UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO - SALAMANCA
DIVISIÓN DE INGENIERÍAS

"Design of a dedicated hardware architecture for biosignal processing"

TESIS PROFESIONAL

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA ELÉCTRICA

Opción: Instrumentación y Sistemas Digitales

PRESENTA:

Ing. Francisco Javier Íñiguez Lomelí

ASESORES:

Dr. Horacio Rostro González

Dr. Yannick Bornat

Salamanca, Guanajuato Noviembre 2018

– Si vous parlez á un homme dans une langue qu'il comprend, cela ira dans sa tête, Si vous lui parlez dans sa langue, cela ira dans son coeur –.

– Nelson Mandela –

Acknowledgements

- Inicialmente me gustaría agradecer a mi madre y hermanos que gracias a su ayuda incondicional siempre han estado conmigo en cada una de las etapas de mi vida, y sin su apoyo, no hubiera podido haber logrado ninguna de mis metas y logros. A mi padre que aunque ya no estás con nosotros, siempre ha sido un ejemplo a seguir para mí y mis hermanos. Sólo espero que sea dónde estés, puedas estar orgulloso de cada una de mis logros.

- Al Dr. Horacio Rostro González, por haber sido mi asesor durante todos estos años en los diferentes projectos en los que he participado desde el año 2014 y hasta ahora en el año 2018 con mi proyecto de tesis. Gracias por toda su confianza y oportunidades que me ha brindado, por hacerme crecer como profesional y despertar en mí el gusto por la investigación.

- A Ana López, por ser mi compañera durante toda esta etapa de mi vida, gracias por estar en mis momentos de flaqueza, por animarme en mis momentos difíciles y llenar de tantas alegrías esta etapa de mi vida.

- A mis compañeros de laboratorio de sistemas bio-inspirados. Abraham Pérez Trujillo, Adan Antonio Alonso Ramírez y Tat'y Mwata Velu, Alberto Patiño.

CONACYT
Consejo Nacional de Ciencia y Tecnología

- Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por otorgarme la beca de estudios de posgrado y beca mixta de estancia de investigación en el extranjero con número 764083 en el Laboratorio IMS en la Universidad de Burdeos en Francia.

UNIVERSIDAD DE GUANAJUATO

- A la Universidad de Guanajuato por haberme aceptado en uno de sus programas de Maestría del Campus Irapuato-Salamanca División de Ingenierías y por el apoyo para la realización del proyecto de investigación y de tesis titulado "Design of a dedicated hardware architecture for bio-signal processing".

- Je voudrais remercier à Sylvie Renaud pour m'avais accepté comme part de l'équipe ELIBIO pendant mon séjour de rechercher au laboratoire IMS dans l'Université de Bordeaux et aussi pour me laisser connaître la culture française et améliore mon français pendant le temps que j'ai été là. Par ailleurs, je voudrais remercier à Yannick Bornat pour tout votre temps que vous m'avez donné pendant mon séjour, merci pour tout ce que vous m'avez enseigné en VHDL et Python, pour clarifier mes doutes existentiels en VHDL, pour la nourriture française, les expressions françaises, les différents accents français. De fond de coeur merci beaucoup. Merci à toutes les intégrantes de l'équipe ELIBIO (Sylvie, Yannik, Gille, Noëlle, Jonathan, Antoine, Loïc, Amélie) pour cet expérience et aussi pour me donner le FPGA Nexys4 et le Pmod OLEDrgb display pour terminer le projet de ma thèse.

  De plus, je voudrais mentionner aux colocataires que j'ai connus a Bordeaux pedant mon séjour . Christian Saint-Jean, Marcela Rodríguez, Krishnanand Shukla, Anabel Argente, Helene Richter et des autres personnes que je ne sais pas comment écrire leur nom. Merci pour toutes les aventures et votre amitié.

Abstract

The extraction of electrical information from bio-siognal recordings is been a field where the Action Potential Detection and Sorting process take place. The analysis when this information is detected is crucial for ***in vivo*** and ***in vitro*** recordings, as well as the time that it takes to detect such electrical information and their analysis in off-line process. In addition, this off-line analysis could take along time and many memory resources. Therefore, we proposed a simple method to achieve Detection and Classification of Action Potentials by using a match filter technique combined with the detection of Correlation Pattern by an adaptive threshold in a hardware architecture.

The architecture was realized in a Field-Programable Gate Array (FPGA) by VHDL language, in order to achieve the goal of real-time performing and be a good candidate in both real-time and off-line process. The architecture consist of a chain module that can be replicated for Micro electrode arrays (MEAs) recordings with some adjusments. The architecture is able to save the sample number of the Action potential detected and sorting this sample in six different cluster depending on the Correlation Pattern detected. This Correlation Pattern is the result of the correlation with the mean Action Potential template and Action Potential template detected at that moment.

The VHDL module have an *Universal asynchronous receiver-transmitter* UART communication protocol to start the process, stop it and read only a sample at the time. An additional feature is added in the off-line mode, where each time an Action Potential is detected, the module stop the reading process in the SD-card and the architecture, and the Action Potential shape, thresholds and Correlation shape can be seen on the OLED display.

The chain module was tested with a Macaque monkey biosignal recorded in vivo at a sampling rate of 40 kHZ and resolution of 16-bits per sample. In the same way the hardware was tested at the sampling rate of 40 kHz (the same sampling rate of the original signal). A second biosignal, a Human pancreatic biosignal, was also tested with a sampling rate of 10 KHz, In addition, an off-line software simulation of the processes was tested in Python 2.7, in order to validate and start the architecture design.

Resumen

La extracción de información eléctrica de bio-señales ha sido un campo donde la detección de potenciales de acción y procesos de clasificación ha tenido lugar en largos periodos de muestreo de señales. El análisis y detección de esta información es crucial en experimentos ***in vivo*** y ***in vitro***, al igual que el tiempo que toma detectar esta información eléctrica y su análisis en procesos fuera de línea. Además, estos procesos fuera de línea pueden tomar mucho tiempo y muchos recursos de memoría. Es por eso que nosotros presentamos un método simple para lograr la detección y clasificación de Potenciales de Acción usando una técnica de coincidencia de formas en conjunto con la detección de Patrones de Correlación por medio de un umbral adaptativo en una arquitectura de hardware.

La arquitectura fue realizada en un *Field-Programable Gate Array* (FPGA) en VHDL, esto con el fin de lograr un desempeño en tiempo real y ser una buena candidata para ambos procesos en tiempo real y fuera de linea. La arquitectura es capaz de guardar el número de muestra donde el Potencial de Acción fue detectado y clasificar la detección dentro de seis diferentes grupos dependiendo del Patrón de Correlación detectado. Este Patrón de Correlación es el resultado de la correlación entre la forma del Potential de Acción promedio con la forma del Potencial de Acción detectado en ese momento.

El módulo de VHDL tiene una comunicación *Universal asynchronous receiver-transmitter* (UART) para empezar el proceso, pararlo y leer sólo una muestra a la vez. Una caracteristica adicional es agregada en el modo fuera de línea, donde cada vez que un Potencial de Acción es detectado, el módulo para el proceso de lectura en la tarjeta SD y en la arquitectura es detenido, y la forma del Potencial de Acción puede ser observada sobre el OLED display.

El módulo de canal fue probado con una bio-señal de Mono Macaco grabada en ***in vivo*** a una frecuencia de muestreo de 40 KHz y con una resolución por muestra de 16-bits. De la misma forma la arquitectura en hardware fue probada a una frecuencia de muestreo de 40 kHZ (la misma tasa de muestro que la seal original). Una segunda bio-señal fue probada, una bio-señal de Pancreas humano, a una frecuencia de muestro de 10 KHz. Además, una simulación fuera de línea del proceso fue probada sobre Python 2.7, para validar y comenzar con el diseõ de la arquitectura en hardware.

Contents

List of Figures

List of Tables

– It's only those who do nothing
that make no mistakes, I suposse
–.

– Joseph Conrad –

1 Introduction

The electro-physiology is the study of the electrical properties of biological cells and tissues that measure and analyse the changes of voltages or electric currents such as the Action Potentials (APs) in neuroscience, which are spontaneous electrical activities collected through an electrode or a network of electrodes. These electrical activities can come from neural population as well as the interactions between neurons. Multi-cell recording is a commonly used technique to extract this electrical information from cell-cultures and tissue slices in order to find treatments for human diseases and disorders. Other electrical information during biosignal recordings with lower frequencies than APs are known as Local Filed Potentials (LFP) and Slow Potentials (SPs), which are groups of hundreds of excitable cells exhibiting continual oscillations during the recording process of biosignals.

Nowadays the bio-electronics field is addressing projects where is important to understand the behaviour of living beings through their electronic information as known as action potentials (APs) and produced by neurons, muscle cells and some endocrine cells as pancreatic cells. The principal application for APs detection is to collect the important electrical information coming from the biological signals and to know how to stimulate properly the biological tissue or cell-culture by an electrical stimulus evoking field potentials in a specific region, for instance, studying the spontaneous activity in a neuronal population, using drugs in cell-cultures or brain slices that can generate spontaneous potentials, which can be recorded to know the drug effects. Furthermore, the toxic effects of the drug need to be tested *in vitro* before their application *in vivo* animal recordings, due to ethical procedures. Additionally exist the Deep Stimulation Brain (DSP), which is one the most common surgical treatments for Parkinson's disease for the patient with motor complications, reducing the symptoms and medication requirements. Besides, in the last years, it has been presented several projects, where the user controls objects or robots by human thought or handicapped people is able to control their wheelchair, as well as the prosthesis for legs, arms or the cochlear implants that stimulate the auditory nerve. All these applications come from the analyses of biosignal recordings, which ones are classified as Extracellular and Intracellular recordings depending on their nature.

The Multi-cell recording is normally performed using multiple electrodes these

multi-electrodes are commonly known as Multielectrode arrays (MEAs), standard MEAs have numerous electrodes commonly made of titanium nitride and these arranged in an 8 x 8 or 6 x 10 configuration on a glass substrate. Implantable MEAs are used actually *in vivo* recordings and non-implantable MEAs *in vitro* recordings. *In vitro* comes from the Latin *"within the glass,"* while *in vivo* comes from *"within the living"*. How their name says *in vitro* experiments are developed with cell-cultures or tissue slices, while *in vivo* experiments with the whole living organism and the electrodes are inserted in the living tissue.

The Action Potentials Detection (APD) involves the study of many types of brain functions, as we know most of the neurons in the brain communicate by firing electric signals, these signals are APs, which are brief voltage pulses appearing in bursting patterns. The APs could belong to one or more neurons or cells and detect their patterns could be challenging, given that there is no shape pattern to follow and to compare with new detections during biosignal recordings, therefore to carry out a sorting process to know which APs belong to each neuron or another cell involve a challenging assignment. Moreover APD could be challenging when certain features in the recording process are presented, as the high amount of background noise, on top of that neurons in a local region tend to have APs with similar shape and size which complicate more the sort process, furthermore APs coming from the same neuron could have the same characteristic shape, but their height could be affected if there are other neurons in the local region with APs of considerable size and similar firing rates, this could happen when two or more neurons or cells are firing simultaneously, one such example is when the peak of the present Action Potential (AP) and the dip of the next AP coming from other neuron occur at the same time, then the APD could be missed, and we would have several overlapping APs.

The thesis is divided in five chapters. First, we briefly review some biological concepts and the electro-physiology history, in order to understand the biological context of this thesis. In chapter 2 we discuss the threshold detection researches and some classification methods of APs. The third chapter is dedicated to the background of the adaptive threshold and the classified method for APs, as well as the methodology used. In the fourth chapter, we present the hardware architecture designed. In this thesis Finally, in the fifth chapter we present the simulation results and tests on the hardware architecture designed, as well as the detection and classification results of APs, using different threshold levels, as well as the conclusions and future works.

1.1 Justification

One of the main problems associated with Action Potential Detection is the background noise, which is caused by biological noise coming from the biological environment, moreover the electrical components by their nature also add certain noise to the biosignal during the recording process by Micro-Electrode Arrays (MEAs). These could hinder the Action Potential detection, so an appropriate treatment is needed. Therefore, we are looking for an algorithm and its implementation on a FPGA to properly perform the detection of these Action Potentials. The use of a FPGA device will allow us to increase the speed in the detection of the desired Action Potentials, saving time for its subsequent analysis.

1.2 General objective

Field Programmable Gates Arrays (FPGA) design and hardware implementation of an architecture for filtering and detection of Action Potentials in macaque and pancreatic biosignals recorded by Multielectrode arrays (MEAs).

1.3 Specific objectives

- Numerical simulation and validation of an algorithm for action potentials detection.
- Estimation of optimal parameters to enhance the algorithm with action potentials from macaque and pancreatic biosignals.
- Design and implementation of a hardware protocol to read the biosignals from an SD card.
- Design of hardware components needed to implement the architecture for filtering and action potential detection over VHDL language.
- Estimate and store the time of each detected action potential from the biosignals for further processing. Saving the detection time of each of the action potential detected into the biosignal.
- Design of a hardware protocol for biosignal visualization through an OLED display

1.4 State of the art

The are many common algorithms and methods that are used to achieve detection and sorting of Action Potentials. Basically, all these methods have the same goal, to accentuate the Action Potentials or get a better relation Signal-to-noise radio (SNR), to detect the Action Potential by a threshold that can be computed of many ways. After, the feature extraction of the Action Potential shapes, and finally some way how to cluster this detection depending on their origin that could be the same cell that generate the Action Potential or other cell close to the electrode that recorded the biosignal.

Here, we present some of the recent architectures, methods and algorithms tested in real-time and in off-line process:

Hardware implementation for detection and sorting models has been presented in Gibson et al. (2013). Here, a FPGA architecture was designed to achieve detection and sorting of Action Potentials in both real-time and off-line. It was achieved in a FPGA Xilinx Virtex-5. This system is able to be configurable with several methods, such as detection by absolute value or nonlinear energy operator(NEO). For the alignment of shapes can be selected maximum value, minimum value, absolute value and NEO maximum. Finally, clustering by the Osort method. These methods can be selected by the use of a Matlab and Python scripts.

Commonly, algorithms for Action Potential sorting are composed of three parts, The Action Potential detection, extraction of some features for Action Potential shapes and clustering from its features. In Takekawa et al. (2014) is proposed a new method using probabilistic tools by detecting and clustering, the different Action Potential in biosignals, where by using the Bayes theorem compute the probability of a sample belongs to an Action potential. This is performed by taking into account the distribution nature of Action Potentials in amplitudes, widths and frequency. This method is useful for analysis of extracellular recordings with linear probes but no hardware implementation took place.

Other systems are made over CMOS technology as Barsakcioglu et al. (2014), They developed an Analog Front-end system for neuronal Action Potential sorting. They use common tools for Action Potential Detection and sorting methods, as well as the common methods: Template Matching (TM), Principle Component Analysis (PCA), First and Second Derivative Features (FSDE),with the use of a Graphical user Interface (GUI) implemented in Matlab.

The CMOS technology has been applied also for detecting Action Potentials, with the use of wavelet coefficients.Yang et al. (2015) made a prototype in a FPGA for 16 channels for Action Potential detection, by applying the use of Stationary Wavelet Transform (SWT), this prototype was afterward mapping in a 130 nm CMOS technology using the lifting wavelet transform, which is a faster implementation in hardware than the common SWT.

Pirog et al. (2015) propose a versatile module in VHDL, which is part of a hardware architecture dedicated to biosignal processing, where, the use of wavelet filters, IIR filters, Action Potential Detection and Slow Potentials takes place. the modules is able to automatically detect certain electrodes where there is not important information excluding this electrodes for the recording process, taking into account their frequency, amplitude of the events and synchronous activity. This module was implemented on a Xilinx Spartan-6 FPGA.

Some other Action Potential classifiers were tested as: Super-Paramagnetic Classification (SPC), Osort , K-means, Moving Centroid K-Means, Moving Centroid K-Means, Hierarchal Adaptive Means (HAM), Fuzzy C-Means (FCM), Mahalanobis classification, Support vector classification (SVC), Self-organizing maps (SOM) and Cosine Similarity classification. Here, Saeed et al. (2017) present a good comparative of existent classifier architectures for implementation in real-time of Neural Action Potential sorting. They concluded that the best classifier is the SOM classifier, when it is taking into account accuracy and complexity. This classifier is based on the use of an Artificial Neuronal Network (ANN). Although we know that the implementation in hardware of classifiers based on an (ANN) consume a lot of memory resources because of the creation

of Lookup tables.

Other methods have been developed in the lasts years, such as the Time-frequency based convolution spike detection algorithm (TIFCO) and the Stationary wavelet based TEO(SWTTEO). The first one is based on the time-frequency of Action Potentials in the range of 500 Hz and 3500 Hz. The second one is based on a low-pass filter using the Discrete Wavelet Transform (DWT) and applying the Teager energy operator (TEO) to each filter sub-bands Lieb et al. (2017). The implementation was made over Matlab.

1.5 Biological Context

In this section, some interesting and important concepts about biology are addressed, as well as some historical events that took place, to achieve which is nowadays known as the Electro-physiology.

1.5.1 First Electro-physiology Experiments

In 1791 Luigi Galvani, a professor from the University of Bologna, discovered the nerve conduction and muscle contraction. Carrying out experiments with frogs (De Viribus Electricitatis in Motu Musculari Commentarius),Galvani (1791)). His results surprised the science community. One of the Galvani's experiments consisted in link a long metallic wire to a frog nerve, locating the wire in a high point of his house. Then, Galvani was waiting only for a storm that was coming to him. After, some flashes of lightning, Galvani observed that in the frog legs some contractions occurred (Figure 1.1).

Subsequently, Galvani performed the experiments on a clear and sunny day, but nothing happened. Then, Galvani, after his defeat, played with the wire connected to the frog nerves, pressed and pushed the wire with an iron railing, and the contraction returned. Finally, Galvani repeated the experiment in a closed room, locating an iron plate this time, and one more time the frog leg contractions appeared. So, Galvani came to the hypothesis that the cause of it was not the atmosphere electricity, otherwise the existence of an intrinsic electricity coming from the animal, which he named "animal electricity" Piccolino (1998),Piccolino (2008); Bresadola (1998).

Alessandro Volta, one of the first scientists that repeat the Galvani's Results from the University of Pavia, was doubting about Galvani's hypothesis. Volta believed that the origin of the frog leg contractions, were caused by the electricity that was conducted by the metals, using a bimetallic arc and connecting it to two points of the frog leg nerves, getting the frog leg contractions for the difference of the metals, like the response to an external electricity, Volta later used this concept to generate his electric battery. How can we think about it ?. A big controversy was opened between Galvani and Volta,(Piccolino, 2008), (Figure 1.2).

Another contribution was performed for the Galvani's nephew. Giovanni Aldani a physicist from the University of Bologna, who since 1782 worked as a research assistant for Galvani. Aldani, in addition, conducted a serial of experiments with birds, lambs, calves and oxen Figure 1.3. One of the most mentioned was the application of electric current to an ox brain, having good results, and stimulating several parts of the brain. Afterward, Aldini was thinking that the brain stimulation could be a good practice for therapeutic procedures. Aldini, In 1802, in Bologna, stimulated three human bodies,

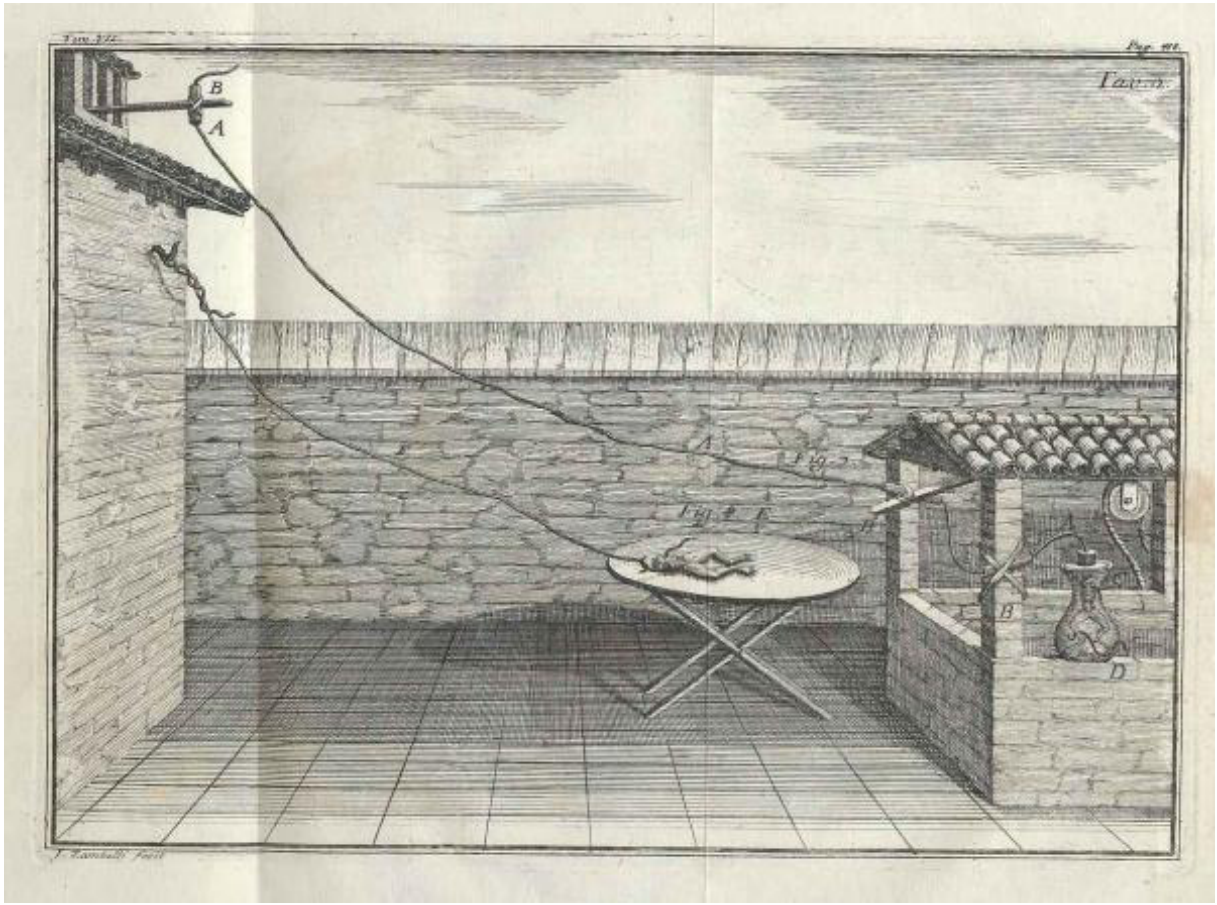


Figure 1.1: Galvani's experiment in a stormy day. A long wire is connected to the frog nerve muscle, the other circuit is a Leyden Jar, which was a capacitive devices during the 18th centuary. Ilustration from: Galvani (1791)

which were from three criminals that had been executed an hour before. The stimulation of the three bodies was achieved, having good muscular contractions during the seasons. Morover, Aldini used one of the Volta's devices, the bimetallic pile, in order to convince the scientist community that the stimulation was a good procedure as a therapeutic tool Parent (2004).

1.5.2 The First Electro-physiology Steps

Carlo Matteucci (1811-1868), also from the University of Bologna, using the Galvani's technique the galvanism, measured the "animal electricity" from the muscles of frogs and other animal preparations. The Matteucci's results gave the possibility to measure from pure muscle preparations, putting on one of the extremes of the galvanometer the intact part of the muscle and on the other one, the cut side. Afterwards, Matteucci cut several frog muscles, and he linked them in manner where he collocated the pieces as a pile, where the intact part of the muscle was connected with the next cut side of the other preparation. He detected in the galvanometer a current increase when more preparations of frog muscle was added to the pile, recording the current between the cut and intact parts of the muscle is due to the potential difference between the interior and the exterior of the muscle fibers. Finally, with this result Matteucci finally ended with the controversy between Galvani and Volta, giving one of the first steps for the electrophysiology,(See Piccolino and Wade (2012)) for Illustration experiment).

Figure 1.2: Galvani and Volta, and their hyphothesis. Ilustration from: backyardbrains

Du Bois-Reymond (1818-1896), was the first to discover the action potential, using a sophisticated galvanometer that he built. He detected a flow of charge that was presented in the muscular and nervous tissue,assumming this observation, as the "resting current", Finkelstein (2006). The Bois-Reymond's experiments consisted in stimulate the muscles and nerves, causing that the resting current almost disappeared and even it could be negative, confirming the Matteucci's observations Navarro (2013). Nowadays this current is known as the action current. Along with thier achievements, Du Bois-Reymond is considered the father of the electro-physiology Pearce (2001).

Later in 1952, Hodgkin and Huxley proposed a model, where they could predict the shape of action potentials, their studies were on a squid giant axon. Moreover, the model was able to know the parts that evoked the action potentials Catterall et al. (2012). Action potentials arise from the synergistic action of sodium channels and potassium channels, each of which opens and closes in a voltage-dependent fashion. A key feature of their model is that the channels open independently of each other; the probability that a channel is open depends only on the membrane voltage history" from: Colwell and Brenner (2009). The model is part of a set of non-linear differential equations, whose propose is to explain the features in excitable cells. The Hodgkin's and Huxley's results presented the concepts of Action potentials, refractory period and the threshold, explaining how by electrical excitability, the action potentials are generated Baravalle et al. (2017).

The advances were growing through the time and the electrophysiology applications at the same level. The Deep Brain Stimulation (DSP) for Parkinson's Disease and Dystonia, lowing the symptoms and medication requirements, using a pacemaker, sending electrical stimulation within the brain. Some treatments for obesity, obsessive-compulsive disorder

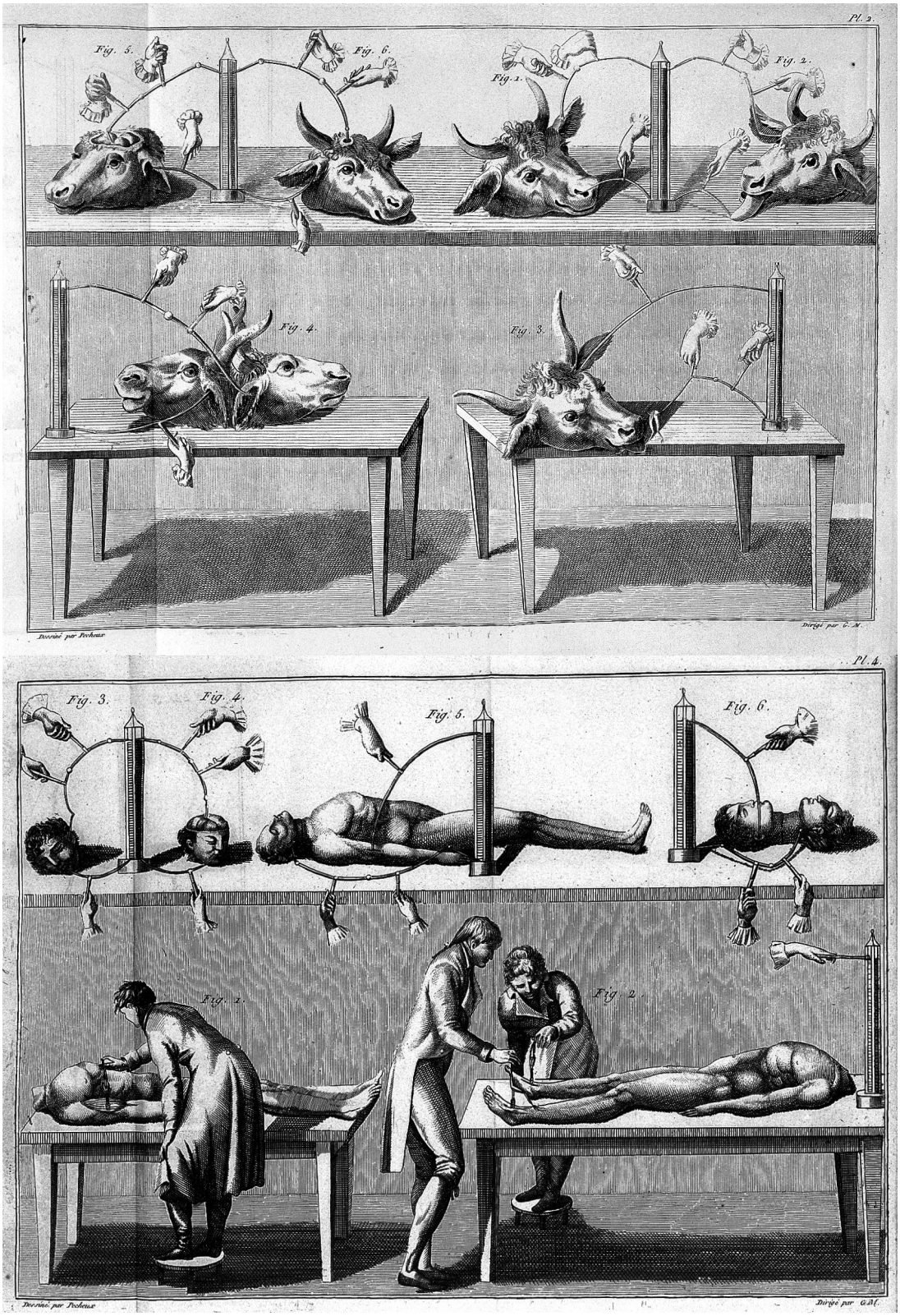


Figure 1.3: These were some stunning pictures of Aldini's experiments. Oxen heads (upper Ilustration), Human bodies (lower Ilustration). Ilustrations from: Aldini (1804)

and depression Gardner (2013). The biosignal recording for medical diagnostics, as the brain, using the electrocardiography, by the detection of epilepsy or determinate the dead of a patient in a deep coma. The electrocardiogram, by cardiac problems or pulmonary diseases Rummens. (2015).

1.5.3 The Nervous System

The nervous system generally control and send signals through our body, by coordinating and integrating the signals of our sense and organs. It is sending and receiving signals through the whole system. Always when we hear about the nervous system, we associate him with the brain, nerves and their cells, maybe the most know cells, the neurons. The mean division of this system is the central nerve system (CNS) and the peripheral nerve system (PNS) Brown (2001). The Fist one CNS ,whose components are the encephalon (brain, cerebellum and the brain stem) and the spinal cord. In the other hand, the second one (PNS) is composed by the nerves that connect the (CNS) with organs of our body and other places, whose parts are divided in the peripheral nerve system autonomous (PNSA) by involuntary moments as in organs. Finally, the peripheral somatic nervous system (PSNS), which is associated with the voluntary movements, by skeletal muscles Herculano-Houzel (2012).

1.5.3.1 The Neuron

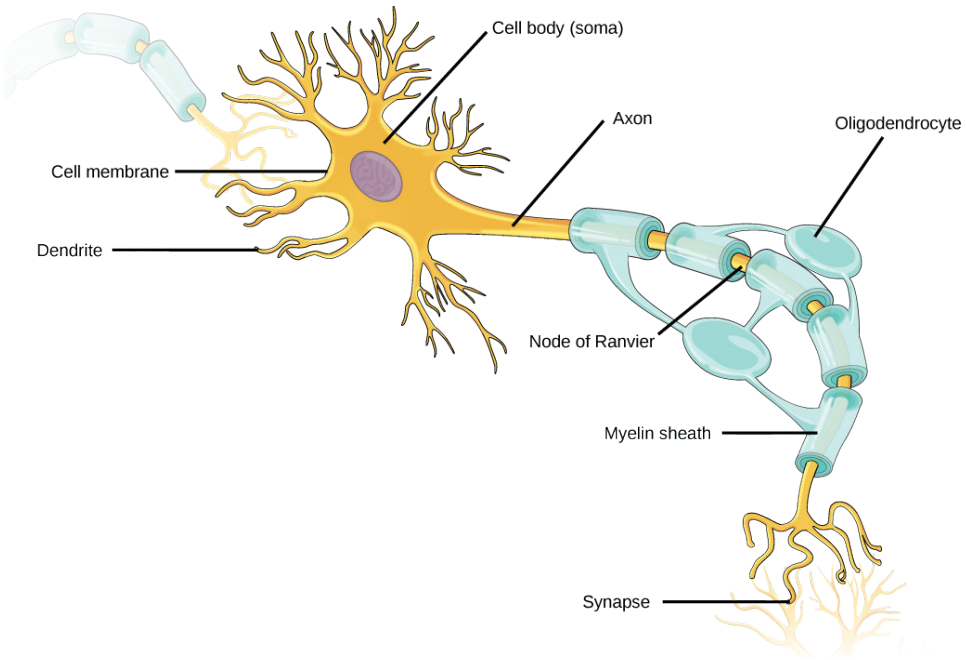


Figure 1.4: Anatomy of a neuron . Ilustration from: Wikimedia Commons

The neurons are excitable cells that transmit and receive information. The neurons are components of the nervous system, their main function is to receive, process and send information, by electric and chemical signals through connection known as synapses. Neurons are consisted of several parts:

- ***Soma***. An spherical central part of the neuron. Where are located the genetic material and organelles, the most part of the neuron's body. It contains the nucleus where most protein synthesis occurs. The soma process the information that it receives, and if a nervous signal is generated, it send the information to others cells or organs by the axon.

- ***Dendrites***. By the dendrites the information is collected from other cells. they are the inputs to the soma , where after the electric signals and chemical signals are processed.

- ***Axon***. The axon is covered for myelin, which help the propagation of the nervous impulses sensed by the soma. Axons could have extensive branches that are connected to other cells thorough the body.

1.5.3.2 The Synapse

The synapses are one of the important connections in the nervous system, by the synapses is transmitted the nervous impulse between neurons or muscle cells connected by the axon's branches. It can be stablish more than 50,000 connections to other neurons, Kolb and Whishaw (2014).

The electric information coming from neurons is transferred to one cell to the other, by the chemical process of the synapses. They can amplify o reduce a signal that is sent from one neuron to the other.

In the order hand, the electrical synapses, when one neuron is connected with the other, and the communication is through a gap junction, where the pre-junction and post-junction cell membranes are connected, The ion channels of both neurons allow ions to pass to one neuron to the other. This type of synapses are found in mammalian brains, and they are faster than chemical synapses. By the way the chemical synapses are commonly connected with other cells, as muscles and glands.

Although, there are several types of synapses, as dendrodendritic, axodendritic, axoextracellular, axosomatic, axosynaptic, axoaxonic, and axosecretory. They are classified in excitatory and inhibitory synapses, each synapse is located in different parts of the body. Excitatory commonly located in shafts or the spines of dendrites, inhibitory synapses are typically located on a cell body. So, the neuron is divided in two zones an excitatory dentritic tree and an inhibitory cell body Kolb and Whishaw (2014).

1.5.4 Electrical Activity in the Nervous System

The neuron have intracellular and extracellular fluids, whose ions are positively charged , as Na^+ (sodium) and K^+ (potassium) and to other side the Cl^- (chloride) negatively charged. Also it could contain protein molecules charged negatively A^-. Positive ions are called *cations*, and negatively charged *anions* . The main factors that create electrical charges in anions and cations are : difussion, concentration gradient, and charge (Voltage gradient).

- ***Diffusion.*** Molecules are always in constant movement looking for the equilibrium in a solution. They always go to the parts where there is less concentration. So the movement of molecules go for places where there is more concentration to less concentration, this effect is called diffusion. When we have the same number of molecules or almost equivalent, in each place.

- ***Concentration gradient***. How ions have a certain charge, in order to achieve the charge equilibrium, they repeal each others. So, the concentration gradient tell us, the tendency of ions movement.

- ***Voltage gradient***. It is the difference of charge in the solution for a type of anions and cations. As we know the voltage is the difference of potential in two places that we can measure, just as voltage gradient tell us the difference of potential in the parts of the solution, where the measure takes places.

Commonly, this factors generate the electrical activity in the cell's membrane that convey information to the nervous system. The movement of ions electrically charged in constant movement or in repose,to determinate the electric charge that can be sent to one neuron to the next Kolb and Whishaw (2014).

1.5.4.1 Resting Potential

The resting potential is generated in an axon's membrane, when we are measuring it by an electrode. The measure is conducted when one extreme of the electrode is collocated in the inner part, and the second one, in the outer part of axon's membrane. The charge measured is commonly - 70 millivolt, when the axon's membrane is not stimulated during the recording. The membrane's resting Potential can vary from different animals between -40 to -90 milivolts Kolb and Whishaw (2014).

The common particles or ions in charge of this resting potential are Na^+, K^+, Cl^- charged negatively and protein molecules A^-. These are the cations and anions. These charged particles are distributed unequally across the cells membrane, Ka^+ cations and A^- anions in the intracellular fluid part, and Na^+ and Cl^- in the extracellular part of the cell's membrane Kolb and Whishaw (2014); see Figure 1.8.

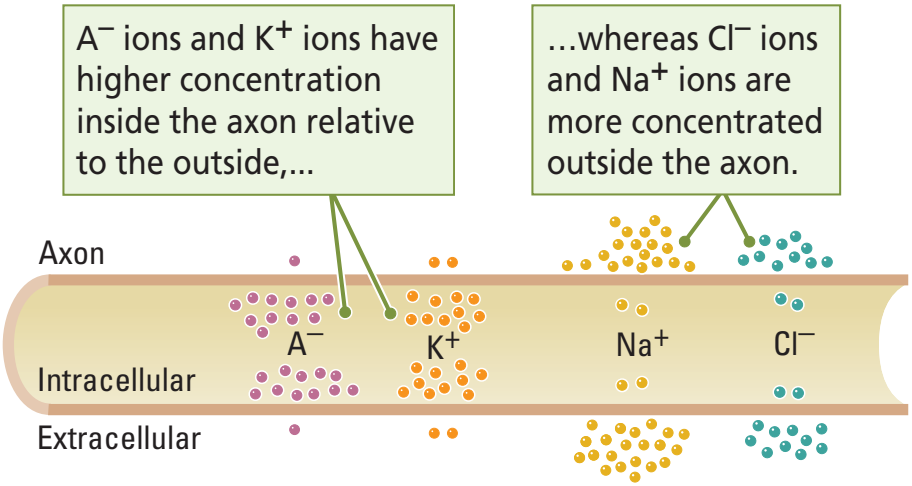


Figure 1.5: Ions distribution in the extracelullar and intracellular in the axon's membrane. Ilustration from Kolb and Whishaw (2014)

The parts that maintain the resting potential are the pumps, gates, and ion channels. Frequently, into the intracellular part of the cell's membrane, the K^+ channels are open, cause of potassium concentration into the intracellular is higher than in the extracellular part of the membrane, this concentration is controlled by the potassium concentration gradient, against the large ions of A^- into the intracellular part. But, there are not sufficient K^+ ions into the intracellular part to beat the large protein molecules A^-. Therefore, the intracellular part of the membrane maintain a negative charge by the majority of large ions of A^-.

By the way, in the extracellular part also exist K^+ ions. but, in a fewer proportion, how we mentioned above, this proportion is controlled by the voltage gradient and the potassium concentration gradient. Now the resting ions, Na^+ and Cl^-, also contribute

to stablish the resting potential, the difference is that the Na^+ channels are commonly closed, stopping the Na^+ ions to cross to the intracellular part of the cell's membrane. But, when the resting potential is altered, the Na^+ pump is activated. This pump is a protein molecule in the cell membrane, the pump activation interchange two K^+ for three Na^+. The Cl^- cations, in a fewer proportion also contribute to stablish the resting potential of the cell's membrane. but, their concentration is lower than the Na^+; see Figure 1.6.

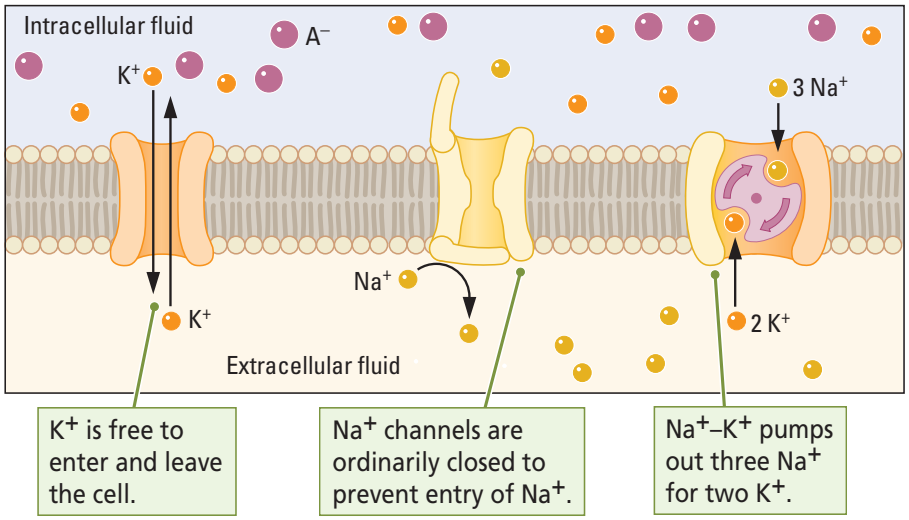


Figure 1.6: Channels, gates, and pumps in the cell membrane. Ilustration from Kolb and Whishaw (2014)

1.5.4.2 Graded Potentials

How was mention above, the cell's membrane store a resting potential. this energy could change depending on the ions concentration in the intracellular and extracellular part of the membrane. When this change happens, the graded potentials are generated. This change is stimulated across the axon's membrane of the cell with a electrode. When a positive charge is stimulated through the electrode the negative charge decreases from -70 milivolts to a lower potential of -65 milivolts , while if a negative charge is applied, the negative charge increase to -73 milivolts Kolb and Whishaw (2014). The fist event is called ***hyperpolarization***. Conversely, the second one is called ***depolarization***.

Hyperpolarization is generated when K^+ anions pass to the extracellular part or Cl^- anions pass to the intracellular part. By other hand ***Depolarization*** is generated when Na^+ anions pass to intracellular part Kolb and Whishaw (2014).

Commonly, these events occur in the soma, which is the cell body, as well as in the dendrites of neuron. This areas have ions channels that can modify the concentration in the cell's membrane: potassium, chloride, and sodium ion channels.

1.5.4.3 Action Potential

An Action potential (AP) is a brief voltage pulse that can lasts about 1 millisecond. Action Potentials occur when the axon membrane is electrically stimulated and depolarized

to about -50 milivolts Kolb and Whishaw (2014). This stimulation cause that the concentration of ions changes in the membrane, producing a positive charge in the intracellular side to respect the extracellular one, after the concentration of ions changes to return the resting potential.

When an Action Potential happens the next phases occur:

1. ***Resting potential***. During the resting potential, one of the two Na^+ gates is open, but not a single Na^+ ions are crossing the membrane to the other side, as well as the K^+ gate.

2. ***Depolarization.*** By the stimulation, the -50 milivolts threshold is reached and the Na^+ open first than the K^+ gate, by their voltage sensibility, producing a Depolarization, and no axon stimulation is able to generate a new Action Potential, and any axon stimulation is able to generate a new Action Potential, because of the membrane stay as absolutely refractory.

3. ***Repolarization.*** The first Na^+gate was opened but almost at the same time the second gate is closed. Now the Repolarization start, and the K^+ gate is opened. The membrane stay absolutely refractory yet; new action potential is not possible.

4. ***Hyperpolarization.*** The K^+ gate is keep opened, and the second Na^+ is opened too, and the first one is closed, the membrane is now relatively refractory, and a new action potential could happens if the stimulus is higher than the last one, a new action potential is possible when the resting potential is reached one more time.

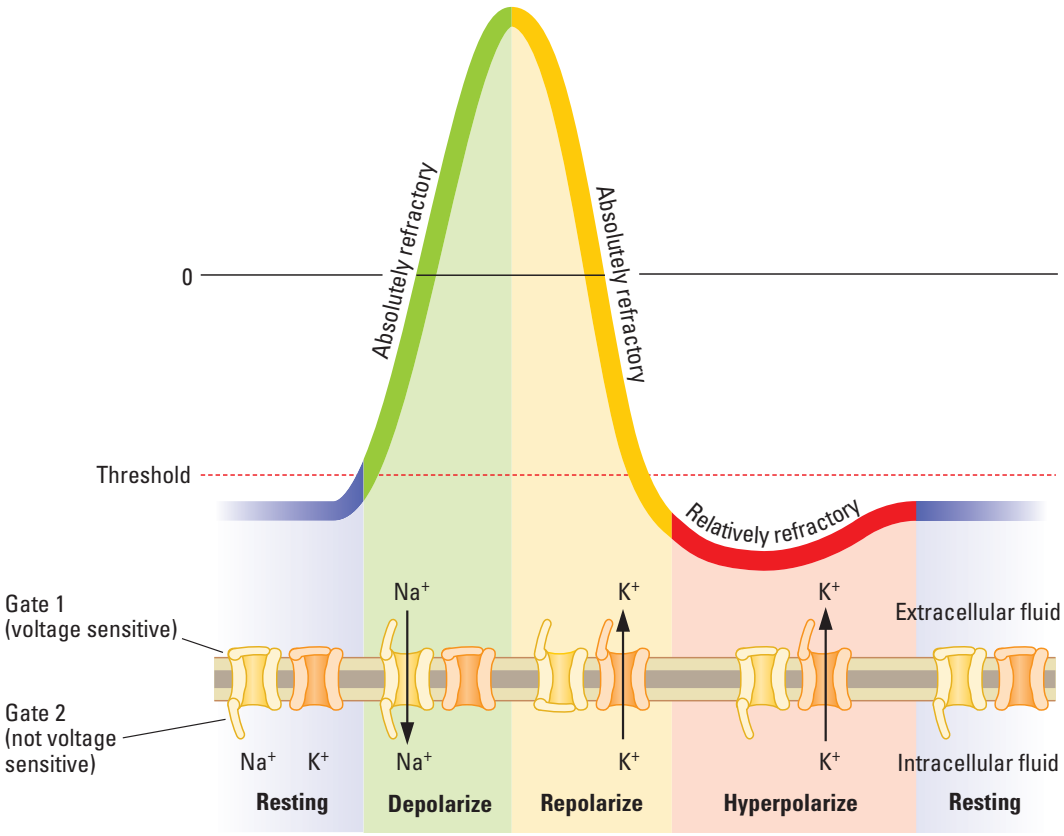


Figure 1.7: Action Potential Phases. Ilustration from Kolb and Whishaw (2014)

1.5.5 Electro-physiology Recordings

1.5.5.1 Intracellular Recordings

The intracellular recordings measure the potential difference between the internal and external part of the cell, where the action potentials are present in the change of ions concentration Rummens. (2015).

The cell-attached patch is performed using a micropipette and sealing it with the cell membrane, by apply positive pressure before inserting the micropipette into the bath, once the electrode tip is close to the neuron, the positive pressure is removed and a suction (negative pressure) is applied to the recording electrode. This procedure is a common first step in many intracellular recordings, only in sharp-electrode recordings is not implementedDong and Graziane (2016).

Generally four techniques are used in intracellular recordings:

- ***Whole-cell***. There two methods. The first perforated patch, which uses substances that form channel pores in the cellular membrane. The second one is using a negative pressure applied for a recording electrode after achieving a cell attached patch. this negative pressure fissure the lipid membrane allowing the recording for the electrode.
- ***Outside-out***. When the recording electrode is moved away in a whole-cell, only a few millimetres for the neuron, this event form a thin fiber that after is broken. Then, a micro-cell is formed, resealing at the tip of the recording electrode.
- ***Inside-out***. From a cell-attached patch the recording electrode, which is filled with an external solution, is moving away leaving a vesicle. The most common way to remove the vesicle is exposing it to the bath-air interface, by lifting the recording electrode out of the bath solution.
- ***Sharp electrode***. Get their name from the recording pipette that is fabricated for cell impalement.

For more information about the advantages and disadvantages of each technique and their application, please see Dong and Graziane (2016) pp. 9-14.

1.5.5.2 Extracellular Recordings

Extracellular recordings in neuronal tissue are generated for the flow ions through the cell membrane. This type of recordings, on the contrary to intracellular recordings, are developed by the observation of the cells activity, without the insertion or perforating of the cell membrane, in most of the cases. Extracellular recordings look for almost not to perturb the cell's activity. Here, the electrodes are put close to the cell but this distance affect the measure of electrical activity, due to intracellular recordings, we have the reference of the intracellular against the extracellular part measuring the potential difference. By other hand, in extracellular recordings the intracellular reference is not possible, so the record measure of the electrode is the ions flow throughout the membrane, this flow perturb the electric field, recording the electrode a little signal about the microvolts, this action potentials that is recorded, is commonly softened and distorted Rummens. (2015).

The most common Extracellular recordings are:

- ***Electro-encephalogram (EEG)***. The brain activity is recorded by electrodes put on the scalp. This electrodes detect the potentials gradients through the time and the tiny potentials over the scalp, using a reference electrode.
- ***Electrocortigram (ECoG)***. This procedure is an invasive recording procedure where the electrode is put on the subdural layer on the cortical surface. To detect Field potentials commonly generated for cortical pyramidal cells.
- ***Local Field Potential (LFP)***. Local Field Potentials are record with a microelectrode put it in brain, the electrical activity comes from a population of neurons .
- ***Whole cell***. This type of recording uses a blind patch clamp approach for in vivo whole cell recordings. One important point to mention, is that with this method the neurons of interest cannot be seen. On the contrary, using the resistance pipette, can indicate if the pipette is close or touching a cell. Using a square voltage step when the micropipette is entered into the brain with a positive pressure, and when the square wave current is reduced the positive pressure is changed for suction (negative pressure), similar to intracellular recordings. This type of technique is useful in regions of high cell density.

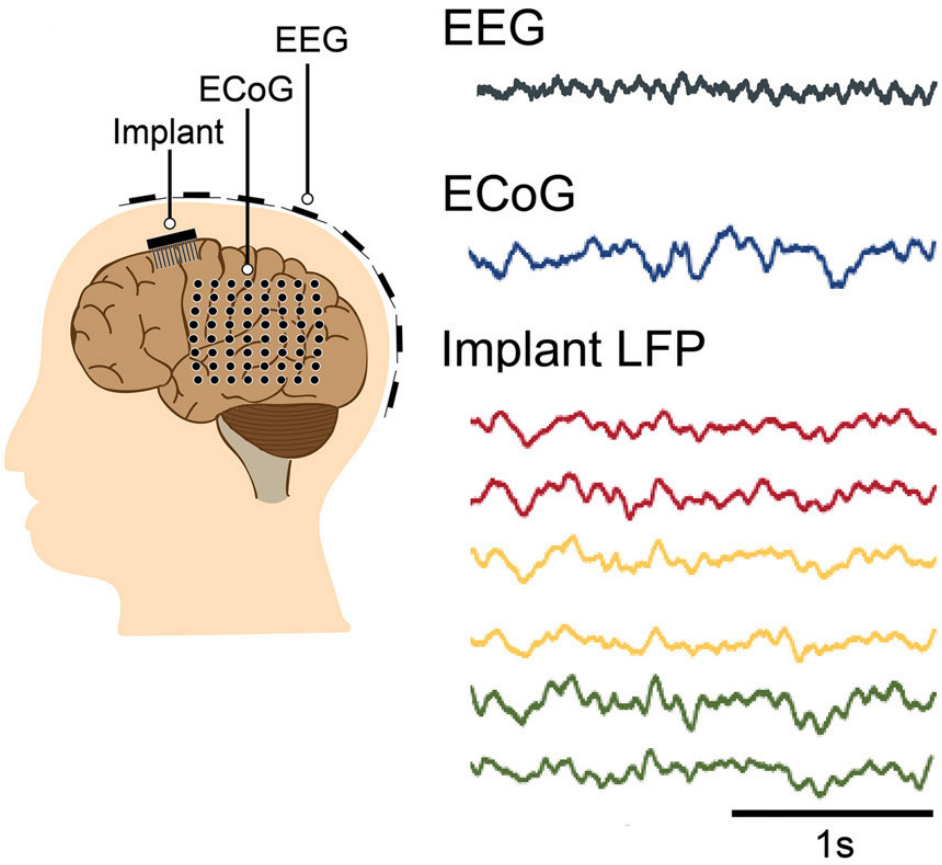


Figure 1.8: Some extracellular recordings methods, Electro-encephalogram (EEG), Electrocortigram (ECoG), Local Field Potential (LFP) . Ilustration from Obien et al. (2015)

The recording in anaesthetized animals is a invasive in vivo recording, which the animal is immobilized, and where a single neuron activity could be recorded up to 3 hours and in

some cases up to 12 hours Wolfgang (2007). The process is by drilling a hole in the skull, by passing the micropipette up to the cell, recording field potentials, as well as whole cell measurements. Some of the aspects that take place is the appropriate pain relief and the duration of immobility. In addition, vital signals and artificial respiratory process, as well as blood pressure and body temperature, have to be monitored Dong and Graziane (2016).

By other hand, freely moving animals recordings, have a similar process to perform but in this case the micropipette has to be attached, and after the anaesthesia has passed, the animal is able to move freely to the stimulus Lee et al. (2006); Dong and Graziane (2016). The stimulation is made with virtual reality when the mouse is running in a spherical treadmill with the head fixed, while the electrical activity is recorded; Dong and Graziane (2016), Harvey et al. (2010).

1.5.5.3 MEAs

Micro electrodes arrays (MEAs), which are commonly made of titanium nitride (TiN), platinum (Pt), stainless steel aluminum (Al), gold (Au) and alloys like iridim oxide (IrOx) Obien et al. (2015), are arranged in 8 x 8 or 6 x 10 configuration on a glass substrate Dong and Graziane (2016).

Depending on the goal of the biological experiment, for instance: in vivo, in vitro, culture or acute preparation, or the type of recording, such as Extracellular Action Potentials, Local Field Potentials, or Intracellular Potentials, cell resolution or other, the microelectrodes have to be selected. In addition, a low electrode impedance is also important. Commonly, a 5:1 or higher signal-to-noise ratio (SNR) Obien et al. (2015).

There is a large amount of MEAs that are classified for different features, as type of transducers used: multi-transistor array, microelectrode array, multielectrode array, micro-nail array, capacitive-coupled array, 3D MEA. For the type of substrate: active array, passive array, silicon array, CMOS array. The shape of the device: needle-type probe, polytrode, neuro dish. The channel count: multichannel array. The electrode density: the electrode density: HDMEA. The application: implantable array, in vivo MEA, in vitro MEA Obien et al. (2015).

One of the approach for recordings with MEAs is their use in experiments, where is needed extract information in real-time.Such as closed-loop experiments and brain machine interfaces (BMIs), where stimulation therapies are taking part of this features, and fast analysis is required Obien et al. (2015). Also, this type of recordings give, after the analysis of experimenters, the relationship between oscillations, as Local Filed Potentials, Extracellular Action Potentials and Intracellular Action Potentials with different brain states. MEAs are commonly used on cell-cultures and tissue slices, this MEAs can record action potentials, evoked field potentials or spontaneous field potentials in a certain population of cells or between neurons. Dong and Graziane (2016). The advantage of using this type of electrodes are that we can measure the activity simultaneously in certain neuronal location, and also is easier to change for recording to stimulation within the neuronal preparation or cell-culture, helping the study for brain slices. Moreover, this type or recordings are used for knowing the effects of pharmacological treatments, and to record how this drugs on cell-cultures and brain slices can generate spontaneus potentials by drug effects. Dong and Graziane (2016).

In the next chapter, we present our literature review, where some of the common and new methods are mentioned to detect and classified Action Potentials. Some of

this methods have been tested in software and other part has been tested in dedicated hardware architectures.

– Aprendí pronto que al emigrar se pierden las muletas que han servido de sostén hasta entonces. Hay que comenzar desde cero, porque el pasado se borra de un plumazo y a nadie le importa de dónde uno viene o qué ha hecho antes –.

– Isabel Allende –

2 Background

2.1 Action Potential Detection

This chapter is dedicated, to the background of Action Potential and sorting. We present the main methods found in the literature and how these methods are addressed in different works.

2.1.1 Introduction

As was aforementioned, the Action Potential activity from biosignals comes from the flow of ions in the cell's membrane where the following phases take place: depolarization, repolarization, hyperpolarization and resting potential. Although this information is not the only one that we can record during the process, also we can find noise, Local Field Potentials (LFP) Lebreton et al. (2015): Belitski et al. (2008): Buzsáki (2009), Obien et al. (2015), and Slow Waves Potentials (SWP), in endocrine cells Pirog et al. (2015). Frequenly, LFP are in the low-frequency band of (1-100 Hz) Perelman and Ginosar (2007). They are the sum of currents source, reflecting the synaptic activity of tens or thousands or nearby neurons. Moreover, Slow Waves Potentials in endocrine cells have a band frequency below than 1 Hz Rummens. (2015). The neural firing rate activity, where Action Potentials are seen, commonly is in the band of 100-10 000 Hz Perelman and Ginosar (2007), see (Figure 2.1).

The use of Micro-electrodes is important to extract the information in local regions, where commonly the activity of many neurons can be recorded. These Action Potentials commonly have similar shape and size. Therefore the detection and classification is a challenging task to achieve. Some other problems appear when we try to sort Action Potentials, due to overlapping, when close cells are recorded with the same electrode, and both neurons are firing almost at the same time Lewicki (1998).

