection of high and low values of $\phi(i)$, becomes troublesome. To counteract this problem an unconventional exponential filter was adopted, which is based on an autoregressive IIR filter. Mainly used for time series, the filter acts as a low pass filter and provides a substantial amount smoothing with very little cost in terms of hardware. the basic equation for the exponential filter is given by:

$$y(i) = \alpha \cdot \phi(i) + (1 - \alpha) \cdot y(i - 1) \tag{3.1}$$

Where, $y(i)$, is the new filtered value, $\phi(i)$, is the new incoming value (in this case new synchronisation value $Sdda(k)$), $y(i-1)$, is the previously filtered value, and α, is a weighting for the function, also known as a *smoothing factor*.

At this point, it is clear to see that this function produces an approximation of what the next value in the sequence should be, based solely on the current synchronisation value and the previously filtered value. This is why only currently relevant data can be passed into the filter, otherwise, its predictions will be erratic.

The *smoothing factor*, can be set such that $0 \leq \alpha \leq 1$, where a $\alpha = 1$ provides no smoothing, hence $y(i) = \phi(i)$, and $\alpha = 0$, provides maximum smoothing.

At first glance, the terms $\alpha \cdot \phi(i)$ and $(1-\alpha) \cdot y(i-1)$, may seem complicated leading to floating point multiplication. However, re-writing the formula we get the equation:

$$y(i) = y(i-1) - y(i-1) \cdot \alpha + \phi(i).\alpha \tag{3.2}$$

By choosing, α, such that it is always a power of 2, we can replace α, with 2^{pos} such that:

$$y(i) = y(i-1) - y(i-1) \cdot 2^{pos} + \phi(i).2^{pos} \tag{3.3}$$

Where *pos*, is an integer value. Since a bit shift right, results in a division, a drastic hardware reduction can be made. This approach limits our range of smoothing, nevertheless, for the generalised purpose of smoothing $\phi(i)$, it is sufficient. As an example, if **pos** is set to 1, then that would correspond to:

$$y(i) = y(i-1) - \frac{y(i-1)}{2} + \frac{\phi(i)}{2} \tag{3.4}$$

It is also important to note that since the filter relies heavily on its previous value, if $y(i)$, is initially set to one, a certain time frame must pass before stabilisation.

Fig.3.3, shows the basic architecture of this block. Firstly, a 10-b shift right register, which is externally adjustable to provide the smoothing factor and a 10-b full subtracter are used to calculate $y(i-1) - y(i-1) \cdot 2^{pos}$. Secondly, a 10-b shift right register is used to calculate $\phi(i).\alpha$, which is added to $y(i-1) - y(i-1) \cdot \alpha$, via a full adder to produce the final value of $y(i)$.

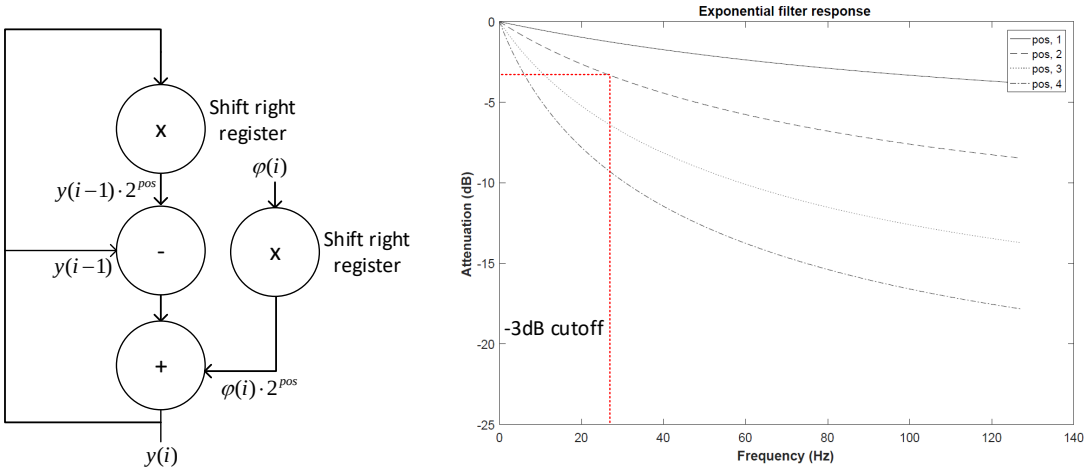

Fig. 3.3 Block diagram of the implemented exponential filter with variable smoothing capabilities (left). Filters magnitude response (right)

3.2.5 Detect

The detection block was designed for the detection of large changes in synchronisation. Namely, this would prove useful for such neurological disorders as epilepsy. In which large changes in synchronisation occur during different periods before, during and after the event. it is important, to stress the fact that since $Sdda(k)$, is now effectively equal to $\sum_{n=0}^{K-1} \triangle T_n$, then a large drop in synchronisation is equivalent to better synchrony, whilst the opposite is true increases in $Sdda(k)$. Therefore if $\phi(i) < Thh$, then an alarm is triggered indicating there has been a significant increase in synchrony, contrarily if $\phi(i) > Thl$, an active high alarm is triggered indicating a significant reduction in synchrony. This is performed by 2 10 bit comparators. Nonetheless, due to the inherent noisy output, 2 small counters of 4 bits are also placed in the detect block. These counters place the constraint that Thl and Thh, must be active for a set number of consecutive outputs. This diminishes the possibility of false positive detections. This block relies heavily on the previous filtering stage such that insufficient smoothing can increase the number of false detections. Thl, Thh and the set number of consecutive outputs are all injected into the device from an outside source and are not set dynamically within. This gives more control for testing.

3.3 Methods

The hardware implementation of the test was built and verified at several keys stages via various platforms.

3.3.1 Design flow

FPGA implementation and verification

The digital design was firstly realised in the VIVADO design environment using a VHDL. After successful simulation of the design within the environment, it was later synthesised and downloaded onto a XILINX ARTIX 7 AC701xc7a200tfbg676-2 evaluation board. For ease of testing, a hardware Co-Simulation of the DDA algorithm was ran through the MATLAB DSP SIMULINK environment along with a MATLAB based calculation of the HT and PLV index.

Fig.3.7, shows an overview of the top-level RTL compiled schematic from the VHDL code. This consists of 4 main blocks. Firstly, the synchronisation processor as described above (a view of the synchronisation processor blocks can be seen in Fig. 3.8).

Secondly, the SPI interface, which used to convert serial input streams to parallel data for the processor. The SPI has 3 main inputs 'datain1', 'datain2' and 'loadparameters'. The first two inputs convert neural signals 1 and 2, from a 1-b binary serial stream into a 10-b parallel unsigned binary sample. Since the processor requires 2x10-b unsigned parallel data at the input (1x10-b for each signal), the conversion of 'datain1' and 'datain2' are made via a 2x10 shift registers which operates 10 times faster than the core clock. Due to the 10 times increase in speed for the datain SPI interface the 'datain1' and 'datain2' samples from the SPI are latched at the output until 1 clock cycle of the main processor is complete. The final input 'loadparameters', allows for all of the static variables of 'K', 'Thh', 'Thl', '2^{pos}' and others, to be uploaded via a single binary stream. This allows us to reduce the total number of PADS used in the ASIC implementation, hence, reducing overall area consumption of the final design.

The third block, 'Check parameters', was used as a test strategy in order to verify that all of the uploaded variables were correct. This, block simply re-downloads the data sent onto the chip for verification.

Lastly, the PSI interface is the download link for the data generated by the synchronisation processor. Similar to the SPI interface this block is used to reduce area consumption of the device. Nevertheless, this block works in the opposite fashion converting parallel data streams into a single serial data stream. Assuming a clock frequency of approximatley 1024Hz operating frequency for the main processor this output PSI can run at the standard processor frequency.

Fig. 3.4, shows an example of how 'loadparameters' and the data stream 'datain1' are uploaded in a serial fashion and ultimately re-downloaded and checked via 'Parameter check' and 'checkdata1'.

Fig. 3.5, shows the minima detected by the system for signals S1 and S2. Here we can see that there exists a delay between the actual minimum and the detection pulse (the delay is indicated by the blue box). This delay exists because we need to gather 10 samples of history before an accurate decision can be made. Nevertheless, this delay in no way disturbs the systems ability to calculate phase as the exact same delay is present in both signals. Furthermore, we can see that no minima were detected during the section of signal encompassed in the red boxes. This is due to the outlier logic described previously which correctly dismisses these very small dips for minima.

Fig. 3.6, shows an example of the output $\phi(i)$ using a filtering of $\alpha = 1$ and two detection thresholds $Thh = 170$ and $Thl = 590$. During the simulation 1 drop in synchrony was detected (i.e when $\phi(i) > Thl$ 'dropdetected' = 1) and one increase in synchrony (i.e when $\phi(i) < Thh$ 'risedetected' = 1) (both of which are highlighted by blue boxes). There do exist false positive at the beginning of the output due to the time needed for the filter to build reliable data.

RTL compiler verification

In the second phase of the design, the VHDL code was implemented using the RTL compiler from CADENCE (An example of the RTL compiler script used to build the RTL schematic can be seen in Appendix B.4. This compiler maps all of the logic gates from the synthesised FPGA design to a specific technology in this case $0.18\mu m$. The RTL compiler delivers a complete schematic of the design at the register level. Once obtained the schematic is then uploaded to the CADENCE environment and simulated for error detection. If the design produces errors, certain attributes and optimisations can be enforced in the RTL compiler and re-simulated as per needed.

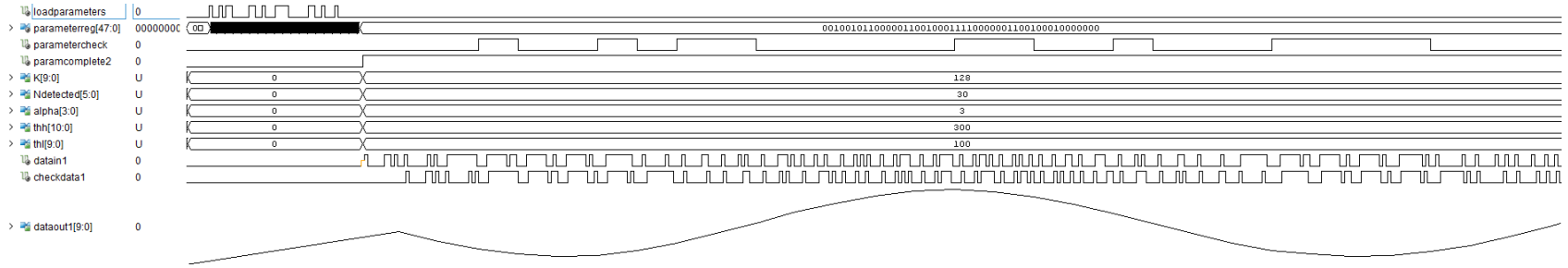

Fig. 3.4 VHDL simulation showing the uploading of parameters and data stream S1 along with the download verification 'Parameters check'

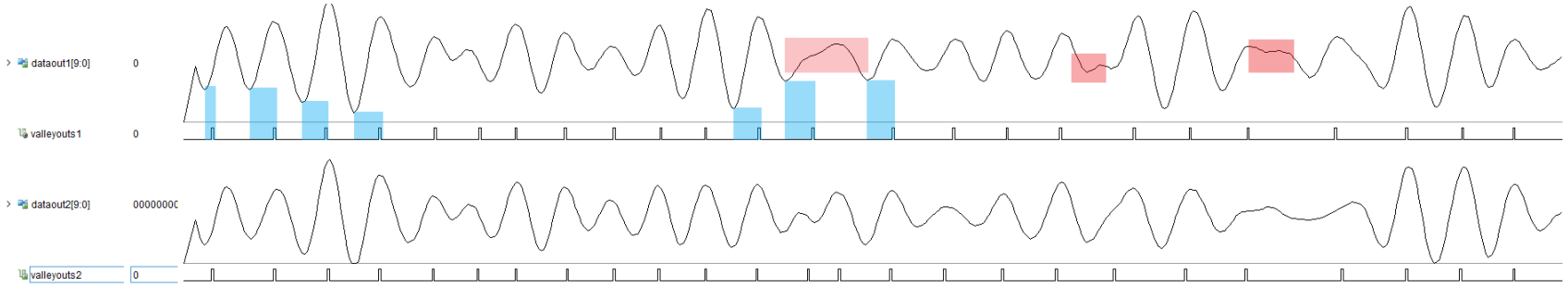

Fig. 3.5 VHDL simulation showing the minima detection of a small section of EEG recording for S1 and S2

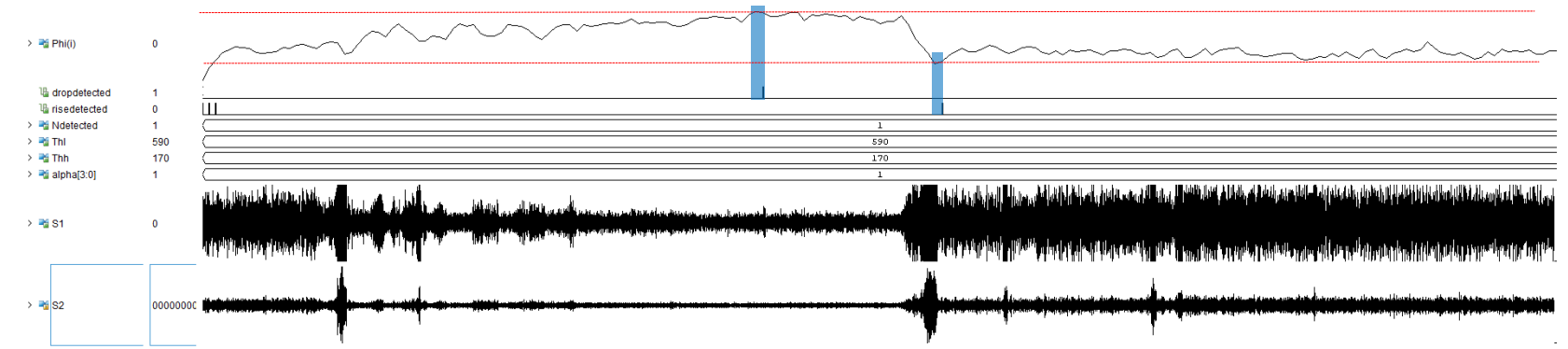

Fig. 3.6 VHDL simulation showing the results for phi for real EEG data and the detection

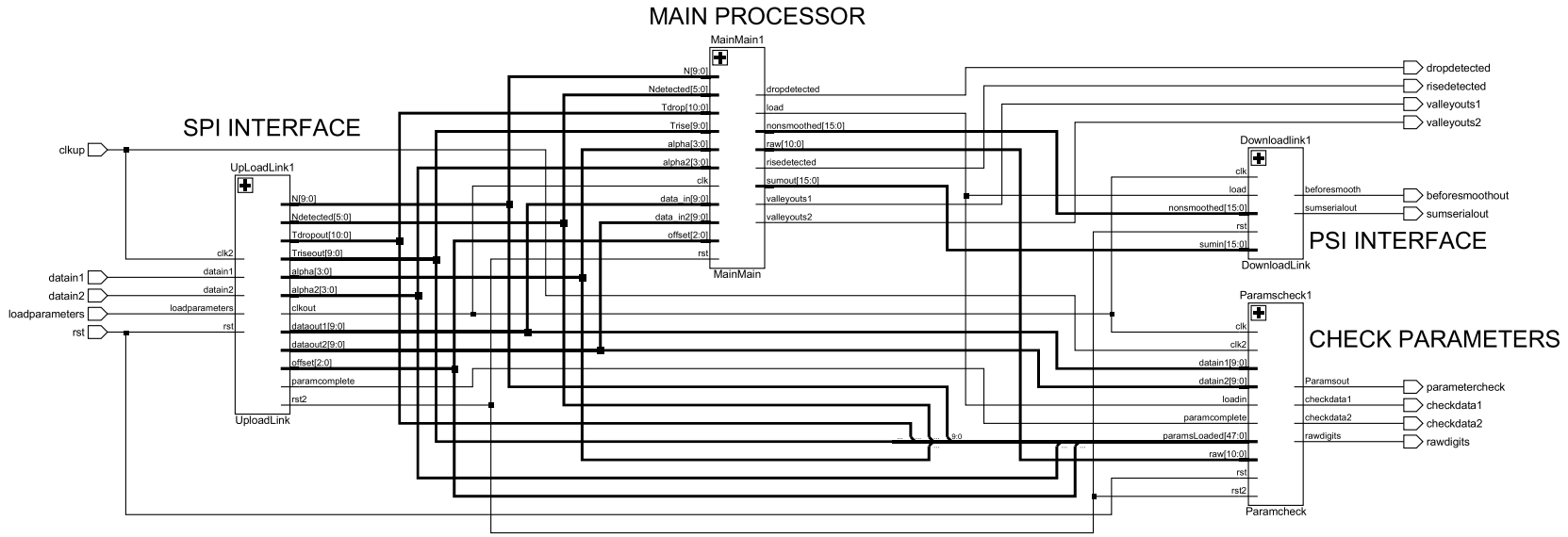

Fig. 3.7 Synthesised block RTL schematic of the prototype processor designed in VHDL. Where, the labled blocks correspond to figure 3.1.

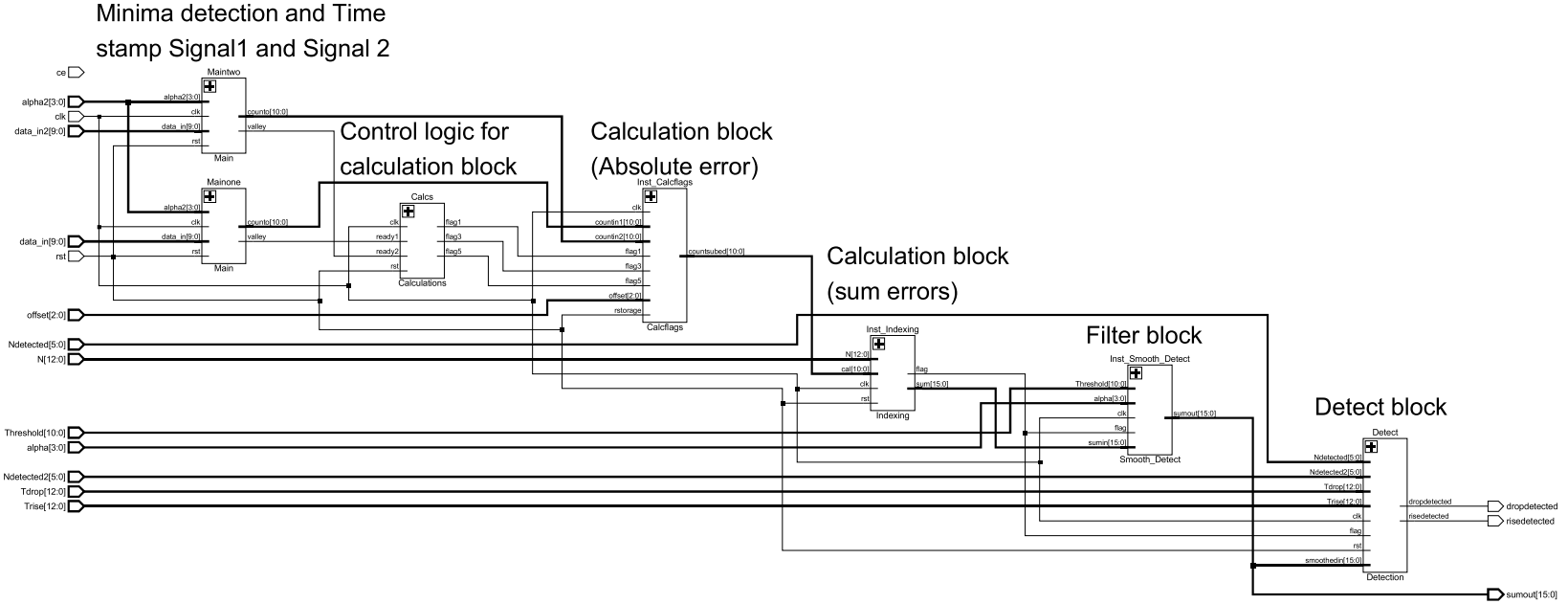

Fig. 3.8 Synthesised block RTL schematic of the prototype processor designed in VHDL. Where, the labled blocks correspond to figure 3.1.

Fig. 3.9 Layout view of the 2-channel 0.5V synchronisation processor from CADENCE

CADENCE layout

Fig 3.9, shows the circuit layout from the CADENCE design environment. Due to a sub-threshold operating voltage the design was implemented via a manual layout approach. The design was fully tested using extracted layouts including parasitic components.

3.4 Discussion

3.4.1 Indexing

The synchronisation index from equation 2.2 described in section 2, is a construct related to the comparative studies of the algorithm and is not actually needed to produce results of synchrony for a patient. In the previous sections the index was used mainly to emphasise the fact that the results obtained follow a similar pattern to that of more complex algorithms. Actually, the SI index is a form of scaling and does not change the results at the output. By omitting this from the hardware implementation. A drastic reduction in hardware can be achieved.

3.4.2 Minimum detection scheme

This overall detection scheme was chosen for several specific reasons, Firstly, due to the quasi-periodic nature and variations in the neural data, it is not guaranteed that a local minimum will have specific identifiers such as a zero crossing point. This means that it is imperative to follow the history of the signal as it moves forward in time.

Secondly, the toggling system was chosen to reduce hardware utilisation. In many systems which do not use a zero crossing identifier. There is a need to record the previous x samples and x future samples of the signal. This can lead to large memory allocations. The use of the outlier logic omits these memories.

The outlier logic in this design was set to a static 1 each side of the switching point. This was set to 1 as observations of many band limited signals there exist only small perturbations.

3.4.3 Filtering

Standard low pass filtering and smoothing such as the moving average filter use the averaging of multiple points in a signal to derive a single output. Defined as:

$$y(i) = \frac{1}{M} \sum_{j=0}^{M-1} \phi[i+j]$$

Where M, represents the total number of points to average, i.e, the window size for the filter. In single-sided averaging, when $j = 0$, the moving average filter sufferers from lag as a new result is only produced every M samples. In hardware, this would be implemented with an accumulator for the addition of samples and a shift right register if M is set to a power of 2. Nevertheless, for sensitive systems such as the detection of epilepsy, new and reliable data is needed fast to increase the reliability of detection.

Another alternative is that of the two sided moving average window, set such that $j = \frac{-(M-1)}{2}$ to $j = \frac{(M-1)}{2}$. This method reduces latencies, however increases hardware resources due to temporary storage of $j = \frac{-(M-1)}{2}$.

The exponential filter avoids the problems of lag and storage, by utilising only the current and previous sample, where more importance is placed on the new value.

We should also point out that although the filter is of an exponential origin, the 10-bit resolution is sufficient as the filter can never overflow, instead, it can only produce a maximum value of 1024. Such that: $y(i) = 1024 - \frac{1024}{x} + \frac{1024}{x} = 1024$

3.4.4 Power/Accuracy trade-off

The power consumption of the chip and its performance largely depends on the sampling frequency of the input signals. The slower the sampling rate, the lower the number of operations per second carried out by the chip and, hence, the lower the power consumption. However, the accuracy achievable on the calculation of transition periods decreases and the overall performance of the DDA algorithm degrades. On the other hand, for high sampling rates, the performance improves at the cost of higher power dissipation. This is illustrated in Fig. 3.10, which shows the power consumption of the chip in terms of the sampling frequency, assuming that both input signals are band-limited in the β band. In the same plot, the correlation between the measured synchronisation index and the PLV value (calculated off-chip with the same input signals) is represented as a measure of the DDA approach accuracy. It is worth noting that beyond a given sampling rate of about 2-3kS/s, no further improvement can be observed in the chip performance as the correlation states around 85%. Hence, a sampling rate of 2 kS/s is chosen for which the power consumption of the chip core is 15nW from a 0.5V supply voltage. Similar trade-offs can be observed for other frequency bands.

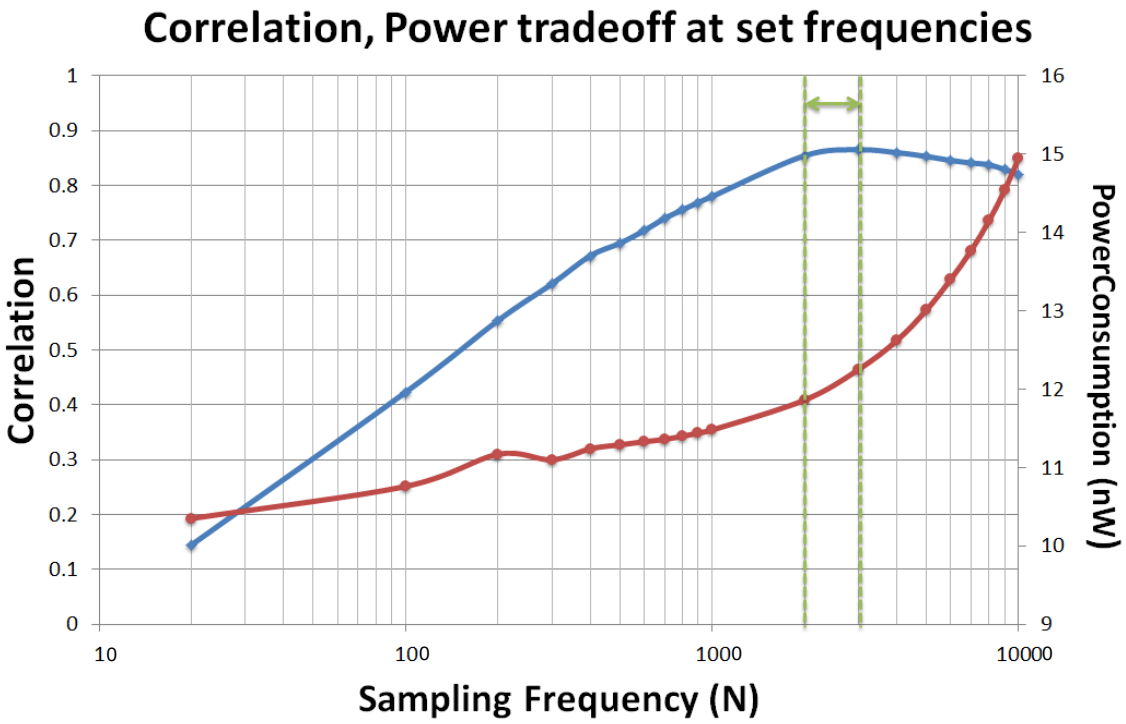

Fig. 3.10 Power/sampling frequency trade off overview

3.4.5 Resource allocation

Fig. 3.11, gives an impression of the hardware reductions which can be obtained by the DDA algorithm. the DDA algorithm can be calculated using as little as 37 slices, 112 flip-flops and 60 LUT(Look-Up Tables), excluding SPI interface, PSI interface and CHECK PARAMETERS. Which is significantly less than other systems, HT1[65] and HT2[66], which both utilize a Hilbert transform based filter and CORDIC arctan phase extraction system. The mean number of resources needed to calculate the discrete distances (DDA approach) is 69.6, which is almost 95% less than system 1 (1348) and almost 96.8% less than system 2 (2149).

