Biochip Architecture for Real-Time Cardiac Pathologies Identification

Pedro Pontes Brito de Sá

Thesis to obtain the Master of Science Degree in Electrical and Computer Engineering

Supervisors: Prof. Dr. Pedro Filipe Zeferino Aidos Tomás
Prof. Dr. Nuno Filipe Valentim Roma
Dr. Helena Isabel Aidos Lopes

Examination Committee

Chairperson: Prof. Dr. António Manuel Raminhos Cordeiro Grilo
Supervisor: Prof. Dr. Pedro Filipe Zeferino Aidos Tomás
Member of the Committee: Prof. Dr. Paulo Ferreira Godinho Flores

November 2017
In loving memory of my grandfathers Armando and Luís.
Acknowledgments

Firstly, I would like to thank Instituto Superior Técnico for 5 years of excellent education and personal growth opportunities. Nonetheless, this work would not have been possible without the guidance of Profs. Pedro Tomás, Nuno Roma and Helena Aidos, who always had an open door to help me, even during the weekends.

A special thank you to André Lourenço and the CardioID Technologies team who introduced me to the topic of ECG processing during my internship, and with whom was such a pleasure to work.

To my colleagues and dear friends, Francisco Teles and Rui Brás, who always kept me on track and never let me go down even in the hardest moments, thank you.

For the help with the anatomic illustrations and constant support every day of the year, I deeply thank my friend, partner and girlfriend Susana Raposo.

Last, and most importantly, my everlasting gratitude to my mother and father, Ana and Armando, for the constant struggle and sacrifice to provide me the best opportunities in life and to whom I owe everything that I am. Any possible acknowledgement of this comes up short.
Resumo

O electrocardiograma (ECGs) é um método crítico na detecção de patologias cardíacas potencialmente mortíferas. No entanto, é invasivo e requer vários sensores, o que torna a monitorização desconfortável no dia-a-dia. Em electrocardiografia não invasiva tem havido estudo da Derivação 1 do ECG a partir das mãos ou braços. A maioria dos algoritmos recentes de detecção destas patologias recorrem a mais do que uma derivação do ECG. Como são testados contra bases de dados standard em contextos clínicos, são susceptíveis a condições de aquisição ruidosas. O objectivo deste trabalho consistiu em projectar um sistema de detecção automática de patologia cardíaca através da Derivação 1 do ECG em tempo real, e implementá-lo numa placa de desenvolvimento FPGA. Para isto, considerou-se investigação em desenvolvimento de algoritmos e arquiteturas de processamento de ECG. O sistema proposto baseia-se num fluxo de processamento com os seguintes passos: remoção de ruído; detecção e segmentação de batimentos cardíacos; extracção de características dinâmicas e morfológicas; e classificação do batimento (normal ou anormal (e respectiva classe)). Para a base de dados InCarTDb, a implementação de software deste fluxo resultou num modelo de 4 classes com uma precisão de 96.5%. Tendo este modelo como base, uma arquitectura foi projectada e simulada em VHDL, e mapeada virtualmente para uma placa de desenvolvimento FPGA. A arquitectura resultante verificou precisões entre 84 a 100% para diferentes paciente, menos de 30% de utilização dos recursos da placa de teste, uma frequência de relógio de 35 MHz e um consumo total de potência de 192 mW.

Palavras Chave

Electrocardiograma, Derivação 1, Patologia Cardíaca, Arquitectura Digital, Processamento de Sinal, FPGA
Abstract

Electrocardiograms (ECGs) are crucial in the identification of life-threatening conditions. However, most heart monitoring methods are invasive and require several sensors, making heart monitoring uncomfortable in daily life situations. For off-the-person electrocardiography, the Lead 1 from a person’s hands or arms is undergoing study. Although many heart disease algorithms have been described in recent years, all of them make use of more than one ECG lead and, by being tested only against standard databases, are sensitive to the acquisition conditions. The purpose of this work was to design a robust Lead 1-only real-time heart disease detection system and implement it on an FPGA-based development board. To achieve this objective, the proposed work considers recent research on the development of ECG processing algorithms and architectures. Hence, the proposed system is based on a signal processing pipeline composed of: ECG signal denoising; heartbeat detection and segmentation; extraction of dynamic morphological features; and heartbeat classification (normal or abnormal and corresponding condition). Resorting to the InCarTDb Database, the proposed methodology resulted in a 4-class model with a classification of accuracy up to 96.5%. Considering this model as a starting point, a biochip architecture for cardiac pathologies identification was designed and simulated in VHDL, and mapped to a virtual FPGA evaluation board. The resulting architecture verified accuracies from 84 to 100% for different patients of the database, less than 30% of the resources of the evaluation board, a working clock frequency of 35 MHz and a total power consumption of 192 mW.

Keywords

Single-Lead Electrocardiogram, Heart Disease Identification, Signal Denoising, Feature Extraction, Signal Processing Architecture, FPGA
Contents

1 Introduction 1
   1.1 Problem Statement ...................................................... 1
   1.2 Objectives ..................................................................... 2
   1.3 Thesis Contributions ..................................................... 2
   1.4 Thesis Outline ................................................................. 4

2 Anatomy and the Clinical ECG 5
   2.1 Heart Anatomy and Physiology ........................................ 5
   2.2 Heart Diseases ................................................................. 8
      2.2.1 Rhythm Disturbances ................................................... 8
      2.2.2 Atrial Ectopy and Fibrillation ...................................... 9
      2.2.3 Ventricular Ectopy and Fibrillation ............................... 10
   2.3 Summary ................................................................... 12

3 Overview of ECG Processing Methods 13
   3.1 Heart Diseases Detection ................................................ 13
      3.1.1 Pre-Processing............................................................. 14
         3.1.1.A Noise Filtering and Artifact Removal ....................... 15
      3.1.2 QRS-Complex Segmentation ...................................... 16
         3.1.2.A Modified Engelse and Zeelenberg (Engzee) ........... 16
         3.1.2.B ECG Slope Sum Function (ESSF) ......................... 18
         3.1.2.C Zero Crossing Counts ......................................... 18
      3.1.3 ECG Delineation and Feature Extraction ..................... 20
         3.1.3.A P-QRS-T Waves Detection ................................... 20
         3.1.3.B Automatic features from the Discrete Wavelet Transform (DWT) ......................................................... 21
         3.1.3.C Mixed methodologies ........................................... 22
      3.1.4 Heart Diseases Detection ........................................... 23
         3.1.4.A Probabilistic Neural Networks ............................... 24
      3.2 Hardware Implementations of ECG Processing Systems .... 25
         3.2.1 FPGA-based Design ............................................... 26
            3.2.1.A FPGA Structural Elements ................................ 26
            3.2.1.B FIR/IIR Filter Implementations .......................... 28
3.2.1.C DWT Implementations ................................................. 29
3.2.1.D PNN Implementation ................................................ 30
3.2.2 FPGA-based ECG processing systems ............................. 32
3.3 Summary ................................................................. 33

4 ECG Processing Software Implementation ......................... 34
4.1 Overview ............................................................... 34
4.2 Pre-Processing .......................................................... 35
  4.2.1 Baseline Filtering ..................................................... 36
  4.2.2 Low-pass Filtering .................................................... 36
  4.2.3 Signal Normalization ................................................ 38
4.3 Features Extraction ..................................................... 39
  4.3.1 Dynamic Features ..................................................... 39
  4.3.2 Heartbeat Resampling and Pruning ............................... 41
  4.3.3 Morphological Features ............................................. 42
4.4 Classification ............................................................ 43
4.5 Summary ................................................................. 43

5 ECG Processing Hardware Architecture ............................... 44
5.1 Overview ............................................................... 44
  5.1.1 Control Unit and Datapath ......................................... 46
5.2 Data I/O ................................................................. 48
5.3 Pre-Processing .......................................................... 48
  5.3.1 Baseline Filtering ..................................................... 48
  5.3.2 Low-pass Filtering .................................................... 50
5.4 Features Extraction ..................................................... 51
  5.4.1 Peak Detection and Signal Normalization ......................... 52
  5.4.2 Template Matching ................................................... 53
  5.4.3 Dynamic Features .................................................... 54
  5.4.4 Resampling ........................................................... 55
  5.4.5 Morphological Features ............................................. 56
  5.4.6 Principal Component Analysis .................................... 58
5.5 Classification ............................................................ 59
5.6 Time Constraints ......................................................... 61
5.7 Summary ................................................................. 62

6 Experimental Results ....................................................... 63
6.1 Databases and Evaluation Standards ................................. 63
6.2 Evaluation of the heart disease identification procedure .......... 66
  6.2.1 Pre-Processing ......................................................... 66
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The dipole field due to current flow in a myocardial cell at the advancing front of depolarization. Vm is the transmembrane potential.</td>
</tr>
<tr>
<td>2.2</td>
<td>Trajectory of a normal cardiac vector.</td>
</tr>
<tr>
<td>2.3</td>
<td>Standard anatomic placement of the 10 sensors array for the 12-Lead ECG. For the picture on the left, two common positions of each limb sensor are represented with the same color, but only one is required in each setting.</td>
</tr>
<tr>
<td>2.4</td>
<td>Normal features of the Lead 1 Electrocardiogram (ECG).</td>
</tr>
<tr>
<td>2.5</td>
<td>Healthy Lead 1 ECG signal. Heart rate = 64 bpm.</td>
</tr>
<tr>
<td>2.6</td>
<td>Representation of bradycardia: 46 beats per minute. Lead 1.</td>
</tr>
<tr>
<td>2.7</td>
<td>Representation of tachycardia: 103 beats per minute.</td>
</tr>
<tr>
<td>2.8</td>
<td>Atrial premature contractions (3rd and 13th beats) in Lead 1.</td>
</tr>
<tr>
<td>2.9</td>
<td>A Lead 1 example of atrial fibrillation.</td>
</tr>
<tr>
<td>2.10</td>
<td>Wavefront trajectory in a ventricular premature contraction.</td>
</tr>
<tr>
<td>2.11</td>
<td>Ventricular premature contraction (5th beat) in Lead 1.</td>
</tr>
<tr>
<td>2.12</td>
<td>Two episodes of non-sustained Ventricular Tachycardia (VT), one comprehending the first 4 Ventricular Premature Contraction (VPC) detected, the other with the last 4.</td>
</tr>
<tr>
<td>2.13</td>
<td>An example of ventricular fibrillation in a Lead 2 ECG.</td>
</tr>
<tr>
<td>3.1</td>
<td>ECG signal processing algorithmic workflow.</td>
</tr>
<tr>
<td>3.2</td>
<td>Noisy electrocardiograms: (a) Instrumentation saturation; (b) powerline interference; (c) electromyographic artifacts.</td>
</tr>
<tr>
<td>3.3</td>
<td>Lourenço et al. adaptation of the Engelse and Zeelenberg algorithm for QRS detection algorithms In blue the original signal, ( x[n] ), and in red ( y_2[n] ).</td>
</tr>
<tr>
<td>3.4</td>
<td>Zero crossing counts algorithm block diagram.</td>
</tr>
<tr>
<td>3.5</td>
<td>Mother Wavelets: (a) First derivative of the Gaussian Function, (b) Daubechie’s 4th order wavelet function (DB4) function, (c) Mexican Hat function.</td>
</tr>
<tr>
<td>3.6</td>
<td>Probabilistic Neural Network functional structure.</td>
</tr>
<tr>
<td>3.7</td>
<td>a) Basic FPGA architecture layout; b) Basic LUT functional architecture; c) Basic FF layout.</td>
</tr>
<tr>
<td>3.8</td>
<td>DSP48 functional architecture.</td>
</tr>
<tr>
<td>3.9</td>
<td>2nd Order Digital Filter in a) Direct-Form 1; b) Direct-Form 2.</td>
</tr>
</tbody>
</table>
List of Tables

6.1 Superclasses and classes of heartbeats recommended for detection by AANSI. . . . . . 64
6.2 Classes of heartbeats and number of occurrences annotated in InCarTDb. . . . . . . 65
6.3 Classes of heartbeats and number of occurrences annotated in InCarTDb after norm-
malization and amplitude thresholding. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
6.4 Average accuracy of the classifier for different values of the baseline filter window size
and the IIR filter cutoff frequency. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
6.5 Average accuracy of the classifier for different orders of the IIR filter. . . . . . . . . 67
6.6 Average accuracy of the classifier with and without the normalization stage . . . . . . . 67
6.7 Average accuracy of the classifier with and without the resampling stage. . . . . . . . 68
6.8 Accuracy average of the classifier while varying both the number of recordings and the
number of samples of each class in the training set used in classification. . . . . . . . . . . 70
6.9 Evaluation of the AAMI parameters for a 4-class classification model. . . . . . . . . . 71
6.10 Average absolute maximum and mean squared errors of format conversions from dou-
ble to 8, 16 and 32 bits for all the 75 InCarTDb recordings. . . . . . . . . . . . . . . 73
6.11 FPGA resources utilization table for the architecture with classifiers of 4, 5, 6, 8 and 16
classes. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
6.12 FPGA static and dynamic power consumption table for the architecture with classifiers
of 4, 5, 6, 8 and 16 classes. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
6.13 FPGA static and dynamic power consumption table for the architecture with different
clock frequencies. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
6.14 Segmentation results of the final architecture for the selected recordings. . . . . . . . 78
6.15 Classification results of the final architecture for the selected recordings. . . . . . . . 79
A.1 Discriminated heartbeats occurrences for each patient after the normalization and prun-
ing. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . A-2
List of Algorithms

1 Signal Normalization Algorithm .................................................. 38
2 Full ECG Processing and Heart Beat Classification ...................... 43
Abbreviations

**AAMI**  Association for the Advancement in Medical Instrumentation

**ADC**  Analog-to-Digital Converter

**ALU**  Arithmetic Logic Unit

**APC**  Atrial Premature Contraction

**ASIC**  Application Specific Integrated Circuit

**BRAM**  Block Random Access Memory

**CLB**  Configurable Logic Blocks

**CWT**  Continuous Wavelet Transform

**DB4**  Daubechie’s 4th order wavelet function

**DB8**  Daubechie’s 8th order wavelet function

**DER**  Detection Error Rate

**DF1**  Direct-Form 1

**DF2**  Direct-Form 2

**DSP**  Digital Signal Processing

**DSP48**  48-bit Digital Signal Processors

**DWT**  Discrete Wavelet Transform

**ECG**  Electrocardiogram

**FF**  Flip-Flop

**FIR**  Finite Impulse Response

**FN**  False Negatives

**FP**  False Positives

**Fs**  Sampling Frequency
FPGA  Field Programmable Gate Array
FPR  False Positive Rate
IC  Independent Components
ICA  Independent Component Analysis
IIR  Infinite Impulse Response
InCarTDb  St.-Petersburg Institute of Cardiological Technics 12-lead Arrhythmia Database
I/O  Input/Output
LDA  Fisher's Linear Discriminant Analysis
LUT  Look-Up Table
MIT-BIH  The Massachusetts Institute of Technology - Beth Israel Hospital Arrhythmia Database
NEB  Nodal Ectopic Beat
PCA  Principal Component Analysis
PLD  Programmable Logic Device
PLL  Phase-Locked Loop
PNN  Probabilistic Neural Network
PPR  Positive Predictivity Rate
RAM  Random Access Memory
RBBB  Right Bundle Branch Block
ROM  Read-Only Memory
SA  Sino-Atrial
SEN  Sensitivity
SVM  Support Vector Machine
TN  True Negatives
TP  True Positives
VFN  Ventricular Fusion Beat
VHDL  VHSIC (Very High Speed Integrated Circuits) Hardware Description Language
VPC  Ventricular Premature Contraction
VT  Ventricular Tachycardia
Introduction

Cardiovascular diseases account for the death of more than 17.3 million people per year around the world. By 2030, this number is expected to have increased to 23.6 million [7]. Since its development, electrocardiography has been used as a diagnostic tool for multiple heart conditions, being most effective in the detection of arrhythmias and myocardial infarctions [8].