In addition, during the recording process there are other sources of noise. These sources generally come from electrodes, other cells and amplifiers, which have to be filtered properly for being able to detect the Action Potential activity presented in our recordings (see Figure 2.2)

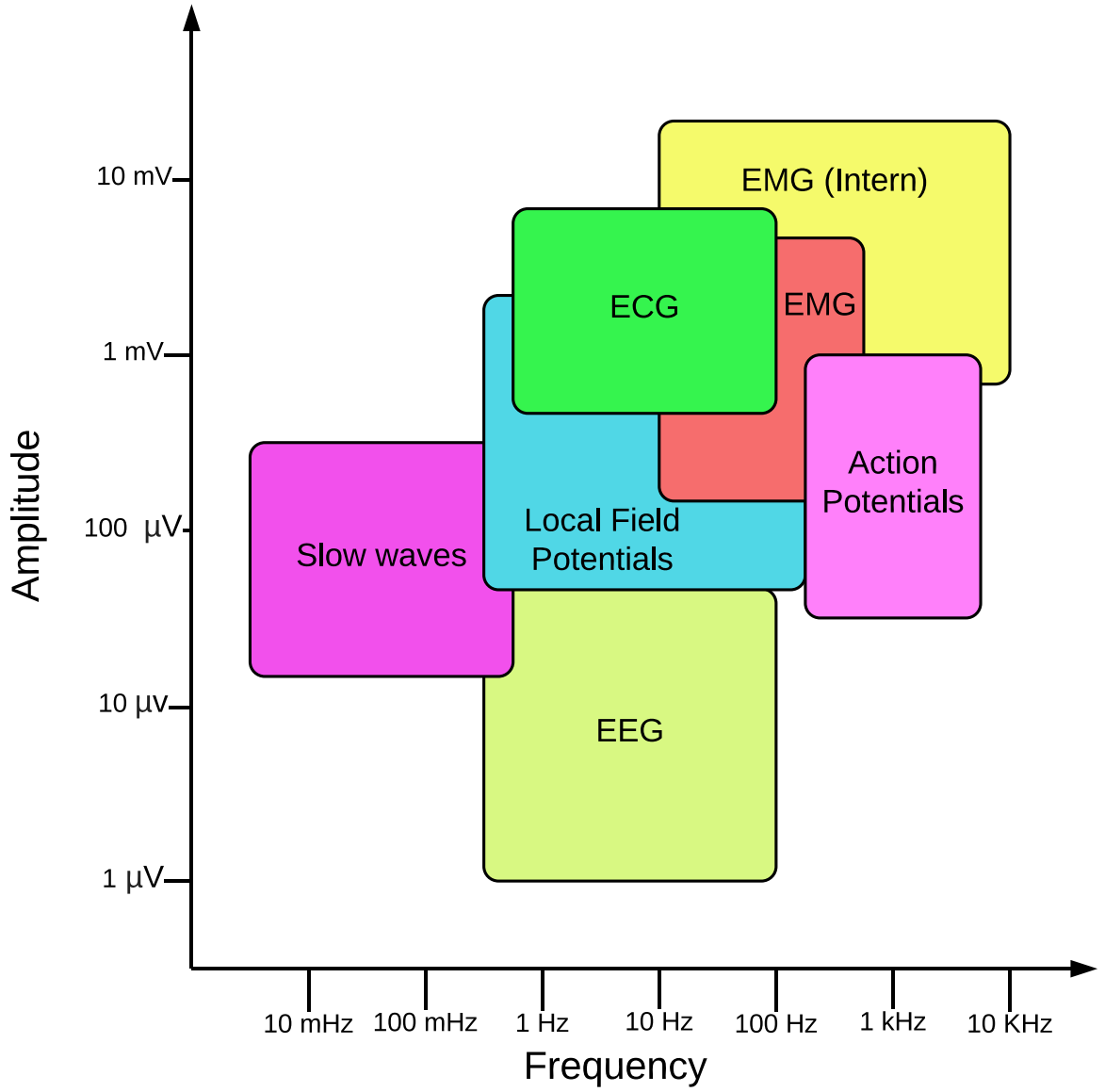


Figure 2.1: Electro-physiology signals with their amplitud and frequency. Electro-encephalogram (EEG), Slow Waves, Local Field potentials (LFP), Electro-cardiogram (ECoG), Electro-myogram (EMG). Ilustration's inspiration : Rummens. (2015)

Pedreira et al. (2012) discuss the typical number of neurons that can be observed in extracellular recordings. Some studies have presented dozen or hundreds of neurons recorded. Others only said that typically the neurons that can be detected per channel is around one of two neurons, with a high amplitude. Other hypotheses says that is cause of the tissue damage, when the electrode is inserted in the recording area. Another says that is cause of neurons that stay almost all the time in silent without presenting firing rates. One interesting fact is addressed here, where by the Coulomb's law the amplitude of Action Potentials decays $v \approx \frac{1}{r^2}$, depending on the distance among the neurons and the electrode. Where Action Potentials among 60 and 110 μV come from a distance around 50 μm, where are able to be detected properly yet, and other more small in a amplitude range of 10 μV taking in account a density of 300,000 neurons/mm^3, commonly in rat hippo-campus recordings. Therefore the Action Potential sorting is complicated taking in account the amplitude as a feature extraction, due to the range of Action Potential

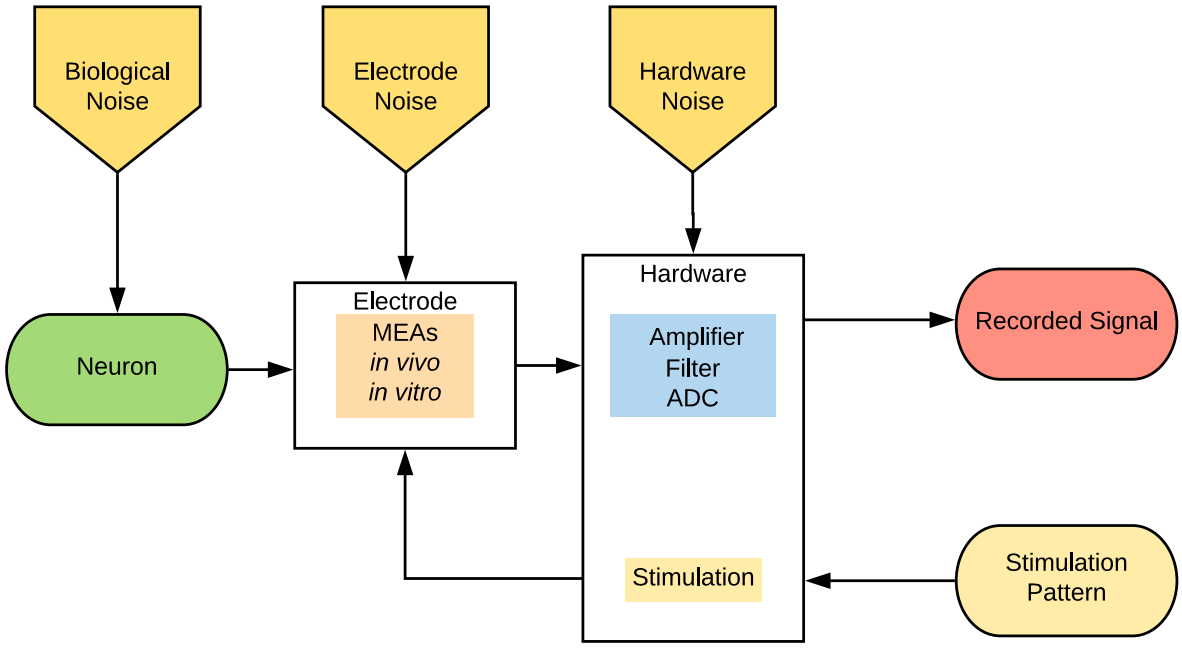


Figure 2.2: Generic parts of noise in biological signals during stimulation and recording . Ilustration's inspiration : Obien et al. (2015)

amplitudes. Action Potentials recorded from a distance between more than 50 μm and around 140 μm from the electrode, their amplitude is not commonly detected and their small amplitude contribute the background noise of the biosignal, Figure 2.3.

As noted above, the purpose of this thesis is to create a module for detecting the Action Potentials present in biosignal recordings. We present now different techniques and projects that have been presented. Here, we will start with a brief study of Action Potential Detection, and how these projects have developed their accomplishments by some methods.

Commonly, Action Potentials detection is composed by the pre-processed signal, where noise, LFP and SWP, are attenuated and Action Potentials are obvious to perform their detection. After, a threshold is used to know the location of Action Potentials in the biosignal, this threshold is commonly set above the background noise. This threshold could be fixed or adaptive. The advantage of adaptive threshold is their constantly compute depending on the background noise presented in the biosignal. On the contrary, the fixed threshold is always the same during the whole detection, and experience is needed to determinate its level. The use of adaptive threshold in MEAs recordings is better, due to each of the hundreds of electrodes have their own noise level, and fixing thresholds levels in each electrode is not a good way to accomplish this task. In order to have a better experiment when biosignals are recorded, the use of adaptive thresholds will give better results, and it will be less tedious than manually change each threshold, Rummens. (2015).

2.1.2 Fixed Threshold

The use of Nonlinear Energy Operator (NEO) for extracting the AP in biosignals has been used for accentuates the high-frecuency content in it. So its function is similar to

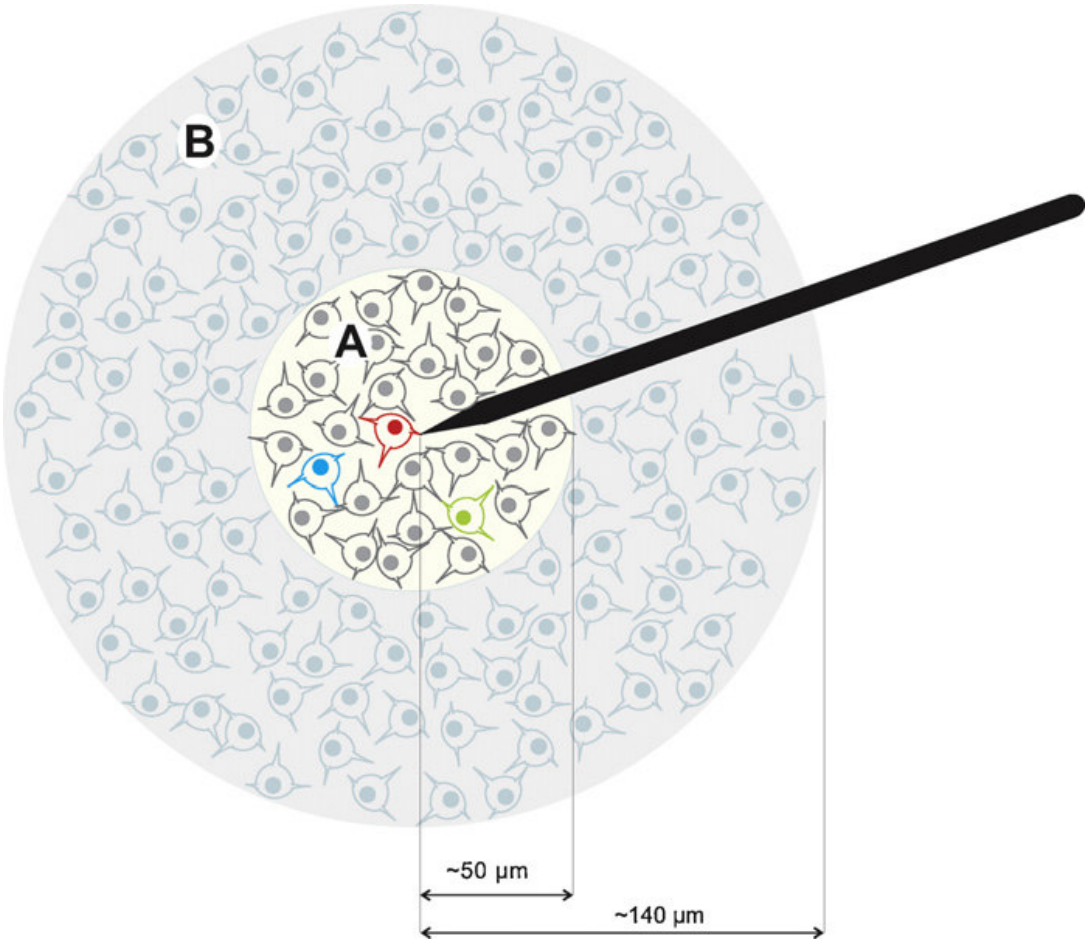


Figure 2.3: Extracellular recording. Action Potentials recording in A area are considered higher than the background noise. Action Potentials recorded in B area are not properly detected by their not large amplitude. Ilustration from : Pedreira et al. (2012)

that of a high-pass output filter, Mukhopadhyay and Ray (1998). The threshold is taken as a scaled version of the mean of the signal and is fixed during the experiment. One of its disadvantages is that the threshold is sensitive to the firing rate. Some projects have addressed this principle as Yang and Mason (2017) even implemented in hardware.

The use of thresholds has been replicated in some off-line software methods to detect APs , but adding some other features, such as compute the average value of Max-Min in the signal, and in this way compute a factor value to be above the background noise Chan et al. (2008).

Other methods have been proposed to have a better detection when two-phases are presented in APs. Maccione et al. (2009) perform the detection by adding some parameters: detection the maximum and minimum point of APs. After the process confirm if the AP was detected into a time window, to determinate if is or not a false positive of negative. This process is applied over raw signals where no filter step is added, and in the most of the cases, this type of methods are tested in software and not in hardware.

In addition mathematical methods are also a tool to compute threshold for AP detection. By the model the differential equations, one of this models was proposed by

Platkiewicz and Brette (2010). Describing how the APs could be changed by channel properties based on Hodgkin and Huxley (1990) equations, but their implementation in hardware could be difficult when certain operations are presented.

The Teager Energy Operator (TEO) is similar to the Energy Detector (ED), both have a similar result taking into account the probability of detection and false detections, and their implementation is easy but no adaptive form of this detector for extracellular recordings or neuronal recordings is available. Therefore Semmaoui et al. (2012) created a new method called the Smoothed TEO (STEO), this method do not need information for the Action Potential shape to be implemented and is most efficient than the other two methods above mentioned.

Other works have compared different techniques OF Feature Extraction (FE) and Dimensionality Reduction (DR), with several methods, for developing Action Potential Detection, and sorting methods to extract the features of Action Potentials in real-time, and classified action potentials coming from other neurons or detecting false positives and false negatives events during the recording or the off-line process. Here Gibson et al. (2010) assess the advantages and drawbacks of each method assuming this method has good candidates to hardware implementation, but some of the techniques for Threshold detection mentioned there are sensitive to the presence of spikes. These algorithms were tested in MATLAB.

Other procedure has been implemented for Action Potential sorting, Yuan et al. (2012). Here, the sorter is based on multiple correlation of wavelet coefficients, using a threshold for detection and template matching for classification. The basis of the detection are based on the multiple correlation of wavelet coefficients, This process is called the M-sorter. No Hardware implementation took place.

Other systems are made over CMOS technology as Barsakcioglu et al. (2014), They developed a Analog Front-end system for neuronal Action Potential sorting. They use common tools for Action Potential Detection and sorting methods, they use the common methods: Template Matching (TM), Principle Component Analysis (PCA), First and Second Derivative Features (FSDE), by the use of a Graphical user Interface (GUI) implemented in Matlab.

The Matched Filter (MF), is used when we have detected the template of an Action Potentialan. Then, we can use it for classification process. Hwang et al. (2014) proposed the use of an Online Sorter (OSort) algorithm Rutishauser et al. (2006), which is an effective unsupervised algorithm for Action Potential Classification, it does not need offline training for Feature Extraction (FE) and clustering. Their algorithm combine normalized correlator with the Osort algorithm, having lower computation time and clustering the Action Potentials present in the biosignal.

Yang et al. (2012). Take into account the exponential form of the noise and the power of Action Potentials, to compute a threshold and an Action Potential pattern. This methods could be implemented together with common methods as NEO.

Ng et al. (2013) use the Smoothed Teager Energy Histogram (STEH) to properly select a threshold, with consideration of signal prewhitening, histogram bin width, and histogram equalization. The signals are accentuated by the STEO, after the STEH select

a threshold automatically.

The CMOS technology has been applied also for detecting Action Potentials, with the use of wavelet coefficients. Yang et al. (2015) made a prototype in a FPGA for 16 channels for Action Potential detection, by applying the use of Stationary Wavelet Transform (SWT). This prototype was after mapping in a 130 nm CMOS technology using the lifting wavelet transform, which is a fast implementation in hardware than the common SWT.

Azami et al. (2015) made uses of a lot techniques, such as the compute of the filters by the use of Genetic Algorithms and the New Particle Swarm Optimization (NPSO). After a phase using the Ensemble Empirical Mode Decomposition (EEMD) is applied as a pre-processing noise reduction step. Here, the use of these techniques consume a lot of resources due to the creation of each Intrinsic Mode Functions (IMFs) to be able to implement this algorithm in hardware. In addition, the use of a Hilbert transform is proposed. So the implementation of this system will consume a lot memory resources. Thereby, this system is not a good candidate for a hardware implementation.

Other methods have been developed in the last years, such as the Time-frequency based convolution Action Potential algorithm (TIFCO) and the Stationary wavelet based TEO(SWTTEO). The first one is based on the time-frequency of Action Potentials in the range of 500 Hz and 3500 Hz and AP are able to be detected. The second one is based on a low-pass filter using the Discrete Wavelet Transform (DWT) and applying the Teager Energy operator to each filter sub-bands Lieb et al. (2017). The implementation was made over Matlab. No hardware architecture took place. In addition, Mayer et al. (2018) implemented this method demonstrating good performance even with a low SNR.

2.1.3 Adaptive Threshold

Chan et al. (2008) consider adaptive thresholds through the Max-Min spread (MMS), reporting similar results than the conventional methods, as Harrison (2003) and Donoho (1995). Although the threshold used in this work are not good candidates for a Hardware implementation by certain operations needed.

One of the adaptive thresholds was proposed by Harrison (2003). This threshold is able to be adapted above the background noise level presented in the biosignal. To get good performance, a high-pass filter is needed to attenuated the LFP in the biosignal. This AP detector provide a logical true when an Action Potential is above the threshold given a binary signal (1,0), during the time the action potential exceed the adaptive threshold. This method has the advantage of not be sensitive to the AP firing rates. These features are addressed in the next chapter.

Horiuchi et al. (2004). The use of adaptative threshold has been implemented in CMOS models as Horiuchi et al. (2004), where the Harrison (2003) loop has been implemented in VLSI processors. Here, the amplitude of Action Potentials are measured as an alternative to develop, Action Potential Sorting by a peak and trough detector.

Action Potentials (APs) usually have two phases, negative and positive, So using this principle the use of two threshold is better to detect an action potential (AP), in a lower period than their duration (1-4 miliseconds). This is a good tool, on the contrary, when

one threshold is used for AP detection. This reduce the detection of false positives and negatives. This principle was addressed by Borghi et al. (2007), and consequently by Hiseni et al. (2009) but adding the analysis of Matched filter.

The use of adaptive threshold has been simulated in software for Action Potential Detection. In addition, some projects as Biffi et al. (2010) use Principal Components Analysis (PCA) and a Hierarchical Classifier (HC), they have been simulated their algorithms to achieve a real-time analysis of Action Potential Detection and Classification. Nonetheless, their design for Hardware implementation has not been tested, and many problems occurs when the software version is translated to hardware architecture.

Quotb et al. (2011). The use of wavelets is also a good tool for Action Potential Detection but its implementation in hardware could be costly, depending on the detail level of the filter banks designed for the architecture. Here, the use of Discrete Wavelet Transform (DWT), and Stationary Wavelet Transform (SWT) were tested with a pre-processing filter to accentuate Action Potential and decrease the background noise, followed by a threshold adaptive circuit computed in real-time.

Commonly algorithms for Action Potential sorting are composed of three parts, The Action Potential detection, extraction of some features for Action Potential shapes and clustering. Takekawa et al. (2014) proposed a new method using probabilistic tools to detect and cluster the different Action Potential in biosignals, and the Bayes theorem compute the probability of a sample to belong to an Action potential, by taking into account the distribution nature of Action Potential amplitudes, widths and frequency. This method is useful for analyses of extracellular recordings with linear probes but no hardware implementation took place.

Wu et al. (2016). Here an Application-Specific Integrated Circuit (ASIC) is able to deal with 16-channels for Action Potential Detection in real-time execution. The system is based on the the Exponential Component-poly-nomial component (EC-PC) algorithm, by applying a probability threshold. This system is able to deal with Action Potential detection, Local Field Potentials (LFP), and Action Potential probability maps. Also the use of Hilbert transform is addressed.

Yang and Mason (2017). The Nonlinear Energy Operator (NEO) is an algorithm which is sensitive to the firing rate of Action Potentials. Here, Yang and Mason (2017), propose a new way to compute this threshold in real-time making it less sensitive to the firing rates of Action Potentials. This methods use as reference the standard deviation and the root-mean-square (RMS) frequency of the background noise in the biosignal. This system was implemented in Very Large Scale Integration (VLSI) for be able to demonstrate real-time processing.

Dwivedi and Gogoi (2018). Here the use of a adaptive threshold is performed, the main idea is similar to the others only during the detection of Action Potential, the system create a window around 2 ms, where the data of the Action Potential detected is not taken into account. In that way the threshold is not involve with the Action Potential data and the threshold is compute only with the rest of the background noise. The Action Potentials are accentuate and the SNR is better by using a Energy-of-derivative method, which is more simple and suitable for an implementation than the common nonlinear energy operator (NEO). The system was implemented in a compact analog

CMOS integrated circuit.

Some other Action Potential classifiers have been tested as: Super-Paramagnetic Classification (SPC), Osort , K-means, Moving Centroid K-Means, Moving Centroid K-Means, Hierarchal Adaptive Means (HAM), Fuzzy C-Means (FCM), Mahalanobis classification, Support vector classification (SVC), Self-organizing maps (SOM) Cosine Similarity classification. Here, Saeed et al. (2017) present a good comparative of existent classifier architectures for implementation in on-line Neural Action Potential sorting. They concluded than the best classifier is the SOM classifier when is taking in account accuracy and complexity. This classifier is based on the use of an Artificial Neuronal Network (ANN) . Although we know the implementation in hardware of a classifier based on an (ANN) consume a lot of memory resource by the creation of Lookup tables.

This work is focused on proportionate and easy method to be implemented in hardware, and change important parameters for computing an appropriate threshold to detect Action Potentials in biosignals recordings. Common methods as NEO are used to compute Action Potentials in hardware but this technique reduce its accuracy when multiple frequency components and noise are presented, and several false detections could take place Azami et al. (2015), Wu et al. (2016). Other methods as the use of wavelet detectors require excessive hardware resources Ng et al. (2013), Wu et al. (2016).So we tested the adaptive threshold of Harrison (2003), which not need many resources and their implementation in hardware is straightforward. In addition this adaptive threshold is not sensible to Action Potentials when goods parameters are selected.

Our strategy is to use this adaptive threshold with the generation of a Mean Action Potential shape to be correlated with the Action Potentials detected, in that way, we use the technique of match filtering using the correlation to generated the autocorrelation signal between those shapes. Afterwards, when Action Potentials and Mean Action Potential shape match, a correlation pattern is generated. Therefore, we detected this pattern to classified each Action Potential detection among six possible groups, depending on its correlation pattern detected. Furthermore, we not found works, in our literature review, where correlation patterns are detected by adaptive thresholds. We describe the methodology followed in the next chapter and the hardware architecture designed.

– On meurt toujours trop tôt - ou trop tard. Et cependant la vie est là, terminée. tu n'es rien d'autre que ta vie –.

– Jean-Paul Sartre –

3 Methodology

3.1 Description

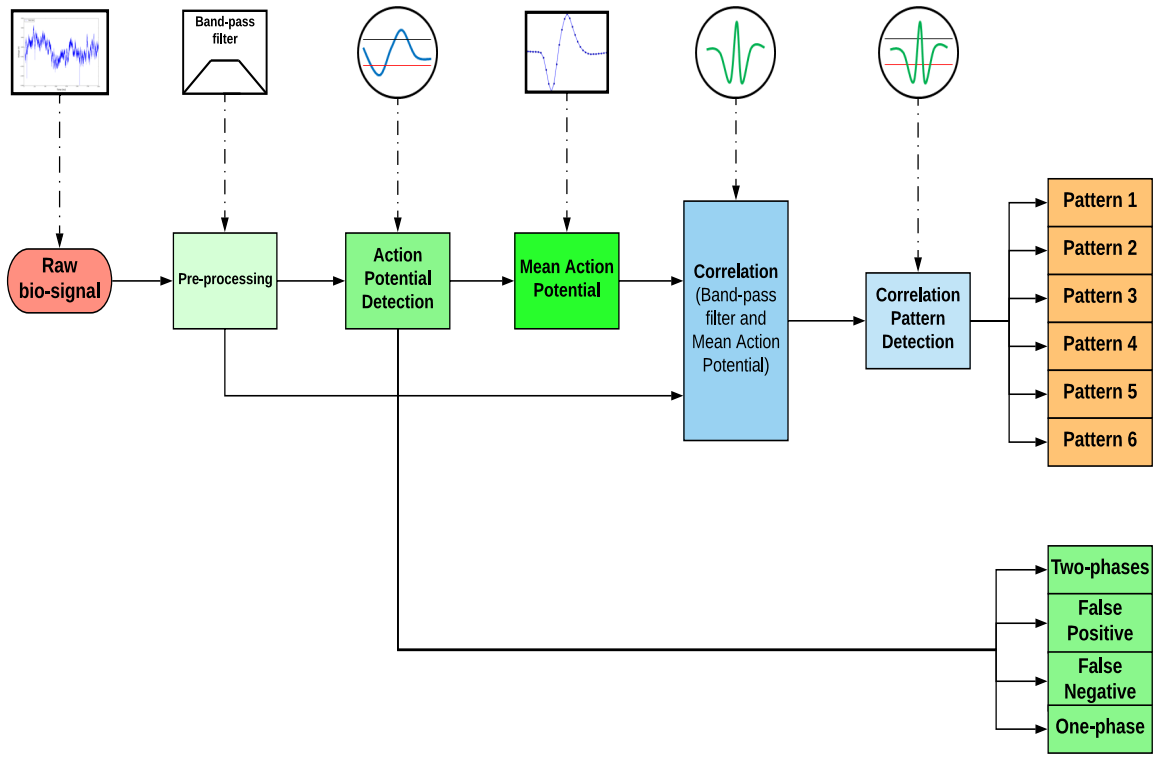


Figure 3.1: General system diagram and its main parts.

In this work, we use two biosignals to test the system a Macaque monkey signal recording ***in vivo***, with a 40 KHz sample frequency and a Human pancreatic signal at 10 KHz sample frequency. After, in the pre-processing section we used a IIR Band-pass filter to accentuate Action Potentials, remove Local Field Potentials and Slow Waves into the recording. The section of Action Potential Detection is implemented by a adaptive threshold, which computes a threshold above the background noise of the biosignal in a real-time computing and saves the maximum value of the Action Potential detected

when a Action Potential is above the threshold.

The detection made for the positive an negative detectors are saved in four detection types. The Two-phases detection are those Action Potentials that triggered both detectors, here they are the Action Potentials from the macaque monkey biosignal and the Human pancreatic biosignal which have a depolarization and hyperpolarization phases. The False Positive are those events where the signal stays above the threshold more time than a regular period of a Action Potential length. In a similar way, the False Negatives are those events where the signal stays more time under the threshold than a regular period of a Action Potential length. The one-phase detections are those Action Potentials that only triggered the positive detector but are in the time length of an Action Potential, these are the Action Potentials from the Human pancreatic biosignal that are detected and saved.

The next part is the Mean Action Potential, where each time a Macaque monkey or a Human pancreatic Action Potential is detected a new Mean Action potential template or shape is computed by a IIR low-pass filter for each of the shape samples. After, the correlation part compute the correlation value between the Mean Action Potential and each new sample that is read and processed by the band-pass filter.

The Correlation Pattern part detects and classifies, by the correlation pattern detected generated between the Mean Action Potential and the Action Potential shape detected at that moment. Finally, in the Correlation Pattern Detection, the correlation shape is detected and classified with the maximum sample number of the Action Potential that was stored in the Action Potential Detection part, this depending on its correlation pattern detected and saved among six possible patterns as shown in Figure 3.1.

3.2 Biosignal processing

Digital filters commonly are implemented in two ways, by convolution as in Finite Impulse Response (FIR), and by recursion Infinite Impulse Response (IIR). FIR have a better performance than IIR filters but its response is slower.

So, we decided to use IIR filters by its long impulse response without having to compute a long convolution, and they have a faster respond than FIR filters. The common difference equation for a IIR filter is presented in Strauss (2000) as follows:

$$y[n] = a_0\,x[n] + a_1\,x[n-1] + a_2\,x[n-2] + ... + b_1\,y[n-1] + b_2\,y[n-2] + b_3\,x[n-3] + ... \quad (3.1)$$

Where $y[n]$ is the output and $x[n]$ is the input signal. The a_n and b_n are the recursion coefficients. This equation can also be interpreted as follows:

$$y[n] = \sum_{i=0}^{L} a_i\,x[n-i] - \sum_{i=1}^{M} b_i\,y[n-i] \quad (3.2)$$

Where, the filter order is $M \geq L$. In our cause, to properly filter the Macaque monkey biosignal, we used a first order low-pass IIR filter with the next coefficients represented in the next equation:

$$y[n] = 0.75\, y[n-1] + 0.125\, x[n] + 0.125\, x[n-1] \tag{3.3}$$

This equation is easy to be implemented in hardware, if we see this equation in the next way:

$$y[n] = \frac{3\, y[n-1]}{4} + \frac{x[n]}{8} + \frac{x[n-1]}{8} \tag{3.4}$$

$$y[n] = y[n-1] - \frac{y[n-1]}{4} + \frac{x[n]}{8} + \frac{x[n-1]}{8} \tag{3.5}$$

Here, the $\frac{y[n-1]}{4}$ value is shifted to the left two times and after is subtracted with $y[n-1]$. In that way we have the equivalent value of $\frac{3y[n-1]}{4}$ in the Equation 3.4. In the same way, $\frac{x[n]}{8}$ and $\frac{x[n-1]}{8}$ are shifted to the left three times and after these values are added up. Finally, both results are added up to obtain the IIR filter output.

An easy way to estimate the time constant ($RC = \tau$) for a recursive single pole filter is similar to a RC filter, where τ is the time that takes to decay to 36.8% its final value, so τ , in Equation 3.6, is the number of samples that takes to reach to this value.

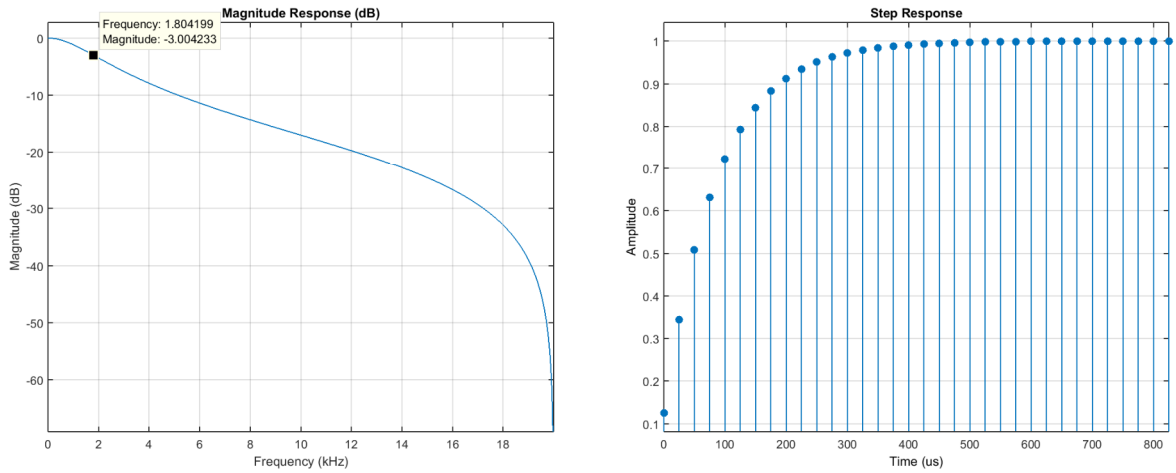
$$x = e^{1/\tau} \tag{3.6}$$

In our case, x= 0.75 from the Equation 3.3, so we have a constant time of $\tau = 3.47$ samples. In the same way, the relation of the cut-off frequency (F_c), is commonly found with a magnitude of -3dB. This relationship is present such as in Strauss (2000) as follows:

$$x = e^{-2\pi F_c} \tag{3.7}$$

One more time x= 0.75, giving a $F_c = 0.045$ and having a sample frequency of ($F_s = 40$ KHz) the cut-off frequency will be around 1.8 KHz, such as in the magnitude response and its Step response in (Figure 3.2).

Commonly, books talks about generally of low-pass filters, due to from a low-pass filter we can create any type of filter by adding parallel stages, in the time domain. This techniques is know as spectral inversion, for instance for creating a high-pass filter we have to put in one stage a low-pass filter and in the other one an all-pass filter that is



(a) Magnitude response

(b) Step response

Figure 3.2: IIR low-pass filter with $F_s = 40$ KHz and $F_c = 1.8$ KHz. Equation 3.3

commonly our signal. In that way, we create a spectral inversion, changing a low-pass filter to a high-pass filter these stages are illustrated in (Figure 3.3). In that way all the components of low frequency will be subtracted, and only the high frequency components remain in the signal. Therefore, we will have a high-pass filter.

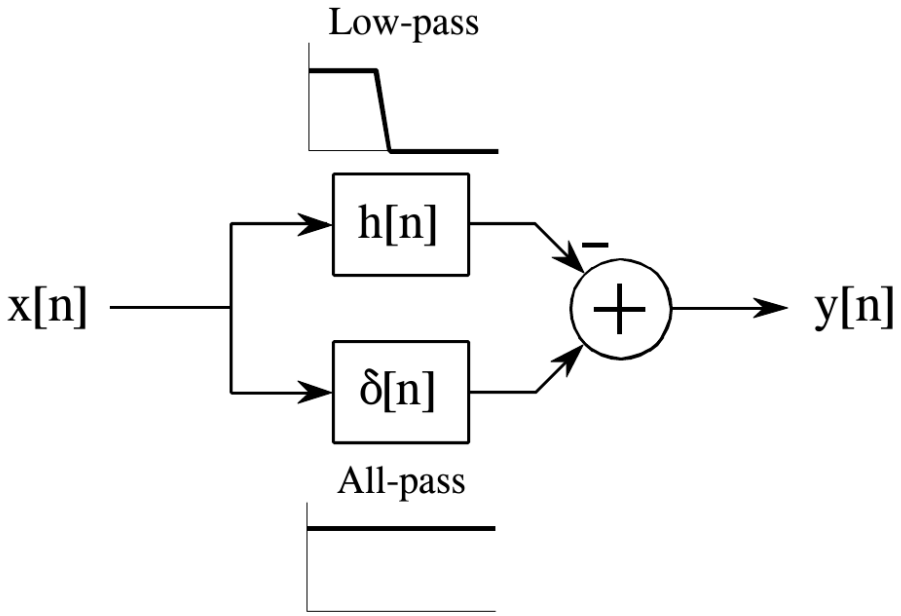
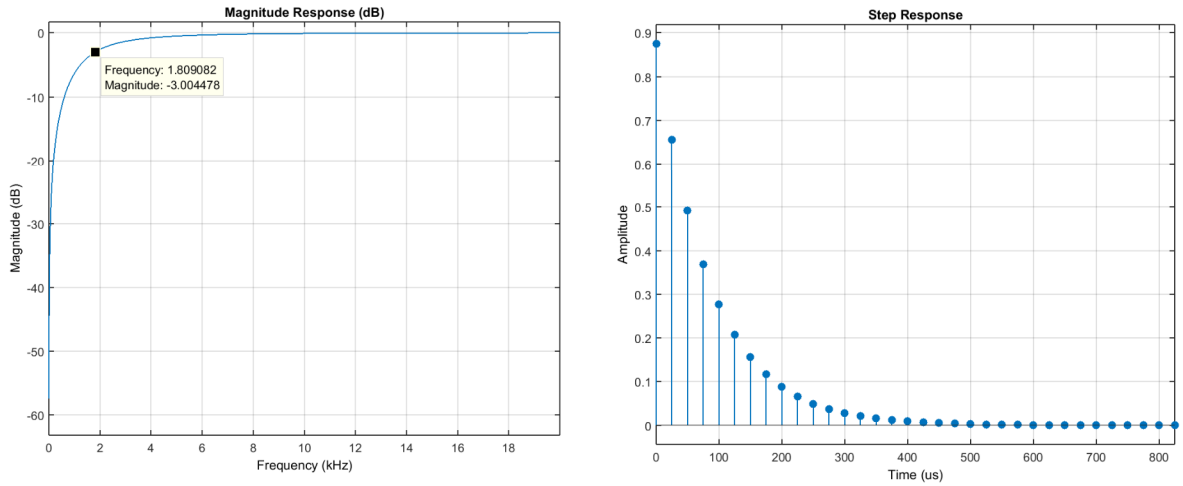


Figure 3.3: Low-pass filter spectral inversion to generate a High-pass filter. Ilustration from : Strauss (2000)

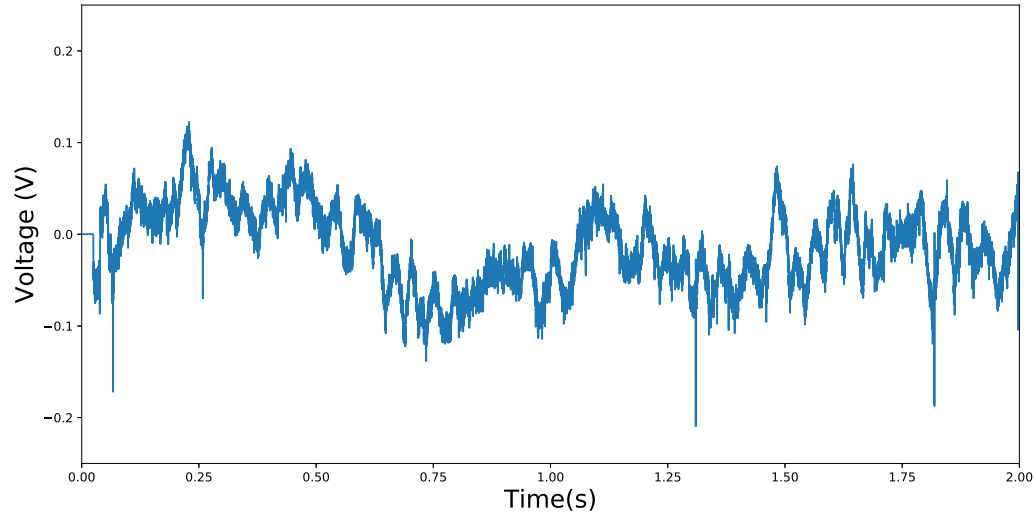
In the same way, we present the magnitude response and step response of the high-pass filter applied the spectral inversion as follows in (Figure 3.4).



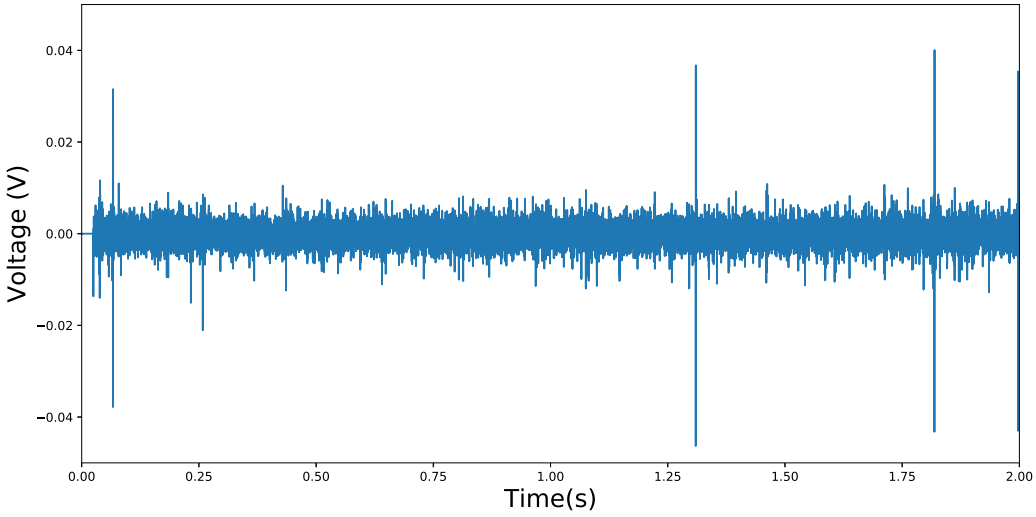
(a) Magnitude response

(b) Step response

Figure 3.4: IIR high-pass filter with $F_s = 40$ KHz and $F_c = 1.8$ KHz Figure 3.3



(a) Raw data, Macaque monkey signal recorded ***in vivo***

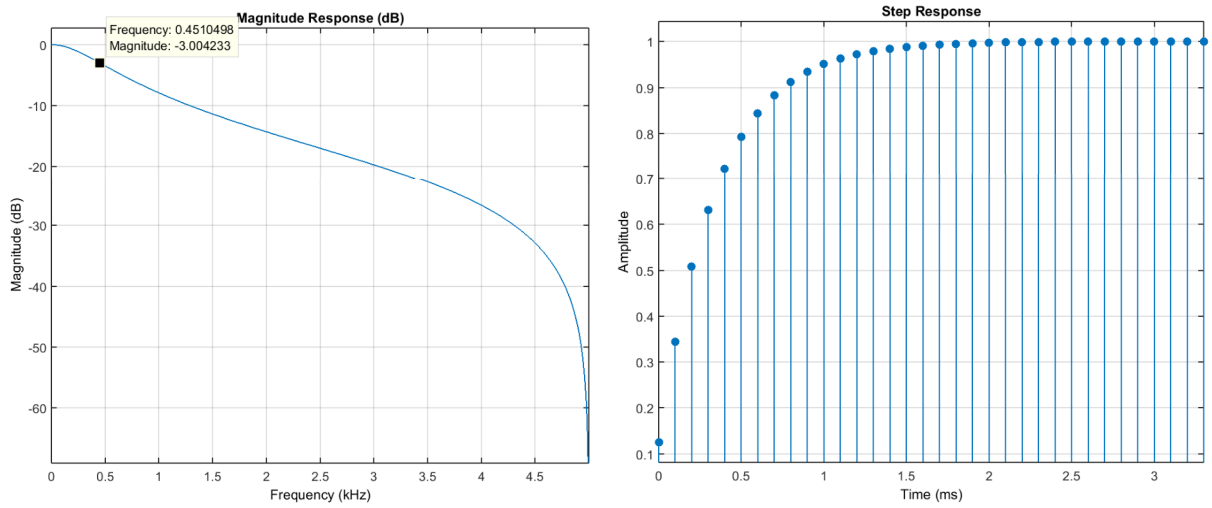


(b) Band-pass filter output. Here, some Action Potentials of Macaque monkey are presented

Figure 3.5: Macaque monkey biosignal, and Band-pass output with Action Potentials accentuated.

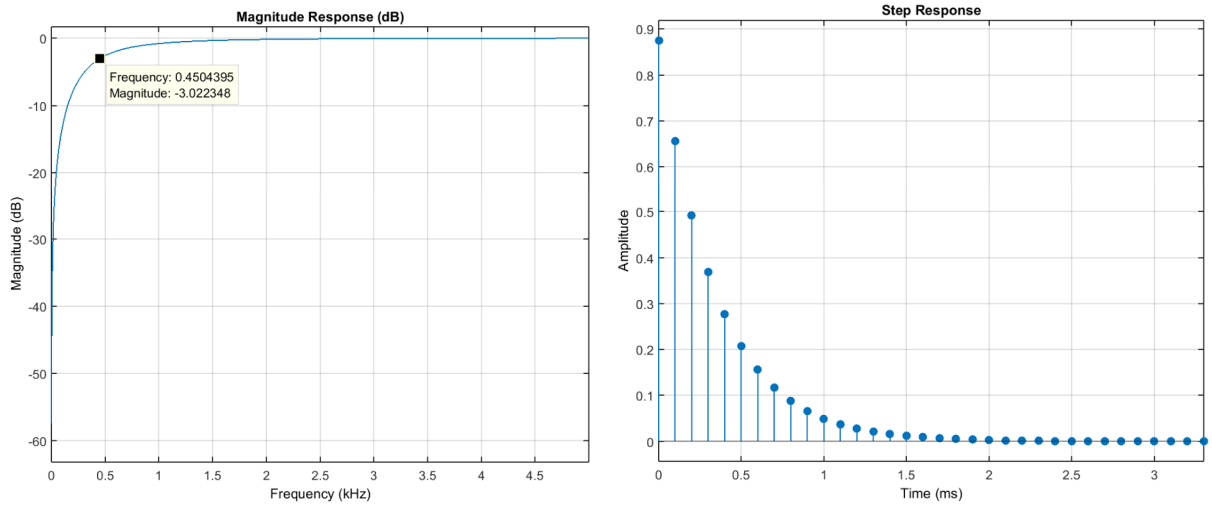
Now, we applied the high-pass filter to the raw data, and by adding it up a low pass filter in the high-pass filter output we get a band-pass filter, the low-pass filter applied was the same that in the Equation 3.3. In this case our biosignal is a Macaque monkey signal recording ***in vivo***, with a 40 KHz sample frequency. The raw data and the output of the band-pass filter are shown in (Figure 3.5). Here, we can observe where the Action Potentials are accentuated in their amplitude. Therefore, it will be easier to detect the Action Potentials by a threshold over the background noise and any dc off-set has been removed. How is shown in (Figure 3.5 (b)).

In the same way, for the Human pancreatic biosignal, we take the same filter equation presented in (Equation 3.3. the cut-off frequency will be around 450 Hz by using a $F_s = 10$ KHz, such as in the magnitude response and its Step response in (Figure 3.6. This is computed by using the same (Equation 3.7) aforementioned.



(a) Magnitude response (b) Step response

Figure 3.6: IIR low-pass filter with $F_s = 10$ KHz and $F_c = 450$ Hz. Equation 3.3

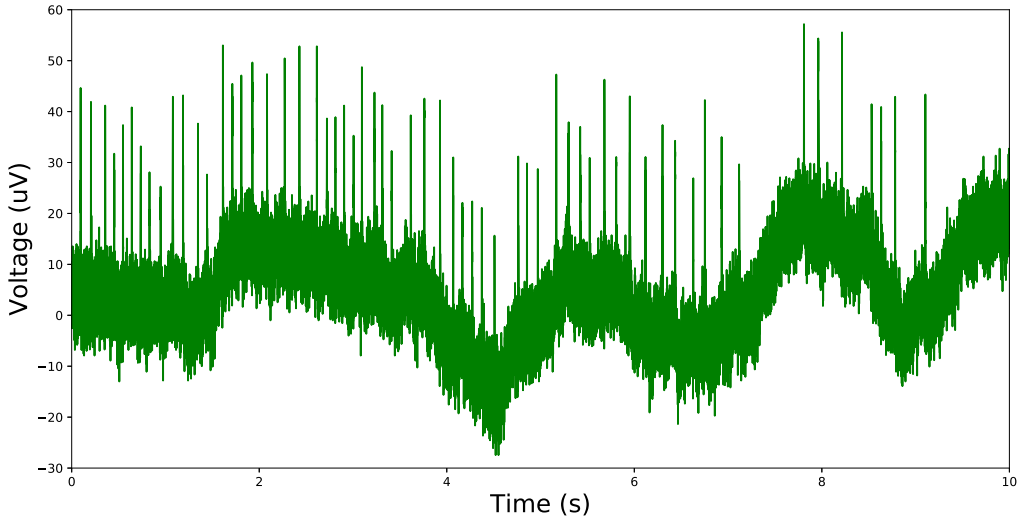


(a) Magnitude response (b) Step response

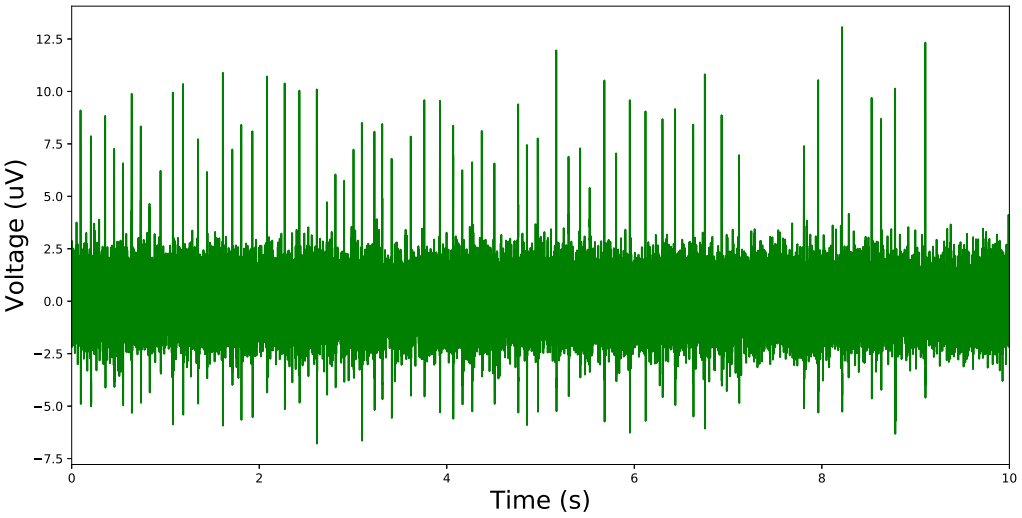
Figure 3.7: IIR high-pass filter with $F_s = 10$ KHz and $F_c = 450$ Hz. Figure 3.3

In the same way, we present the magnitude response and step response of the high-pass filter applied the spectral inversion as follows in (Figure 3.7). The Human pancreatic

raw data and band-pass filter output are shown in (Figure 3.8).



(a) Raw data, Pancreatic biosignal from human



(b) Band-pass filter output. Here, some Action Potentials of Pancreatic biosignal from human are shown

Figure 3.8: Pancreatic biosignal from human, and Band-pass filter output with Action Potentials accentuated.

In addition, the implementation in hardware is easier, due to all the filters that we have presented are first order, and only by using some multiplications and additions, we can built the architecture of each filter. In this case we do not need the use of multipliers, due to by following the Equation 3.3 only by shifting the values we can obtain the desired filter. Therefore, we can design each filter as a entity in VHDL language by properly describing each filter here mentioned. Thereby, the use of resources of memory as operations that can be costly in hardware implementation of filters will be reduced.

3.3 Action Potential Detection

The Action Potential Detection, as we mentioned above, in the background section, can be addressed in many ways, but in our case, the most important is to take a method that is easy to be implemented in hardware, which is our final goal. Therefore, the use of a threholds adaptative is needed, when each time a new sample of the biosignal is read, the adaptive threholds is computed, before the next sample is read. Thereby, the thredholds will be adaptive and computed in real-time. Futhermore, the method that we used was the adaptive threshold mentioned in Harrison (2003). Here, the Action Potential Detection is automatic in noisy waveforms, where the threshold is adapted and stays above the background noise level. Thereby, each time an Action Potential is above the threshold, a logical true signal is emitted, and vice-versa, with a false signal, when Action Potential is not presented. The main idea is to put a threshold low enough to detect Action Potentials and high enough to not detect peaks from the background noise. The general diagram system is shown in (Figure 3.9).

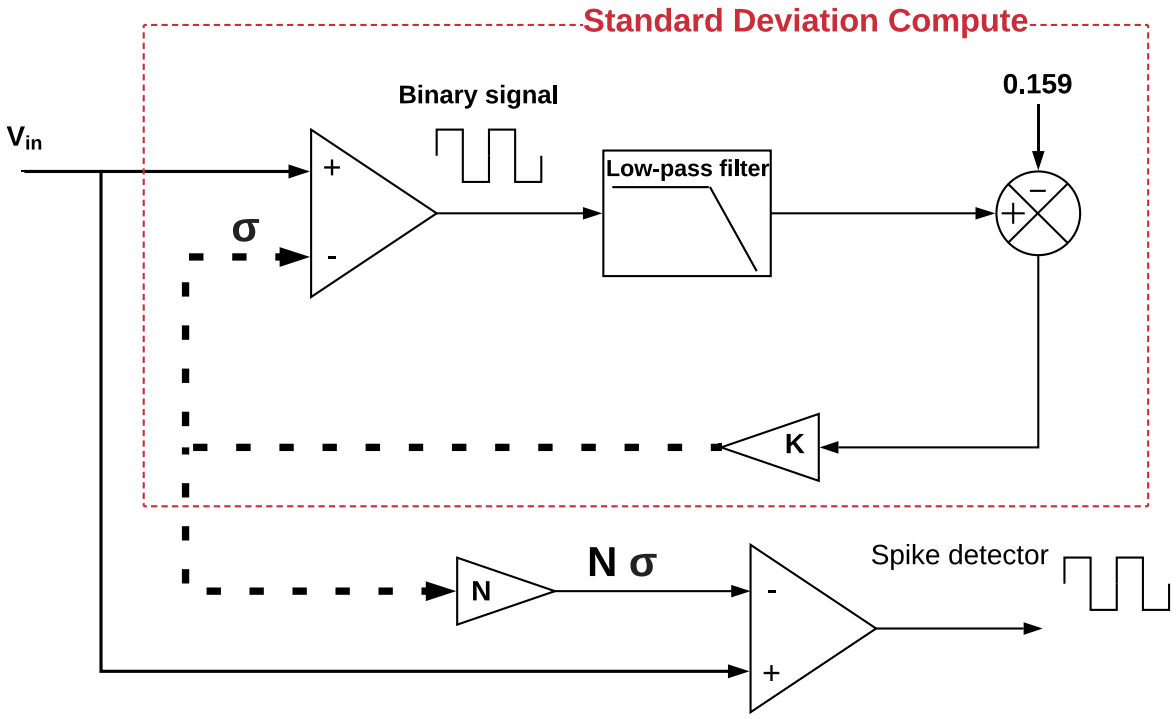


Figure 3.9: Standard Deviation approximation circuit diagram and Action Potential Detection. Illustration inspiration from Harrison (2003).

Action Potential detection using this technique is based on the measure of the Gaussian noise of a biosignal, due to during the acquisition of biosignals, biological noise tend to have a Gaussian distribution. Therefore, when we use a high-pass filter the mean value of the biosignal has a mean of zero. Thereby, any dc offset has been removed and the noise is equivalent to its Root Mean Square (RMS) value, which is the standard deviation σ of our biosignal Harrison (2003). This hypothesis rely on one of the features of a Gaussian distribution, where taking into account a random variable, with a probability density function of a Gaussian distribution with mean μ and standard deviation σ. The probability of the random variable in one event to be above the $\mu + \sigma$ is 15.9 percent (15.9 %). In addition, this hypothesis is applied when we have a signal of N samples, which have a Gaussian distribution, so the 15.9 % of these samples will be

above $\mu + \sigma$. Consequently, the noise is equivalent to the standard deviation σ, due to the null mean μ has mentioned before, Rummens et al. (2015).

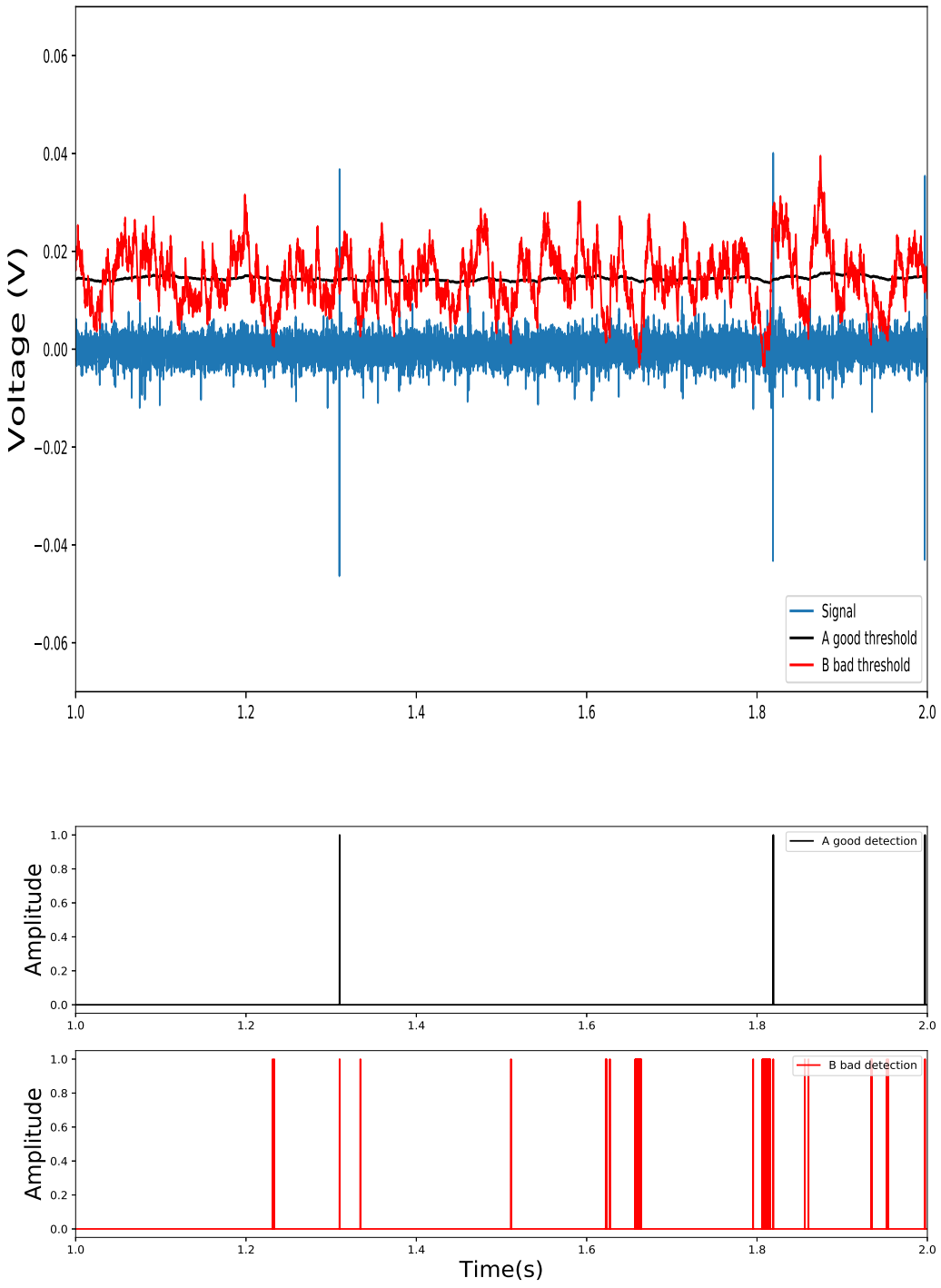


Figure 3.10: Threshold parameters. Configuration A good detection of Action Potentials, configuration B bad detection of Action Potentials. The signal shown is a part of the Monkey macaque biosignal where three Action Potentials appear.