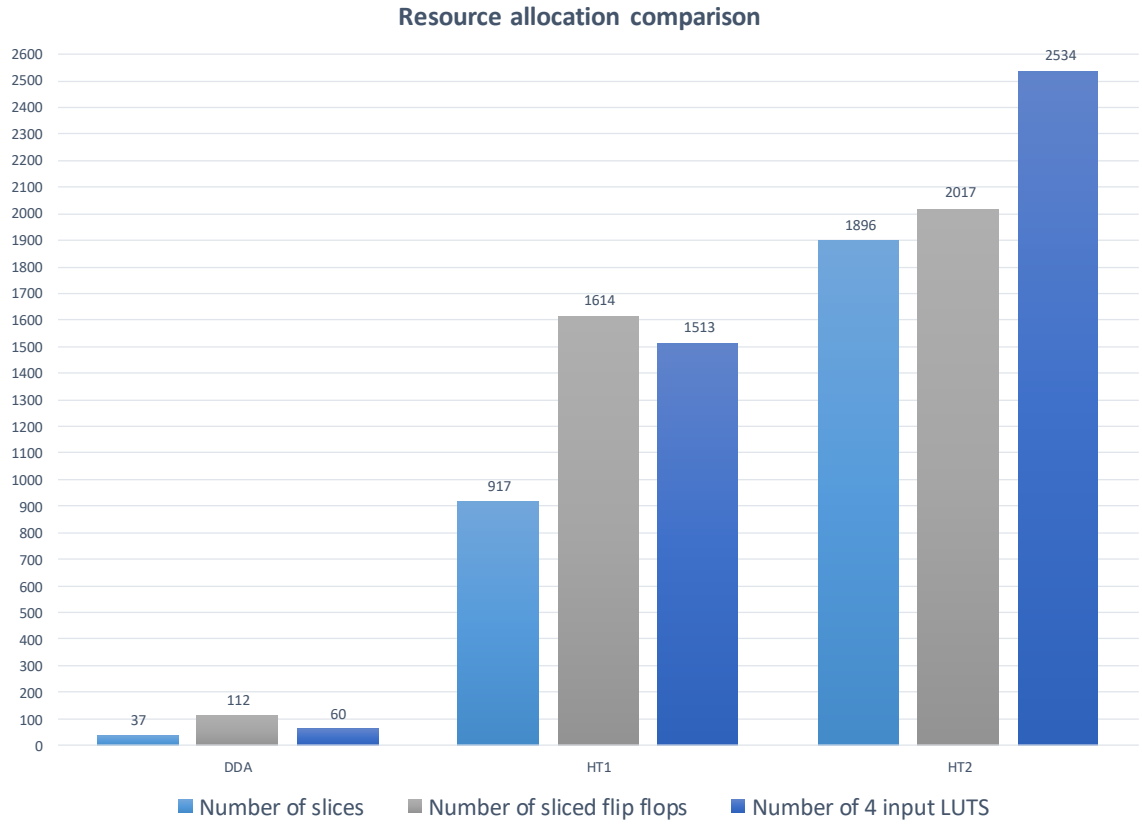

Fig. 3.11 Bar chart showing the FPGA resource allocation for the designed DDA algorithm and two other FPGA based implementations of the Hilbert transform and Cordic PLV values

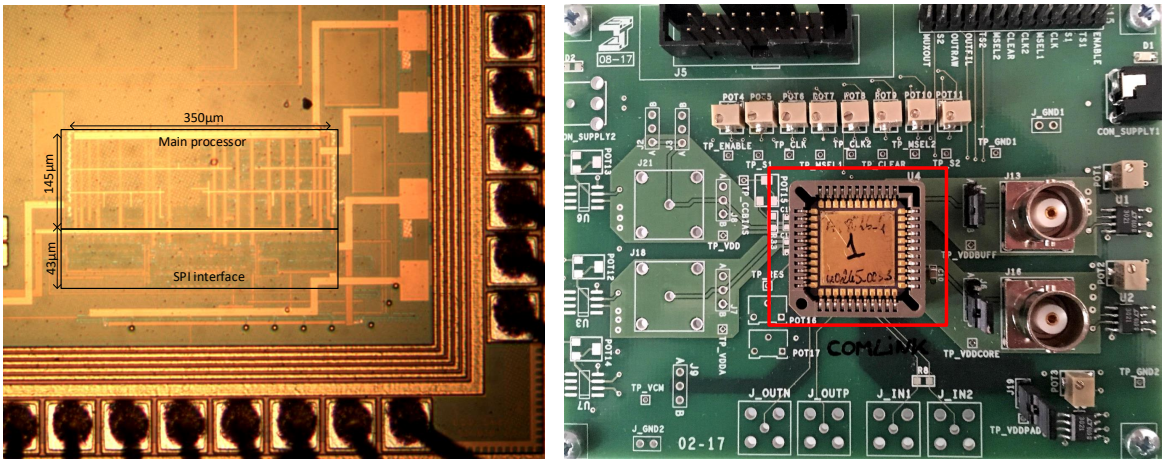

Fig. 3.12 Circuit layout fabricated in a 0.18μm CMOS process and occupies an active area of 0.05mm^2(left) adn dedicated PCB testing platform (right)

Fig. 3.13 Laboratory functional test setup, showing the logic analyser, power supply and the laptop used to transfer data files.

3.5 Experimental results

Fig. 3.12 shows a microphotograph of the chip implementing the *DDA* algorithm. The chip has been fabricated in a 0.18μm CMOS process and occupies an active

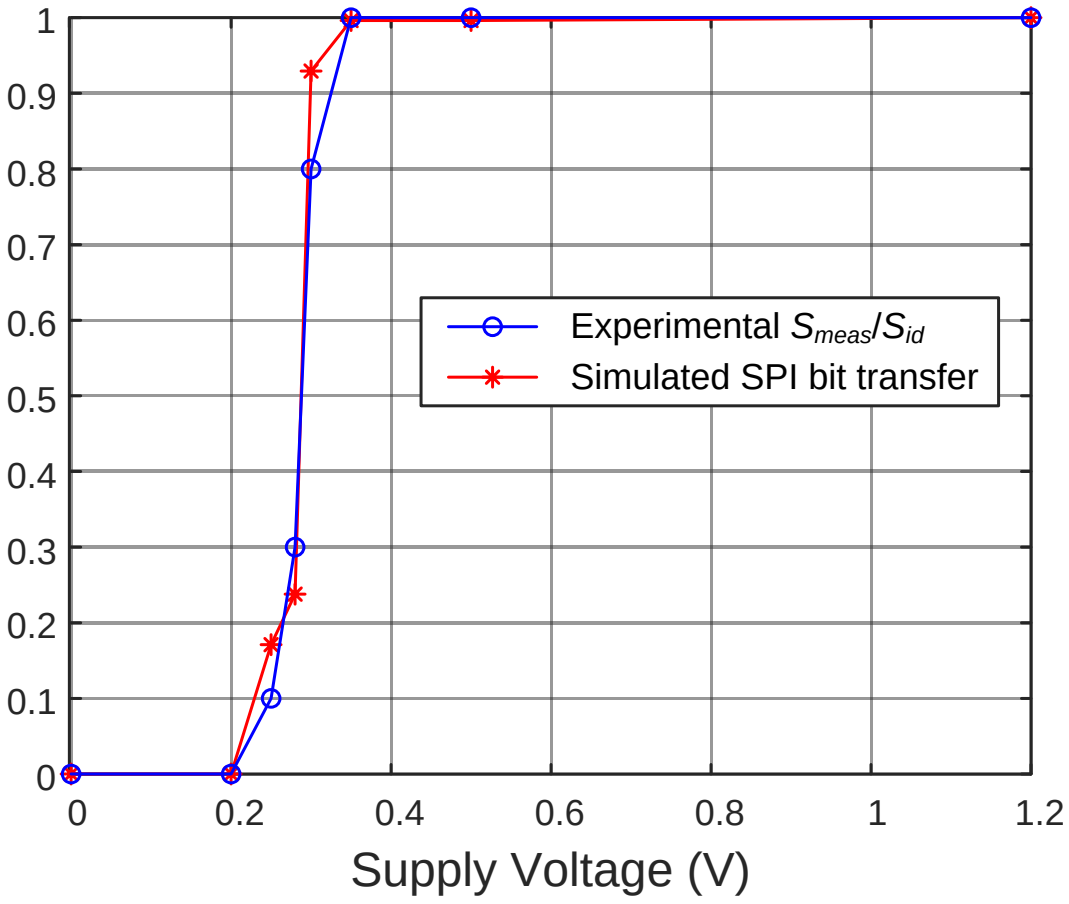

Fig. 3.14 Illustration of the system response versus supply voltage for two input tones at 25 and 30Hz.

area of 0.05mm^2 including the main processor core, two series-to-parallel input (SPI) ports, one parallel-to-series output (PSO) port and some additional buffers for testing intermediate nodes. The I/O ports work 10 times faster than the rest of the ASIC in agreement with the 10-b resolution used for the input signal samples. The total number of integrated gates is 6053.

Fig. 3.13 shows the test setup used for functional verification. The chip was mounted on a dedicated PCB together with a set of adjustable level shifters and regulators for scaling the logic levels and supply voltage of the ASIC. Firstly, Low voltage operation was experimentally verified by driving the chip with different tones in the β band and comparing the generated synchronisation index stream with electrical and behavioural simulations. A logic analyser (Agilent 16823A) was used for synthesising the tones, sampled at 1024S/s, and for serially retrieving binary information from the output and

other test points of the ASIC through a pod connector. For two tones at 25 and 30Hz (100k samples each), deviations from the ideal synchronisation index were noticeable for supply voltages below 0.35V. At this point, the index starts to drop and, for 0.2V supply, the system breaks down completely with no observed activity. The reset signal generated in the ASIC after and-ing the latched outputs of the two minima logic blocks (see Fig. 3.1) was also monitored along this transition. This signal goes high every time a transition period is computed. For supply voltages above 0.35V, the train generated by the ASIC is quite regular (only minor variations due to the quantisation of the inputs) in good agreement with the expected response, however, pulses are eventually missed below 0.35V and, at 0.2V supply, they are no longer observable.

Electrical simulations show that this behaviour is compatible with data transfer failures from the SPI input ports of the ASIC. This is illustrated in Fig. 3.14 which plots: (i) the experimentally observed degradation of the synchronisation index as the supply voltage decreases according to S_{meas}/S_{id}, where S_{meas} and S_{id} denote, respectively, the measured and ideal values of the index and, (ii) the simulated SPI register position, normalised to the vector resolution, at which input data fails to be shifted. In both cases, 1 indicates no error while 0 means complete breakdown. It is worth observing the good agreement between both curves. Similar behaviour was obtained for other tones in the β band.

At a signal input rate of 1024S/s and 0.5V supply voltage, the power consumption of the chip core is 15nW. Table 3.1 shows the power budget of the chip. Event detection and time stamp calculations by the finite state machine are the two most power demanding tasks of the processor due to the higher switching activity. Table 3.2 shows the normalised performance of the ASIC compared to other designs in Table 1.3. Variables have been normalised to a single channel and, in the case of core area occupation, an additional normalisation has been applied by arbitrarily assuming all chips are fabricated in CMOS 65nm. Accordingly, the original area per channel has been scaled by $(65nm/W_{min})^2$, where W_{min} is the feature size of the fabrication technology. Note that the proposed implementation exhibits the smallest normalised area and it is among the designs with lower energy consumption.

3.5.1 Experimental Setup with Neural Recordings

The processor has been also verified and validated using neural recording data available in the European Epilepsy database [57]. This database contains recordings of over

Table 3.1 Power breakdown of the synchronisation processor based on the blocks in Fig.3.1

Block	Average Power (nW)
Event detection	4.01
Finite state machine	4.52
Counter	1.46
Indexing	2.64
Filtering	2.86

Table 3.2 Normalised performance summary

	Norm. area (mm^2/ch)	Power (μW/ch)	Energy (pJ/ch/bit)
[49]	0.0073	6.25	108
[50]	0.0050	4.06	NA
[51]	0.0043	49.22	NA
[48]	0.0184	0.23	0.36
[52]	0.0223	51	3.75k
[53]	0.0047	0.15	63.5
[55]	0.5320	57.3	14k
This work	0.0033	0.0075	0.73

275 patients including 225 scalp recordings and 50 intracranial recordings. As very long recording sessions, often lasting for more than 6h, are included in the database, a specific experimental set-up has been devised. As shown in Fig. 3.15, it uses an mbed LPC1768 MCU based on a 32-bit ARM® Cortex™-M3 core from NXP Semiconductors. The test platform has been expanded with an application board which includes an RJ45 Ethernet connector and a USB-A Host/Device port (in addition to the built-in USB drag'n'drop FLASH programmer). The MCU defines settings and transmits driving stimuli to the *DDA* chip through the available GPIO ports and other I/O interfaces. Neural input signals are stored as CSV files in a 32 GB Fat-32 formatted stick memory plugged into the USB-A port of the test platform. These files are sequentially read by the MCU and serially transmitted to the SPI ports of the ASIC. Long testing sessions have been run uninterruptedly using this setup with no need to split the database recordings into smaller blocks (this would clear the internal states of the

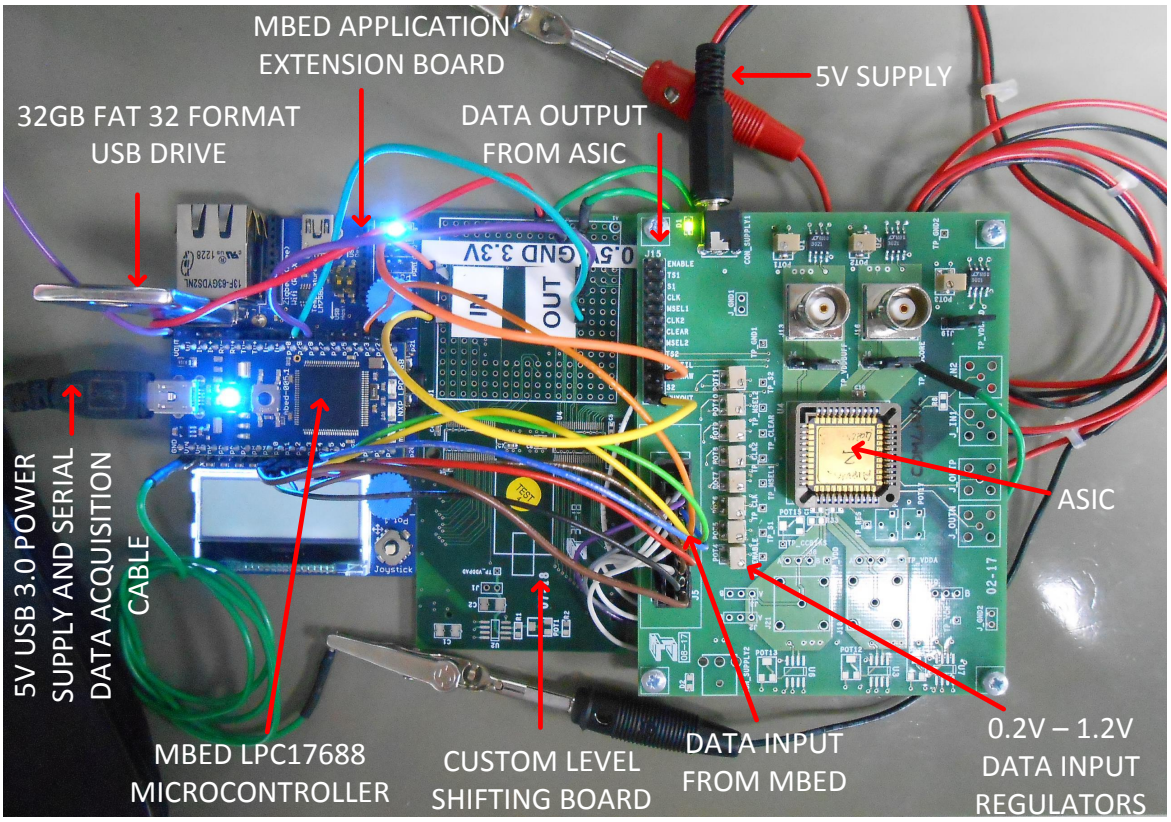

Fig. 3.15 Test setup showing the ASIC mounted on a dedicated PCB, mbed LPC1768 micro controller and dedicated level shifters.

ASIC). The synchronisation index generated by the chip is transferred back to the MCU and steered through the USB port to a host computer where data are further analysed with MATLAB®.

The C++ code to send data from the MBED device to the ASIC and retrieve data from the ASIC can be seen in appendix B.5.

Neural recording signals from the database have to be pre-processed prior to being used as stimuli in the described platform. This involves the following steps: (i) bandpass-filtering for selecting the frequency band of interest (fourth-order Butterworth filters have been used), (ii) resampling for matching the frequency rates of the recordings and the ASIC (only if needed), (iii) adding noise for a given input-referred SNR value, (iv) digitising the samples to the resolution of the ASIC (10-b), and (v) casting data in CSV format files for storing in the stick memory. As mentioned, files as large as 10GB have been handled without any observed problem.

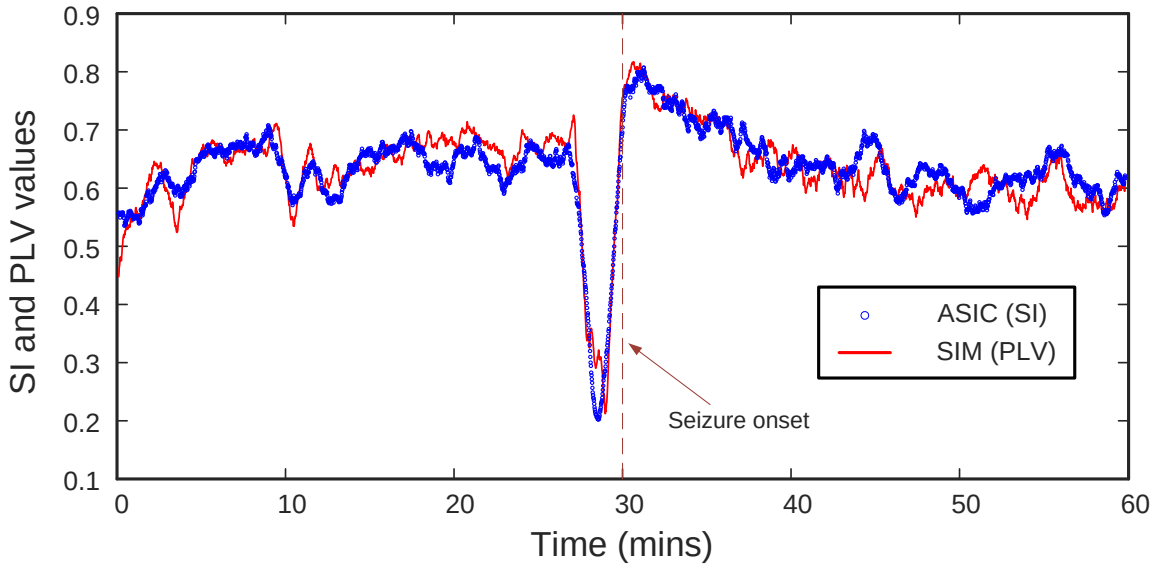

Fig. 3.16 Experimental synchronisation index and simulated PLV value from recordings at positions M5 and M8 of a 1×8 EcoG strip implanted in patient 1. Signals were pre-filtered in the β-band. The input rate of the chip was adjusted to the sample rate of the recordings, i.e., 1024S/s. The S_{dda} curve (labeled "ASIC SI") was obtained at a sample rate of 1S/s.

3.5.2 Epilepsy Detection Results

Different blocks of intracranial data from the referred database were used for assessing the epilepsy detection capabilities of the DDA chip. These blocks were selected for their high confirmed seizure rate. No other consideration was taken into account. For each block, signals were first pre-filtered in MATLAB® into conventional neuro-physiological bands. After a preliminary computer analysis of the records, the frequency band with the largest synchronisation index activity was finally selected. For each experiment, the set of valid values were plotted alongside with the theoretical PLV value obtained through the Hilbert Transform method.

For illustration purposes, Fig. 3.16 shows the synchronisation results obtained from two channels of a recording block measured in a 48 years old, female patient. In this block, one low amplitude fast activity (lafa) seizure was annotated (dotted line). Before driving the ASIC, Gaussian noise were added to the signals for a spot SNR of 30dB at 30Hz. The plot shows the experimental synchronisation index (labeled as "ASIC SI") and the calculated PLV value (labeled as "SIM PLV"). The DDA algorithm detects all major changes in synchrony and, indeed, follows a similar trend as the PLV algorithm.

This is verified by the high correlation measure of approximately 91% (in line with Fig. 2.9) .

Fig. 3.16 also shows that the seizure manifests with a remarkable increase in the synchronisation measures. This feature is in fact on the basis of the detection mechanism presented in [52] and [50] in which seizures are detected by applying thresholds on the PLV indicators calculated between pairs of channels. In some cases, as in Fig. 3.16, seizures are preceded by a sudden drop in synchronisation [67], however, we have not verified this feature in all the analysed blocks.

3.5.3 Functional Connectivity Results

The proposed DDA prototype has been also used for estimating functional brain connectivity. As in the previous section, chip results were later compared with computer simulations of the Hilbert Transform approach with PLV indexing. Recordings from patient #7 have been considered. This patient suffers from simple and complex partial seizures originated from temporal lobe. Among the recordings, a block including one rhythmic β wave episode has been analyzed. This has been the only aspect taken into account for the selection.

To assess the functional connectivity strength between neural assemblies, a mapping approach has been followed in which synchronisation indexes for every possible electrode combination at a given observation window were arranged into a symmetric matrix. A 1h long recording block including inter-ictal, pre-ictal, ictal and post-ictal periods was examined. Signals were band-pass filtered in the β band.

Both EEG and ECoG recordings are available from patient #7, however, only the results from EEG analysis are herein presented. The EEG recordings consisted of 16 EEG captures selected from a standard 10:20 system. The resulting 120 pairwise combinations were arranged in a 16×16 matrix. For easy visualisation, index values were colour-coded between yellow (index close to 1) and dark blue (no coupling).

Figure 3.17 shows the connectivity maps obtained through the Hilbert Transform (first column) and the DDA chip (second column) for observation windows in the four mentioned time segments. Both sets are clearly correlated as verified by the absolute differences between the two methods (third column). This is particularly noticeable during the ictal stage (first row), where the difference across all neural combinations is almost null. This shows that for high values of synchronisation, DDA retains the detectability of high synchronisation changes when compared to other more

complex algorithms. For smaller synchrony levels, as occurs in the pre-ictal, post-ictal and inter-ictal periods, the variations between both methods become a little more pronounced. That said, the biggest difference can be seen in the post-ictal period (third row) between electrodes T3-F8 at approximately 10%.

Similar conclusions were drawn from the analysis of ECoG maps on the same recording block (not shown). Good correlation between both approaches was also observed. Nevertheless, it was noticed an overall increase in synchrony for all the observation windows. This is consistent with the closer proximity of ECoG electrodes compared to surface electrodes. In any case, the ictal-period was still clearly identifiable.

Next, we analysed the graph theoretical metrics as described in Appendix A. Using non-directional, un-weighted binary matrices were the thresholds were set for each individual partition. To maintain consistency each threshold was set as 25% less than the maximum swing between the maximum and minimum respective values, such that: $Threshold_{partition} = max - \frac{max-min}{4}$.

In order to calculate the small world metric we introduce a comparison between the inter-ictal and ictal periods. Notably, we use the graph metrics from the inter-ictal period as a random reconnection of the ictal period such that $\lambda = \frac{C_{ictal}}{C_{inter-ictal}}$ and $\gamma = \frac{L_{ictal}}{L_{inter-ictal}}$. Conversely, for the inter-ictal period $\lambda = \frac{C_{inter-ictal}}{C_{ictal}}$ and $\gamma = \frac{L_{inter-ictal}}{L_{ictal}}$.

Fig.3.18, shows the various average graph theoretical metrics over the ictal and inter-ictal partitions. As expected the average nodal degree and clustering increases substantially during the ictal period. Conversely, the average path length drops over the ictal period. This indicates an overall more heavily synchronised and connected network, as can be seen in the ictal connectivity graph. Fig. 3.18, also shows that the ictal partition tends towards a higher small world connectivity than the inter-ictal period. Nevertheless, this highly connected ictal period as compared to the inter-ictal period indicates a mass synchrony between multiple nodes. In a normal brain network, however, these characteristics can be considered undesirable as neural clusters should operate with a certain degree of independence and not oscillate at the same frequencies.

Further analysis, was made for patient #7 in terms of the variance in nodal degrees and clustering coefficients. Using, 'topoplots' in the form of colour graded contour plots projected onto a 2D representation of the brain.

Fig. 3.19, show the variance of these metrics for patient #7. Over the ictal period, there exists a high nodal degree towards the right temporal lobe which would indicate higher connectivity from other nodes to this area. This is indeed consistent with the

clinical prognosis for this patients type of epilepsy. Nevertheless, The clustering during the ictal period tends to shift to the left hemisphere, indicating that although the right temporal lobe is heavily connected during this period the connections tend to be longer range and less clustered than the left hemisphere, Nevertheless they remain high at approximately 0.6. Over the next three periods, noticeable reductions in nodal degree in the right temporal lobe can be seen, indicating that this area actually has a lower connectivity than other regions. In fact, a reduction in both the inter-ictal and pre-ictal periods accounts for an approximate 23% reduction from in the epileptic event time. Interestingly, the highest connectivity during the non-ictal period shifts towards the P4 parietal electrode which is consistent with the correlation colour maps seen in Fig. 3.17.

Another interesting, characteristic for this patient is the reduction in clustering during the pre-ictal period, indicating a sudden drop in clustering around the epileptic centre before the ictal event is triggered.

Once again the gathered DDA data shown in the plots was compared with that of the PLV and HT method for a comparative visual aid. In all cases, the two algorithms show a strong correlation with all shifts in clustering and nodal degrees trending towards the same pattern.

In conclusion, the ictal period proves itself to evolve from a non-hyper synchronous network to a hypersynchronous network. Where the epileptic event originates from one point of origin and incorporates new clusters into the firing process. This is indicated by the short path lengths, meaning that long-range synchrony does not necessarily take place, instead, the synchrony evolves from one cluster to another. On the contrary, during the inter-ictal period, long rage synchrony is more apparent and the average nodal degree is low.

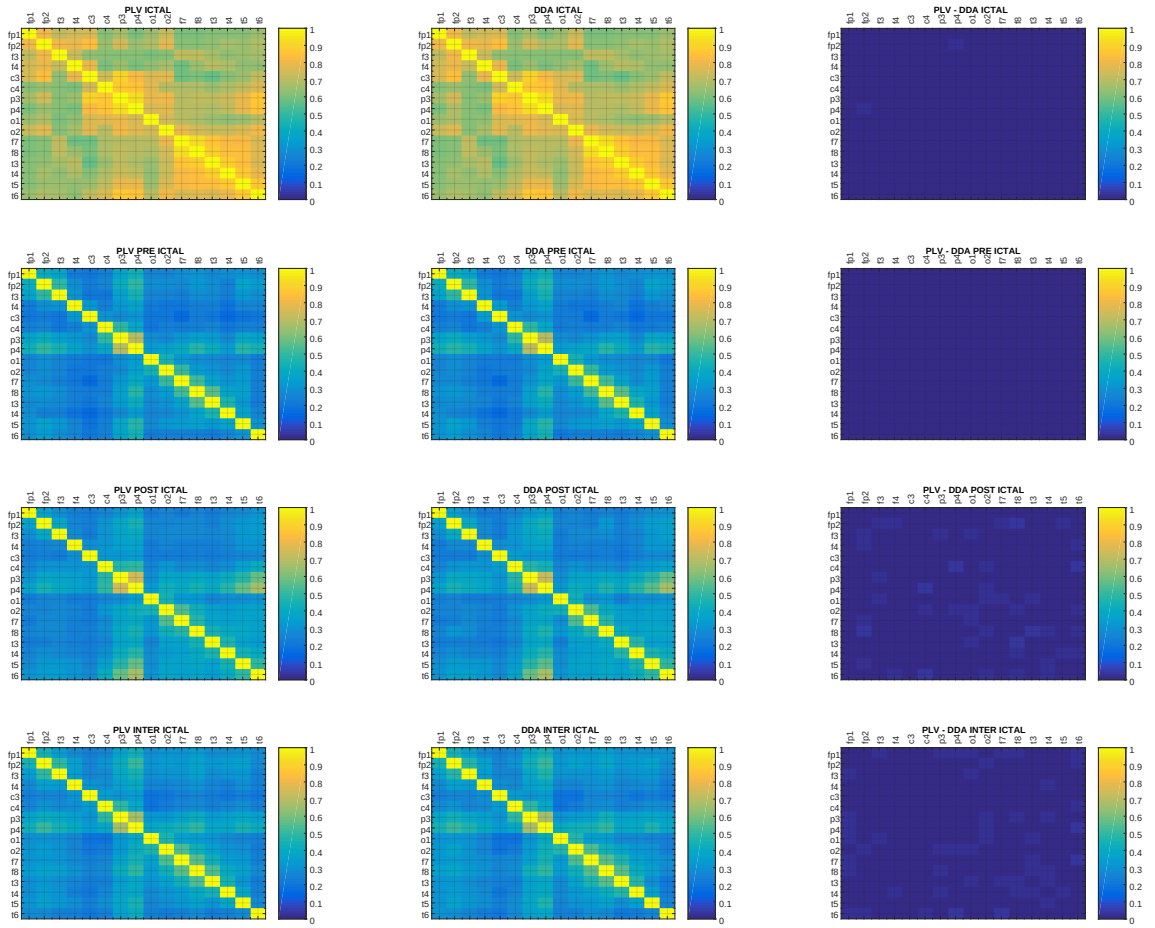

Fig. 3.17 EEG connectivity maps for patient #7. Columns 1 through 2 are colour maps representing the mean values for both the PLV and DDA for all possible combinations of EEG signals. Column 3, shows the absolute difference between columns 1 and 2. EEG data were organised into a standard 10-20 format.