Early diagnosis of heart conditions can be life-saving, and the advent of computational systems turned the automatic analysis of the Electrocardiogram (ECG) into a trending topic in the scientific community over the last thirty years [9]. This may be very relevant in some types of collective transportation, where the driver is responsible for passenger safety. Professional long distance athletes can also benefit from this kind of monitoring. In these cases, a quick response to an acute episode of heart disease can help prevent avoidable deaths and injuries.

1.1 Problem Statement

Several electrocardiography devices are currently available on the market, ranging from high-end hospital equipment to portable domestic appliances. The acquisition methods vary with the device, although the most common practice is the 12-lead ECG which relies on an array of ten adhesive sensors on the subject’s torso and limbs [1]. Although there is redundancy in this setup, it prevents inconsistencies and provides a full spatial overview of the heart’s electrical behavior. However, it makes prolonged monitoring impractical for a lay person such as a bus driver. Nonetheless, it is possible to measure one view of the ECG by taking the electrical potential with a contact based sensor on each of the subject’s hands [10, 11]. This single-lead view, although not as complete as the 12-lead view, still provides relevant information on the subject’s biometrics and pathologies detectable through the ECG’s Lead I [10]. The leads positioning will be further elaborated in Section 2.1.
Aside from the non-intrusiveness and easiness of use, contact based single-lead acquisitions also imply a smaller amount of data \[10\]; therefore, the required computational and processing power of the system decreases as well. Such a setting allows for the current embedded computational platforms to be regarded as a possibility for ECG processing in real-time environments. Although there are authors already implementing ECG processing algorithms, for example, on Field Programmable Gate Arrays (FPGAs) \[6, 12–17\], there is still work to be done towards a reliable automatic real-time pathology detection embedded platform.

The problem itself branches in two: evaluation and testing of state-of-the-art pathology detection algorithms, taking into account the limitations of single-lead analysis for this case; development of a computationally efficient, effective, dependable and scalable architecture which implements the pathology detection algorithms.

### 1.2 Objectives

The main objective of this work is to have an architecture of a real-time single-lead heart disease detection system designed and tested against standard databases and signals acquired in real-time settings.

To attain this goal, firstly the database options for the problem at hand should be analyzed. Then the efficiency and efficacy of the commonly used methods of disease detection in ECGs should be assessed. This requires several steps to be considered, such as signal filtering and denoising, heart-beat identification and segmentation, feature extraction from the segmented heartbeat, and finally, classification.

The last stage is to plan, design and implement an architecture of the system designed, while bearing in mind the following points: the architecture should require as little computational resources as possible for it to be implementable in a device of lower complexity; it should process in real-time processing with as little delays as possible, while keeping the clock frequency at the minimum necessary; by optimizing the previous two points, the power is automatically decreased, and having a small power consumption is a priority; it should allow the usage of several different settings for every one of its stages, which means customization for different acquisition setups and processing algorithms.

### 1.3 Thesis Contributions

This dissertation follows the tendency of off-the-person electrocardiography and is the first end-to-end architecture specially designed for Lead-1 real-time processing, with computational and energetic efficiencies which enable the usage in wearable applications. In this setting, it lays the ground work for real-time Lead 1-only heart disease detection, in particular with the St.-Petersburg Institute of Cardiological Technics 12-lead Arrhythmia Database (InCarTDb). A whole system was implemented taking the 75 Lead 1 annotated recordings of this database as a starting point, although it can be applied to any other set of signals. The system developed in software considered the following steps:
The signal denoising and artifact removal methods were assessed and the pre-processing considered baseline removal by subtracting a moving average window and low-pass filtering with an Infinite Impulse Response (IIR) filter;

The filtered signal then has its amplitude normalized by dividing the samples of each consecutive window by the a normalization parameter calculated from the maximum of the previous window;

From the annotated peaks location, a window of 250 ms before each peak and 375 ms afterwards is taken;

The dynamic features of each peak are derived from the RR interval, and the segmented heartbeat is downsampled if the heart rate is below 60 bpm and upsampled if it is above 133 bpm;

The morphological features of the heartbeat are obtained from the analysis coefficients of the Discrete Wavelet Transform (DWT) and Principal Component Analysis (PCA) is applied only to these features for dimensionality reduction, being then concatenated with the dynamic features;

The recordings are randomly separated between a training and a test set and the features are clustered, having the training cluster being fed into a Probabilistic Neural Network (PNN) classifier, and the testing cluster verifying its performance.

Classification of 4, 5 and 6 different classes were attempted, but due to the lack of occurrences of the 5th and 6th classes and the poor results they provided, the 4-class classifier was set as final. Verifying an overall accuracy of over 96.5% in a mostly inter-patient paradigm, it is on par with the multiple-lead 4-class classifier described by Jannah et al [18].

These stages were then projected onto a multiple-cycle hardware architecture, considering them independent but interconnected blocks. All these blocks have their own Control Unit state-machine, and most of them have also a Datapath sub-unit to aid in the arithmetic operations. However, instead of having the annotated peak location, two heartbeat detection stages were implemented. One considered a fixed amplitude threshold for the normalized signal, above which a peak occurrence would be considered. The other took a cross-correlation of a fixed size window of the normalized signal with a template heartbeat. In the particular case of the InCarTDb this was necessary to identify a pathology class that had no positive peak. The architecture is also scalable and programmable, which allows it to be tailored to: different acquisition setups; different parameters in most of the filtering, segmentation, feature extraction and classification stages; different abnormalities to be classified; different leads and even more than one lead simultaneously.

The final design of the architecture was then mapped to the virtual resources of a FPGA development board verified very satisfactory results. The signal classification verified accuracies of the correctly segmented heartbeats between 84 to 100% in some patients. The resources utilized does not reach 30% of the available resources of the board, the clock frequency can go up to 35 MHz and the power consumption can be as low as 122 mW.
Although being a prototype and having its limitations, this architecture establishes important ground work for the future development or integration of heart disease monitoring solutions into low-power and computationally simple environments.

1.4 Thesis Outline

The remaining content of this work is divided in six chapters. An introduction to the anatomy and physiology of the heart, as well as the clinical introduction to electrocardiography, are presented in Chapter 2. In Chapter 3, a selection of the most relevant ECG processing algorithms for the stages defined in Section 1.2 are presented, as well as the hardware architectures that implement them and an overview of some recent ECG processing systems. The final methodologies and algorithms used to provide the whole classification model are defined in Chapter 4, as well as some of the reasoning behind the decisions made. Chapter 5 contains both a global overview and a description of each stage of the implemented hardware architecture. The results of both the software and hardware implementations are presented and discussed in Chapter 6, whereas the conclusions of this work are in Chapter 7.
2 Anatomy and the Clinical ECG

2.1 Heart Anatomy and Physiology

The heart is a complex muscular organ that drives the circulation of blood when contracts. Every normal heartbeat is preceded by an electrical current impulse that leads to the heart contraction. Originating at a specific area of the heart, this impulse has a defined propagation path which ensures the correct coordination of the heart's contraction, also known as *systole*. The ECG consists in the filtered acquisition of this electrical signal. As stated in the previous section, ECGs have been used for diagnosis of multiple pathologies for several decades now. This is due to the fact that the ECG is affected by structural and metabolic abnormalities of the heart. The former includes problems with the conducting fibers and the latter are mostly *ischemias* (lack of oxygen in the tissues due to blood restriction). Some geometry-related abnormalities such as the orientation of the heart over a different axis can be the cause of changes in the ECG. Due to the heart's vital role, most medical evaluations includes an ECG analysis, especially since it can be performed through noninvasive methods. The underlying mechanisms of the ECG are the electrophysiological behavior of the heart cells, the propagation of the electrical wave through the myocardium, how the specific areas of the heart react to it, and why it is possible to do measurements on the body's surface [1]. Due to the context of this work, an overview of these components will be presented in this section.

The simplest way to mathematically represent the relationship between the heart as a generator and the body's surface is through the *dipole model*. The dipole itself comprehends the electrical activity of the heart and the rest of the model describes the geometry and electrical properties of its surroundings. For the electrical activity of the heart, we should first consider that as a potential propagates through a cell in the myocardium, a current is generated towards the propagation's direction. The media outside the cell is conductive; therefore the electrical circuit is closed and the charge is
The heart displays four major cavities, two atria and two ventricles. The heart as a whole functions as a hydraulic pump. The right atrium receives the blood from the whole body and feeds it into the right ventricle. When it contracts, the blood flows to the lungs where it captures oxygen and releases carbon dioxide. The blood then reenters the heart through the left atrium, flows into the left ventricle and is then pumped back to the rest of the body through the aorta. This coordinated contraction results from the propagation of the impulse originated in the cardiac cells of the Sino-Atrial (SA) node, which are called pacemakers due to their ability to generate this impulse automatically. The propagation of the electrical impulse through the heart is known as the cardiac vector. Due to the heart’s uneven geometry and tissue variability, the current propagation in the heart is not linear, as seen in Figure 2.2.

Since the cardiac vector varies in a three-dimensional space, the measurement of this electrical wave depends on the position of the sensors. In most cases, the ECG converts 12 different projections...
of the vector into lines of well-defined orientation. The central terminal is the average of the limb leads (I to III) voltages. Lead I records the potential difference between the left and right arms, Lead II the left leg and right arm, and Lead III the left leg and arm. The leads aVF, aVL and AVR are mathematical derivations of the previous ones. Six leads (V1 to V6) originate from the potential between sensors in specific places on the subject’s thorax and the central terminal. These non-orthogonal projections yield a redundancy level that compensates for inconsistencies in the electrode placement and helps in the analysis of the ECG. The sensor placement is shown in Figure 2.3.

Figure 2.3: Standard anatomic placement of the 10 sensors array for the 12-Lead ECG. For the picture on the left, two common positions of each limb sensor are represented with the same color, but only one is required in each setting.

Although the leads present different perspectives, the normal features of the ECG are well defined, with different waves labeled P, Q, R, S and T (sometimes also U, when this wave is identifiable) [1]. This structure is illustrated in Figure 2.4.

Figure 2.4: Normal features of the Lead 1 ECG [1].
An example of real ECG signal is shown in Figure 2.5.

![Figure 2.5: Healthy Lead 1 ECG signal. Heart rate = 64 bpm.](image)

2.2 Heart Diseases

The surface ECG is subject to some limitations when it comes to the diagnosis of heart disease. Due to the heterogeneity of the body, electrical potentials are not absolutely precise in detailing electrophysiological behavior of the heart. In the 19th century, Hermann von Helmholtz established the impossibility of characterizing a current generator by solely measuring electrical potentials. However, the surface ECG is still valid in the assessment of heart disease, thus being common practice in clinical examinations [1].

2.2.1 Rhythm Disturbances

In general terms, electrical heart conditions can be divided in two categories: bradyarrhythmias and tachyarrhythmias. The former relate to the conditions which make the heart beats to be further apart than normal, or with an irregular frequency. This may jeopardize life due to inability of the heart to pump enough blood for the body to function. The latter relate to those where the heart beats have a frequency far above the desired, which makes the heart to pump without having enough time to fill completely between contractions. This may result in diminished blood flow and organ tissue damage [1].

There are arrhythmias originating from changes in the firing frequency of the SA node. These may be due to physiologic responses to outside stimuli, or outstanding physical condition of the subject. High performance athletes tend to develop a natural sinus bradycardia (see Figure 2.6), where the SA node fires slower than usual, resulting in a lower normal heart rate. This also happens in patients using heart-slowing medication. Sinus tachycardia (see Figure 2.7) may happen as a response to fear or strenuous exercise, where it is a normal physiological reaction, or due to chemical stimulants such as cocaine. Breathing can also be the source of an irregular firing frequency of the SA node, commonly known as sinus arrhythmia, which constitutes a pattern with slower and faster beats [1].
2.2.2 Atrial Ectopy and Fibrillation

Some arrhythmias have a different origin from the ones described in the previous section. Instead of having the SA triggering the depolarization, this happens in another region of the myocardium. This is known as *ectopic depolarization*, which leads to *ectopic beats*. An Atrial Premature Contraction (APC) is an ectopic beat which originates in the atria. These beats are not immediately dangerous, being sometimes part of the regular behavior of the subject’s heart, and are roughly similar to the sinus beats in the ECG. APC examples are shown in Figure 2.8 [1].