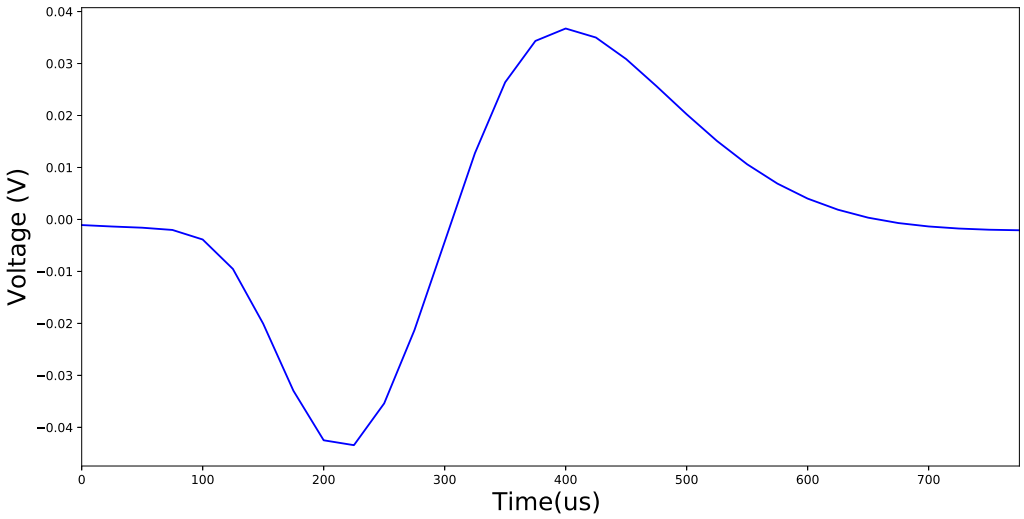
As it is appreciated in (Figure 3.9), the compute of the threshold is similar to a Proportional Controller, where the final goal is to compute an approximation to the

Standard deviation (σ), with the hypothesis that only the 15.9 % of the samples will be above this (σ) approximation computed by the system. Thereby, the biosignal is compared with the Standard deviation (σ) approximation computed by the system. After, a low-pass filter compute the mean proportion of samples that have been above the Standard Deviation (σ). Consequently, this mean proportion is compared with the reference of 15.9 % that follows the hypothesis that the 15.9 % of the samples will be above the Standard deviation (σ) before mentioned. Finally, The loop is completed by the constant K, that correct the Standard deviation (σ) approximation value to be compared with the biosignal and the loop is closed. In order to detect the Action Potentials the Standard Deviation (σ) approximation is amplified by a constant N, the bigger the N value the lower the probability of the noise to be above this value. Therefore, Action Potentials are detected and the probability of the noise to pass the N σ value is lower. As Harrison (2003) mentioned, to put a threshold of 5σ have a probability around $3x10^{-7}$ of the Action Potential Detector to be triggered by the Gaussian Noise.

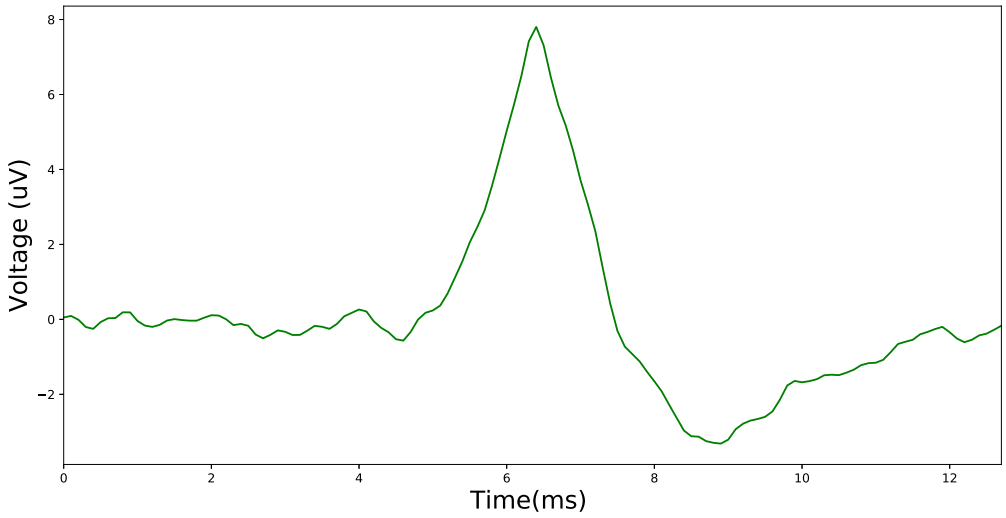
In addition, Action Potentials are very brief voltage pulses that not contribute to the compute of the standard deviation (σ). This last sentences is true when good parameters are found by computing the adaptive threshold for each signal by changing the time constant of the low-pass filter and the K constant value and N value. This is shown in (Figure 3.10), where in configuration A good parameters were put for the time constant of the low-pass filter, the K constant value, and N value. In configuration B the compute of the adaptive thresholds is more sensible to the presence of Action Potentials. Commonly a factor of 5 is used in N, but this could be higher or lower depending on the background noise in the biosignal, Harrison (2003).

3.3.1 Detection of Action Potentials from Macaque monkey and Pancreatic human biosignals

The general Action Potentials shapes to detect are shown in (Figure 3.11). These two Action Potentials shapes are originated commonly from one cell of more cells depending on the recording conditions. We can appreciate that the Action Potential from Macaque monkey has two phases big enough to be detected for two thresholds, a positive and a negative threshold. While, the Human Pancreatic Action Potential only have one a phase big enough to be detected, due to its negative phase is not big enough to be detected properly, and it could lead to a bad detection. Therefore in this biosignal we decided to detect both Action Potentials with two-phases and Action Potentials with one positive phase, due to this information may be important but the compute of the Mean Action Potential shapes is made with two-phases detections.



(a) Action Potential from Macaque Monkey biosignal. This shape is the mean result of 101 detections



(b) Action Potential from Human pancreatic biosignal. This shape is the mean result of 28 detections

Figure 3.11: Action Potentials of both biosignals Macaque monkey and Human pancreatic.

The detection of Macaque monkey Action Potentials was made for two thresholds, in that way the true detection of Action Potentials will have a lower uncertainty against false negatives and false positives detections. Subsequently, we will always detect a negative phase and a positive phase. This is shown in (Figure 3.12), where both phases are detected for the adaptive threshold, and both detectors generate a true pulse, with the width of the peak time when is above the adaptive threshold or under, such as the case in the negative phase detected.

In order to detect properly the Action Potentials, from the Human pancreatic biosignal, the use of one threshold could be better to detect its Action Potentials, due to in the negative phase, called hyperpolarization phase, the noise could affect the detection , such as in (Figure 3.13). Here, the hyperpolarization phase is not low enough and the (- Detector) could produces two pulses, due to the background noise in the biosignal.

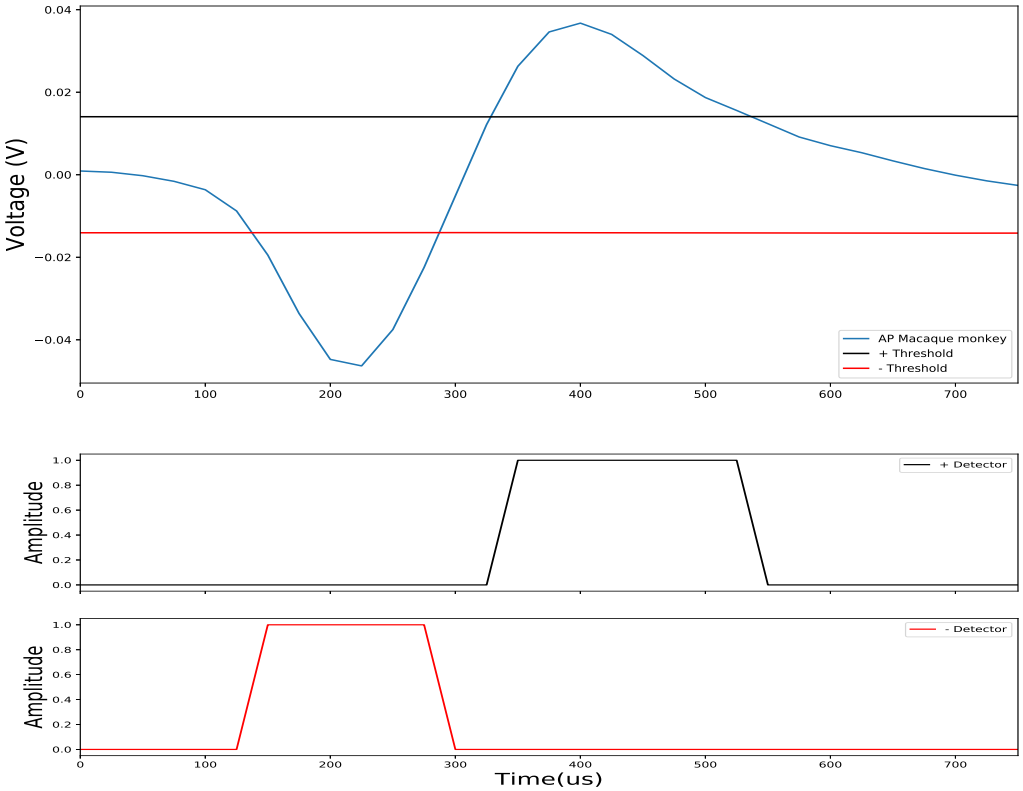


Figure 3.12: Macaque monkey Action Potential detected for both detector positive and detector negative.

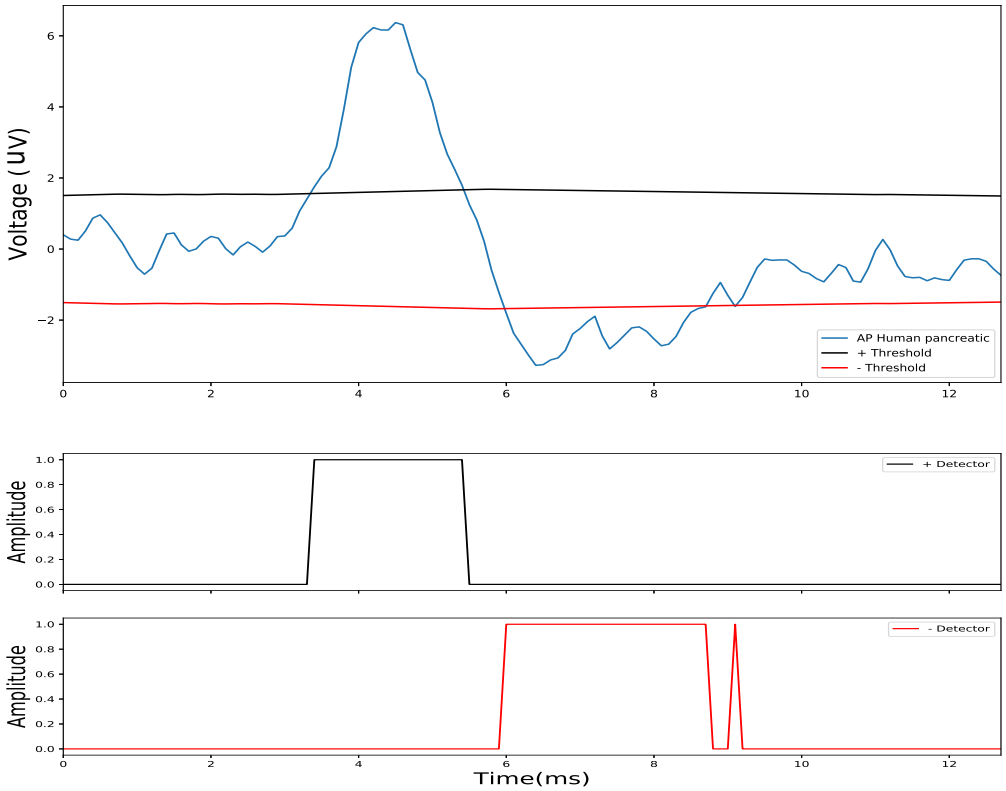


Figure 3.13: Human pancreatic Action Potential detected for both detector positive and detector negative.

3.4 Mean Action Potential Compute

The Mean Action Potential is computed in the same way as a low pass filter, using the Equation 3.3. Where each sample shape of each Action Potential sample detected pass through a low-pass filter. In order to have an average value of each sample and generates a Mean Action Potential shape during the detection process. This can be seen in the step response of a low-pass filter as in (Figure 3.4b), where the low-pass filter output reach to the step value after certain samples. The use of a low-pass filter is approximated to the well know average value for certain amount of data as shown in Equation 3.8. In our case, we have to average each sample of the Action Potential detected, in order to save the whole Action Potential shape.

$$M[n] = \sum_{i=1}^{D} \frac{S_i[n]}{D} \tag{3.8}$$

In Equation 3.8, $M[n]$ is the average value of one of the n samples of the whole Mean Action Potential shape. For instance, the Macaque Action Potential takes 32 samples to properly observe the whole Mean Action Potential shape. In that way, n goes as follows: (0, 1, 2, ...31), in order to obtain the 32 samples that form the whole Mean Action Potential shape. D is the number of Action Potentials detected, and i is the index to change to one shape to the other.

Using a low-pass filter, we not need to develop the division by the whole amount of data and neither to store all the Action Potentials shapes detected. In that way, in hardware implementation, we do not use this costly operation, and only saving the last output of the low-pass filter, last and present detections we can generate the Mean Action Potential shape by saving each sample. (Figure 3.14) represents the compute of the Mean Action Potential shape by passing the low-pass filter and its result of each of the sample that form the mean shape.

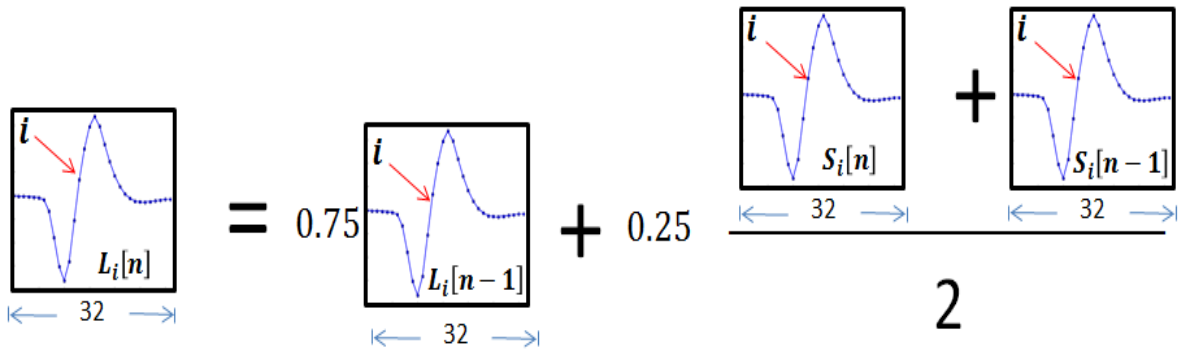


Figure 3.14: General diagram for computing the Mean Action Potential shape by using a low-pass filter equation. The last output shape $L_i[n-1]$, present shape $S_i[n]$, and last shape detected $S_i[n-1]$ are stored to compute the new mean shape.

Here in (Figure 3.14), L is the output of the filter, i is the index of the 32 possible samples of the Action Potential shape, S represents the Action potentials detected. In order to compute the Mean Action Potential shape, the last shape computed for the

filter is needed and stored, as well as the present detection and last detection shape.

The use of a low-pass filter is better than the average, due to memory resources. The comparative of both shapes by using a low-pass filter and by using the common average equation is shown in Figure 3.15, where the approximation using the low-pass filter gives a good result.

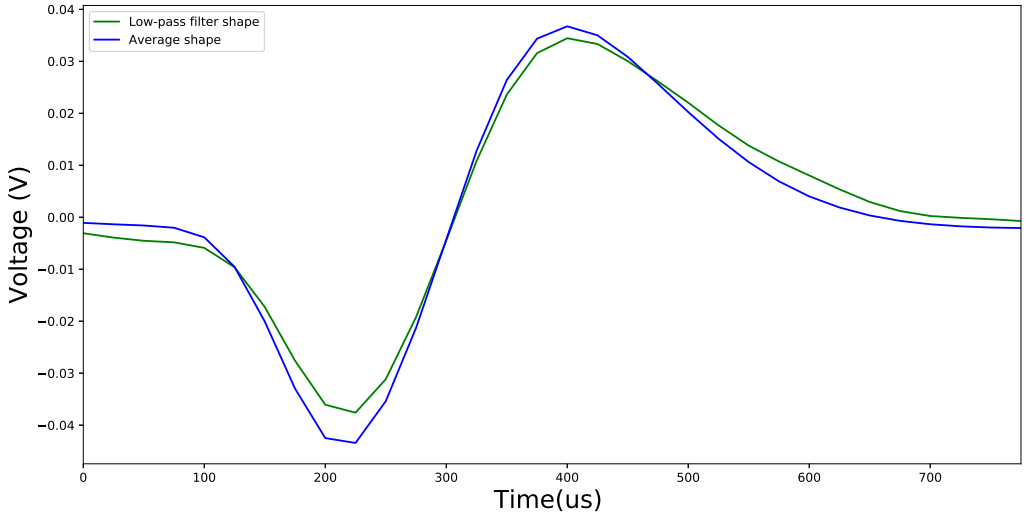


Figure 3.15: Compute and comparative of average and low-pass filter approximation for computing the Mean Action Potential shape using 101 detections.

3.5 Correlation

The correlation is commonly seen as a measure to compare two signals. If our signals are similar a positive correlation result is expected. In that way, we use correlation to compare the actual Action Potential detected, with the Mean Action Potential shape that is computed by using a low-pass filter. Then, the correlation is enable to determinate the measure between the two Action Potentials shapes, each time and Action Potential is detected. Thereby, having a measure where we can know if the Action Potential detected is similar in shape to the Mean Action Potential shape by observing the correlation shape generated. The common equation of correlation is presented in Equation 3.9.

$$Corr_{x,y} = \sum_{n=0}^{N-1} x[n]\, y[n] \tag{3.9}$$

Here, $y[n]$ has a specific length N and is the known shape, while $x[n]$ is shifting signal which elements change with each iteration, in each iteration a new samples is taken and a new correlation value is computed. Therefore, if we correlate all the samples of both Action Potential shape detected which will be the $x[n]$ signal and Mean Action Potential which will be the $y[n]$ signal commonly called target signal, a positive value will be presented when both signals $x[n]$ and $y[n]$ match Strauss (2000).

If there is noise in the signal, also the correlation signal will present a certain noise level. In addition, the correlation result will have twice the size of Action Potentials peak, due to both phases of the Action Potential will generate a positive value when Action Potential shapes match which other. That is, the peak generated by the correlation will be high over the the noise level. The use of this technique two detect a certain shape is also called as ***matched filtering*** and when a signal is correlated with itself the output signal is called autocorrelation Strauss (2000).

3.5.1 Correlation Patterns

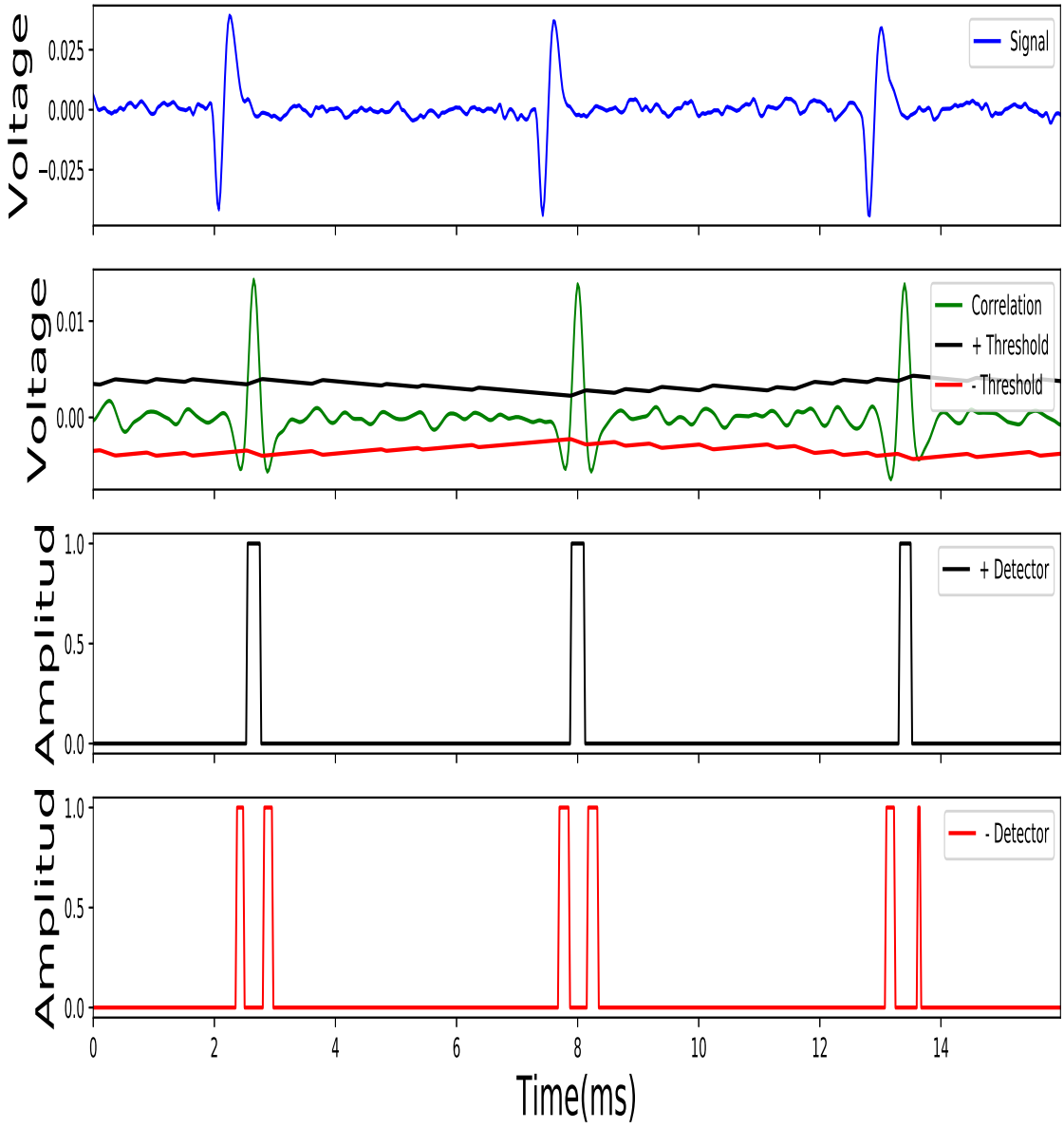


Figure 3.16: First, Action Potentials detected are illustrated. Second, the correlation shapes are computed and detected using the + Threshold and - Threshold. Finally, the + Detector and - Detector trigger each time a positive or negative phase is above or under the thresholds.

The result of the correlation between the Action Potential detected and Mean Action

potential shape is shown in (Figure 3.16). The correlation shape is the result expected, with a positive value when both shapes match. Nevertheless, two negatives values are generated, this is the result of both phases a negative and positive been correlated, the negative value tell us that while one phase increase the other one decrease. This is the result of both phases been correlated the positive one and the negative one, two times when the signals is shifted.

We observe in both biosignals Macaque monkey and Human pancreatic that six possible patterns could be easily detected, by using the same technique of two adaptive thresholds as in (Figure 3.12), but now we applied this to detect Correlation Patterns and theirs phases, such as the positive that tell us that an Action Potential is similar to the Mean Action Potential computed, and the negative ones, that tell us when both phases exist in the Action Potential detected. The six patterns observed are illustrated in the next diagram in(Figure 3.11).

The correlation signal generated has a certain noise level. So, one more time we compute these two thresholds to detect the correlation peaks above and under the noise level, in order to detect each of the six Correlation Patterns.

3.6 Simulation

The system was simulated in Python 2.7, a free software, where many packages for signals processing have been created. Two scripts was generated in python, one for each signal, and their respective Action Potentials as their Correlation Patterns where detected. We looked forward the maximum voltage value in the Action Potential shape, in that way we save the sample number of this maximum value. Finally, the correlation pattern is detected and this pattern determinate where to save this detection within the six patterns, as shown in (Figure 3.17).

In these scripts, two mean Finite State Machines was designed, one for the detection of Action Potentials shapes, and the second one to detect the Correlation Patterns and classified the Action Potentials among the six possible patterns. The general diagram of the first Finite State Machine (FSM), for Action Potential Detection, is shown in (Figure 3.18), where the control of the states of the FSM is controlled for the inputs P which is 1 when the positive detector is triggered, and state in 0 when no detection takes place. In the same way N with negative detections. Finally, C is a signal for a counter who is triggered to 1 when no trigger event took place in both detectors. For example, for the Action Potential of Macaque monkey signal we used 32 samples to properly save its shape, so in order to know that not trigger event took place, the counter have to trigger before the positive or negative detectors. On the contrary, an Action Potential has been detected and saved.

The second FSM, for detecting the Correlation Patterns, follows the same principle of Action Potentials shapes as aforementioned. The General FSM diagram for Correlation Patterns is shown in (Figure 3.17). Where, in the same way, the shapes are detected for two detectors a positive and a negative one. The second FSM is enable when a Action Potential is detected in the first FSM. Afterwards, the second FSM detect the correlation pattern generated. This generation can be see in the correlation signal in (Figure 3.16). The detection of these Correlation Patterns is made in a similar manner, when the detectors trigger with the sequence shown in the general diagram in (Figure 3.16).

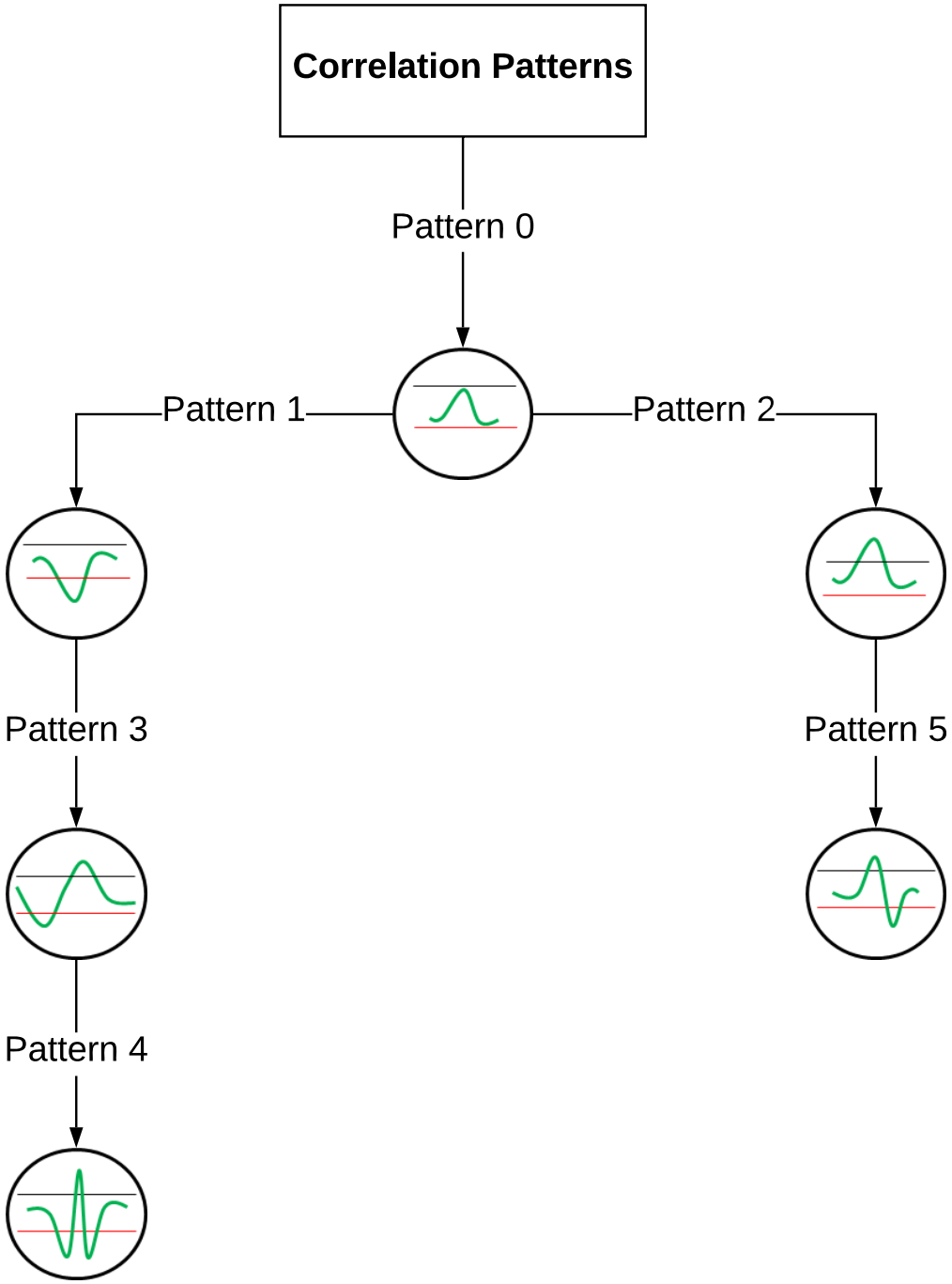


Figure 3.17: Correlation patterns to be classified and detected.

Consequently, the FSM wait a certain time, commonly the half width of a Action Potential shape. Finally, the Correlation Pattern is detected and the Action Potential is classified.

In the next (Figure 3.20), we show some detections made in Python for Macaque monkey Action Potentials (3.20a), where 4 Correlation Patterns were detected. the pattern 4 (3.20b), have the majority of detections with similar shape. Pattern 3 (3.20d) classified small Action Potentials. In pattern 5 (3.20e), an Action Potential shape for pattern 4 was saved, this could happen at the beginning of the classification or with the first detection, due to the Mean Action Potential shape computed does not have enough detections. Finally in patter 2 (3.20c), some detections that triggered both thresholds

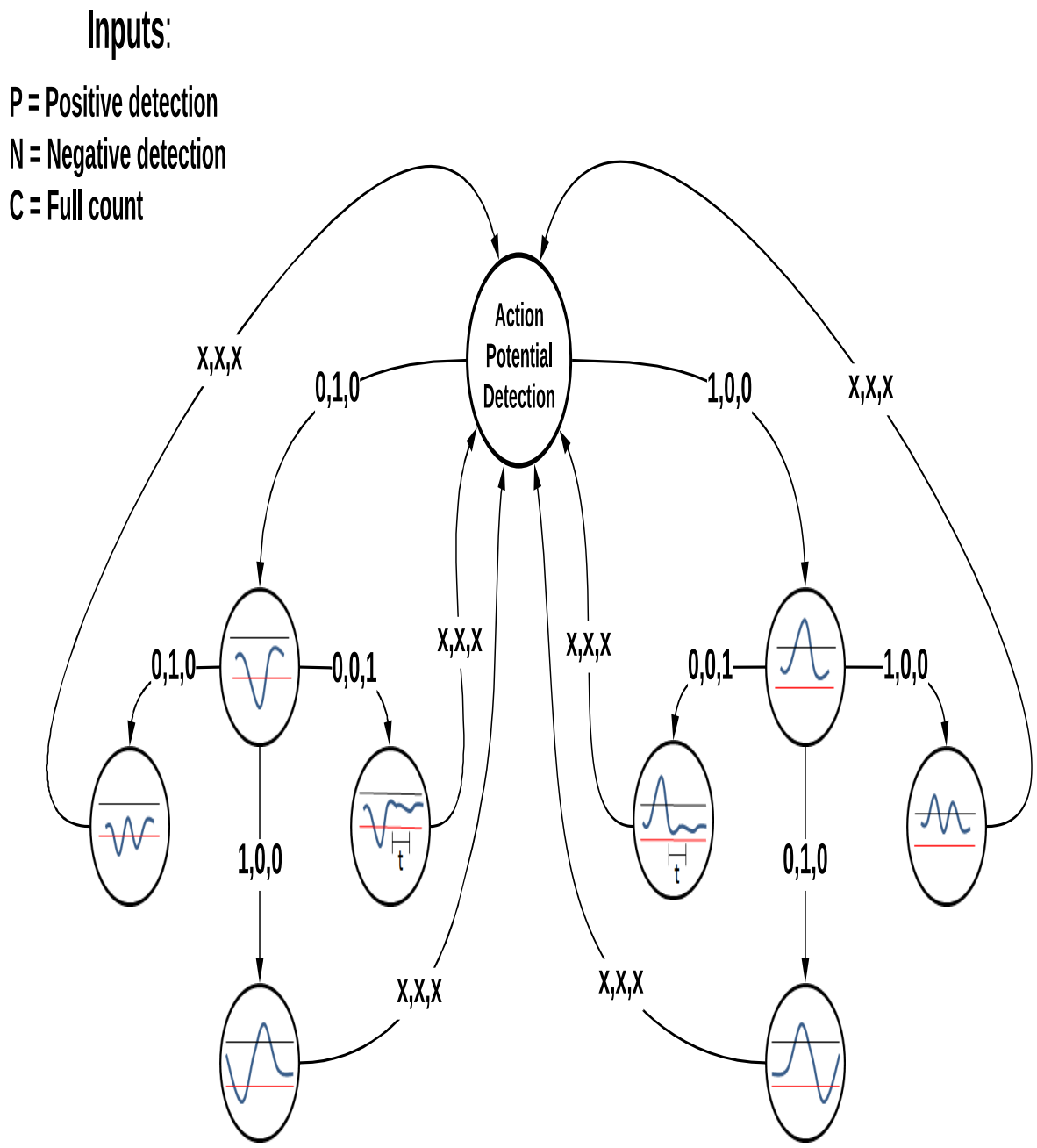


Figure 3.18: General FSM diagram for Action Potential detection.

are shown. Detections in patterns 2,3,5 possibly are Action Potentials for cells that were far enough for the recording electrode, due to their low amplitude.

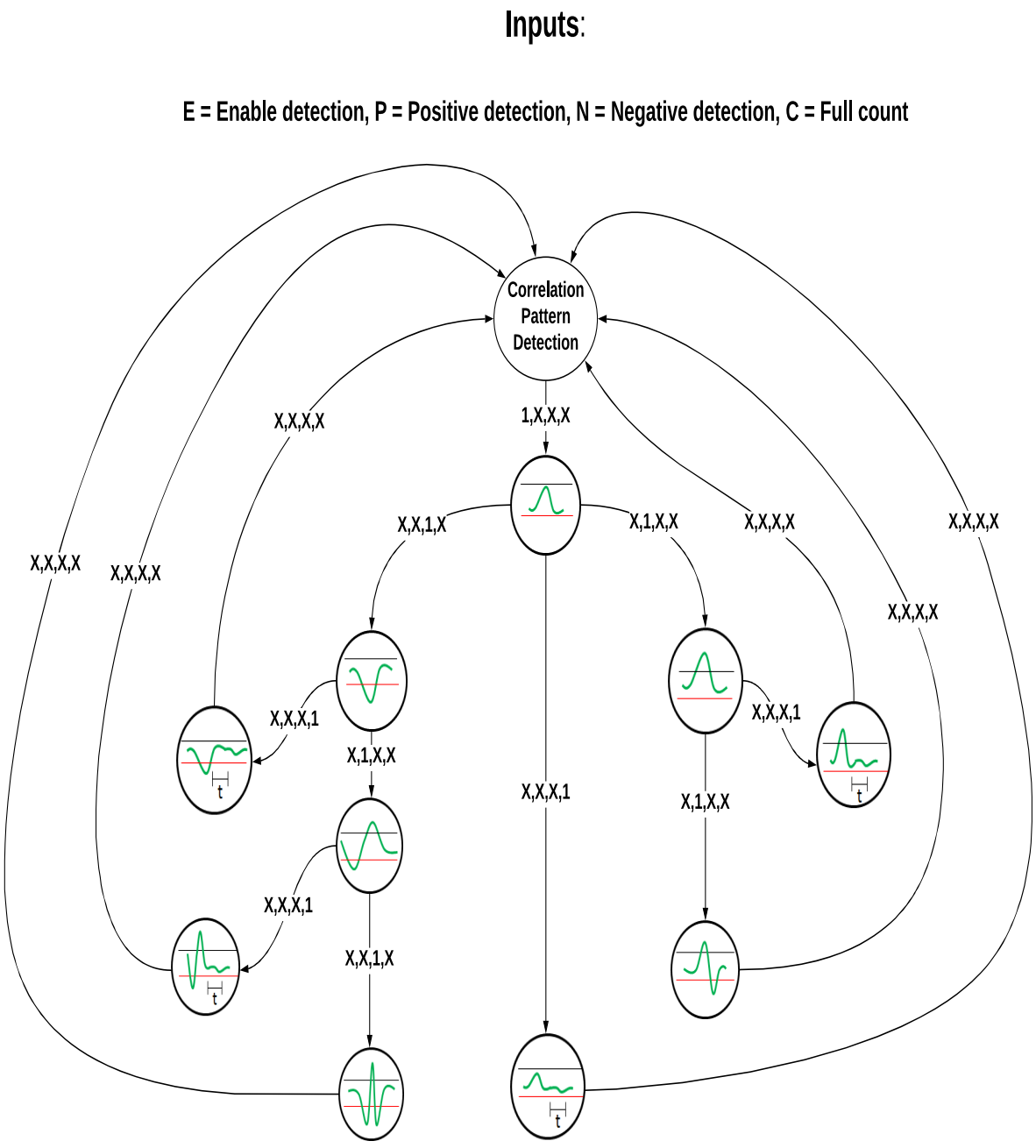
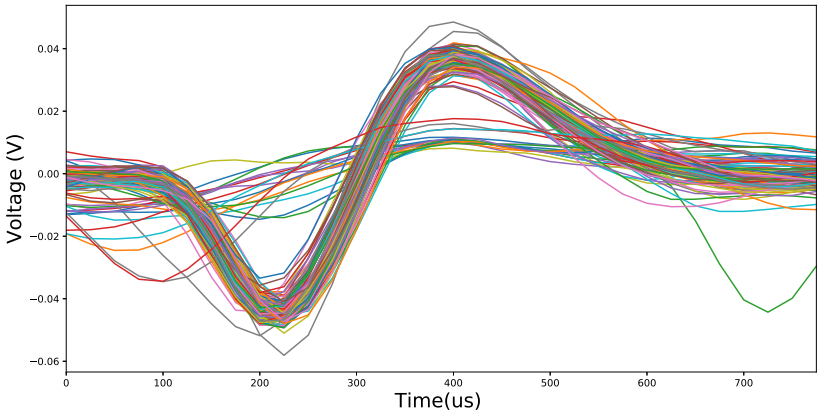
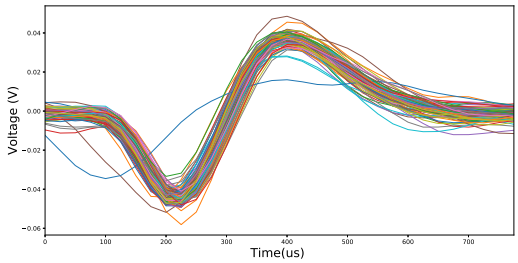


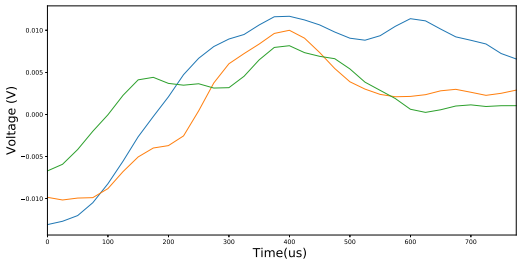
Figure 3.19: General FSM diagram Correlation Pattern detection.



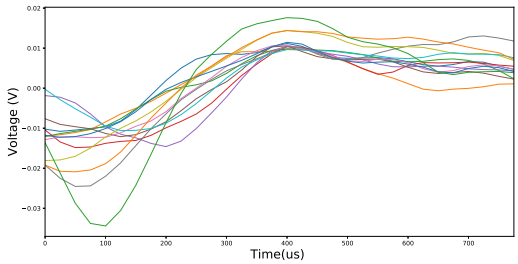
(a) Action Potentials detected. 118 detections



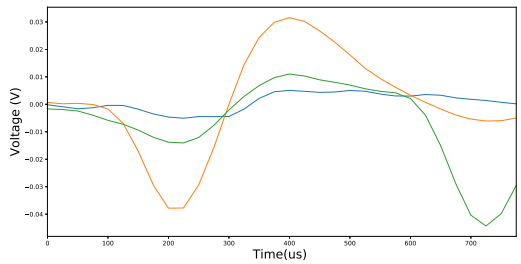
(b) 99 detections classified in Pattern 4



(c) 3 detections classified in Pattern 2



(d) 13 detections classified in Pattern 3



(e) 3 detections classified in Pattern 5

Figure 3.20: In (a) Action Potential detections are shown. In (b) are shown the Action Potentials classified in Correlation Pattern 4. In (c) the ones classified in Correlation Pattern 2. In (d) the ones classified in Correlation Pattern 3. Finally in (e) the ones classified in Correlation Pattern 5. The rest of Correlation Patterns not presented Action Potentials.

– Change your thoughts and you change your world –
– Norman Vincent Peale –

4 Hardware Implementation

4.1 Description

The general hardware architecture diagram is shown in (Figure B.1). Here, an Universal asynchronous receiver-transmitter (UART) manages the start, stop and pause of the system in the next way: when ***a*** is pushed in the keyword, the read process of the biosignal in the SD card is initialized and the detection and classification process are running. If ***b*** is pushed in the keyword, the system stops the read process. Finally, if ***c*** is pushed, one sample will be read at the time and processed for all the processes in the hardware architecture, and after the system will stop. Then, after if ***a*** is pushed,the system will be running one more time or also by pushing ***c***, for reading only one sample more at the time of the biosignal from the SD card module. This process are controlled for a FSM that synchronise the FSM that exist in the whole architecture.

The SD card module gives us a sample of the biosignal, with a certain frequency, for example for emulate a real-time process, we can create a counter that gives a true pulse each 40 KHz which is the frequency sample rate for the Macaque monkey biosignal. Finally, this module will give us a sign through a certain signal that will tell us that the sample biosignal is ready in output of the module.

The filter section contains the IIR filters mentioned in section 3.2 for accentuating the Action Potentials of both biosignals. Consequently, the Action Potentials are detected and classified in the detection and classification parts where the Maximum value from each Action Potential shape is stored in the FIFO memories and as well as Correlation Pattern detected. The Counter section contains the number of Action Potentials detected, as well as the number of detections classified in each of six Correlation patterns as is shown in (Figure 3.17). In addition, also de False Positives and False Negatives are stored and counted.

The Pmod OLED part is able to display the detection thresholds for Action Pontential or the thresholds for Correlation Patterns detection. In addition, both signals are displayed, the filtered biosignal and the Correlation signal where Action Potentials and Correlation Patterns can be observed in this Pmod OLEDrgb display. The use of this

module permit only four signals that can be displayed at the same moment. Finally, the 7-segment display is used to show the detections saved in each FIFO memory output as well as their Correlation Patterns and its last value saved in FIFO memories, as well as the output of each counter to know exactly how many detections has been detected and classified. In addition, we also added counters to know the number of false negatives and false positives during the detection process.

4.2 FPGA (Field-Programmable Gate Array)

FPGA is a reconfigurable integrated circuit. Commonly this circuits have the advantages of parallel processing, this feature is one of the reasons why FPGA are attractive in the academic and industry level. Their flexibility to be reconfigured is a big advantage, where only changing some parts in the descriptive hardware language, the architecture designed can be modified without problems and without the need of buying a new integrated circuit.

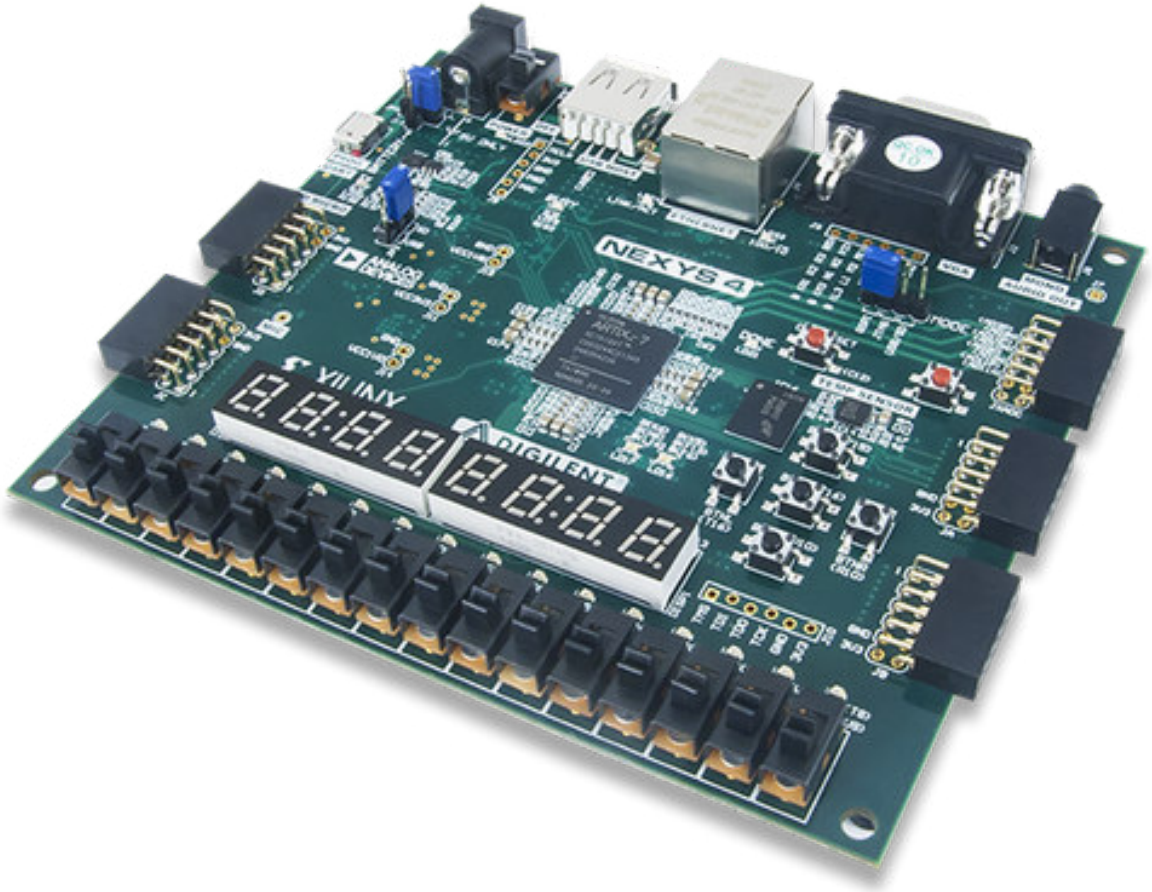


Figure 4.1: FPGA Nexys 4. Ilustration from Digilent.

FPGA contains Configurable Logic Blocks (CLBs), as well as Input/Outputs blocks (IOB) which can be connected to any of the CLB by a hierarchy of reconfigurable interconnections. CLB can be configured to perform simple or complex logic functions, as well as memory resources in the FPGA by describing Random Access Memory (RAM), Read Only Memory (ROM) and flip-flops. In this work we use a Nexys 4 FPGA to implement our architecture for biosignal processing by using the ISE Design Suite. This

FPGA is shown in (Figure 4.1).

4.3 VHDL (VHSIC hardware description language; VHSIC: very-high-speed integrated circuit)

VHDL is a hardware description language (HDL), where the behaviour or structure of electronic circuits is described and compiled. Commonly, VHDL is used in Complex Programmable Logic Device (CPLD), FPGA and Application-Specific Integrated Circuit (ASIC) fabrication. Although, it exist several computer programming languages, such as C, Fortan, Java, Python, they are not able to describe digital hardware, due to their limitations. In addition, these programmable language are designed in sequential form.

On the contrary, hardware description language for describing digital hardware needs to be build in small parts, wiring and connecting the inputs and outputs of each digital circuit to their respective ports, the signals are connected to each little part have a certain block of instructions running concurrently, each block of instructions has its respective delay and timing to be completed. Therefore, the use of traditional sequential languages can not emulates this features, and the use of HDL is need for properly describing digital hardware Pong P.chu (2006).

4.4 VHDL structure

VHDL scripts commonly starts with the declaration of the IEEE library. Afterwards, some important packages are included as ***std_logic_1164*** which enable the use of types as ***std_logic*** and ***std_logic_vector***, and some boolean operations. Finally, in the headers the package called ***numeric_std*** which defines a set of numeric types as ***signed*** and ***unsigned*** for representing magnitude numbers or signed numbers (2's complement).

After, the entity is declared which is the name of the circuit designed and inputs and output ports among other types. Consequently, in the architecture part is defined the content of the circuit and processes that describe the behaviour of the circuit and signals which manage the data values around the architecture.

For creating structures of a module is useful two connect sub-modules to easily built a complex architecture, this sub-modules are commonly called components. This components are connected by a sentence called ***port map*** which connect each port of the sub-modules with internal signals in the architecture to create a top-module, in that way the code to type is reduce and any module pre-designed could be connected and reused.

4.5 Basic memory components

Commonly the basic components of memory for design any architecture are registers , RAM and FIFO memories. In (Figure 4.2), we show the basic register module that we reused in each part of the hardware architecture.

LDR is the signal that control the load of data with a length of ***K*** the word size. ***D*** is the value to be loaded. ***CLK*** is the clock signal, ***RST*** put the output Q of the

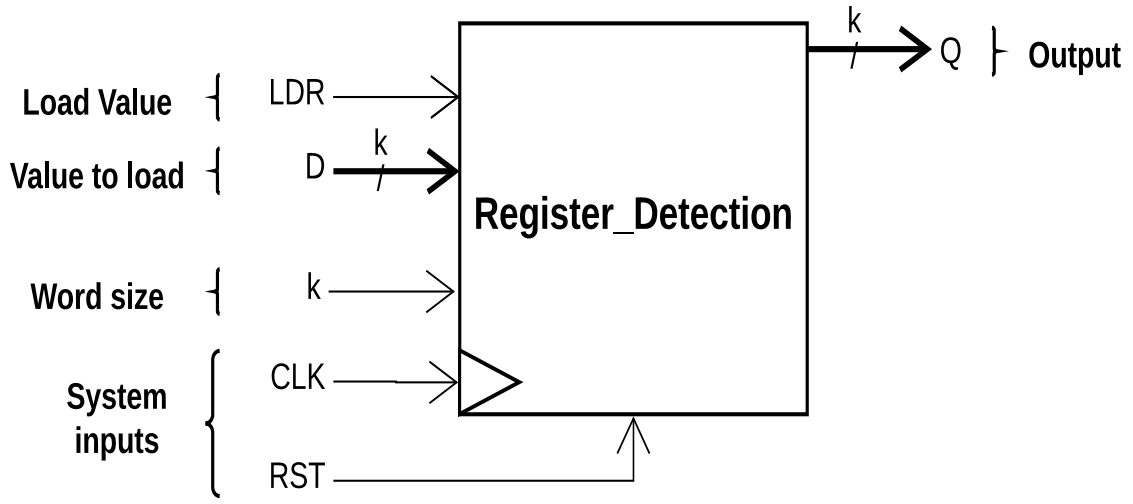


Figure 4.2: Register module.

register to 0 when receives a logic 1 in the next clock cycle. Finally, Q is the output value that hold the register until a new value is loaded.

The use of RAM memories is needed in this hardware architecture, due to we need to save four Action Potential shapes, one for the present detection, another for the past detection and another for the last shape computed for the IIR low-pass filter and one more for the result. The module for only reuse the RAM memory module in the hardware architecture is presented in (Figure 4.3).

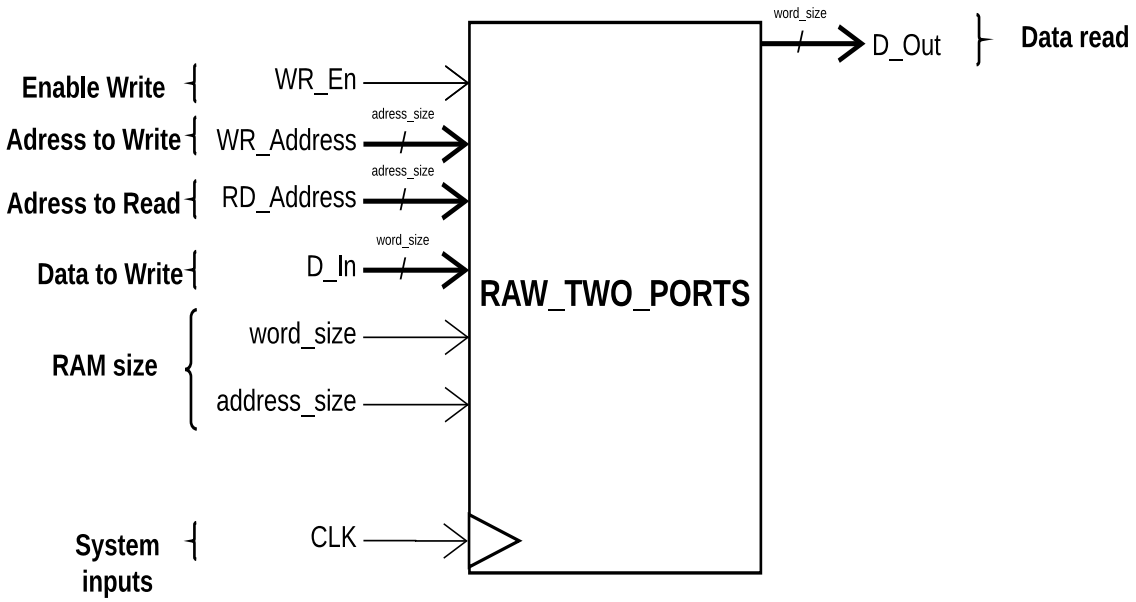


Figure 4.3: Two ports RAM memory module.

This RAM memory modules contain two ports one for the data to be saved ***D_In*** with its respective address port ***WR_Address***. The output port of the RAM memory

is ***D_OUT*** which will show the data saved in the address value ***RD_Adress*** one clock cycle after, due to this is a good practice when RAM memories are described Rushton (2011). The ***word_size*** input is a integer value that determinate the width of the data. Finally, ***address_size*** determinate the size of the address bus of the RAM memory locations, commonly a value of a power of two (2^n) is recommended, due to is easier to be infer by the synthesis process.

The use of First-In First-Out (FIFO) memories is important when some process work with different clock frequency. Therefore, the detections of Action Potential can be stored in these memories for reading their values with a different clock frequency by other processes and no data will be lost. The FIFO module used in the hardware architecture is shown in (Figure 4.4).

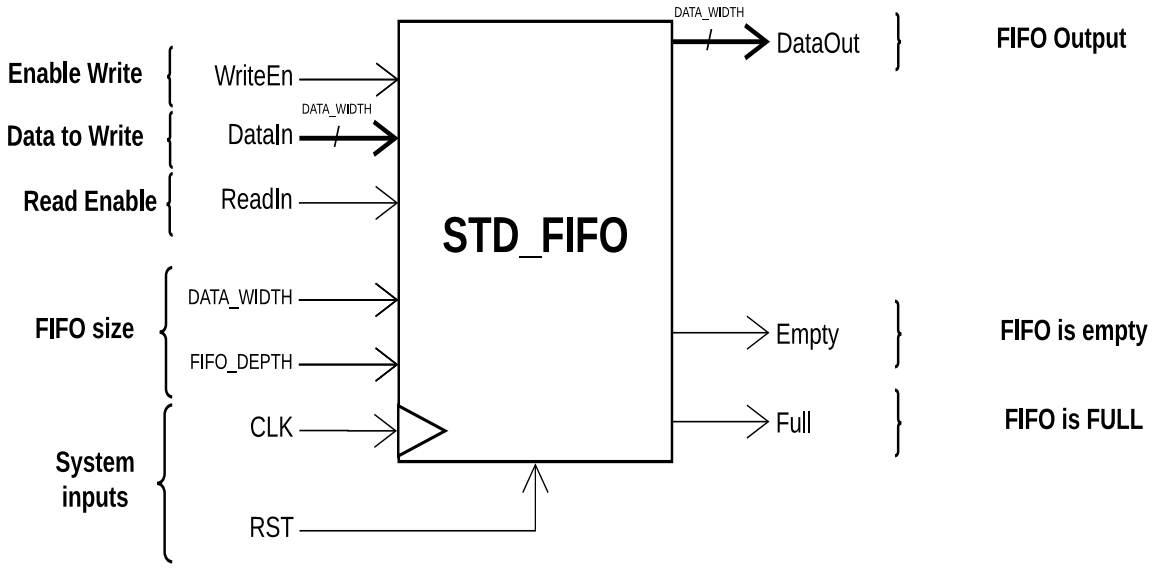


Figure 4.4: FIFO memory module

As well as the RAM memory, FIFO memories contain two ports ***DataIn*** where the data to write is put, and ***DataOut*** where the first value that was enter will be the first value to go out, when ***Readin*** have a true value of 1. Internally, the FIFO memory brings two counters one commonly called tail and another called head which respectively bring the number of data read and written in the FIFO memory. These counters activate the flags ***Full*** and ***Empty*** when the memory is full or empty. Finally, ***DATA_WIDTH*** is the word size of the data and ***FIFO_DEPTH*** determinate the number of FIFO memory locations.