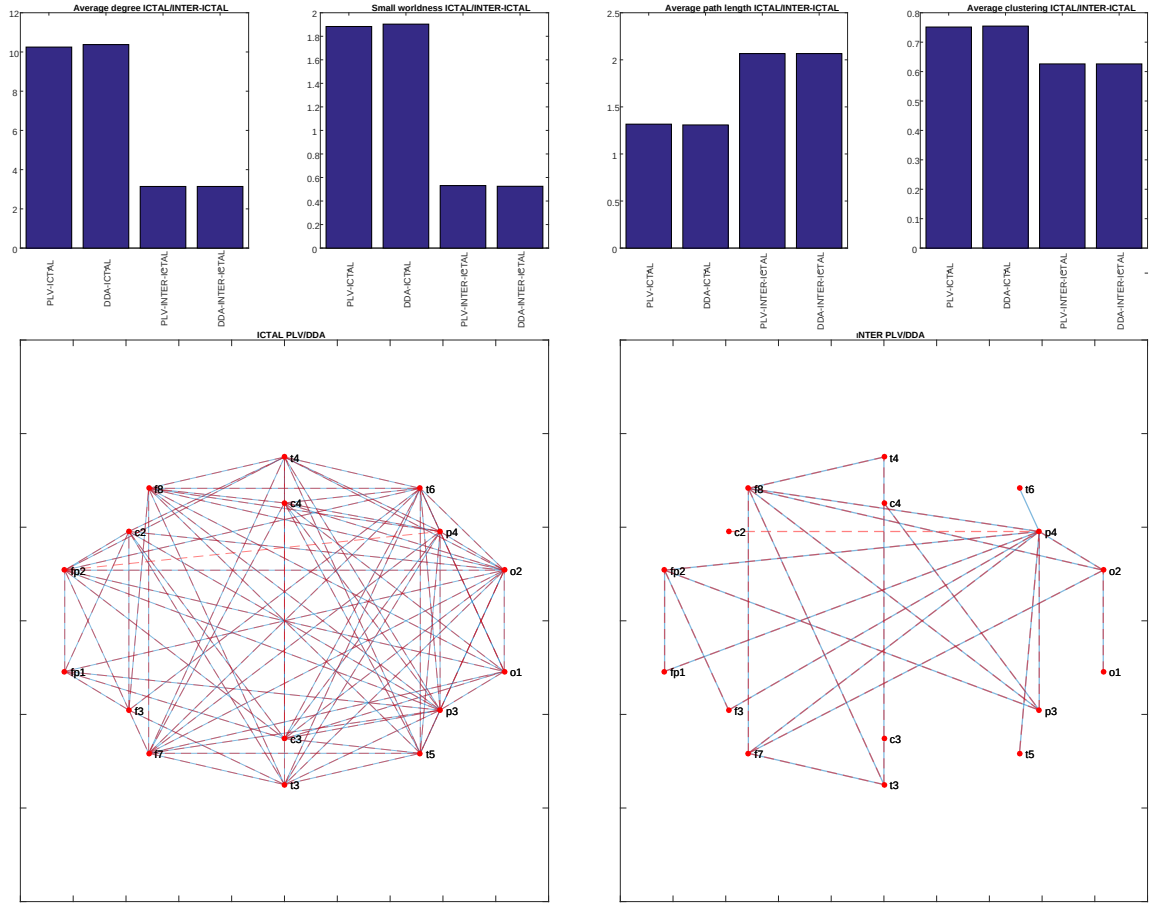

Fig. 3.18 Graph theoretical results for patient#7. Row 1, shows the average metrics for both the PLV and DDA approach over both the ictal and inter-ictal periods. Row 2, shows the functional connectivity representation of the brain using a standard 10-20 EEG electrode set-up for both the ictal and inter-ictal partitions. The blue solid line represents the connectivity of the PLV value and the dashed red line represents the connectivity of the DDA approach

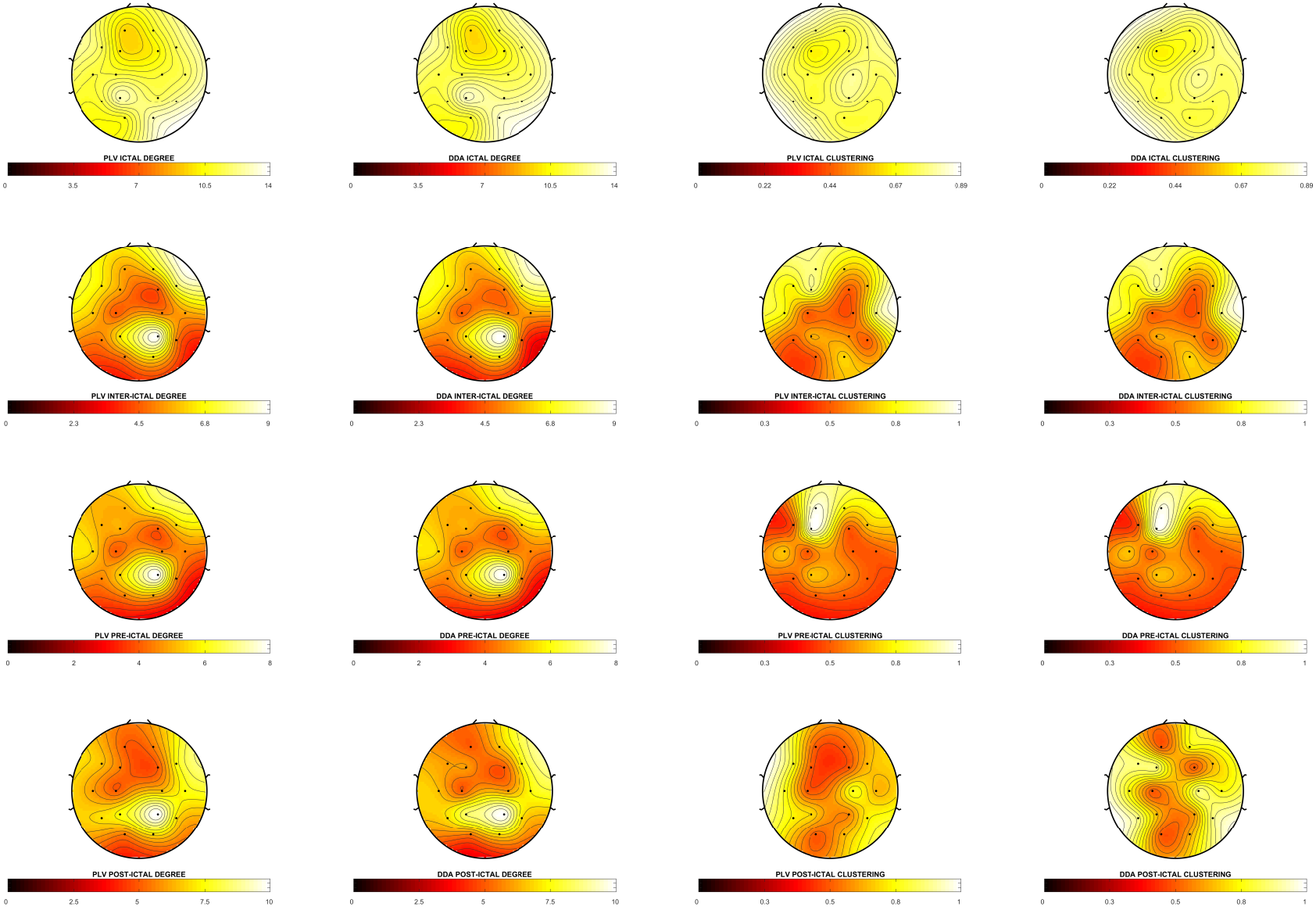

Fig. 3.19 Graph theoretical coloured contour maps for patient 442012 during each partition period ictal,inter-ictal,pre-ictal and post-ictal. Where each row shows both the variance of nodal degree and clustering for both the PLV and DDA approaches.

3.6 Conclusion

A dedicated processor, fabricated in a 0.18μm CMOS process, has been proposed for the estimation of phase synchronisation between neural signals. It obtains similar results as those achievable with more computationally demanding algorithms, but it consumes much less power and it is highly scalable. This makes the proposal suitable for parallel multi-channel phase synchronisation calculations on-chip.

In some sense, the proposed algorithm can be regarded as an inexact computing paradigm [68] in which a small amount of errors can be tolerated, without sensibly degrading the quality of results, but attaining considerable efficiency gains.

Although the processor performance has been demonstrated for surface and intracranial EEG signals, its usage can be extended to other scenarios where a low-cost, low-complexity, programmable and portable solution for computing functional connectivity is needed as, for instance, in functional magnetic resonance imaging (fMRI) or positron emission tomography (PET).

Chapter 4

Multi-channel synchronisation processor

The contributions in this section include the design, implementation and verification of a sub-threshold phase mass integrated phase 16-channel VLSI synchronisation processor capable of calculating 'phase synchronisation between 9 independent neural signals and 120 neural combinations. The processors design and verification process was firstly implemented in an FPGA environment for architecture verification, later the design was implemented manually in the CADENCE layout environment and verified through simulation.

All of the design and verification processes were carried out by the author of this thesis.

4.1 Overview

This design was implemented with a multi-functional purpose in mind, firstly, as a clinical aid for the identification of pathological brain states, by monitoring and mapping synchronisation features of patients using graph theory. Secondly, as a dedicated epileptic seizure diagnosis tool.

The ASIC was designed in an AMS $0.18\mu m$ high voltage, low leakage technology using a fully custom 0.5v threshold digital logic library to reduce total power consumption and operates from a main clock frequency of 2KHz.

Incorporated into the design are 16, individual synchronisation processors (15 on-line processors and 1 test processor) each with a dedicated *training and calculation module*, used to build a specialised epileptic detection system based on patient-specific synchrony thresholds. Each of the 15 main synchronisation processor are based on the previously verified synchronisation processor in section 3 and are capable of calculating the phase synchrony between 9 independent EEG signals over 8 epochs of time totalling 120 EEG combinations. Furthermore, two SPI interfaces are incorporated for ease of communication to 3^{rd} party devices.

To optimise area, the processors were organised into an array of 4X4 of processors (P) and training and calculation modules (T), this is illustrated in Fig. 4.1. In addition, an independent control logic unit was implemented for each row of the array.

The last processor in the array was used as a test processor which was implemented completely independently from the other array.

The final VLSI implmented design can be seen in Fig 4.2. The core area of the design occupies only $3.64mm^2$, whilst the SPI input interface occupies $0.091mm^2$ and the output interface occupies $0.063mm^2$. Each synchronisation processor occupies $0.04mm^2$ (This is less than the original processor described in section 3, due to a reduction in control logic) and consumed on average 12-15nW (Simulated power estimation), whilst, each training and calculation module occupies $0.066mm^2$.

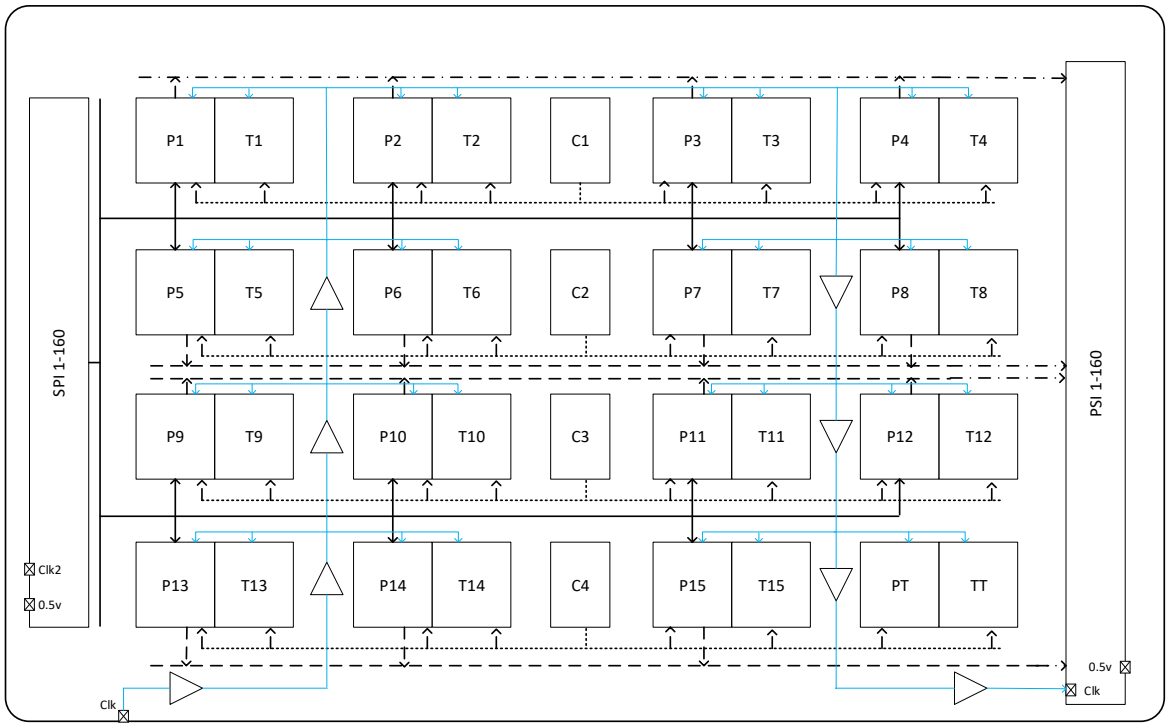

Fig. 4.1 Top level matrix layout for the 16 synchronisation processors and associated training and calculation modules

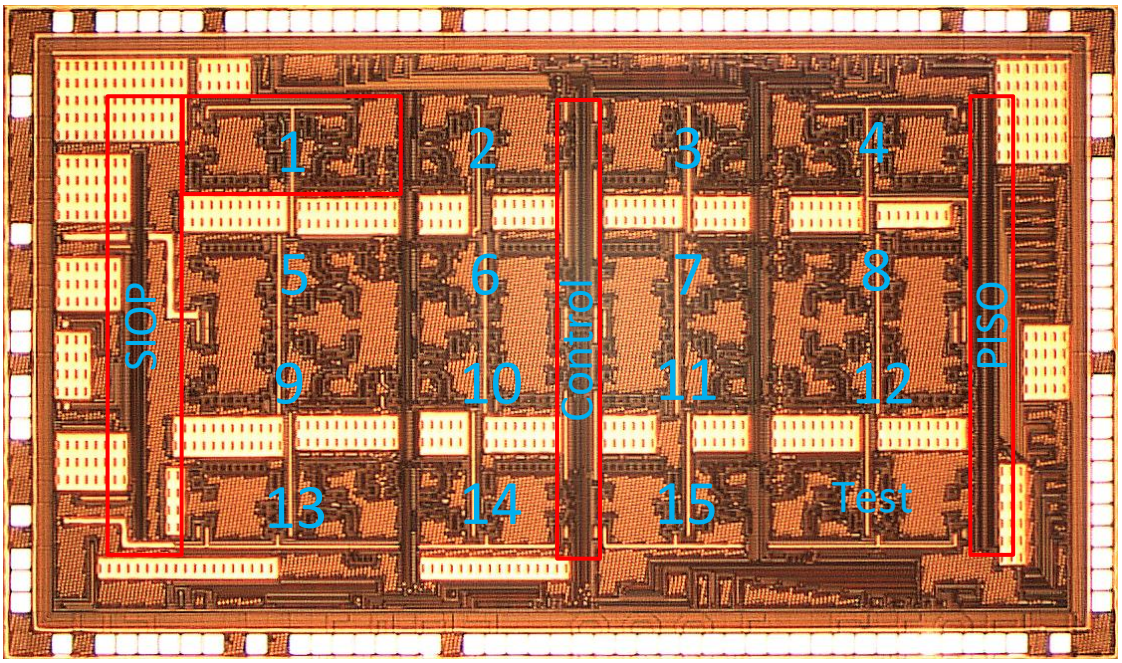

Fig. 4.2 Microscopic view of 16-Channel ASIC

4.2 Design

4.2.1 Input conversion SPI

A single 1 to 160-b serial to parallel converter is used at the input of the ASIC to convert a single 1 bit EEG data stream into a 16x10 bit array of neural data. Where each 10-b of data represents a single sample from every 16 channels of data. Fig. 4.3 shows a block diagram of the converter which is made up of three distinct layers. Firstly, a concatenated serial array of D-type flip-flops are interconnected via their inputs and outputs, where the input of the n^{th} flip-flop is connected to the output of the $(n-1)^{th}$ flip-flop for a total of 160 flip-flops. Since each neural data stream sample consists of 10 unsigned bits, the converter can translate 1 sample for each neural data stream over a total of 160 clock cycles. The serial data stream is feed into the input of the converter as the least significant bit first. Once 160 cycles are complete, the data is shifted out of the registers in parallel, which then gets stored into an individual latch for each bit of data (i.e there are 160 1-b latches). This latching is necessary as the clock of the input converter (Clk2) is running 160 times faster than the main clock of the circuit (Clk), the latching system is therefore used to hold the data in place for 1

complete cycle of the main Clk. This way, the initial registers can continue loading the next samples, while the previous sample is held stable.

Finally, the latched data is pushed into an individual buffer for each data bit, in order to drive the logic 1 values high and pull the logic 0 values low. This is a crucial part of the system as the long conversion dissipates power every cycle, leading to a diminished voltage. The new data samples are then pushed onto a 160-b bus which interconnects all of the processors.

The serial interface is beneficial for interfacing the ASIC with 3^{rd}party hardware platforms such as FPGAs and reduces overall area consumption from 160 individual inputs.

4.2.2 Output conversion PSI

The output converter is a 160 to 1 bit parallel to serial converter and is used at the output to capture synchronisation values from the 15 main processors. Fig. 4.4 shows a block diagram of the converter. The architecture consists solely of a concatenated array of D-type flip-flops and a single buffer at the output. An enable line is activated every time new synchronisation values are ready, which tells the converter to accept the data at its input. If no new values are available the converter blocks new inputs and starts shifting the values out of the registers. The long period between updated synchronisation values means that this converter can run using the master clock of the circuit.

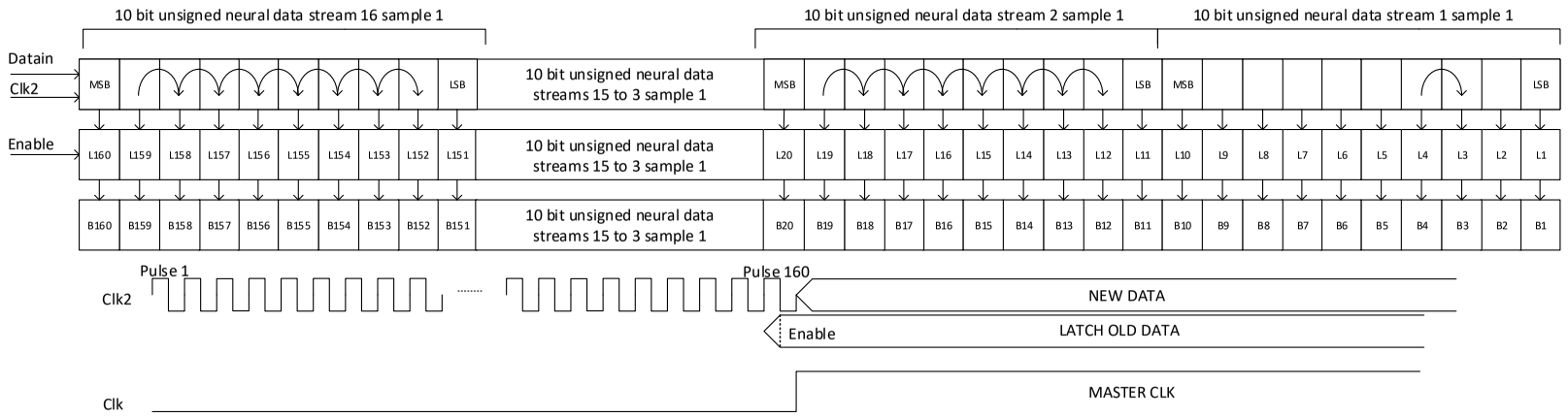

Fig. 4.3 1 to 160 SPI for the conversion into 16-10bit unsigned digitised neural signals

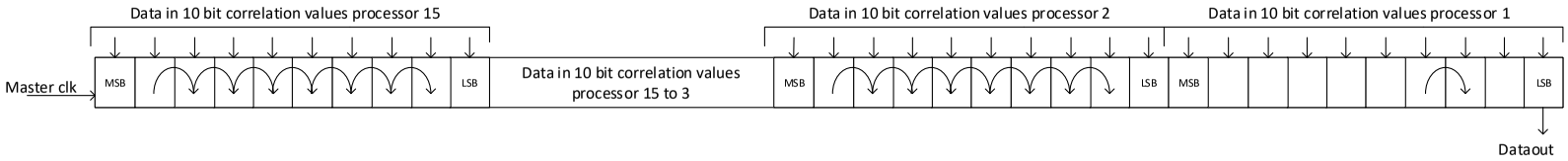

Fig. 4.4 160 to 1 PISO converter for the conversion of 16-10bit unsigned correlation results into a single digital stream

4.2.3 Epoch selection

Using 16 main electrodes, a combinational total of 120 neural combinations need to be assessed. Normally, this would lead to a large array of 120 synchronisation processors, where each processor is capable of handling two data streams. To combat this, we use a custom epoch based multiplexing of neural data to reduce the array of processors from 120 down to just 15.

The 'epoch', on which the method is based, can be considered as a small time segment window, where 1 epoch lasts for a total window size of x samples. The actual time of the window is dependent on the sampling frequency of the master clock, however, for argument's sake, let's assume for now that it is set to a standard sampling rate of 1024S.

During an epoch, each processor will receive 1024 samples from any two given neural data streams. Therefore, during 1 epoch the 15 processors can calculate the correlation between 15 neural data combinations. In order to reach the total of 120 combinations, 8 epochs are required.

The epoch system, therefore, reduces the number of processors from 120 to 15, at the cost of losing small amounts of data between epoch cycles. In terms of samples, this means that each neural data combination will get updated every 8192 samples or in terms of time every 8 seconds.

The tradeoff between data loss and realistic implantation schemes is questionable. Nevertheless, it is clear that as a diagnostic tool for pathological brain states such as Alzheimers, Parkinsons or schizophrenia, the need for constant data monitoring is unnecessary. Mainly due to the constant and stable change or deterioration in the neural network. That's to say, the instability in the neural network does not change rapidly or fluctuate from a better to a worse state or visa versa. In epilepsy, however, abrupt rapid changes in the neural network are the most common attribute. A paper by et. al Sigmund Jenssen [69], reports that on average a Tonic seizures last for a median time of 18.5 seconds, simple partial seizures last for 28 seconds, primary generalised tonic-clonic seizures lasted approximately 66 seconds and complex partial seizures lasted for as long as 78 seconds. From these results, it is clear that an 8-second refresh rate for neural synchronisation values is sufficient. In the fastest seizures of 18.5 seconds there would be a minimum of two attempts at detection of the epileptic event and in the slowest seizures, there would be almost 10 attempts at identification.

Processor	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5	Epoch 6	Epoch 7	Epoch 8
1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	2,10
3	3,4	3,5	3,6	3,7	3,8	3,9	3,10	3,11
4	4,5	4,6	4,7	4,8	4,9	4,10	4,11	4,12
5	5,6	5,7	5,8	5,9	5,10	5,11	5,12	5,13
6	6,7	6,8	6,9	6,10	6,11	6,12	6,13	6,14
7	7,8	7,9	7,10	7,11	7,12	7,13	7,14	7,15
8	8,9	8,10	8,11	8,12	8,13	8,14	8,15	8,16
9	9,10	9,11	9,12	9,13	9,14	9,15	9,16	10,1
10	10,11	10,12	10,13	10,14	10,15	10,16	11,1	11,2
11	11,12	11,13	11,14	11,15	11,16	12,1	12,2	12,3
12	12,13	12,14	12,15	12,16	13,1	13,2	13,3	13,4
13	13,14	13,15	13,16	14,1	14,2	14,3	14,4	14,5
14	14,15	14,16	15,1	15,2	15,3	15,4	15,5	15,6
15	15,16	16,1	16,2	16,3	16,4	16,5	16,6	16,7

Table 4.1 Table showing which neural data streams are connected to specific processor during each of the 8 epochs of time

Table 4.1, shows a grid of the neural signals being computed for each processor during any given epoch. As an example processor 1 (P1), calculates the synchrony between neural data channels, [1-2,1-3,1-4,1-5,1-6,1-7,1-8,1-9] at epochs 1 through 8 respectively. Whilst processor 15 (P15), calculates the synchrony between [15-16,16-1,16-2,16-3,16-4,16-5,16-6,16-7].

4.2.4 Window, epoch and memory addressing logic

Structurally speaking, the control logic for the entire system is simple and consists of basic logic gates and counters. Firstly, as can be seen in Fig. 4.5 (Window counter), the window logic consists of a simple 10-b up counter with self-reset. This counts from 0 to 1024, when the maximum of 1024 is reached a single clock cycle positive pulse 'K' is created. From observations in the previous design discussed in chapter 3, a fixed window of 1024 was chosen. This allows for a reduction in hardware and power consumption by omitting the variable shifting aspect.

The epoch selection of the neural signal inputs and addressing of memory elements is driven by a 3 bit up counter, built from D-type flip-flops as seen in Fig. 4.5 (Epoch selection and memory addressing logic). Where the output of each flip-flop is hard-wired to 8 distinct 3 input AND gates in combinations which create an up sequence ranging from 1 through 8 (a1 through a8). The counter has an enable line connected to 'K',

hence the counter only counts up once 'K' is introduced. Contrarily, when 'K' is low, the counter is paused in its current state meaning that the count is kept constant until the next 'K'. This combination allows us to produce logic high signals of exactly 1024 samples long. Using the hard-wired AND gates at the output has the advantage of hardware reduction over ring counters as a reduction of five D-type flip-flops can be introduced. The long pulses of a1 through a8 are used as a reference for the current epoch as well as address memory locations, which will be described later. Fig. 4.5 shows a basic timing diagram, where the 1024 sample pulses of a1 through a8 rotate continuously. In the zoomed view of a1, we can see how a single cycled pulse of 'K' activates the epoch a1.

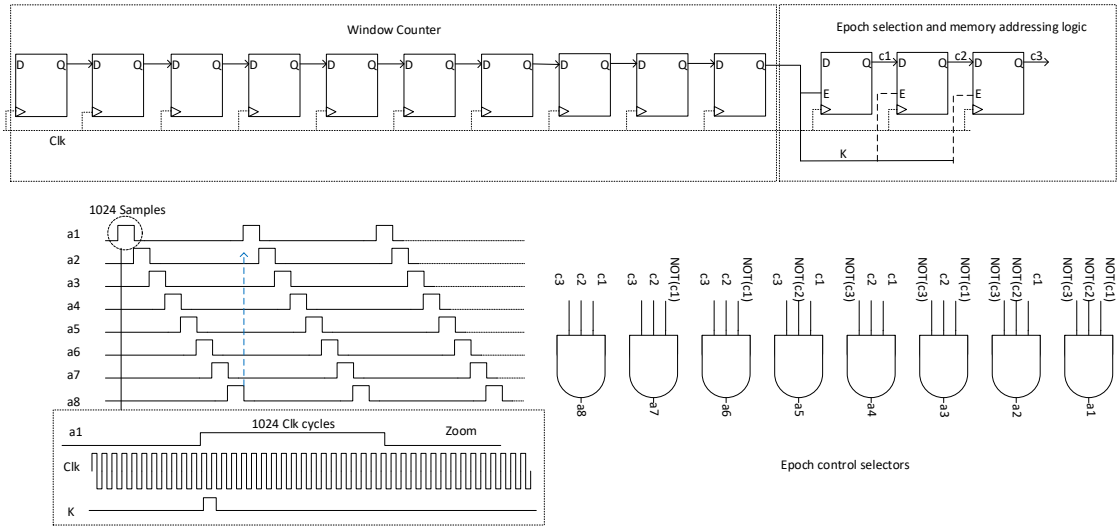

Fig. 4.5 Dedicated control logic for the window length, epoch selection and memory addressing.

4.2.5 Flow

From Fig. 4.6, we can see an overview of the hardware implemented for processor 1 (P1) (all 15 processors are identical). Firstly, during any given epoch, 2 of the 9 possible channels of parallel neural data are multiplexed into the synchronisation processor as per, table 4.1. During epoch 1 signals S1 and S2 are evaluated, the phase error between these two signals is then continuously monitored and updated until the window has finished (K = 1) producing a phase value ϕ_{epoch1}. Simultaneously, the Epoch has increased to 2, hence new data streams S1 and S3 are feed into the

synchronisation processor for assessment. Working as a pipelined system ϕ_{epoch2}, is now being processed whilst ϕ_{epoch1} continues to be post-processed. In general $\phi_{epoch(n)}$, is always being processed whilst $\phi_{epoch(n-1)}$, is always being post-processed.

ϕ_{epoch1}, is then passed into a dedicated custom low pass filter and stored in a custom dedicated memory for future retrieval. After filtering, ϕ_{epoch1}, is then passed to a dedicated training and calculation module which uses the values of synchrony to produce patient-specific epileptic detection and graph theoretical thresholds for online epileptic detection and offline graph analysis respectively.

The dedicated thresholds are then used as a comparison in a detection block which compares new synchrony values to determine whether a rise or fall in a patient's specific synchrony values has occurred.

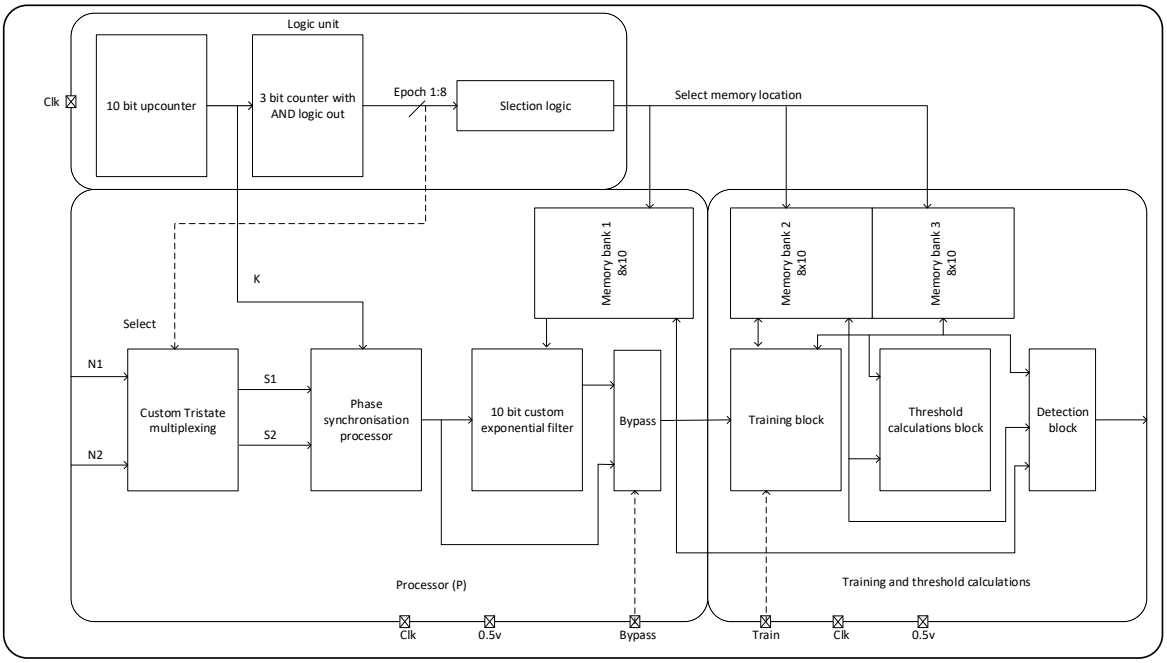

Fig. 4.6 Overview of the 16-channel processors control and flow

4.2.6 Neural multiplexing

To accommodate all 120 neural data combinations over 8 epochs of time, each processor must manage the phase calculations of 9 independent neural combinations. In order to minimise the large area occupation enforced by standard multiplexing schemes, our design incorporates a non-conventional hard-wiring via tristate buffers.