Figure 2.6: Representation of bradycardia: 46 beats per minute. Lead 1.

Figure 2.7: Representation of tachycardia: 103 beats per minute.

Figure 2.8: Atrial premature contractions (3rd and 13th beats) in Lead 1.
Fibrillation occurs when the electrical wavefront stops being unified, having instead several smaller wavefronts propagating almost randomly across the myocardium. Atrial fibrillation is fibrillation originating in the atria. This kind of arrhythmia does not present immediate risk as longs as the ventricular response rate does not go too fast. By itself, it just reduces the performance of the heart, since there is not one strong atrial contraction to pump the blood into the ventricles. However, the lack of contractions due to atrial fibrillation leads to blood stasis in the atria, thus enhancing the chance of blood clots formation. If set loose in the blood flow, these clots may cause a stroke. This pathology causes the disappearance of the P wave in the ECG, but the QRS complex is still well visible, as verified in most heartbeats in Figure 2.9.

Figure 2.9: A Lead 1 example of atrial fibrillation.

2.2.3 Ventricular Ectopy and Fibrillation

As the APC, a Ventricular Premature Contraction (VPC) originates in the ventricles. Since the beat is triggered from the geometric opposite of the SA node, the wavefront moves backwards, as illustrated in Figure 2.10. This behavior generates an inverted ECG peak which is easily distinguishable from the regular sinus beats in Figure 2.11.

Figure 2.10: Wavefront trajectory in a ventricular premature contraction.
A VPC is usually followed by a pause where the myocardium remains unpolarized, after which it repolarizes and regains the natural sinus rhythm. In some cases, the system may go back to its natural state although the conditions which triggered the ectopic beat remain the same. However, the subject’s life may be at risk if there is a continuous sequence (or a series of continuous sequences interleaved with a few sinus beats) of VPCs, almost in a stable way. These cyclic patterns are known as reentrant arrhythmias and Figure 2.12 presents an example of one, in this case a Ventricular Tachycardia (VT). If not detected and treated, in most cases these states may be fatal within minutes or even seconds. This is due to the incompatibility of the depolarization frequency with effective cardiac pumping. To assess the presence of VT in an ECG, three consecutive VPCs with a heart rate over 100 bpm must be detected. If the VT persists over a 30 seconds period, it is considered to be a sustained VT. Otherwise, it is non-sustained [1].

One of the most severe pathologies is the ventricular fibrillation. Since heart contractions stop being coordinated, the pumping function is interrupted (circulatory arrest), leading to death in minutes, if not seconds. These are easily detected through ECG analysis due to its chaotic oscillatory behavior, as seen in the example of Figure 2.13 [1].
2.3 Summary

The heart is a complex system and, as such, it can run into several different problems. There are many other heart diseases that directly affect the ECG which are not going to be described in this section, for extension’s sake. After being able to identify different ECG conditions through a visual approach, an automatic detection mechanism that mimics this process is desirable. This leads us into the pre-processing, processing and classification algorithms and their hardware implementations described in Chapter 3.
Overview of ECG Processing Methods

3.1 Heart Diseases Detection

Automatic ECG processing has been a trending topic in the scientific community for several decades now [9]. As in many signal processing applications, drawing out conclusions from an ECG requires several steps. In general terms, these are: pre-processing, feature extraction and classification. Subsection 3.1.1 covers the pre-processing phase, where current filtering and artifact removal techniques are presented. Feature extraction mechanisms are described in Subsections 3.1.2 and 3.1.3. QRS-complex segmentation is the cornerstone of ECG processing and it is elaborated in Subsection 3.1.2. Subsection 3.1.3 extends Subsection 3.1.2 as ECG delineation requires the heartbeats to have been segmented in order to be applied. In this subsection it is shown how to extract other features than the QRS points that are important for the pathology detection. Subsection 3.1.4 contains the classification part where the decision-making processes for classifying the signals from the delineated features are elaborated. This standard algorithmic workflow for ECG signals processing is illustrated by Figure 3.1.

Figure 3.1: ECG signal processing algorithmic workflow.
3.1.1 Pre-Processing

As in most signal processing application, denoising and filtering are fundamental steps to have signal with more quality. ECG processing is no exception to this, and the characterization of the most relevant noise types that can be present in such signals has been presented over 25 years ago in the work of Friesen et al [9]. These are: power line interference, electrode contact noise, motion artifacts, muscle contraction (electromyographic), baseline drift and ECG amplitude modulation due to respiration, instrumentation noise due to the analog front-end, and electrosurgical noise.

Power line interference consists in the addition of 50 Hz (in Europe) sinusoidal noise and its respective harmonics. This kind of noise has a well-defined frequency content at 50 Hz. An example is shown in Figure 3.2(b)).

Electrode contact noise results from a temporarily loose electrode that effectively disconnects the measuring system. The usual behavior of the signal in this case is to saturate the input, having it look like a step function to one of the Analog-to-Digital Converter (ADC) output voltage extrema. The occurrence of this type of noise renders that section of the ECG useless.

Motion artifacts are a softer form of electrode contact noise where, instead of getting loose, the electrode simply slides. This changes the source impedance which causes transient baseline changes.

Muscle contraction interferences are usually high frequency and low amplitude components that can be compared to zero-mean Gaussian noise. They can range from a few Hertz to 10 000 Hz[9]. An example is shown in Figure 3.2(c).

Baseline drift is most commonly caused by the subject's respiration. It creates a variable 0.15 to 0.3 Hz sinusoidal oscillation which can be compared to an ECG amplitude modulation by a sinusoid which is added to the baseline.

Instrumentation noise can also be damaging in the cases where the signal saturated due to, for example, ADC voltage limitations. In ECG acquisition electronic front-ends, there are cases where the occurrence of this kind of saturation depends on the polarization of the electrodes, working only with the right setup. An example is shown in Figure 3.2(a).

The noise caused by electrosurgical tools renders the ECG useless. Its behavior is similar to that of adding a sinusoid with frequency ranging from 100 kHz to 1 MHz with an amplitude much higher
than the ECG signal[9].

The amplitude of the mentioned noise sources is variable with the electronic acquisition setup. If application specific, analog front-ends may be designed to filter some of the sources with no need to digitally filtering them.

3.1.1.A Noise Filtering and Artifact Removal

As of 2016, countless filtering techniques for noise removal have been tested and perfected. Phukpattaranont [19] provides a comparison of several of the most recent techniques developed (2009-2014) which provide outstanding results. All the algorithms were tested over the MIT-BIH arrhythmia database, which is the scientific gold standard for ECG algorithm performance evaluation. In this work Phukpattaranont used the following metrics for comparison: True Positives (TP), True Negatives (TN), False Negatives (FN), False Positives (FP), Sensitivity (SEN), Positive Predictivity Rate (PPR), and Detection Error Rate (DER). Zhu and Dong’s work [20] performed best in all these metrics for over 100,000 peaks of the MIT-BIH database. Their initial linear filtering process removes respiration and motion artifacts, muscle contraction, powerline noise and baseline drift by resorting to a 6th order Chebyshev bandpass filter with cutoff frequencies at 5 and 18 Hz. A forward and backward pass is used in order to revert the phase distortion. This kind of filtering does successfully remove all the noise in the signal, however it also removes relevant features of the ECG like P and T waves. Their filtering method is useful for the sole purpose of detecting the R peaks, and the removal of P and T waves prevents more FP’s during this task.

The bandpass filtering for noise removal has been used since 1985 in the work of Pan and Tompkins [21], where they use a 5 to 15 Hz filter to maximize the QRS energy detection. Their method has been cited and used (with different tunings) throughout the years in several papers [19]. The tuning of this filter is extremely relevant as it can allow for the P and T waves to still be present, while still cleaning the signal from most of the noise. It is known that the ECG has frequency content from about 0.5 Hz to slightly over 100 Hz, but most bandpass filtering methods don’t go over the 3 to 40 Hz band. This allows for a clear dismissal of the powerline interference (either at 50 or 60 Hz) and removes the low frequency components which cause the baseline drift and respiration artifacts. Although still containing some high frequency noise, this kind of filtering usually allows for well-defined P and T waves, whose presence is required for clinical diagnosis of several heart diseases. If a higher frequency band is desired, a notch filtered can be applied at the powerline frequency to remove this interference while keeping the higher frequency information.

More specific artifacts and noise removal can be performed through wavelet transform processing. Wavelet transform is a technique for simultaneous time and frequency signal analysis. It works by decomposing the signal into blocks with different time and frequency settings which allow the observation of the chronological evolution of the signal in different frequency bands. This is used to identify the components with noise and artifacts, and remove them to leave only the desired ECG waves. The work of Li, Zheng & Tai [22] in 1995 pioneered the use of this method in ECG processing, which would become a trend in ECG preprocessing algorithms, being still used in recent works [19].
Other mathematical processes have been put to use in this problem such as mathematical morphology, artificial neural networks and S-transform, but its weight is least expressive in the literature and its results do not overcome those of the previously mentioned methods [19].

### 3.1.2 QRS-Complex Segmentation

The time structure of a standard ECG signal is well-known, so having the ECG segmented is extremely useful to extract features of the P,Q,R,S and T waves (delineation). From the analysis of an ECG signal the focus usually goes to the QRS complex, as it is the most distinct part of the signal [23]. This complex is also the centerpiece of an ECG segment since it represents the electrical behavior of the ventricle contraction, providing crucial features for diagnosis. Being the most striking feature of the waveform, most methods focus on finding the R-peak first. Knowing the time structure of the standard P-QRS-T signal and the location of the R-peak, segmentation comes easy, as a simple window upon the R-peak does so (see standard segment in Figure 2.4). As seen in Section 3.1.1 methods for QRS segmentation are still a trending topic in the scientific community. However, most of the segmentation strategies have been divided until 2002, time of the review paper of Köhler, Hennig and Orglmeister [23]. This paper is a reference in this context and it provides a description of 18 different methodologies of QRS segmentation used in several distinct works of the 30 years before it. Due to content size restrictions, this work only presents a portion of the methodologies reviewed by Köhler. Most of the chosen algorithms were based on the comparison of algorithms presented in the recent work of Canento et al. [24]. This review takes into account the algorithms’ suitability for real-time ECG processing for biometric applications, which has similar requirements to the real-time heart disease detection application of the present work. A method developed by Köhler, Hennig and Orglmeister [23] is also analyzed, due to its high efficiency combined with low computational load [23]. Other recent approaches are described as well for informative purposes.

According to Phukpattaranont [19] there are several novel methods (more recent than 2009, most of them from 2014) which provide the best results yet for the ECG QRS segmentation problem against standard databases. There are works based in Quadratic Filtering [19], Linear Filtering + Energy envelope peaks [20], Wavelet Transform [26], Artificial Neural Networks [27] and Mathematical Morphology [28], among others. Most of these methods provide some updates and improvements, however the results are always tested for the same standard databases, where the margin for improvement was already particularly low. Some of them are also more complex, thus requiring a heavier processing power. The algorithms described in the next subsections were chosen due to having been put to test in noisier environments and being of low computational requirements [24].

#### 3.1.2.A Modified Engelse and Zeelenberg (Engzee)

Lourenço et al. [2] developed a real-time implementation of the algorithm by Engelse and Zeelenberg [29] for ECG segmentation, known for its robustness, although originally stricly applicable offline. This real-time algorithm is based on a digital filter, where the input ECG signal \( x[n] \) is passed through the differentiator in Eq. \( 3.1 \) being then low-pass filtered by the filter in Eq. \( 3.2 \).
\[ y_1[n] = x[n] - x[n - 4], \]  
\[ (3.1) \]

\[ y_2[n] = \sum_{i=0}^{4} c_i y_1[n - i], \quad \text{where} \quad c_i = [1, 4, 6, 4, 1]. \]  
\[ (3.2) \]

The resulting signal \( y_2[n] \) (see Figure 3.3) will have negative lobes in the same time of the R peaks in the original signal, which are identified through two thresholding operations. The first threshold is for the points \( Thi^+ \) and \( Thf^+ \). These points correspond to the limits of the first interval where \( y_2[n] > Th \), with \( Th = 0.6 \max(y_2[n]) \). The search for the negative blob follows with a 160 ms sweep in a window \( W \) after \( n_{Thf^+} \). If there are at least 10 points which verify \( y_2[n] < -Th \), then a candidate R peak is found. The peak then corresponds to the highest amplitude point of the 160 ms window \( W \) in the original signal \( x[n] \).

After having identified the R peak, the other waves still have to be identified. The Q and S points correspond to the minimums of \( x[n] \) immediately to the left and right of the R peak, respectively.

The problem with fixed thresholds is that for noisy settings the robustness of the algorithm is affected. To fix this, a second threshold was introduced [2], which takes the lowest amplitude of \( y_2[n] \) and uses it to set the threshold with the rules that follow:

\[ y_2[n] < Th_{New} \]  
\[ Th_{New} = 0.7 \times \min(y_2[n]) \]  
\[ (3.3) \]

All peaks that are below this threshold are then selected as R peak candidates.

However, for the real-time implementation, this adaptive thresholding mechanism was extended [2] to take an averaged update of the previous thresholds. Hence, the algorithm starts by taking the amplitude threshold as previously described for a 5 seconds window as Eq. \( 3.4 \). This window is represented by a 5-entry FIFO array that is then continuously slided, having the new thresholds being calculated as in Eq. \( 3.5 \) and the old ones being discarded.
\[ Th_i = 0.6 \times \max(y[n]) \]  

\[ Th = \begin{cases} Th_i & \text{during the initial 5s} \\ \frac{1}{5} \sum_{i=1}^{5} Th_i & \text{rest of acquisition} \end{cases} \]  

For the cases where \( Th_{\text{new}}^5 > 1.5 Th_5 \), it is considered that \( Th_{\text{new}}^5 = 1.1 Th_5 \). After the detection of a QRS, the algorithm blocks new detections for 200 ms. In the 1000 ms interval afterwards, the threshold is slowly decreased to 60% of its value, stabilizing until the next peak is detected.

This approach performs well in real-time due to the simplicity of the computations involved and the fact that the Engelse and Zeelenberg algorithm requires a buffer of only 4 samples.