4.6 Toplevel module

We designed two main projects one for the Macaque monkey biosignal and another one for the Human pancreatic biosignal. We called this two modules ***Monkey_module*** and ***Pancreatic_module***. The only differences is that in the pancreatic module we detect positive Action Potentials and two-phases Action Potentials that have a positive and a negative phases, as well as the False positives and False negatives. Therefore, we used one more FIFO memory. Both modules are designed in the same way as it is mentioned in this section that describes the components and sub-modules in the Toplevel module. The Toplevel module for the Macaque monkey biosignal designed is shown in (Figure 4.5) and for the the Human Pancreatic biosignal in (Figure B.2). These modules were built with eight sub-modules and one global FSM that controls and coordinate the Action Potential detection and classification. The sub-modules utilized are:

- ***UART_recv***
- ***UART_send***
- ***PmodOLEDrgb_sigplot***
- ***periodic_timer***
- ***Sdcard_readstream***
- ***IIR_FILTER***
- ***Top_level_Threshold***
- ***Detection_and_Classification***.

The generic inputs of this module are explained as follows:

- ***Input_Frequency***. It is the frequency at which we provide new samples in the the ***Sdcard_readstream*** module. This input goes to the ***periodic_timer*** module which produces a pulse that indicates to the ***Sdcard_readstream*** module when has to give a new sample. This input value has to be in hertz.
- ***N_Threshold***. It is the adjustment value for the positive and negative thresholds. The higher the value the higher the positive threshold and the lower the negative threshold. The range of this adjustment is in a integer range of 0 to 15. The changes of the thresholds can be seen in the Pmod OLEDrgb display when ***Switch_3*** input is 0. These thresholds are used to detect the Action Potentials.
- ***K_Adjustment***. It is the adjustment of the standard deviation computed to be compared with the band-pass filter output. The range of this inputs goes for 0 to 16383. Although, we recommend the use of values such as 10,100,1000,1000. The default value is 1000 whose performance was better with both signals that we use to compute their standard deviation value.
- ***N_Threshold_C***. It is the adjustment value for the positive and negative correlation thresholds, the higher the value the higher the positive threshold and the lower the negative threshold. The range of this adjustment is in a integer range of 0 to 15. The changes of the thresholds can be seen in the Pmod OLEDrgb display when ***Switch_3*** input is 1. These thresholds are used to detect the Correlation Patterns.

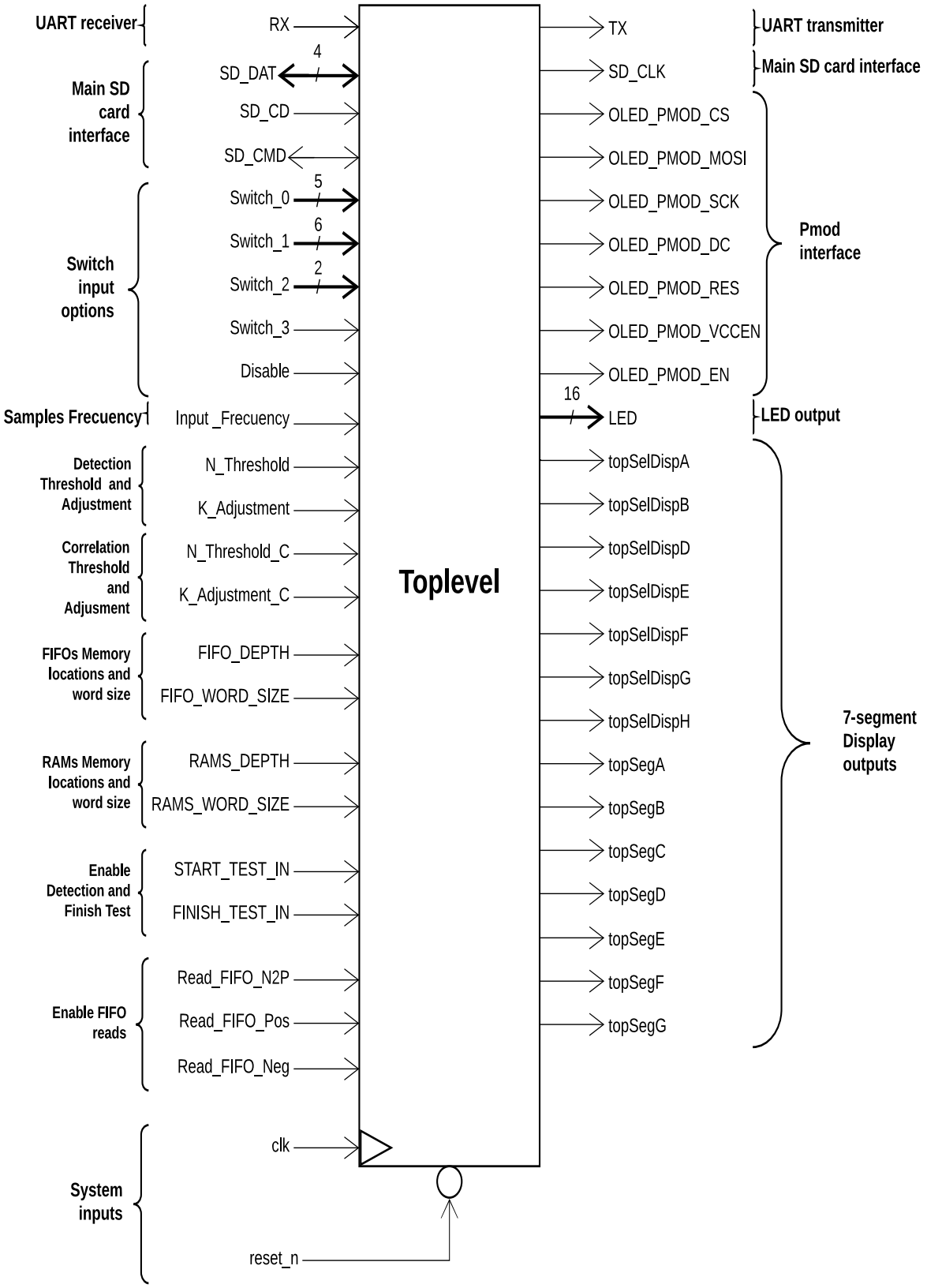


Figure 4.5: Toplevel module (Monkey_module).

- ***K_Adjustment_C***. It is the adjustment of the standard deviation computed to be compared with the Correlation signal output. The range of this inputs goes for 0 to 16383. Although, we recommend the use of values such as 10,100,1000,1000. The default value is 100 whose performance was better to properly compute their

standard deviation value.

- ***FIFO_DEPTH*** .It is the deep of the FIFO memories or memory locations available in each FIFO memory.
- ***FIFO_WORD_SIZE***. It is the size length of the detection samples saved in the FIFO memories. Their default value is 32-bit.
- ***RAMS_DEPTH***. It is the deep of the RAM memories or memory locations available in each RAM memory. This memories are used to compute the Mean Action Potential shape. This value is the exponent number of a power of two. That means ($2^{R}AMS\_DEPTH$).
- ***RAMS_WORD_SIZE***. It is the size length of the samples saved in the RAM memories. Their default value is 16-bit which is the same length of the samples coming from the SD card.
- ***START_TEST_IN*** .Here, this value has the range between integer range 0 to 4095. The number input is the sample number at which the detection process and classification process are enabled.
- ***FINISH_TEST_IN***. This input determinate when the detection and classification processes end. When this happens the module will stop all the process.

The Global FSM that controls all the process is presented in (Figure B.3). The main states are explained as follows:

- ***Waiting***. This states waits for the ***timer_strobe*** signal which comes from the periodic timer module and the ***Authorized*** signal which comes from the FSM that controls the UART communication and passes to the next transition the ***next_sd_data*** state when both signals are set to high.
- ***next_sd_data***. This state indicates by the ***read_sample*** output to the ***Sdcard_readstream*** module when a new sample needs to be read. Consequently, the FSM passes to next transition the waiting_SD state where until the ***data_ready*** input stays on high. Then, the new sample is ready to be read from the data_out port output of the ***Sdcard_readstream*** module.
- ***passing_Low***. This state enable the ***IIR filter*** module by the ***read_sample*** output for computing a new filtered value when the data is ready in the ***data_out*** port output from the ***Sdcard_readstream*** module. Consequently the FSM passes to the next ***waiting_Low*** state where the filter module is waited until the ***ready_IIR*** input is set to high. Then the high-pass filter value is ready too.
- ***passing_Low***. This state enables the second IIR filter module which compute the value for the ***Band-pass*** filter when the ***read_sample_Band*** output is set to high. Therefore, the FSM passes to the next transition the ***waiting_Band*** state where the module is waited until the ***ready_Band_IIR*** input is set to high by the Band-pass filter module indicating that the filtered value is ready to be read.
- ***Detection***. The state enables the compute of a new Threshold value when ***read_detection*** output is set to high. This new Threshold value computed is used to be compared with the Band-pass filtered value for detecting Action Potentials. In addition, when the ***read_detection*** output is triggered

also the Correlation compute is enabled, and after the FSM passes to the ***waiting_Corre_Compute*** until the ***ready_Corre_Compute*** input is set to high, it indicates that the Correlation value is ready to be read for the ***Top_Level_Threshold_Correlation*** module and passing to the next transition the ***start_Corre_threshold*** state where new Correlation threshold value is computed in the ***Detection_and_Classification*** module when the FSM passes by the ***waiting_Corre_Detection*** state, hereby waiting for the Correlation threshold compute which is ready when the ***ready_Corre_Threshold*** input is set high.

- ***Plots***.This states has to possible transition depending on the ***Disable*** input value which comes from the switch inputs in the FPGA board. The first transition is when the Disable input is low, due to when this input is low the FSM machine does not wait for the ***Pmod OLEDrgb*** module to properly plot the signals, so the system is running emulating a real-time acquisition processing each sample read thought the frequency which is proportionate by the ***periodic_timer*** module. The second transition is when the ***waiting_OLED*** state is reached, that is when the ***Disable*** input is set high. Consequently, the FSM waits until all the samples has been input into the ***Pmod OLEDrgb*** module, therefore it waits for the ***Shift_done*** input, when this is set high indicates that the samples in display have been properly shifted .

4.6.1 UART modules

The purpose of these two sub-modules is to communicate the FPGA by a UART controller to star, stop and pause the intern process in the ***Topmodel*** module. The UART receiver module is shown in (Figure 4.6). Here, the data is sent by the keyboard in the common communication protocol RS-232. The data transfer rate is 115200 kbps. When the UART receiver module validate the data in the UART line, the ***dat_en*** output is set to high for one clock cycle and the chart data is ready in the***dat*** module output.

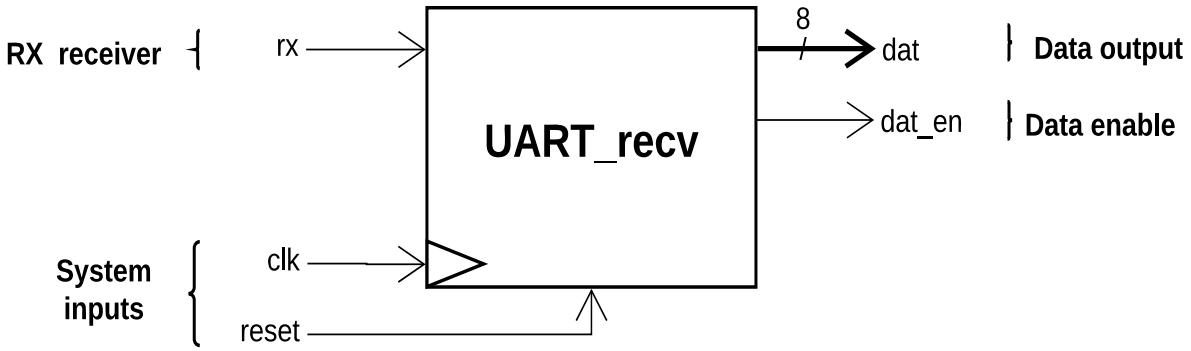


Figure 4.6: UART receiver module. author's module: Yannick Bornat

After, the UART sender module, shown in (Figure 4.7), receives the pulse generated from the UART receiver module by the ***dat_en*** module input, and the chart data in the ***dat*** module input is sent to be display in the terminal. In our case we use the ASCII values ***01100001*** (a) to start the process, ***01100010*** (b) to stop and ***01100011*** (c) for reading only one sample at the time.

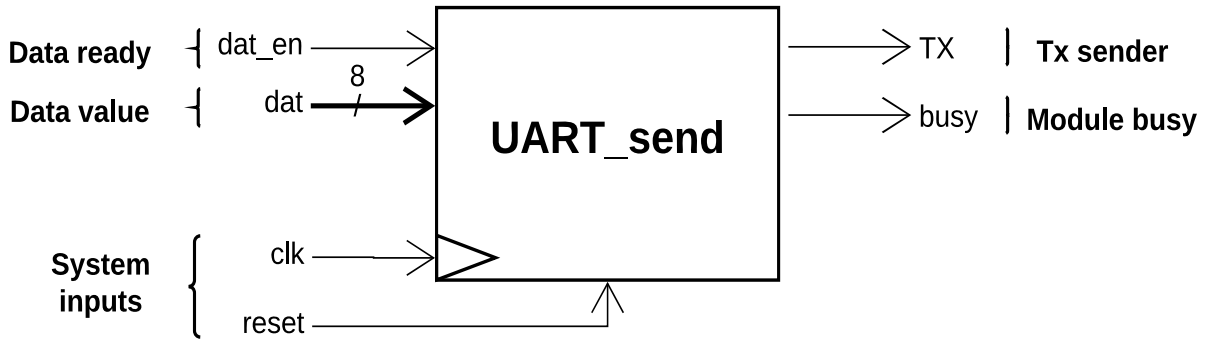


Figure 4.7: UART sender module. author's module: Yannick Bornat

The FSM that controls both modules is shown in (Figure B.5) and its main states are explained as follows:

- ***stopped***. When the FSM stays in this state, the ***Authorized*** output is set low, then the whole system is stopped, such as the detection and classification process which are enabled by the FSM explained in (section 4.6). The possible transitions are the ***running*** or ***single*** states. The ***running*** state is reached if the ***data_en_from_UART*** input is set to high and the character sent in the ***data_from_UART*** input signal is the ***a*** character. Similarly, the ***single*** state is reached if the ***data_en_from_UART*** input is set to high and the character sent in the ***data_from_UART*** input signal is the ***c*** character.

- ***running***. As aforementioned this state is reached when the ***data_from_UART*** input signal is the ***a*** character. The transition to the ***stopped*** state is reached in four possible ways. The first one when he ***data_from_UART*** input signal is the ***b*** character. The second one is when the ***Disible*** input value is ***01*** or ***10*** and the ***Detected*** input signal is high triggered, indicating that a Detections has been detected by the system, such as an Action Potential, False Positive or False Negative. Finally, the fourth one when the ***read_Finished*** input is set to high when the sample counter reach to the sample number entered by the generic ***FINISH_TEST_IN*** input in the ***Toplevel*** module. In addition, the ***single*** also is reached when the ***data_from_UART*** input signal is the ***c*** character and the ***data_en_from_UART*** input signal is set high indicating that a new character value has been typed in the keyword. Besides, when the FSM stays in this state, the ***Authorized*** output is set high by indicating that the samples have to be processed for the different modules aforementioned in (section 4.6).

- ***single***. As aforementioned this state is reached when the ***data_from_UART*** input signal is the ***c*** character. Also, such as the passed states both the ***stopped*** the ***running*** states are reached. The ***stopped*** states is reached after a sample is read from the SD card module which is indicated by the ***timer_strobe*** input signal which comes from the ***periodic_timer*** module, in that way, only one sample is read at the time. In addition also it is reached when the ***b*** character is sent by the ***data_from_UART*** and when the ***data_en_from_UART*** input signal is set high. Finally the as it been described the ***running*** state is reached if the ***data_en_from_UART*** input is set to high and the character sent in the ***data_from_UART*** input signal is the ***a*** character. Besides, when the FSM stays in this state, the ***Authorized*** output is set high by indicating that the samples have to be processed for the different modules aforementioned in (section 4.6).

4.6.2 Periodic timer module

This module proportionate a pulse frequency of the value entered in the ***TIMER_FREQU_HZ*** module input in Hz. This pulse is taken into account to proportionate the saved samples in the SD card by the ***Sdcard_readstream*** module. The module is shown in (Figure 4.8).

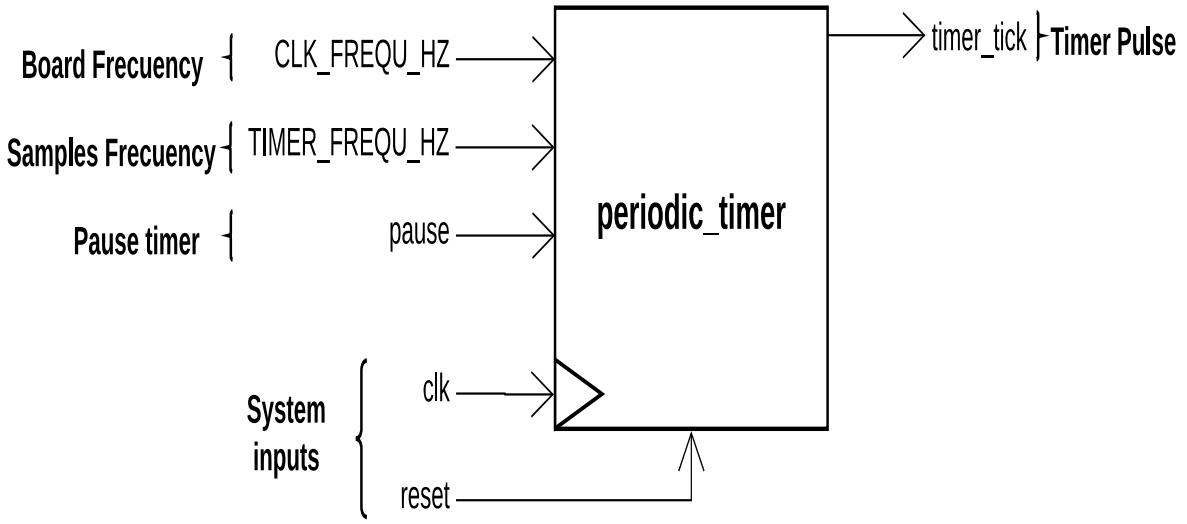


Figure 4.8: Periodic timer module. author's module: Yannick Bornat

The generic input ***TIMER_FREQU_HZ*** determinate the pulse frequency in the ***timer_tick*** output, while the generic ***CLK_FREQU_HZ*** input is the clock frequency of the FPGA. The ***pause*** input pause the generation of the pulse in the ***timer_tick*** output.

4.6.3 SD card module

The SD card modules utilized in this hardware architecture were ***SDcard_raw_access_v2*** and ***SDcard_readstream*** modules, the documentation of these modules could be found in Teaching resources - Y. Bornat- SDcard. These modules makes possible to transfer data between the local buffers in the FPGA and the SD card. Reading data for the buffers is similar to RAM access, the data can be 8, 16, 32 and 64 bits wide. In our case, we used a 16-bit data word size. The module ***SDcard_readstream*** module is shown in (Figure 4.9).

The use of this modules is very straightforward, only the ***Main SDcard interface*** signals have to be connected properly in the file with extension ***.ucf***. For reading new samples from the output ***data_out***, the ***data_read*** input have to be set high for a clock cycle and return to low. After, ***data_empty_n*** is set low and when the data is ready to be read in ***data_out*** the ***data_empty_n*** signal will be set high one more time and then the process can be repeated. The system clock frequency for the module is 100 MHz.

The code in Python than generates the .bin file is shown in (Appendix A). In this code the minimum positive value is found in the raw-data, after this value is subtracted to the whole signal. One more time, after subtracted this value, in the signal is found the minimum positive, and in this time we divided the whole signal for this second minimum

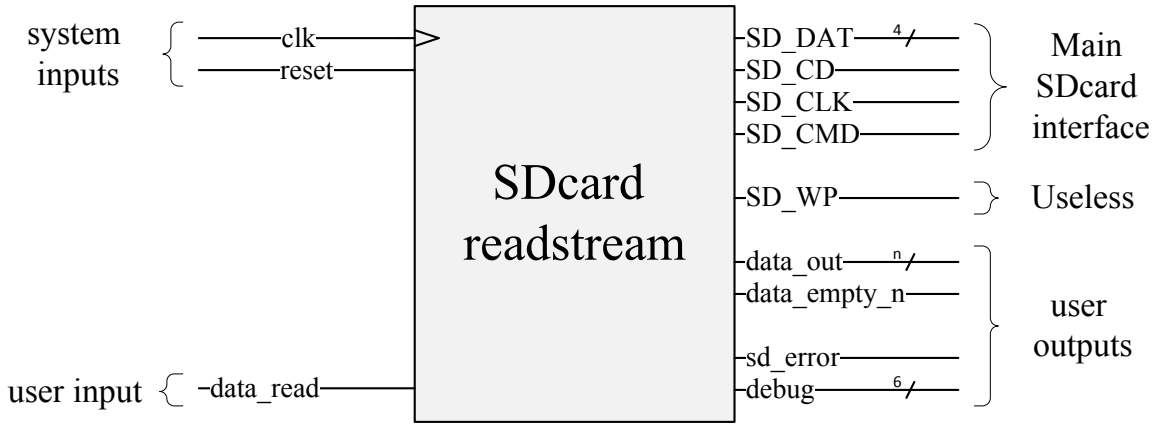


Figure 4.9: SDcard_readstream for reading biosignal samples from SD card in slot in Nexys 4 FPGA. Ilustration from Teaching resources - Y. Bornat

positive value. In that way, we will have data values close to integer values, this helps the precision of the signal, due to the data values of the raw-data will be save as two's complement with 16-bits wide.

When the .bin file has been created the next step is to write the binary file in the SD card. The easiest way is by using a software, in our case we use the win32DiskImager, in windows 7, where by only some clicks the .bin raw-data file will be written and ready to be read for the SD card module. However the use of command prompts in Windows or Linux also can be used.

4.6.4 Pmod OLED module

The use of a display in a FPGA is helpful, in our case, we use a 96 x 64 pixel RGB OLED Display with 16-bit color resolution, Diligent link, This display can plot signals, pictures and more, through a standard SPI interface. The Pmod OLEDrgb used is shown in (Figure 4.10).

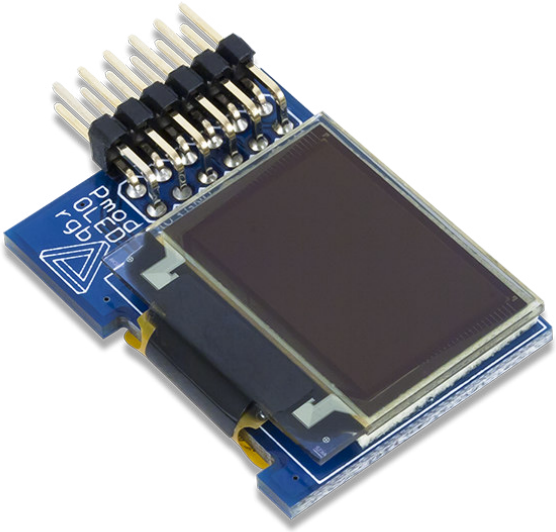


Figure 4.10: Pmod OLEDrgb features a 96 x 64 pixel RGB OLED display that is capable of 16-bit color resolution.. Ilustration from Diligent

For ploting the signals in the Pmod OLEDrgb, we used a module which its documentation can be found in Teaching resources - Y. Bornat- Pmod OLEDrgb, this module is able to plot four signals up in the OLED screen, the four signals that we showed in the OLED display were the band-pass filter output to observe the Action Potentials, the Correlation signal to observe the Correlation Patterns and the positive and negative thresholds. The ***PmodOLEDrgb_sigplot*** module which controls the signal plotting and shifting is presented in (Figure 4.11).

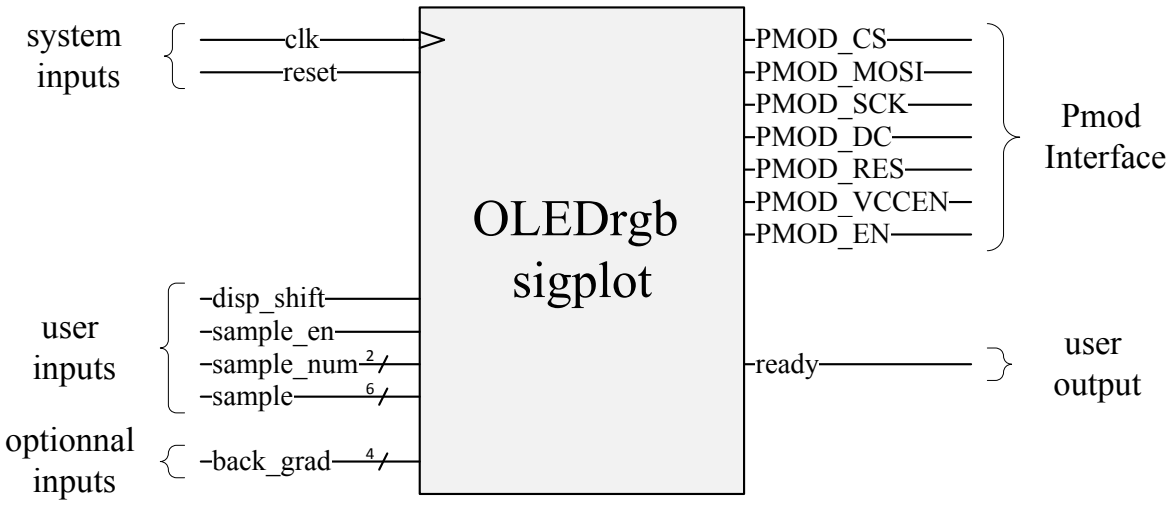


Figure 4.11: SDcard_readstream for reading biosignal samples from SD card in slot in Nexys 4 FPGA. Ilustration from Teaching resources - Y. Bornat

Here, the Pmod Interface signals are properly connected in any of the Pmod headers that contains the Nexys 4 FPGA by the file with extension ***.ucf***. To plot a signal we have to enter the sample value in the ***sample*** module input, and at the same time indicate the signal number to know where the signal belongs by the input

Table 4.1: The ***sample_num*** color value for each of the four possible signals to plot in Pmod OLEDrgb display

Sample_num	Color
00	Cyan
01	Green
10	Purple
11	Yellow

sample_num. These numbers are represented in the next Table 4.1, as well as their respective colors. Afterwards, the ***sample_en*** module input is set high each time a sample is input for each of the four signals. Finally, the plots have to be shifted to display a new sample, that is by setting ***disp_shift*** high and the cycle can be repeated.

To properly plot the signals using this module, the creation of FSM is needed such as in the (Figure B.4). The input ***Plot*** indicates when the plotting process can be started, in the next state the ***sending_sample1*** output signal ***data_en_to_display*** is set high and after waiting for the OLED_ready signal that indicates the sample has been saved to be plotted. Consequently, this states are repeated with each of the four samples until the data is plotted and shifted to be on the Pmod OLEDrgb display, this happens in the ***shift_sample*** and ***waiting_for_shift*** states, after the module send a signal through the ***OLED_ready*** input when the four signals have been shown in the Pmod OLEDrgb display. Finally, the ***Shift_finished*** state send a pulse by the ***Shift_done*** output.

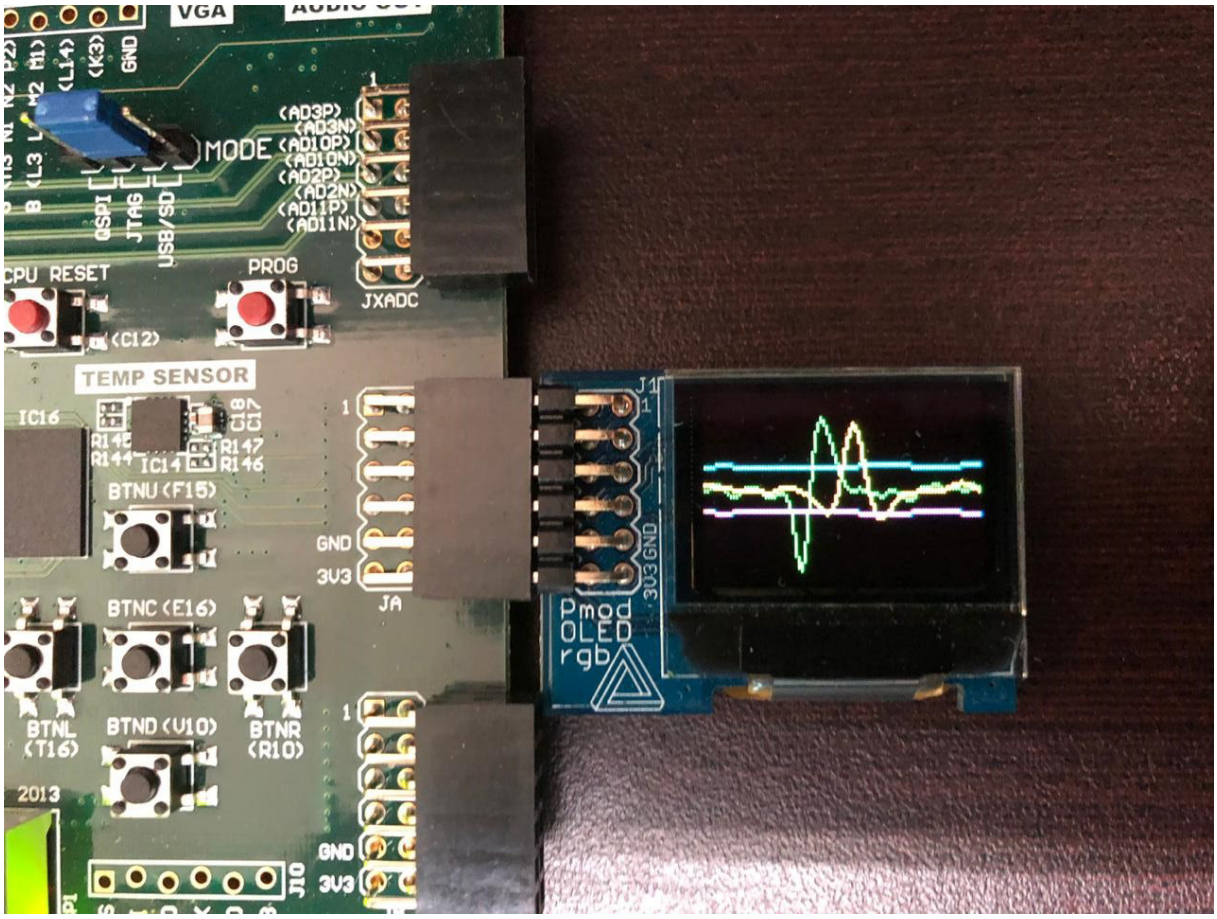


Figure 4.12: Pmod OLEDrgb display. Filtered signal (Action Potential) green, Correlation Pattern yellow, Threshold positive cyan, Threshold negative purple.

The (Figure 4.12) shows the signals plotted in the Pmod OLEDrgb display. The Band-pass filter output is the green plot where we can observe an Macaque monkey Action Potential. The yellow one is the Correlation signal computed where we can appreciate the Correlation Pattern generated to classify this Action Potential detected. The cyan one is the Positive Threshold computed, this could be the detection positive threshold or the correlation positive threshold depending on the ***Switch_3*** input. In the same way, the detection negative threshold is plotted in the purple plot, where depending on the ***Switch_3*** input, the detection negative threshold or the correlation negative threshold are plotted.

4.6.5 IIR filter module

This module is based on the Equation 3.5 where the raw data saved in the SD card is low-pass filtered this IIR filter has a internal 16.3 fix point precision where 16-bit are for the integer part and 3-bit are for the decimal part, and the output of the filter only gives the 16-bit integer part. The module is shown in (Figure 4.13). Here, the ***Sample*** input receives the data coming from the output ***data_out*** of the ***SDcard_readstream*** module. Then, the ***new_sample*** input is set two high when a new sample is ready to be filtered. Consequently, the ***IIR_ready*** output is set to low until the compute has finished and the computed value is put on the ***IIR_output*** which has a size of 16-bit.

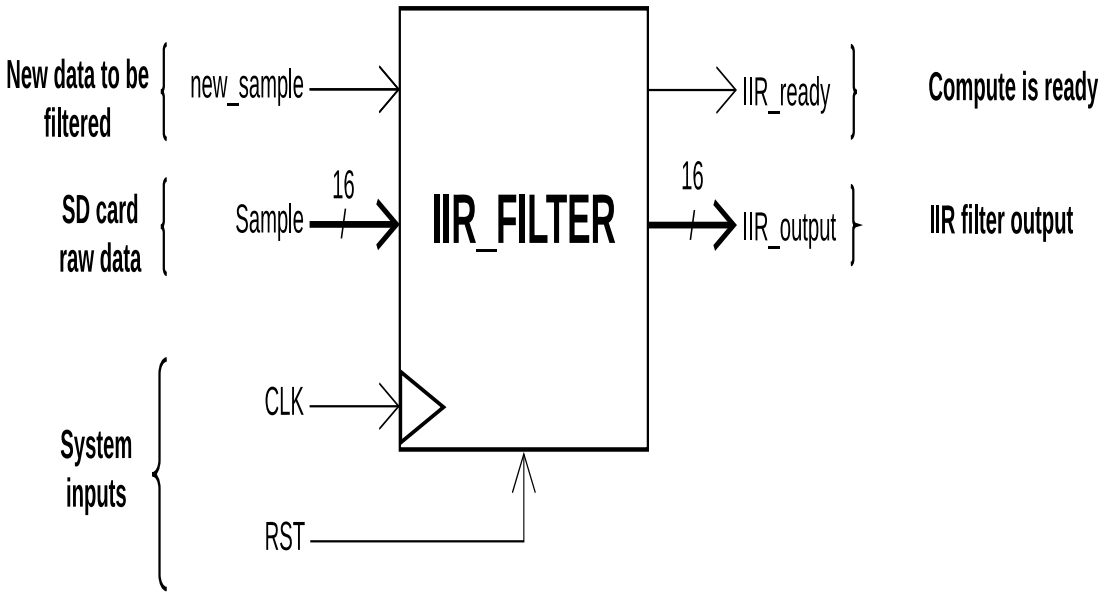


Figure 4.13: IIR filter module

The FSM that controls the compute of the low-pass filter output is shown in (Figure B.6) and the components that controls this FSM are illustrated in (Figure B.7). The main states are explained as follows:

- ***waiting***. the state passes to next transition the ***saving_past*** state until the ***new_sample*** input is set high, indicating that a new sample is ready to be processed.
- ***saving_past***. This state loads the last filtered output and last sample input in both ***Register_Yn_1*** and ***Register_Xn_1*** registers and past to the next ***saving_new_sample*** state.

- ***saving_new_sample***. Here, the new sample is loaded in the ***Register_Xn*** and in the next clock cycle the the FSM past to the next transition the ***saving_Result*** state.
- ***saving_Result*** . In this state the filtered data computed is loaded in ***Register_Result*** and the new value is ready and put on the ***IIR_output*** . Finally, the FSM reaches one more time to the ***waiting*** state triggering the ***IIR_ready*** output indicating that the module is ready to compute a new value.

4.6.6 Top level Threshold module

This module is based on the diagram in Figure 3.9 where the σ approximation is computed, the fixed point for the standard deviation value is 17.14, where only 16-bit of the integer part are taken in adjustment K part to be compared with the 16-bit IIR band-pass filter input. By the way, for the Threshold value it has a fixed point of 22.14, and the same way 16-bit are taken for the integer part in adjustment N. The main parts of this diagram are implemented in this module, a comparator between the σ approximation and the output of the band-pass filter. A low-pass filter is needed to compute the mean value of the samples that have been above the threshold. An subtracter to make the operation with the theory 15.9 % . After, two multipliers are needed, one for the K factor, and the other one for the N factor. The compute of the negative threshold is only the compute of the second complement of the (Nσ) value.

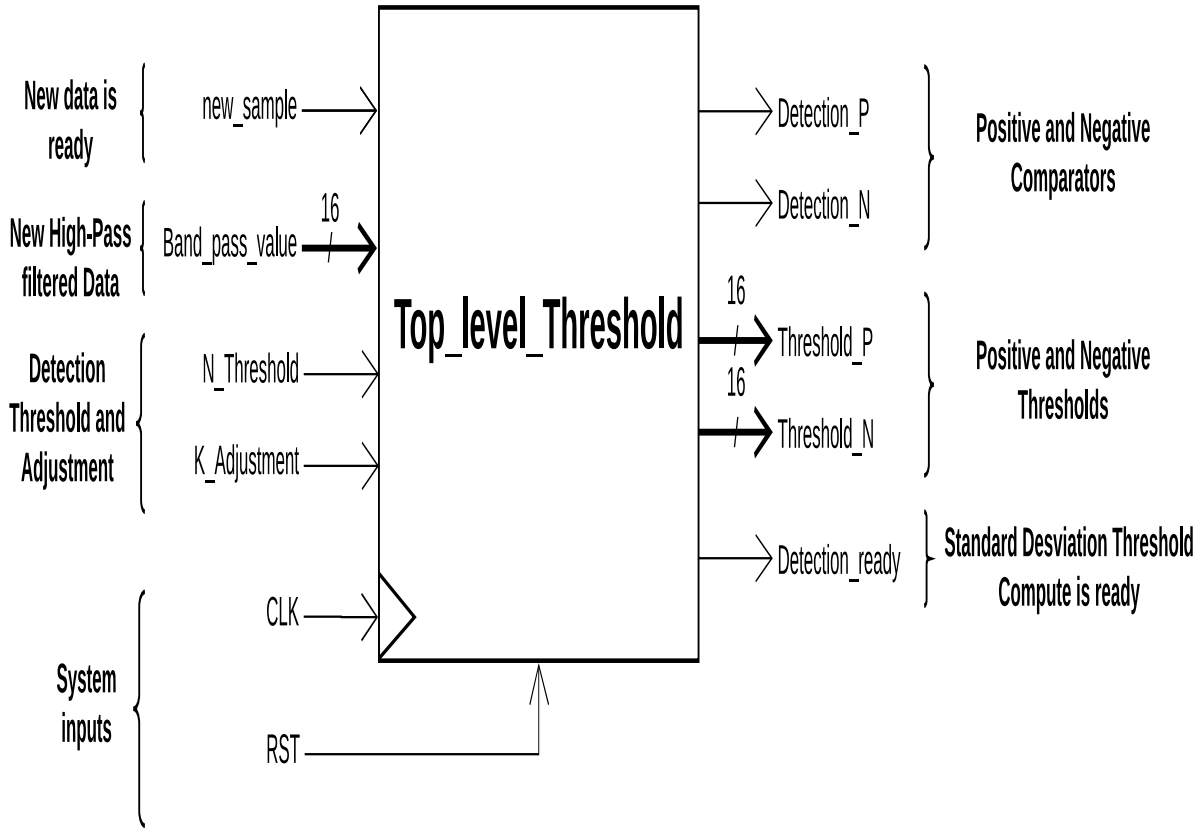


Figure 4.14: Threshold Module.

This module can be configurable with both the N factor and K factor in the ***N_Threshold*** and ***K_Adjustment*** generic inputs respectively, by computing a

better threshold but if not adjustment is needed the threshold is computed with the default values. As the passed modules, the ***new_sample*** module input is set high for one clock cycle and at the same moment the band-pass filter output is put in the ***Band_pass_value*** module input. Then, the compute of a new threshold value is computed and when the value is ready the ***Detection_ready*** outputis set high. The new threshold value is put on ***Threshold_P*** output for the positive threshold, and on ***Threshold_N*** output for the negative threshold value. Finally, if in that cycle the ***Band_pass_value*** is higher than ***Threshold_P*** value, then ***Detection_P*** output will be put it high. On the contrary, if the ***Band_pass_value*** is lower than ***Threshold_N*** value, then ***Detection_N*** will be put it high. These output are ready when the ***Detection_ready*** output module is set high and is held it on and the threshold value is put on both the ***Threshold_P*** and ***Threshold_N*** module outputs with size of 16-bit, the same size of the band-pass filter output which need to be compared.

The FSM that controls the compute of the adaptive threshold value is show in (Figure B.11) and the components that controls this FSM are shown in (Figure B.10). The main states of this FSM are explained as follows:

- ***waiting***. This state waits for the ***new_sample*** input in the module for starting to compute a new threshold value each time a new sample was read from the SD card module. The next transition is the ***enable_filter*** state.

- ***enable_filter***. This state enables the process in the ***Top_IIR_Threshold*** by the ***En_filter*** signal for computing the mean value of the samples that have been above the (σ) approximation value. When the value is ready the IIR_ready module output is set high and the FSM passes to the next transition the ***waiting_filter*** state.

- ***waiting_filter***. This state waits until the ***IIR_ready*** module output is set high, indicating that the module has finished to compute the new value, and the FSM passes to the next transition the saving_factor state.

- saving_factor. This module load the value after the Factor k multiplier which comes from the 0.159 subtraction by the ***LD_Reg_Factor_1*** signal. Consequently, the FSM passes to the ***saving_Result*** state.

- ***saving_Result***. In this ***saving_Result*** state the adaptive threshold value is loaded in ***Register_Threshold*** register by the ***LD_Threshold*** signal. Then, the Threshold values compute are set on the ***Threshold_P*** and ***Threshold_N*** module outputs, as well as the module outputs of the comparators are put on ***Detection_P*** and ***Detection_N*** module outputs. Finally, the FSM returns to the ***waiting*** state triggering the***Detection_ready*** output, indicating that the module has finished the compute.

  The next sub-module is shown in (Figure B.8). This sub-module is similar to one mentioned in subsection 4.6.5 and is part of the components needed for achieving the principle module aforementioned. The purpose is to proportionate a mean value of the number of samples above the σ computed, the output of this IIR low-pass filter is 16-bit with a 2.14 fix point representation at the output. The ***En_filter*** module input is set high when a new value for the main module is presented. On the contrary, to the past IIR module aforementioned in (subsection 4.6.5), this module only have a ***Sample*** module input of 2-bit size, due to this input is the output of the first comparator in (Figure B.10, so its output is only a binary output

(1-0) and its sign. The FSM that deal with the compute of IIR low-pass filter is shown in (Figure B.9).

4.6.7 Detection and Classification module

This module is the most extensive one designed in the hardware architecture, due to many FSM are needed for saving the maximum and minimum values, for saving the Action Potential shapes detected in the RAM memories, to compute the Mean Action Potential shape, in order to compute the correlation signal, for detecting the Action Potentials, to detect the Correlation Patterns and save and classifier the Action Potential detections. The principal module is shown in the (Figure 4.15).

The main parts that contains this module are as follows:

- Saving maximum and minimum
- Action Potential Detection
- Action Potential Save Order
- Action Potential Saved in RAM
- Mean Action Potential Shape Compute
- Correlation Compute
- Correlation Thresholds
- Correlation Pattern Detection
- 7-segment display

In the ***Saving maximum and minimum*** part, we show how the maximum and minimum values are found. The detection of Action Potentials, as well as the detection of False Positives and False Negatives is shown in ***Action Potential Detection***. The ***Action Potential Save Order*** part is dedicated to create a signal that order to the ***Action Potential Saved in RAM*** saves the samples saved in the ***register_window*** backup samples. The ***Mean Action Potential Shape Compute*** describes the compute of the Mean Action Potential, it started when the Action Potential shape detected has been saved in present spike RAM. The ***Correlation Compute*** parts show how the correlation signals is computed with the ***register_window*** samples and the Mean Action Potential shape saved in the Mean spike RAM. The ***Correlation Thresholds*** part is based on the module aforementioned in (Figure 4.14) and carry out the same task over the correlation signal for computing the adaptive threshold above the background noise level. The ***Correlation Pattern Detection*** detects the correlation patterns aforementioned in (Figure 3.17). Finally, the system results, such as the detection numbers and FIFO outputs, are displayed by the ***7-segment display*** part.

The ***module inputs***, shown in (Figure 4.15), are connected in the architecture of the ***Topmodule*** mentioned in section 4.6, where ***P_Detection*** and ***N_Detection*** outputs comes from the ***Top level Threshold*** module which are the positive and negatives thresholds values for detecting Action Potentials. The ***Band_pass*** input is the present value in the output of the band-pass filter. The visualization options ***Switch_0*** and

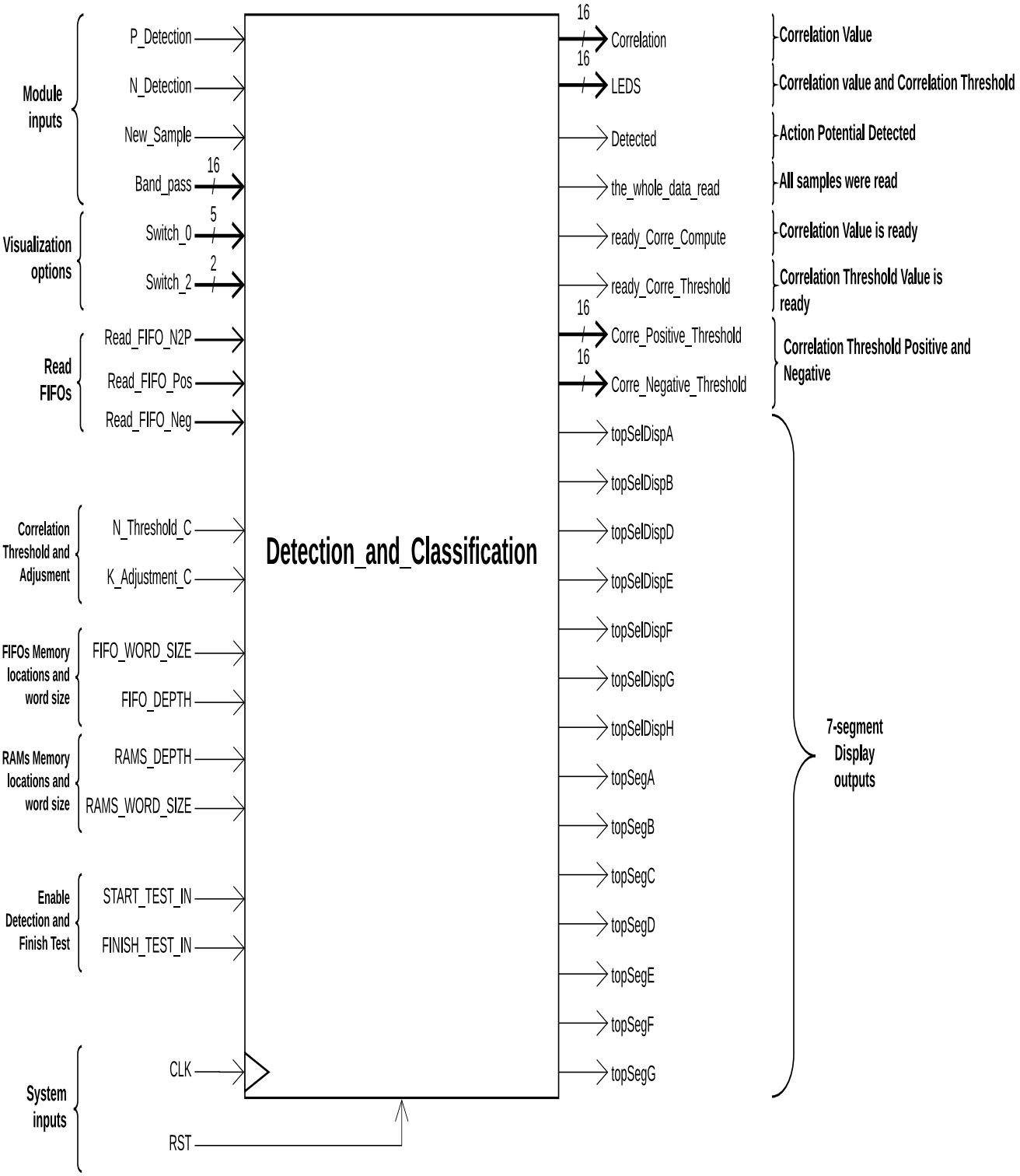


Figure 4.15: Pricipal module for saving and sorting Action Potentials

Switch_2 are input system to choose the 7-segment display number depending on the binary number enter by the switches on the FPGA and for display the Correlation signal and the Positive correlation threshold on the leds output. Correlation Threshold and Adjustment are integer values to determinate the K adjustment and N threshold level in the Correlation Threshold part. The RAM adjustment are dedicated to determinate the RAM word size and the deep locations RAM. Enable Detection and Finish Test literately determinate at which number sample the detection process stars and finish.

The output section are the ***correlation*** output which is the correlation value computed for the Mean Action Potential shape and the ***register_window*** samples. The ***LEDS*** output displays depending on the ***Switch_2*** input value, either the correlation value computed or the adaptive threshold for the correlation signal. The ***Detected*** output is set high during when an Action Potential is been detected. The ***whole_data_read*** output is set high when the sample input in FINISH_TEST_IN has been read adn reached for the SD card module. After, the ***ready_Corre_Compute*** and ***ready_Corre_Threshold*** outputs are set high when the Correlation signal value and the Correlation adaptive threshold have been computed. In addition, the Correlation Thresholds negative and positive are put on Pmod OLEDrgb display by the Corre_Positive_Threshold and Corre_Negative_Threshold outputs . Finally, the 7-segment display outputs display the FIFO, Action Potentials detected counter, classified detections counters by pattern and the present sample read by the SD card module.

4.6.7.1 Saving maximum and minimum sub-modules

These modules are dedicated to save the maximum and minim values when a Action Potential is detected and are shown in (Figure 4.16) and (Figure 4.17). The ***Top_level_Saving_Maximum*** module is triggered when a new detection is presented in the ***Positive_Detection*** module input, this input comes from the ***Top_level_Threshold*** module shown in (Figure 4.14).

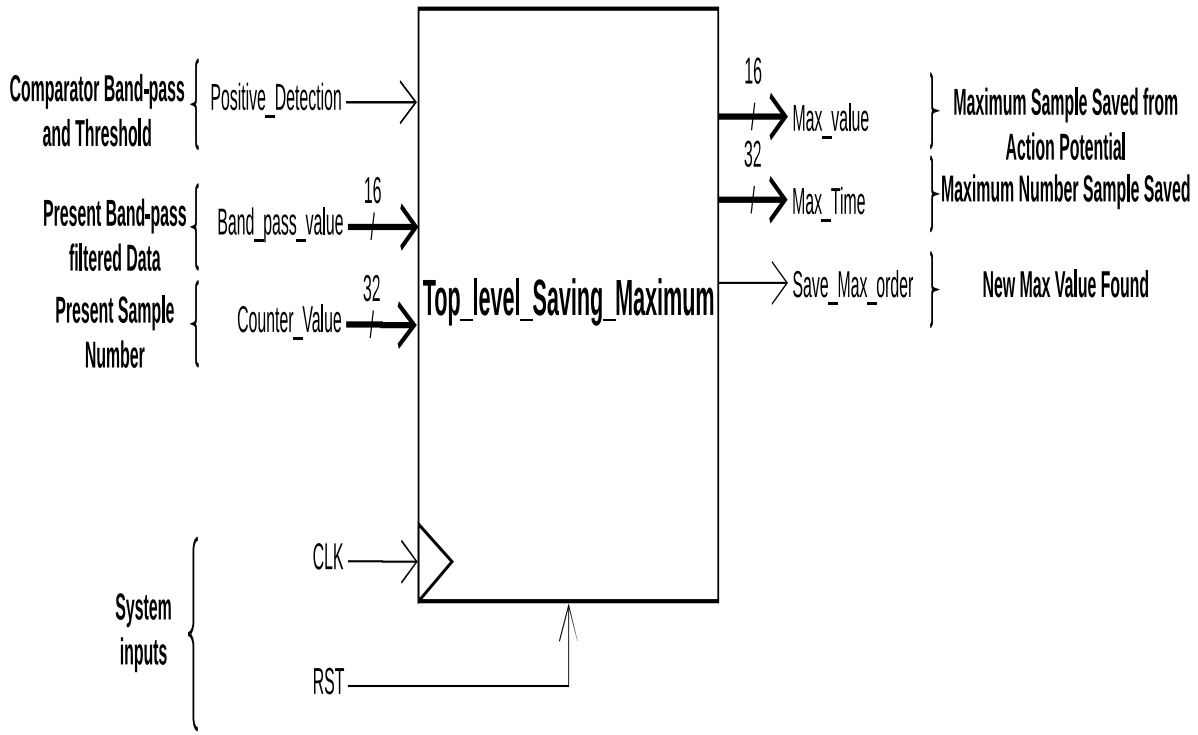


Figure 4.16: Correlation thresholds module.

The main purpose of this module (Figure 4.16) is to save the maximum positive value found and the samples number of that value when the detector stays high during an Action Potential detection. These values come from the band-pass filter output by

the ***band_pass_value*** module input, and the principal counter which gives the count of samples read from the SD card by the ***Counter_Value*** module input. Finally, each time the detector has triggered and a new maximum value is found, their sample values are set on the ***Max_value*** and ***Max_Time*** module outputs. Consequently, the ***Save_Max_order*** module output emits a high pulse during a clock cycle, for resetting a counter that will save the Action Potential shape in the ***Detection_and_Classification*** module in a RAM memory, exactly when the maximum saved value sample is at the middle location of the RAM, to further processing.

The FSM that controls the saving maximum process is shown in (Figure C.1 and the components that controls are shown in (Figure C.2). The main FSM states are explained as follows:

- ***clean0***. Here, the ***Register_Max*** register is cleaned to save the first sample that has been detected above the threshold value computed by the ***Top_level_Threshold*** module. The signal which cleans the value saved is the ***Clean_value*** signal. Consequently. the FSM passes to the next transition the ***waiting*** state.

- ***waiting***. This states is where the FSM stays until a new Positive detection is triggered by the ***Positive_Detection*** module input. Then the FSM passes to next ***clean1*** state where the register is clean by the ***Clean_value*** signal for saving the new value in ***Register_Max*** register. Afterwards, the FSM passes to the next transition the ***Searching*** state.

- ***Searching***.This states detects when the Action Potential has been above or under the present threshold value. So, if the present value in ***band_pass_value*** module input is higher than the value saved in ***Register_Max***. the ***Maximum_Point*** comparator sets high the ***Save_Max*** input, ,

- ***Saving***. This states indicates when a new maximum value has been detected by the ***Maximum_Point*** comparator, if that happens the ***Saving_Max*** output signal, which is the same signal that goes to the ***Save_Max_order*** module output, is set high saving the new maximum value in the ***Register_Max*** register. When the ***Positive_Detection*** module input returns to 0 that means that the Maximum value, as well as its sample number were saved in both the ***Register_Max*** and ***Register_time*** registers. In that way, the Action Potential has been detected and stored for further processing. Finally, when the detector returns to low, the FSM goes to the ***Waiting*** state until a new detection takes place.

The ***Top_Level_Saving_Minimum*** module for saving the minimum value during a Action Potential detection is shown in (Figure 4.17). This module follows the same procedure as the ***Top_Level_Saving_Maximum*** module. The only difference is that this module save the minimum value during a negative detection of an Action Potential detection. In addition, the ***Save_Max_order*** module output has been removed, due to we save the Action Potential shape in the memory RAM exactly when the maximum value is at he middle of the RAM locations. The FSM that controls the saving process of minimum values is shown in (Figure C.4), as well as the components that controls are show in (Figure C.3).

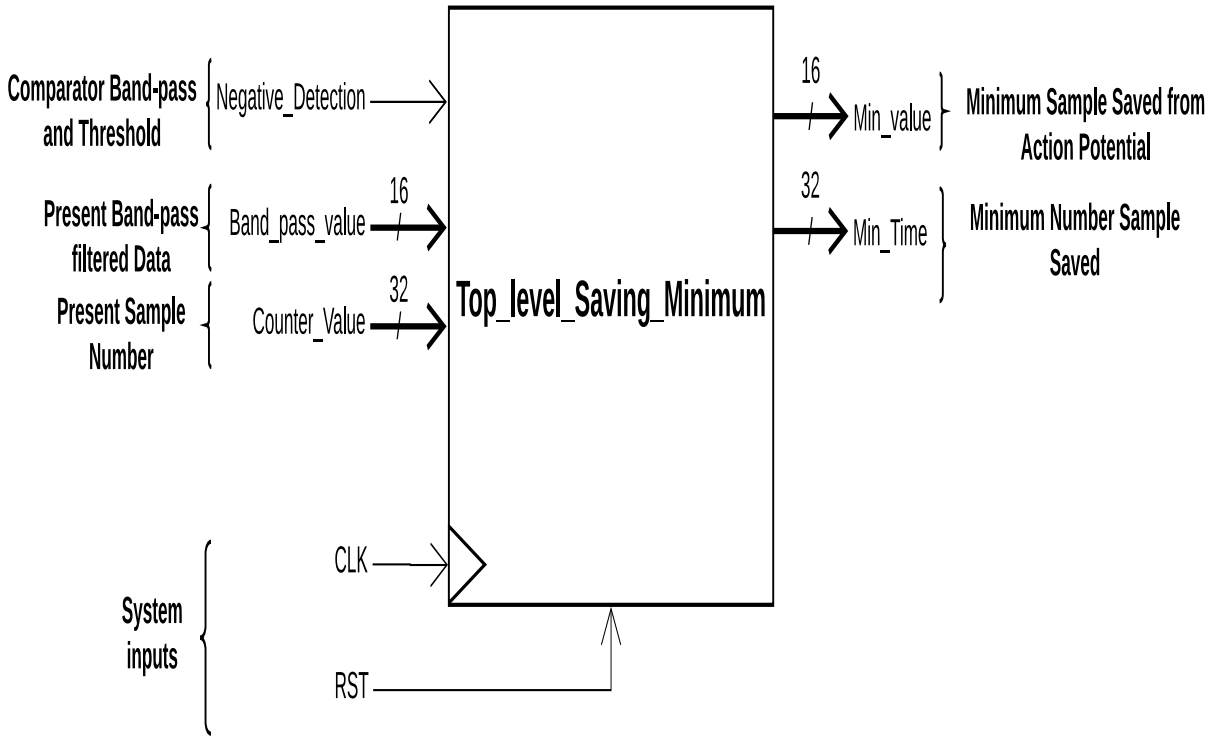


Figure 4.17: Correlation thresholds module.