Fig. 4.7 shows the basic hard-wiring technique. Where select processor input channels are connected via a multi-dimensional 8x10 array of tristate buffers. Each of the 8 rows of 10 tristate buffers connects to an individual neural data stream of 10-b, which all have a dedicated enable line connected to a specific epoch. This allows us to connect 8, 10-b neural data streams to a single bus.

When a current epoch is active (a1 through a8, see section 4.2.4), the output of that given tristate array is set to low impedance, Hence, the data can pass onto the main line and feed into the synchronisation processor unimpeded. Contrarily, when an epoch is not active the output is set to a state of high impedance (Z), meaning that the neural data is blocked from passing onto the main line.

From Table. 4.1, for synchronisation processors 1 through 8, the first channel is always hard connected to neural data streams 1 through 8, respectively. Whilst channel 2, is connected to each processor's dedicated tristate array. As an example in Fig. 4.7, channel 1 (Ch1) of processor 1, is hard-wired to the neural data stream 1 (D1), whilst channel 2 (Ch2), is connected to the tristate array, the first row in the array is connected to neural data stream 2 (D2), the second row to neural data stream 3 (D3) and so on until the 8^{th} row, which is connected to neural data stream 9 (D9).

Synchronisation processors 9 through 15 incorporate an extra 2-row tristate array at channel 1. As per table 4.5, eventually the input at channel 1 will have to change, however, this is not a uniform process so a small piece of distinct logic is added to control the switching of these tristate rows. As an example during epoch 8 of processor 9, the Ch1 input will need to switch from D9 to D10. To achieve this a simple inverter is added to the enable line of the first row, such that if epoch 8 is not active then neural data D9 can flow into Ch1, else D10 is allowed to pass unimpeded. The additional logic for each of the processors 10 through 15 can be seen in Fig.4.7.

The timing diagram shows the currently active rows for processors 1 and 9 during epochs 1 (highlighted in green) and 8 (highlighted in blue).

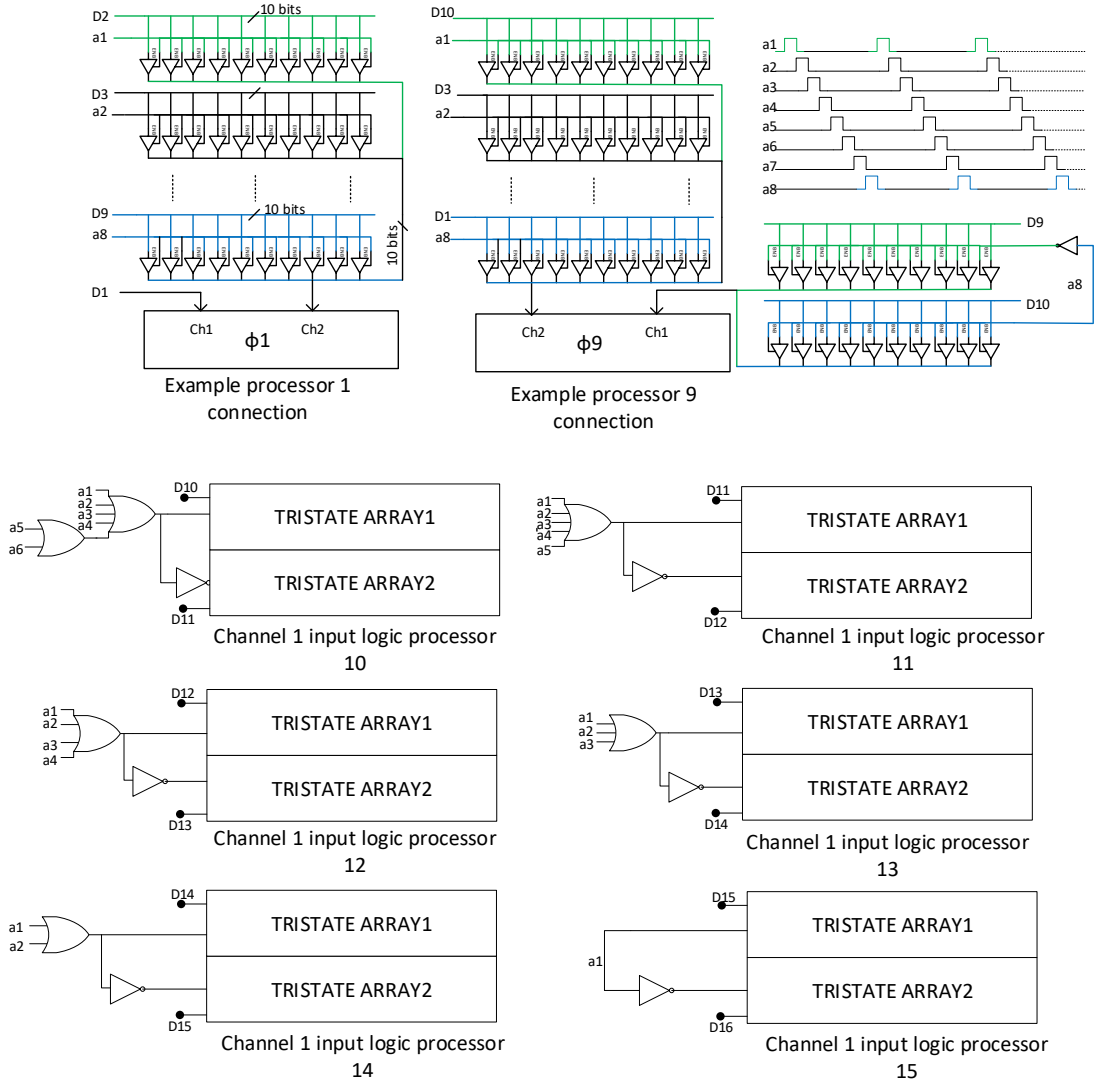

Fig. 4.7 Example of the neural input multiplexing scheme and dedicated control logic. The top figures show how tristate buffer arrays are used to control 8 separate neural streams to one processor. The bottom figures shows the extra control logic needed in order to control the first input channel

4.2.7 Post processing

The epoch based method chosen means that temporary variables and data need to be stored for future access. The exponential filtering is an example of this, where the original low pass filtered $\phi(i)$, from Fig. 3.3 of section 3.2.4 in chapter 3:

Now needs to store the previous value of $y(i)$ for each epoch, until a full rotation of the signals has been complete. The equation can then be expressed as:

$$y(i)^{(P)} = y(i-1)^{(P)}_{epochx} - y(i-1)^{(P)}_{epochx} \cdot 2^{pos} + \phi(i)^{(P)}_{epochx} \cdot 2^{pos}$$

Were $y(i-1)_{epochx}$, represents the previously stored filtered value for that current epoch and P, represents the processor.

From herein, we will omit the term epochx and (P) for clarity of reading, hence we assume that all further references are indicative of all 15 processors and epochs

Since each processor manages multiple neural data streams, a dedicated 8x10 memory, which is illustrated in Fig. 4.8 (a), is used to store the previous values of $y(i-1)$, until the epoch selection module makes a full round and returns back to the chosen epoch. The memory itself consists of a 10x8 array of D-type latches all interconnected at the output via tristate buffers (T1 through T8) onto a single 10-b bus. This allows for both the read and write operations to be carried out simultaneously. The latching of the data to the correct address requires a single cycle positive pulse in this case 'a1Lf' through 'a8Lf'. Which are activated by 'anding' the current epoch with the single cycled window pulse of 'K' as illustrated in Fig. 4.8 (b). Once the new data is latched to the correct address, the output of the memory is read constantly throughout the life of each individual epoch via epoch selection signals a1 through a8.

Fig. 4.8 (c), shows an overview of the hardware requirements for such a filter. Which is similar to the hardware described in from Fig. 3.3 of section 3.2.4 in chapter 3. Nevertheless, by assuming a constant value of *pos*, in this case, $pos = 2$ a reduction of two 10-b shift registers can be omitted by hard wiring the shift into the dedicated hardware. Firstly, a 10-b subtracter calculates $y(i-1) - y(i-1) \cdot 2^{pos}$ (collected from memory), where, $y(i-1) \cdot 2^{pos}$ is hard-wired to the subtracter by taking the most significant bits from 10 down to 2 and grounding out the first 2 most significant bits and concatenating bits 10 to 2 in position 8 down to 1

The result from the subtracter, is then added to $\phi(i) \cdot 2^{pos}$ (once again hard-wired), using a 10-b adder to create the final result $y(i)$, which is simultaneously passed on to the next stage and stored into memory as $y(i-1)$.

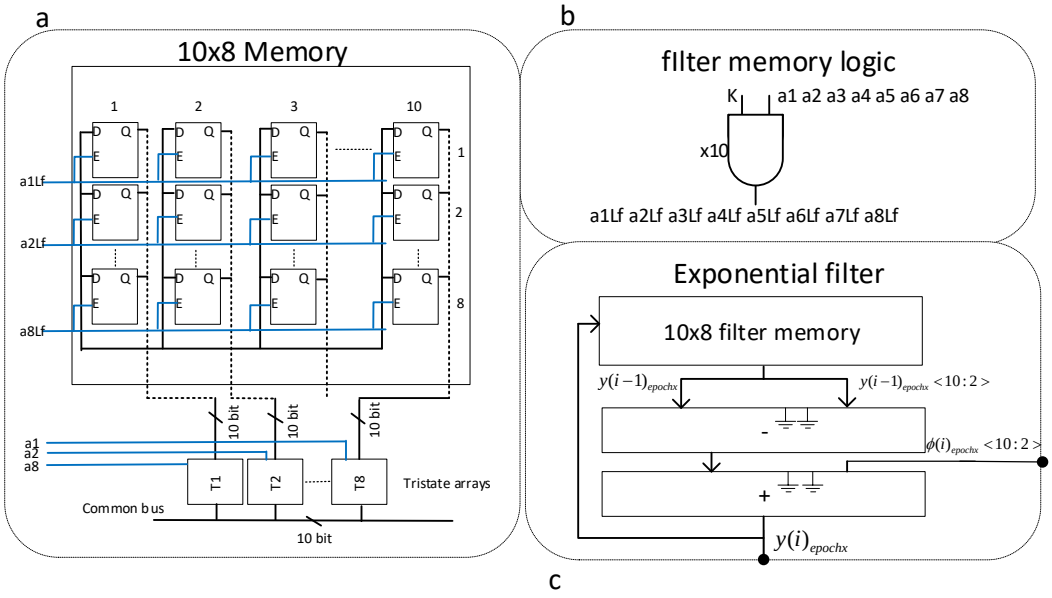

Fig. 4.8 Adapted exponential filter from processor 1, using a hard-wired smoothing factor and memory array

4.2.8 Training and Thresholds overview

It is well known that there are a wide variety of different forms of epilepsy which can produce different varying quantities of synchrony between different regions. Additionally, each and every patient will produce varying amounts of synchrony due to the uniqueness of every brain. These variations make it difficult to set reliable static thresholds for epileptic detection as seen in Fig. 3.6 of section 3.2.5 in chapter 3, where the thresholds were set based on pre simulation. Therefore, in this design, a patient-specific epileptic threshold calculator was implemented.

To begin, an initial training phase is set, such that 2 separate arrays are built, 1 containing the maximum filtered synchronisation values and the other the minimum values, from each processor at each epoch (i.e the maximum and minimum filtered values for each neural data combination).

In an ideal situation, the training period would be performed during a patient's epileptic episodes by clinical specialists. The arrays, give an accurate starting point for

the patient's pathological synchronisation state. Once training is complete, the data accumulated is then used to set a patient specific low and high threshold voltage such that:

$$Thh = max(y(i)) - \Delta \tag{4.1}$$

$$Thl = min(y(i)) + \Delta \tag{4.2}$$

Where, p, is the processor in question and Δ is a margin calculated as the voltage swing between the maximum and minimum values stored in the array divided by a constant such that::

$$\Delta = \Big[max(y(i)) - min(y(i))\Big] \cdot \frac{1}{2^n} \tag{4.3}$$

A threshold low (Thl) is set to produce alerts when the patient synchrony levels are increasing towards the maximum recorded value in the array, warning the patient to a possible upcoming episode. The threshold high (Thh) is set as a diagnostic tool to track a possible upcoming epileptic seizure.

The time for an accurate array to be built depends greatly on the patient and the amount of successfully induced episodes recorded. Take for example a patient who, did not reach a positive seizure state during the training period, this patients Thh and Thl thresholds will not give an accurate gauge of his/her hypersensitive network detection.

This dynamic approach to thresholds gives rise to several advantages over fixed threshold based architectures. Firstly, since the thresholds are based from a patient's specific source, the thresholds are dynamically adjusted to account for a synchronisation offset which may occur from patient to patient. Secondly, In the fixed threshold hold based algorithmic designs a normalization between 0 and 1 is usually applied. This makes setting the thresholds easier, however, the thresholds are still susceptible to large error margins due to offsets of patient-specific synchrony.

The dynamic thresholds also provide an all-important feature towards the graph theoretical analysis. The complexity of threshold setting for the binary adjacent matrices (see appendix A for definition of graph theory), usually leads researches to handle multiple data sets at varying thresholds.

It is pertinent to point out that, these dynamic thresholds can be recalibrated at any moment in time. This may be desirable to change when different frequency bands are being evaluated, due to synchrony baseline offsets.

The architecture can be seen in Fig. 4.9. In order to minimise hardware utilisation, a combination of multi-purpose/reusable, 8x10 D-type latch memories are used for the 2 arrays, in combination with a 10-bit subtracter, several hard-wired tristate arrays, twin MSB comparators and control logic.

Training

1. During the initial set up of the device, a small initialisation period is set (Inital = 1). This allows for the low memory $MemL$, which will ultimately store the low thresholds Thl to be loaded with a logic level 1 and the high memory $MemH$ which will ultimately store the high thresholds Thh, to be preloaded with a logic level 0. The initial period lasts for a total round of epochs from 1 to 8, hence lasts approximately $8 \cdot 1024$ samples in order to fully initialise the memories.

2. Once complete, the training period is set (Train = 1). In this case, the twin comparators are activated, one of which, compares the newly filtered synchronisation value $y(i)$, with the currently stored value in the memory $MemL$. If $y(i)$ is less than that in the memory it overwrites the current value in memory location $MemL$ by activating the tristate array T6, if false tristate array T7 is activated which rewires the output of the memory back to the input. Conversely, the second comparator checks to see whether $y(i)$ is greater than the current value in the memory $MemH$, if true tristate array T3 is activated, allowing for the new data to overwrite the data in location $MemH$, otherwise, tristate array T1 is activated re-storing the current value of $MemH$.

   Over a designated period of time, the memory arrays will slowly build the lowest and highest recorded values. We should note, that we initially pre-set the memories to 0 and 1, for memories high and low respectively, hence the probability that all values in the memories will initially be overwritten is high.

   When a suitable amount of data has successfully been gathered for the patient in question the true thresholds Thl and Thh, can be calculated.

Stage	MemL	MemH
Inital	$Min(y(i))$	$Max(y(i))$
Stage1	Δ	$MemH$
Stage2	Δ	Thh
Stage3	$Thh - 2\Delta$	Thh

Table 4.2 Table showing the memory allocation during calculation phase of the patient specific thresholds

Calculation

In order to reduce area occupation, we re-utilise the high and low memories and pipeline information. The calculations of the thresholds occur over three master clock cycles per epoch, these stages are named S1, S2 and S3. The information during each stage can be seen in table 4.2.

The stages are controlled via a simple 4 stage ring counter as seen in Fig. 4.10, where the fourth stage is a delay element needed for synchronising internal signals of the device. The memories in the training module are latched via a1L through a8L, the logic for latching can be seen in Fig. 4.10 and depends on the multiple stages being accessed.

1. Firstly, during S1, the value of Δ, is calculated, by activating T8 and subtracting the maximum and minimum values from training, hence making the calculation $MemH - MemL$. T4 is then activated which is a hard-wired tristate buffer array which mimics a shift left of 2^2, to create a final division by 4, to create Δ which is then stored in $MemL$. T1 is activated for $MemH$ keeping it constant.

2. During stage S2, Thh, is calculated and stored in $MemH$, by calculating $MemH-(\Delta)$. In this case T2 and T8 is activate for memory $MemH$ and T7 is activated for memory $MemL$, in order to preserve Δ.

3. Finally, during stage S3, Thl is calculated. In order to minimise hardware, we forgo the use of an additional adder which would normally be required to calculate equation 4.2 and re-utilise the subtracter such that $Thl = Thh - 2 \cdot \Delta$. Where, tristate T5 are activated to push the subtraction to memory $MemL$ and T9 is activated to perform a hard-wired shift right of 2^1, i.e, a multiplication of 2. The high memory maintains its value of Thh, once again by activating T1.

The processes of calculation run through all 8 epochs, hence lasting approximately $3 \cdot 1024$, samples. Once, the calculation mode has finished, a stable mode of synchronisation values can begin. At this point, the write enable to $MemL$ and $MemH$, are disabled in favour of read-only mode via the epoch selection signals a1 through a8.

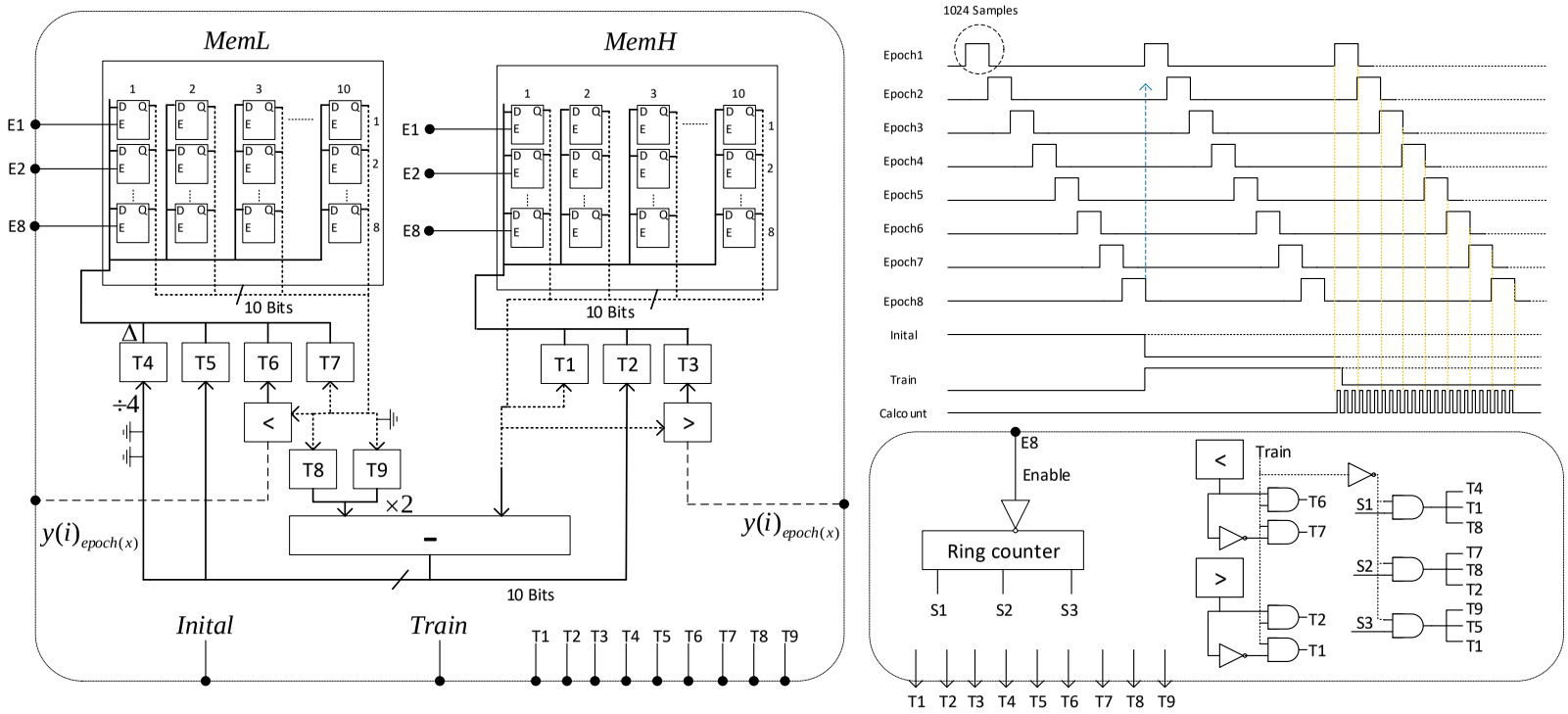

Fig. 4.9 Training and calculation module block diagram showing the twin high and low threshold memories and accompanied tristate pipeline arrays

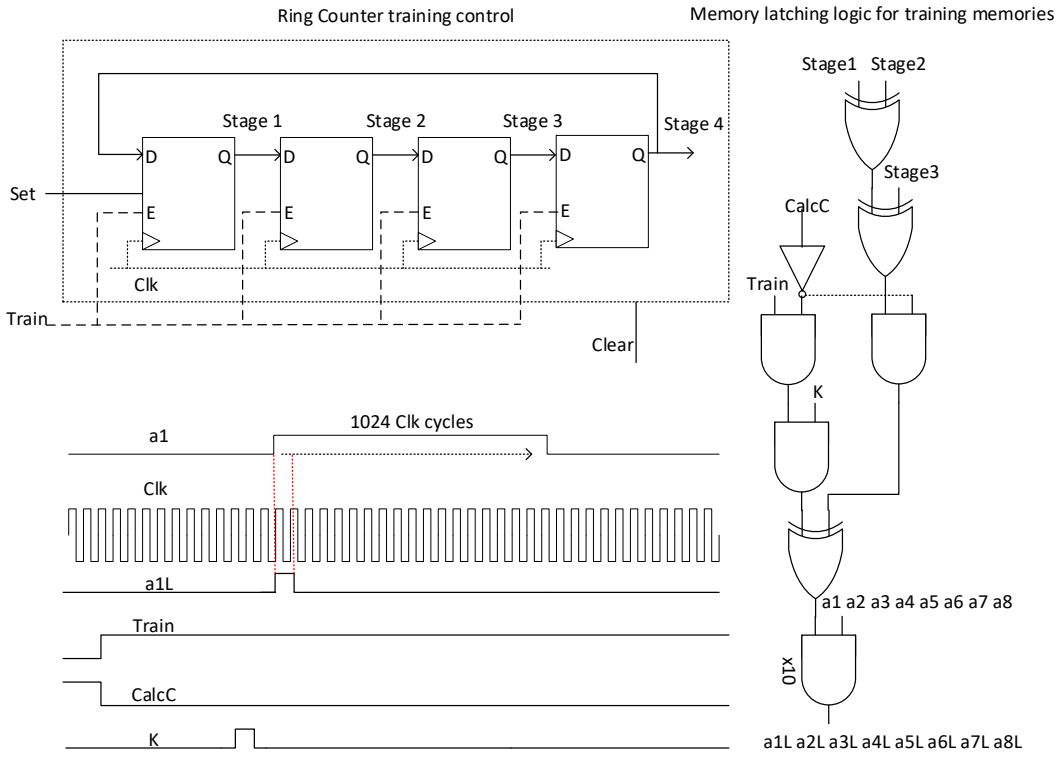

Fig. 4.10 Dedicated control logic overview for memory latching

Detection

The thresholds Thl and Thh held in the two memories are then feed into the first terminal of two 10 bit comparators C_l and C_h at each given epoch respectively. At the same time, the most current synchronisation value $\phi(i)$ is feed into the two comparators.

Comparator C_l checks if $\phi(i) < Thl$ and C_h checks if $\phi(i) > Thl$.

4.3 FPGA Implementation

In order to design the circuit in a VLSI environment, the design was first implemented and verified in the VIVADO design suit and eventually passed on to the ARTIC 7 FPGA used in section 3.3.1. An elaborated schematic view of the design can be seen in Fig. 4.11, which shows block diagrams of the tristate multiplexing arrays as well as the 15 synchronisation processors. To be architectural correct, the inputs and outputs

of the design incorporated an SPI and PSI, respectively. Whilst, Fig. 4.12 shows a schematic of the inner blocks which make up a single synchronisation processor, which are largely the same as in Fig. 3.8 in chapter 3, with the addition of the 3 memories and a training and calculation module.

Fig. 4.13, shows a VHDL simulation of the basic control unit signals a1 through a8, as well as the preloading of the high and low memory modules for training and calculation.

Fig. 4.14, Shows a VHDL simulation example of the threshold calculations. In this case the highlighted circles in red represent the calculation of *Thh(1)* and *Thl(1)*. Firstly, when the training is turned of and the calculation signal goes high. $max(y(i)) = 292$ and $min(y(i)) = 18$. In this case n, from equation 4.3 is set to 2, hence we get:

$$\Delta = \left[292 - 18\right] \cdot \frac{1}{4} = 68$$

which represents a quarter of the maximum voltage swing. Calculating Thh and Thl from here is easy.

$$Thh = 292 - 68 = 224$$

$$Thl = 224 - 2 \cdot 68 = 88$$

Fig. 4.15 shows an example of how the resulting outputs of the design change over various epochs for all of the 15 main processors. In this case, each vertical line represents the accumulated results gathered by the synchronisation processor during 1 epoch, where each consecutive line, represents epoch+1. To test the full range of possible combinations for the synchronisation processors, 16 synthetic signals were created. Each signal had a fixed sampling frequency of 1024 S/s, all odd numbered signals were assigned an increasing frequency chirp starting from the range 26 to 51Hz, and increasing by 1 every following signal. All of the even channels, however, were assigned a fixed sine wave of 37Hz.

As an example the signals introduced into processor P1 can be seen in table 4.3. Referring to table 4.1, we know that processor 1 receives signal combinations [1-2,1-3,1-4,1-5,1-6,1-7,1-8,1-9], over epochs 1 through 8 respectively.