### 3.1.2.B ECG Slope Sum Function (ESSF)

The Slope Sum Function is a derivative-based R peak detection algorithm which is characterized by its low computational requirements and easy implementation [24]. It was first used by Zong et al. [30] in the detection of arterial blood pressure pulses. Lourenço et al. [24] adapted this algorithm to the detection of ECG peaks, as the arterial blood pressure signals have a similar behavior to ECGs due to its cardiac origin.

This algorithm attempts to enhance the R-peaks while suppressing the rest of the signal. The discrete derivative and a half-wave rectification (Equation 3.6) are taken to make the R peaks’ rising slope stand out. This is particularly effective due to the slower and less intense behavior of the rest of the waveform.

\[ \delta x[n] = \begin{cases} x[n] - x[n-1] & x[n] - x[n-1] > 0 \\ 0 & \text{otherwise} \end{cases} \]  

A moving average 250 ms window is then taken to smoothen the derivative, followed by onset detection when the values exceed a fixed threshold. The original signal is then scanned in the 100 ms after the onset was detected, and its maximum is validated as an R peak if it corresponds to a beat in the 20 to 200 beats per minute range.

Since the algorithm is based on simple arithmetic operations, its computation is extremely fast and inexpensive, although its performance is not as good as the modified Engelse and Zeelenberg algorithm described in the previous subsection [24].

### 3.1.2.C Zero Crossing Counts

The zero crossing counts developed by Kohler et al. [25] was designed to be used in low computational power environments. It consists in summing a high-frequency sequence with amplitude lower than the R peak to the ECG signal and detecting the amount of times the signal crosses zero. A general block diagram of the algorithm is shown in Figure 3.4.
After the common band-pass filtering in the pre-processing, the signal is enhanced through the nonlinear transform expressed in Equation 3.7, which makes the ECG peaks more protuberant.

\[
y[n] = \text{sign}(x[n]) \cdot x^2[n], \tag{3.7}
\]

where \(x[n]\) corresponds to the band-pass filtered signal.

The high frequency sequence is given by

\[
b[n] = (-1)^n \cdot K[n], \tag{3.8}
\]

where \(K[n]\) is the amplitude of the high-frequency sequence. Therefore, the final signal is

\[
z[n] = y[n] + b[n]. \tag{3.9}
\]

This high frequency sequence increases the number of zero-crossings in the non-QRS areas. However, \(K[n]\) has to be tuned. If its value is too large the algorithm will be pointless, if too small it will be noisy and insignificant. Therefore, \(K[n]\) is given by

\[
K[n] = \lambda_k K[n-1] + (1 - \lambda_k)|y[n]| \cdot c, \tag{3.10}
\]

where \(\lambda_k \in [0; 1]\) is a forgetting factor, and the design parameter \(c\) denotes a constant gain, e.g., \(c = 4\) as suggested by Köhler et al. [25].

For the zero crossings detection, Köhler et al. [25] use

\[
d[n] = \left| \frac{\text{sign}[z[n]] - \text{sign}[z[n-1]]}{2} \right|, \tag{3.11}
\]

\[
D[n] = \lambda_D D[n-1] + (1 - \lambda_D)d[n], \tag{3.12}
\]

having also \(\lambda_D \in [0; 1]\). This corresponds to an autoregressive low pass filter which is computationally efficient and easily adjustable. When the value of \(D[n]\) goes below an adaptive threshold given by

\[
\Theta[n] = \lambda_\Theta \Theta[n-1] + (1 - \lambda_\Theta)D[n], \tag{3.13}
\]

having \(\lambda_\Theta\) as forgetting factor for \(\Theta[n]\), a peak is detected. However, for the same R peak more than one peak in \(D[n]\) may be detected, hence a time out must be established for an interval, e.g. 100 ms.
3.1.3 ECG Delineation and Feature Extraction

To provide information on the different waves of the ECG, QRS complex segmentation is only the first stage. As soon as it is detected, extraction of P and T waves’ features follows to provide more information on the heart’s condition. In this section, three different methods will be described.

3.1.3.A P-QRS-T Waves Detection

In 2008, Singh and Gupta [31] presented a derivative-based feature delineator for biometric applications. After having the QRS complex mapped through the algorithm of Pan and Tompkins [21], the method starts by delineating the P wave. To find it more easily, the first derivative is taken for every time sample through the difference equation

\[ y[n] = -2x[n - 2] - x[n - 1] + x[n + 1] + 2x[n + 2]. \]  (3.14)

For the determination of the P wave’s fiducials, a time window is set heuristically from the beginning of the heartbeat to the beginning of the QRS complex. It can be considered that the heartbeat begins in the first isoelectric sample before the atrial slope. An adaptive slope threshold (\( \theta \)) for the detection of the wave is then computed from the mean of the most recently detected significant slopes (\( \mu_{MS} \)) and the mean of the high frequency noise (\( \mu_{HF_{noise}} \)) present in the detected beats as

\[ \theta = 0.30(\mu_{MS} - \mu_{HF_{noise}}), \]  (3.15)

whereas the level of the high frequency noise is given by

\[ y'[n] = x[n] - 2x[n - 1] + x[n - 2]. \]  (3.16)

After taking the mean of \( y'_{nT} \) over a stream of samples, \( HF_{noise} \) comes from the ratio of the maximum averaged signal and the QRS amplitude as

\[ HF_{noise} = K\frac{HF_{MA_{noise}}}{H_{QRS}}, \]  (3.17)

having K set to 40 in their case. The P wave fiducials are computed backwards. Its end, \( P_{offset} \), corresponds to the minimum of the derivative \( y'[n] \). The peak would correspond to the zero-crossing point of \( y'[n] \), however the real maximum corresponds to the neighboring point where the slope exceeds \( HF_{noise} \). The initial point, \( P_{onset} \), is the inflexion point of the derivative before its positive slope.

The T wave delineation requires a low-pass filtering of the original signal, which is done recursively through


A time window is then heuristically set from \( QRS_{offset} + 80 \) ms to \( QRS_{offset} + 470 \) ms, where the T wave should be found. Its peak is found with a derivative and adaptive thresholding approach analogous to the P wave’s. The detection of the endpoint \( T_{offset} \) starts with the peak of the derivative
inside this time window. Setting a line \( L_R \) from \( d_{\text{peak}} \) to the final point of the derivative \( d_R \). The slope of this line is given by these points' coordinates as

\[
\Delta = \frac{y_R - y_{\text{peak}}}{x_R - x_{\text{peak}}}. \tag{3.19}
\]

For any point \((x_i, y_i)\) in the interval \([d_{\text{peak}}; d_R]\), its vertical offset to the line \( \delta_i \) is given by

\[
\delta_i = |y_{\text{peak}} + \Delta(x_i - x_{\text{peak}}) - y_i|. \tag{3.20}
\]

The point \( i \) where this offset is maximum will correspond to the location of \( T_{\text{offset}} \). Having the same process for a line between the first point of the window and \( d_{\text{peak}} \) will provide the T wave's initial point, \( T_{\text{offset}} \).

3.1.3.B Automatic features from the DWT

Since ECG delineation is not always possible in some acquisition setups, automatic feature extraction methods have been developed. However, researchers agree that the wavelet transforms are the best methods to extract features from the an ECG signal [32]. This is also due to the computational simplicity of methods such as the DWT.

In 2013, Bustamante et al [33] combined the DWT and the Continuous Wavelet Transform (CWT) to feed delineation features into a support vector machine classifier.

Their preprocessing started with the signal being reconstructed through the detail coefficients \( c_{D2} \) to \( c_{D5} \) of the DWT (for a sampling rate of 250 Hz) with the Daubechies’ 4th order wavelet function (DB4) (see Figure 3.5 (b)) as mother wavelet. The correlation between its shape and the QRS complex makes it a good choice for this task. The reconstructed signal is then put through the Pan and Tompkins algorithm [21] for the QRS complex detection.

For the QRS complex detection, the signal is put through the scale 8 of the CWT, being the first derivative of the Gaussian function (Figure 3.5 (a)) the mother wavelet. The signal is reconstructed with the detail coefficients \( c_{D2}, c_{D3}, c_{D4} \) and \( c_{D5} \), and the R-peak is detected through the adaptive thresholding proposed by Pan and Tompkins algorithm [21]. The T wave peak is detected through the Modulus Maxima method [33]. The maximums and minimums of the window are then found. After the last maximum modulus pair of the QRS complex detected in this signal, which corresponds to S, an inflection point, a small depression or a zero crossing may be used to set the initial point of the T wave \( W_{\text{Tin}} \). The T wave window will correspond to the 0.4*(R-to-R distance) ms after this point, where the final point will be \( W_{\text{Tend}} \). Inside this window, if there's only one maximum, its location will decide whether its a positive T-wave (maximum in the first half of the window) or an ascending T-wave (maximum in second half of the window); if only a minimum, it is a negative T-wave. For the case where both a maximum and a minimum are found: if there is only one zero crossing, it is considered for the one with highest absolute amplitude (either the minimum or the maximum); otherwise, when two are detected, it is a biphasic T-wave.

The P wave peak is detected through the same method of the T wave, having the time window ending in the QRS complex onset and starting 0.25*(R-to-R distance) ms before that.
For the detection of the onsets and offsets of the QRS complex and both T and P waves, different settings of both DWT and CWT were tried. Both the onset and offset of the QRS complex, as well as the P wave onset, were better obtained through the signal reconstructed from the detail coefficients cD4 to cD8 of the DWT (at 250 Hz sampling rate) with DB4 as mother wavelet. The T wave onset and offset, and the P wave offset resulted from the CWT at scale 6 with the Mexican hat function (Figure 3.5 (c)) as mother wavelet. In each case, the derivative of the signal from either the DWT or CWT was searched for zeros, and the first zero on the left and right of the P, T and QRS-complex peaks would determine the onset and offset, respectively, of each wave.

Figure 3.5: Mother Wavelets: (a) First derivative of the Gaussian Function, (b) DB4 function, (c) Mexican Hat function

3.1.3.C Mixed methodologies

As previously referred, the use of DWT for feature extraction is one of the preferred methods by researchers in the last few years. Some have extended its use and coupled it with other automatic features and dimensionality reduction methods. That is the case for the work of Ye et al [34], where features from the DWT were mixed with features from applying Independent Component Analysis (ICA) and some dynamic features. The resulting feature set was put through PCA to reduce the number of features to be used by the classifier. Martis et al [35] draws a set of features from the same methods and extends the dimensionality reduction by applying Fisher’s Linear Discriminant Analysis (LDA) as well.

Ye et al [34] apply a 4 level DWT with Daubechie’s 8th order wavelet function DB8 to every segment obtained from the database annotations. The detail coefficients cD3 and cD4 and the approximation coefficients cA4 are used as features. Since most of the energy of the ECG signal is concentrated in frequency range from 0.5 to 40 Hz [34], these coefficients cover the most significant part of that frequency range.

In parallel, ICA is applied. As a blind source separation algorithm, ICA provides a set of independent source signals (known as Independent Components (IC’s) from a set of observed signals. It assumes that N observed signals result from a linear combination of M source signals, which is translated by

$$x(t) = A \cdot s(t),$$

where \(x(t) = [x_1(t), ..., x_N(t)]^T\), \(s(t) = [s_1(t), ..., s_M(t)]^T\) and \(A\) is called the mixing matrix. In this application, \(s(t)\) is a matrix whose rows contains the IC’s in a descending order of energy, and thus of relevance. The resulting \(s(t)\) is then used for extracting the features of each signal of the training set.
The ICA coefficients $C_{ICA}$ used as features result from the projection of each signal onto the source signals as described by

$$C_{ICA} = x(t) \cdot s(t),$$

(3.22)

where $C_{ICA}$ is a matrix with $M$ columns and $N$ rows.

Ye et al. [34] concluded that using the first 14 ICA's and the 114 cD3, cA4 and cD4 coefficients from the DWT allowed obtaining the best results. However, this feature set results in a computationally heavy classifier, while having some redundancy in the feature set. Thus the usage of PCA to reduce the number of features. This method computes a vector space that projects the data into its directions of highest variability [35]. From applying the PCA to a data set, a matrix is obtained where its columns are the basis vectors of this projection space (principal components) and are sorted from the highest to the lowest variability. This projection matrix is computed through the following steps:

**Step 1:** Compute the covariance matrix from the data matrix $X$, whose rows correspond to the feature sets of the different segments, as

$$C = (X - \bar{x})(X - \bar{x})^T,$$

(3.23)

where $\bar{x}$ is the mean vector of $X$.

**Step 2:** Compute the eigenvectors and eigenvalues matrices $V$ and $D$ which verify

$$V^{-1}CV = D.$$

(3.24)

**Step 3:** Obtain the projection matrix $W$ by sorting the eigenvectors in descending order of their corresponding eigenvalue. $W$ is a square matrix with the size of the feature set used to generate it.

After these steps, new feature sets can be projected to any $M$ directions by using the first $M$ columns of $W$ as in

$$Y = W_M^T X.$$

(3.25)

For Ye et al [34], the dynamic features are calculated for each beat from the previous RR intervals. They considered three features: the immediate previous RR interval; the local average of the RR-intervals of the past 10 seconds; and the average of the RR-intervals from the previous 5 minutes. Due to the different focus of the characteristics of the features, the dynamic features are only concatenated with the morphological features after these are put through PCA.

The final feature set is then fed into a classifier, as will be described in Section 3.1.4.