4.6.7.2 Action Potential Detection

The FSM that performs Action Potential Detection of Macaque monkey Action Potentials, as well as the detection of False negative and False positives is shown in (Figure C.5) and the components that controls are shown in (Figure C.6). The transition between some states depends on the current state and the current inputs, while in others only depends on the current state. The main idea of this FSM is to save the samples number of the maximum and minimum values detected in both ***Top_Level_Saving_Maximum*** and ***Top_Level_Saving_Minimum*** modules , that is by saving the Action Potential, False positive and False negatives into their respective FIFO memory.

The description of the main state in the FSM are as follows:

- ***Waiting***. it is waiting the ***enable*** input to be set high and ***Disable_Detection*** input for staying low. As long as, the FSM stays in this state, the ***Counter Samples*** counter is resetting, this counter help us to know if a Action Potential, False positive or negative, has been detected. The ***enable*** input is set high enabling the Detection of Action Potentials, False positives and False negatives. After, depending on the current inputs the ***P_Detection*** and the***N_Detection*** inputs which are positive and negative detectors from ***Top level Threshold*** module, the transition could be either the ***Counting_Negative*** or the ***Counting_Positive*** states.

- ***Counting_Negative***. This state is reached when the ***N_detector*** input has been triggered. The ***Detected*** output during this stated is set high, that is indicating that a new detection took place in the system. The next transition depends on which signal is triggered before, the ***P_Detection*** or ***Count_Finish_Samples*** inputs. If ***P_Detection*** input is triggered before the ***Counter Samples*** counter reaches to the size of RAM memory locations when ***Count_Finish_Samples*** input is set

high, then the next transition will pass to the ***En_count***state. On the contrary, if ***Count_Finish_Samples*** output is set to high before, the next transition will be ***Write_FIFO_Neg*** state and the internal count will start in 0.

- ***En_count***. When this state is reached, an Action Potential with two phases has been detected. The FSM that manage the detection of Correlation Patterns is enabled by the ***Enable_FSM_Correlation_Pattern*** output and the FSM that determinate exactly in which sample is saved the Action Potential shape into the RAM memory is enabled by the ***Enable_FSM_Count_half*** output . Finally, the next ***N2P_spike*** state is reached in the next clock cycle.

  N2P_spike. During this state the ***Count_Enable_Samples*** output enables the count of samples of the Action Potential shape and the ***Enable_FSM_Spike*** output is set high to enable the FSM that saves the Action Potential shape in the RAM memory. Consequently, the detection is saved in the FIFO memory when the ***Write_FIFO*** state is reached which happens when the***P_Detection*** input is set to low and the ***Counter Samples*** counter has reached the full count which is indicated by setting high the ***Count_Finish_Samples*** input. Finally, ***Counter Samples*** is cleaned by the ***Clean_Count*** state and the FSM returns to the ***Waiting*** state.

- ***Counting_Positive***. This state is reached when the ***P_Detection*** input has been trigged before the ***N_Detector*** input and the ***Detected*** output is set to high during this state. Consequently, the ***Count_Finish_Samples*** signal will be triggered and the detection will be saved in the next transition the ***Write_FIFO_Pos*** state, this is into the FIFO memory locations whose store values are only positive detection without negative phase detected by the the negative detector. Finally, the counter is cleaned in the next ***Clean_Count*** state.

The FSM that performs Action Potential Detection of Human Pancreatic Action Potentials, as well as the detection of False negative and False positives is shown in (Figure C.7) and the components that controls are shown in (Figure C.8). The transition between some states depends on the current state and the current inputs, while in others only depends on the current state. The main idea of this FSM is to save the samples number of the maximum and minimum values detected in both ***Top_Level_Saving_Maximum*** and ***Top_Level_Saving_Minimum*** modules , that by saving the Action Potential, False positive and False negatives into their respective FIFO memory. This FSM is similar to the one mentioned above.

The description of the main state in the FSM are as follows:

- ***Waiting***. It is waiting the input ***enable*** to be set to high and ***Disable_Detection*** input stays low. As long as the FSM stays in this state, the ***Counter Samples*** counter is resetting, this counter help us to know if a Action Potential, False positive or negative, has been detected. The ***enable*** input when is set high enables the Detection of Action Potentials, False positives and False negatives. After, depending on the current inputs ***P_Detection*** and ***N_Detection*** which are positive and negative detectors from ***Top level Threshold*** module, the transition could be either the ***Counting_False_Negative*** and ***En_Count*** states.

- ***Counting_False_Negative***. This state is reached when the ***N_detector*** has been triggered. The ***Detected*** output during this stated is set high, that is indicating that a new detection took place in the system. The next transition depends on the ***N_Detection*** input, when this input is set low, indicates that the biosignal sample

returned between the two thresholds where any detector is triggered. Then, the next transition is the ***Write_FIFO_Neg*** state where the minimum value detected is saved in the ***FIFO_time_Neg*** memory by the ***FIFO_write_Neg*** signal. Finally, the ***Clean_Count*** state is reached and the FSM one more time starts in the ***Waiting*** state.

- ***En_count***. When this state is reached, an Action Potential with positive phase has been detected, due to the negative Action Potential phase can be confused when the noise triggers the ***N_Detection*** detector. During this state the ***Enable_FSM_Count_Half*** output enables the FSM that determinates exactly in which sample is saved the Action Potential shape into the RAM memory. Finally the FSM passes to the next transition the ***Counting_Positive*** state.

- ***Counting_Positive***. During this state the ***Counter_Samples*** counter is enabled to count the samples in the Action Potential shape. The next transition depend on if this counter reaches its final count and if a detector is triggered. The ***En_Corre*** states is reaches when ***Counter_Samples*** counter indicates that it reached its full count by the ***Count_Finish_Samples*** input. The ***Write_False_Positive*** state is reached when the ***P_Detection*** detector input is still high when the ***Counter_Samples*** counter reached the full count detecting a false positive, after the ***Clean_Count*** state is reached. The ***P2N_Spike*** state is reached when the Action Potential detected triggered both detectors before the ***Counter_Samples*** counter reaches the full count, consequently, the ***Write_FIFO_P2N*** state is reached for saving the detection in the ***FIFO_time_P2N*** memory and the ***Counter_Samples*** counter is cleaned in the ***Clean_Count*** state and the FSM one more time starts in the ***Waiting*** state.

- ***En_Corre***. When this state is reached the FSM that manages the detection of Correlation Patterns is enabled by the ***Enable_FSM_Correlation_Pattern*** output and the FSM that determinates exactly in which sample is saved the Action Potential shape into the RAM memory is enabled by the ***Enable_FSM_Count_half*** output . Finally, the next ***Write_FIFO_Pos*** state is reached in the next clock cycle where the detection is saved in ***FIFO_time_Pos*** memory. Finally ***Counter_Samples*** is cleaned in the ***Clean_Count*** state and the FSM one more time starts in the ***Waiting*** state.

4.6.7.3 Action Potential Save Order

This FSM, in (Figure C.9) and the components that controls in (Figure C.10), execute the process to indicate to the system when has to be saved the Action Potential shape detected into the RAM memory, that is by the ***register_window*** samples where the Action Potential shape fits. So, when the maximum value has been saved by the ***Top_level_Saving_Maximum*** module whose value saved is put on the.***Max_Time*** output, and after is saved into the FIFO memory. This FSM is enabled by the FSM described in (subsubsection 4.6.7.2) when both detectors positive and negative have been triggered.

The description of the main states in the FSM are as follows:

- ***Waiting***. It is waiting for the ***enable*** and ***Enable_FSM_Count_Half*** inputs that indicates that a Action Potential has been detected, as long as a counter is reset by the ***Clear_Count_Half*** output. Consequently, when the enable inputs are set high the FSM past to the next transition the***waiting_save*** state.

- ***Waiting_save***. In this state the FSM waits for the ***Save_Max_order*** input, which comes from the ***Top_level_Saving_Maximum*** module which indicates that a new maximum value was found. Then, the FSM passes to the next transition the ***Count_up*** state.

- ***Count_up***. In this state the counter is enable by the ***Enable_Count_Half*** output to start to count the samples after a Action Potential has been detected. After, the next ***is_last_count*** state is reached.

- ***Is_last_count***. This state verifies if the counter has reached to the full count by the ***Count_Finish_Half*** input for passing to the ***Save_Spike*** state .On the contrary, if the full count has not been reached the FSM passes to the ***waiting_New_Sample*** state.

- ***Waiting_New_Sample***. This state wait from the ***Save_Max_order*** input from the ***Top_level_Saving_Maximum*** module, if a new maximum value has been found the next ***Clear_Count*** state will be reached where the count will be cleaned in the counter. On the contrary, if a new sample is read from the ***SDcard_readstream*** module, the count is increased and after verifying in ***is_last_count*** state if the full count has been reached. In that way, the FSM will send the save order signal by the ***Save_Spike_Now*** output exactly when the maximum detected value is in the half of the ***register_window*** samples. When this signal is sent, the Action Potential shape will start to be saved in the RAM memory for further processing. The Action Potential shape saving process is explained in the next (subsubsection 4.6.7.4).

4.6.7.4 Action Potential Saved in RAM

This FSM, in (Figure C.11) and its components that controls in (Figure C.12), save the Action Potential detected when the maximum value from the Action Potential shape is aligned in the middle of the the ***register_window*** samples . That is when the FSM described in (subsubsection 4.6.7.3) set to high during a clock cycle the ***Save_Spike_Now*** input.

The description of the main state of this FSM are as follows:

- ***Waiting***. This state waits for the enable ***Enable_FSM_Spike*** input from the FSM described in (subsubsection 4.6.7.2) that indicated than an Action Potential has been detected and passes to next transition the ***waiting_Save_Half_32*** state.

- ***waiting_Save_Half_32***. This state is dedicated to wait the moment when the maximum value is detected, from the Action Potential shape, it is exactly in the half of the memory locations size of our RAM memory. That is when the ***Save_Spike_Now*** input is set to high during a clock cycle and the FSM passes to the next transition the ***Saving_New_Spike*** state. In that way, all the Action Potential detected will match exactly in this location and the Mean Action Potential shape could be computed properly in further processes.

- ***Saving_New_Spike***. When this state is reached, the Action Potential shape in the ***register_window*** samples is transfer to the RAM memory locations and is saved. In addition, the ***Save_Spike_sample*** and ***En_Count_Detected*** outputs are set high. When ***Save_Spike_sample*** signal is high the sample is saved in the RAM location that indicates the count in the counter was reached. As long as, the

En_Count_Detected is high the counter is enable and increasing. In that way, we save in each clock cycle a sample into the RAM memory locations until the full count is reached, this is indicated when the ***Count_Finish_spike_Detected*** input is set high and the FSM passes to next transition the ***switching*** state which saves the last sample from the Action Potential shape.

- ***Spike_Saved***. Finally, when the FSM reaches to this state, the ***Clear_Count_Detected*** and ***Start_to_Compute_Mean*** outputs are set high. When ***Clear_Count_Detected*** signal is high the count of the counter is cleaned, as long as ***Start_to_Compute_Mean*** output is high enabling the compute of the Mean Action Potential in its respective FSM. This compute is explained in the next (subsubsection 4.6.7.5).

4.6.7.5 Mean Action Potential Shape Compute

This FSM, in (Figure C.13) and its components that controls are shown in (Figure C.14). The FSM computes the Mean Action Potential shape each time a new Action Potential shape has been saved in the RAM memory.

The description of the main FSM states are as follows:

- ***Waiting***. This state waits for the enable ***Start_to_Compute_Mean*** input from the FSM described in (subsubsection 4.6.7.4)indicating that a new Action Potential shape has been saved in the RAM memory. When ***Start_to_Compute_Mean*** is high the FSM passes to the next transition the ***Compute_Mean*** state. In addition, during ***Waiting*** state, the counter that gives the write/read address locations in the RAM memories is reset.

- ***Saving_RAM***. When this state is reached, the ***En_Count_Adress*** and ***Save_Mean*** outputs are set high. The first one enables the count in ***Counter RAM Adress*** counter which gives the write/read address location in the RAM memories. The second one saves the value into the memory location in the ***RAM_Mean_Spike*** memory that has saved the Mean Action Potential shape result until the full count is reached, that is indicating when the ***Count_Finish_temp_Rams_Adress*** input is high. Then FSM passes to the next ***Mean_Spike_Done*** state when the whole Mean Action Potential shape has been saved. If this input stays low the next transition passes to the ***Waiting_count*** state until ***Counter RAM Adress*** counter reaches to the full count.

- ***Mean_Spike_Done***. This state is reached when the new Mean Action Potential shapes has been saved in each of the locations in ***RAM_Mean_Spike*** and ***RAM_Feed_Back_Spike*** memory. During this state the ***Clear_Count_Adress*** signal is set to high to clean the count in ***Counter_RAM_Adress*** counter that gives the write/read address in the RAM memories and passes to the next transition the ***Saving_Last_Spike*** state.

  Saving_Last_Spike. During this state, the last Action Potential used to compute the new Mean Action Potential passes to ***RAM_Last_Spike*** memory that saves the last Action Potential shape detected. That is by preparing the FSM for the next compute when a new detection is saved and detected. Finally, when the Action Potential shapes has passed to one RAM to the other, the FSM reaches to the ***Waiting*** state one more time.

4.6.7.6 Correlation Compute

This FSM, shown in (Figure C.15) and its controlled components are show in (Figure C.16). This FSM is dedicated to compute the correlation value between the Mean Action Potential saved in ***RAM_Mean_Spike*** described in (subsubsection 4.6.7.5) and the samples saved in the ***register_window*** which length is the same to the number of memory locations needed to save the Mean Action Potential shape. The compute for the correlation signal starts each time a new value is read from the ***Sdcard_readstream*** module. Finally, when the compute is ready the ***ready_Corre_Compute*** output is set high. The correlation compute is similar to the architecture needed in a Iterative multiplier.

The description of the main FSM states are as follows:

- ***Waiting***. This state waits for the enable ***New_sample*** input which is controlled by the FSM explained in (section 4.6). After, one clock cycle it is waited to compute the multiplication result between the first value saved in the ***register_window*** samples and the first value saved in ***RAM_Mean_Spike*** which contains the Mean Action Potential shape computed. In addition, as long as the FSM remains in this state the output ***ready_Corre_Compute*** is set high indicating that the correlation value has been computed, as well as the ***Clear_Accumulator*** output which clean the ***Accumulator register*** , and the***Clear_Count_Mux_RAM*** signal which set to 0 the internal count of the counter which control the read address in ***RAM_Mean_Spike*** memory and the sample selected for the Multiplexer which is connected to the ***Register_window*** samples. memory

- ***Save_Multl***. This state saves the Multiplication result aforementioned between the samples coming from the ***RAM_Mean_Spike*** memory and the ***Register_window*** samples by setting high the ***Save_Mult*** output and saving the result in the ***Register_Multiplication*** register . Then the FSM passes to the next transition the***Save_Accum*** state.

- ***Save_Accum***. In order to accumulate and saved each time a new value is multiplied and added up with the past ones, such as in a for loop , the ***Register_Accumulator*** register saves each result needed. This state is in charge of this operation and saved this value in the accumulator to be added with the next input values by setting high the ***Save_Accumulator*** output. Afterwards, the ***count_up*** state is reached.

- ***count_up***. This state enables the increase count of the counter that gives the address for the next value to be input in the iterative multiplier by setting high the ***En_Count_Iterative_M*** output. In this same state,it is seen if the full count has been reached for the count and if this is the case the next transition will be the ***waiting_last*** state. On the contrary, the next transition will pass to the ***waiting_count*** state where new values will been multiplier and accumulated until the full count is reached.

- ***Save_Result***. If the full count was reached the result is ready to be saved in ***Register_Correlation*** register which will holds the value to by plotted in the Pmod OLEDrgb display and for computing the correlation thresholds for the correlation signal and detecting the correlation patterns aforementioned in (Figure 3.17). Finally, the last ***Done*** state is reached sending a high signal by the ***Correlation_Done*** output and the ***Waiting*** state is reached one more time to compute a new Correlation value when this is needed.

4.6.7.7 Correlation Thresholds sub-module

This modules shown in (Figure 4.18) and it is practically similar and identical to one aforementioned in (subsection 4.6.6), only the adjustment K and adjustment N are adjusted to give a 32-bit sample for being compared with the Correlation input value. This module follows the same logic. The ***new_sample*** module input is set high when a new Correlation value has been computed, the respective thresholds are computed to carry out the Correlation Pattern detection and Classification. The compute of thresholds and detection such as positive and negative are ready when the ***Detection_ready*** module output is set to high. In addition, the FSM that controls the compute of thresholds is the same that was mentioned in (subsection 4.6.6) and components that controls this new module are shown in (Figure C.17) , only the ***Band_pass_value*** input was replaced by the ***Correlation_value*** input as is shown in the module in the (Figure B.11).

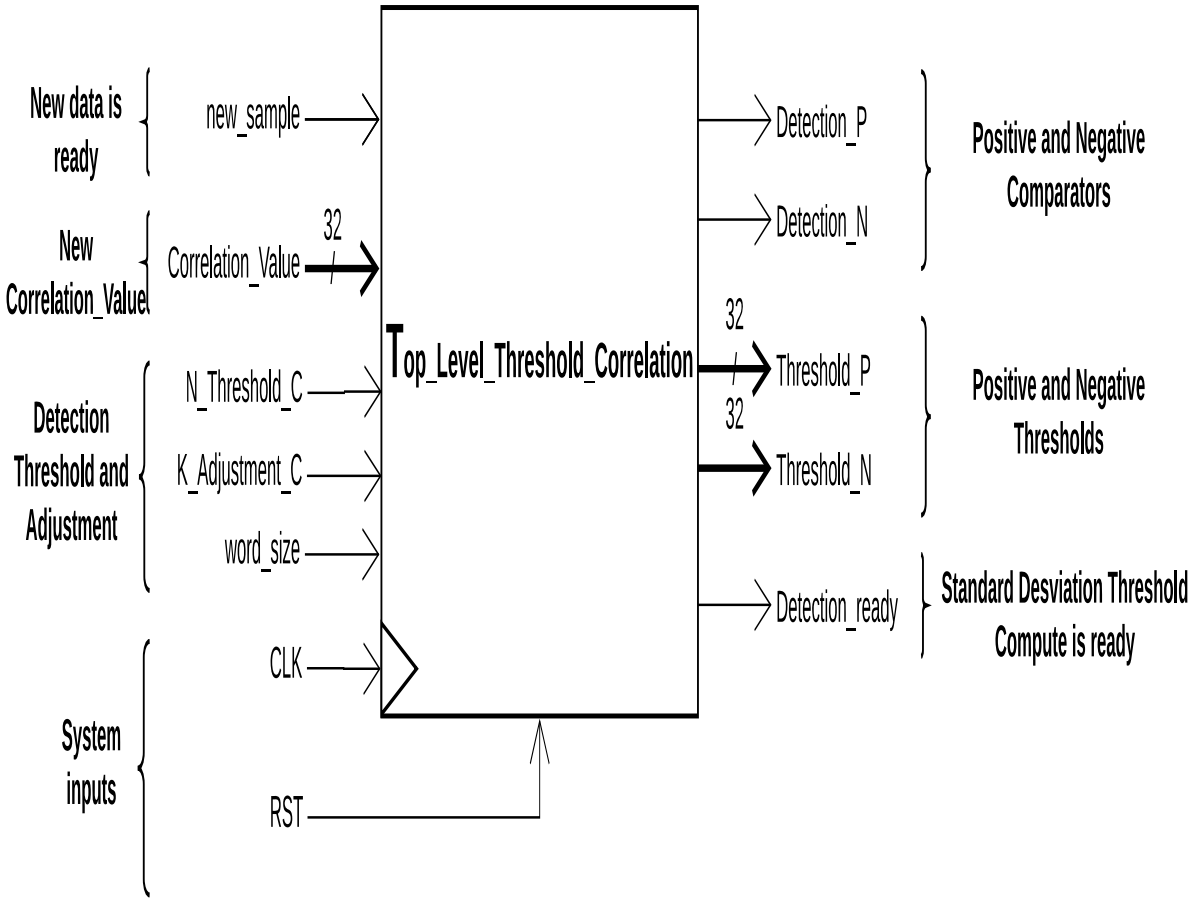


Figure 4.18: Correlation thresholds module.

4.6.7.8 Correlation Pattern Detection

This FSM, shown in (Figure C.18) and the components that controls are shown in (Figure C.19), This FSM is dedicated to detect the correlation patterns generated between the Action Potential Detected which is for a moment in the ***register_window*** and the Mean Action Potential computed. In addition this FSM manages the write/read process to save de pattern detected in the ***FIFO_pattern*** memory , as well as the counters for each of the six patterns detected. The main idea of this FSM is illustrated in (Figure 3.19).

Where the each time a Action Potential was detected a Correlation Pattern is detected depending on the phases that triggered each of the two detectors and their sequence, and after the Correlation Pattern is classified and saved in the ***FIFO_pattern*** memory. When this FSM is detecting Correlation Patterns the FSM in (subsubsection 4.6.7.2) is disable by the ***Disable*** output when is set low. In that way, as long as the detections are read in both ***FIFO_pattern*** and ***FIFO_time_N2P*** memories in the Monkey_module, and ***FIFO_pattern*** and ***FIFO_time_P2N*** in the Pancreatic_module, we will be reading the sample number where was detected the Action Potential and the Correlation Pattern detected at the same moment, due to always we will have the same number of elements in both FIFO memories.

The description of the main FSM states are as follows:

- ***Waiting***. This state waits for the ***enable*** and ***Enable_FSM_Correlation_Pattern*** inputs which are enable for the FSM described in (subsubsection 4.6.7.2) when a Action Potential has been detected. After, the FSM passes to the next transition the ***En_waiting_Detection*** state.

- ***En_waiting_Detection*** . This state waits for the trigger events from the detectors which are the ***Thre_Neg*** and ***Thre_Pos*** FSM inputs, either a positive detection or a negative detection, and in addition if the full count is reached in the ***Correlation Shape counter*** counter, after an Action Potential detection, the ***Window_passed*** signal is set high and the pattern value ***000*** will be saved in ***FIFO_pattern*** when the ***Read_Fifo_No_Class*** and ***Not_Classificated*** states are reached . Therefore, depending on these events the next transitions could be either the ***Pattern_1*** state when a negative detection occurs or ***Pattern_2*** state when a positive detection occurs and the ***Read_Fifo_No_Class*** state when not detection occurs in any of the two detectors.

- ***Pattern_1***. When this state is reached a negative detections has happened and the ***Correlation Shape counter*** is reseating by the ***Clear_Count_Samples*** output and the FSM passes to the next transition, the ***Counting_Pattern_1*** state where the count is enable by the ***En_Count_Correlation_Pattern*** output, and finally, if the count reaches to the half value count which is indicated by the ***half_Count*** FSM input a Correlation Pattern 1 has been classified and the FSM passes to the next the ***Read_Fifo_Pattern_1*** and the ***Saving_Pattern_1*** states which saves the pattern value ***001*** saved in the ***FIFO_pattern*** memory. On the contrary, if a positive detection happens before the counter reaches to the half count by the ***half_Count*** input, the FSM passes to the next transition, the ***Pattern_3*** state. In addition, the ***Pattern 1 Counter*** is enable by the ***Count_Pattern_1*** FSM output when the ***Read_Fifo_Pattern_1*** state is reached.

- ***Pattern_2***. This state is reached when a positive detection has happened, and one more time the FSM passes to the next transition the***Counting_Pattern_2*** state where the counting process is enable in the ***Correlation Shape counter***. If the counter count triggers the ***half_Count*** input before a negative detection happens the FSM passes to the next transition the ***Read_Fifo_Pattern_2*** state by producing the pattern value ***010*** and saving it such as a Correlation Pattern 2 detected in the ***FIFO_pattern*** memory when the ***Saving_Pattern_2*** state is reached. In addition, the ***Pattern 2 Counter*** is enable by the ***Count_Pattern_2*** FSM output when the ***Read_Fifo_Pattern_2*** state is reached.

- ***Pattern_3***. This state is reached when a negative detection has happened with a positive detection. After, the ***Counting_Pattern_3*** state enables the counting

process in ***Correlation Shape counter*** and if a negative detection happens before the ***half_Count*** input is high, the FSM passes to the ***Pattern_4*** state. On the contrary, a Correlation Pattern 3 is detected, and the detection is classified as the pattern value ***011*** and saved in ***FIFO_pattern*** memory by the next transitions the ***Read_Fifo_Pattern_3*** and ***Saving_Pattern_3*** states. In addition, the ***Pattern 3 Counter*** is enabled by the ***Count_Pattern_3*** FSM output when the ***Read_Fifo_Pattern_3*** is reached.

- ***Pattern_4***. This state is reached when three detections events have happened in the next order, negative positive, negative. Consequently, the detection is saved in the ***FIFO_pattern*** memory as a Correlation Pattern 4 with the the ***100*** pattern value by the next ***Read_Fifo_Pattern_4*** and ***Saving_Pattern_4*** states. In addition, the ***Pattern 4 Counter*** is enabled by the ***Count_Pattern_4*** FSM output when the ***Read_Fifo_Pattern_4*** state is reached.

- ***Pattern_5***. This state is reached when a positive detections follows a negative detection. Consequently the detection is saved in the ***FIFO_pattern*** memory as a Correlation Patterns 5 with the the ***101*** pattern value by the next ***Read_Fifo_Pattern_5*** and ***Saving_Pattern_5***states. In addition, the ***Pattern 5 Counter*** is enabled by the ***Count_Pattern_5*** FSM output when the ***Read_Fifo_Pattern_5*** is reached.

4.6.7.9 7-Segment display and switch inputs

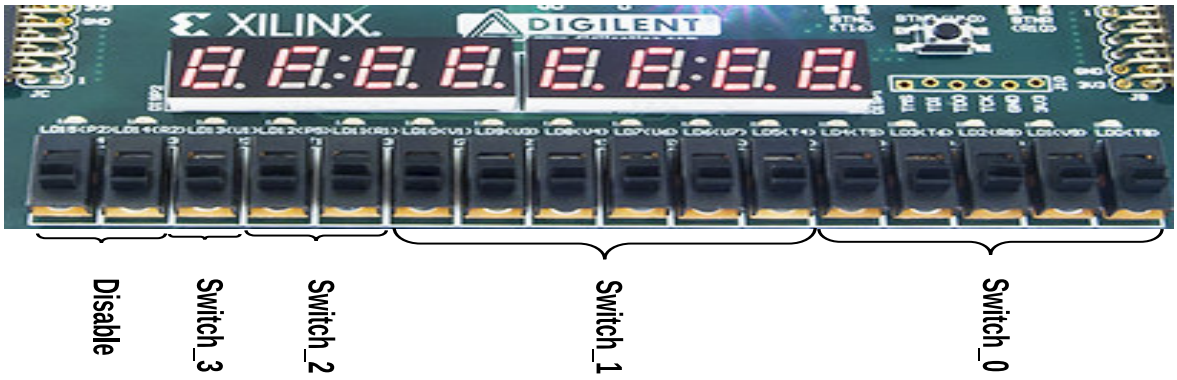


Figure 4.19: Switch Inputs

The use of the 7-segment display is only for display of FIFO outputs, Detection Counters, Pattern counters, SD outputs, Actual sample read and other outputs needed for creating the architecture. Where, depending on this value, 7-segment displays will show the output in its corresponding Hexadecimal representation. The possible outputs for the module that detects Action Potentials with the ***Switch_0*** input is presented in (Table 4.2). While, in the same way for the ***SPancreatic_module*** the possible outputs by the ***Switch_0*** input are shown in (Table 4.3).

The use of the ***Switch_1*** input is for the zoom in the Pmod OLEDrgb display for properly seeing the Correlation Patterns and Correlation Thresholds. The adjustment of the signals are presented in the table (Table 4.4), where depending on the input value, the signals displayed in the Pmod OLEDrgb display can be seen smaller or higher.

The use of the ***Switch_2*** input, is for displaying the Correlation signal value as well as the Positive Correlation Threshold on the led output. That is by changing the ***Switch_2*** input value as is shown in the (Table 4.5).

The use of the ***Switch_3*** input is for determinate which threshold will be plotted in the Pmod OLEDrgb display. That is depending on the ***Switch_3*** input value as is shown in the (Table 4.6).

The use of the ***Disable*** input is for determinate how is going to running the module and is shown in (Table 4.7). Three possible ways can be selected depending on the value in this Disable input. The First one is when the ***Disable*** input is ***00***. Then, the module will be running at the input frequency given in the respective generic input of the Toplevel module as was aforementioned in (section 4.6). The second one is when the input value is ***01***.Here, the module will be running at the frequency at which the Pmod OLEDrgb module works, that means that the Pmod OLEDrgb module will display the four signals input properly, and the module will be stopped all the processes each time the module has detected something (Action Potential, False positive, False negative). The third one is when the input value is ***10***. Here, the module will be running at the input frequency given in the respective generic input of the Toplevel module, and only when a detection is presented the module will stop the processes to properly see the Action Potential detected, False positive or False negative. In that way, we cover this three operation ways and depending on the goals the user can use whichever of theses options by only changing the Disable input value.

Table 4.2: Switch_0 input and its possible outputs displayed on the 7-Segment display in Hexadecimal for Monkey_module

Switch_0 input	Signal name	Description
00000	temp_Detection	Action Potentials detected count (which triggered both detectors)
00001	temp_Correlation_Pattern_1	Action Potentials classified in Correlation Pattern 1
00010	temp_Correlation_Pattern_2	Action Potentials classified in Correlation Pattern 2
00011	temp_Correlation_Pattern_3	Action Potentials classified in Correlation Pattern 3
00100	temp_Correlation_Pattern_4	Action Potentials classified in Correlation Pattern 4
00101	temp_Correlation_Pattern_5	Action Potentials classified in Correlation Pattern 5
00110	temp_No_Pattern	Action Potentials classified in Correlation Pattern 6
00111	temp_Positive	False Positives count
01000	temp_Negative	False Negatives count
01001	counterValue	Sample number read from SD
01010	FIFO_OUPUT	The last sample detected as Action Potentials with two phases
01011	FIFO_OUPUT_Neg	The last sample detected as False Negative

Table 4.2: Switch_0 input and its possible outputs displayed on the 7-Segment display in Hexadecimal for Monkey_module

Switch_0 input	Signal name	Description
01100	FIFO_OUPUT_Pos	The last sample detected as False Positive which triggered the positive detector
01101	FIFO_Output_Pattern	Last Correlation Pattern detected (0,1,2,3,4,5,6)
01110	Empty_Pattern , Full_Pattern Empty_Neg, Full_Neg Empty_Pos, Full_Pos Empty_P2N, Full_P2N Empty_False_Pos, Full_False_Pos	FIFO flags from each FIFO memory
01111	SD_DEBUG	SD error code
10000	SD_ERROR	It is asserted during one clock cycle if an error occurs
10001	Max_time	The last maximum sample number detected
10010	Min_Time	The last minimum sample number detected
10011	voltage_value	The last maximum value detected
10100	voltage_value_Min	The last minimum value detected
10101	Value_Last_sample	The last sample value from register_window

Table 4.3: Switch_0 input and its possible outputs displayed on the 7-Segment display in Hexadecimal for Pancreatic_module

Switch_0 input	Signal name	Description
00000	temp_Detection	Action Potentials detected count (which triggered the positive detector)
00001	temp_Correlation_Pattern_1	Action Potentials classified in Correlation Pattern 1
00010	temp_Correlation_Pattern_2	Action Potentials classified in Correlation Pattern 2
00011	temp_Correlation_Pattern_3	Action Potentials classified in Correlation Pattern 3
00100	temp_Correlation_Pattern_4	Action Potentials classified in Correlation Pattern 4
00101	temp_Correlation_Pattern_5	Action Potentials classified in Correlation Pattern 5
00110	temp_No_Pattern	Action Potentials classified in Correlation Pattern 6
00111	temp_Positive	Action Potentials detected with two phases (positive and negative one)
01000	temp_Negative	False Negatives count
01001	temp_False_Positive	False Positives count
01010	counterValue	Sample number read from SD
01011	FIFO_OUPUT	The last sample detected as Action Potentials with two phases
01100	FIFO_OUPUT_Neg	The last sample detected as False Negative

Table 4.3: Switch_0 input and its possible outputs displayed on the 7-Segment display in Hexadecimal for Pancreatic_module

Switch_0 input	Signal name	Description
01101	FIFO_OUPUT_Pos	The last sample detected as Action Potentials which triggered the positive detector
01110	FIFO_OUPUT_False_Pos	The last sample detected as False Positive
01111	FIFO_Output_Pattern	Last Correlation Pattern detected (0,1,2,3,4,5,6)
10000	Empty_Pattern , Full_Pattern Empty_Neg, Full_Neg Empty_Pos, Full_Pos Empty_P2N, Full_P2N Empty_False_Pos, Full_False_Pos	FIFO flags from each FIFO memory
10001	SD_DEBUG	SD error code
10010	SD_ERROR	It is asserted during one clock cycle if an error occurs
10011	Max_time	The last maximum sample number detected
10100	Min_Time	The last minimum sample number detected
10101	voltage_value	The last maximum value detected
10110	voltage_value_Min	The last minimum value detected
10111	Value_Last_sample	The last sample value from register_window

Table 4.4: Switch_1 input, where depending on the input value, the Correlation and Correlation Thresholds signals are displayed with a certain zoom in the Pmod OLEDrgb display

Switch_1 input	Signal name: Corre_40 Corr_Positive_threshold Corr_Negative_threshold
000000	(5 downto 0)
000001	(6 downto 1)
000010	(7 downto 2)
000011	(8 downto 3)
000100	(9 downto 4)
000101	(10 downto 5)
000110	(11 downto 6)
000111	(12 downto 7)
001000	(13 downto 8)
001001	(14 downto 9)
001010	(15 downto 10)
001011	(16 downto 11)
001100	(17 downto 12)
001101	(18 downto 13)
001110	(19 downto 14)
001111	(20 downto 15)
010000	(21 downto 16)
010001	(22 downto 17)
010010	(23 downto 18)

010011	(24 downto 19)
010100	(25 downto 20)
010101	(26 downto 21)
010110	(27 downto 22)
010111	(28 downto 23)
011000	(29 downto 24)
011001	(30 downto 25)
011010	(31 downto 26)
011011	(32 downto 27)
011100	(33 downto 28)
011101	(34 downto 29)
011110	(35 downto 30)
011111	(36 downto 31)
100000	(37 downto 32)
100001	(38 downto 33)
100010	(39 downto 34)
others	(39 downto 34)

Table 4.5: Switch_2 input, where depending on the input value, the Correlation and the Positive Correlation Thresholds signals are displayed on the led output

Switch_2 input	Signal name	Description
00	Correlation_Value(15 downto 0)	Correlation signal
01	Correlation_Value(31 downto 16)	Correalation signal
10	Correlation_Value(39 downto 32)	Correalation signal
11	Corre_threshold(15 downto 0)	Positive Correlation Threshold
others	Corre_threshold(31 downto 16)	Positive Correlation Threshold

Table 4.6: Switch_3 input values for selecting the threshold to display in the Pmod OLEDrgb display.

Switch_3 input	Signal name:	Description
0	Corr_Pos_6_bit Corr_Neg_6_bit	Correlation Thresholds are plot in Pmod OLEDrgb
1	Threshold_Pos Threshold_Neg	Detection Thresholds are plot in Pmod OLEDrgb

Table 4.7: Disable input for module operation.

Disable input	Description
00	The module is running at the input frequency (the module does not wait for the Pmod OLEDrgb display and the module does not stop in each detection)

Table 4.7: Disable input for module operation.

Disable input	Description
01	The module waits for the Pmod OLEDrgb display for properly plotting signals (the module stops in each detection)
10	The module is running at the input frequency (the module does not wait for the Pmod OLEDrgb module and the module stops in each detection)
others	The module is running at the input frequency (the module does not wait for the Pmod OLEDrgb display and the module does not stop in each detection)

4.6.7.10 Matlab scripts

The use of this Matlab scripts is for creating the Multiplexer needed for the RAM memory which saves the Action Potential detected in the memory RAM locations and for the Iterative Multiplier which is in charges of compute the correlation result, each time a new is read from the SD card. For instance, for the Macaque monkey signal we need 32 samples to properly save its Action Potential shapes. By the way, for the Human pancreatic signal we needed 128 samples to properly save its Action Potential shapes. This is different depending on the sample frequency of each signal and also the period of their Action Potentials.

The Matlab scripts are called ***Mux_To_Multiplicator.m*** and ***Mux_To_Ram.m*** both scripts generate the VHDL code for the multiplexers that needs to be replaced at the end of the code of the ***Detection_and_Classification*** module. The first one is the multiplexer that connects the ***register_window*** backup registers with the multiplier needed to compute the Correlation signal values. The second script connects the ***register_window*** backup registers with ***RAM_Present_Spike*** data input. This two scripts are shown in (section D.1)and (section D.2).

– The size of your success is measured by the strength of your desire; the size of your dream; and how you handle disappointment along the way –

– Robert Kiyosaky –

5 Results and Conclusions

5.1 Simulation Macaque monkey results

The Signal-to-noise ratio (SNR) is defined such as in Yang and Mason (2017) and Ludwig et al. (2006) as follows:

$$SNR = \frac{peak\ to\ peak\ mean\ Action\ Potentials\ Amplitud}{2\sigma\ of\ noise} \tag{5.1}$$

Here the mean Action Potentials amplitude is computed by all the peak to peak amplitude at each detection. The σ of noise is computed by the biological signal taking into account that the Action Potentials are not frequent, due to they only appear in brief periods. Therefore, by following the (Equation 5.1) the SNR for the filtered Macaque monkey biosignal is 14.42 and by applying the (Equation 5.2) the SNR is equal to 11.6 dB. The amplitude of Action Potentials detected in the Monkey macaque biosignal can be appreciated in (Figure 5.1) where is represented the Mean Action Potential shape as well as the maximum a minimum values in each of the 32 samples that makes the Action Potential shape. Here, we can see the noise level and amplitude of Action Potentials detected.

$$SNR = 10\ log_{10}\left(\frac{peak\ to\ peak\ mean\ Action\ Potentials\ Amplitud}{2\sigma\ of\ noise}\right) \tag{5.2}$$

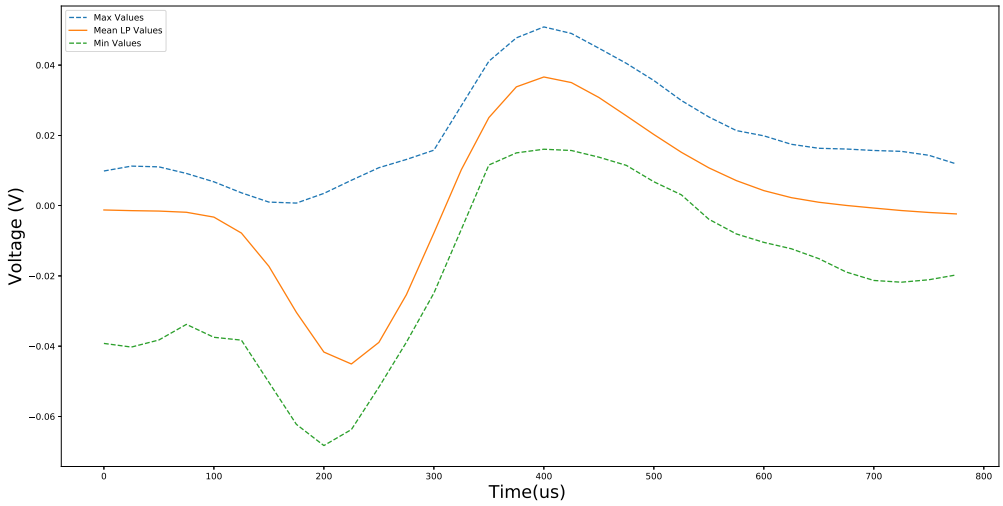


Figure 5.1: Mean Action Potential shape from Macaque monkey biosignal computed after 235 seconds and 960 detections using a threshold value of 15

The simulation was made on Python 2.7, It is important to mention that during these simulation the signal was not normalized. Therefore the parameters used here and the ones used in the Hardware architecture can be different. The parameters for the Macaque monkey biosignal are presented in (Table 5.1).

Table 5.1: Simulation parameters for Macaque monkey biosignal

Parameter Variable	Values	Description
N	5, 6, 7, 9, 11, 13, 15	Detection threshold level value
K	0.01	Internal K value into detection threshold compute
C_N	10	Correlation threshold level value
C_K	0.001	Internal K value into correlation threshold compute
Coficient_XN	0.999	Internal coefficient of low-pass filter from threshold value compute
Coficient_Yn_1	0.001	Internal coefficient of low-pass filter from threshold value compute
C_Coficient_XN	0.999	Internal coefficient of low-pass filter from correlation threshold value compute
C_Coficient_Yn_1	0.001	Internal coefficient of low-pass filter from correlation threshold value compute
SamplesNUM	32	Samples number for detecting the Action Potential

The results of each simulation using a different Threshold value is shown in (Table 5.2). Here, the Action Potentials that triggered both detector are shown in the two-phases columns, as well as the False positives and False negatives detected in their respective columns. For False Negatives or Falses positives are those detections that did not have a second phases detection between the Action Potential period.

Table 5.2: Two-phases Action Potential, False negatives, False positives detected using different threshold detection levels

Simulation Macaque monkey biosignal detections			
Threshold value	Two-phases	False negaives	False Positives
5	4696	19949	12232
7	2053	7707	2716
9	1291	4271	552
11	1064	2342	134
13	993	1155	40
15	960	542	15

The (Table 5.3) shown where each two-phases Action Potential were classified or sorted out. This is depending on the Correlation Pattern detected after its detection. The possible patterns are six, as it is shown in the table. Finally, the Action Potentials detected as their classification can be seen in (Appendix E), where the illustration are sorted by the threshold value used during the simulation.

Table 5.3: Two-phases Action Potentials sort them out by Correlation Pattern detected

Simulation Macaque monkey biosignal using 10							
Threshold value	Pattern 0	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Total detections
5	1604	21	1401	673	790	147	4696
7	30	2	437	604	912	56	2053
9	0	0	56	18	958	18	1289
11	0	0	7	87	960	10	1064
13	0	0	1	30	954	8	993
15	0	0	0	8	942	10	960

5.2 Simulation Human pancreatic results

The SNR presented for the filtered Human pancreatic biosignal by using the (Equation 5.1) is 3.89 and by applying the (Equation 5.2) the SNR is equal to 5.89 dB . The amplitude of Action Potentials detected in the Human pancreatic biosignal can be appreciated in (Figure 5.2) where is represented the Mean Action Potential shape as well as the maximum a minimum values in each of the 128 samples that makes the Action Potential shape. Here, we can see the noise level and amplitude of Action Potentials detected.

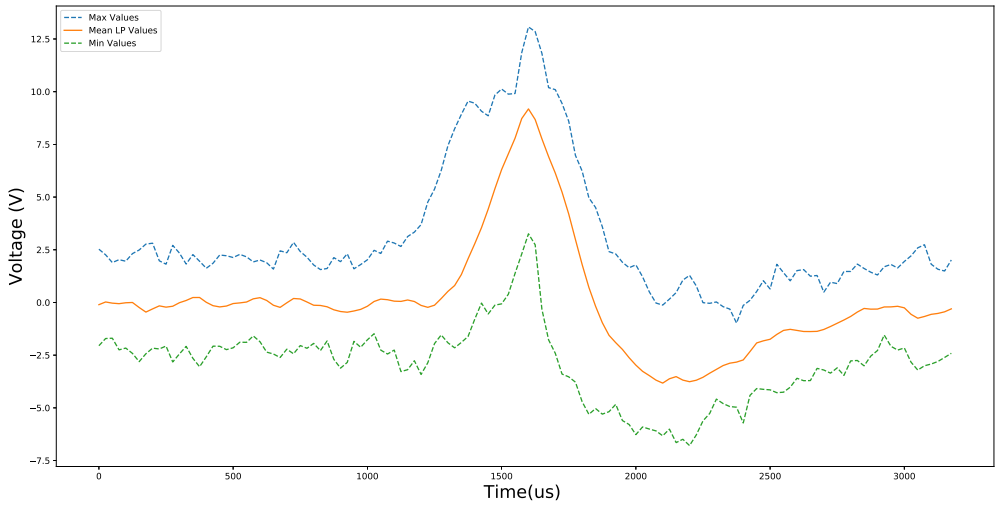


Figure 5.2: Mean Action Potential shape from Human pancreatic signal computed after 13 seconds and 53 detections using a threshold value of 15

The parameters for the Human pancreatic biosignal are presented in (Table 5.4). as aforementioned the parameters used here and the ones used in the Hardware architecture can be different. In addition, the file of this biosignal had been normalized before. So, the amplitude values not correspond to the original voltage value.

Table 5.4: Simulation parameters for Human pancreatic biosignal

Parameter Variable	Values	Description
N	9 , 11 , 13, 15	Detection threshold level value
K	1.0	Internal K value into detection threshold compute
C_N	10	Correlation threshold level value
C_K	100.0	Internal K value into correlation threshold compute
Coficient_XN	0.999	Internal coefficient of low-pass filter from threshold value compute
Coficient_Yn_1	0.001	Internal coefficient of low-pass filter from threshold value compute
C_Coficient_XN	0.999	Internal coefficient of low-pass filter from correlation threshold value compute
C_Coficient_Yn_1	0.001	Internal coefficient of low-pass filter from correlation threshold value compute
SamplesNUM	128	Samples number for detecting the Action Potential

The results of each simulation using a different Threshold value is shown in (Table 5.5). Here, the Action Potentials that triggered both detector are shown in the two-phases columns, as well as the False positives and False negatives detected in their respective columns and in addition we added an extra detection for those detections that only had a positive trigger event between an Action Potential period. For False Negatives are those detections that did not have a second phases detection between the Action Potential period. For False positives are those that stayed above the threshold for more than an

Action Potential period.

Table 5.5: Two-phases Action Potential,Positive Action Potential, False negatives, False positives detected using different threshold detection levels

Threshold value	Two-phases	Positives	False positives	False negatives
9	261	45	0	58
11	114	111	0	109
13	63	62	0	61
15	53	17	0	19

The (Table 5.6) shown where each two-phases Action Potential were classified or sorted out. This is depending on the Correlation Pattern detected after its detection. The possible patterns are six, as it is shown in the table. Finally, the Action Potentials detected as their classification can be seen in (Appendix F), where the illustration are sorted by the threshold value used during the simulation.

Table 5.6: Two-phases Action Potentials sort them out by Correlation Pattern detected

Simulation Human pancreatic biosignal using 10							
Threshold value	Pattern 0	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Total detections
9	213	1	29	7	3	8	261
11	63	2	8	6	24	11	114
13	8	0	2	5	46	2	63
15	1	0	1	3	44	4	53

5.3 FPGA Macaque monkey results

The hardware architecture for the Monkey_module dedicated to Macaque monkey biosignal was designed using the ISE WebPACK of XILINX on a NEXYS 4 Artix-7 FPGA. The module uses 2,083 slice registers (1% utilization of available slice registers), 2,915 LUTs (4% utilization of available slice LUTs), and can operate at a maximum frequency of 104.965 MHz. The next (Figure 5.3) shows when a Action Potential is detected in the FPGA showing the band-pass filter signal, the Correlation signal and the threshold for detecting Action Potentials or the Correlation Patterns generated.

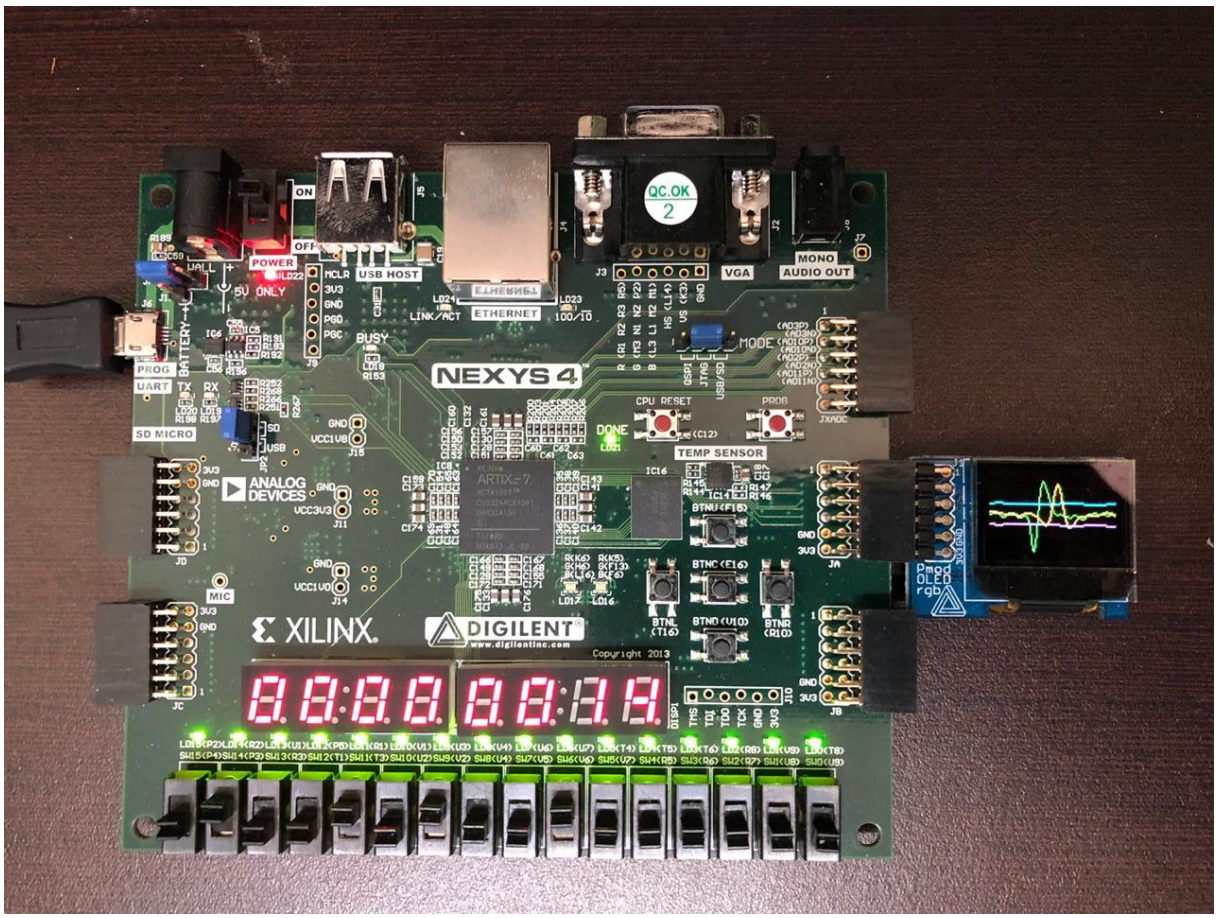


Figure 5.3: Monkey_module detecting a two-phases Action Potential (green), Correlation Pattern (yellow) and positive and negative threshold values. 7-segment displays showing the actual number of two-phases Action Potentials detected

In this module the Generic inputs for properly running the detection and classification process are shown in (Table 5.7). The ***K_Adjustment*** and ***K_Adjustment_C*** input values were tested in the FPGA architecture showing a good performance when computing the detection threshold and correlation threshold being less sensitive to the presence of Macaque monkey Action Potentials and Correlation Patterns. This thresholds were put above the background noise on the biosignal and on the Correlation signal computed internally. This was proved by using the Pmod OLEDrgb display as it is shown in the (Figure 5.3).

Table 5.7: Generic parameters used in Monkey_module

Generic input	Input value
Input_Frequency	40000
N_Threshold	[0,..,15]
K_Adjustment	1000
N_Threshold_C	10
K_Adjustment_C	100
FIFO_DEPTH	4
FIFO_WORD_SIZE	32
RAMS_DEPTH	5
RAMS_WORD_SIZE	16
START_TEST_IN	2500
FINISH_TEST_IN	9411000

The detections result by using this Monkey_module are shown in (Table 5.8). The results of each Threshold value is shown in this table. Here, the Action Potentials that triggered both detector are shown in the two-phases columns, as well as the False positives and False negatives detected in their respective columns. For False Negatives or Falses positives are those detections that did not have a second phases detection between the Action Potential period. These values were extracted from the FPGA by using the 7-segment displays as was aforementioned in (subsubsection 4.6.7.9).

Table 5.8: FPGA results. Two-phases Action Potentials, False positives, False Negatives for Macaque monkey biosignal

Threshold value	Two-phases	False positives	False negatives
5	2055	5470	15071
7	1228	931	5902
9	1037	225	2817
11	979	68	1402
13	952	19	689
15	921	10	395

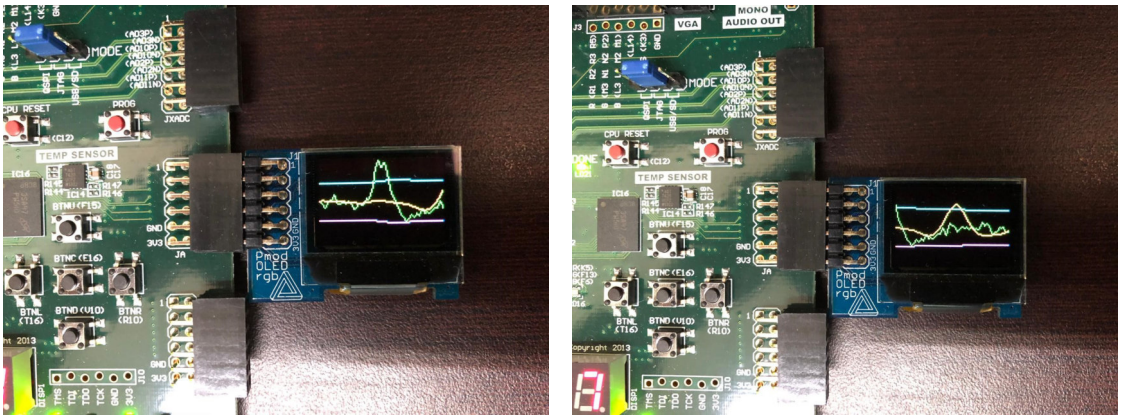
The (Table 5.9) shows where each two-phases Action Potential were classified or sorted out. This is depending on the Correlation Pattern detected after its detection. The possible patterns are six, as it is shown in the table. All the Correlation patterns were detected using a Correlation threshold value of 10.

Table 5.9: FPGA results. Two-phases Action Potentials classified by Correlation Pattern for Macaque monkey biosignal

FPGA Macaque monkey biosignal using 10							
Threshold value	Pattern 0	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Total detections
5	680	3	331	178	766	97	2055
7	99	2	94	121	868	44	2053
9	30	0	24	53	912	18	1037
11	9	0	8	26	923	13	979
13	7	0	0	18	916	11	952
15	5	0	0	12	896	8	921

5.4 FPGA Human pancreatic results

The hardware architecture for the Pancreatic_module dedicated to the Human pancreatic biosignal was designed using the ISE WebPACK of XILINX on a NEXYS 4 Artix-7 FPGA. The module uses 3,645 slice registers (4% utilization of available slice registers), 3,985 LUTs (7% utilization of available slice LUTs), and can operate at a maximum frequency of 100.654 MHz. The next (Figure 5.4) shows when a Action Potential is detected in the FPGA showing the band-pass filter signal, the Correlation signal and the threshold for detecting Action Potentials or the Correlation Patterns generated.



(a) Pmod OLEDrgb display showing a Human pancreatic Action Potential detected by its thresholds

(b) Pmod OLEDrgb display showing a Correlation Pattern generated after detected a Human pancreatic Action Potential detected

Figure 5.4: Pancreatic_module detecting a two-phases Action Potential (green), Correlation Pattern (yellow) and positive and negative threshold values

In this module the Generic inputs for properly running the detection and classification process are shown in (Table 5.10). The ***K_Adjustment*** and ***K_Adjustment_C*** input values were tested in the FPGA architecture showing a good performance when computing the detection threshold and correlation threshold being less sensitive to the presence of Human pancreatic Action Potentials and Correlation Patterns. By the way this values are the same that in the Monkey_module. These thresholds were put above the background noise on the biosignal and on the Correlation signal computed internally. This was proved by using the Pmod OLEDrgb display as it is shown in the (Figure 5.4).

Table 5.10: Generic parameters used in Pancreatic_module

Generic input	Input value
Input_Frequency	10000
N_Threshold	[0,..,15]
K_Adjustment	1000
N_Threshold_C	10
K_Adjustment_C	100
FIFO_DEPTH	4
FIFO_WORD_SIZE	32
RAMS_DEPTH	7
RAMS_WORD_SIZE	16
START_TEST_IN	750
FINISH_TEST_IN	130000

The detections result by using this Pancreatic_module are shown in (Table 5.11). The results of each Threshold value is shown in this table. Here, the Action Potentials that triggered both detector are shown in the two-phases columns, as well as the False positives and False negatives detected in their respective columns and in addition we added an extra detection for those detections that only had a positive trigger event between an Action Potential period. For False Negatives are those detections that did not have a second phases detection between the Action Potential period. For False positives are those that stayed above the threshold for more than an Action Potential

period. These values were extracted from the FPGA by using the 7-segment displays as was aforementioned in (subsubsection 4.6.7.9).

Table 5.11: FPGA results. Two-phases Action Potentials, False positives, False Negatives for Human pancreatic biosignal

FPGA Human pancreatic detections using 10				
Threshold value	Two-phases	Positive	False positive	False negative
2	415	128	2	518
3	92	192	1	202
4	48	30	0	17
5	24	35	0	2
6	5	51	0	0

The (Table 5.12) shows where each two-phases Action Potential were classified or sorted out. This is depending on the Correlation Pattern detected after its detection. The possible patterns are six, as it is shown in the table. All the Correlation patterns were detected using a Correlation threshold value of 10.