Therefore, we can see a distinct triangular shaped output (identified by the red line) as the chirps of the odd signals converge towards the tone frequency (37Hz) of the stationary even signals. This effect can be seen in all processors, however, the tone

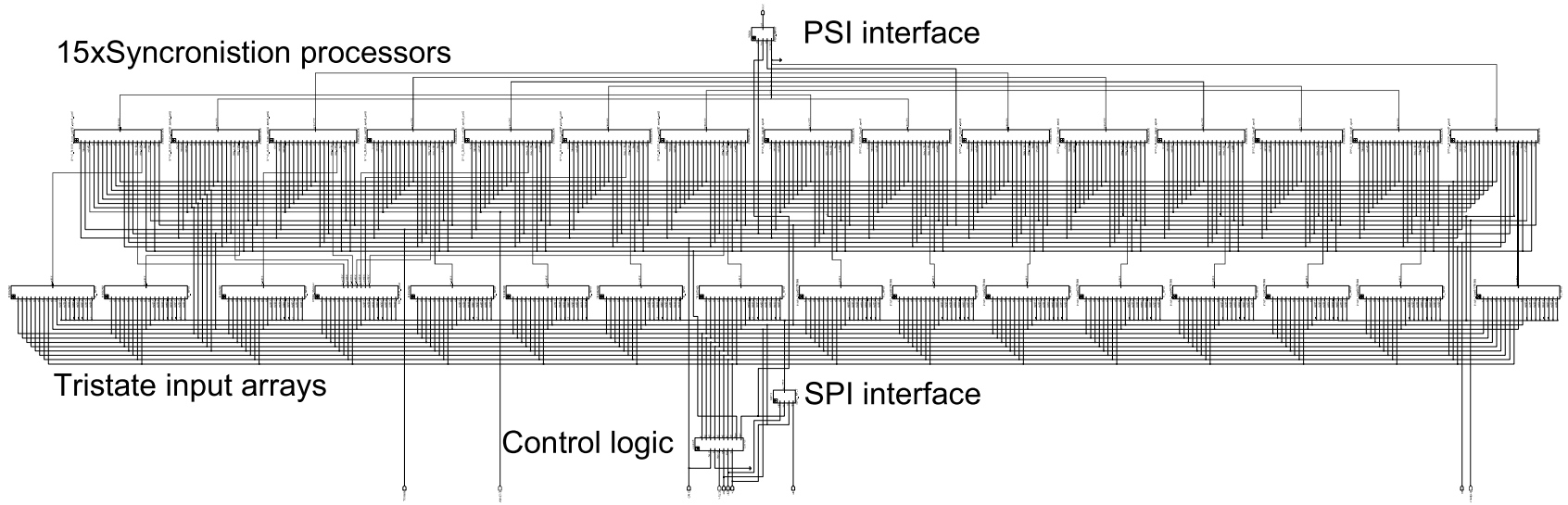

Fig. 4.11 Array of 15 synchronisation processors synthesised using VHDL code

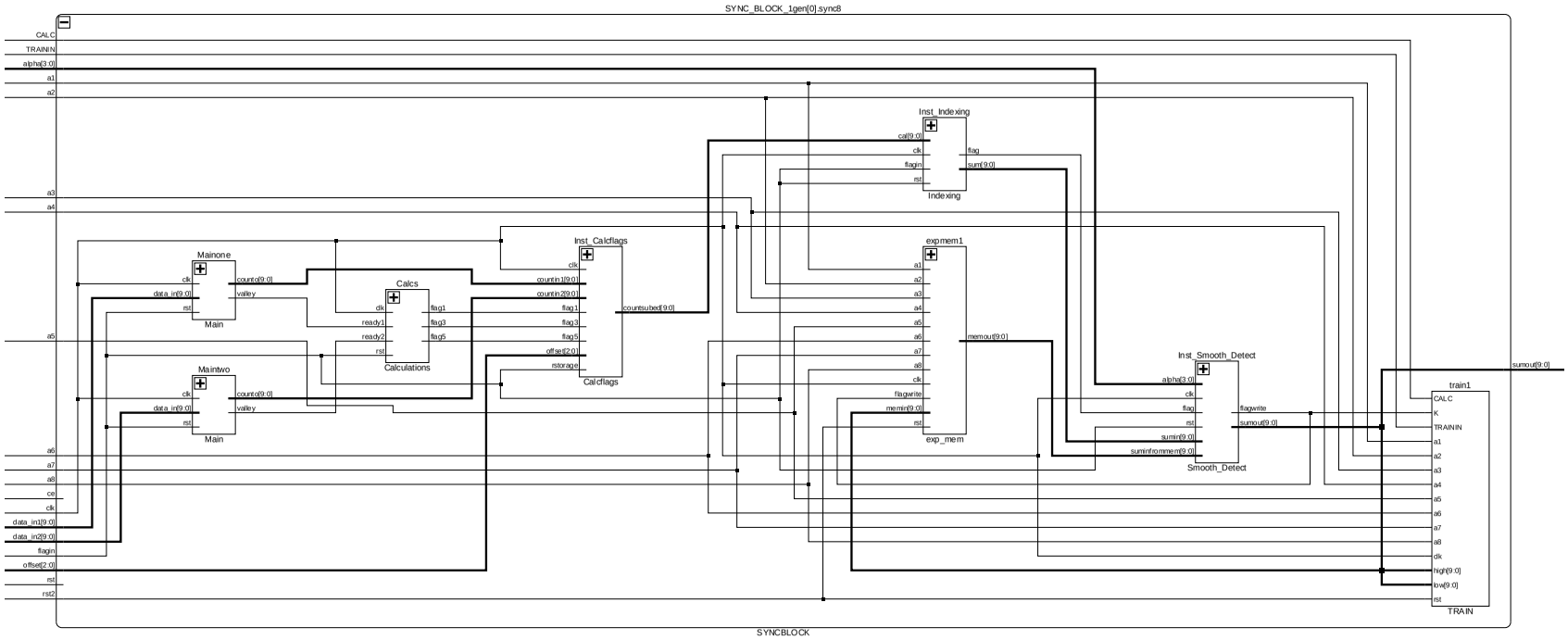

Fig. 4.12 Single synchronisation processor including all of the blocks from Fig. 3.8 in chapter 3 and an additional train and calculation block

Fig. 4.13 Simulation of the basic control logic and threshold memory storage

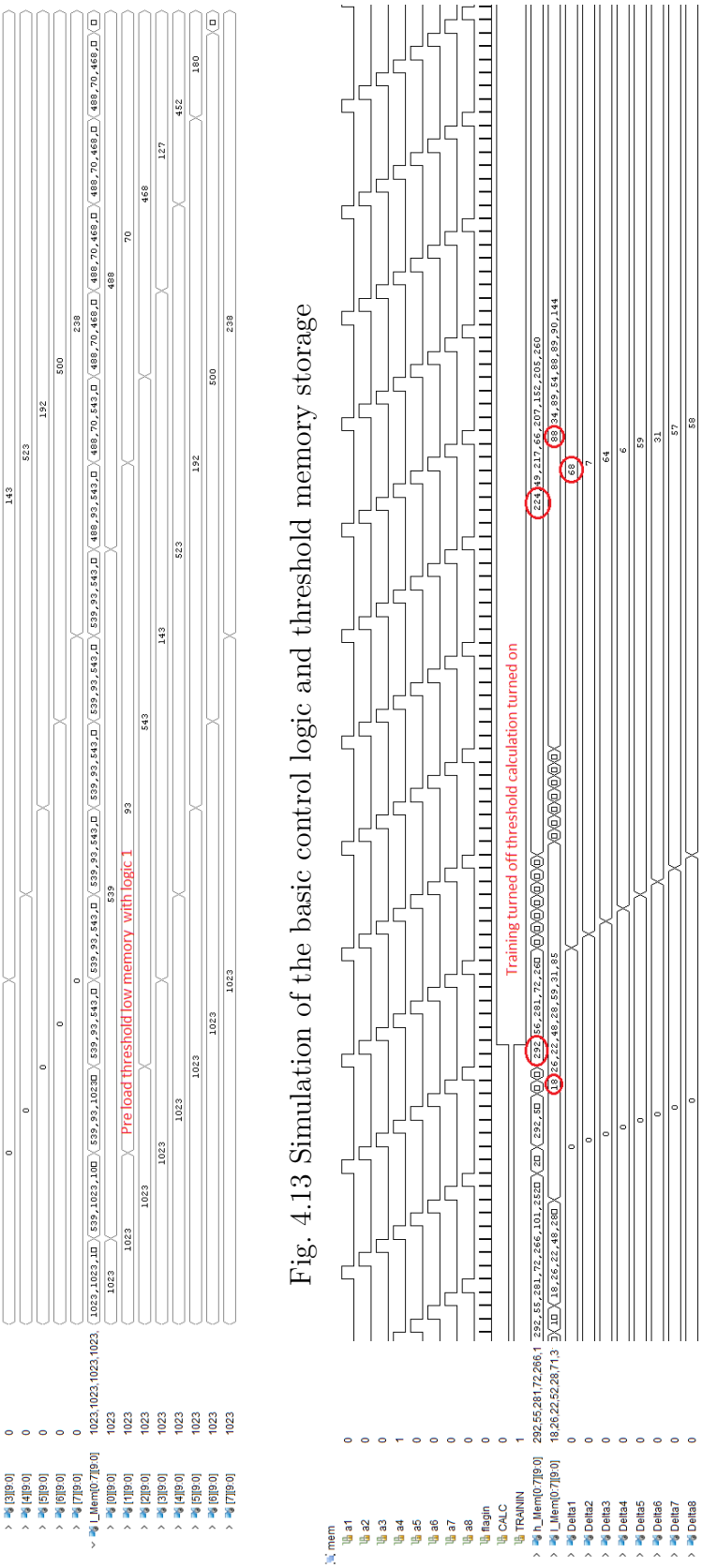

Fig. 4.14 Simulation of the basic calculation stage

Signal (P1)	Static or chirp	frequency range
1	chirp	26-51Hz
2	static	37Hz
3	chirp	27-52Hz
4	static	37Hz
5	chirp	28-53Hz
6	static	37Hz
7	chirp	29-54Hz
8	static	37Hz
9	chirp	30-55Hz

Table 4.3 Table showing the signals for processor 1 during VHDL simulation of 16-channel design

Resource	Utilisation
LUT's as logic	4035(3%)
Register as Flip Flop	4418(2%)
Input/output	25(9%)
Clocking	2(6%)

Table 4.4 Table showing all design resource allocation for the DDA hardware implementation.

frequencies converge at an earlier stage as the processors get higher due to the odd chirp signals increasing in frequency the higher the signal number.

As another example the first result for epoch 1 is 292 which corresponds to the synchronisation results for S1(26Hz) and S2(37Hz), since their frequencies are distant from one another we get a large value at the output. However, during epoch 2 signals 1(26Hz) and 3(27Hz) are compared hence producing a small value of 43.

Table 4.4, shows the resources used in the FPGA implementation of the 16-channel synchronisation processor.

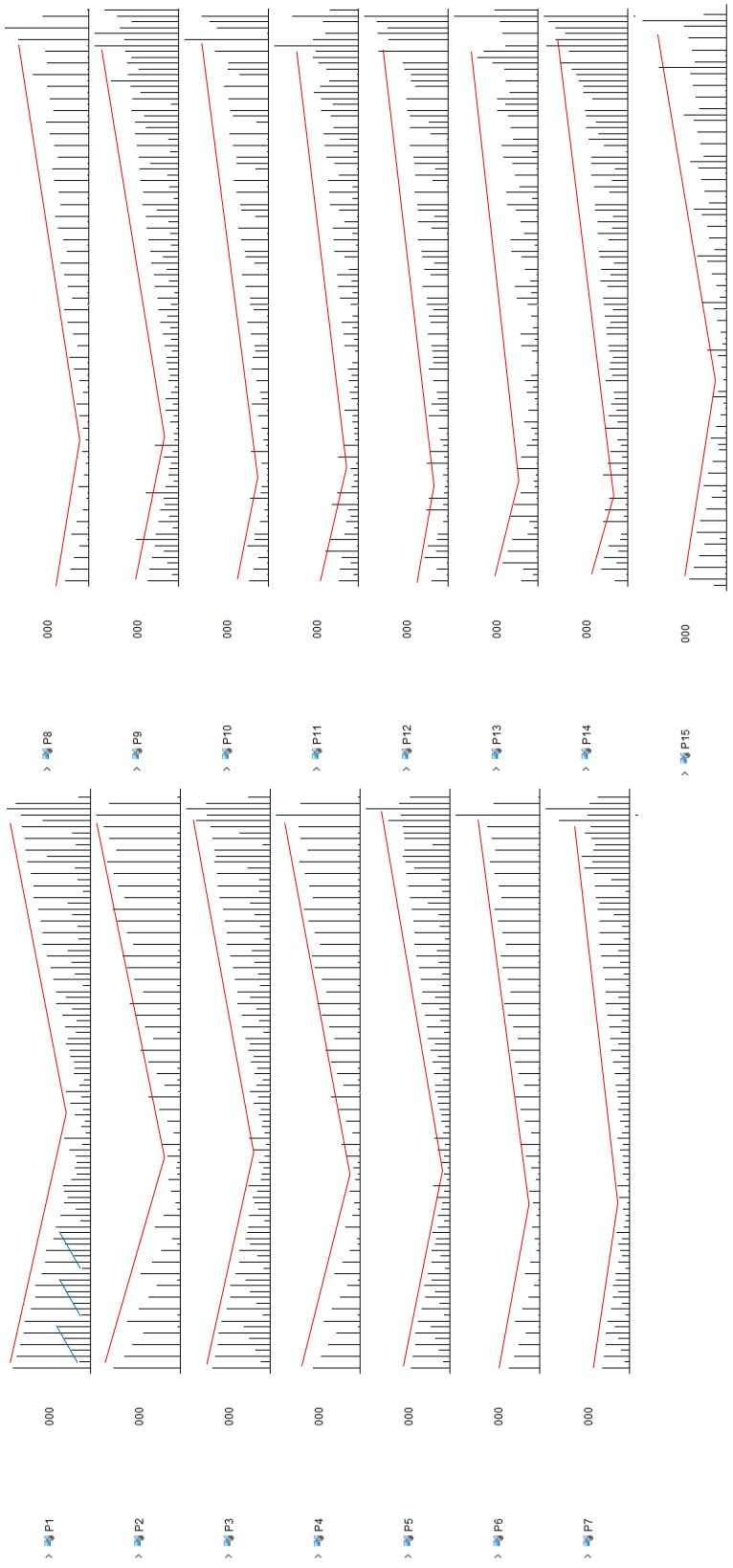

Fig. 4.15 Output results from VHDL simulation of 16-channel processor

4.4 Conclusions

In this chapter a 16-channel VLSI phase synchronisation processor was implemented The design only occupies $3.64mm^2$. Due to a change in the MPW policy of the selected foundry, AMS AG, regarding its 0.18μm CMOS process, the processor was received 13 months after the tape-out and finally it was not tested. Nevertheless, the design has been fully verified in an FPGA environment.

Chapter 5

Conclusions

5.1 Conclusions

In this thesis a new algorithm and synchronisation index for the detection of phase synchronisation between neural signals has been introduced. The algorithm shows a high correlation of approximately 90% when compared to more complex transform-based methods with only a slight decrease on noise performance. In addition, an exhaustive sensitivity analysis with human neural recordings shows that the algorithm produces average values of over 80% for FPR values of 0.6 and 0.75.

Two VLSI phase synchronisation processors have been introduced.

The first circuit, implemented with a custom 0.5V subthreshold digital library in a 0.18μm standard CMOS process, allows for the detection of phase synchronisation between two independent neural signals. The circuit occupies just 0.05 mm^2, including SPI AND PSO interfaces. The design consumes on average 15nW of power at 2KHz operating frequency which has been proven to be much lower than current state-of-the-art. In fact, when normalised to a 65nW technology, the design consumes just 0.0075μW per channel. That is a remarkable 20 times less power.

The functionality of the prototype has been extensively tested with neural recordings from an epilepsy database and its correct operation demonstrated both in seizure detection tasks and functional connectivity computation.

The second multi-channel processor, using the same library and technology node as above, is capable of calculating phase synchrony between 16 independent neural signals. The circuit occupies 3.64 mm^2 in total and incorporates novel patient-specific training and on-the-fly seizure detection mechanisms. The design has been implemented in FPGA and shows very small resource allocations of just above 4000 LUTs and 4400 registers.

En esta tesis se ha introducido un nuevo algoritmo e índice de sincronización para la detección de la sincronización de fase entre señales neuronales.

Los resultados del algoritmo propuesto mostraron una alta correlación de aproximadamente el 90% en comparación con el método PLV basado en la transformación más complejo.

También se demostró que cuando se realizó un análisis de SNR de ruido puntual en el algoritmo, se produjeron valores de correlación con una señal ideal libre de ruido superior a 90% para valores de SNR de 25 a 70 dB. Esto es solo un poco más bajo que el del método más complejo de HT + PLV.

Además, un fuerte análisis de sensibilidad absoluta mostró que el algoritmo produjo valores promedio de más del 80% para valores de FPR de 0,6 y 0,75. Increíblemente, se demostró que el algoritmo DDA superó el método más complejo de HT + PLV en todos los valores de FPR.

Además, se han introducido dos procesadores de sincronización de fase implementados por VLSI. El primero de ellos permitió detectar la sincronización de fase entre 2 señales neuronales independientes y tenía una frecuencia de operación principal de 2 KHz. El VLSI se implementó utilizando una biblioteca digital por debajo del umbral de 0.5V y una tecnología estándar de 0.18 μm.

La implementación de FPGA del diseño mostró que el diseño usaba hasta un 90% menos de recursos de FPGA en comparación con otras 2 implementaciones avanzadas.

El VLSI ocupaba solo 0.05 mm ^{2} que incluía interfaces SPI Y PSO. Además, cuando se normalizó en torno a una tecnología de 65 nm, el diseño ocupó solo 0,0033mm^2 por canal, que es 1,5 veces más pequeño que el diseño más pequeño del estado de la técnica.

El diseño consumió un promedio de 15 nW de potencia a 2 KHz de frecuencia de funcionamiento, que se ha demostrado que es mucho más bajo que el estado actual de la técnica; de hecho, cuando se normaliza a una tecnología de 65 nW, el diseño consume solo 0,0075μ W por canal. Eso es un notable 20 veces menos poder. Además, a través de una prueba de laboratorio, se demostró que el diseño puede operar tan solo a 0.3 voltios antes de que la corrupción de datos comience a infringir el circuito.

Los resultados tomados de la implementación VLSI a través de la configuración de prueba de laboratorio mostraron que el ASIC estaba funcionando correctamente, lo que se indica por un aumento repentino en la sincronización justo después del inicio de la crisis. Esto fue respaldado por la simulación del método HT + PLV que muestra un patrón muy similar.

Además, el análisis estadístico mostró altos aumentos en la sincronía alrededor del evento epiléptico en términos de conectividad funcional y análisis teórico de gráficos.

El segundo procesador multicanal fue capaz de calcular la sincronía de fase entre 16 señales neuronales independientes. Una vez más, se implementó utilizando una biblioteca digital por debajo del umbral de 0.5V y una tecnología estándar de 0.18μm.

El VLSI ocupó 3,64 mm^2 en total e incorpora el entrenamiento específico del paciente y el umbral de cálculo, así como la detección de la mosca.

Los umbrales específicos del paciente son de última generación y no se han integrado hasta este punto.

El diseño se implementó en FPGA y mostró asignaciones de recursos muy pequeñas de poco más de 4000 LUT y 4400 registros.

References

[1] Cold Spring Harbor Laboratory. 3D Brain. *DNA Learning Center, AXS Studio, Vivid Apps*, 2009. URL http://www.brainfacts.org/3d-brain#intro=truehttps://itunes.apple.com/nl/app/3d-brain/id331399332?mt=8.

[2] Wulfram Gerstner. Neuronal dynamics: From single neurons to networks and models of cognition, 2014. URL http://neuronaldynamics.epfl.ch/online/Ch1.S1.html.

[3] Khan Academy. Neuron action potentials: The creation of a brain signal (article) | Khan Academy. pages 1–13, 2017. URL https://www.khanacademy.org/test-prep/mcat/organ-systems/neuron-membrane-potentials/a/neuron-action-potentials-the-creation-of-a-brain-signal.

[4] 12.4 The Action Potential – Anatomy and Physiology. URL https://opentextbc.ca/anatomyandphysiology/chapter/12-4-the-action-potential/.

[5] Chemical and Electrical Synapses Two Kinds of Synapses. URL http://cbm.msoe.edu/markMyweb/ddtyResources/documents/synapseTypes.pdf.

[6] Tomas Ros, Bernard J Baars, Ruth A Lanius, and Patrik Vuilleumier. Tuning pathological brain oscillations with neurofeedback: a systems neuroscience framework. *Frontiers in human neuroscience*, 8:1008, 2014. ISSN 1662-5161. doi: 10.3389/fnhum.2014.01008. URL http://www.ncbi.nlm.nih.gov/pubmed/25566028http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4270171.

[7] R Caton. The Electric Currents of the Brain. *Brit Med J*, 2(1):278–, 1875. ISSN 09598138. doi: 10.1136/bmj.2.765.257. URL https://www.tandfonline.com/doi/abs/10.1080/00029238.1970.11080764.

[8] THOUGHT AND BRAIN: A CAMBRIDGE EXPERIMENT » 5 Oct 1934 » The Spectator Archive. URL http://archive.spectator.co.uk/article/5th-october-1934/10/thought-and-brain-a-cambridge-experiment.

[9] Fernando Lopes da Silva. EEG and MEG: Relevance to neuroscience, 12 2013. ISSN 08966273. URL http://www.ncbi.nlm.nih.gov/pubmed/24314724http://linkinghub.elsevier.com/retrieve/pii/S0896627313009203.

[10] Mikail Rubinov and Olaf Sporns. Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, 52(3):1059–1069, 9 2010. ISSN 1053-8119.

doi: 10.1016/J.NEUROIMAGE.2009.10.003. URL https://www.sciencedirect.com/science/article/pii/S105381190901074X.

[11] Robert W Thatcher and Robert W Thatcher. Coherence , Phase Differences , Phase Shift , and Phase Lock in EEG / ERP Analyses Coherence , Phase Differences , Phase Shift , and Phase Lock in EEG / ERP Analyses. 5641(March): 1–22, 2017. doi: 10.1080/87565641.2011.619241. URL http://www.appliedneuroscience.com/COHERENCE-DevNeuropsy-WEB.pdf.

[12] Michael Rosenblum, Arkady Pikovsky, Jürgen Kurths, C Schafer, and Peter A Tass. Phase synchronization: from theory to data analysis. *Handbook of Biological Physics*, 4(August 2001):279–321, 2003. doi: 10.1016/S1383-8121(01)80012-9. URL https://pdfs.semanticscholar.org/365a/269c80413afb0f46ffd55c58508180f4b2e9.pdfpapers://0cc5b277-5c1a-4a08-a78e-b6f9891a651e/Paper/p2553.

[13] Robson Scheffer-Teixeira and Adriano Bl Tort. On cross-frequency phase-phase coupling between theta and gamma oscillations in the hippocampus. *eLife*, 5(DECEMBER2016), 2016. ISSN 2050084X. doi: 10.7554/eLife.20515. URL http://www.ncbi.nlm.nih.gov/pubmed/27925581http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5199196.

[14] Tomislav Stankovski, Valentina Ticcinelli, Peter V. E. McClintock, and Aneta Stefanovska. Neural Cross-Frequency Coupling Functions. *Frontiers in Systems Neuroscience*, 11:33, 6 2017. ISSN 1662-5137. doi: 10.3389/fnsys.2017.00033. URL http://journal.frontiersin.org/article/10.3389/fnsys.2017.00033/full.

[15] Charles M. Gray, Peter König, Andreas K. Engel, and Wolf Singer. Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties. *Nature*, 338(6213):334–337, 3 1989. ISSN 0028-0836. doi: 10.1038/338334a0. URL http://www.nature.com/doifinder/10.1038/338334a0.

[16] Richard A Andersen, G K Essick, and Ralph M Siegel. Encoding of spatial location by posterior parietal neurons. *Science (New York, N.Y.)*, 230(4724):456–8, 10 1985. ISSN 0036-8075. doi: 10.1126/science.4048942. URL http://www.ncbi.nlm.nih.gov/pubmed/4048942http://www.sciencemag.org/cgi/doi/10.1126/science.4048942.

[17] Jan Mathijs Schoffelen, Robert Oostenveld, and Pascal Fries. Neuronal coherence as a mechanism of effective corticospinal interaction. *Science*, 308(5718):111–113, 4 2005. ISSN 00368075. doi: 10.1126/science.1107027. URL http://www.ncbi.nlm.nih.gov/pubmed/15802603.

[18] Pascal Fries. A mechanism for cognitive dynamics: Neuronal communication through neuronal coherence, 2005. ISSN 13646613. URL https://4509dc21-a-62cb3a1a-s-sites.googlegroups.com/site/cnbctheory/oscillations-synchrony/Fries2005p879-Amechanismforcognitivedynamicsneuronalcommunicationthroughneuronalcoherence.pdf?attachauth=ANoY7cqYFxX0r3rHeF2Y_QKmLDSHVtLm4gzzpQvsg5WLDoVAu0b9m3RGUD.

[19] Ole Jensen and Ali Mazaheri. Shaping Functional Architecture by Oscillatory Alpha Activity: Gating by Inhibition. *Frontiers in Human Neuroscience*, 4:186, 2010. ISSN 1662-5161. doi: 10.3389/fnhum.2010.00186. URL

http://www.ncbi.nlm.nih.gov/pubmed/21119777http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2990626http://journal.frontiersin.org/article/10.3389/fnhum.2010.00186/abstract.

[20] Peter J. Uhlhaas and Wolf Singer. Neural Synchrony in Brain Disorders: Relevance for Cognitive Dysfunctions and Pathophysiology, 10 2006. ISSN 08966273. URL https://www.sciencedirect.com/science/article/pii/S0896627306007276.

[21] C. J. Stam, Y. Van Der Made, Y. A. L. Pijnenburg, and Ph. Scheltens. EEG synchronization in mild cognitive impairment and Alzheimer's disease. *Acta Neurologica Scandinavica*, 108(2):90–96, 8 2003. ISSN 00016314. doi: 10.1034/j.1600-0404.2003.02067.x. URL http://doi.wiley.com/10.1034/j.1600-0404.2003.02067.x.

[22] C. J. Stam, T. Montez, B. F. Jones, S. A.R.B. Rombouts, Y. Van Der Made, Y. A.L. Pijnenburg, and Ph. Scheltens. Disturbed fluctuations of resting state EEG synchronization in Alzheimer's disease. *Clinical Neurophysiology*, 116(3):708–715, 3 2005. ISSN 13882457. doi: 10.1016/j.clinph.2004.09.022. URL https://www.sciencedirect.com/science/article/pii/S1388245704003797?via%3Dihub.

[23] C. J. Stam, B. F. Jones, G Nolte, M Breakspear, and Ph Scheltens. Small-world networks and functional connectivity in Alzheimer's disease. *Cerebral Cortex*, 17(1):92–99, 2 2007. ISSN 10473211. doi: 10.1093/cercor/bhj127. URL https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhj127.

[24] Mahdi Jalili. Graph theoretical analysis of Alzheimer's disease: Discrimination of AD patients from healthy subjects. *Information Sciences*, 384:145–156, 4 2017. ISSN 00200255. doi: 10.1016/j.ins.2016.08.047. URL https://www.sciencedirect.com/science/article/pii/S0020025516306193.

[25] C L Grady. Altered brain functional connectivity and impaired short-term memory in Alzheimer's disease. *Brain*, 124(4):739–756, 4 2001. ISSN 14602156. doi: 10.1093/brain/124.4.739. URL http://www.ncbi.nlm.nih.gov/pubmed/11287374https://academic.oup.com/brain/article-lookup/doi/10.1093/brain/124.4.739.

[26] V N Murthy and E E Fetz. Oscillatory activity in sensorimotor cortex of awake monkeys: synchronization of local field potentials and relation to behavior. *Journal of Neurophysiology*, 76(6):3949–3967, 12 1996. ISSN 0022-3077. doi: 10.1152/jn.1996.76.6.3949. URL http://www.ncbi.nlm.nih.gov/pubmed/8985892.

[27] Jan Mathijs Schoffelen, Robert Oostenveld, and Pascal Fries. Neuronal coherence as a mechanism of effective corticospinal interaction. *Science*, 308(5718):111–113, 4 2005. ISSN 00368075. doi: 10.1126/science.1107027. URL http://www.ncbi.nlm.nih.gov/pubmed/15802603.

[28] Thomas Boraud, Peter Brown, Joshua A Goldberg, Ann M Graybiel, and Peter J Magill. Oscillations in the basal ganglia: the good, the bad, and the unexpected. In *The basal ganglia VIII*, pages 1–24. Springerb, 2005.

[29] Peter Brown. Oscillatory nature of human basal ganglia activity: Relationship to the pathophysiology of parkinson's disease, 4 2003. ISSN 08853185. URL http://doi.wiley.com/10.1002/mds.10358.

[30] J. P. Ginsberg. Setting domain boundaries for convergence of biological and psychological perspectives on cognitive coordination in schizophrenia, 2 2003. ISSN 0140525X. URL http://www.ncbi.nlm.nih.gov/pubmed/14598440.

[31] S. Rampp and H. Stefan. Fast activity as a surrogate marker of epileptic network function?, 10 2006. ISSN 13882457. URL https://www.sciencedirect.com/science/article/pii/S1388245706001131?via%3Dihub.

[32] Theoden I Netoff and Steven J Schiff. Decreased neuronal synchronization during experimental seizures. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 22(16):7297–7307, 8 2002. ISSN 1529-2401. doi: 20026711. URL http://www.ncbi.nlm.nih.gov/pubmed/12177225.

[33] Michel Le Van Quyen, Vincent Navarro, Jacques Martinerie, Michel Baulac, and Francisco J Varela. Toward a neurodynamical understanding of ictogenesis. *Epilepsia*, 44 Suppl 1:30–43, 2003. ISSN 0013-9580. doi: 12007[pii]. URL http://www.ncbi.nlm.nih.gov/pubmed/14641559.

[34] Florian Mormann, Klaus Lehnertz, Peter David, and Christian E. Elger. Mean phase coherence as a measure for phase synchronization and its application to the EEG of epilepsy patients. *Physica D: Nonlinear Phenomena*, 144(3): 358–369, 10 2000. ISSN 01672789. doi: 10.1016/S0167-2789(00)00087-7. URL https://www.sciencedirect.com/science/article/pii/S0167278900000877.

[35] Meenakshi Aggarwal, Richa Barsainya, and T.K. Rawat. FPGA implementation of Hilbert transformer based on lattice wave digital filters. In *2015 4th International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2015*, pages 1–5. IEEE, 9 2015. ISBN 9781467372312. doi: 10.1109/ICRITO.2015.7359331. URL http://ieeexplore.ieee.org/document/7359331/.

[36] V. Sakkalis. Review of advanced techniques for the estimation of brain connectivity measured with EEG/MEG. *Computers in Biology and Medicine*, 41(12):1110–1117, 12 2011. ISSN 00104825. doi: 10.1016/j.compbiomed.2011.06.020. URL https://www.sciencedirect.com/science/article/pii/S0010482511001405.

[37] Andrea Pigorini, Adenauer G. Casali, Silvia Casarotto, Fabio Ferrarelli, Giuseppe Baselli, Maurizio Mariotti, Marcello Massimini, and Mario Rosanova. Time-frequency spectral analysis of TMS-evoked EEG oscillations by means of Hilbert-Huang transform. *Journal of Neuroscience Methods*, 198(2):236–245, 2011. ISSN 01650270. doi: 10.1016/j.jneumeth.2011.04.013.