### 3.1.4 Heart Diseases Detection

As every heart is unique, the shape of the ECG waveform varies from person to person. However, from the normal heartbeats to the irregular ones there are morphologic changes that are independent from the subject. Several approaches have been described in literature, being some briefly elaborated in this section.
Patel et al. [36] considered simple morphological metrics for their real-time classification system on a mobile platform. A heart rate lower than 60 beats per minute would be classified as bradycardia, where one over 110 beats per minute would qualify for tachycardia. VPC does not affect the heartbeat, but it affects the duration of the QRS complex. Therefore, a beat containing a QRS complex with over 120 ms of duration and without a P wave is classified as VPC. As for APC detection, two conditions have to be verified. The first one is to check if a single premature beat was verified, and this happens when a drastic change of the length of the RR interval occurs. The threshold applied in this case is

\[
\frac{RR_n}{RR_{n-1}} > 1.2. \tag{3.26}
\]

If this condition is verified to be true, a second stage will define if it is an APC or a VPC. If one of the three conditions of Equation 3.27 is verified, an APC is detected.

\[
\begin{cases}
  w_n < 5\% \text{ or } \\
  w_n < 10\% \text{ and } a_n < 25\% \text{ or } \\
  g_n < 25^\circ \text{ and } w_n < 50\% 
\end{cases} \tag{3.27}
\]

where \( w_n \) is the difference between the widths of the current QRS and the previous one; \( a_n \) is the difference between the area of the current QRS and the previous one; \( g_n \) is the difference between the angle \( Q\hat{R}\hat{S} \) of the current QRS and the previous one. Otherwise, the single premature beat is considered a VPC.

However, the most recent and most effective heartbeat classification mechanisms have a different approach from this method, being much more complex and computationally expensive. Being published with only a few months of distance in 2012 and 2013, the works of Martis et al. [35], Wang et al. [37] and Ye et al. [34] have about the same approach, varying only some of the used tools. All of these start by taking a DWT-based denoising approach. A signal reconstructed from some detail coefficients has the QRS complexes segmented and each sample is considered to be a feature. For the purpose of dimensionality reduction all three approaches make use of PCA. Then Wang et al. [37] uses LDA while Ye et al. [34] resorts to ICA having Martis et al. [35] combining both of the methods. These three approaches are different methods to project the data into other dimensions of higher variance and correlation, having a reduced set of features that corresponds to a good approximation of the data. Ye et al. [34] then use a Support Vector Machine (SVM) to classify each heartbeat into one of 16 different classes. Both Martis et al. [35] and Wang et al. [37] try different settings of Neural Networks and SVM’s. Their results unanimously state that the use of a PNN classifier after dimensionality reduction with PCA+LDA (+ICA in Martis et al. [35]) provides the best result possible, all around 99.5% of accuracy.

### 3.1.4.A Probabilistic Neural Networks

From the results of the work of Martis et al [35] and Wang et al [37], and with the knowledge that neural networks can usually be implemented with standard arithmetic blocks, thus being fit for hardware implementation, it was decided that a PNN would be used as a classifier for the system.
PNNs are usually composed by three layers: Input Layer, Radial Basis Layer and Competitive Layer [38]. A functional representation of the general structure is shown in Figure 3.6.

### Figure 3.6: Probabilistic Neural Network functional structure.

The Input Layer consists of the signal or feature set to be classified. For the current method, we will consider it as a one-dimensional array of \( R \) features. The Radial Basis Layer implements

\[
A_i = \text{radbas}(||IW_i - F||b_i),
\]

where \( A_i \) is the \( i \)-th element of the Radial Basis Layer \( A \), \( IW_i \) is the \( i \)-th row of the matrix input weights, and \( b_i \) the \( i \)-th element of the biases vector. Each row of the \( IW \) matrix is a feature vector of the training set, which excludes the necessity of a backpropagation algorithm for obtaining a network model, unlike other neural networks. The radial basis function can be any gaussian function of the format

\[
\text{radbas}(x) = ae^{-\frac{(x-b)^2}{2c^2}},
\]

where \( a \), \( b \) and \( c \) are arbitrary constants. This layer takes the input features, computes the distance to each of the training feature vectors, multiplies it with the bias and computes an absolute similarity metric, in this case with a squared negative exponential function. This metric is then fed to the Competitive Layer, where the similarity values corresponding to each class are summed according to the values of the \( LW \) matrix. For each of its columns, only one row is 1, whereas the others are 0, and that is the corresponding class of that neuron. Once all values have been summed, the class with the total maximum probability is outputted in a vector format [38].

### 3.2 Hardware Implementations of ECG Processing Systems

As the software current strategies of ECG processing were introduced in the previous section, the same is presented for the hardware. Since this work is mainly based on FPGA-based systems, these will be compared to other technologies, while also providing an introduction to their key aspects of structure and behavior, in Subsection 3.2.1. Subsection 3.2.1.3 contains a modular approach of
the hardware implementation considerations of some of the most relevant software strategies of the previous section. Finally, hardware implementations of full ECG processing systems are presented.

3.2.1 FPGA-based Design

Computational platforms can be set upon different kinds of device technologies. Integrated Circuits are the foundation of most electronic devices nowadays, but they can be of two kinds: Application Specific Integrated Circuit (ASIC)’s or programmable devices. The former corresponds to a single or multi layer integrated circuit which is designed to have a fixed behavior. Programmable devices have pre-determined layers and are customizable by modifying an internal memory. They can be subdivided into Programmable Logic Device (PLD)’s and FPGA’s. PLD’s are simple two-level arrays of AND and OR gates, plus a few registers, which can perform simple logic and arithmetic operations. FPGA’s, on the other hand, combine the structure of ASIC gate-arrays with the programmability of PLD’s, and are composed mainly by arrays of generic logic cells in an interconnected structure.

FPGA’s are a key component in high-performance Digital Signal Processing (DSP) devices due to their programmable structure and computational power [16]. Several authors have been implementing ECG processing algorithms in FPGA’s for the past six years [12–17], however there are also researchers developing ASIC’s for this purpose [6], or even mobile phone platforms [36]. In the present work, the focus is solely on FPGA’s due to their easy prototyping capabilities.

3.2.1.A FPGA Structural Elements

Having the ability to be programmed after fabrication, modern FPGA’s pack more than two million logic cells [3]. This allows for the implementation of a wide array of software algorithms. Its basic structure is composed of four kinds of elements: Look-Up Table (LUT), Flip-Flop (FF), Wires for routing and Input/Output (I/O) pads. The former two are the key components of the Configurable Logic Blocks (CLB), which compose the main structure of the FPGA, which is as shown in Figure 3.7 a).

More recent FPGA’s can also have columns of dual-port Random Access Memory (RAM), 48-bit Digital Signal Processors (DSP48), high speed serial transceivers and Phase-Locked Loop (PLL) clock generators.

LUT’s are the basic building blocks of an FPGA as they allow for the implementation of any logic function of $N$ boolean variables. By acting as truth table, the $N$ inputs of a general $N$-input LUT will have $2^N$ memory locations to access, which result in up to $2^{2N}$ functions. The value of $N$ is typically 6 for Xilinx’s FPGA’s [3]. Its structure allows for a usage both as a function computation unit and a data storage unit. This is due to a hardware implementation with a functioning that mimics a collection of memory cells connected to a set of multiplexers, where the inputs act as the selector bits, as seen in Figure 3.7 b).

Flip-flops are the basic storage elements of FPGA’s. Together with some control logic, they aid LUT’s in logic pipelining and data storage by composing a CLB. Data I/O set, reset, clock and clock
enable pins make up the standard structure of FF as in Figure 3.7 c). The data inputed is read every clock pulse, if the clock enable is on, and outputted in the next clock pulse.

![Figure 3.7: a) Basic FPGA architecture layout; b) Basic LUT functional architecture; c) Basic FF layout.](image)

The DSP48 is an embedded Arithmetic Logic Unit (ALU) and the most complex computational block in the fabric of a Xilinx FPGA. It is composed of three pipeline stages: an add/subtract unit, a multiplier and a final add/subtract/accumulate unit. This unit can implement either

\[ p = a \times (b + d) + c \]  \hspace{1cm} (3.30)

or

\[ p + = a \times (b + d), \]  \hspace{1cm} (3.31)

and its architecture is illustrated by Figure 3.8.

![Figure 3.8: DSP48 functional architecture](image)

Aside from the LUTs, Block Random Access Memory (BRAM) and shift registers (composed of FFs and control logic) can be used as RAM, Read-Only Memory (ROM) or shift registers. BRAMs can hold either 18 or 36 kbits of data from 16/32 k \times 1 bit to 512 \times 36/72 bits, depending on the ports usage. These blocks can be used as true dual-port blocks, meaning that data can be written to or read from the two ports during the same clock cycle. As storage elements, LUTs in Xilinx FPGAs can be used as 64-bit distributed memories. Since they can be instantiated in any part of the FPGA.
fabric, they decrease dramatically the routing time. This makes these structures the fastest memory blocks available, hence improving the circuit’s performance. However, distributed memories occupy more area than BRAM for the same amount of data storage. Shift-registers are chains of registers whose purpose is to reuse a stream of data along a computational path. This type of memory can be useful for the implementation of filters.

### 3.2.1.B FIR/IIR Filter Implementations

Finite Impulse Response (FIR) and IIR filters are two types of filters commonly used in digital signal processing [39] [40]. The simplicity of their functioning while guarantying good filtering makes them desirable for hardware implementation. The IIR filter has two parts: one depends on the filter input and the other on its output. For every moment, the output of an IIR filter is given by:

$$y(n) = \sum_{i=0}^{M} b_i x(n - i) - \sum_{j=1}^{N} a_j y(n - j), \quad (3.32)$$

where $a$ and $b$ are the filter coefficients and $x(n)$ and $y(n)$ are the input signal and the filter output, respectively. An FIR filter verifies Equation 3.32 for $a_1,...,N = 0$. This function is usually implemented either in the Direct-Form 1 (DF1) or the Direct-Form 2 (DF2). The DF1 is the most intuitive and straightforward representation of Equation 3.32 and a 2nd order filter is represented in Figure 3.9 a). However, DF1 has twice as many delays than necessary, so it is not the most suited for hardware implementation [41]. Thus, the DF2 comes as the most suitable format, and its output is given by

$$v(n) = x(n) - a_1 v(n - 1) - a_2 v(n - 2), \quad (3.33)$$

$$y(n) = b_0 v(n) + b_1 v(n - 1) + b_2 v(n - 2), \quad (3.34)$$

as represented in Figure 3.9 b).

![Figure 3.9: 2nd Order Digital Filter in a) Direct-Form 1; b) Direct-Form 2.](image)

In cases like low-pass, high-pass or band-pass filtering, a decomposition of a filter in a series of second-order sections is usually a good choice of implementation [41]. The second-order stages, such as the ones in Figure 3.9 are usually connected by a gain stage. In terms of hardware, this means that, if the time constraints allow it, it is possible to implement hardware for only one single second-order section and to reuse it for every stage.
More complex implementations of IIR/FIR filters can be of use when implementing high order filters, such as using Distributed Arithmetic [39] or an extended double base number system [40]. However, for the current work they were not necessary.

### 3.2.1. C DWT Implementations

As seen in Section 3.1.3, the DWT is one of the main methods for feature extraction in ECG processing, as well as in digital signal processing in general. Therefore, its hardware implementation has been object of thorough research. The most successful DWT algorithm is the Pyramid Algorithm Analysis by Mallat [42] [43]. It is mainly filter bank structure such as the one in Figure 3.10 composed of a low-pass filter \((G_0)\) and a high pass filter \((H_0)\), as well as down sampling stages at the output of each filter. These two filters decompose the signal into the input signal’s approximation coefficients and detail coefficients, respectively. The down sampling is applied to allow the reutilization of the same filters on the approximation coefficients in order to generate coarser decompositions.

The filters are chosen according to the desired wavelet, being the Daubechie’s wavelet the most common choice. They are ideal for digital hardware implementations since they are maxflat filters. Among other properties, their coefficients are dyadic, which means they are rational numbers with a power of two as denominator. This makes up for a conversion to binary with little to no losses of accuracy, which is certainly useful.

![Figure 3.10: Mallat’s Pyramid Algorithm Analysis three-level wavelet decomposition structure.](image)

However, due to the down-sampling of the filters’ outputs at each stage, several arithmetic operations are pointless since their output is discarded. To make up for this issue, the low-pass and high-pass can be decomposed in their odd and even counterparts, having only the respective odd or even signal samples put through them. Hence the hardware can be halved since the odd and even parts are active in alternate moments, as in

\[
\begin{bmatrix}
LP_{even} & LP_{odd} \\
HP_{even} & HP_{odd}
\end{bmatrix}
\begin{bmatrix}
X_{even} \\
Z^{-1}X_{even}
\end{bmatrix}
=
\begin{bmatrix}
Y_0 \\
Y_1
\end{bmatrix}.
\]

(3.35)

This is called polyphase filtering, and a single-level structure is illustrated by Figure 3.11.
From the FPGA implementation results of Al-Haj [44], Chilo et al [42] and Jarrah et al [45], it was considered that the polyphase structure by itself would qualify as the structure which would better fit this work, presenting a good compromise between performance and area utilization. It can provide a better performance than the conventional structure while keeping low area utilization [44]. Since the use of the DWT in this context is only as a feature extraction method, only the analysis component of the DWT is required.

3.2.1. D PNN Implementation

The structure of a PNN allows for several different implementations, depending on the desired system requirements. Figueiredo and Gloster [4], probably due to the FPGA limitations at the time of their work (1998), have developed a compact fixed-point architecture for their PNN for satellite image processing.

In order to reduce the area utilization, the arithmetic units process 4 10-bit elements simultaneously in a structure like the one of Figure 3.12. This creates the need for accumulation units, both for the distance sums and for the class sums.

In their implementation, the $K^2$ values correspond to $-\frac{1}{2^c}$ and $K^1 = a$ from Equation 3.29. Most of the structure is straightforward arithmetic functions like subtractions and multiplications. However, the exponential function is the one with a trickier implementation. Figueiredo and Gloster decided to implement it as a 14 bit LUT which makes up for 16,384 positions of 32 bits, in this case. With this amount of values, a set of BRAMs is the most indicated choice, as distributed memory of this size would cause a bigger area utilization.