Table 5.12: FPGA results. Two-phases Action Potentials classified by Correlation Pattern for Human pancreatic biosignal

FPGA Human pancreatic biosignal using 10 two-phases							
Threshold value	Pattern 0	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Total detections
2	394	1	13	0	2	5	415
3	47	1	22	9	4	9	92
4	0	0	5	6	33	4	48
5	0	0	1	4	19	0	24
6	0	0	1	0	3	1	5

5.5 Conclusions and future works

In this work we create two modules one dedicated to detection of Action Potentials in Macaque monkey signals and other one dedicated to Action Potentials from Human pancreatic biosignals. Both modules save the sample number where each two-phases Action Potential was detected, which is the maximum value found in the Action Potential shape. This was performed by the adaptive threshold presented in Harrison (2003) where follows the hypothesis that the background noise will overtake its standard deviation value with 15% of the samples. This adaptive threshold presented good performance when good coefficients are selected, whose values were presented in (Table 5.1) and (Table 5.4) for software, and (Table 5.7) and (Table 5.10) for hardware. The use of FIFO memories in the modules was thinking about the connection to other devices that probably works at a lower clock frequency, with the use of FIFO memories we guaranty that the detection data is not loosed.

The advantages of the use of Correlation Patterns is that we do not need knowledge of the Action potential shape in biosignals, as other methods needs. The Mean Action Potential shape is computed autonomously depending on the present detections through

the time, The only data needed is to proportioned the samples length need for saving the Action potential shape depending on the sample frequency in the biosignal and adapting the module by the Matlab scripts and generic inputs. Communly, the duration of a Action Potential is around 1 ms and 10 ms depending on the nature of the biosignal, in our case the Action potentials from Macaque monkey biosignal have a period around 0.8 ms and those for the Human pancreatic biosignal have a period around 12 ms.

One of the disadvantages at the beginning is that we do not have enough detections to compute a good Mean Action potential shape, and this can cause problems sorting Action Potentials of high amplitude, due to probably the Mean Action Potential shape is not yet so similar to the Action Potential shape detected. But we think is a good cost to pay for the classification of the next detections when the Mean Action Potential shape has been computed with more Action Potentials shaped detected. The compute by using a low-pass filter also is a good tool, due to we do not need to save a lot of Action Potential shapes and we do need to wait until we get a good amount of Action Potential shapes.

In addition, we would like to mentioned some important future works. The first one is the use of one more adaptive threshold which will be put in a higher N value, and it will manage the compute of the Mean Action Potential shape. That means that we will have a threshold for the detection part and other one for the classification part. This second threshold will help the classification process creating a Mean Action Potential shape only with the Action Potentials with a High amplitude, which will be less perturbed by Action Potentials with low amplitude, making it better for generate the Correlation patterns explained in the (Figure 3.17). In that way, the Action Potentials with lower amplitude will be separated more efficiently, as it was shown for instance in the (E.6a), where almost all the Action potentials with high amplitude where sorted in the Correlation Pattern 4, and only a few in the Correlation Pattern 5. Also this behaviour was presented in the Human pancreatic results in (F.4a. In conclusion the use of a low N value threshold will detected both Action Potentials with high amplitude and low amplitude, as it was shown in (E.1a) where Action Potentials with low amplitude were sorted in several correlation patterns and by adding a second high N value threshold for only the classification part, we will help the classification process sorting the Action Potentials with high amplitude to the Action Potentials with low amplitude more efficiently.

The second important test of this module is when we will be able to implement the behaviour of the VHDL module during a real-time acquisition. Although, we did not have the resources to do it, we presented a good tool to off-line processing emulating a real-time acquisition by the sd card module that gives the raw data samples of the biosignal at its sample frequency recorded, with the advantage that the user can see the Action Potential detected through the Pmod OLEDrgb display, as well as the threshold behaviour to know if the detection process will carry out a good result.

A

Python script

```
filename = "G20151110A-01.txt" ##---------- Raw-data file location
signal1 = []
signal2 = []
print 'reading signal ...'
with open(filename) as f:
    for numstring in f:
        signal1.append(eval(numstring))
        #if len(signal)>10000:
        #    break
print ' signal loaded'
##---------------------------------------- Finding minimum value
auxmin=999999999
for i in range(0, len(signal1)):

    if signal1[i] < auxmin and signal1[i] > 0:
        auxmin = signal1[i]

print ("Min found",auxmin)
##---------------------------------------- Signal - minimum value founded
signal1[:] = [x - auxmin for x in signal1]

##---------------------------------------- Finding minimum value2
auxmin2=999999999
for i in range(0, len(signal1)):

    if signal1[i] < auxmin2 and signal1[i] > 0:
        auxmin2 = signal1[i]
##---------------------------------------- Signal / minimum value founded
signal1[:] = [x / auxmin2 for x in signal1]

print ("Min found2",auxmin2)

##---------------------------------------- 2's Complement for 16-bit format
for x in range (0,len(signal1)):
    if signal1[x]<0:
        signal2.append(signal1[x]+65536)
    else:
        signal2.append(signal1[x])
##---------------------------------------- Creating Bin file
```

```
40 doc = open("My_Monkey_16.bin", "wb")
41
42
43 for x in range(0, len(signal2)):
44
45     doc.write((chr(int(signal2[x] / 256))))
46     doc.write((chr(int(signal2[x] % 256))))
47
48 doc.close()
49
50 print("Done")
```

B

Hardware architecture

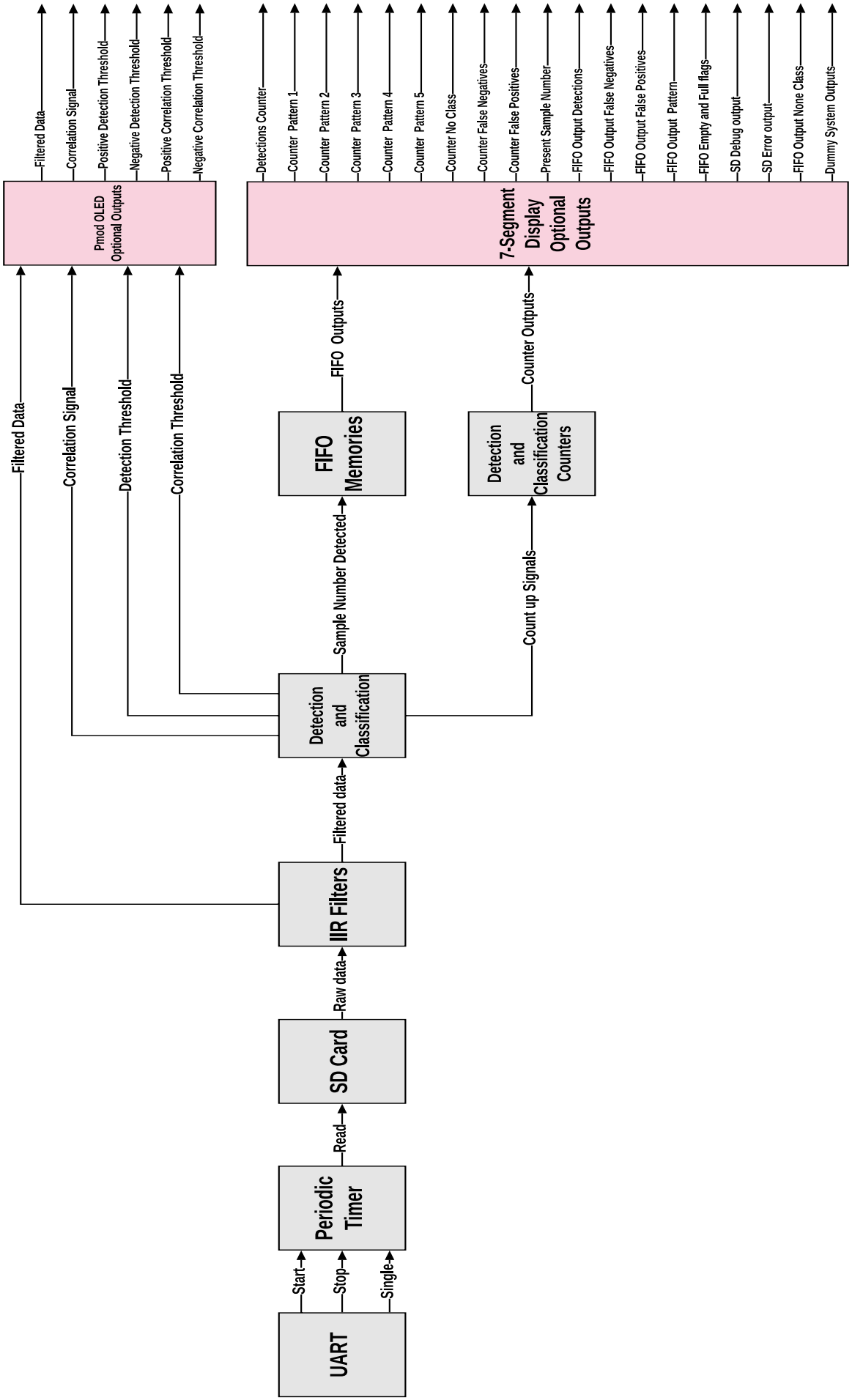


Figure B.1: General Hardware architecture module diagram. For return click here (section 4.1)

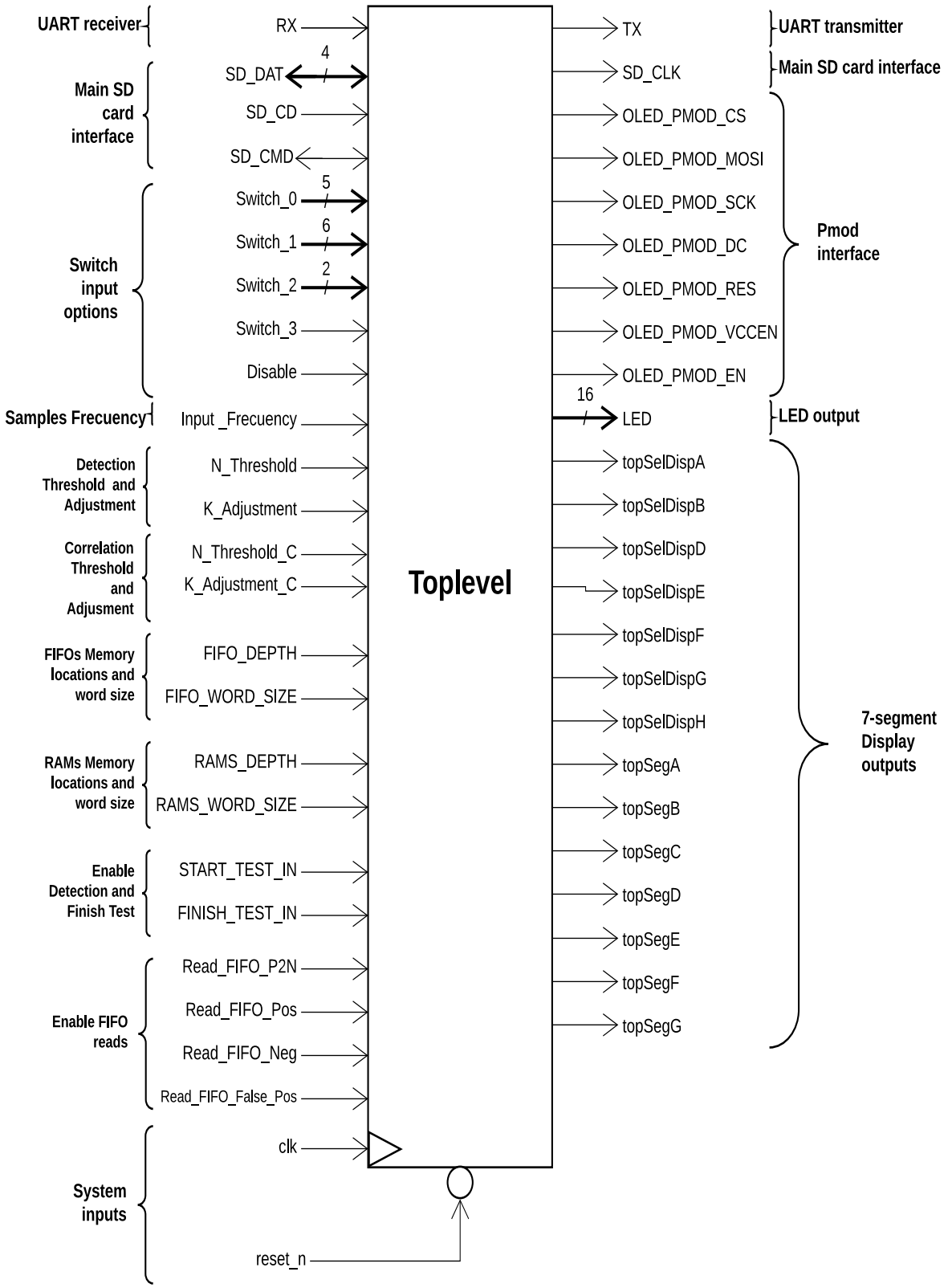


Figure B.2: Toplevel module (Pancreatic_module).

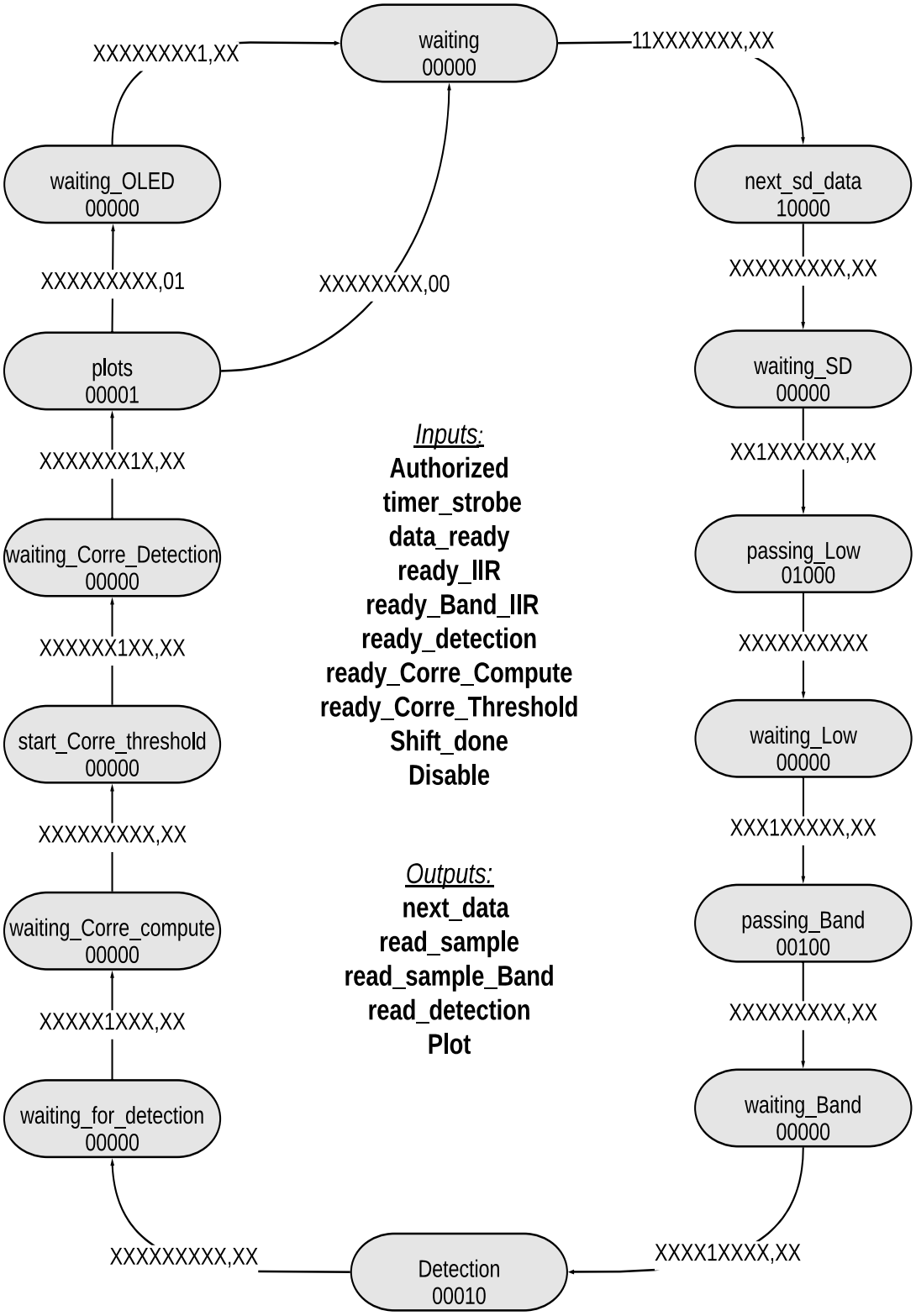


Figure B.3: Global FSM that controls all the process in the architecture. return click here subsection 4.6.1

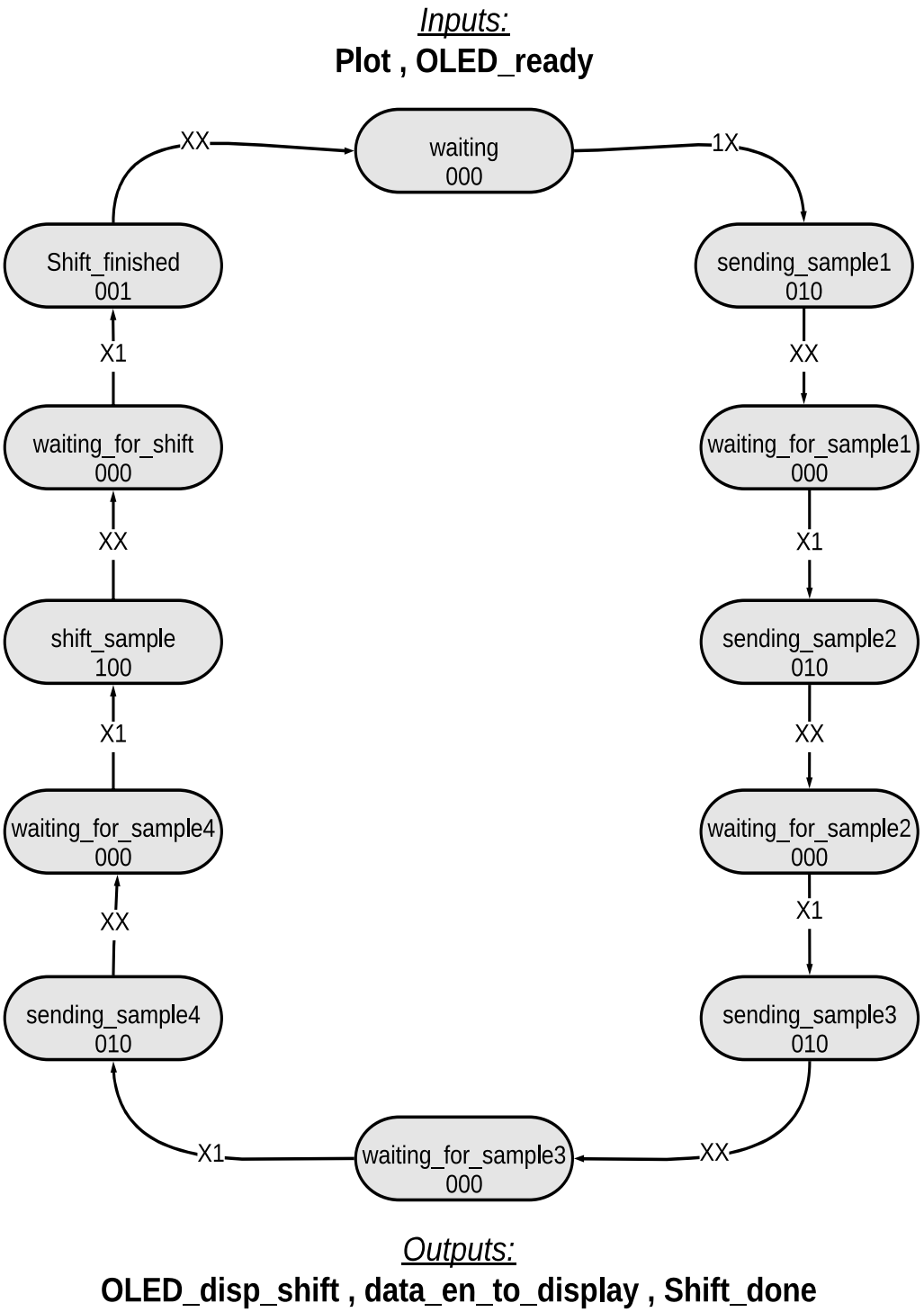


Figure B.4: FSM for plotting the signals over the Pmod OLEDrgb display. return click here subsection 4.6.4

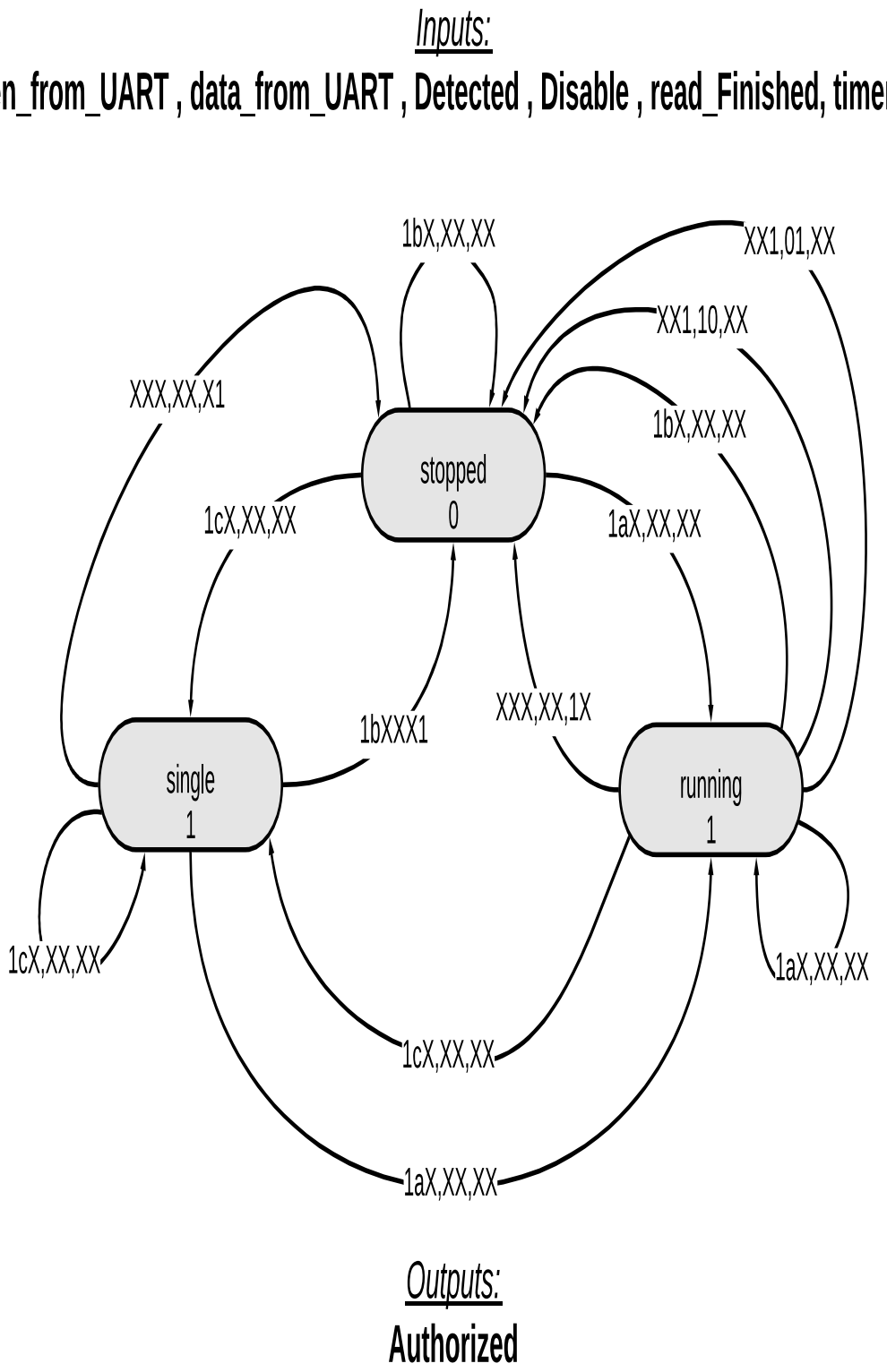


Figure B.5: UART FSM for receiving, sending and controlling the system process, such as star, stop and pause by the characters a,b,c.For return click here subsection 4.6.1

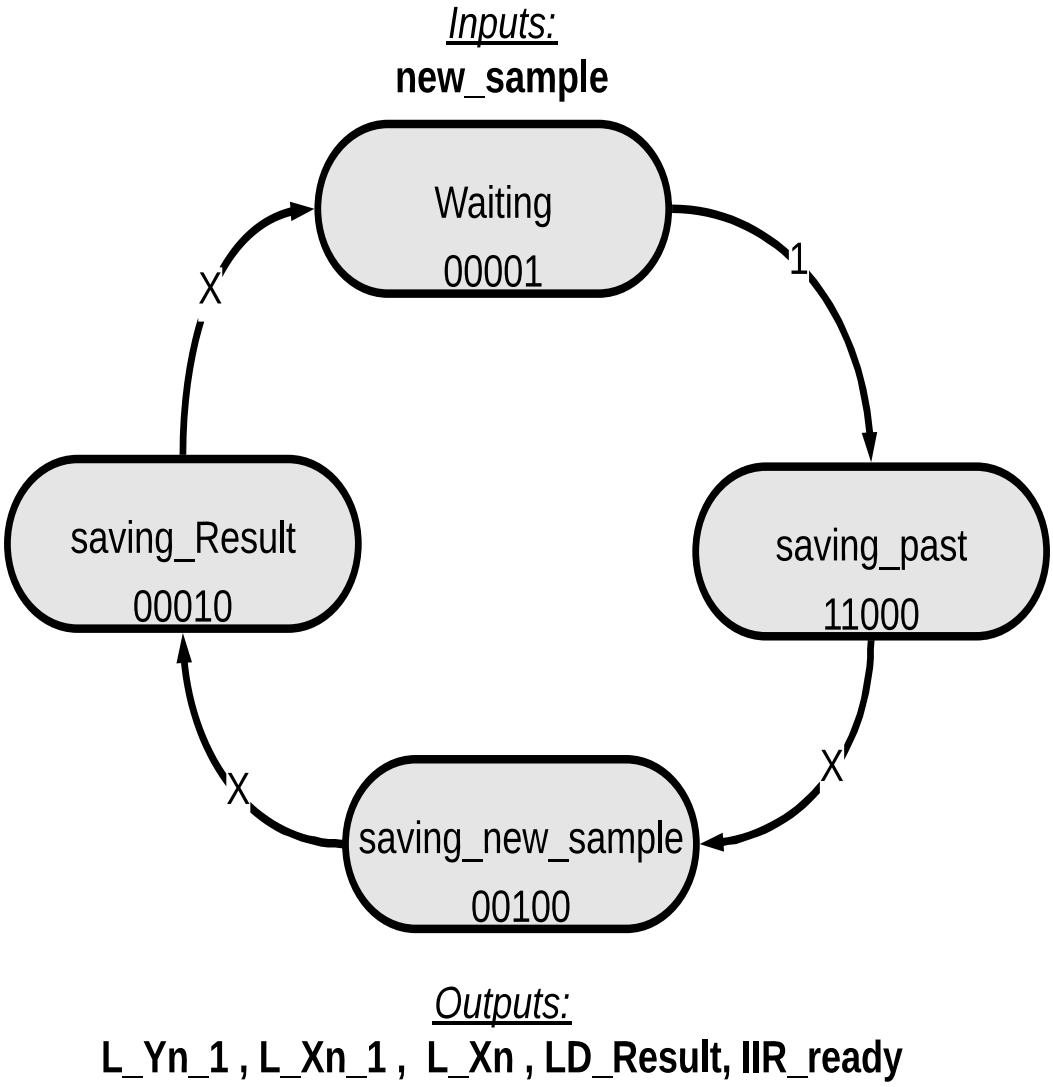


Figure B.6: FSM to compute the IIR low-pass filter output. For return click here subsection 4.6.5

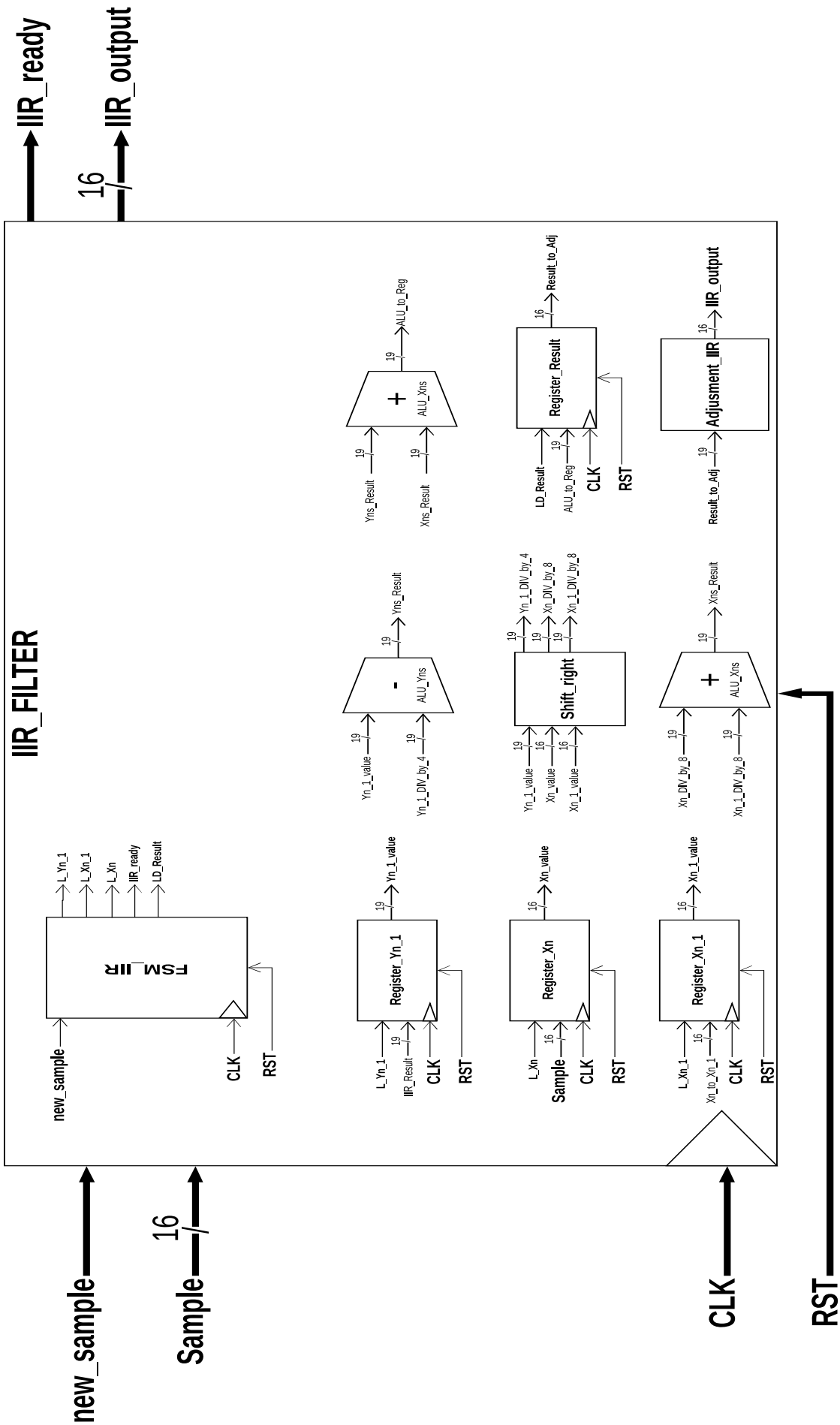


Figure B.7: IIR low-pass filter architecture components. To return click here subsection 4.6.5

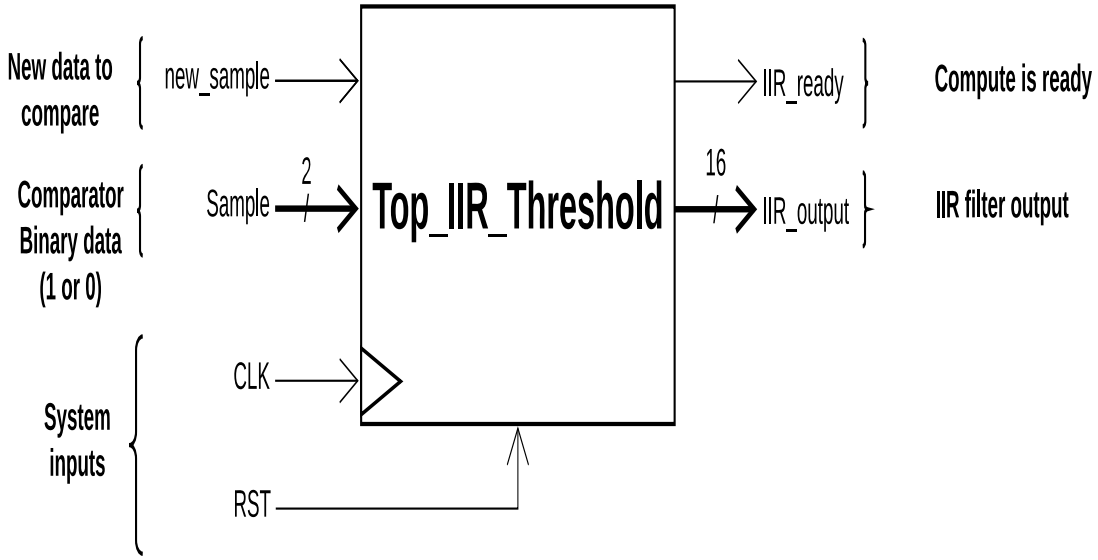


Figure B.8: IIR low-pass filter threshold.

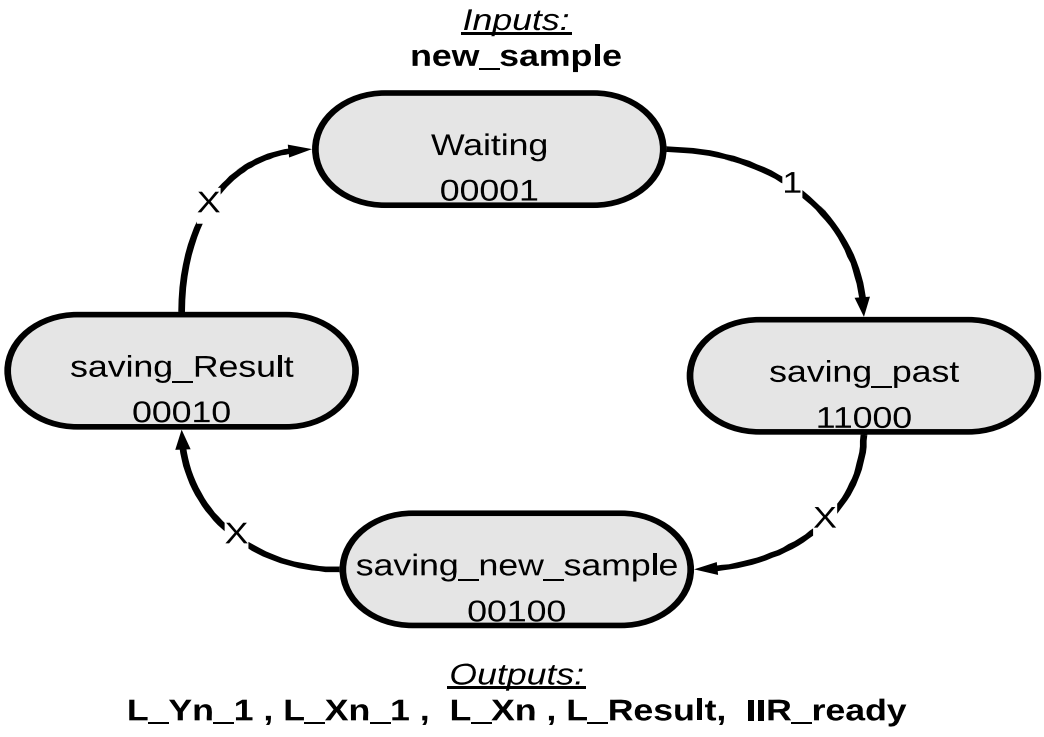


Figure B.9: IIR low-pass filter threshold. To return click here subsection 4.6.6

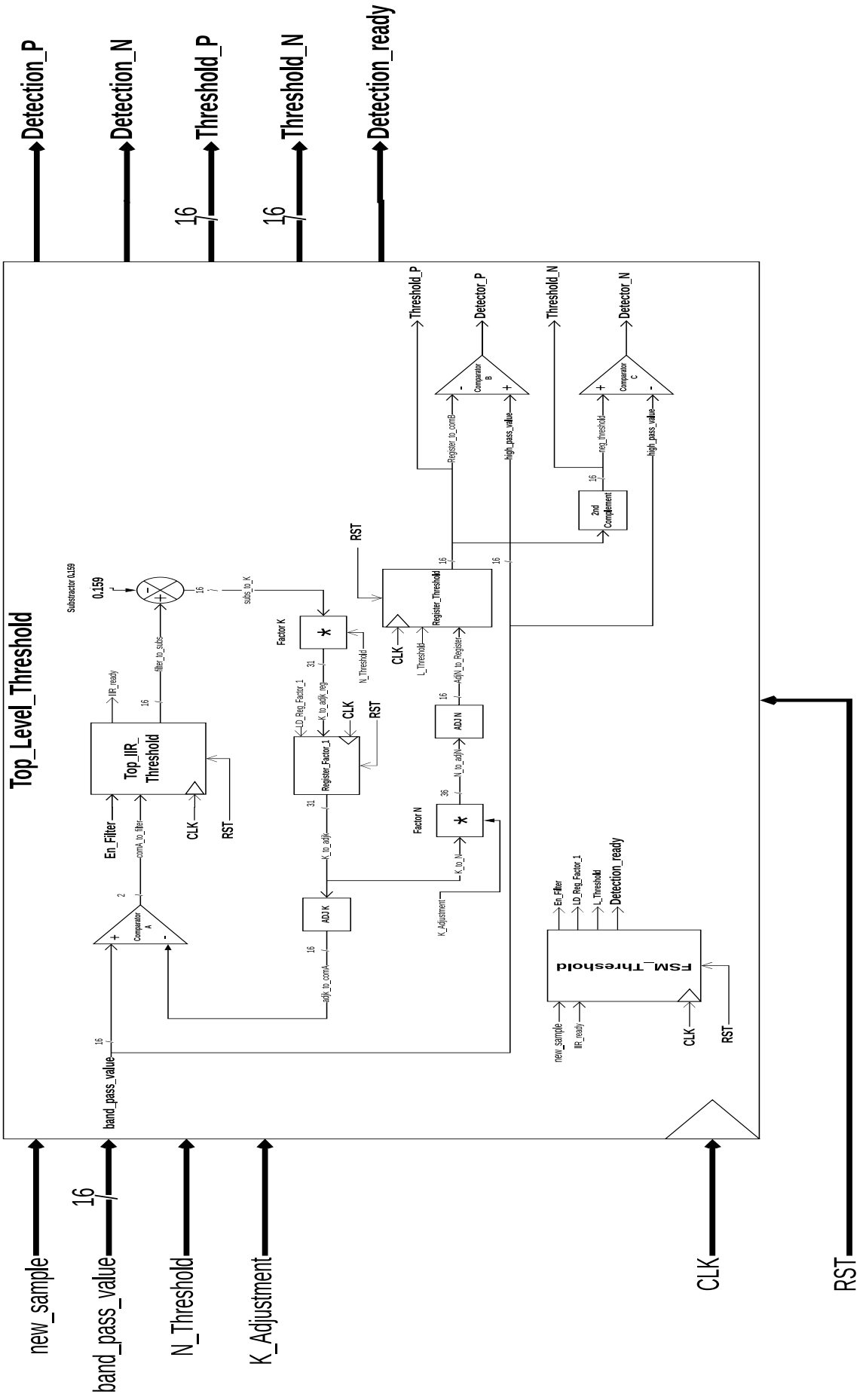


Figure B.10: Top level Threshold module filter architecture components. To return click here subsection 4.6.6

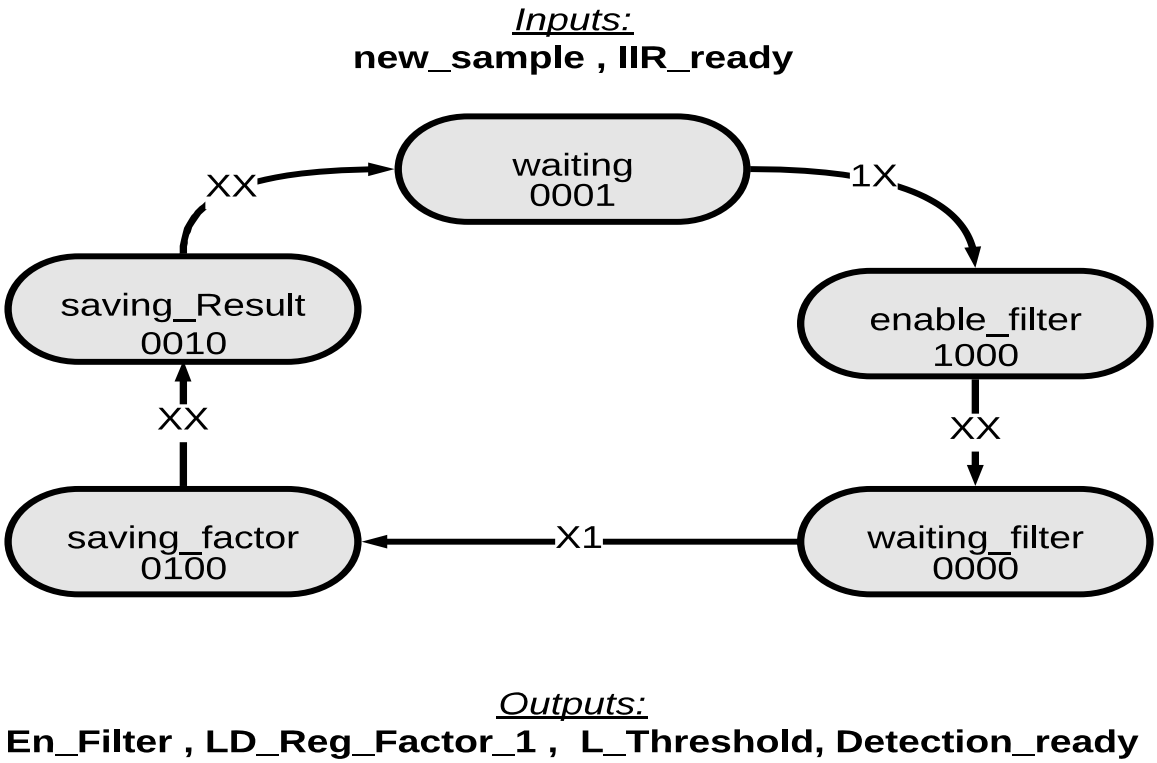


Figure B.11: FSM for computing the Threshold value. To return click here subsection 4.6.6

C

FSM Detection Save and Classification module

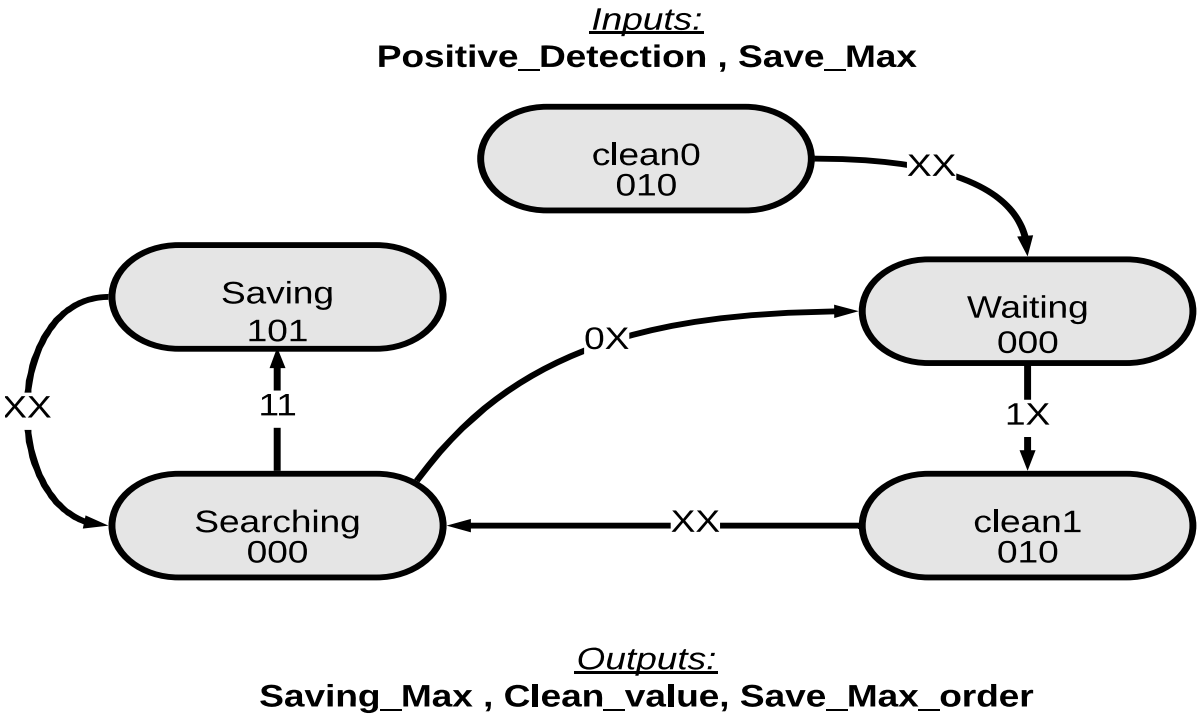


Figure C.1: FSM for saving maximum values . To return click here subsubsection 4.6.7.1

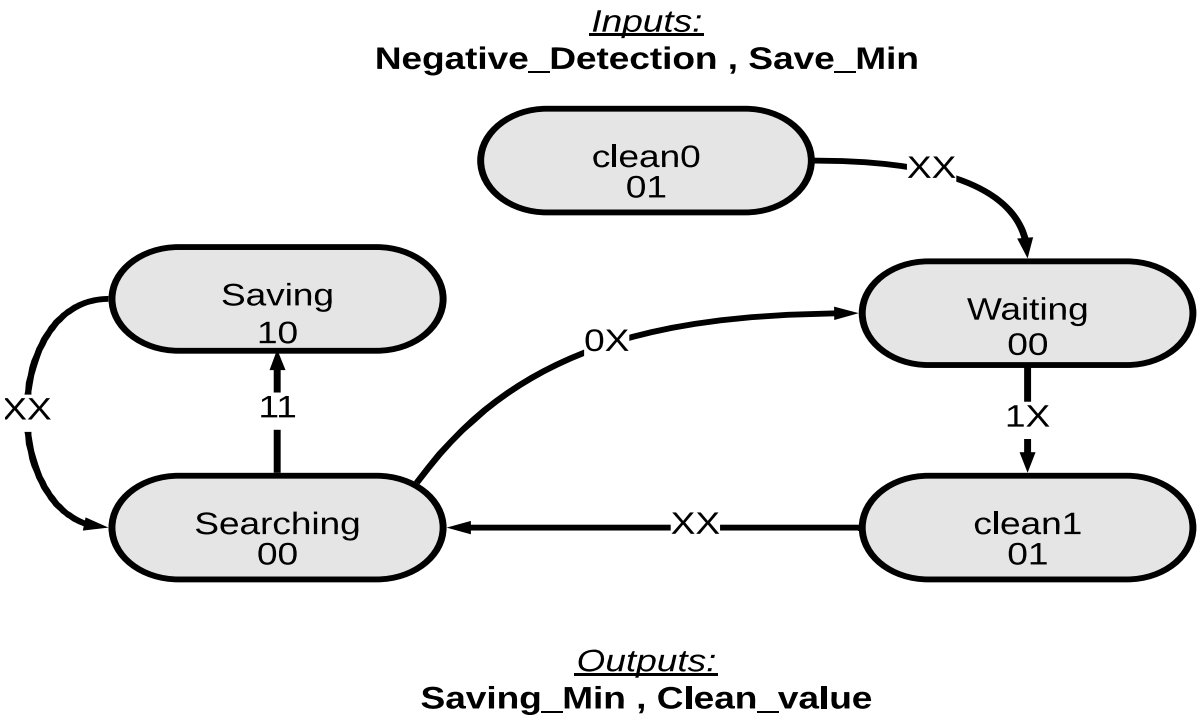


Figure C.2: FSM for saving minimum values. To return click here subsubsection 4.6.7.1

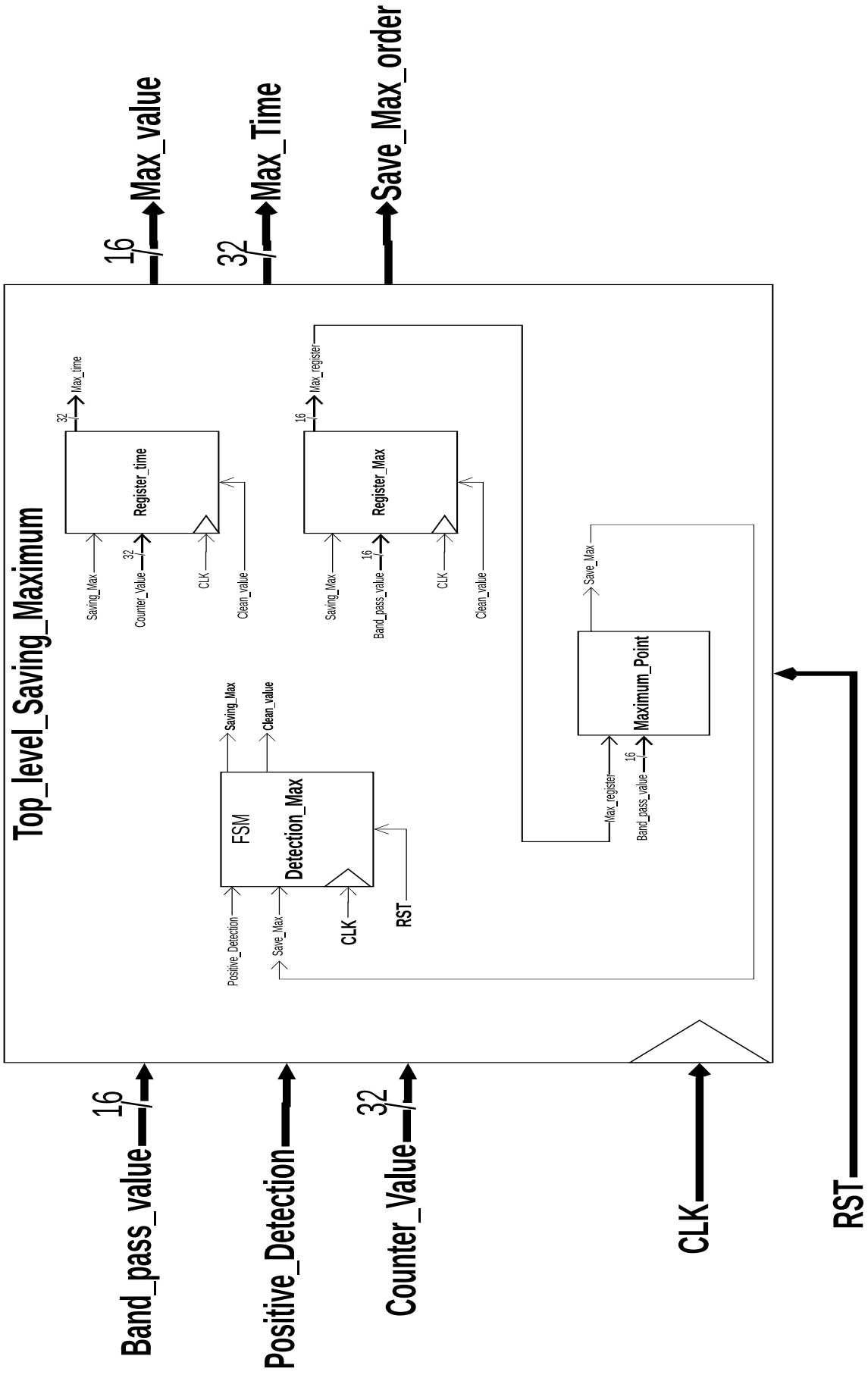


Figure C.3: FSM for saving maximum values. To return click here subsubsection 4.6.7.1

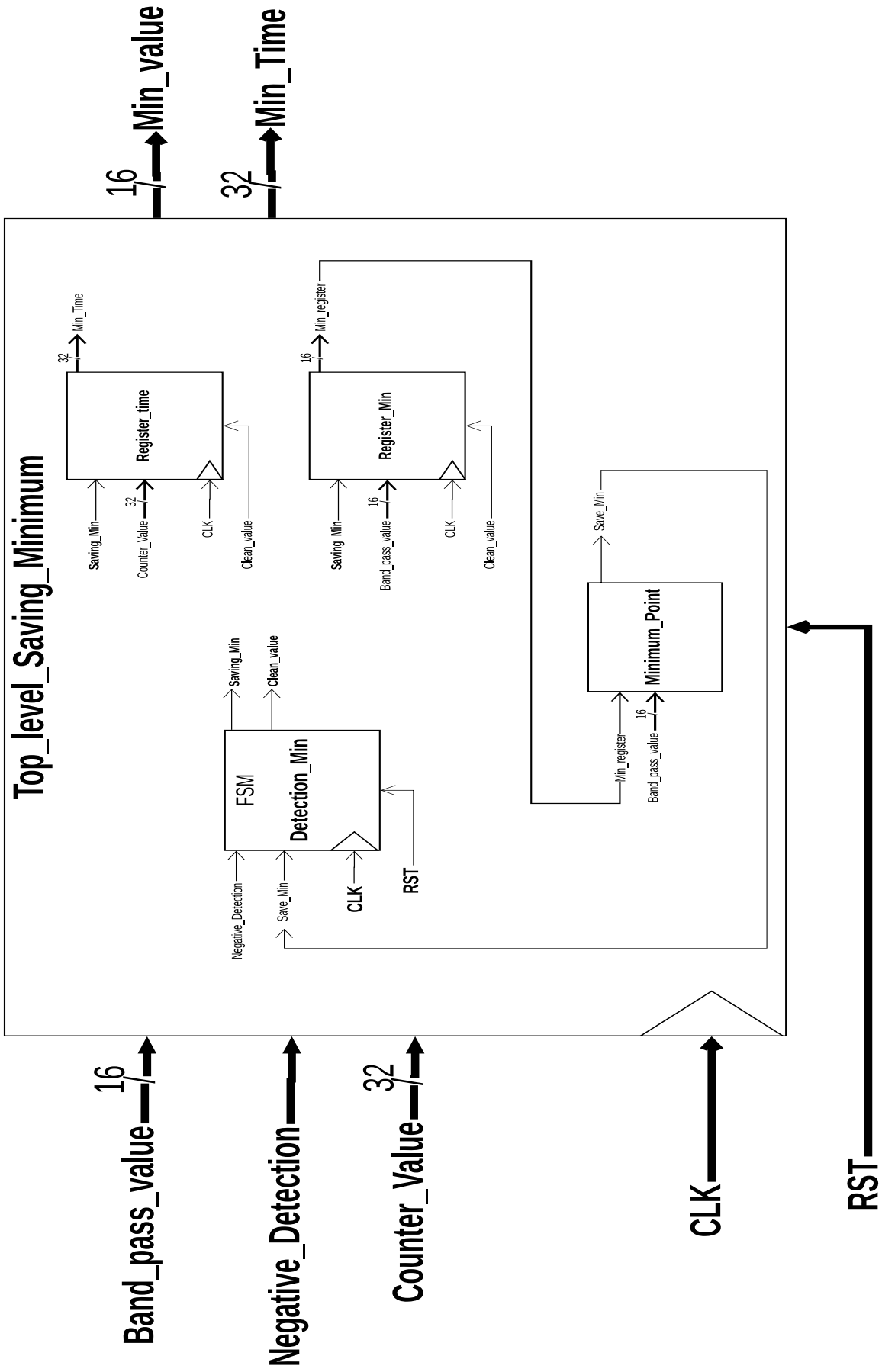


Figure C.4: FSM for saving minimum values. To return click here subsubsection 4.6.7.1