[38] Turkey N Alotaiby, Saleh A Alshebeili, Tariq Alshawi, Ishtiaq Ahmad, and Fathi E Abd El-Samie. EEG seizure detection and prediction algorithms: a survey, 2014. ISSN 16876180. URL https://link.springer.com/content/pdf/10.1186%2F1687-6180-2014-183.pdf.

[39] Jean Philippe Lachaux, Eugenio Rodriguez, Jacques Martinerie, and Francisco J Varela. Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8 (4):194–208, 1999. ISSN 10659471. doi: 10.1002/(SICI)1097-0193(1999)8:4<194::AID-HBM4>3.0.CO;2-C. URL http://www.ncbi.nlm.nih.gov/pubmed/10619414.

[40] Michel Le Van Quyen, Jack Foucher, Jean-Philippe Lachaux, Eugenio Rodriguez, Antoine Lutz, Jacques Martinerie, and Francisco J. Varela. Comparison of Hilbert transform and wavelet methods for the analysis of neuronal synchrony. *Journal of neuroscience methods*, 111(2):83–98, 9 2001. ISSN 0165-0270. doi: http://dx.doi.org/10.1016/S0165-0270(01)00372-7. URL https://www.sciencedirect.com/science/article/pii/S0165027001003727.

[41] Jürgen Kurths, M. Carmen Romano, Marco Thiel, Grigory V. Osipov, Mikhail V. Ivanchenko, István Z. Kiss, and John L. Hudson. Synchronization analysis of coupled noncoherent oscillators. In *Nonlinear Dynamics*, volume 44, pages 135–149. Kluwer Academic Publishers, 6 2006. ISBN 0924-090X. doi: 10.1007/s11071-006-1957-x. URL http://link.springer.com/10.1007/s11071-006-1957-x.

[42] D. Rangaprakash. Connectivity analysis of multichannel EEG signals using recurrence based phase synchronization technique. *Computers in Biology and Medicine*, 46(1):11–21, 3 2014. ISSN 00104825. doi: 10.1016/j.compbiomed.2013.10.025. URL https://www.sciencedirect.com/science/article/pii/S0010482513003144.

[43] Hoda Rajaei, Mercedes Cabrerizo, Panuwat Janwattanapong, Alberto Pinzon, Sergio Gonzalez-Arias, Armando Barreto, and Malek Adjouadi. Connectivity Dynamics of Interictal Epileptiform Activity. In *2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 425–430. IEEE, 10 2017. ISBN 978-1-5386-1324-5. doi: 10.1109/BIBE.2017.00-17. URL http://ieeexplore.ieee.org/document/8251327/.

[44] P. Tass, M. G. Rosenblum, J. Weule, J. Kurths, A. Pikovsky, J. Volkmann, A. Schnitzler, and H. J. Freund. Detection of n:m phase locking from noisy data: Application to magnetoencephalography. *Physical Review Letters*, 81(15):3291–3294, 10 1998. ISSN 10797114. doi: 10.1103/PhysRevLett.81.3291. URL https://link.aps.org/doi/10.1103/PhysRevLett.81.3291.

[45] Lauren J O'Donnell and Carl Fredrik Westin. An introduction to diffusion tensor image analysis, 4 2011. ISSN 10423680. URL http://www.ncbi.nlm.nih.gov/pubmed/21435570http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3163395.

[46] Susumu Mori and Jiangyang Zhang. Principles of Diffusion Tensor Imaging and Its Applications to Basic Neuroscience Research. *Neuron*, 51(5):527–539, 2006. ISSN 08966273. doi: 10.1016/j.neuron.2006.08.012.

[47] Venkateswaran Rajagopalan, Zhiguo Jiang, Guang H Yue, Jelena Stojanovic Radic, Erik P Pioro, Glenn R Wylie, and Abhijit Das. A Basic Introduction to Diffusion Tensor Imaging Mathematics and Image Processing Steps. *Brain Disorders & Therapy*, 06(02), 2017. ISSN 2168975X. doi: 10.4172/2168-975X.1000229. URL https://www.omicsonline.org/open-access/a-basic-introduction-to-diffusion-tensor-imaging-mathematics-and-imageprocessing-steps-2168-975X-1000229.pdfhttps://www.omicsgroup.org/journals/a-basic-introduction-to-diffusion-tensor-imaging-mathematics-and-image.

[48] Tong Wu, Jian Xu, Yong Lian, Azam Khalili, Amir Rastegarnia, Cuntai Guan, and Zhi Yang. A 16-Channel Nonparametric Spike Detection ASIC Based on

EC-PC Decomposition. *IEEE Transactions on Biomedical Circuits and Systems*, 10(1):3–17, 2 2016. ISSN 19324545. doi: 10.1109/TBCAS.2015.2389266. URL http://www.ncbi.nlm.nih.gov/pubmed/25769170.

[49] Karim Abdelhalim, Hamed Mazhab Jafari, Larysa Kokarovtseva, Jose Luis Perez Velazquez, and Roman Genov. 64-Channel UWB Wireless Neural Vector Analyzer SOC With a Closed-Loop Phase Synchrony-Triggered Neurostimulator. *IEEE Journal of Solid-State Circuits*, 48(10):2494–2510, 10 2013. ISSN 0018-9200. doi: 10.1109/JSSC.2013.2272952. URL http://ieeexplore.ieee.org/document/6572893/.

[50] Hossein Kassiri, Muhammad Tariqus Salam, Mohammad Reza Pazhouhandeh, Nima Soltani, Jose Luis Perez Velazquez, Peter Carlen, and Roman Genov. Rail-to-rail-input dual-radio 64-channel closed-loop neurostimulator. *IEEE Journal of Solid-State Circuits*, 52(11):2793–2810, 2017. ISSN 00189200. doi: 10.1109/JSSC.2017.2749426. URL http://ieeexplore.ieee.org/document/8068946/.

[51] J. Aziz, R. Karakiewicz, R. Genov, A.W.L. Chiu, B.L. Bardakjian, M. Derchansky, and P.L. Carlen. In Vitro Epileptic Seizure Prediction Microsystem. *2007 IEEE International Symposium on Circuits and Systems*, pages 3115–3118, 5 2007. ISSN 02714310. doi: 10.1109/ISCAS.2007.378090. URL http://ieeexplore.ieee.org/document/4253338/.

[52] Karim Abdelhalim, Vadim Smolyakov, and Roman Genov. Phase-synchronization early epileptic seizure detector VLSI architecture. In *IEEE Transactions on Biomedical Circuits and Systems*, volume 5, pages 430–438, 10 2011. doi: 10.1109/TBCAS.2011.2170686. URL http://ieeexplore.ieee.org/document/6046232/.

[53] Yu Hsin Chen, Tung Chien Chen, Tsung Hsueh Lee, and Liang Gee Chen. Sub-microwatt correlation integral processor for implantable closed-loop epileptic neuromodulator. In *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, pages 2083–2086. IEEE, 5 2010. ISBN 9781424453085. doi: 10.1109/ISCAS.2010.5537220. URL http://ieeexplore.ieee.org/document/5537220/.

[54] V Karkare, S Gibson, and D Markovic. A 130- uW, 64-Channel Neural Spike-Sorting DSP Chip. *Solid-State Circuits, IEEE Journal of*, 46(5):1214–1222, 5 2011.

[55] Nai Fu Chang, Tung Chien Chen, Cheng Yi Chiang, and Liang Gee Chen. On-line empirical mode decomposition biomedical microprocessor for Hilbert Huang transform. In *2011 IEEE Biomedical Circuits and Systems Conference, BioCAS 2011*, pages 420–423. IEEE, 11 2011. ISBN 9781457714696. doi: 10.1109/BioCAS.2011.6107817. URL http://ieeexplore.ieee.org/document/6107817/.

[56] R. Quian Quiroga, A. Kraskov, T. Kreuz, and P. Grassberger. Performance of different synchronization measures in real data: A case study on electroencephalographic signals. *Physical Review E*, 65(4):041903, 3 2002. ISSN 1063-651X. doi: 10.1103/PhysRevE.65.041903. URL https://link.aps.org/doi/10.1103/PhysRevE.65.041903.

[57] Juliane Klatt, Hinnerk Feldwisch-Drentrup, Matthias Ihle, Vincent Navarro, Markus Neufang, Cesar Teixeira, Claude Adam, Mario Valderrama, Catalina Alvarado-Rojas, Adrien Witon, Michel Le Van Quyen, Francisco Sales, Antonio Dourado, Jens Timmer, Andreas Schulze-Bonhage, and Bjoern Schelter. The EPILEPSIAE database: An extensive electroencephalography database of epilepsy patients. *Epilepsia*, 53(9):1669–1676, 9 2012. ISSN 00139580. doi: 10.1111/j.1528-1167.2012.03564.x. URL http://www.ncbi.nlm.nih.gov/pubmed/22738131http://doi.wiley.com/10.1111/j.1528-1167.2012.03564.x.

[58] S. J. van Albada and P. A. Robinson. Relationships between Electroencephalographic Spectral Peaks Across Frequency Bands. *Frontiers in Human Neuroscience*, 7(March):1–18, 2013. ISSN 1662-5161. doi: 10.3389/fnhum.2013.00056. URL http://journal.frontiersin.org/article/10.3389/fnhum.2013.00056/abstract.

[59] Kai J. Miller, Larry B. Sorensen, Jeffrey G. Ojemann, and Marcel Den Nijs. Power-law scaling in the brain surface electric potential. *PLoS Computational Biology*, 5(12), 2009. ISSN 1553734X. doi: 10.1371/journal.pcbi.1000609.

[60] R R Harrison and C Charles. A low-power low-noise CMOS amplifier for neural recording applications. *Solid-State Circuits, IEEE Journal of*, 38(6):958–965, 2003.

[61] A T Gardner, H J Strathman, D J Warren, and R M Walker. Impedance and Noise Characterizations of Utah and Microwire Electrode Arrays. *IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology*, 2(4):234–241, 12 2018. ISSN 2469-7249. doi: 10.1109/JERM.2018.2862417.

[62] William Smith, Brian Mogen, Eberhard E Fetz, Visvesh S Sathe, and Brian P Otis. Exploiting Electrocorticographic Spectral Characteristics for Optimized Signal Chain Design: A 1.08 μW Analog Front End With Reduced ADC Resolution Requirements. *IEEE transactions on biomedical circuits and systems*, 10, 2016. doi: 10.1109/TBCAS.2016.2518923.

[63] Florian Mormann, Thomas Kreuz, Christoph Rieke, Ralph G. Andrzejak, Alexander Kraskov, Peter David, Christian E. Elger, and Klaus Lehnertz. On the predictability of epileptic seizures. *Clinical Neurophysiology*, 116(3):569–587, 3 2005. ISSN 13882457. doi: 10.1016/j.clinph.2004.08.025. URL https://www.sciencedirect.com/science/article/pii/S1388245704004638?via%3Dihub.

[64] Armin Tajalli, Elizabeth J. Brauer, Yusuf Leblebici, and Eric Vittoz. Subthreshold Source-Coupled Logic Circuits for Ultra-Low-Power Applications. *IEEE Journal of Solid-State Circuits*, 43(7):1699–1710, 7 2008. ISSN 0018-9200. doi: 10.1109/JSSC.2008.922709. URL http://ieeexplore.ieee.org/document/4550646/.

[65] Jonathan D Jones and Monte Tull. Embedded Hilbert Transform-Based Algorithm within a Field Programmable Gate Array to Classify Nonlinear SDOF Systems. ISSN 21915644. URL https://pdfs.semanticscholar.org/65d8/9067e284e08250fed4c3d2d1caac79ee6b70.pdf.

[66] Anupama Patil and Ritu Saini. Design and implementation of FPGA based linear all digital phase-locked loop. In *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, pages

1–1, 2014. ISBN 978-1-4799-4982-3. doi: 10.1109/ICGCCEE.2014.6922342. URL https://pdfs.semanticscholar.org/23f6/05ebba13b1ac014bbf3e6d737200ae3f5c15.pdfhttp://ieeexplore.ieee.org/document/6922342/.

[67] Florian Mormann, Thomas Kreuz, Ralph G Andrzejak, Peter David, Klaus Lehnertz, and Christian E Elger. Epileptic seizures are preceded by a decrease in synchronization. *Epilepsy Research*, 53(3):173–185, 2003. ISSN 09201211. doi: 10.1016/S0920-1211(03)00002-0. URL https://pdfs.semanticscholar.org/6506/728a434aa128ac9961c0104dd5914c2848f6.pdf.

[68] Sparsh Mittal. A Survey of Techniques for Approximate Computing. *ACM Computing Surveys*, 48(4):1–33, 3 2016. ISSN 03600300. doi: 10.1145/2893356. URL http://dl.acm.org/citation.cfm?doid=2891449.2893356.

[69] Sigmund Jenssen, Edward J. Gracely, and Michael R. Sperling. How long do most seizures last? A systematic comparison of seizures recorded in the epilepsy monitoring unit. *Epilepsia*, 47(9):1499–1503, 9 2006. ISSN 00139580. doi: 10.1111/j.1528-1167.2006.00622.x. URL http://doi.wiley.com/10.1111/j.1528-1167.2006.00622.x.

[70] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736.

[71] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Publishing Group*, 10(3):186–198, 3 2009. ISSN 1471-0048. doi: 10.1038/nrn2575. URL http://www.nature.com/articles/nrn2575http://www.ncbi.nlm.nih.gov/pubmed/19190637.

[72] Hunter C Gits. Relating Connectivity and Graph Analysis to Cognitive Function in Alzheimer's Disease. pages 45–65, 2016. ISSN 2470-9727. doi: http://dx.doi.org/10.3998/mjm.13761231.0001.111.

[73] Bernadette C. M. van Wijk, Cornelis J. Stam, and Andreas Daffertshofer. Comparing brain networks of different size and connectivity density using graph theory. *PLoS ONE*, 5(10):e13701, 10 2010. ISSN 19326203. doi: 10.1371/journal.pone.0013701. URL http://dx.plos.org/10.1371/journal.pone.0013701.

[74] Duncan J. Watts and Steven H. Strogatz. Collectivedynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998. ISSN 00280836. doi: 10.1038/30918. URL http://www.nature.com/nature/journal/v393/n6684/abs/393440a0.html%5Cnhttp://www.nature.com/nature/journal/v393/n6684/pdf/393440a0.pdf%5Cnhttp://www.nature.com/doifinder/10.1038/30918.

Chapter A

Graph Theory concepts

Graph theory is common mathematical approach which provides interesting metrics for the determination of networks. The applications for graph theory have a strong hold in many disciplines including computer science, biology and sociology to name a few. A historical mathematical problem named *Seven bridges of Konigsberg* written by Leonhard Euler in 1736, is considered to be the first introduction to graphs and topologies, where Euler proved using graphs that it was impossible to traverse the city of Königsberg exactly once and return to the starting point using seven different interconnected bridges [70].

In neuroscience graph theory is used as a tool to further identify underlying characteristics of the brains structural, functional and effective connectivity.

Figure A.1, shows an overview of some of the underlying concepts behind graph theory. Firstly, every object in the network is assigned as a node (identified as a grey dot). Where each node in the network are connected via edges, which are the abstract connections between the nodes. As an example, in a typical EEG/MEG scalp recording set-up the electrodes would be considered the nodes and all the interconnections between the electrodes would be considered the edges of the network. In the 10-20 system this would typically consist of 16 nodes and 120 edges (excluding isolated nodes).

In general, graph networks are complex networks which should follow the following logical steps[71, 72]:

1. Define all nodes in the network: As stated above, the definition of the nodes could be in the form of EEG/MEG electrodes or as anatomically defined regions via the interpretation of MRI or DTI data.

2. The second step, is to identify a continuous measure of association between nodes. This is usually done between all combinations of nodes. This continuous measure could be time based correlation, spectral analysis or DTI analysis.

3. The next step is to derive an 'association matrix', M, which consists of a square NxN, map of all the continuous measures from the previous step.

4. The next step is to derive an 'adjacency matrix', A, from the association matrix. These can be constructed in several ways depending on the application intended. Nevertheless, the most common A, is a binary matrix, in which a static or dynamically varying threshold is applied the M. In this case anything above the threshold is set to 1, conversley, anything below is set to 0, this produces a symmetrical A matrix filled with zeros and ones.

5. The final step in the process is to conduct analysis on A, using graph theoretical based metrics to build information maps on the network.

Other types of networks do exist beyond the non-weighted non-directional network, such as weighted/directed graphs, binary/directed graphs, however functional connectivity is usually set as binary/undirected graphs.

In general there are three main desirable metrics which can prove useful to the identification of structural and functional connectivity in brain networks[73]:

1. **Degree**

   The degree k, is a metric which identifies the number of edges connected to a given node such that:

$$k_i = \sum_{i}^{N} A_{i,j} \tag{A.1}$$

   Where A, is the adjacency matrix and i and j are the row and coulomb in the matrix respectively. In the case of binary threshold matrices, this gives a clear indication of how heavily each node in the network is connected with other nodes.

   More often than not the averaging of the edges per node is calculated such that:

$$\langle k \rangle = \frac{1}{N} \sum_{i}^{N} k_i \tag{A.2}$$

This gives an overall value for the brains connectivity patterns (i,e) the total number of nodes which passed the threshold. This metric is especially useful for time varying analysis and dynamic thresholds as time lapse maps can be built.

2. **Path length**

   The average path length L, indicates the minimum number of edges between any two nodes within a network, using global efficiency E to account for isolated nodes. The path length indicates how well information or other entities in the network can be transferred from one region to another. Given as:

$$E = \frac{1}{N(N-1)} \sum_{i,j,i \neq j}^{N} \frac{1}{d_{i,j}} \tag{A.3}$$

$$L = \frac{1}{E} \tag{A.4}$$

   Where, d, is the distance between two nodes.

3. **Clustering coefficient**

   The clustering coefficient C, measures the occurrence of clusters in a given network by showing the probability that a nodes neighbours are also connected. This measure is an index of local structure and is used as a resilience factor, i.e if a node is lost (doesn't pass threshold), what's the probability that its nearest neighbours are still connected.

$$C = \frac{1}{N} \sum_{i}^{N} \frac{2n_i}{k_i(k_i - 1)} \tag{A.5}$$

   Where n_i, represents the number of edges between the neighbours of node i.

4. **Edge density**

   The edge density m, identifies the fraction of existing edges in a network out of the total number of edges in the network:

$$m = \frac{1}{N(N-1)} \sum_{i,j}^{N} A_{i,j} \tag{A.6}$$

figure A.1, shows an example of how graph theory can be applied to a network. In this case the network (shown in the upper left hand corner), comprises of 5 nodes, A,B,C,D and E.

To calculate the average nodal degree of the network the mean of the total number of edges per node is taken. In this case, $A = 3$, as it is connected to 3 other nodes via edges, $B = 4$, etc... To grade the overall network the average is taken such that: $avg(k_i) = \frac{1}{5} \cdot (3 + 4 + 2 + 2 + 1)$, giving a mean total of 2.4.

The average minimum distance between any two nodes is such that: $E = \frac{1}{5 \cdot 4} \cdot (3.5 + 4 + 3 + 3 + 2.5)$, giving a mean total of 0.8, this is known as the global efficiency. The Path length is calculated as $L = \frac{1}{0.8} = 1.25$.

As an example in the case of the distance between nodes A and E, there exists a shortest route of 2 edges, therefore, $\frac{1}{d_{i,j}} = \frac{1}{2} = 0.5$.

For the clustering coefficient node $A = \frac{2 \cdot 2}{3 \cdot (3-1)} = \frac{1}{3}$, the denominator signifies the amount of edges connected to the node A, whereas the numerator signifies the number of connections made between other nodes connected to these edges. In A the numerator $n_i = 2$, since C is connected to B and D is also connected to B.

In the case that isolated nodes are located such as node E, the clustering coefficient is set to 0, because the denominator $k_i(k_i - 1)$, is set to 0, and is hence undefined.

From the metrics above it is possible to construct special networks topologies which derive an association to brain connectivity and can be seen in figure A.1. These are,ordered, small-world and random networks. The ordered network is a such that each node is connected to its K nearest neighbours. In general the ordered network has a high C and high L (neighbouring nodes are well connected and the path length is long).

The random network is such that all edges are randomly connected to nodes, in this case C tends to be very small L very small (neighbouring nodes are not well connected and the path length is short).

The final network, small world, is a form of the ordered network, where some nodes are disconnected and reconnected based on probability, Small world tends to have a high C but a very low L.

The brains connectivity tends to show characteristics similar to that of a small world network, where neuronal clusters are well connected via short connections [74].

The small-worldness of a given network H, which comprises of N nodes can be obtained by normalising its C and L with that of a random network such that: $\lambda = \frac{C_H}{C_{random}}$ and $\gamma = \frac{L_H}{L_{random}}$. If $\lambda > 1$ and $\gamma \approx 1$, the network is said to be characteristic of a small world network. $\eta = \frac{\lambda}{\gamma}$, defines the small worldness but must strictly be greater than 1 [73].

It is however important to stress that the probability factor allows for many networks between $p = 0$ and $p = 1$, By increasing the rewiring probability.

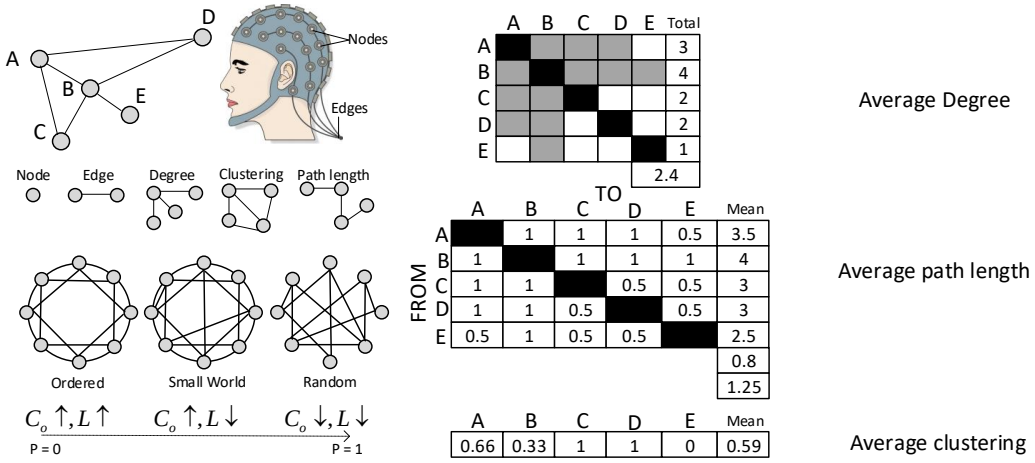

Fig. A.1 Graph theory: Overview of some of the basic concepts in graph theoretical analysis. A node is represented as a grey dot and an edge is represented as the connection between two nodes. The degree defines the number of connections connected to a node, in this case the number of connections connected to the blue dot. The path length, is the shortest distance between two nodes and clustering refers to the probability that neighbouring nodes are connected to each other. In the top example, which consists of nodes A through E, the degree of node A is 3, the minimum path length between nodes A and E is 2. The clustering coefficient of A is $\frac{2}{3}$. The three network topologies show an ordered network, which has a high C and long L, the random network has a small C and a short L, the small world network has a high C and shot L, which are properties similar to that of the human brain. In the ordered case the number of nodes are 8 N=8 and k=4

Chapter B

Code listings

B.1 Matlab Code DDA

```
clc;
clear all;
%Inital setup of signals
samples = 1024;
bins = 2048;
t  = 0:1:samples*bins;
fs = 2048;
fc = 20;
%Signal 1
S1 = 10*sin(2*pi*fc/fs*t);
f1 = 10;
f2 = 30;
%Signal 2
S2 = 10*chirp(t,f1/fs,t(end),f2/fs);
r = 4;
% Initiate variables
j = 0; p = 0; l = 0; n = 0; k = 0; storagea1 = 0; storageb1 = 0;
count1 = 0; count2 = 0; v = 1; Deltas(1) = 0; Deltas(2) = 0;
wincount = 0; RS = 0; window = samples;


for i = 2:length(s1)
    windowcount = windowcount + 1;

    %Signal 1 event detection based on variable M and Q
```

```
26      %Toggeling comparator
27      if s1(i) < s1(i-1)
28          M(end) = 0;
29      elseif s1(i) > s1(i-1)
30          M(end) = 1;
31      end
32      %Shift register minima
33      M = circshift(M,[0 1]);
34
35      %Logic outlier decision
36      if sum(M(1:(length(M)/2))) == length(M)/2 && ...
            sum(M((length(M)/2)+1:end)) == 0;
37          j = j + 1;
38      elseif nnz(~M(1:length(M)/2)) < Q && ...
            nnz(M((length(M)/2)+1:end)) < Q
39          j = j + 1;
40      end
41
42      %Same set-up for signal 2
43      if s2(i) < s2(i-1)
44          M2(end) = 0;
45      elseif s2(i) > s2(i-1)
46          M2(end) = 1;
47      end
48      M2 = circshift(M2,[0 1]);
49
50
51      if sum(M2(1:(length(M2)/2))) == length(M2)/2 && ...
            sum(M2((length(M2)/2)+1:end)) == 0;
52          p = p + 1;
53      elseif nnz(~M2(1:length(M2)/2)) < Q && ...
            nnz(M2((length(M2)/2)+1:end)) < Q
54          p = p + 1;
55      end
56
57
58
59      % signal1 minimum found start count
60      if j == 1
61          count1 = count1 + 1;
62          % signal1 second minimum found store count and reset
63      elseif j == 2
64          count1 = count1 + 1;
65          storagea1 = count1;
```

```
66          count1 = 0;
67          j = 1;
68          n = 2;
69      end
70
71      % signal2 minimum found start count
72      if p == 1
73          count2 = count2 + 1;
74          % signal2 second minimum found store count and reset
75      elseif p == 2
76          count2 = count2 + 1;
77          storageb1 = count2;
78          count2 = 0;
79          p = 1;
80          l = 2;
81      end
82
83      % Minimum trasition period found in both signals abs(diff)
84      if n == 2 && l == 2
85          Deltas(i-3) = abs(((storagea1) - (storageb1)));
86          storagea1 = 0;
87          storageb1 = 0;
88          n = 1;
89          l = 1;
90          v = v + 1;
91      else
92          Deltas(i-3) = 0;
93      end
94
95      % accumulate error for N samples (window)
96      if wincount < window
97          RS = Deltas(i-3) + RS;
98      else
99          %If window reached store phi(i)
100         k = k + 1;
101         RSfinal(k,:) = [k*window RS];
102         wincount = 0;
103         RS = 0;
104     end
105 end
106
107
108 %Synchronisation index
109 SI = RSfinal(2,:);
```

```
SIH2 = 1-min(SI(:,2),samples/2^r)/(samples/2^r);
SIH2 = smooth(SIH2,15);
```

B.2 Matlab Code Threshold

```
%%%%%%%%%%%%%%%%%%Signal 1 dda%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This is how many false positives are allowed over the length of ...
    the signal
%Find total length of SI values minus the pre, post and ictal periods
%Multiply by the desired FPR rate in order to find the total ...
    number of
%false positives
totalpeaks015 = (length(Mineoverall1(SI))/60/60)*0.15;
%Find all of the highest peaks within this period
peaks = findpeaks(Mineoverall1(SI));
%Organise the peaks from highest to lowest
[val ind] = sort(peaks,'descend');
%Take the x highest peaks
Threshold015 = val(round(totalpeaks015));

%FPR @ 0.3
totalpeaks03 = (length(Mineoverall1(SI))/60/60)*0.3;
peaks = findpeaks(Mineoverall1(SI));
[val ind] = sort(peaks,'descend');
Threshold03 = val(round(totalpeaks03));

%FPR @ 0.45
totalpeaks045 = (length(Mineoverall1(SI))/60/60)*0.45;
peaks = findpeaks(Mineoverall1(SI));
[val ind] = sort(peaks,'descend');
Threshold045 = val(round(totalpeaks045));

%FPR @ 0.6
totalpeaks06 = (length(Mineoverall1(SI))/60/60)*0.6;
peaks = findpeaks(Mineoverall1(SI));
[val ind] = sort(peaks,'descend');
Threshold06= val(round(totalpeaks06));

%FPR @ 0.75
totalpeaks075 = (length(Mineoverall1(SI))/60/60)*0.75;
```

```
peaks = findpeaks(Mineoverall1(SI));
[val ind] = sort(peaks,'descend');
Threshold075 = val(round(totalpeaks075));


threshold(1,:)= zeros(1,length(Mineoverall1)) + Threshold015;
threshold(2,:)= zeros(1,length(Mineoverall1)) + Threshold03;
threshold(3,:)= zeros(1,length(Mineoverall1)) + Threshold045;
threshold(4,:)= zeros(1,length(Mineoverall1)) + Threshold06;
threshold(5,:)= zeros(1,length(Mineoverall1)) + Threshold075;
```