On a different perspective, Zhou et al [5] main concern was towards the maximization of performance, thus using most of the resources available of the FPGA. A 32-bit floating-point was implemented with a structure as illustrated by Figure 3.13. For the distances, 32 subtraction units are used, followed by 32 multipliers and 5 stages of adders to sum the outputs of the multipliers and accumulate. Meanwhile the values are stored in BRAM dedicated to each arithmetic function.
Unlike Figueiredo and Gloster [4], the exponential function is expressed through a 7-level Taylor expansion as in

\[ \exp(n) = 1 + \sum_{n=1}^{\infty} \frac{x^n}{n!}, \]  

for \( n = 7 \). Its implementation takes two blocks, one to decompose the floating point value into integer and fractional parts (Figure 3.14 a)), and another block which processes both parts simultaneously (Figure 3.14 b)). The integer part gets its exponential value from a 32 elements LUT and the fractional part from a bank of adders and multipliers.
In total, 234 DSP48s, 140 kBytes of BRAM, 26,855 LUTs and 19,423 slice registers were used, which makes this implementation impossible in lower grade FPGAs, but was suitable for the used Xilinx Virtex-5. This implementation performed 37.9 times faster than its Matlab implementation running on a PC.

### 3.2.2 FPGA-based ECG processing systems

The current state-of-the-art works gather a set of different applications, having simple arrhythmia detectors based on the heart rate or other straightforward features [14, 17, 36], more complex algorithms to detect P and T waves in real-time [13] or ST segment elevation/depression [16], and, at the top, neural networks [6, 15] and fuzzy clustering [46] based algorithms, which make up for a bigger challenge.

However, most authors either do not fully implement a classification system or, when they do, the architectures they describe have incomplete information or used micro-processors to aid the processing. Jewajinda et al [15] and Chatterjee et al [13] developed dedicated VHSIC (Very High Speed Integrated Circuits) Hardware Description Language (VHDL) architectures, however no data on the hardware resources, timing or power was provided. Both Mimouni et al [16] and Junior et al [46] implemented the architecture in FPGAs MicroBlaze processing unit, also with no indications of the performance aside from the signal classification quality.

The work of Sun et al [6] offers a comparison between the use of 200 Hz smartphone processor programmed through software and a 500 MHz ASIC which is presented in Figure 3.15. The ASIC which uses 45 nm CMOS technology, was first prototyped in a Xilinx Virtex-5 FPGA with 65 nm scale.
This yields faster execution time and lower power consumption for the ASIC, but due to the high costs and quantities required to produce a batch of ASICs, the circuit should be designed, verified and then tested on an FPGA to avoid wasting money and production resources.

![Comparison between a PNN based algorithm implementation in an ASIC and a Smartphone software implementation: (a) execution time comparison; (b) energy consumption comparison](image)

On another hand, an FPGA based feature extraction system for prenatal ECGs was developed by Abburi et al. [47] verified a maximum clock frequency of 22.4 MHz and a total power consumption of 68.25 mW.

### 3.3 Summary

An overview of the state-of-the-art methods for ECG processing was described in this chapter, both for software and hardware. In Section 3.1 were analyzed: filtering and noise removal methods; peak detection and QRS location mechanisms; fiducial points and automatic features detection algorithms; a neural network classifier. Section 3.2 verified the correspondent hardware implementations of some methods of the ECG processing algorithms from the previous section. More specifically, basic FPGA components and functioning were introduced, as well as implementations from different researchers of FIR/IIR filters, DWT stages, PNNs and different ECG processing systems.
ECG Processing Software Implementation

In Section 3.1 several software methods were analyzed for each of the processing stages of heart detection. Chapter 4 presents the implemented software methodology. Firstly, a version of the system was developed on Mathworks’ Matlab R2016a, which was used to test its functionality. The first iteration of the system was made for offline usage solely to set the basic functions and test the feature extraction and classification algorithms. The second iteration was developed bearing in mind the future hardware implementation, having utilized and created fixed-point analysis tools to mimic the final system as close as possible. Matlab’s inbuilt functions were also reproduced in order to understand all the algorithms to their full extent, and provide the means for a hardware implementation.

Section 4.1 contains a global overview of the implemented software classification system. The remaining sections explore the processes introduced in said section, and different considerations and decisions taken at each stage of the processing workflow are described, from the pre-processing strategies until the classification methods. Most figures in this chapter were obtained from the software implementation of the particular case of Lead 1-only processing described in Chapter 6.

4.1 Overview

The processing workflow is, as previously mentioned, composed by four main stages: pre-processing, heartbeat segmentation, feature extraction and classification. Figure 4.1 contains block diagram of the dataflow of the whole process.

The current software system implements a three-part pre-processing stage. First, the baseline is removed by subtracting the result of a moving average filter to the signal. A high-pass filtering is
then applied to remove the high-frequency noise. The pre-processing ends after the amplitude of the signal is normalized to have its peaks around 1 mV.

Since the databases used for model development must have annotations of the heartbeats, the heartbeat identification (the first step of the heartbeat segmentation) was skipped. Thus, a window is applied around the annotated peaks and the heartbeats are then segmented.

Feature extraction begins with the dynamic features of the signal being calculated from the time intervals between the R peaks of each heartbeat. Once extracted, the heartbeat segment goes into the resampling block where, if needed, it is stretched or compressed in order to have the full length of the P-QRS-T waves one only one heartbeat inside the segment window. This step is followed by the extraction of the morphological features of the segment through the analysis coefficients of the DWT. Dimensionality reduction ensues, and with a reduced set of morphological features plus the dynamic features, the system is ready for classification.

The final set of heartbeat features is divided into a training set and a test set. The former is used to generate the parameters of a PNN classifier, which is then applied to the test set. This process renders a set of labels for each of the test set heartbeats, which are the final product of the ECG heart disease detection system.

**Figure 4.1:** Global overview of the ECG processing software implementation dataflow and main processing blocks.

### 4.2 Pre-Processing

As seen in Subsection 3.1.1, there are several filtering and noise removal methods that allow for a better feature extraction and, consequently, beat classification. In this section, the approach which provided the best results for noise removal and signal standardization for real-time applications is described. Since ECG pre-processing requires filtering at both low and high frequencies, the first approach was to design a band-pass filter. However, sufficiently good filtering was only achieved with a band-pass IIR filter of an order of over 150, which is fine for software supported simulation, but unreasonable for an optimized hardware architecture. Therefore, the filtering was separated into a high-pass filtering based on a moving average window and a low-order low-pass IIR.
4.2.1 Baseline Filtering

Baseline drift is a dynamic low-frequency offset which can be removed with a high-pass filter. However, this baseline drift can be approximated by the mean of the signal for well calibrated window. Since our goal is to have the signal oscillate around zero, a moving-average filter was applied to the signal and the result subtracted to it. This method is an extremely efficient high-pass filter which bears almost no computational load. Different window sizes were tested: 125 ms, 250 ms, 500 ms, 1000 ms and 2000 ms. For each sample of the signal, the average of the amplitudes of all the samples inside the window centered on it would be subtracted. The spectral differences in the filtering with each window is evident in Figure 4.2.

![Single-Sided Amplitude Spectrum of InCarTDb Recording 1](image)

**Figure 4.2:** Frequency spectrum of an ECG signal after baseline filtering with different window sizes.

Analyzing Figure 4.2 it is clear that the larger the window size, the steeper the low-pass filtering at 0 Hz, thus less signal is lost with the filtering. To achieve a filtering of this quality, an extremely high order high-pass filter would be required, which justifies the usage of the moving average filter. Bearing in mind further applications, the window of 500 ms was chosen. This is the window size which provides the best compromise between filtering quality and computational efficiency.

4.2.2 Low-pass Filtering

After the baseline filtering, the most concerning noise is of high frequency. Thus, a low-pass filter was dimensioned in as to remove this noise. As seen in section 3.1.1, authors have different opinions as to which should be the cutoff frequency at the higher frequencies. Given that the purpose of filtering for this work is to have well-defined P, Q, R, S and T waves with little to no noise, different filters were designed. Since FIR filters have a low stop-band attenuation for lower orders and is computationally more complex than an equivalent IIR filter, the aim was set to design an IIR filter with the lowest order possible, since the phase shifts do not affect the signal greatly. Different quantized IIR filters were tested, with Chebyshev Type II filters with 45 Hz of cutoff frequency and 80 dB of stopband
attenuation of orders 6, 8 and 10 providing the best results. The spectral analysis of the filtering with each of those filters after baseline filtering with a window of 500 ms is illustrated in Figure 4.3. One of the noticeable advantages of the Chebyshev Type II filters is the absence of ripples in the attenuation band.

![Single-Sided Amplitude Spectrum of InCarTDb Recording 1](image)

**Figure 4.3:** Frequency spectrum of an ECG signal after baseline filtering and after baseline filtering + IIR filtering of orders 6, 8 and 10.

Since from the frequency spectrum isn’t really possible to be sure which is the most suitable filter, an analysis of the time signal is required. In Figure 4.4 it is evident how each filter affects the previously baseline filtered signal. From here, it was decided that the 8th order filter provided the best compromise between high frequency noise removal and similarity to the original signal.

![ECG signal after baseline filtering and after baseline filtering + IIR filtering of orders 6, 8 and 10](image)

**Figure 4.4:** ECG signal after baseline filtering and after baseline filtering + IIR filtering of orders 6, 8 and 10.
4.2.3 Signal Normalization

ECGs have tremendous amplitude variability between patients, sometimes even for the same patient, and also between acquisition setups. Given that the aim of this work is to provide a one-fits-all processing system and morphological features of the signal are to be analyzed, signals have to be normalized in order to guarantee a more reliable classification. The current work’s real-time aims at standardizing the signal in order to have its peaks around 1 mV.

For this, the following steps were adopted (see also Algorithm 1): take a window of the signal with no overlap; detect the window’s maximum and consider it the standardization parameter; take the closest power of two of the normalization parameter and divide the values of the next window by it; check if the maximum of the window is within a desirable band and, if not, updates it.

The presented algorithm comprises an initialization stage, which considers that the first window of the signal is not normalized, only being used to generate the first normalization parameter. The constants 0.3125, 1.25 and 0.7188 are amplitude thresholds in mV of the normalized signal, and are the result of a trial-and-error optimization process. The $currMax$ update law takes an average between the previous value and the new one as to perform a smoother transition between the amplitudes.

**Algorithm 1 Signal Normalization Algorithm**

1: **Initialization:**
2: currMax ← $\max(|signal(1 : \text{windowSize} - 1)|)$
3: newMax ← currMax
4: for $i = \text{windowSize} + 1 : \text{windowSize : signalLength} - \text{windowSize}$ do
5:  signalWindow ← signal($i : i + \text{windowSize} - 1$)
6:  tempMax ← $\max(|signalWindow|)$
7:  if tempMax > 0.3125 then
8:     newMax ← tempMax × closestPowerOf2(lastMax)
9:     end if
10: lastMax ← currMax
11: if $\max(|signalWindow|) > 1.25$ or $\max(|signalWindow|) < 0.7188$ then
12:     currMax ← 0.5currMax + 0.5newMax
13: end if
14: end for

Applying this algorithm to a sample with most of its peaks around 0.2 mV, the effect of the size of the window can be seen in Figure 4.5. As it can be seen, with a window of 1000 ms the signal keeps a shape almost identical to the original one but with its amplitude around 1. Windows bigger than 1000 ms were tested, but the transitions between normalization parameters were too slow, thus not being shown in this work.
4.3 Features Extraction

Once the signal is filtered and normalized, the next step is to extract the features to be used in classification. For software analysis, the annotations of the database were used to get the segments of signal around each peak, as is standard procedure [32]. However, the hardware implementation will contain a peak thresholding and a template matching strategies for beat detection, as described in Section 5.4. The heartbeat detection was not implemented in the workflow of the software implementation since the aim is to detect the beats as in the annotations. Therefore, only small functional tests were made in software for that part, not being worth of elaboration.

It was considered that the standard processing window would take the samples 250 ms before the peak and 375 ms afterwards.

4.3.1 Dynamic Features

Dynamic features were considered in this work, namely the ones used by Ye et al [34] and described in Subsection 3.1.3.C. Thus, these features are easily derived from each annotated beat by taking for each beat

\[ RR_{prev}(n) = \frac{\text{peakPosition}_n - \text{peakPosition}_{n-1}}{F_s}, \]  

(4.1)

where \( F_s \) is the sampling frequency, \( \text{peakPosition} \) is an array with the location of the peak of each beat in samples, and by storing these values in an array. Each of the entries of this array is the the first dynamic feature for each beat, and the other two are computed from them as in

\[ RR_{local}(n) = \sum_{i=j}^{n} RR_{prev}(i), \]

(4.2)
\[ RR_{\text{avg}}(n) = \frac{\sum_{i=k}^{n} RR_{\text{prev}}(i)}{n-k}, \]

(4.3)

where \( j \) and \( k \) are the positions of the first RR interval that is at a time stamp of less than 10 seconds and 5 minutes, respectively.

These features are illustrated by Figures 4.6 and 4.7, which combine a closer look on the dynamic features on the first figure with a broader time frame in the second.

Figure 4.6: Closeup of the filtered and normalized signal and the corresponding Dynamic Features of InCarTDb Recording 1.

Figure 4.7: The first 16.7 minutes of the Dynamic Features of InCarTDb Recording 1.
4.3.2 Heartbeat Resampling and Pruning

Similarly to the amplitude variability, ECG signals also present considerable dynamic variations. Since heart rates can range from 30 to 180 bpm or more, the P-QRS-T waves can be seen in windows that range from just 333 ms to 2000 ms. However, it is common among authors [1][32][34] to use a window of 600 to 850 ms (always with 33.3% of the samples before the peak, and the remaining 66.7% after) to consider the segment. This means that for higher heart rates there may appear waves from the adjacent beats inside the window, or for lower heart rates that some of the waves are not fully captured. Therefore, and considering that classification requires some time stability in terms of positioning of the features, resampling of some segments becomes necessary. In software, this process was implemented through the Matlab inbuilt resampling functions, and the best results were given for frequencies equal to 0.8 Sampling Frequency (Fs), if the heart rate is below 60 bpm, and to 1.2Fs if is above 133 bpm. A sample-by-sample resampling process was also developed to aid the design of the hardware implementation of this block, being it described in Subsection 5.4.4.

Figure 4.8: 0.8Fs resampling of a segment with an incomplete T wave.