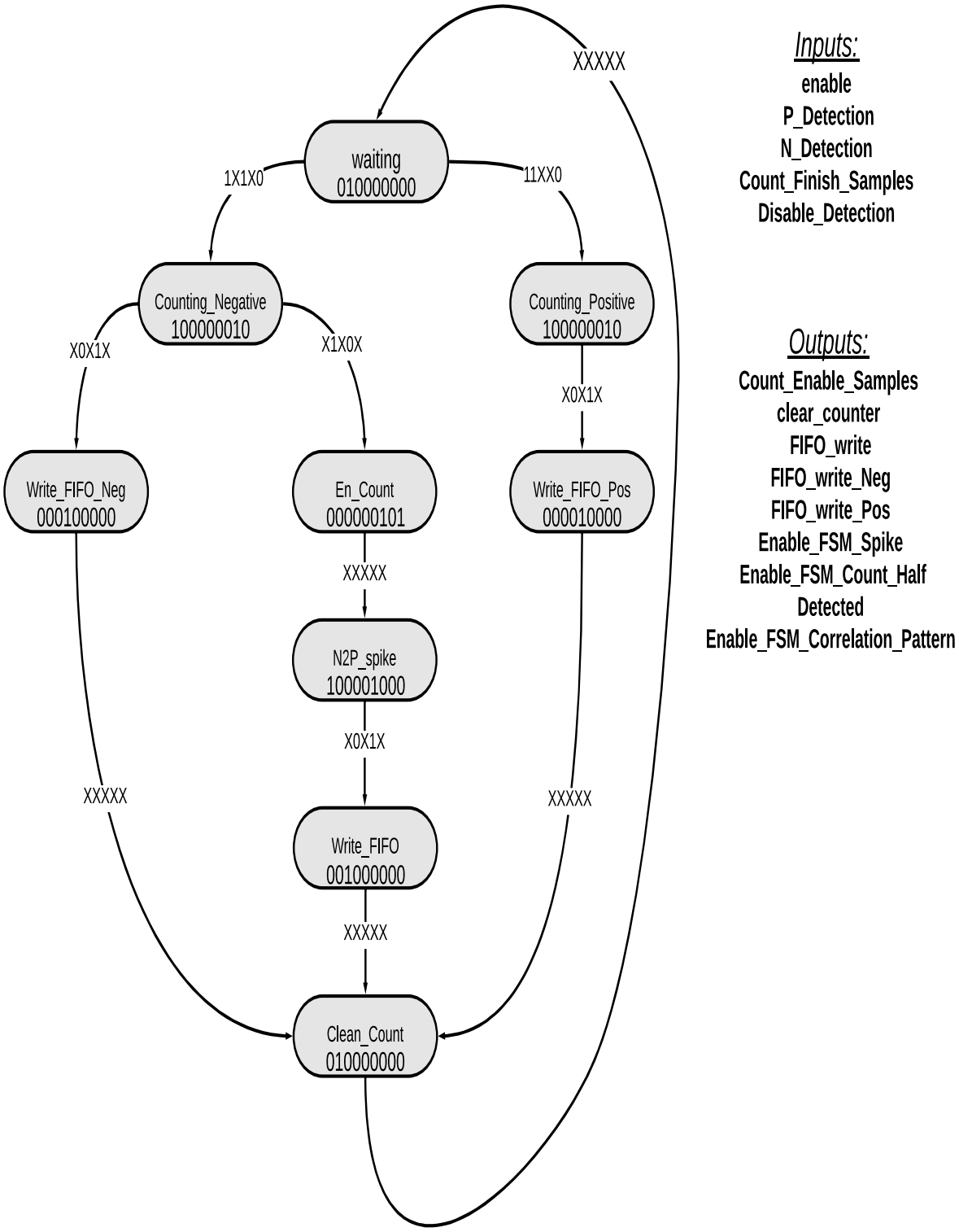


Figure C.5: FSM Action Potential Detection from Macaque Action Potentials

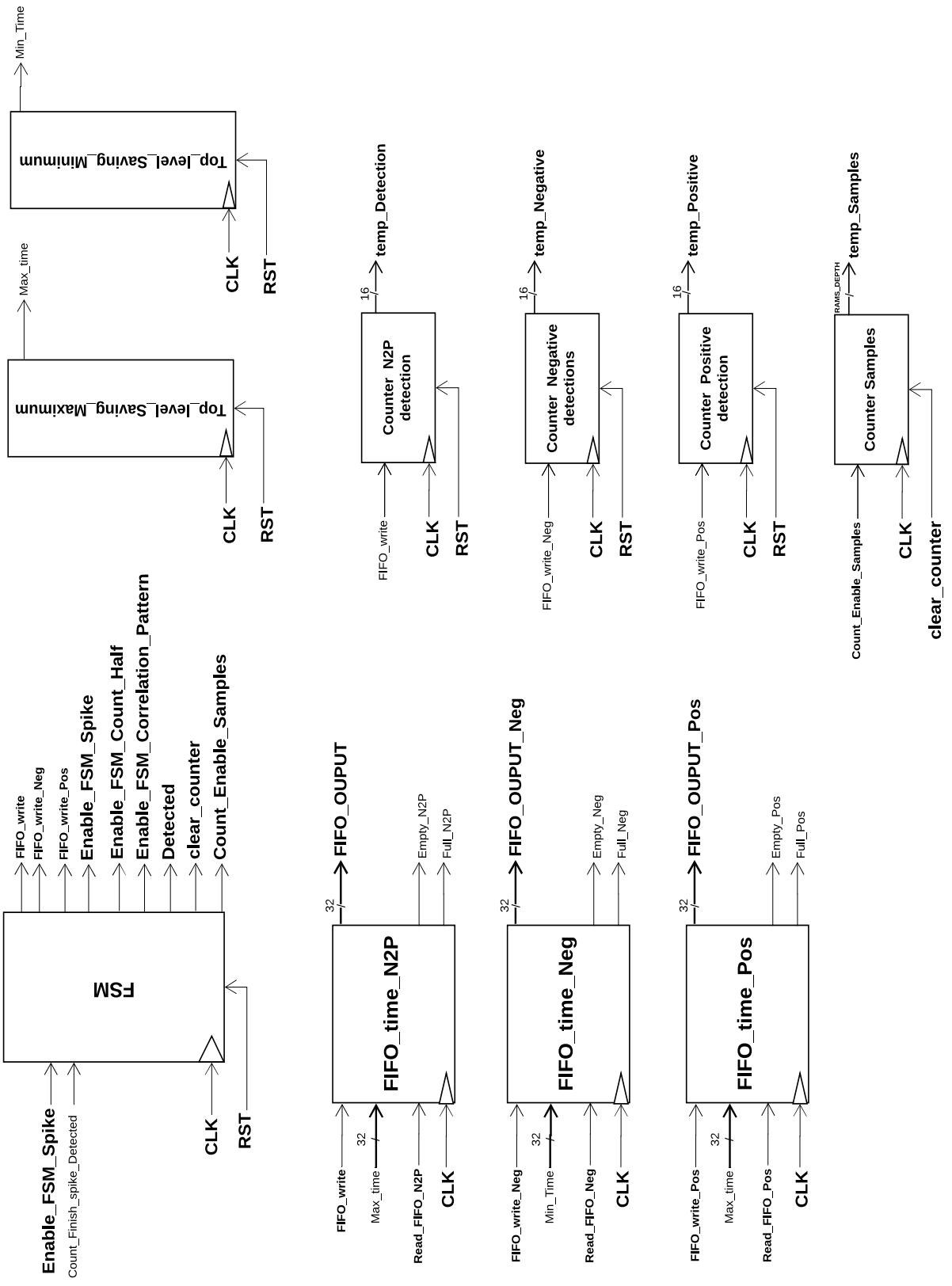


Figure C.6: FSM Action Potential Detection and controlled components for Macaque Action Potentials

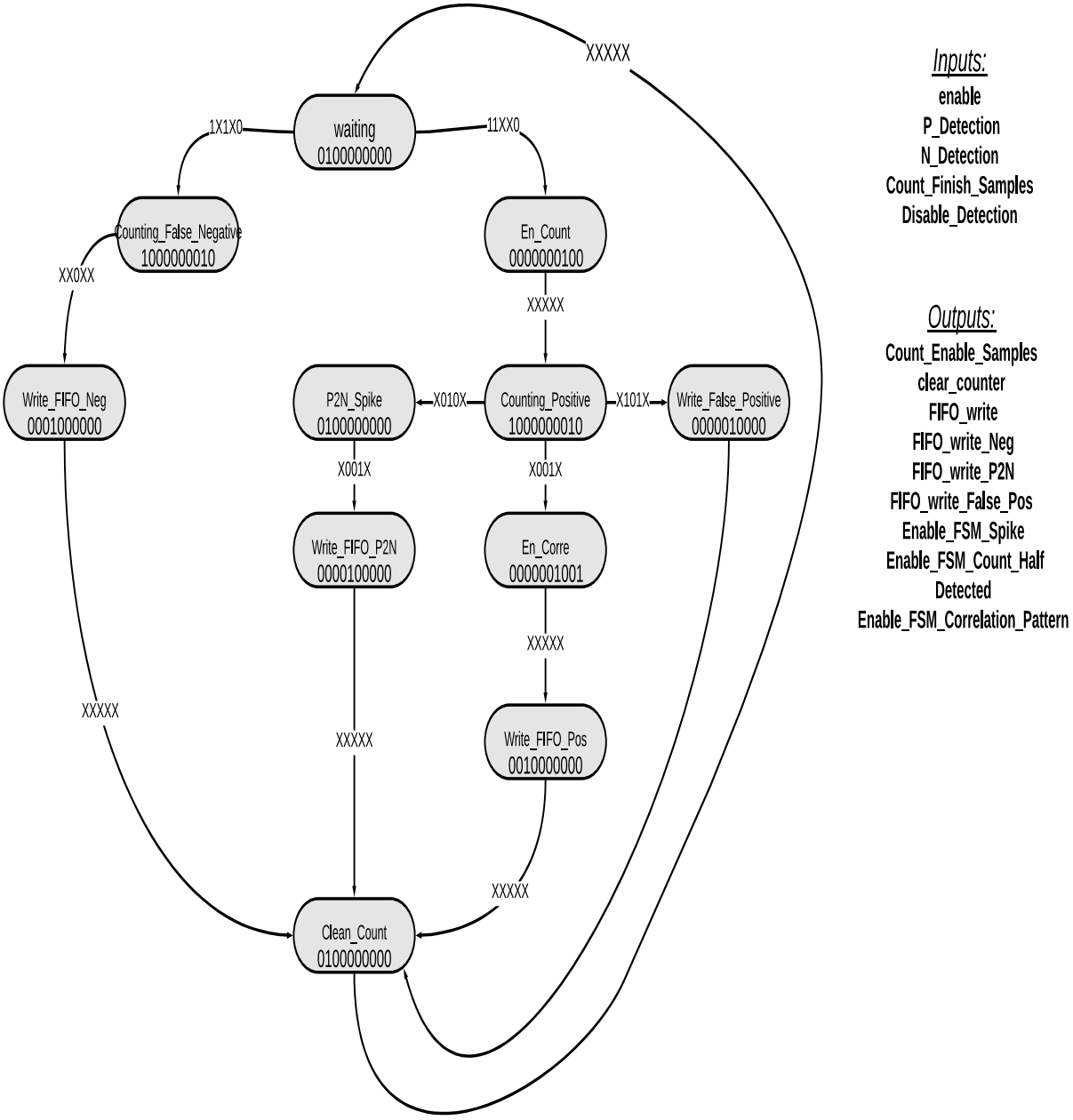


Figure C.7: FSM Action Potential Detection for Pancreatic Action Potential

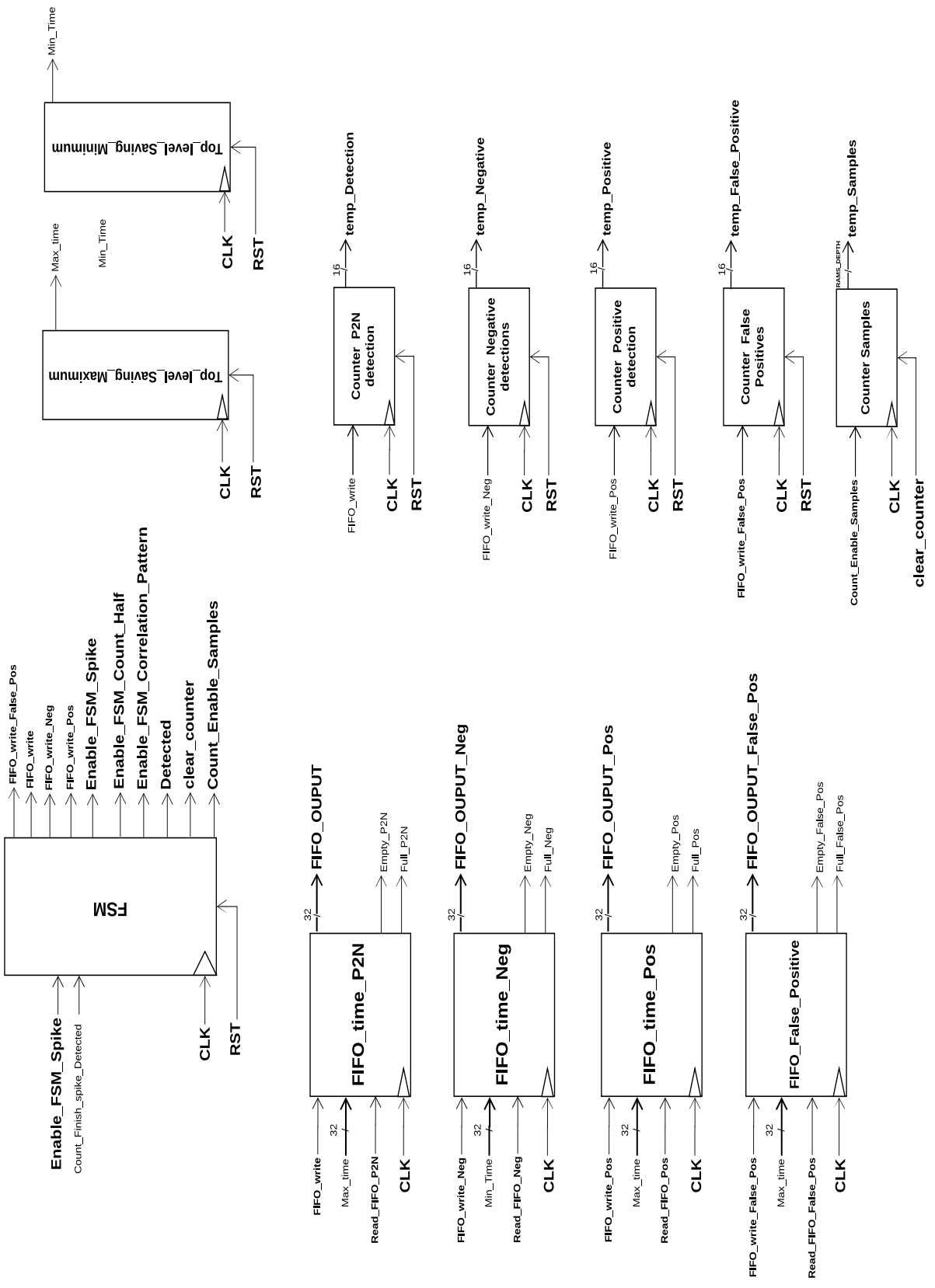


Figure C.8: FSM Action Potential Detection and controlled components for Pancreatic Action Potential

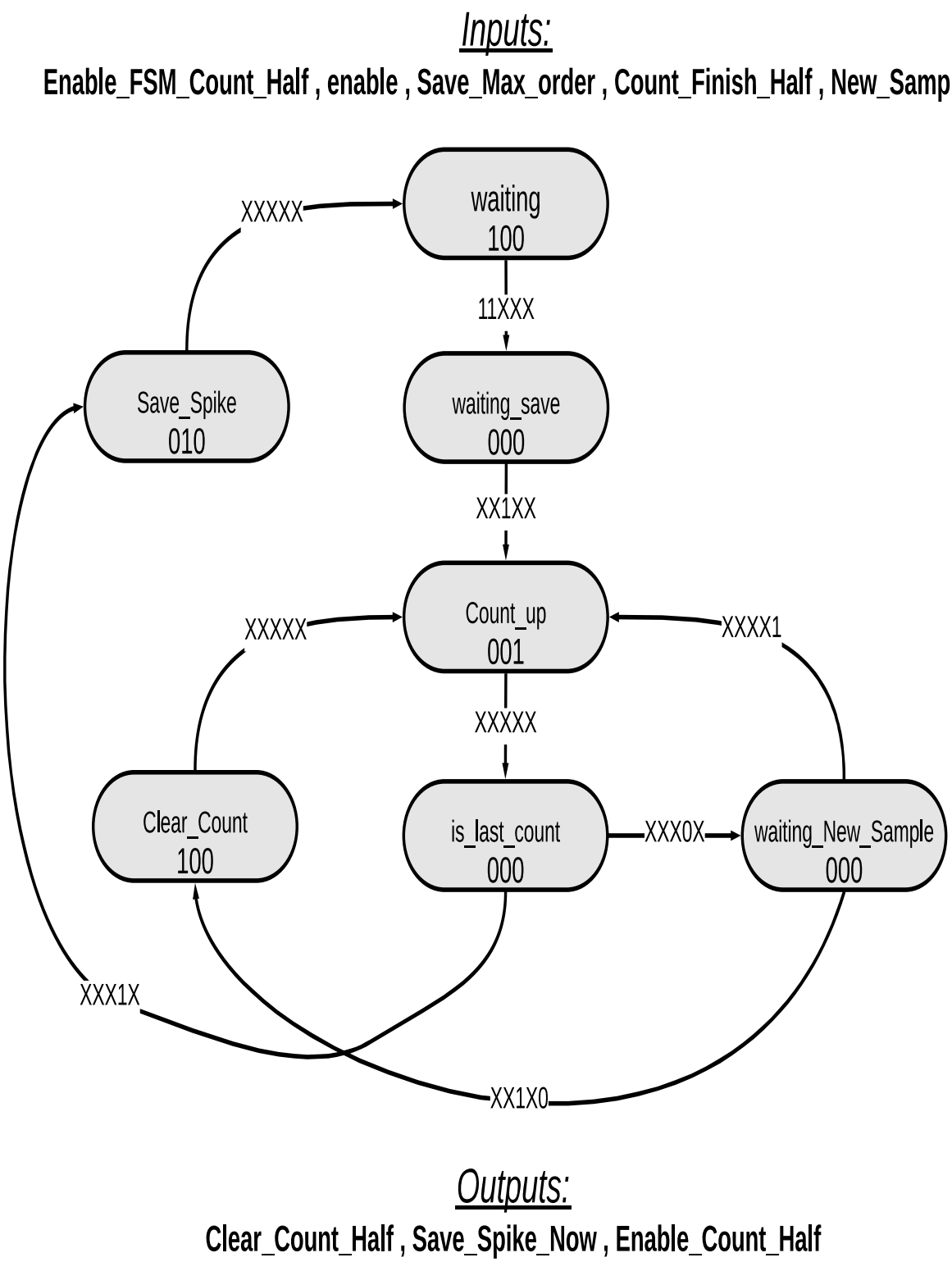


Figure C.9: FSM save Action Potential order

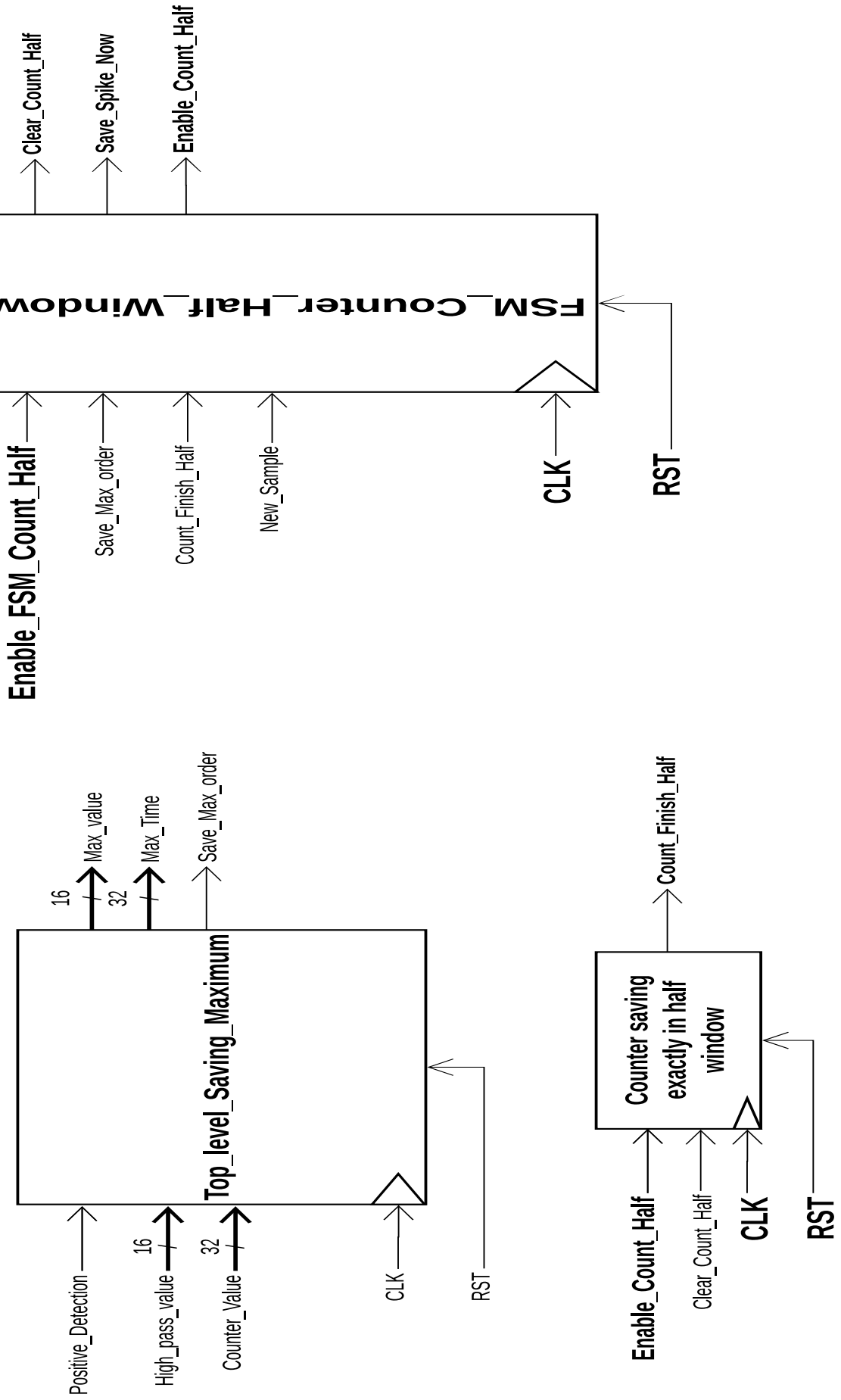


Figure C.10: FSM Counter Half Window and controlled components

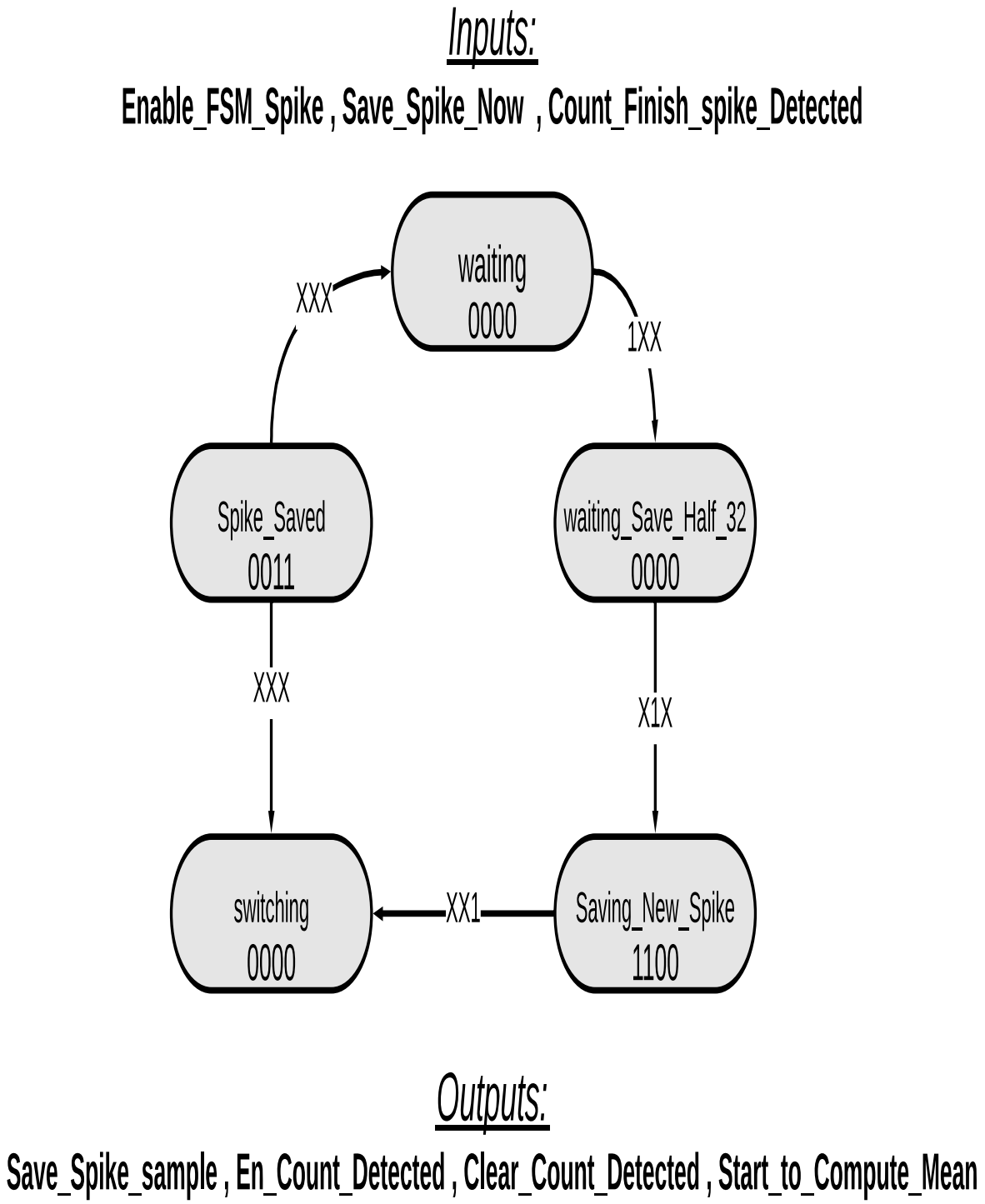


Figure C.11: FSM for saving Action Potential shape into RAM memory.

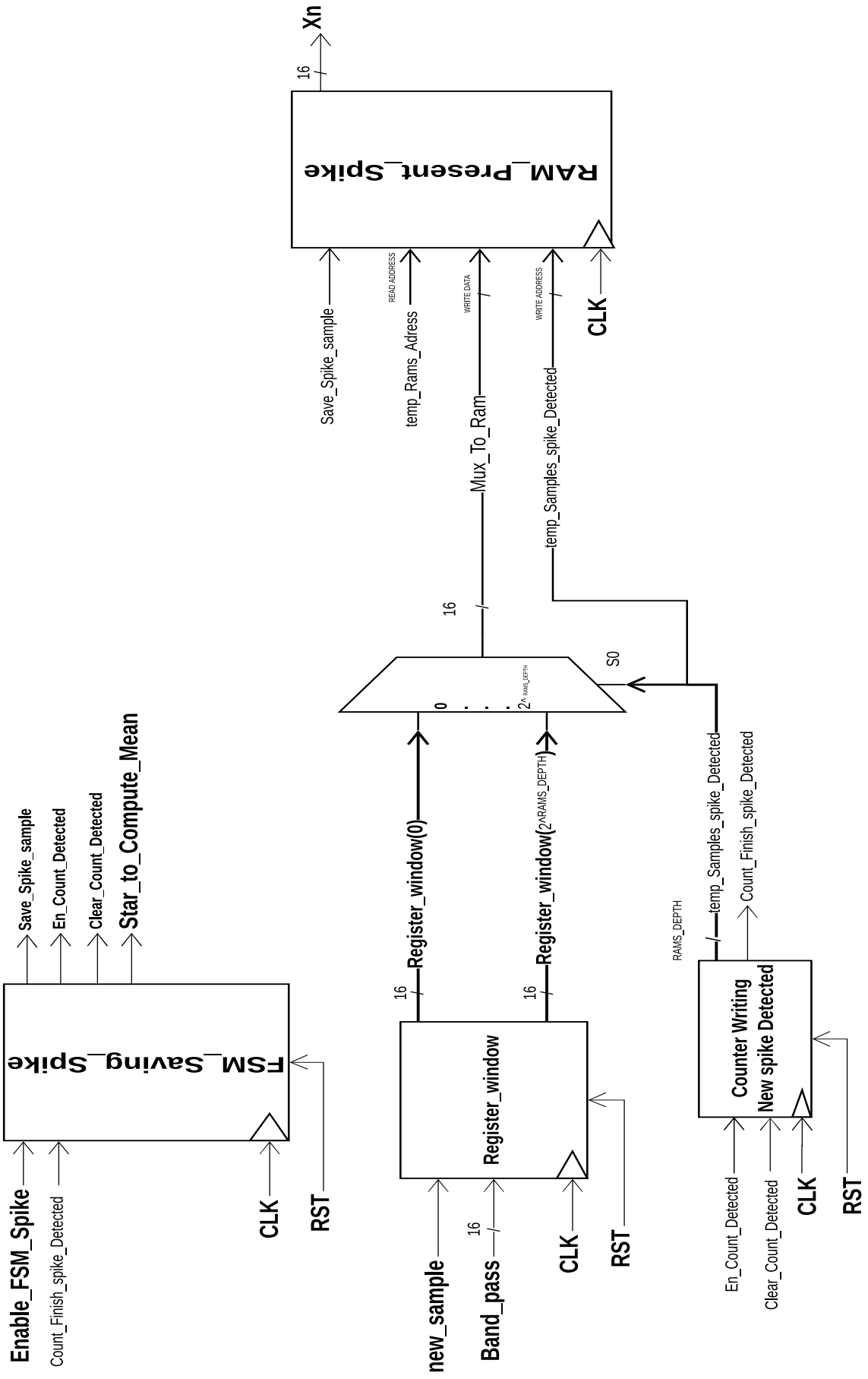


Figure C.12: FSM Saving Spike and Controlled components

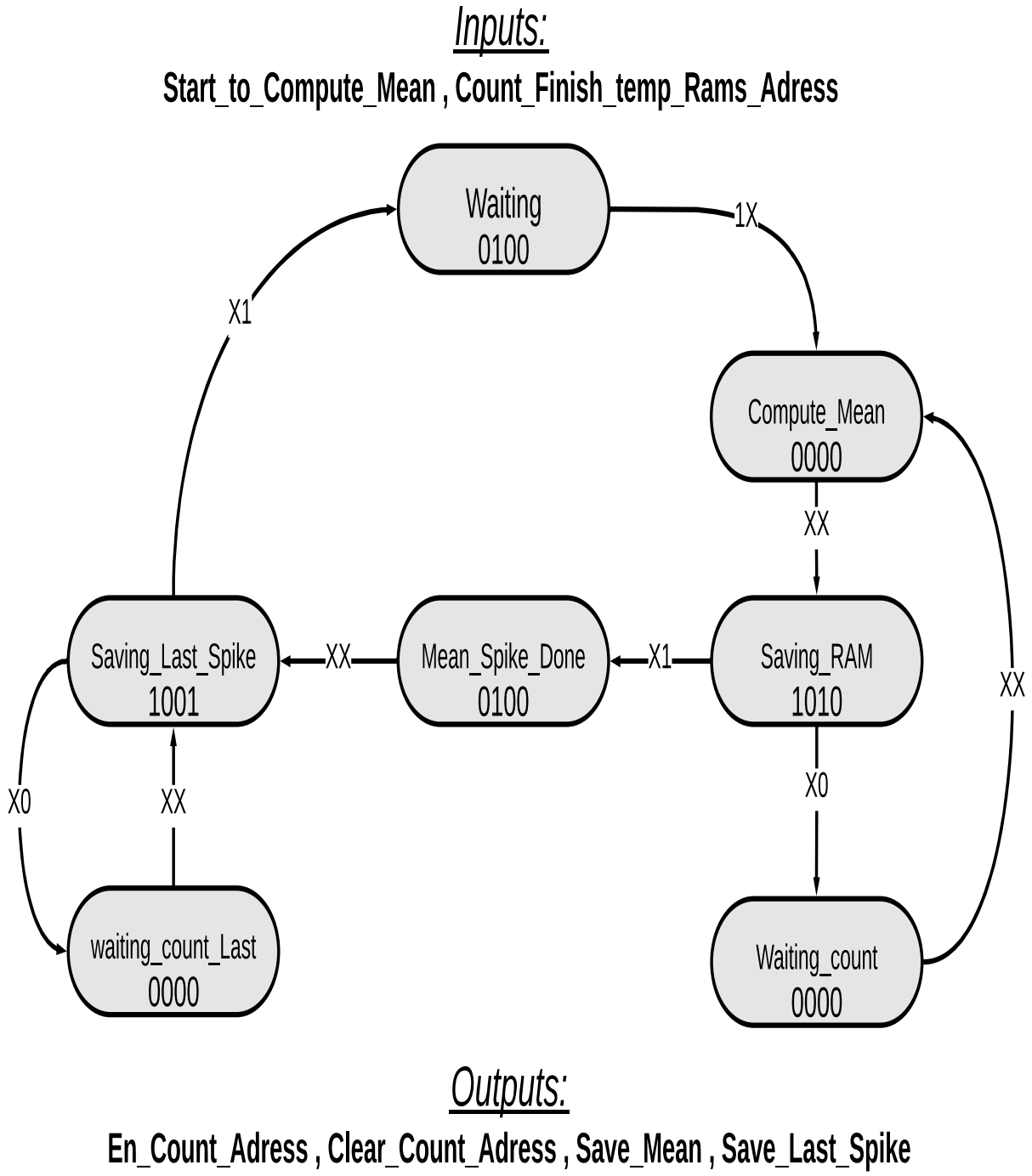


Figure C.13: FSM for computing the Mean Action Potential shape.

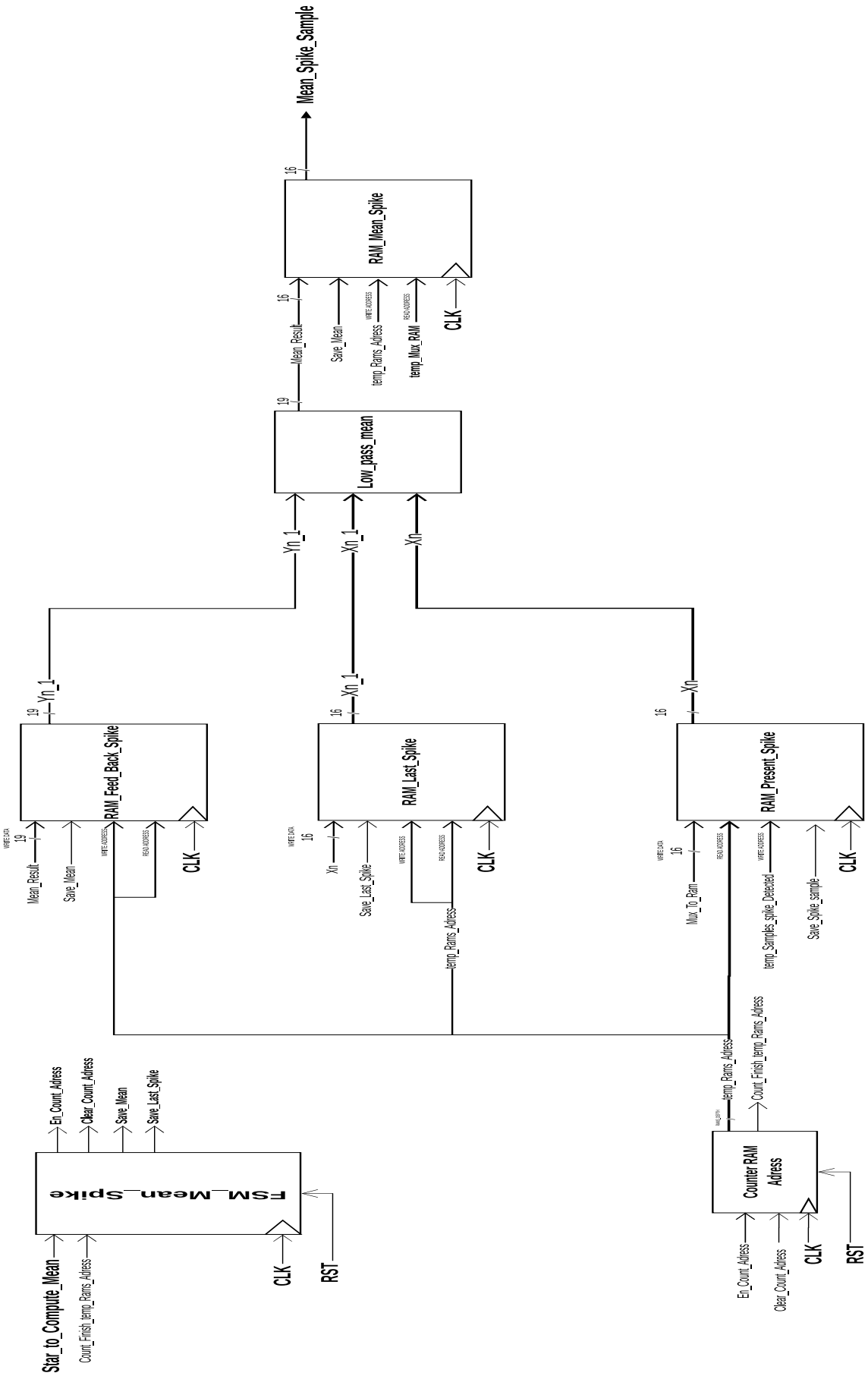


Figure C.14: FSM Mean Spike and Controlled components

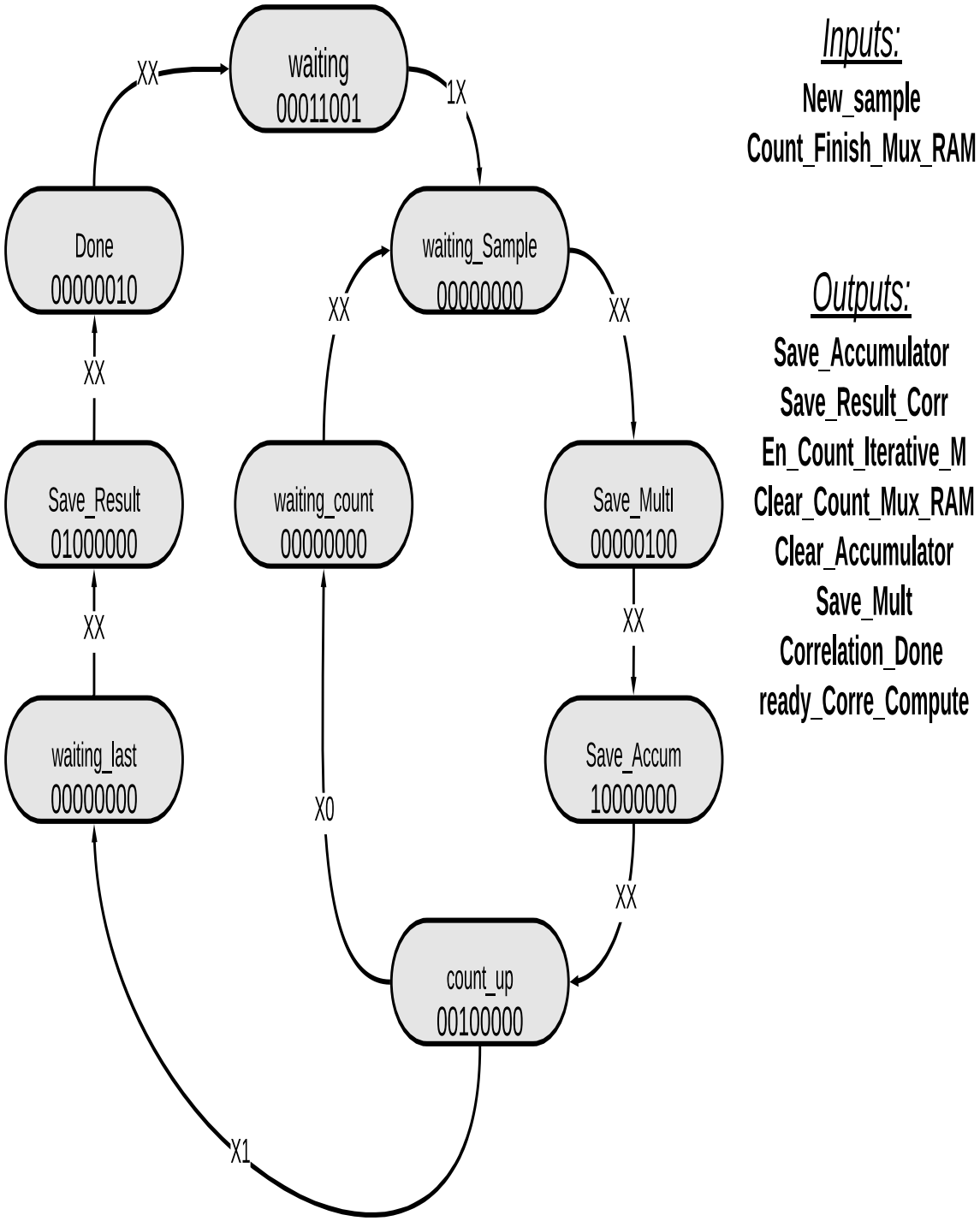


Figure C.15: FSM for computing Correlation signal.

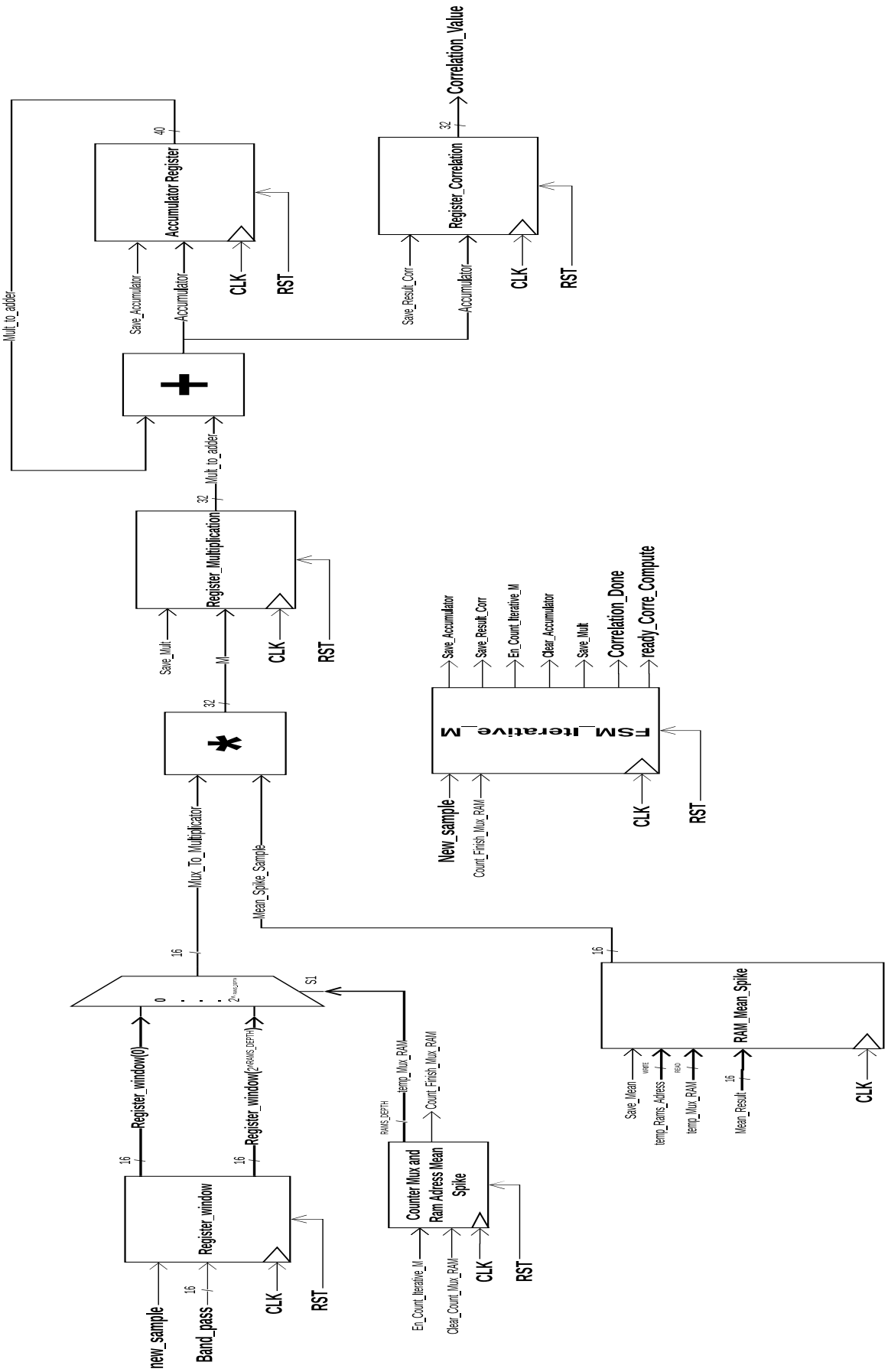


Figure C.16: FSM_Iterative_M for computing Correlation value.

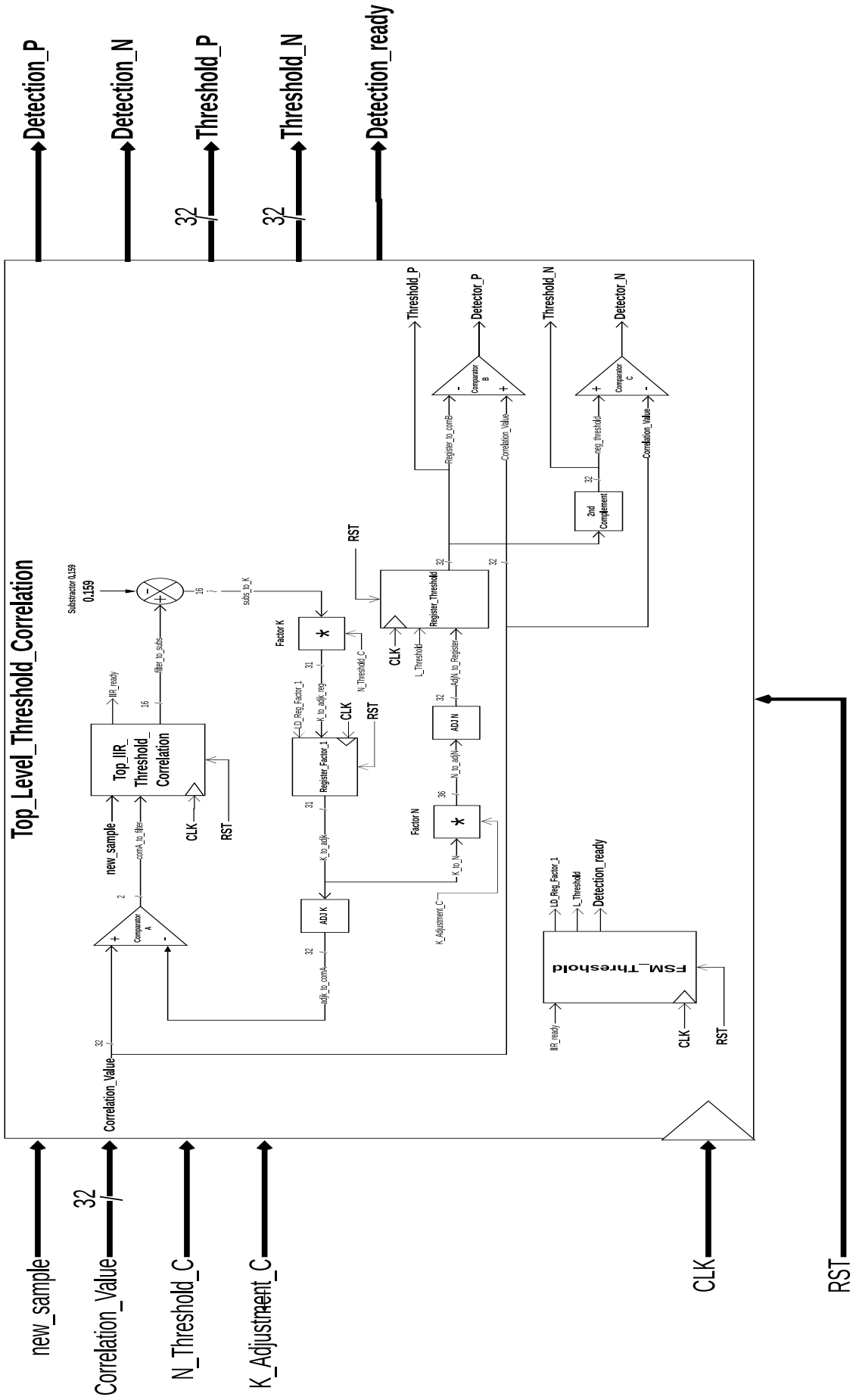


Figure C.17: FSM and components for computing Correlation Thresholds.

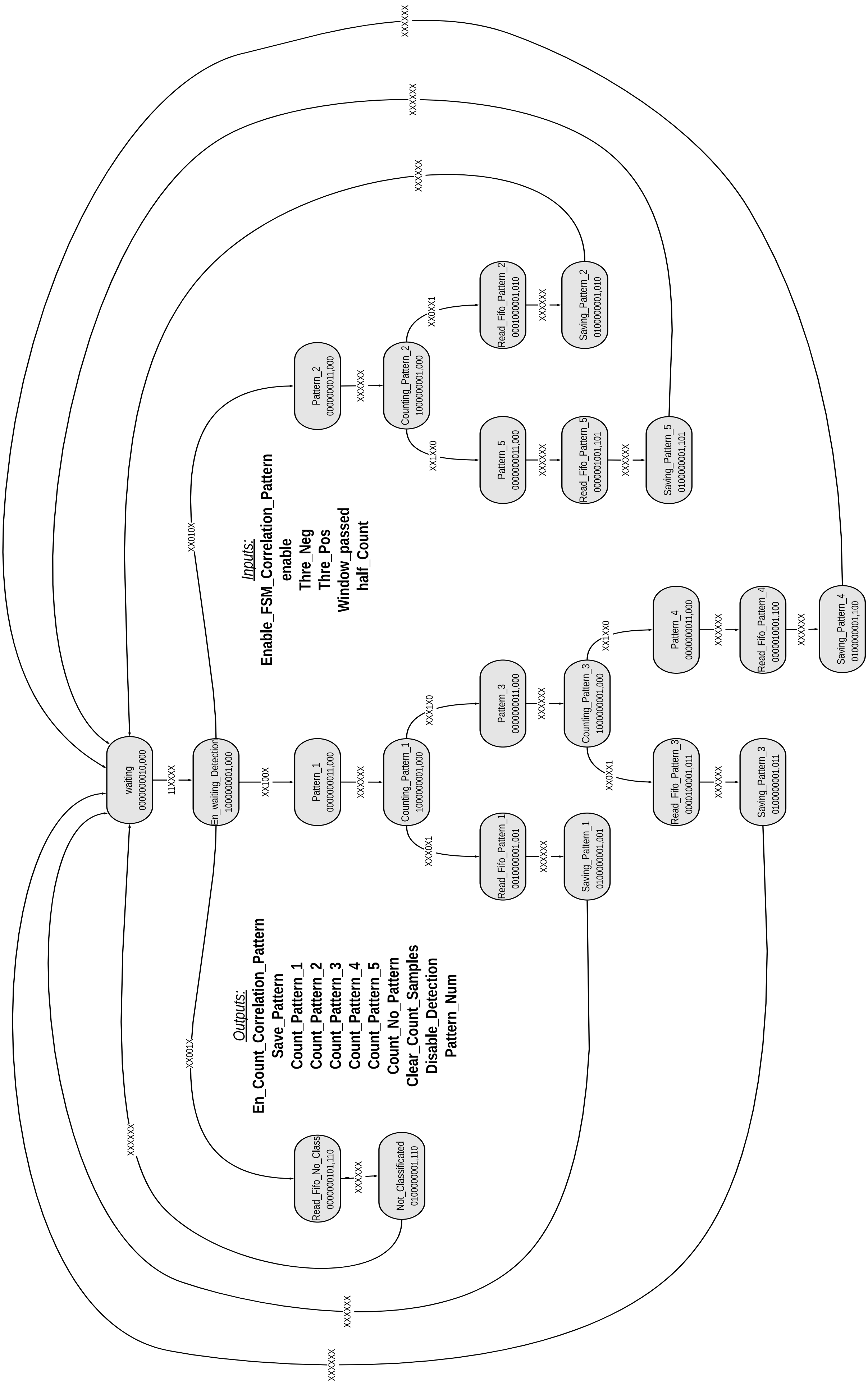


Figure C.18: FSM for detecting Correlation Patterns.

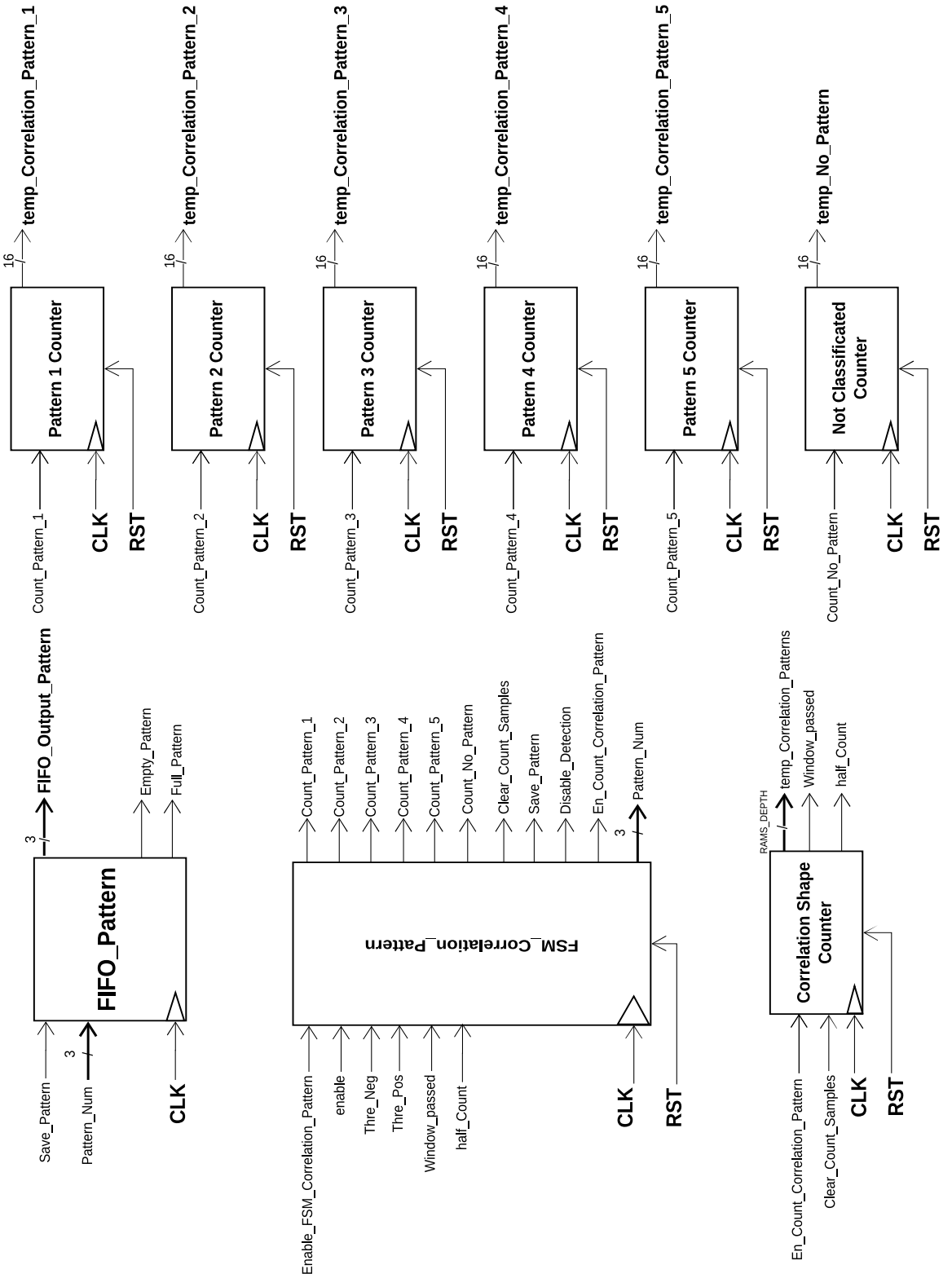


Figure C.19: FSM and components for Action Potential Classification by Correlation Pattern detected and number of detections in each pattern.