B.3 VHDL Code DDA

```
----------------------VARIABLE EXPONENTIAL FILTER INPUT ...
    SIGNALS-------------------------
entity adder is
port(adder_in_s1: in std_logic_vector(9 downto 0);
      clr,clk: IN std_logic;
      alpha2: in std_logic_vector(3 downto 0);
      adder_out:   out std_logic_vector(9 downto 0)
      );
end adder;

architecture Behavioral of adder is
signal tmp2: std_logic_vector(9 downto 0):= (others => '0');
attribute dont_touch : string;
attribute dont_touch of tmp2 : signal is "true";
begin
process(clk,tmp2)
begin
if clk'event and clk = '1' then
if clr = '1' then
tmp2 <= (others => '0');
adder_out <= (others => '0');
else
--this is the calculation to create the exponential filter using ...
    shift left registers and bitvectors
tmp2 <= ((tmp2 - to_stdlogicvector((to_bitvector(tmp2) srl ...
    to_integer(unsigned(alpha2))))) + ...
```

```
      to_stdlogicvector((to_bitvector(adder_in_s1) srl ...
      to_integer(unsigned(alpha2)))));
24 end if;
25 end if;
26 adder_out ≤ tmp2;
27 end process;
28 end Behavioral;
29
30 ---------------- MINIMA DETECTION -------------------------------
31 ---------------COMPARATOR VALLEY DETECTION---------------------------------
32 entity Comparator is
33 port(compare_in_sample: in std_logic_vector(9 downto 0);
34       compare_in_ref: in std_logic_vector(9 downto 0);
35       clk,rst: in std_logic;
36       compare_out: out std_logic);
37 end Comparator;
38
39 architecture Behavioral of Comparator is
40 signal tempcomp: std_logic := '0';
41 begin
42
43 process(clk,tempcomp)
44 begin
45 if clk'event and clk = '1' then
46 if rst = '1' then
47 tempcomp ≤ '0';
48 compare_out ≤ '0';
49 else
50 -- Compare the previous value and new value coming value and ...
      assign 1,0 or equal to accordingly
51 if (compare_in_sample > compare_in_ref) then
52 tempcomp ≤ '0';
53 elsif(compare_in_sample < compare_in_ref) then
54 tempcomp ≤ '1';
55 else
56 tempcomp ≤ tempcomp;
57 end if;
58 end if;
59 end if;
60 compare_out ≤ tempcomp;
61 end process;
62
63 end Behavioral;
64 ----------------SHIFT REGISTER OUTLIER LOGIC-------------------------
```

```
65 entity Shifter is
66 port(reg2_in: in std_logic;
67      reg2_out: out std_logic_vector(9 downto 0);
68      rst: in std_logic;
69      clk: in std_logic
70       );
71
72 end Shifter;
73 architecture Behavioral of Shifter is
74 signal sig2: std_logic_vector(9 downto 0):= (others => '0');
75 begin
76 process(clk,sig2)
77 begin
78 if (clk'event and clk = '1') then
79 if rst = '1' then
80 reg2_out <= (others => '0');
81 sig2 <= (others => '0');
82 -- Shift left and store the values out from COM1 these are telling ...
       us about the signal if we are
83 --going down the signal or up the signal for the outlier event
84 else
85     sig2(0) <= reg2_in;
86     sig2(1) <= sig2(0);
87     sig2(2) <= sig2(1);
88     sig2(3) <= sig2(2);
89     sig2(4) <= sig2(3);
90     sig2(5) <= sig2(4);
91     sig2(6) <= sig2(5);
92     sig2(7) <= sig2(6);
93     sig2(8) <= sig2(7);
94     sig2(9) <= sig2(8);
95 end if;
96 end if;
97 reg2_out <= sig2;
98 end process;
99
100 end Behavioral;
101
102
103  ---------------------------------MINIMA ...
       LOGIC-----------------------------------------
104 signal temp20: std_logic_vector(9 downto 0);
105 signal temp31: std_logic ;
106 signal ones: std_logic:='0';
```

```
107 signal valleyfoundtemp: std_logic:='0';
108 signal zeros: std_logic:='0';
109 signal change: std_logic:='0';
110 begin
111  ADD  : adder PORT MAP(adder_in_s1 => data_in,adder_out => ...
       temp1,clr => clr,clk =>clk,alpha2=>alpha2);
112  COM1:   Comparator PORT MAP (compare_in_sample => ...
       temp11,compare_in_ref => temp12,compare_out => temp31,clk ...
       =>clk,rst=>clr);
113  SHFT: Shifter PORT MAP(rst => clr, reg2_in => temp31,clk => ...
       clk,reg2_out => temp20);
114  SHFT2: shft22 PORT MAP(rst => clr,shft_in => temp1,  clk => clk,  ...
       shft_out(21 downto 11) => temp11, shft_out(10 downto 0) => ...
       temp12);
115
116
117 -- Calculations to find the valleys in the signal using data from ...
      shifter
118 --ones, if we detect 4 ones in a row outut 1
119 ones <= ((temp20(3) AND temp20(2)AND temp20(1)AND temp20(0))) OR
120         (NOT(temp20(3)) AND temp20(2)AND temp20(1)AND temp20(0)) OR
121         (temp20(3) AND NOT(temp20(2))AND temp20(1)AND temp20(0)) OR
122         (temp20(3) AND temp20(2)AND NOT(temp20(1))AND temp20(0)) OR
123         (temp20(3) AND temp20(2)AND NOT(temp20(1))AND NOT(temp20(0)));
124
125 --zeros, if we detect 4 zeroes in a row output 1
126 zeros <= ((NOT(temp20(9)) AND NOT(temp20(8)) AND NOT(temp20(7)) AND ...
      NOT(temp20(6))))OR
127          ((temp20(9)) AND NOT(temp20(8)) AND NOT(temp20(7)) AND ...
             NOT(temp20(6))) OR
128          (NOT(temp20(9)) AND (temp20(8)) AND NOT(temp20(7)) AND ...
             NOT(temp20(6))) OR
129          (NOT(temp20(9)) AND NOT(temp20(8)) AND (temp20(7)) AND ...
             NOT(temp20(6))) OR
130          (NOT(temp20(9)) AND NOT(temp20(8)) AND NOT(temp20(7)) AND ...
              (temp20(6)));
131
132 -- if the middle two bits are 0 and 1 output  is one this ...
      indicates a change in direction
133 change <= (NOT(temp20(5)) AND temp20(4));
134
135 -- If all are true possible valley detected output a 1
136 valley_found <= (ones AND zeros AND change);
137 end Behavioral;
```

```
138
139
140
141
142
143 ----------------TIME STAMP TRANSITION ...
        COUNTER-----------------------------------------
144 entity CountMod is
145     port(startstop,rst,clk: in std_logic;
146           countout: out std_logic_vector(9 downto 0)
147     );
148     end CountMod;
149 -- This is a start, stop and reset counter, when a valley is ...
        detected from the valdet block
150 -- the counter starts if another is detected we reset the count, ...
        and start counting again.
151     architecture Behavioral of CountMod is
152     signal count: std_logic_vector(9 downto 0) := (others => '0');
153     signal startcount: std_logic := '0';
154     begin
155     process(clk,count)
156     begin
157     if clk'event and clk = '1' then
158     if rst = '1' then
159     count <= (others => '0');
160     countout <= (others => '0');
161     startcount <= '0';
162     else
163     --if a valley is detected
164     if startstop = '1' then
165     --send a flag to start the transitional up counter
166     startcount <= '1';
167     end if;
168     --if the counter is currently counting a transition and a ...
            second transition has yet to be found
169     --then start counting. The variable startstop is sent in from ...
            a control block which indicates
170     -- a reset
171     if startcount = '1' and startstop = '0' then
172     count <= count + 1;
173     else
174     count <= (others => '0');
175     end if;
176     end if;
```

```
177     end if;
178     countout ≤ count;
179     end process;
180     end Behavioral;
181
182     ----------------CONTROL BLOCK FLAGS FOR CALCULATION ...
            BLOCK----------------------
183 entity Calculations is
184 PORT(ready1, ready2,clk,rst: in std_logic;
185       flag1,flag3,flag5: out std_logic);
186
187 end Calculations;
188
189 architecture Behavioral of Calculations is
190 signal countv1: std_logic:='0';
191 signal countv2: std_logic:='0';
192 type states is (sr,calc1);
193 signal current_s,next_s: states;
194 begin
195
196 process (clk)
197 begin
198  if clk'event and clk = '1' then
199  if rst='1' then
200   current_s ≤ sr;
201 else
202   current_s ≤ next_s;
203   end if;
204   end if;
205   end process;
206
207
208
209  process(current_s,countv1,countv2,rst)
210   begin
211   case current_s is
212   when sr => if  rst = '1' then
213      next_s ≤ sr;
214         else
215         ---countv1 is 1 if a minima was detected in signal 1 and ...
               countv2 is 1 if --
216         ---a minima was detected in signal 2, else ...
               0-------------------------------
217             if countv1 = '1' and countv2 = '1' then
```

```
218                   next_s ≤ calc1;
219                    else
220                    next_s ≤ sr;
221            end if;
222            end if;
223    when others => next_s ≤ sr;
224    end case;
225    end process;
226
227  process(clk)
228    begin
229    if clk'event and clk = '1' then
230    if next_s = calc1 then
231    ---2 minima detected reset------
232    countv1 ≤ '0';
233    countv2 ≤ '0';
234    else
235    -----this is the block which sets the minima detected ...
          signals----------------
236    ------ready1 cis a flag which comes from the minima detection ...
          logic block--
237    if ready1 = '1' then
238    countv1 ≤  '1';
239    else
240    countv1 ≤ countv1;
241    end if;
242
243    if ready2 = '1' then
244    countv2 ≤ '1';
245      else
246    countv2 ≤ countv2;
247    end if;
248  end if;
249  end if;
250
251    end process;
252
253
254  process (current_s,ready1,ready2)
255  begin
256
257          if current_s = sr then
258          flag5 ≤ '0';
259           if ready1 = '1' then
```

```
260           flag1 ≤ '1';
261           else
262           flag1 ≤ '0';
263          end if;
264           if ready2 = '1' then
265           flag3 ≤ '1';
266           else
267           flag3 ≤ '0';
268           end if;
269
270
271          else
272          flag5 ≤ '1';
273           if ready1 = '1' then
274          flag1 ≤ '1';
275          else
276          flag1 ≤ '0';
277          end if;
278          if ready2= '1' then
279          flag3≤ '1';
280          else
281          flag3 ≤ '0';
282          end if;
283
284          end if;
285
286 end process;
287 end Behavioral;
288
289     ----------------CALCULATION BLOCK ABSOLUTE VALUE OF TRANSITIONS ...
            FROM TIMESTAMP BLOCK-----------
290     entity Calcflags is
291 PORT(clk: in std_logic;
292       flag1,flag3,flag5,rstorage: in std_logic;
293       offset: in std_logic_vector(2 downto 0);
294       flagout: out std_logic;
295       countsubed: out std_logic_vector(9 downto 0);
296       raw: out signed(9 downto 0);
297       countin1, countin2: in std_logic_vector(9 downto 0)
298 );
299 end Calcflags;
300
301 architecture Behavioral of Calcflags is
302 signal storagea1 : std_logic_vector(9 downto 0) := (others => '0');
```

```
303 signal storageb1 : std_logic_vector(9 downto 0) := (others => '0');
304 signal calcu1 :  signed(9 downto 0) := (others => '0');
305 begin
306
307
308 process(clk,calcu1)
309 begin
310 if clk'event and clk = '1' then
311 if rstorage = '1' then
312 storagea1 <= (others => '0');
313 storageb1 <= (others => '0');
314 calcu1 <= (others => '0');
315 countsubed <= (others => '0');
316 flagout <= '0';
317 else
318
319 if flag1 = '1' then
320 --if a transition period was found then store the value of count--
321 --to the storage registers
322 storagea1 <= countin1;
323 else
324 storagea1 <= storagea1 ;
325 end if;
326
327 if flag3 = '1' then
328 storageb1 <= countin2-offset;
329 else
330 storageb1 <= storageb1;
331 end if;
332 if flag5 = '1' then
333 --If a transition period in both signals has been detected then ...
      calculate the error
334 calcu1 <= (signed(storagea1) - signed(storageb1));
335 flagout <= '1';
336 else
337 calcu1 <= (others => '0');
338 flagout <= '0';
339 end if;
340 end if;
341 end if;
342 raw <= calcu1;
343 --Calculate the absolute value of the error
344 countsubed <= std_logic_vector(abs(calcu1));
345 end process;
```

```
346
347 end Behavioral;
348
349 ----------------------INDEXING AND ACCUMULATION ...
       BLOCK-------------------------------
350 entity Indexing is port
351 (clk,rst: in std_logic;
352  flag: out std_logic;
353  flagin: in std_logic;
354   N: in std_logic_vector(9 downto 0);
355       cal: in std_logic_vector(9 downto 0);
356       sum: out std_logic_vector(9 downto 0)
357 );
358 end Indexing;
359
360 architecture Behavioral of Indexing is
361 signal accumvalue: std_logic_vector(9 downto 0) := (others => '0');
362 signal numberofsamples:std_logic_vector(9 downto 0) := (others => ...
       '0');
363 attribute dont_touch : string;
364 attribute dont_touch of sum,accumvalue : signal is "true";
365 begin
366 process(clk)
367 begin
368 if clk'event and clk = '1' then
369 if rst = '1' then
370 sum <= (others => '0');
371 accumvalue<= (others => '0');
372 numberofsamples <= (others => '0');
373 else
374 if numberofsamples < N then
375 if flagin = '1' then
376 --while the the number of accumulations are less than N accumlate ...
       errors
377 accumvalue <= accumvalue + cal;
378 flag <= '0';
379 numberofsamples <= numberofsamples + 1;
380 sum <= (others => '0');
381 else
382 accumvalue <= accumvalue;
383 flag <= '0';
384 numberofsamples <= numberofsamples;
385 sum <= (others => '0');
386 end if;
```

```
387 else
388 flag ≤ '1';
389 sum ≤ accumvalue;
390 numberofsamples ≤ (others => '0');
391 accumvalue≤ (others => '0');
392 end if;
393 end if;
394 end if;
395 end process;
396 end Behavioral;
397
398 ----------------OUTPUT EXPONENTIAL ...
      FILTER----------------------------------------
399
400 library IEEE;
401 use IEEE.STD_LOGIC_1164.ALL;
402 USE IEEE.STD_LOGIC_UNSIGNED.ALL;
403 use IEEE.NUMERIC_STD.ALL;
404
405
406 entity SmoothDetect is port
407 (sumin: in std_logic_vector(9 downto 0);
408  flag,rst: in  std_logic;
409  sumout: out std_logic_vector(9 downto 0);
410  alpha: in std_logic_vector(3 downto 0);
411  clk: in std_logic);
412
413 end SmoothDetect;
414
415 architecture Behavioral of SmoothDetect is
416 signal y_minus_1: std_logic_vector(9 downto 0):= (others => '0');
417 attribute dont_touch : string;
418 attribute dont_touch of y_minus_1 : signal is "true";
419 attribute dont_touch of sumout : signal is "true";
420 begin
421
422 process(clk,y_minus_1)
423 begin
424 if clk'event and clk = '1' then
425 if rst = '1' then
426 y_minus_1 ≤ (others => '0');
427 sumout ≤ (others => '0');
428 else
429 if flag = '1' then
```

```
430 --This is the main calculation for the output exponential filter
431 y_minus_1 <= (y_minus_1 - ...
       to_stdlogicvector((to_bitvector(y_minus_1) srl ...
       to_integer(unsigned(alpha))))) + ...
       to_stdlogicvector(to_bitvector(sumin) srl ...
       to_integer(unsigned(alpha)));
432 else
433 y_minus_1 <= y_minus_1;
434 end if;
435 end if;
436 end if;
437 sumout <= y_minus_1;
438 end process;
439 end Behavioral;
440
441 ----------UPLOAD LINK BLOCK UPLOADS ALL OF THE PARAMETERES FOR THE ...
       ASIC-------
442 ----------NOTE THAT THIS USES A 48 BIT SIPO REGISTER--------------------
443 entity UploadLink is
444 port(clk2,rst,datain1,datain2,loadparameters:in std_logic;
445      clkout,paramcomplete,rst2:out std_logic;
446      dataout1,dataout2: out std_logic_vector(9 downto 0);
447       N: out std_logic_vector(9 downto 0);
448      alpha: out std_logic_vector(3 downto 0);
449      alpha2: out std_logic_vector(3 downto 0);
450      offset: out std_logic_vector(2 downto 0);
451      Ndetected: out std_logic_vector(5 downto 0);
452      Tdropout: out std_logic_vector(9 downto 0);
453      Triseout: out std_logic_vector(9 downto 0)
454 );
455 end UploadLink;
456
457 architecture Behavioral of UploadLink is
458
459 signal shiftdata1: std_logic_vector(9 downto 0):= (others => '0');
460 signal shiftdata2: std_logic_vector(9 downto 0):= (others => '0');
461 signal dummy1: std_logic_vector(9 downto 0):= (others => '0');
462 signal dummy2: std_logic_vector(9 downto 0):= (others => '0');
463 signal parameterreg: std_logic_vector(47 downto 0):= (others => '0');
464 signal counterd1d2: std_logic_vector(3 downto 0):= (others => '0');
465 signal countert1t2: std_logic_vector(5 downto 0):= (others => '0');
466 signal slavecounter: std_logic_vector(3 downto 0);
467 signal paramcomplete2: std_logic := '0';
468 signal clktemp,test: std_logic;
```

```
469 attribute dont_touch : string;
470 attribute dont_touch of dummy1,dummy2: signal is "true";
471 begin
472 process(clk2,dummy1,dummy2,paramcomplete2)
473 begin
474
475 if clk2'event and clk2 = '1' then
476 ----------clockslave----------
477
478 if rst = '1' then
479 shiftdata1 <= (others => '0');
480 shiftdata2 <= (others => '0');
481 paramcomplete2 <= '0';
482 parameterreg <= (others => '0');
483 counterd1d2 <= (others => '0');
484 countert1t2 <= (others => '0');
485 dataout1 <= (others => '0');
486 dataout2 <= (others => '0');
487 Tdropout <= (others => '0');
488 Triseout <= (others => '0');
489 N <= (others => '0');
490 alpha <= (others => '0');
491 alpha2 <= (others => '0');
492 offset <= (others => '0');
493 Ndetected <= (others => '0');
494 slavecounter <= (others => '0');
495 dummy1 <= (others => '0');
496 dummy2 <= (others => '0');
497 clktemp <= '0';
498 rst2 <= '1';
499 else
500 if slavecounter <= "0101" then
501 slavecounter <= slavecounter + 1;
502 clktemp <= '1';
503 elsif slavecounter >= "0110" and slavecounter < "1011" then
504 clktemp <= '0';
505 slavecounter <= slavecounter + 1;
506 else
507 rst2 <= '0';
508 slavecounter <= (others => '0');
509 end if;
510 ----------datain----------
511 if paramcomplete2 = '1' then
512 if  counterd1d2 <= "1001" then
```

```
513 shiftdata1(9 downto 1) <=  shiftdata1(8 downto 0);
514 shiftdata1(0) <=  datain1;
515 shiftdata2(9 downto 1) <=  shiftdata2(8 downto 0);
516 shiftdata2(0) <=  datain2;
517 counterd1d2 <= counterd1d2 + 1;
518 elsif counterd1d2 = "1010" then
519 dummy1 <= shiftdata1;
520 dummy2 <= shiftdata2;
521 counterd1d2 <= counterd1d2 + 1;
522 else
523 shiftdata1 <= (others => '0');
524 shiftdata2 <= (others => '0');
525 counterd1d2 <= (others => '0');
526 end if;
527 else
528 shiftdata1 <= (others => '0');
529 shiftdata2 <= (others => '0');
530 counterd1d2 <= (others => '0');
531 end if;
532 ----------------------constants-------------------------------------
533 if countert1t2 < "110001" then
534 countert1t2 <= countert1t2 + 1;
535 parameterreg(47 downto 1) <=  parameterreg(46 downto 0);
536 parameterreg(0) <=  loadparameters;
537 paramcomplete2 <= '0';
538 else
539 --parametercheckreg <= parameterreg;
540 Tdropout <= parameterreg(47 downto 37);
541 Triseout <= parameterreg(36 downto 27);
542 Ndetected <= parameterreg(26 downto 21);
543 offset <= parameterreg(20 downto 18);
544 alpha <= parameterreg (17 downto 14);
545 alpha2 <= parameterreg(13 downto 10);
546 N <= parameterreg(9 downto 0);
547 paramcomplete2 <= '1';
548 end if;
549 end if;
550 end if;
551 dataout1 <= dummy1;
552 dataout2 <= dummy2;
553 paramcomplete <= paramcomplete2;
554 end process;
555
556 clkout <= clktemp;
```

```
557 end Behavioral;
558 ----------------------------DOWNLOAD LINK THIS IS THE PISO REGISTER------
559 ----------------------------FOR THE SYNCHRONISATION VALUES---------------
560 ----------------------------AND TEST PARAMETERS--------------------------
561 entity DownloadLink is
562 port(clk,rst,load: in std_logic;
563      sumin:in std_logic_vector(9 downto 0);
564      nonsmoothed: in STD_LOGIC_VECTOR (9 downto 0);
565      sumserialout,beforesmooth: out std_logic
566 );
567 end DownloadLink;
568
569
570
571 architecture Behavioral of DownloadLink is
572
573 signal holdsum: std_logic_vector(9 downto 0):= (others => '0');
574 signal nonsmoothedsum: std_logic_vector(9 downto 0):= (others => '0');
575 begin
576 process(clk,holdsum,nonsmoothedsum)
577 begin
578 if clk'event and clk = '1' then
579 if rst = '1' then
580 holdsum <= (others => '0');
581 sumserialout <= '0';
582 nonsmoothedsum <= (others => '0');
583 beforesmooth <= '0';
584 else
585 if load = '1' then
586 holdsum <= sumin;
587 nonsmoothedsum <= nonsmoothed;
588 else
589 holdsum(9 downto 1) <= holdsum(8 downto 0);
590 holdsum(0) <= '0';
591 nonsmoothedsum(9 downto 1) <= nonsmoothedsum(8 downto 0);
592 nonsmoothedsum(0) <= '0';
593 end if;
594 end if;
595 end if;
596 sumserialout <= holdsum(9);
597 beforesmooth <= nonsmoothedsum(9);
598 end process;
599 end Behavioral;
```

B.4 RTL-compiler

```
#set_attribute hdl_vhdl_environment common
set_attribute hdl_vhdl_read_version 1993
set_attribute hdl_vhdl_case original
# this tells the compiler where to look for the libraries
set_attribute lib_search_path ...
    /cad/KITS/CADENCE_13/ams_cdk_4.11_upd1/liberty/h18_1.2V
#this defies the libraries to use
set_attribute library {h18_CORELIB_HVT_TYP.lib}
#convert non conventional operands to library specified operands
set_attribute lp_insert_operand_isolation true /
#Clock gating on (reduces power consumption)
set_attribute lp_insert_clock_gating true /
#How much information to show user on design
set_attribute information_level 7
#Flag black box errors
set_attribute hdl_error_on_blackbox 1
#Flag latch errors
set_attribute hdl_error_on_latch 1
set_attribute interconnect_mode ple
#Load all VHDL files
read_hdl -vhdl ../rtl/adder.vhd
read_hdl -vhdl ../rtl/Calcflags.vhd
read_hdl -vhdl ../rtl/Comparator.vhd
read_hdl -vhdl ../rtl/CountMod.vhd
read_hdl -vhdl ../rtl/Detection.vhd
read_hdl -vhdl ../rtl/Indexing.vhd
read_hdl -vhdl ../rtl/Main.vhd
read_hdl -vhdl ../rtl/MainMain.vhd
read_hdl -vhdl ../rtl/shft22.vhd
read_hdl -vhdl ../rtl/Shifter.vhd
read_hdl -vhdl ../rtl/SmoothDetect.vhd
read_hdl -vhdl ../rtl/Valleydetect.vhd
read_hdl -vhdl ../rtl/Comlink.vhd
read_hdl -vhdl ../rtl/UploadLink.vhd
read_hdl -vhdl ../rtl/DownloadLink.vhd
read_hdl -vhdl ../rtl/Paramcheck.vhd
#Elaborate the design
elaborate Comlink
#Read constraints file
read_sdc ../constrains/CountMod.sdc
```

```
40 #setup clocks in this case max frequency 125KhZ
41 define_clock  -domain "system" -period 8000000 -fall 50 -rise 50  ...
      -name clkup [find / -port clkup]
42 #Define the amount of skew
43 set_attribute clock_setup_uncertainty {100 50} [find -clock clkup]
44 #Define the amount of slew {minrise,minfall,maxrise,maxfall}
45 set_attribute slew {1000 1000 1500 1500} [find -clock clkup]
46 #Define input delays #all inputs will change 10000 ps after rising ...
      edge
47 external_delay -input 100000 -clock [find -clock clkup] -edge_rise ...
      [find /des* -port ports_in/*]
48 #Define output delays #all output data will be ready 10000 ps ...
      before rising edge
49 external_delay -output  100000 -clock [find -clock clkup] ...
      -edge_rise [find /des* -port ports_out/checkdata1]
50 external_delay -output  100000 -clock [find -clock clkup] ...
      -edge_rise [find /des* -port ports_out/checkdata2]
51 #Synthesise the design
52 synthesize -to_generic -effort high
53 synthesize -to_mapped -effort high
54 #Write final verilog file to be used in encounter layout
55 write_hdl > ../Outputs/Phase_final_VERILOG_preserved.v
```

B.5 MBED-C++ test platform code

```
1  #include "mbed.h"//import mbed libraries
2  #include "MSCFileSystem.h"
3  #include "rtos.h"
4  #include "FastPWM.h"
5  //Setup all the GPIO pins and special pins
6  DigitalOut Clear(p14);
7  DigitalOut Msel1(p13);
8  DigitalOut Msel2(p15);
9  DigitalOut Enable(p10);
10 DigitalOut aaout(p29);
11 DigitalOut Sig1Out(p11);
12 DigitalOut Sig2Out(p16);
13 DigitalOut CLK(p12);
14 FastPWM CLK2(p23);
15 DigitalIn SIindex(p27);
```

```
16 //Use the MSCFilesystem to read from the Fat32 SUB stick
17 MSCFileSystem fs("usb");
18 Ticker ticker2;
19 Ticker ticker3;
20 InterruptIn CLKs(p18);
21 InterruptIn Mux(p28);
22 Thread thread;
23 //Open serial USB port
24 Serial pc(USBTX, USBRX);
25 //Intermidiate variables
26 bool rising = false;
27 bool rising2 = false;
28 bool rising3 = false;
29 char s1;
30 char s2;
31 bool s1int = false;
32 bool s2int = false;
33 int loop = 1;
34 bool runit = true;
35 int temp = 0;
36 //This is a seperate thread which reads the output of the ASIC ...
       every clock cycle it is active only if the end
37 //of the binary neural data files are still not empty, if the ...
       variable K from the ASIC is active high and if the processors
38 //CLK is also high. It reads 10 samples.
39 void clockslow_thread()
40 {
41     while(runit == true) {
42         if(rising2 == true) {
43             for(int nn = 0; nn < 11; nn++) {
44                 if(CLK == true) {
45                     pc.printf("%d%d \r\n",SIindex.read(),rising2);
46                 }
47
48             }
49
50         }
51     }
52 //Seperate function which sets up the main synchronisation CLK for ...
       the processor 2KHz
53 //This gets called using a timer and is called periodically.
54     void clocktick2() {
55         CLK = !CLK;
56     }
```

```
57 //This is a function which is activated by an external interrupt ...
       Mux.rise.
58 //How this is activated is explained later
59     void interrupt_triggered2() {
60         rising2 = true;
61     }
62 //The next functions setup a matched register case PWM fast clock ...
       of 20KHz on a dedicated
63 //PWM pin on the MBED
64     void PWMInit();
65     void PWM1_IRQHandler () {
66         if((LPC_PWM1->IR & 0x01) == 0x01) { // Check whether it is ...
               an interrupt from match  0
67             //If the PWM is rising then set this variable to true, ...
                   this is an edge detector
68             rising = true;
69             LPC_PWM1->IR |= 1<<0; // Clear the interrupt flag
70         }
71         return;
72     }
73
74     void PWMInit() {
75         LPC_SC->PCONP |= 1 << 6; // enable power for PWM1
76         LPC_SC->PCLKSEL0 |= 1 << 12; // PCLK_PWM1 = CCLK
77
78         LPC_PWM1->MR0 = 16000; // PWM freq = CCLK/16000
79
80         LPC_PWM1->MCR |= 1 << 0; // interrupt on Match0
81
82         LPC_PINCON->PINSEL3 |= (2 << 14 ); // P2.4 works as PWM1 ...
               output
83         LPC_PINCON->PINMODE3 |= 2 << 14; // enable neither pull up ...
               nor pull down
84
85
86         LPC_PWM1->PCR |= 1 << 4; // Configure channel 4 as double ...
               edge controlled PWM
87         LPC_PWM1->PCR |= 1 << 12; // Enable PWM channel 4 output
88
89         LPC_PWM1->MR0 = 16000; // PWM freq = CCLK/16000
90
91         LPC_PWM1->TCR = (1 << 0) | (1 << 3); // enable timer ...
               counter and PWM
92
```

```
93
94          NVIC_SetVector(PWM1_IRQn, (uint32_t)&PWM1_IRQHandler);
95          NVIC_EnableIRQ(PWM1_IRQn);
96          return;
97      }
98
99
100     int main() { // start main function
101
102
103         PWMInit();
104
105         //Set the serial communication bandwidth for the USB port
106         pc.baud(115200);
107         //These variables set an internal MUX inside the ASIC such ...
                that the ASIC sends out
108         //a pulse every 1024 samples. This tells us every time ...
                that an accumulation of errors
109         //has been colleted
110         Msel1=1;
111         Msel2 = 1;
112         //Enable sets an internal variable in the ASICs SPI ...
                interface which tells it to activate
113         Enable = 1;
114         //Open the two neural signal files which are stored on the ...
                Fat32 formatted memory stick
115         FILE *fp = fopen("/usb/Signal1.txt", "r");
116         FILE *fp2 = fopen("/usb/Signal2.txt", "r");
117         //Print an error message if there was a problem opening ...
                the files
118         pc.printf("%s\n", (!fp ? "Fail :(" : "OK"));
119         if (!fp)
120             error("error: %s (%d)\n", strerror(errno), -errno);
121         wait(2);
122         //Start runnning a seperate thread which calls the CLK ...
                funtion as described above
123         //This calls the Clk function every 500 micro seconds ...
                which is equal to 2KHz main
124         //processor operating frequency
125         thread.start(&clockslow_thread);
126         ticker2.attach_us(&clocktick2, 500);
127         //This is an external interrupt which is connected to the ...
                value K from the ASIC.
```

```
128         //This is a single pulse every 1024 samples which sets the ...
                variable rising2 to true
129         Mux.rise(&interrupt_triggered2);
130         //This is the main file loop it continues until the end of ...
                the fneural signal file
131         while (!feof(fp)) {
132             LPC_PWM1->MR3 = 6000;
133             LPC_PWM1->MR4 = 15000;
134             LPC_PWM1->LER |=  (1<<3) | (1<<4) ;
135             //If the variable rising is true it means that a ...
                    rising edge on the 20KHz clock has been detected
136             //If this is true then we want to read data from the ...
                    Fat32 USB stick and send it to the ASIC
137             if(rising == true) {
138                 rising = false;
139                 //Collect 1 bit of data from each of the two ...
                        neural signal files
140                 s1 =  fgetc(fp);
141                 s2 =  fgetc(fp2);
142                 //Since the function fgetc is a character we have ...
                        to do a test and assign an intermidate
143                 //variable with a boolean variable which is ...
                        representative of the character.
144                 if(s1 == '1') {
145                     s1int = true;
146                 } else if (s1 == '0') {
147                     s1int =false;
148                 }
149                 if(s2 == '1') {
150                     s2int = true;
151                 } else if (s2 == '0') {
152                     s2int  = false;
153                 }
154
155                 //Send the data to the GPIO pins which are ...
                        connected to the ASICS data intput pins
156                 Sig1Out = s1int;
157                 Sig2Out = s2int;
158             }
159         }
160         fclose(fp);
161         fclose(fp2);
162         runit = false;
163     }
```

Chapter C

Pseudo codes

C.1 Pseudo code DDA hardware flow

The following is a pseudo code representation of the synchronisation algorithm hardware flow as seen in chapter 3.