Figure 4.9: 1.2Fs resampling of a segment with a P wave from the beat after.
The original heartbeats in Figures 4.8 and 4.9 justify the need of this resampling process. An example of a segment where resampling was necessary due to the presence of an incomplete T wave at very low heart rate is shown in Figure 4.8. Figure 4.9 contains the opposite situation, where due to a high heart rate the P wave of the beat after is shown in the segmentation window. The resampling is not applied to a window of 62.5 ms before and after the peak in order to keep the width of the R peak. This is due to the fact that the R peak width does not vary relevantly enough with the heart rate changes to require resampling. Hence, doing so would lead to a misleading classification.

4.3.3 Morphological Features

The implemented extraction of morphological features was also based on the work of Ye et al [34]. Both DWT and ICA was applied to the segments. However, ICA usage was discarded as it did not provide any advantage in the classification stage. On top of that, for the separate usage of each method, the DWT features rendered better accuracy. These features are a concatenation of the 3rd and 4th order details and 4th order approximation coefficients obtained from applying the DWT with DB8 to each segment. The number of coefficients depends on the sampling frequency, as does the segment size. An example of these coefficients for a signal sampled at 257 Hz is shown in Figure 4.10. The coefficients shown in this figure were obtained with the inbuilt functions of Matlab. However, a polyphase filter based DWT was developed and the exact same results were obtained after adding symmetric mirroring of the first and last eight coefficients of the input of each filter. This is necessary due to a property of FIR filters, which do not provide a reliable output until the input buffer is filled with a number of samples equal to half of the order of the filter. For the first or second stage of the DWT this is not a serious issue, but for the third and fourth stages the amount of features is significantly lower the features become almost irrelevant at that point. These features are then put through PCA. An analysis of the most suitable number of principal components that can be used for the classifier is performed in Chapter 6.

![Figure 4.10: A pre-processed segment and its corresponding DWT 3rd and 4th order Details and 4th order Approximation coefficients.](image-url)
4.4 Classification

Once the feature set is ready, classification is the last remaining stage. To promote an inter-patient paradigm, the annotated heartbeats’ features should be separated by subjects, and the subjects divided into training and testing clusters. The PNN classifier used in the tests was the inbuilt Matlab PNN which uses the radial basis function

\[ r_{\text{radbaf}}(n) = e^{-n^2}. \] (4.4)

The PNN was also fully implemented step by step as described in Subsection 3.1.4 in order to verify the expected behavior of the hardware implementation, which is described in Chapter 5.

4.5 Summary

Throughout this chapter the different stages of the full ECG processing and heart diseases classification implemented in software were described. The described ECG processing system follows the steps described by Algorithm 2.

Algorithm 2 Full ECG Processing and Heart Beat Classification

1: The baseline is removed by subtracting a 500 ms moving average window to the signal;
2: The signal is filtered with an 8th order Chebyshev Type 2 [IIR] low-pass filter with cut-off frequency of 45 Hz and 80 dB of stopband attenuation;
3: The filtered signal is normalized to have peaks with an amplitude around 1 mV, by dividing the values of each window of 1000 ms by a normalization parameter;
4: From the annotated peaks location, a window of 250 ms before each peak and 375 ms afterwards is taken;
5: The dynamic features of each peak are derived from the RR intervals and the segmented heart-beat is resampled to 0.8Fs if the heart rate is below 60 bpm and to 1.2Fs if is above 133 bpm;
6: The morphological features are obtained by taking the 3rd and 4th order details and 4th order approximation coefficients obtained from applying the DWT with DB8
7: PCA is applied only to the morphological features for dimensionality reduction;
8: The reduced morphological features are concatenated with the dynamic features;
9: The recordings are randomly separated between a training and a test set;
10: The features are clustered, having the training cluster being fed into a PNN classifier, and the testing cluster verifying its performance.
ECG Processing Hardware Architecture

Section 3.2 described the proposed methodology for implementing the heart disease detection system. Chapter 5 presents the hardware architecture for the blocks that implement the described functionalities. This development was aided by the usage of Matlab, where every function was decomposed into hardware synthesizable arithmetic and tested. The architecture was then described in VHDL in Vivado 2016.4 with the purpose of being implemented on an FPGA system.

In Section 5.2 the data input and output flow for the testing is described. Section 5.3 contains the description of the pre-processing blocks for filtering and normalization of the signal. The dynamic and morphological features extraction and dimensionality reduction blocks are described in Section 5.4. The implementation of the PNN classifier is contained by Section 5.5. In the implementation of all the blocks the parameters were described with generics such that their actual implementation depends on the defined settings during the synthesis process. This makes the architecture programmable in several different fields: the format of the I/O data, the filter sizes and types (either FIR or IIR) for both the low-pass filtering and the DWT stage, the window sizes for the baseline filtering, normalization, segmentation, the templates to be used in the template matching, the PCA coefficients and number of principal components, and even the structure of the PNN classifier.

5.1 Overview

The sampling frequency of an ECG is rarely above the 500 Hz. Since both ASIC and FPGA technologies usually operate in the MHz and GHz range, a multiple-cycle architecture was preferred to a pipelined one as to reduce the resource utilization. This architecture is structured in the following way: a main circuit contains the Control Unit, the sub-blocks of the Datapath Unit and the memory
blocks to input data and store the circuit output. BRAMs were used for the signal input and output since the interface with signal acquisition and post-processing structures was not established. In a physical context, the architecture requires storage units between the acquisition setup and itself, as will be described ahead in this section. The dataflow of the architecture is illustrated in Figure 5.1. For this figure, every single-lined box represents a processing block that encompasses control and support logic, storage units and Datapath sub-units, which will be described in the coming sections. The double-lined boxes represent the storage BRAMs and the cell-vectors represent shared storage arrays that allow the transition from one block to the next. Throughout the remaining figures of this chapter, the double-line boxes in schematics always represent storage units, and for the cases where the type of unit is not specified it should be considered a register. The overall processing flow goes as follows:

1. The samples are processed in sets of variable size. Each new set of samples is streamed from the input memory block and stored in a FIFO array of 500 ms that may have previously read samples. This array is henceforth called a window.

2. The samples of this window are put sequentially through the Moving Average baseline filter to be high-pass filtered (see also Figure 4.1) and stored in a new FIFO window of the same size.

3. When this step is done, the samples are low-pass filtered sequentially through the IIR filter and stored in the same array during the intermediate stages.

4. During the final stage of the IIR filtering, each new sample is directly put through the amplitude normalization process and stored in a third FIFO window with a fixed size for processing. Every normalized sample is also immediately written to memory for usage outside the architecture, if needed.

5. For each new normalized sample, this last window is put through the template matching process, where the cross-correlation of the window with a heart disease template loaded from memory is calculated.

6. If the cross-correlation is above a certain threshold, it is considered a match and the processing moves to the feature extraction.

7. If not, the sample is analyzed by the peak detection block. Here, when a peak is detected, the block will ensure that more new samples are stored after the peak in the normalized samples window. In all cases, this requires that the set of samples finishes pre-processing and that new samples are read from memory and pre-processed. The number of new samples required is calculated as to place the peak sample in a specific position of the window, more precisely at the point corresponding to 250 ms as in Section 4.3. When no peak is detected, a default number of new samples is read from memory and pre-processed.

8. The feature extraction starts either when a peak is in the right place of the normalized window, or there was a template match. In both cases the processing is the same and it disables the
reading of new samples from the input memory. The dynamic features are calculated and stored in an array which will contain the final features for the classifier. The features are also outputted to a BRAM since they contain relevant information on the heart rate.

9. Depending on the value of one of the features (the instantaneous RR interval), the normalized window may be up-sampled, down-sampled or left as-is.

10. After the resampling block, the DWT analysis coefficients are extracted as morphological features and stored in an array.

11. This array is fed into the Principal Component Analysis for dimensionality reduction and the outputted features are joined with the dynamic features in the final features array.

12. The PNN classifier is fed the final features and outputs the classification of the processed heartbeat, which is then written to an output BRAM. When the classification is finished, the processing of new samples is resumed, either from reading the signal from the initial BRAM or from the point where it stopped after the template match.

![Figure 5.1: Functional architecture of the end-to-end dataflow in the implemented circuit.](image)

5.1.1 Control Unit and Datapath

The Control Unit commands all the data transfers and processes. It works as a sequential state-machine, and it is composed of independent blocks of smaller state-machines. These parts interact with each other by using flag signals. These enable and disable its functioning and access to the shared storage units, both inside and outside of the Control Unit. With the exception of the Enable
Read block, all single-lined boxes of Figure 5.1 are controlled by an independent state-machine inside the Control Unit.

For each processing stage, the Control Unit interacts with the storage and the Datapath units of the circuit, here represented by Figure 5.2. The input of raw signal and the architecture’s outputs are received and sent by the Control Unit.

For all stages of the processing except the Amplitude Normalization and the Resampling, there is a corresponding Datapath sub-unit. These sub-units contain mostly arithmetic functions which are dedicated to process both the inputs from the Control Unit and from the necessary memory blocks. In particular, the PNN is decomposed in several blocks, all similar to each other, for parallel processing of different inputs. In this architecture, the number of PNN blocks is equal to the number of classes of the model.
5.2 Data I/O

To develop an architecture, one of the first considerations made should be the format of the data to be used. Since ECG processing systems do not require extremely high dynamic range and it would increase the complexity of the hardware, the floating-point format was dismissed, with fixed-point format being used instead. Being the purpose of this architecture to be configurable for as many setups as possible, the integer and fractional parts of the format can be defined through generic parameters prior to the synthesis. In the results Section 6.3.1 an analysis of the most suitable fixed-point format is presented for a specific context.

5.3 Pre-Processing

As in the software implementation, the ECG signal pre-processing is performed in three stages: baseline removal with the subtraction of the moving average filtered signal, high frequency noise filtering with an IIR low-pass filter, and the amplitude normalization. However, this last stage’s main processing unit is contained inside the peak detection state-machine, thus being described in Subsection 5.4.1. In this section and the next, the window sizes are described as a function of the number of samples in one second \((samps_{Fs})\), for a generic sampling frequency. The values of the windows should be approximated to the nearest power of two as to minimize the hardware resources utilization. This enabled significant improvements to the area utilization and time performance by replacing several multiplication and division operations to be replaced by arithmetic shift operations.

5.3.1 Baseline Filtering

The baseline filtering is the first block of the data processing and its main behavior is described and implemented as the state-machine in Figure 5.3 and the Datapath sub-unit of Figure 5.4. The moving average baseline filter is implemented as

\[
Accum(n) = (New + Accum(n - 1)) - Oldest, \tag{5.1}
\]

\[
Out(n) = \frac{Accum(n)}{WindowSize} - Current, \tag{5.2}
\]

where the \(Accum\) register accumulates the sum of the samples of the window around the that is being processed, \(New\) is the sample most recently read from memory, \(Current\) is the sample currently being filtered, and \(Oldest\) the first sample of the window that is now being removed. Each \(Current\) sample is subtracted the average of this window, given by the division of the \(Accum\) value by the number of samples of the window, hence the signal baseline is removed.
As is customary for error prevention purposes, the state-machine starts with an Initial state, which waits for the system reset flag to be turned off. Once this happens, the reading of the signal from the BRAM is enabled and the Load state begins. Here, a FIFO array of samples stores the first 0.5\(samps_{Fs}\) readings from the BRAM. This value corresponds to 500 ms of signal, the time window chosen for the baseline filtering in Subsection 4.2.1. As these values are being read, they are also being summed and accumulated in the Datapath. When the array is full, the state machine enters the Processing state. As the accumulator register already contains the sum of the window’s samples, this value is put through an arithmetic shift right unit which implements the division operation. The number of shifted bits is given by

\[
\text{WindowSizeBits} = \log_2(\text{WindowSize}).
\]

The current sample is then subtracted to this value. For every iteration, the sample being filtered (current) is the one on the center of the window, the sample at position 0.25\(samps_{Fs}\). This means that the first 0.25\(samps_{Fs}\) samples of the signal are not processed. Since this architecture is meant for real-time monitoring during bigger periods of time (several minutes/hours), losing the first 250 ms of signal is not harmful. In the next iterations of the Processing state, the oldest sample of the array is subtracted to the accumulated value before it is removed from the array, and the new one is added. This way, there is no need to repeatedly sum the same values.

When a certain number of samples are filtered, the state-machine moves to the Waiting state, where it waits for those samples to be processed. This number of samples is defined by the Peak Detection state, and will be elaborated in the corresponding subsection. As the processing of those samples ends, the state machine returns to the Processing state. This loop is only broken when the system Finish flag is on. In this scenario, the state machine will enter the Done state and wait for the system Reset flag to go on. When this happens, it will revert to the Initial state.
5.3.2 Low-pass Filtering

Similarly to the Baseline Filtering, the Low-pass Filtering state-machine (Figure 5.5) starts with an Initial state, which also waits for the system Reset flag to be turned off. When this happens, the Loading state is set and the IIR filter coefficients are loaded into a 2-D array, where each line contains the six coefficients of each second-order section. The number of second-order sections is defined before synthesis in the System Settings. It should be noted that the difference between implementing an FIR and an IIR filter lies only on the former having the coefficients corresponding to the determinant of the transfer function to be equal to zero. Since this loading process is faster than the Baseline Filtering Loading and Processing tasks, the IIR state-machine enters a Waiting state. As soon as the Baseline Filtering state-machine enters the Wait state, the Low-pass filtering begins.

Figure 5.5: Flowchart of the IIR Low-pass Filtering state-machine.

Figure 5.6: Low-Pass IIR Filtering Datapath sub-unit.
Figure 5.6 illustrates the main arithmetic structure of the IIR filter in the Datapath. It is composed of two adders and two multipliers, as well as supporting multiplexers and pipeline registers. This block was developed from the Direct-form 2 structure in Figure 3.3 using ASAP (As Soon As Possible) scheduling and operator sharing accordingly. Pipeline registers were added for this purpose, being the longest critical path composed by the data flow through a multiplexer, a multiplier and an adder. This block reveals a compromise between area utilization and performance. Reducing the quantity of arithmetic units would increase the number and size of support multiplexers and registers, as well as increase the latency of the block. Due to the sequential dependency of some arithmetic operations, the circuit latency is of 3 clock cycles per filtered sample, which is the latency best possible for the Direct-Form 2. To each of these three clock cycles corresponds a state of the IIR state-machine: Stage 1, Stage 2 and Stage 3. For each of these states, control signals are fed into the Datapath sub-unit in order to enable and disable the registers and to choose the appropriate multiplexer channels. The arithmetic operations and storage of values is performed to prevent overflows, but the inputs of every operation and the output are truncated to the original signal's input fixed-point format.