D

Matlab scripts and VHDL generated

D.1 Mux_To_Multiplicator

```
RAMS_DEPTH = 5;                 %% A power of two
RAM_Locations = 2^RAMS_DEPTH; %% example 2^5 = 32 memory locations
fid = fopen('Mux_To_Multiplicator.vhd','w');

fprintf(fid,' process(S1,register_window)\n');
fprintf(fid,' begin\n');
fprintf(fid,'   case S1 is\n');
for i=RAM_Locations:-1:1
fprintf(fid,'       when "%s" =>...
 ...Mux_To_Multiplicator <= register_window(%d) ;...
 ...\n',dec2bin((RAM_Locations)-i,RAMS_DEPTH),i-1);
end
fprintf(fid,'       when others  =>...
... Mux_To_Multiplicator <= register_window(%d) ;...
\n',RAM_Locations-1);
fprintf(fid,'   end case;\n');
fprintf(fid,' end process;\n');
```

```
process(S1,register_window)
begin
  case S1 is
      when "00000" => Mux_To_Multiplicator <= register_window(31) ;
      when "00001" => Mux_To_Multiplicator <= register_window(30) ;
      when "00010" => Mux_To_Multiplicator <= register_window(29) ;
      when "00011" => Mux_To_Multiplicator <= register_window(28) ;
      when "00100" => Mux_To_Multiplicator <= register_window(27) ;
      when "00101" => Mux_To_Multiplicator <= register_window(26) ;
      when "00110" => Mux_To_Multiplicator <= register_window(25) ;
      when "00111" => Mux_To_Multiplicator <= register_window(24) ;
      when "01000" => Mux_To_Multiplicator <= register_window(23) ;
      when "01001" => Mux_To_Multiplicator <= register_window(22) ;
      when "01010" => Mux_To_Multiplicator <= register_window(21) ;
      when "01011" => Mux_To_Multiplicator <= register_window(20) ;
      when "01100" => Mux_To_Multiplicator <= register_window(19) ;
      when "01101" => Mux_To_Multiplicator <= register_window(18) ;
      when "01110" => Mux_To_Multiplicator <= register_window(17) ;
      when "01111" => Mux_To_Multiplicator <= register_window(16) ;
      when "10000" => Mux_To_Multiplicator <= register_window(15) ;
      when "10001" => Mux_To_Multiplicator <= register_window(14) ;
      when "10010" => Mux_To_Multiplicator <= register_window(13) ;
      when "10011" => Mux_To_Multiplicator <= register_window(12) ;
      when "10100" => Mux_To_Multiplicator <= register_window(11) ;
      when "10101" => Mux_To_Multiplicator <= register_window(10) ;
      when "10110" => Mux_To_Multiplicator <= register_window(9) ;
      when "10111" => Mux_To_Multiplicator <= register_window(8) ;
      when "11000" => Mux_To_Multiplicator <= register_window(7) ;
      when "11001" => Mux_To_Multiplicator <= register_window(6) ;
      when "11010" => Mux_To_Multiplicator <= register_window(5) ;
      when "11011" => Mux_To_Multiplicator <= register_window(4) ;
      when "11100" => Mux_To_Multiplicator <= register_window(3) ;
      when "11101" => Mux_To_Multiplicator <= register_window(2) ;
      when "11110" => Mux_To_Multiplicator <= register_window(1) ;
      when "11111" => Mux_To_Multiplicator <= register_window(0) ;
      when others  => Mux_To_Multiplicator <= register_window(31) ;
  end case;
end process;
```

D.2 Mux_To_RAM

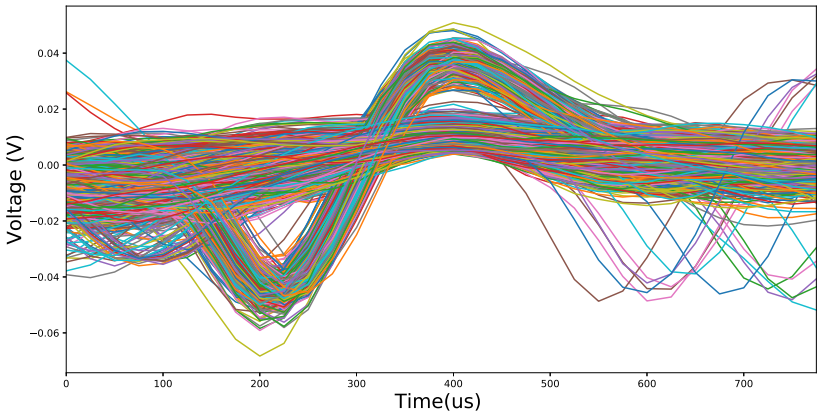
```
RAMS_DEPTH = 5;                 %% A power of two
RAM_Locations = 2^RAMS_DEPTH; %% example 2^5 = 32 memory locations
fid = fopen('Mux_To_Ram.vhd','w');

fprintf(fid,' process(S0,register_window)\n');
fprintf(fid,' begin\n');
fprintf(fid,'   case S0 is\n');
for i=RAM_Locations:-1:1
fprintf(fid,'       when "%s" => Mux_To_Ram <= register_window(%d) ;...
...\n',dec2bin((RAM_Locations)-i,RAMS_DEPTH),i-1);
end
fprintf(fid,'       when others  => Mux_To_Ram <= register_window(%d) ;...
...\n',RAM_Locations-1);
fprintf(fid,'   end case;\n');
fprintf(fid,' end process;\n');
```

```
 process(S0,register_window)
 begin
   case S0 is
        when "00000" => Mux_To_Ram <= register_window(31) ;
        when "00001" => Mux_To_Ram <= register_window(30) ;
        when "00010" => Mux_To_Ram <= register_window(29) ;
        when "00011" => Mux_To_Ram <= register_window(28) ;
        when "00100" => Mux_To_Ram <= register_window(27) ;
        when "00101" => Mux_To_Ram <= register_window(26) ;
        when "00110" => Mux_To_Ram <= register_window(25) ;
        when "00111" => Mux_To_Ram <= register_window(24) ;
        when "01000" => Mux_To_Ram <= register_window(23) ;
        when "01001" => Mux_To_Ram <= register_window(22) ;
        when "01010" => Mux_To_Ram <= register_window(21) ;
        when "01011" => Mux_To_Ram <= register_window(20) ;
        when "01100" => Mux_To_Ram <= register_window(19) ;
        when "01101" => Mux_To_Ram <= register_window(18) ;
        when "01110" => Mux_To_Ram <= register_window(17) ;
        when "01111" => Mux_To_Ram <= register_window(16) ;
        when "10000" => Mux_To_Ram <= register_window(15) ;
        when "10001" => Mux_To_Ram <= register_window(14) ;
        when "10010" => Mux_To_Ram <= register_window(13) ;
        when "10011" => Mux_To_Ram <= register_window(12) ;
        when "10100" => Mux_To_Ram <= register_window(11) ;
        when "10101" => Mux_To_Ram <= register_window(10) ;
        when "10110" => Mux_To_Ram <= register_window(9) ;
        when "10111" => Mux_To_Ram <= register_window(8) ;
        when "11000" => Mux_To_Ram <= register_window(7) ;
        when "11001" => Mux_To_Ram <= register_window(6) ;
        when "11010" => Mux_To_Ram <= register_window(5) ;
        when "11011" => Mux_To_Ram <= register_window(4) ;
        when "11100" => Mux_To_Ram <= register_window(3) ;
        when "11101" => Mux_To_Ram <= register_window(2) ;
        when "11110" => Mux_To_Ram <= register_window(1) ;
        when "11111" => Mux_To_Ram <= register_window(0) ;
        when others  => Mux_To_Ram <= register_window(31) ;
   end case;
 end process;
```

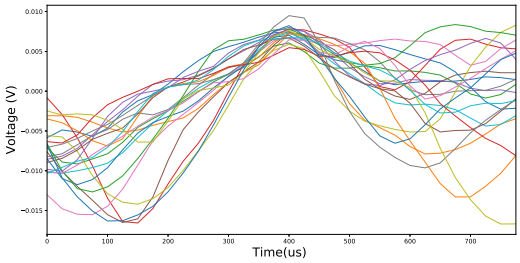
E

Simulation Pattern Results for Macaque monkey biosignal

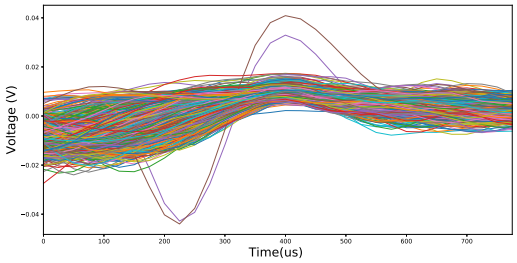
E.1 Threshold value 5



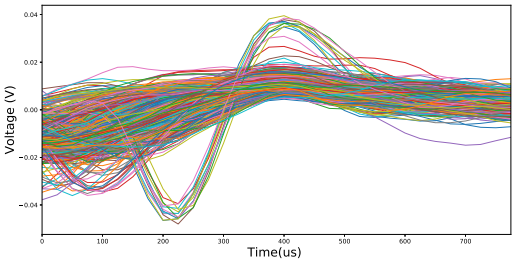
(a) Action Potentials detected. 4696 detections



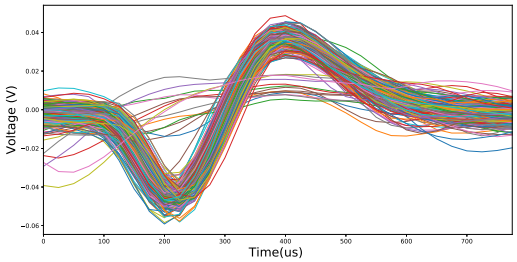
(b) 21 detections classified in Pattern 1



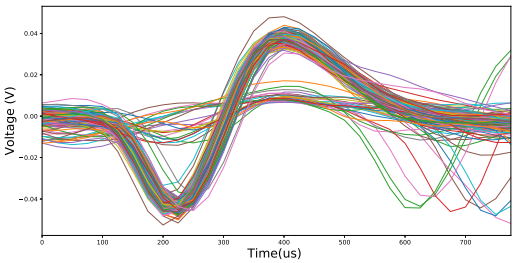
(c) 1401 detections classified in Pattern 2



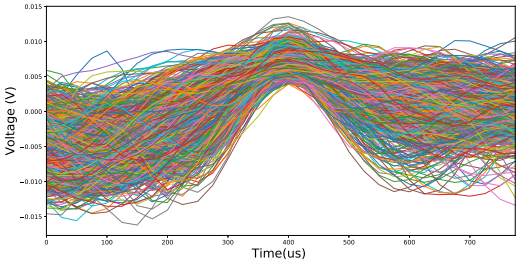
(d) 673 detections classified in Pattern 3



(e) 790 detections classified in Pattern 4



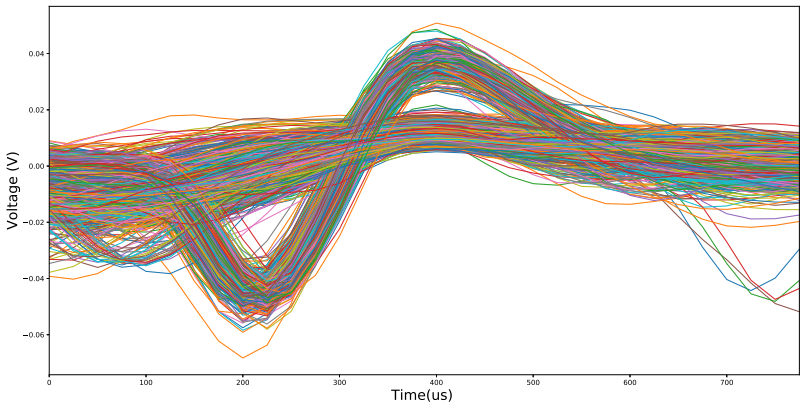
(f) 147 detections classified in Pattern 5



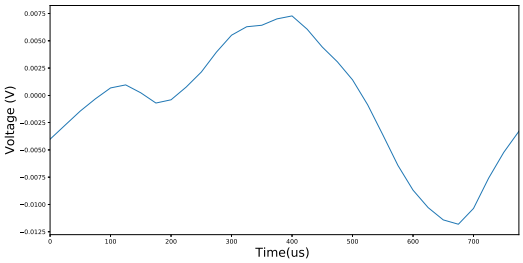
(g) 1604 detections classified in Pattern 0

Figure E.1: Action Potentials detected and classified by Correlation Pattern. Threshold value = 5, Correlation Threshold = 10

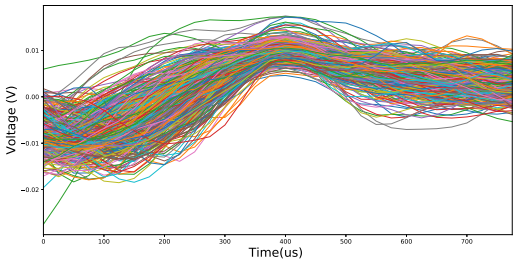
E.2 Threshold value 7



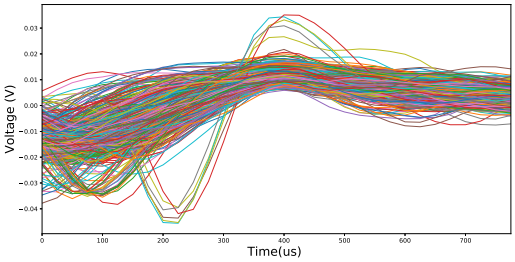
(a) Action Potentials detected. 2053 detections



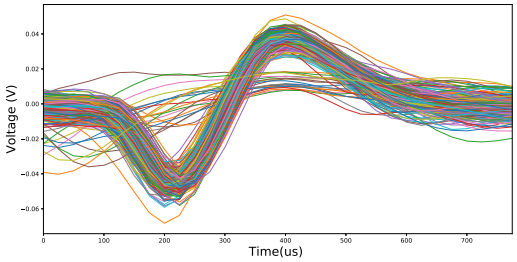
(b) 2 detections classified in Pattern 1



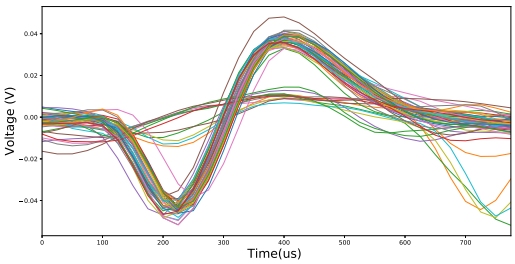
(c) 437 detections classified in Pattern 2



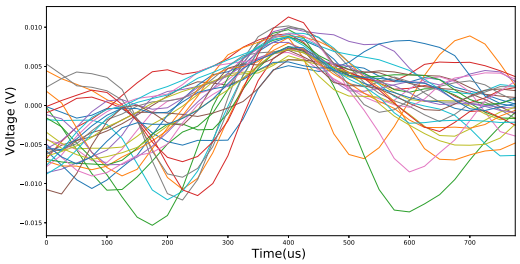
(d) 604 detections classified in Pattern 3



(e) 912 detections classified in Pattern 4



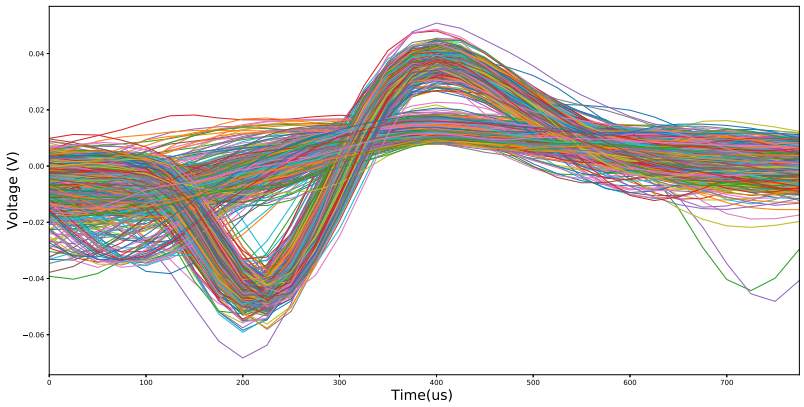
(f) 56 detections classified in Pattern 5



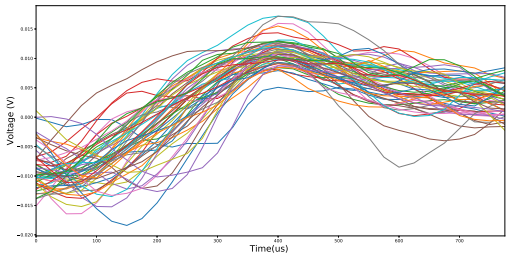
(g) 30 detections classified in Pattern 5

Figure E.2: Action Potentials detected and classified by Correlation Pattern. Threshold value = 7, Correlation Threshold = 10

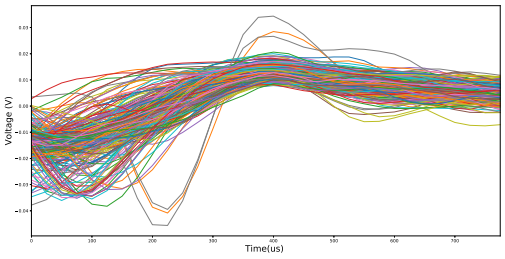
E.3 Threshold value 9



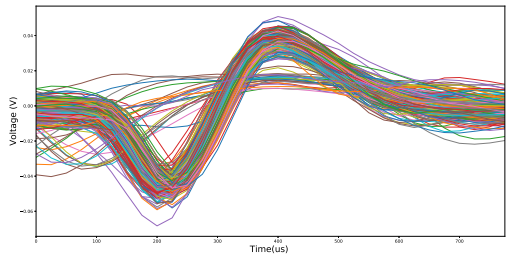
(a) Action Potentials detected. 1289 detections



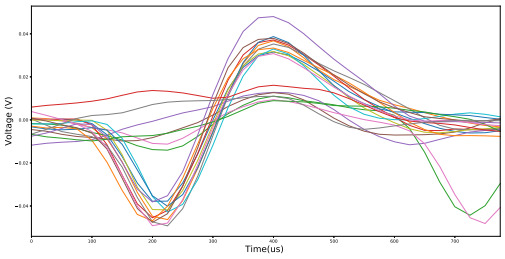
(b) 56 detections classified in Pattern 2



(c) 18 detections classified in Pattern 3



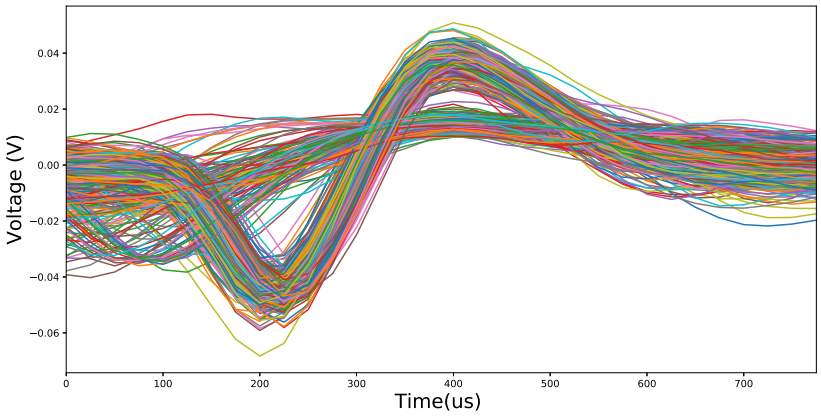
(d) 958 detections classified in Pattern 4



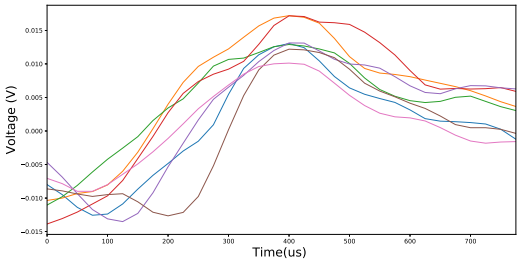
(e) 18 detections classified in Pattern5

Figure E.3: Action Potentials detected and classified by Correlation Pattern. Threshold value = 9, Correlation Threshold = 10

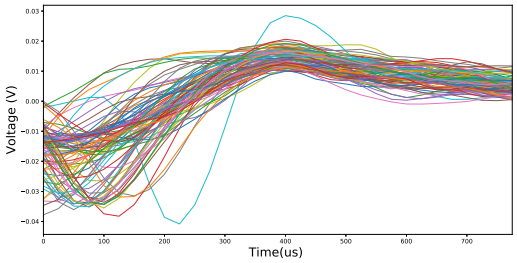
E.4 Threshold value 11



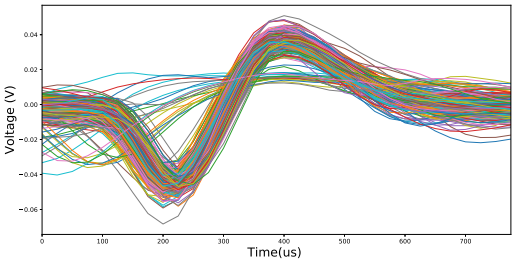
(a) Action Potentials detected. 1064 detections



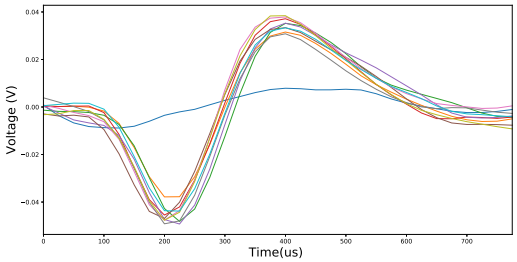
(b) 7 detections classified in Pattern 2



(c) 87 detections classified in Pattern 3



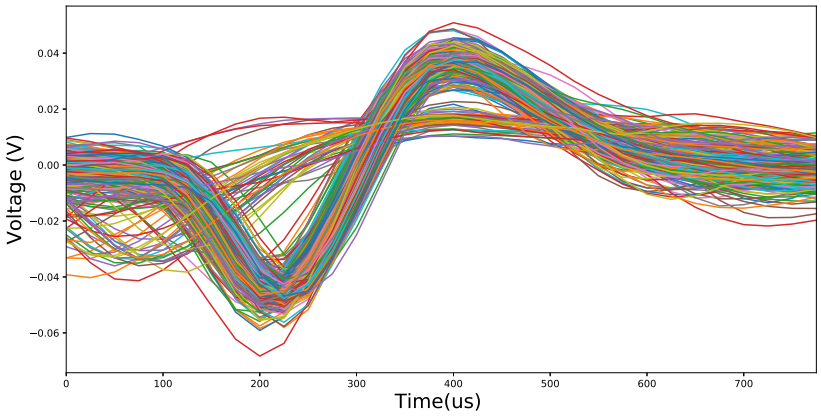
(d) 960 detections classified in Pattern 4



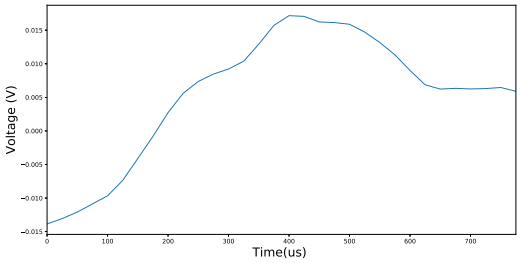
(e) 10 detections classified in Pattern 5

Figure E.4: Action Potentials detected and classified by Correlation Pattern. Threshold value = 11, Correlation Threshold = 10

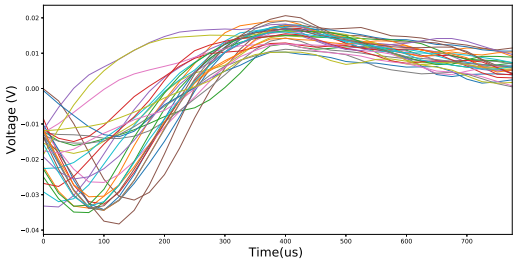
E.5 Threshold value 13



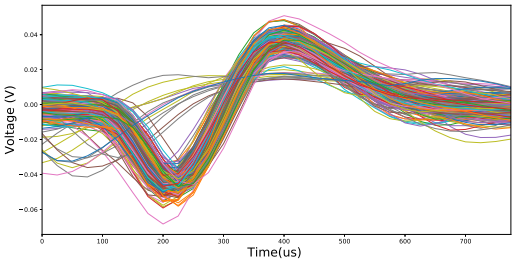
(a) Action Potentials detected. 993 detections



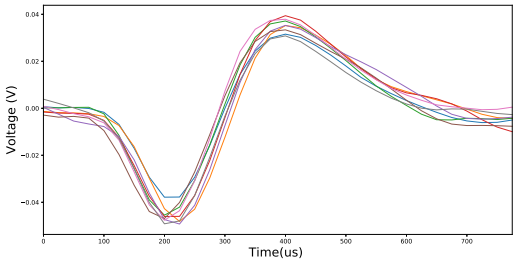
(b) 1 detections classified in Pattern 2



(c) 30 detections classified in Pattern 3



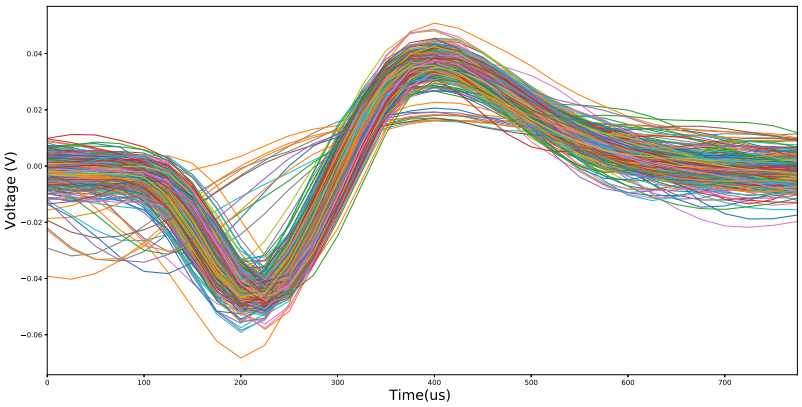
(d) 954 detections classified in Pattern 4



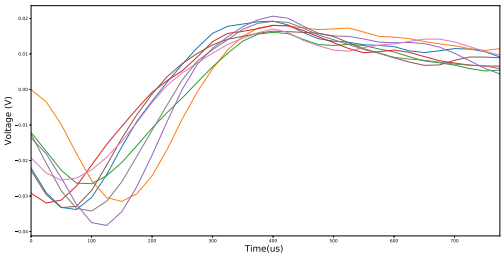
(e) 8 detections classified in Pattern 5

Figure E.5: Action Potentials detected and classified by Correlation Pattern. Threshold value = 13, Correlation Threshold = 10

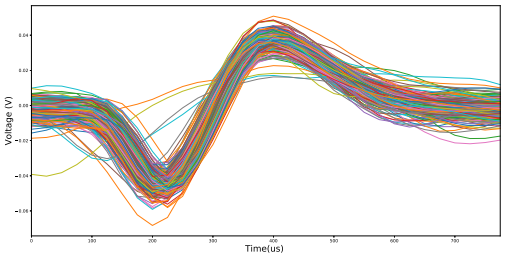
E.6 Threshold value 15



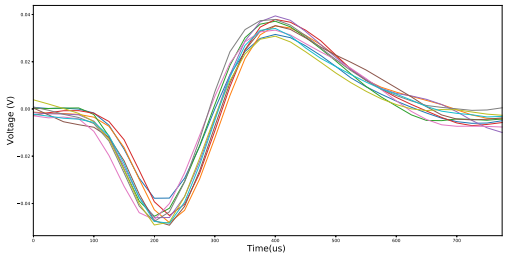
(a) Action Potentials detected. 960 detections



(b) 8 detections classified in Pattern 3



(c) 942 detections classified in Pattern 4



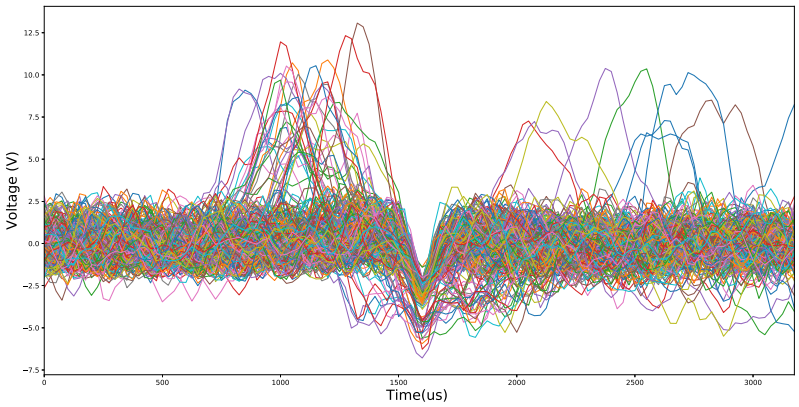
(d) 10 detections classified in Pattern 5

Figure E.6: Action Potentials detected and classified by Correlation Pattern. Threshold value = 15, Correlation Threshold = 10

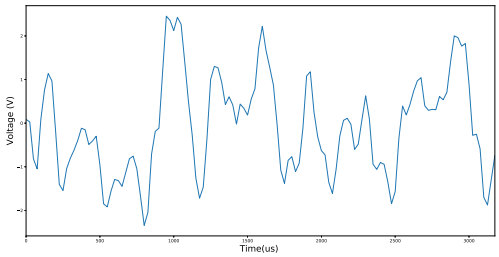
F

Simulation Pattern Results for Human pancreatic biosignal

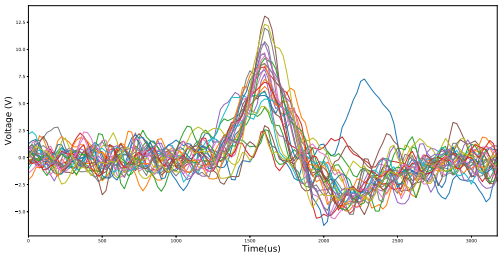
F.1 Threshold value 9



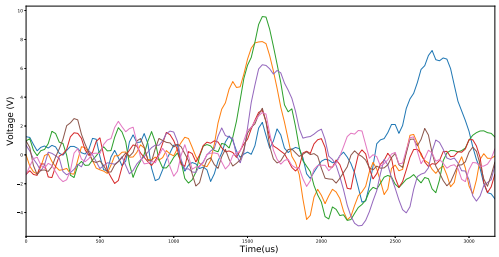
(a) Action Potentials detected. 261 detections



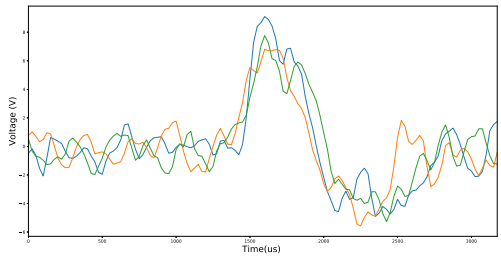
(b) 1 detections classified in Pattern 1



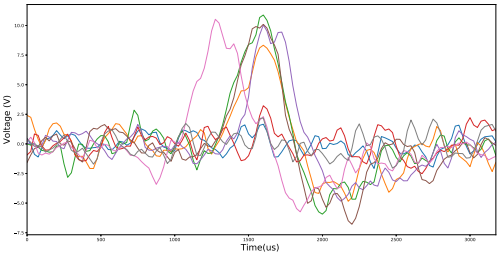
(c) 29 detections classified in Pattern 2



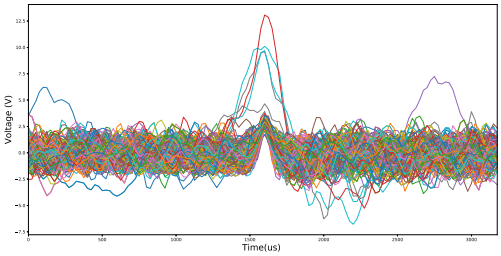
(d) 7 detections classified in Pattern 3



(e) 29 detections classified in Pattern 4



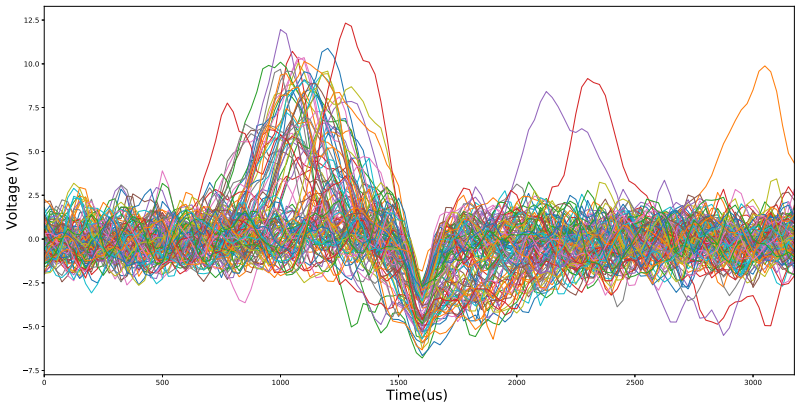
(f) 8 detections classified in Pattern 5



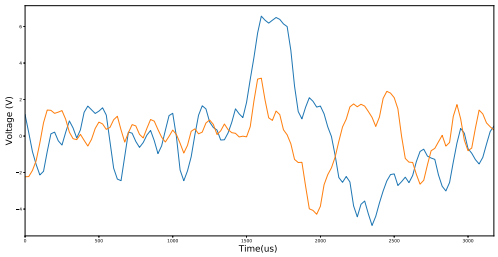
(g) 213 detections classified in Pattern 0

Figure F.1: Action Potentials detected and classified by Correlation Pattern. Threshold value = 9, Correlation Threshold = 10

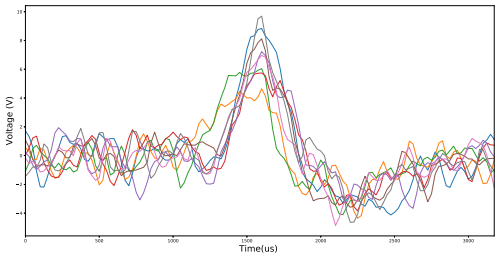
F.2 Threshold value 11



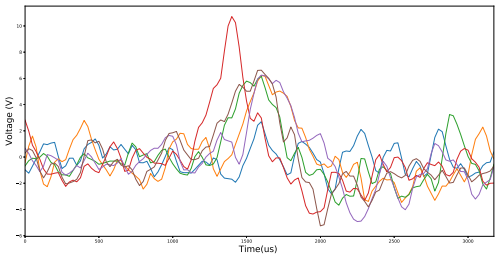
(a) Action Potentials detected. 114 detections



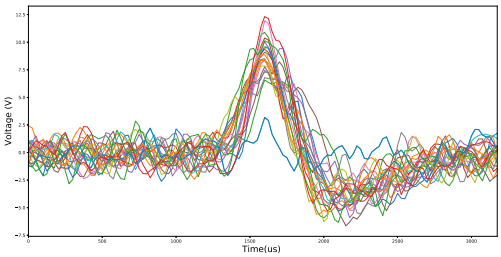
(b) 2 detections classified in Pattern 1



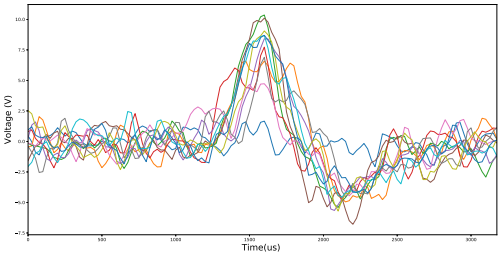
(c) 8 detections classified in Pattern 2



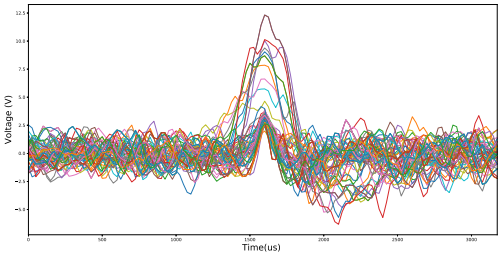
(d) 6 detections classified in Pattern 3



(e) 24 detections classified in Pattern 4



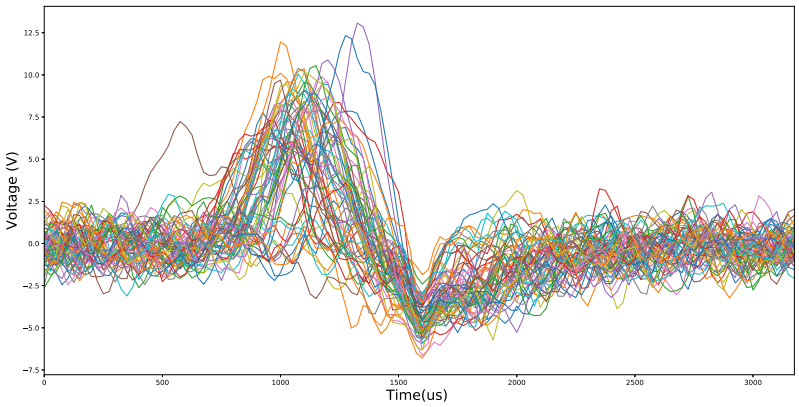
(f) 11 detections classified in Pattern 5



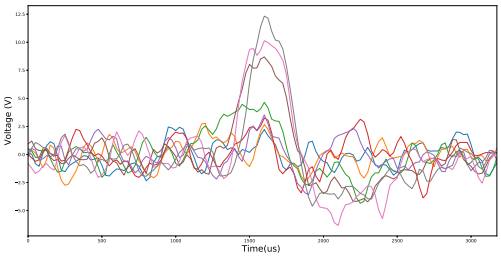
(g) 63 detections classified in Pattern 0

Figure F.2: Action Potentials detected and classified by Correlation Pattern. Threshold value = 11, Correlation Threshold = 10

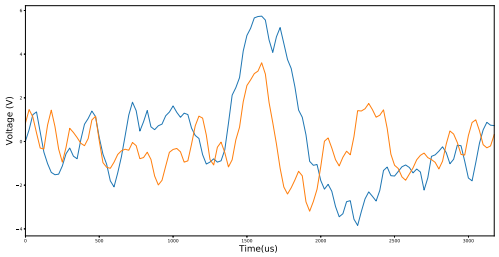
F.3 Threshold value 13



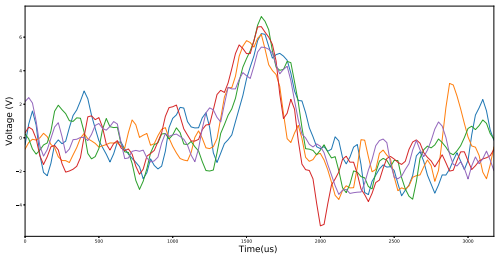
(a) Action Potentials detected. 63 detections



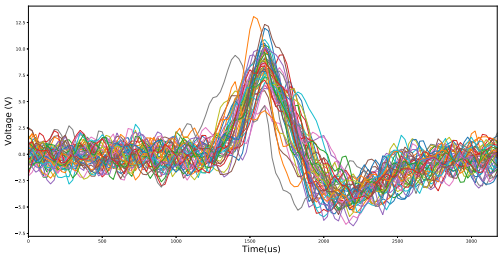
(b) 8 detections classified in Pattern 0



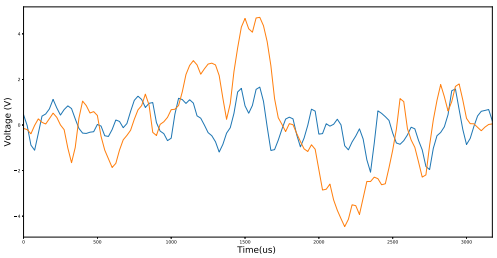
(c) 2 detections classified in Pattern 2



(d) 5 detections classified in Pattern 3



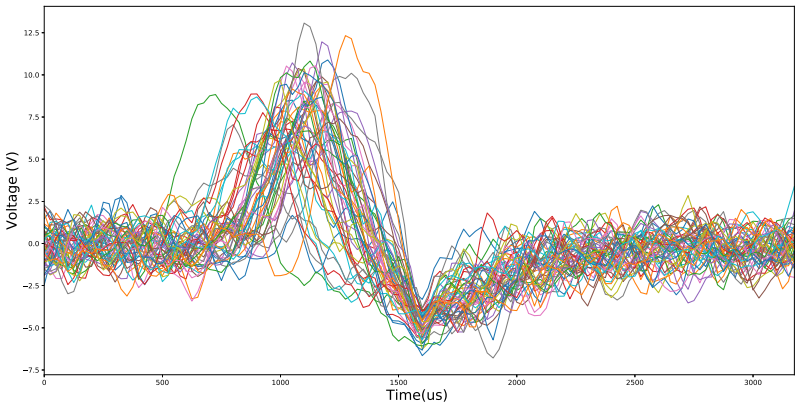
(e) 46 detections classified in Pattern 4



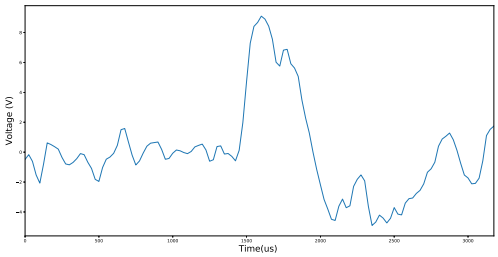
(f) 2 detections classified in Pattern 5

Figure F.3: Action Potentials detected and classified by Correlation Pattern. Threshold value = 13, Correlation Threshold = 10

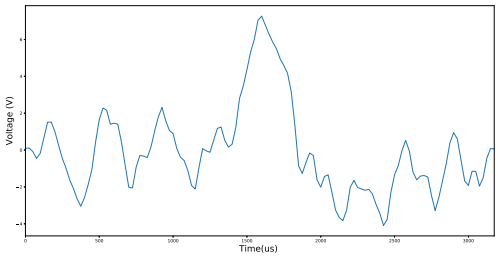
F.4 Threshold value 15



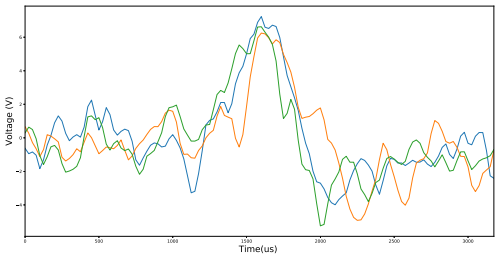
(a) Action Potentials detected. 53 detections



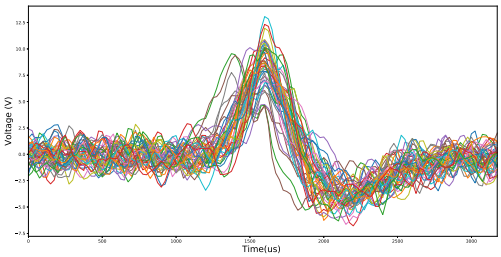
(b) 1 detections classified in Pattern 0



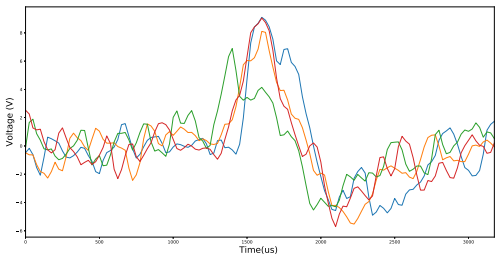
(c) 1 detections classified in Pattern 2



(d) 3 detections classified in Pattern 3



(e) 44 detections classified in Pattern 4



(f) 4 detections classified in Pattern 5

Figure F.4: Action Potentials detected and classified by Correlation Pattern. Threshold value = 15, Correlation Threshold = 10

Bibliography

G. Aldini. Essai Théorique et expérimental sur Le Galvanisme. pages 381–385. 1804.

H. Azami, J. Escudero, A. Darzi, and S. Sanei. Extracellular spike detection from multiple electrode array using novel intelligent filter and ensemble fuzzy decision making. *Journal of Neuroscience Methods*, 239:129–138, 2015. ISSN 1872678X. doi: 10.1016/j.jneumeth.2014.10.006. URL http://dx.doi.org/10.1016/j.jneumeth.2014.10.006.

R. Baravalle, O. A. Rosso, and F. Montani. A path integral approach to the HodgkinHuxley model. *Physica A: Statistical Mechanics and its Applications*, 486: 986–999, 2017. ISSN 03784371. doi: 10.1016/j.physa.2017.06.016. URL http://dx.doi.org/10.1016/j.physa.2017.06.016.

D. Y. Barsakcioglu, Y. Liu, P. Bhunjun, J. Navajas, A. Eftekhar, A. Jackson, R. Quian Quiroga, and T. G. Constandinou. An analogue front-end model for developing neural spike sorting systems. *IEEE Transactions on Biomedical Circuits and Systems*, 8(2): 216–227, 2014. ISSN 19324545. doi: 10.1109/TBCAS.2014.2313087.

A. Belitski, A. Gretton, C. Magri, Y. Murayama, M. A. Montemurro, N. K. Logothetis, and S. Panzeri. Low-Frequency Local Field Potentials and Spikes in Primary Visual Cortex Convey Independent Visual Information. *Journal of Neuroscience*, 28(22): 5696–5709, 2008. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.0009-08.2008. URL http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.0009-08.2008.

E. Biffi, D. Ghezzi, A. Pedrocchi, and G. Ferrigno. Development and validation of a spike detection and classification algorithm aimed at implementation on hardware devices. *Computational Intelligence and Neuroscience*, 2010, 2010. ISSN 16875265. doi: 10.1155/2010/659050.

T. Borghi, A. Bonfanti, G. Zambra, R. Gusmeroli, A. S. Spinelli, G. Baranauskas, M. Iu, and L. Vinci. A Compact Multichannel System for Acquisition and Processing of Neural Signals. *IEEE Conf.*, pages 441–444, 2007.

M. Bresadola. Medicine and science in the life of Luigi Galvani (1737-1798). *Brain Research Bulletin*, 46(5):367–380, 1998. ISSN 03619230. doi: 10.1016/S0361-9230(98)00023-9.

A. Brown. *Nerve Cells and Nervous Systems: An Introduction to Neuroscience*. 2001. ISBN 3540760903. URL https://books.google.com/books?id=jxzhW0ydydwC{&}pgis=1.

G. Buzsáki. *Rhythms of the Brain*. 2009. ISBN 9780199863716. doi: 10.1093/acprof:oso/9780195301069.001.0001.

W. A. Catterall, I. M. Raman, H. P. C. Robinson, and T. J. Sejnowski. The Hodgkin-Huxley Heritage : From Channels to Circuits. 32(41):14064–14073, 2012.

H. L. Chan, M. A. Lin, T. Wu, S. T. Lee, Y. T. Tsai, and P. K. Chao. Detection of neuronal spikes using an adaptive threshold based on the max-min spread sorting method. *Journal of Neuroscience Methods*, 172(1):112–121, 2008. ISSN 01650270. doi: 10.1016/j.jneumeth.2008.04.014.

L. J. Colwell and M. P. Brenner. Action potential initiation in the Hodgkin-Huxley model. *PLoS Computational Biology*, 5(1):1–8, 2009. ISSN 1553734X. doi: 10.1371/journal.pcbi.1000265.

Y. Dong and N. Graziane. *Electrophysiological Analysis of Synaptic Transmission*, volume 112. 2016. ISBN 978-1-4939-3273-3. doi: 10.1007/978-1-4939-3274-0. URL http://link.springer.com/10.1007/978-1-4939-3274-0.

D. L. Donoho. De-Noising by Soft-Thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995. ISSN 15579654. doi: 10.1109/18.382009.

S. Dwivedi and A. K. Gogoi. A novel adaptive real-time detection algorithm for an area-efficient CMOS spike detector circuit. *AEU - International Journal of Electronics and Communications*, 88(November 2017):87–97, 2018. ISSN 16180399. doi: 10.1016/j.aeue.2018.02.023. URL https://doi.org/10.1016/j.aeue.2018.02.023.

G. Finkelstein. Emil du Bois-Reymond vs Ludimar Hermann. *Comptes Rendus - Biologies*, 329(5-6):340–347, 2006. ISSN 16310691. doi: 10.1016/j.crvi.2006.03.005.

L. Galvani. De viribus electricitatis in motu musculari commentarius. In *De Bononiensi Scientiarum et Artium Instituto atque Academia commentarii*, pages 64–65. 1791.

J. Gardner. A history of deep brain stimulation: Technological innovation and the role of clinical assessment tools. *Social Studies of Science*, 43(5):707–728, 2013. ISSN 03063127. doi: 10.1177/0306312713483678.

S. Gibson, J. Judy, and D. Markovic. Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction. *Neural Systems and ...*, 18(5):469–478, 2010. URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=5477171.

S. Gibson, J. W. Judy, and D. Marković. An FPGA-based platform for accelerated offline spike sorting. *Journal of Neuroscience Methods*, 215(1):1–11, 2013. ISSN 01650270. doi: 10.1016/j.jneumeth.2013.01.026.

R. Harrison. A low-power integrated circuit for adaptive detection of action potentials in noisy signals. *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No.03CH37439)*, 4:3325–3328, 2003. ISSN 1094-687X. doi: 10.1109/IEMBS.2003.1280856. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1280856.

C. D. Harvey, F. Collman, D. a. Dombeck, and D. W. Tank. Intracellular dynamics of hippocampal place cells during virtual navigation. *October*, 461(7266):941–946, 2010. doi: 10.1038/nature08499.Intracellular.

S. Herculano-Houzel. *Brain Evolution*. 2012. ISBN 9780123742360. doi: 10.1016/B978-0-12-374236-0.10001-X.

S. Hiseni, C. Sawigun, W. Ngamkham, and W. A. Serdijn. A compact, nano-power CMOS action potential detector. *2009 IEEE Biomedical Circuits and Systems Conference, BioCAS 2009*, (December):97–100, 2009. doi: 10.1109/BIOCAS.2009.5372074.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bulletin of Mathematical Biology*, 52(1-2):25–71, 1990. ISSN 00928240. doi: 10.1007/BF02459568.

T. Horiuchi, T. Swindell, D. Sander, and P. Abshier. A low-power CMOS neural amplifier with amplitude measurements for spike sorting. *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, 4:4–7, 2004. ISSN 078038251X. doi: 10.1109/ISCAS.2004.1328932.

W. J. Hwang, S. H. Wang, and Y. T. Hsu. Spike detection based on normalized correlation with automatic template generation. *Sensors (Switzerland)*, 14(6):11049–11069, 2014. ISSN 14248220. doi: 10.3390/s140611049.

B. Kolb and I. Q. Whishaw. *An Introduction to Brain and-Behavior*. Fourth edition, 2014. ISBN 9781429242288.

F. Lebreton, A. Pirog, I. Belouah, D. Bosco, T. Berney, P. Meda, Y. Bornat, B. Catargi, S. Renaud, M. Raoux, and J. Lang. Slow potentials encode intercellular coupling and insulin demand in pancreatic beta cells. *Diabetologia*, 58(6):1291–1299, 2015. ISSN 14320428. doi: 10.1007/s00125-015-3558-z.

A. K. Lee, I. D. Manns, B. Sakmann, and M. Brecht. Whole-Cell Recordings in Freely Moving Rats. *Neuron*, 51(4):399–407, 2006. ISSN 08966273. doi: 10.1016/j.neuron.2006.07.004.

M. S. Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network (Bristol, England)*, 9(4):R53–78, 1998. ISSN 0954-898X. doi: 10.1088/0954-898X/9/4/001. URL http://www.ncbi.nlm.nih.gov/pubmed/10221571.

F. Lieb, H. G. Stark, and C. Thielemann. A stationary wavelet transform and a time-frequency based spike detection algorithm for extracellular recorded data. *Journal of Neural Engineering*, 14(3), 2017. ISSN 17412552. doi: 10.1088/1741-2552/aa654b.

K. A. Ludwig, J. D. Uram, J. Yang, D. C. Martin, and D. R. Kipke. Chronic neural recordings using silicon microelectrode arrays electrochemically deposited with a poly(3,4-ethylenedioxythiophene) (PEDOT) film. *Journal of Neural Engineering*, 3 (1):59–70, 2006. ISSN 17412560. doi: 10.1088/1741-2560/3/1/007.

A. Maccione, M. Gandolfo, P. Massobrio, A. Novellino, S. Martinoia, and M. Chiappalone. A novel algorithm for precise identification of spikes in extracellularly recorded neuronal signals. 177:241–249, 2009. doi: 10.1016/j.jneumeth.2008.09.026.

M. Mayer, O. Arrizabalaga, F. Lieb, M. Ciba, S. Ritter, and C. Thielemann. Electrophysiological investigation of human embryonic stem cell derived neurospheres using a novel spike detection algorithm. *Biosensors and Bioelectronics*, 100(September 2017):462–468, 2018. ISSN 18734235. doi: 10.1016/j.bios.2017.09.034. URL http://dx.doi.org/10.1016/j.bios.2017.09.034.

S. Mukhopadhyay and G. C. Ray. A New Interpretation of Nonlinear Energy Operator and Its Efficacy in Spike Detection. 45(2):180–187, 1998.

F. A. Navarro. Archives Italiennes de biologie. *Panacea*, 14(38):279, 2013. ISSN 15371964. doi: 10.12871/000398292017126.

A. K. Ng, K. K. Ang, C. Guan, and S. Member. Automatic Selection of Neuronal Spike Detection Threshold via Smoothed Teager Energy Histogram. pages 6–8, 2013.

M. E. J. Obien, K. Deligkaris, T. Bullmann, and D. J. Bakkum. Revealing neuronal function through microelectrode array recordings. 8(January):1–30, 2015. doi: 10.3389/fnins.2014.00423.

A. Parent. Giovanni Aldini: From animal electricity to human brain stimulation. *Canadian Journal of Neurological Sciences*, 31(4):576–584, 2004. ISSN 03171671. doi: 10.1017/S0317167100003851.

J. M. Pearce. Emil Heinrich Du Bois-Reymond (1818-96). *Journal of neurology, neurosurgery, and psychiatry*, 71(5):620, 2001. ISSN 00223050. doi: 10.1136/jnnp.71.5.620.

C. Pedreira, J. Martinez, M. J. Ison, and R. Quian Quiroga. How many neurons can we see with current spike sorting algorithms? *Journal of Neuroscience Methods*, 211(1):58–65, 2012. ISSN 01650270. doi: 10.1016/j.jneumeth.2012.07.010.

Y. Perelman and R. Ginosar. An integrated system for multichannel neuronal recording with spike/LFP separation, integrated A/D conversion and threshold detection. *IEEE Transactions on Biomedical Engineering*, 54(1):130–137, 2007. ISSN 00189294. doi: 10.1109/TBME.2006.883732.

M. Piccolino. Animal electricity and the birth of electrophysiology: The legacy of Luigi Galvani. *Brain Research Bulletin*, 46(5):381–407, 1998. ISSN 03619230. doi: 10.1016/S0361-9230(98)00026-4.

M. Piccolino. Visual images in Luigi Galvani's path to animal electricity. *Journal of the History of the Neurosciences*, 17(3):335–348, 2008. ISSN 0964704X. doi: 10.1080/09647040701420198.

M. Piccolino and N. J. Wade. Carlo Matteucci (1811-1868), the "frogs pile", and the Risorgimento of electrophysiology. *Cortex*, 48(6):645–646, 2012. ISSN 19738102. doi: 10.1016/j.cortex.2011.08.002. URL http://dx.doi.org/10.1016/j.cortex.2011.08.002.

A. Pirog, Y. Bornat, S. Renaud, R. Perrier, M. Jaffredo, M. Raoux, and J. Lang. A versatile electrode sorting module for MEAs: implementation in a FPGA-based real-time system. 15(19):33400, 2015.

J. Platkiewicz and R. Brette. A threshold equation for action potential initiation. *PLoS Computational Biology*, 6(7):25, 2010. ISSN 1553734X. doi: 10.1371/journal.pcbi.1000850.

Pong P.chu. *RTL HARDWARE DESIGN USING VHDL*. 2006. ISBN 9780471720928.

A. Quotb, Y. Bornat, and S. Renaud. Wavelet Transform for Real-Time Detection of Action Potentials in Neural Signals. *Frontiers in Neuroengineering*, 4(July):1–10, 2011. ISSN 1662-6443. doi: 10.3389/fneng.2011.00007. URL http://journal.frontiersin.org/article/10.3389/fneng.2011.00007/abstract.

F. Rummens. Systmes intégrés pour l'hybridation vivant-artificiel : modélisation et conception d'une chaine de détection analogique adaptative. Sciences de l'ingénieur [physics]. Université de Bordeaux, 2015. 2015. URL http://www.theses.fr/2015BORD0431.

F. Rummens, S. Renaud, and N. Lewis. CMOS differential neural amplifier with high input impedance. *Conference Proceedings - 13th IEEE International NEW Circuits and Systems Conference, NEWCAS 2015*, pages 8–11, 2015. doi: 10.1109/NEWCAS.2015.7182037.

A. Rushton. *VHDL FOR LOGIC SYNTHESIS SYNTHESIS Third Edition.* Third edition, 2011. ISBN 9780470688472.

U. Rutishauser, E. M. Schuman, and A. N. Mamelak. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of Neuroscience Methods*, 154(1-2):204–224, 2006. ISSN 01650270. doi: 10.1016/j.jneumeth.2005.12.033.

M. Saeed, A. A. Khan, and A. M. Kamboh. Comparison of Classifier Architectures for Online Neural Spike Sorting. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(4):334–344, 2017. ISSN 15344320. doi: 10.1109/TNSRE.2016.2641499.

H. Semmaoui, J. Drolet, A. Lakhssassi, and M. Sawan. Setting adaptive spike detection threshold for smoothed TEO based on robust statistics theory. *IEEE Transactions on Biomedical Engineering*, 59(2):474–482, 2012. ISSN 00189294. doi: 10.1109/TBME.2011.2174992.

W. Strauss. *Digital signal processing*, volume 17. 2000. ISBN 1053-5888. doi: 10.1109/79.826412. URL http://ieeexplore.ieee.org/document/826412/.

T. Takekawa, K. Ota, M. Murayama, and T. Fukai. Spike detection from noisy neural data in linear-probe recordings. *European Journal of Neuroscience*, 39(11):1943–1950, 2014. ISSN 14609568. doi: 10.1111/ejn.12614.

W. Wolfgang. *Patch-Clamp Analysis*. 2007. ISBN 9781597454926.

T. Wu, J. Xu, Y. Lian, A. Khalili, A. Rastegarnia, C. Guan, and Z. Yang. A 16-Channel Nonparametric Spike Detection ASIC Based on EC-PC Decomposition. *IEEE Transactions on Biomedical Circuits and Systems*, 10(1):3–17, 2016. ISSN 19324545. doi: 10.1109/TBCAS.2015.2389266.

Y. Yang and A. J. Mason. Hardware Efficient Automatic Thresholding for NEO-Based Neural Spike Detection. *IEEE Transactions on Biomedical Engineering*, 64(4):826–833, 2017. ISSN 15582531. doi: 10.1109/TBME.2016.2580319.

Y. Yang, C. S. Boling, A. M. Kamboh, and A. J. Mason. Adaptive Threshold Neural Spike Detector Using Stationary Wavelet Transform in CMOS. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(6):946–955, 2015. ISSN 1534-4320. doi: 10.1109/TNSRE.2015.2425736. URL http://ieeexplore.ieee.org/document/7101292/{%}0Ahttp://ieeexplore.ieee.org/ielx7/7333/7321076/07101292.pdf?tp={&}arnumber=7101292{&}isnumber=7321076.

Z. Yang, W. Liu, M. R. Keshtkaran, Y. Zhou, J. Xu, V. Pikov, C. Guan, and Y. Lian. A new EC-PC threshold estimation method for in vivo neural spike detection. *Journal of Neural Engineering*, 9(4), 2012. ISSN 17412560. doi: 10.1088/1741-2560/9/4/046017.

Y. Yuan, C. Yang, and J. Si. The M-Sorter: An automatic and robust spike detection and classification system. *Journal of Neuroscience Methods*, 210(2):281–281, 2012. ISSN 01650270. doi: 10.1016/j.jneumeth.2012.07.012. URL http://dx.doi.org/10.1016/j.jneumeth.2012.07.012.