1: **procedure** MINIMA DETECTION ▷ Example signal 1, same applies to signal 2
2: *loop*:
3: **if** $S1(i) < S1(i-1)$ **then**
4: $C1 \leftarrow 0$.
5: **else if** $S1(i) > S1(i-1)$ **then**
6: $C1 \leftarrow 1$.
7: **else**
8: $C1 \leftarrow C1$.
9: **if** $O1 = 0000011111$ **then** ▷ Possible to have one outlier each side
10: $M1 \leftarrow 1$.
11: **else**
12: $M1 \leftarrow 0$.
13: **procedure** TIME STAMP
14: **if** $M1 = 1$ **then**
15: $Co \leftarrow Count$.
16: $Co \leftarrow 0$.
17: **else**
18: $Co \leftarrow Co + 1$.
19: **procedure** CALCULATION AND FILTER

```
20:     if K < Window then
21:         if M1 = 1 ∧ m2 = 1 then
22:             L1 ← Co.
23:             L2 ← Co.
24:             Sub ← L1 - L2.
25:             if s = 0 then
26:                 Abs ← Sub.
27:             else
28:                 Abs ← |Sub|.
29:             Adderout ← Abs+feed.
30:             M1 ← 0.
31:             M2 ← 0.
32:         else
33:             null
34:     else
35:         φ(i) ← phiraw.
36:         Exp ← smoothed(Addreg).
37:         y(i) ← y(i − 1) − y(i − 1)α + φ(i)α.
38: procedure DETECT
39:     if y(i) < thl then
40:         Detect ← 1.
41:     else if y(i) > thh then
42:         Detect ← 1.
43:     else
44:         Detect ← 0.
```

C.2 Pseudo code for training and calculation hardware flow

procedure TRAIN AND CALCULATE
  if $Inital = 1$ **then**
    $MemL_{epoch,1:8} \leftarrow 1.$
    $MemH_{epoch,1:8} \leftarrow 0.$
  else if $Train = 1$ **then**
    if $y(i)_{epoch(x)} < MemL_{epoch(x)}$ **then**
      $MemL_{epoch(x)} \leftarrow y(i)_{epoch(x)}.$
      $T6 \leftarrow 1.$
    else
      $MemL_{epoch(x)} \leftarrow MemL_{epoch(x)}.$
      $T7 \leftarrow 1.$
    if $y(i)_{epoch(x)} > MemH_{epoch(x)}$ **then**
      $MemH_{epoch(x)} \leftarrow y(i)_{epoch(x)}.$
      $T2 \leftarrow 1.$
    else
      $MemH_{epoch(x)} \leftarrow MemH_{epoch(x)}.$
      $T1 \leftarrow 1.$
  else
loop while epoch < 9:
    if *S1 = 1* **then**
      $MemL_{epoch(x)} \leftarrow \Delta.$
      $MemH_{epoch(x)} \leftarrow MemH_{epoch(x)}.$
      $T4 \leftarrow 1.$
      $T1 \leftarrow 1.$
      $T8 \leftarrow 1.$
    else if *S2 = 2* **then**
      $MemL_{epoch(x)} \leftarrow \Delta.$
      $MemH_{epoch(x)} \leftarrow MemH_{epoch(x)} - \Delta.$
      $T7 \leftarrow 1.$
      $T8 \leftarrow 1.$
      $T2 \leftarrow 1.$
    else if *S3 = 3* **then**
      $MemH_{epoch(x)} \leftarrow MemH_{epoch(x)}.$
      $MemL_{epoch(x)} \leftarrow MemH_{epoch(x)} - 2\Delta.$
      $T9 \leftarrow 1.$
      $T5 \leftarrow 1.$
      $T1 \leftarrow 1.$

Chapter D

Control signals for future 16-channel testing platform

The following appendix shows the main control signals for the 16-channel ASIC implementation of the DDA algorithm. Where, the first column indicates if the signal is an input or an output. Next, the signal name is given as seen in the layout view of the device. The number column indicates the pin number as per the selected encapsulation package. The Voltage is the desired voltage for the pin. The pin type differentiates between pad ring power supply voltages, circuit supply voltages and general input output pins. The function column gives a brief description of the pin and the last column dictates whether the pin is active on a high or low signal.

I/O	*Name*	*Number*	*Voltage*	*Pin type*	*(V) reg*	*Function*	*Signal*
I	VSSPAD1	5	0v	GROUND	NO	PADRING SUPPLY	
I	VDDPAD	7	3.3v	POWER	NO	PADRING SUPPLY	
I	vssd05!	8	0v	GROUND	NO	CIRCUIT SUPPLY	
I	vddd05!	9	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	Inital_PAD1	10	0.5v	GPIO	YES	Pre load training memories	active high
I	RAWBYPASS_PAD1	11	0.5v	GPIO	YES	Dont bypass filter stage (testing)	active high
I	FILLBYPASS_PAD1	12	0.5v	GPIO	YES	Bypass filter stage (testing)	active high
O	HL9_14PBUFEDPAD	14	3.3v	GPIO	YES	detection values processors 9 to 14	
I	vssd05!	16	0v	GROUND	NO	CIRCUIT SUPPLY	
I	vddd05!	17	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	CLEARSYNC_PAD	18	0.5v	GPIO	YES	Clear processors 1 to 16	active low
I	set_PAD1	19	0.5v	GPIO	YES	set bit for logic unit ring counter	active high
I	CalcC_PAD1	20	0.5v	GPIO	YES	Training finished calculate thresholds	active high
I	CLEAR1 _PAD1	21	0.5v	GPIO	YES	Clear epoch counter	active low
I	VSSPAD1	22	0v	GROUND	NO	PADRING SUPPLY	
I	VDDPAD	23	3.3v	POWER	NO	PADRING SUPPLY	
I	vssd05!	24	0v	GROUND	NO	CIRCUIT SUPPLY	
I	vddd05!	25	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	CLEAR2_PAD1	26	0.5v	GPIO	YES	Clear epoch counter	active low
I	TRAIN_PAD1	27	0.5v	GPIO	YES	Set training mode on	active high
O	HL11_16PBUFEDPAD	28	3.3v	GPIO	YES	detection values processors 11 to 16	
I	test1in1_PAD1	29	0.5v	GPIO	YES	bit1 LSB for signal 1 of test processor	
I	test1in2_PAD1	30	0.5v	GPIO	YES	bit2 for signal 1 of test processor	
I	vssd05!	31	0v	GROUND	NO	CIRCUIT SUPPLY	
I	vddd05!	32	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	test1in3_PAD1	33	0.5v	GPIO	YES	bit3 for signal 1 of test processor	
I	test1in4_PAD1	34	0.5v	GPIO	YES	bit4 for signal 1 of test processor	
I	test1in5_PAD1	35	0.5v	GPIO	YES	bit5 for signal 1 of test processor	
I	test1in6_PAD1	36	0.5v	GPIO	YES	bit6 for signal 1 of test processor	
I	test1in7_PAD1	37	0.5v	GPIO	YES	bit7 for signal 1 of test processor	
I	test1in8_PAD1	38	0.5v	GPIO	YES	bit8 for signal 1 of test processor	
I	test1in9_PAD1	39	0.5v	GPIO	YES	bit9 for signal 1 of test processor	
I	test1in10_PAD1	40	0.5v	GPIO	YES	bit10 MSB for signal 1 of test processor	
I	test2in1_PAD1	41	0.5v	GPIO	YES	bit1 LSB for signal 2 of test processor	
I	test2in2_PAD1	42	0.5v	GPIO	YES	bit2 for signal 2 of test processor	
I	test2in3_PAD1	43	0.5v	GPIO	YES	bit3 for signal 2 of test processor	
I	test2in4_PAD1	44	0.5v	GPIO	YES	bit4 for signal 2 of test processor	
I	test2in5_PAD1	45	0.5v	GPIO	YES	bit5 for signal 2 of test processor	
I	vssd05!	46	0v	GROUND	NO	CIRCUIT SUPPLY	
I	vddd05!	47	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	VSSPAD1	48	0v	GROUND	NO	PADRING SUPPLY	
I	VDDPAD	49	3.3v	POWER	NO	PADRING SUPPLY	
I	test2in6_PAD1	63	0.5v	GPIO	YES	bit6 for signal 2 of test processor	
I	vddd05!	65	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	vssd05!	66	0v	GROUND	NO	CIRCUIT SUPPLY	
I	test2in7_PAD1	67	0.5v	GPIO	YES	bit7 for signal 2 of test processor	
I	test2in8_PAD1	68	0.5v	GPIO	YES	bit8 for signal 2 of test processor	
I	test2in9_PAD1	69	0.5v	GPIO	YES	bit9 for signal 2 of test processor	
I	test2in10_PAD1	71	0.5v	GPIO	YES	bit10 MSB for signal 2 of test processor	
O	BOUT16test1_PAD	72	3.3v	GPIO	YES	bit1 of synchronisation values from test processor	

O	BOUT16test2_PAD	73	3.3v	GPIO	YES	bit2 of synchronisation values from test processor	
O	BOUT16test3_PAD	75	3.3v	GPIO	YES	bit3 of synchronisation values from test processor	
I	VDDPAD	76	3.3v	POWER	NO	PADRING SUPPLY	
I	VSSPAD1	78	0v	GROUND	NO	PADRING SUPPLY	
O	BOUT16test4_PAD	79	3.3v	GPIO	YES	bit4 of synchronisation values from test processor	
O	BOUT16test5_PAD	80	3.3v	GPIO	YES	bit5 of synchronisation values from test processor	
O	BOUT16test6_PAD	81	3.3v	GPIO	YES	bit6 of synchronisation values from test processor	
O	BOUT16test7_PAD	83	3.3v	GPIO	YES	bit7 of synchronisation values from test processor	
O	BOUT16test8_PAD	84	3.3v	GPIO	YES	bit8 of synchronisation values from test processor	
O	BOUT16test9_PAD	85	3.3v	GPIO	YES	bit9 of synchronisation values from test processor	
O	BOUT16test10_PAD	86	3.3v	GPIO	YES	bit10 of synchronisation values from test processor	
I	ENABLE_INPUTCONVERTER_PAD2	87	0.5v	GPIO	YES	Enable input converter latach data	active high
I	EXTRIG_PAD1	88	0.5v	GPIO	YES	Enable output converter (read data)	active high
I	vddd05!	90	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	vssd05!	91	0v	GROUND	NO	CIRCUIT SUPPLY	
I	VDDPAD	109	3.3v	POWER	NO	PADRING SUPPLY	
I	VSSPAD1	111	0v	GROUND	NO	PADRING SUPPLY	
I	vddd05!	112	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	vssd05!	113	0v	GROUND	NO	CIRCUIT SUPPLY	
O	OUTMAINPAD	115	3.3v	GPIO	YES	Ouput from PSI interface	
O	KPAD	116	3.3v	GPIO	YES	Test ouput for K	
I	RAWBYPASS_PAD2	117	0.5v	GPIO	YES	Dont bypass filter stage (testing)	active high
I	FILLBYPASS_PAD2	118	0.5v	GPIO	YES	Bypass filter stage (testing)	active high
I	Inital_PAD2	119	0.5v	GPIO	YES	Pre load training memories	active high
O	HL3_8PBUFEDPAD	120	3.3v	GPIO	YES	detection values processors 3 to 8	
O	STAGE4_PAD	121	3.3v	GPIO	YES	Test output from logic control	
O	STAGE3_PAD	122	3.3v	GPIO	YES	Test output from logic control	
O	STAGE2_PAD	123	3.3v	GPIO	YES	Test output from logic control	
O	a8TRAIN_PAD	124	3.3v	GPIO	YES	Test output from logic control	
O	a7TRAIN_PAD	125	3.3v	GPIO	YES	Test output from logic control	
O	a6TRAIN_PAD	126	3.3v	GPIO	YES	Test output from logic control	
I	vddd05!	127	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	vssd05!	128	0v	GROUND	NO	CIRCUIT SUPPLY	
O	a5TRAIN_PAD	129	3.3v	GPIO	YES	Test output from logic control	
O	a4TRAIN_PAD	130	3.3v	GPIO	YES	Test output from logic control	
O	a3TRAIN_PAD	131	3.3v	GPIO	YES	Test output from logic control	
O	a2TRAIN_PAD	132	3.3v	GPIO	YES	Test output from logic control	
O	a1TRAIN_PAD	133	3.3v	GPIO	YES	Test output from logic control	
I	vddd05!	134	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	vssd05!	135	0v	GROUND	NO	CIRCUIT SUPPLY	
I	VDDPAD	136	3.3v	POWER	NO	PADRING SUPPLY	
I	VSSPAD1	137	0v	GROUND	NO	PADRING SUPPLY	
I	CLEAR2_PAD2	138	0.5v	GPIO	YES	Clear epoch counter	active low
I	CLEAR1 _PAD2	139	0.5v	GPIO	YES	Clear epoch counter	active low
I	set_PAD2	140	0.5v	GPIO	YES	set bit for logic unit ring counter	active low
I	CalcC_PAD2	141	0.5v	GPIO	YES	Training finished calculate thresholds	
I	vddd05!	142	0.5v	POWER	NO	CIRCUIT SUPPLY	
I	vssd05!	143	0v	GROUND	NO	CIRCUIT SUPPLY	
I	TRAIN_PAD2	144	0.5v	GPIO	YES	Set training mode on	active high
O	HL1_6PBUFEDPAD	145	3.3v	GPIO	YES	detection values processors 1 to 6	
I	CLEARSYNC_PAD2	146	0.5v	GPIO	YES	Clear processors 1 to 16	active low

O	MEM_TESTBITPAD	147	3.3v	GPIO	YES	Test output from memory module
O	INPUT_TESTBITPAD	148	3.3v	GPIO	YES	Test bit from SPI interace
I	vddd05!	149	0.5v	POWER	NO	CIRCUIT SUPPLY
I	vssd05!	150	0v	GROUND	NO	CIRCUIT SUPPLY
I	VDDPAD	151	3.3v	POWER	NO	PADRING SUPPLY
I	VSSPAD1	152	0v	GROUND	NO	PADRING SUPPLY
I	vssd05!	169	0v	GROUND	NO	CIRCUIT SUPPLY
I	vddd05!	171	0.5v	POWER	NO	CIRCUIT SUPPLY
I	CLEARSYNC_PAD3	177	0.5v	GPIO	YES	Clear processors 1 to 16 active low
I	CLK2_PAD	182	0.5v	CLK PIN	YES	Input clock for SPI interface
I	VSSPAD1	184	0v	GROUND	NO	PADRING SUPPLY
I	VDDPAD	186	0.5v	POWER	NO	PADRING SUPPLY
I	ENABLE_INPUTCONVERTER_PAD1	187	0.5v	GPIO	YES	Enable input converter latach data active high
I	CLEARINPUTCONV_PAD	188	0.5v	GPIO	YES	Clear flip flops and internal counter of SPI interface active low
I	CLK_PAD	192	0.5v	CLK PIN	YES	Master clock
I	vssd05!	195	0v	GROUND	NO	CIRCUIT SUPPLY
I	vddd05!	196	0.5v	POWER	NO	CIRCUIT SUPPLY
I	DATAIN_PAD	197	0.5v	GPIO	YES	Serial Data in to SPI interface

Chapter E

James Romaine, Publication list and curriculum

Phase Synchronization Operator for on-chip Brain Functional Connectivity Computation Manuel Delgado-Restituto, James B. Romaine and Ángel Rodríguez-Vázquez Journal Paper-IEEE Transactions on Biomedical Circuits and Systems (TBioCAS) 2019

Highly scalable real time epilepsy diagnosis architecture via phase correlation and functional brain maps J.B. Romaine, M. Delgado-Restituto and A. Rodríguez-Vázquez Conference - IEEE Biomedical Circuits and Systems Conference BioCAS 2018

Highly Scalable Real Time Epilepsy Diagnosis Architecture Via Phase Correlation J.B. Romaine, M. Delgado-Restituto and A. Rodríguez-Vázquez Journal Paper - Procedia Technology, vol. 27, pp 55-56, 2017 ELSEVIER DOI: 10.1016/j.protcy.2017.04.026 ISSN: 2212-0173

Real-time phase correlation based integrated system for seizure detection J.B. Romaine, M. Delgado-Restituto, J.A. Leñero-Bardallo and A. Rodríguez-Vázquez Conference - Bio-MEMS and Medical Microdevices III Conference 2017

Integer-based digital processor for the estimation of phase synchronization between neural signals J.B. Romaine, M. Delgado-Restituto, J.A. Lenero-Bardallo and A. Rodriguez-Vazquez Conference - Conference on Ph.D Research in Microelectronics and

Electronics PRIME 2016

Highly scalable real time epilepsy diagnosis architecture via phase correlation and functional brain maps J.B. Romaine, L. Acasandrei, M. Delgado-Restituto, A. Rodríguez-Vázquez Conference - World Congress on Biosensors BIOSENSORS 2016

Hardware friendly algorithm for the calculation of phase synchronization between neural signals J.B. Romaine and M. Delgado-Restituto Conference - IEEE Biomedical Circuits and Systems Conference BioCAS 2014

Curriculum Vitae

Personal information

First name(s) / Surname(s)	Romaine, James
Address	Calle Camino de la valdovina,Castilleja de la cuesta 30, 41950, Sevilla
Telephone(s)	Mobile: +34 654567811
E-mail	jamesbrianromaine@gmail.com
Nationality	British
Date of birth	10.06.1985
Gender	Male

Work experience

Dates	2014-2019
Main activities and responsibilities	PhD researcher (Microelectronics digital/analogue ASIC design and test engineer), Institute of microelectronics Sevilla (CSIC and University of Seville). This position included the design, implementation and verification of low powered and low area consuming medical application specific integrated circuits, for the diagnosis and detection of pathological brain states. This position included a multi-disciplinary approach to work by incorporating, Mathematics and programming using Matlab, python and C++ for algorithmic designs and Electronics and physics for digital and analogue circuit layouts. I have successfully integrated 3 separate integrated circuits. All of which started with the design of efficient algorithms in Matlab which were exhaustively simulated. Secondly, the algorithms were then coded into an FPGA environment using VHDL and Verilog. This stage provides hardware verification, the FPGA designs were then tested using a FPGA to Simulink co-simulation over JTAG connection. Finally, the circuit designs were implemented in CADENCE using RTL-COMPILER and Encounter layout managers. Skills: Matlab, VHDL, Verilog, C++, CAD design, Analogue electronic design, Digital electronic design, FPGA, Mbed microcontrollers, Electronic circuit schematic designs, Electronic circuit physical layout, Mathematical modelling, Data analysis and Arduino. Programmes: Simulink, CADENCE design suite, Encounter, Matlab, Vivado design suite, RTL compiler and Visual studio. Other Skills: Presentations, Group work and publication writing. Responsibilities: Time management for strict circuit integration deadlines, Circuit design, Circuit testing, publication of results and Organisation of workload for group projects.
Name and address of employer	Manuel Delgado Restituto and Ángel Rodríguez Vázquez, Instituto de microelectrónica de Sevilla (IMSE-CNM), Parque científico y tecnología cartuja, calle amercio Vespucio s/n 41902 Sevilla
Dates	2017-2017
Main activities and responsibilities	Design engineer, University college London Working in the electronic engineering department at the University College London as part of a 3-month exchange from the institute of microelectronics Seville. My main activities included: • The manual layout design of an application specific integrated medical device for the treatment of epilepsy. • Organisation of time.

- Presentations of previous work carried as well as the state of the art.
- Mathematical modelling and verification of microprocessors.

Skills: Matlab programming, CAD design, Analogue electronic design, Digital electronic design, Electronic circuit schematic designs, Electronic circuit physical layout, Mathematical modelling, Data analysis.

Programmes: CADENCE design suite, Matlab.

Other Skills: Presentations, Group work and publication writing.

Name and address of employer	Prof Andreas Demosthenous, UCL department of electronic and electrical engineering. Department of Electronic & Electrical Engineering - University College London - Torrington Place - London WC1E 7JE - +44 (0)20 3108 1123
Dates	2013
Main activities and responsibilities	Project (Magnetic Liquids), University of York This project involved the characterisation of several different magnetic liquids with the focus on Magneto rheological fluids. The goal of the project was to determine which fluid would produce the best results for a vehicle magneto rheological damping system. To achieve this goal a set of test apparatus was constructed which simulated a magneto rheological damping system. The test apparatus used a wide variety of skills. Skills: C++ programming, Analogue electronic design, Mbed microcontrollers, Electronic circuit schematic designs, Mathematical modelling, Data analysis, Tunnelling electron microscopy, Scanning electron microscopy, VSM, PCB design, Characterisation and formulating, Magneto rheological fluids, Nano laboratory access and use of specialist Nano laboratory equipment. Programmes: Proteus design suit, Visual studio, Complex Tunnelling/Scanning electron microscopy software. Other Skills: Presentations, Group work and publication writing.
Name and address of employer	Prof. Yongbing Xu, University of York Heslington, York, YO10 5DD, United Kingdom
Dates	2004-2007
Occupation or position held	Satellite technician
Main activities and responsibilities	Installation of satellite systems, Maintenance, Driving, customer advisor, schedule organizer, Sales and accounting.
Name and address of employer	Satellite systems, 7 Rio Vinalopo, La Colina, Javea, Spain.

Education and training

Dates	2014-2019
Title of qualification awarded	PhD Physics
Principal subjects/occupational skills covered	Analogue integrated circuits, Micro and nanometric technologies, Bio inspired algorithms and circuits, Digital and analogue CAD design (cadence), Treatment of information, Electromagnetics, Advanced analogue design, Mathematical modelling and simulation using MATLAB (Including GUI interfaces).
Name and type of organisation providing education and training	Universidad de Sevilla, C/ S. Fernando, 4, C.P. 41004-Sevilla
Dates	2014-2015
Mark	8.1 (0 – 10)
Title of qualification awarded	Master of Microelectronics
Principal subjects/occupational skills covered	Analogue integrated circuits (RF,AMS/RF), Micro and nanometric technologies, Bio inspired algorithms and circuits, Digital and analogue CAD design (cadence), Treatment of information, Electromagnetics, Advanced analogue design, Final project (Design and implementation of phase synchronization algorithm for the early detection of epileptic seizures).
Name and type of organisation	

providing education and training	Universidad de Sevilla, C/ S. Fernando, 4, C.P. 41004-Sevilla
Dates	2009 – 2013
Mark	2:1
Title of qualification awarded	BEng Electronic engineering
Principal subjects/occupational skills covered	Programming software/ Hardware (C, Java, Assembly, VHDL, M-code) Math, Physics, Analogue/Nano electronics, Digital electronics Control engineering, Communications, Project management, Electromagnetics
Name and type of organisation providing education and training	University of York, Heslington, York, YO10 5DD, United Kingdom

Dates	2008- 2009
Title of qualification awarded	Access to Higher Education Diploma Computing, Lincoln College
Principal subjects/occupational skills covered	Computer systems, Networks and communications, Web page design and production, Systems analysis and design, Presentation skills, Advanced database.
Name and type of organisation providing education and training	Lincoln College,Monks Road, Lincoln, LN2 5HQ

Personal skills and competences

Mother tongue(s) English

Other language(s) Spanish

Self-assessment	Understanding		Speaking		Writing
European level (*)	Listening	Reading	Spoken interaction	Spoken production	
	C1 Independent	C1 Independent	C1 Independent	C1 Independent	C1 Independent

(*) Common European Framework of Reference for Languages

Social skills and competences	The ability to work and communicate well with others in order to achieve mutual goals, gained from my work experience. Solid presentation skills, gained from presenting my PhD work in various conferences such as BioCas. I am also a very social person who enjoys social events with work colleagues.
Organisational skills and competences	Organisation of large quantities of work with close deadlines along with self- dedication and discipline gained during university and through work experience. I also have to abide by very strict integration deadlines.
Technical skills and competences	Analogue integrated circuits, Micro and nanometric technologies, Bio inspired algorithms and circuits, Digital and analogue CAD design (cadence), Treatment of information, Electromagnetics, Advanced analogue design, Mathematical modelling and simulation using MATLAB (Including GUI interfaces), Programming software/ Hardware (C, Java, Assembly, VHDL, M-code), Math, Physics, Analogue/Nano electronics, Digital electronics, Control engineering, Communications, Project management, Electromagnetics.
Computer skills and competences	Good control over all Microsoft office programs (Word, Excel, PowerPoint and Access).gained through a qualification from college and work experience.

Driving licence Full, category B

Referees

Current supervisor
INSTITUTE OF MICROELECTRONICS
Manuel Delgado Restituto
Department IMSE
mandel@imse-cnm.csic.es
+34 954466666

Supervisor
INSTITUTE OF MICROELECTRONICS
Ángel Rodríguez-Vázquez
Department IMSE
angel@imse-cnm.csic.es

Additional activities

Teaching experience

- 72 hours teaching circuits and systems at the University of Seville. Teaching gave me the opportunity to help others in the discipline of electronics. It also helped me perfect my communication skills and how to approach new learners to a subject. The practical laboratory sessions allowed me to show students how to build basic circuits and characterise them using theoretical and real values.

Courses

- LabVIEW Core 1, National instruments
- Event-based auditory processing with spiking silicon cochleas and deep network
- Matlab beginner-Intermediate
- Matlab Advanced
- Arduino and robotics

Attended conferences

- PRIME 2016, IEEE Circuits and systems society, 2016 12th Conference on Ph.D.Research in Microelectronics and Electronics (PRIME)
- BIOSENSORS 2016, Elsevier
- IEEE Biomedical Circuits and Systems Conference BioCAS 2014

This thesis introduces a state-of-the-art algorithm and indexing equation for the detection of phase synchronisation in pathological brains. When tested on Epileptic data the algorithm produced results similar to that of more complex methods, such as the Hilbert transform and phase locking value. The proposed algorithm produced an absolute sensitivity of over 80 % for an FP/h of 0.75 and as much as 90% at the upper limit of the 95% confidence thresholds. Furthermore the algorithm allows for nearly a 90% reduction in digital logic when compare to other methods with the total number of flip-flops, LUTs and slices of 112, 60 and 37 respectively.

In addition, a sub-threshold digital VLSI processor was designed. The main processor consits of 15 sub-processors (including dedicated training and threshold calculation modules for patient specific epilepsy treatment and detection) and is capable of calculating the phase synchrony between 9 independent EEG signals over 8 epochs of time totalling 120 EEG combinations. The ASIC was designed in an AMS 0.18μm high voltage, low leakage technology using a fully custom 0.5v threshold digital logic library to reduce total power consumption and operates from a main clock frequency of 2KHz. Each sub processor consumes just 15nW of power.

The core area of the design occupies only 3.64mm^2, whilst the SPI input interface occupies 0.091mm^2 and the output interface occupies 0.063mm^2. Each synchronisation processor occupies 0.04mm^2, whilst, each training and calculation module occupies 0.066mm^2.