One concern to bear in mind during filtering of windows of samples is the filter’s continuity, which can cause signal distortions. This phenomenon is prevented by the $Z_n$ arrays of registers at the center of Figure 5.6. These are two arrays with a number of entries equal to the number of second-order sections, and they contain the delayed $v(n)$ sum (Equation 3.33) at the end of each stage, which are used on the next time that the corresponding second-order section is used. This ensures the continuity of the signal.

Whenever the filtering is in its first stages, states Stage 1 to Stage 3 run in a loop. However, when it reaches its last second-order section, for every new sample the state machine will wait for the Template Matching block to process is task. When this task is finished, it either goes back to Stage 1, if there are still samples to be processed. If not, it will enter the Check Done state, where the arithmetic units are reset and the flags that activate the next processing blocks are enabled. If the system Finish flag is enabled by this point, the state-machine enters the usual Done state. Otherwise, it will revert to the Wait For Baseline Filtering state.

5.4 Features Extraction

After the IIR filtering, the signal enters the processing phase with the aim of extracting features for classification. This can only be done by identifying the heartbeats in the signal, and two methodologies that work in parallel were employed: Peak Detection and Template Matching. As soon as one of these detect a heartbeat, features are to be extracted from it. Depending on the dynamic features, the segment might require resampling, thus justifying the morphological features’ extraction to be performed after the dynamic features, and not before or in parallel.
5.4.1 Peak Detection and Signal Normalization

As the Peak Detection block does not require reading from memory, the Initial state of the state-machine is used to reset all the support and storage signals, while waiting for the flag that marks the end of the Baseline Filtering.

Figure 5.7: Flowchart of the Peak Detection and Signal Normalization state-machine.

The Loading state waits for the samples to go through IIR filter and the Template Matching block, which is elaborated in Subsection 5.4.2. For the Peak Detection, when the filtered samples are outputted from the IIR filter Datapath sub-unit, they are compared with a predefined threshold. The first sample to exceed it becomes the new maximum and has its amplitude and index stored. During that window’s run, if a higher value comes up, it becomes the maximum.

After the window is fully ran, the decision making is done in the Check state. It is here that will be decided how many samples are to be read from memory in the next run. As defined in Section 4.3, a heartbeat segment is required to consider the samples 250 ms before and 375 ms after the peak. However, due to the needs of the resampling process described in Subsection 5.4.4, this array is extended by \( ResampDownBefore \) ms before and \( ResampDownAfter \) ms after the intended heartbeat segment array. These two parameters are defined as a generic prior to synthesis. Therefore, the peak needs to be stored in the position corresponding to \( 250 + ResampDownBefore \) ms in the array. With this knowledge, the filtering runs will be, at most, \( 375 + ResampDownAfter \) ms long. This way, it is made sure that, if a peak is detected, there are still enough samples before the peak in order to have the whole required segment. When a peak is detected in a window, the next run will read and filter only the amount of samples necessary to place the peak sample in the required position of the array. If a higher peak is detected in this new run, it becomes the real peak and a third run is performed. This makes sure that the detected peak is the real maximum of the segment, while making sure the peak afterwards is not detected in this process. This process may require resetting some support signals and storage units, thus requiring some accesses to the Init state. Whenever the peak is set in place, the state-machine goes to the Done state, where it waits for the peak to be processed. As soon as it is, it reverts to the Init stage, where the support signals and storage units are reset.

In parallel to this process, during the Load state the Signal Normalization algorithm described in Subsection 4.2.3’s Algorithm 1 is implemented through an extremely similar process. This means that it waits for the first peak to be detected and uses its closest power of two to divide the samples of the next windows. Hence, each output of the IIR is divided by the power of two through an arithmetic shifter fed with the exponent of that power of two. As in the algorithm, this value is constantly adapted through exactly the same control law. It is the result of that operation that is used by the Peak Detection
and stored in the Normalized Samples Window of Figure 5.1. This normalization process allows for
the usage of a fixed threshold for the peak detection.

5.4.2 Template Matching

Every time a sample ends its filtering through the IIR filter, the whole window is put through the
Template Matching block. Here, the samples are cross-correlated with a template in order to check
the similarity of the waveform. The discrete instantaneous cross-correlation value is given by

\[
\text{CrossCorrelation} = \sum_{i=1}^{N} x[i] \cdot t[i],
\]

where \(x\) is the segment array, \(t\) is the template array and \(N\) is the segment size. In this case, the
segment array corresponds to the 625 ms window with the peak at 250 ms position, which means
that Template Matching disregard the signal extensions added in the Peak Detection and Signal Nor-
malization process.

This block’s state-machine, illustrated in Figure 5.8, starts with the customary Initial state, which
simply waits for the system Reset flag to be turned off.

\[\text{TM_Init} \rightarrow \text{TM_Load} \rightarrow \text{TM_Process_Wait} \rightarrow \text{TM_Process} \rightarrow \text{TM_Check_Wait} \rightarrow \text{TM_Done} \rightarrow \text{TM_Check} \rightarrow \text{TM_Load}\]

Figure 5.8: Flowchart of the Template Matching state-machine.

Template Matching Multiply Accumulate Unit

Sample \rightarrow MUL \rightarrow ADD \rightarrow ACCUM \rightarrow Out

Figure 5.9: Template Matching Multiply Accumulate Datapath sub-unit.

Then, the template coefficients are loaded from memory into an array which will be of the same
size as the desired segment window, 625 ms. Before starting the processing, the state-machine has
a Waiting state which waits for a new pre-processed sample. As soon as it arrives, the Process
state will resort to the Multiply Accumulate block in the Datapath (Figure 5.9) to compute the cross-
correlation between the template and the segment with the new sample. Privileging area utilization
over performance, only one Multiply Accumulate block is used. For low clock frequencies, this may be
an issue, but for the defined clock frequency it did not compromise the system, as will be discussed in Chapter 6. Once the cross-correlation is computed, a Wait state is implemented just to guarantee the arrival of the Multiply Accumulate block output to the Control Unit. In the Checking state, this output is compared with a fixed threshold previously defined from the analysis of software simulations of the cross-correlation, being equal to 3. If the output is higher than the threshold, the maximum is set, the state machine moves to the Done state and back again to Waiting for the next pre-processed sample. If the new cross-correlation value is higher than the previous, it repeats this cycle until a lower one is found. When that happens, it is considered that the previous segment was the one containing the desired heartbeat, and is immediately sent for the dynamic features extraction block, disregarding the needed samples for the Peak Detection. This was set this way because since there are heart conditions that cannot be detected by the Peak Detection. Thus, it justifies the need to have the Peak Detection to be activated only after the Template Matching blocks. The state-machine then waits for the heartbeat to be processed, and when it is, goes to the Done state which resets the support signals and registers. This is always followed by a new waiting period for a new sample, unless the system Finish flag is enabled, whereas the state-machine will be sent to the Init state once again.

This template matching process can be used for more than one class in parallel only by adding more Datapath sub-units and some simple control logic at their outputs to verify the results of the matching.

5.4.3 Dynamic Features

The extraction of the dynamic features of the segments is controlled by a state-machine as well, which is illustrated by Figure 5.10.

![Flowchart of the Dynamic Features Extraction state-machine.](image)

Since nothing is required to be loaded from memory, its initial state waits for a heartbeat to be detected by either the Peak Detection or the Template Matching blocks. When one of them flags the detection, the corresponding Stage 1 state is selected, and the global index of the peak/matched sample is extracted. As in the software implementation, three dynamic features are to be extracted: the instantaneous RR interval, the local average of 10 seconds of RR intervals, and the average of 3 minutes of RR intervals. Since the values of the RR intervals can oscillate, the amount of samples
for each feature varies as well. However, the arrays that store the values of the RR intervals need to have a fixed size. The storage arrays are initialized with a heart rate of 70 bpm, which corresponds to the average heart rate of an adult at a resting state.

The processing is performed in the Dynamic Features Datapath sub-unit, but it will not be illustrated since its functioning is more based in storage than arithmetic operations. In the Stage1 state, the peak index is used to calculate the instantaneous RR interval as in

\[ RR_{prev} = newPeak - lastPeak, \]

where the \( lastPeak \) is initially zero, and is updated to the position of every subsequent \( newPeak \). In parallel, the oldest RR interval of each average dynamic feature is subtracted to the respective accumulated sum of RR intervals.

In Stage2, the new RR interval is summed to the accumulated sums of the average dynamic features, whereas in Stage3 the division is performed to both of them. As before, this is done through and arithmetic shift right, with the sums being shifted by the exponents of the corresponding powers of two. Every operation contains overflow and underflow prevention bits. The resulting features are finally outputted in Stage4. When they arrive to the Control Unit, the instantaneous RR interval is used to Check if Resampling is needed. The same heart rate bounds of Subsection 4.3.2 were used here: \(<60 \text{ bpm requires down-sampling and }>133 \text{ bpm up-sampling.}\) If any of these conditions is verified, the Resampling block will be signaled for the corresponding process. Either way, the Dynamic Features block will then follow its course to the Done state, which will simply revert back to the Init state.

5.4.4 Resampling

The Resampling is the simplest process of this architecture due to the control process (Figure 5.11) and the absence of a Datapath sub-unit. Its Init state waits for the system Reset to be turned off and Waits for the end of the extraction of the dynamic features. If the instantaneous RR interval verifies a heart rate within 60 and 133 bpm, the 625 ms segment with the peak at the 250 ms position is directly extracted from the extended array of the normalized samples. Otherwise, the state-machine will enter the state corresponding to the flag signaled by the Dynamic Features block, either for up-sampling or down-sampling.

![Figure 5.11: Flowchart of the Resampling state-machine.](image)
In both cases, the new segment has its samples set one at a time. The Up-Sampling state is performed for the heartbeats over 133 bpm, and it will cut the excess samples from both the beginning (ResampUpBefore) and the end (ResampUpAfter) of the segment. This means that from the NormalizedSamplesWindow array, the first sample of the final segment the one in position ResampDownBefore + ResampUpBefore. From there, every subsequent sample is copied to the final segment array. However, for every ResampUpCount samples from the NormalizedSamplesWindow array, the counter is paused and a new sample is added. This sample is the average of the last sample written and the next one, and is equal to their sum arithmetically shifted one bit to the write as to divide by 2. This process is done for the whole array except for the samples 62.5 ms immediately before and after the peak, as to preserve its width. It stops when all the positions of the final segment array are filled.

The Down-Sampling requires the whole NormalizedSamplesWindow array and is the reason why it has the extra (ResampDownBefore) samples before and (ResampDownAfter) afterwards. The final segment arrays starts with the first sample of the previous array. Every (ResampDownCount) samples it skips one sample, as to reduce the length of the waveform to the desired size of regular 625 ms window. Once again, this is performed for the whole array with the exception of the 62.5 ms samples immediately before and after the peak.

Once one of these processes is complete, the state-machine enters the Done state, signals the end of the resampling to the Morphological Features extraction block and reverts to the Init state.

5.4.5 Morphological Features

After the resampling, the heartbeat segment is finally ready to be processed. The morphological features extraction is performed as it was implemented in software, through the DWT. The process illustrated by Figure 5.12 is initiated by the usual wait for the system Reset to be turned off. When it is, the FIR filter coefficients are loaded to an array. It then enter an initial Wait state, where it waits either for the end of the Dynamic Features block or the Resampling block. The detection of the new heartbeat leads to an initial segment extension at the Initial Extend Segment state, where the first Mirror samples and last Mirror samples of the segment are mirrored into the beginning and ending of the segment. This Mirror value corresponds to the filter order minus two samples, one for the even filter and other for the odd filter. Since the size of the mirroring is defined previously to the synthesis, this can be mapped to be performed in one cycle. Before processing the mirrored segment, the system requires a cycle for the data to be available, and then it enters the processing.

The polyphase filter based design was chosen for this architecture, and the corresponding Datapath processing sub-unit is composed simply by a multiply accumulate block with accumulation registers for the four sub filters. This architecture is illustrated by Figure 5.13. It is considered that both the details coefficients and the approximation coefficients are to be extracted. Filtering is done for each sample of the subsequently, and the starting iteration includes all the first N_filter_size samples of the mirrored segment. The first sample is considered to be odd and the approximation coefficient is required, so the low-pass odd filter is the first to be applied.
Each multiplication between the eight first odd samples and the coefficients is calculated sequentially and accumulated in the low-pass odd register. When this is done, the accumulator register and other support signals are cleaned in the Low-pass Odd Clean state. The odd and even parts of each filter work alternately for both the low-pass and high-pass filters. Therefore, the processing is performed in the same way for the even samples, having the accumulated value being stored in the low-pass even register of the Datapath sub-unit and the cleaning state afterwards. Since each order is calculated only from the approximation coefficients, for setups where there is no need to extract the details coefficients for a specific order those coefficients are not calculated. This leads to the low-pass filter always being used to filter the whole segment before the high-pass filter.

The Check Out state will verify in which stage of the filtering the process is in and direct it to the intended state. While the low-pass filtering of a segment is incomplete, it activates a flag to perform the sum of both low-pass accumulation registers at the Datapath sub-unit in order to output the approximation coefficients. It will also return to the respective odd filtering state. When the low-pass filtering is done, it will require a Waiting cycle to retrieve the last sum and it will enter the Segment Extension state. This state is separated from the Initial one because here its not possible to directly map the symmetries to the array that is inputted to the low-pass. The symmetry in the beginning of the array can be mapped directly, but since the number of approximation is progressively smaller, the starting point of the symmetry is always different. Therefore, this was solved by writing each of the Mirror$_n$ symmetric samples sequentially. Back to the Check Out state, if the system is processing the an order for which the details coefficients are required, an exactly analogous process is started for the high-pass filter, which is performed for the whole segment as well. From here result the details coefficients.

Every time one of the coefficients is calculated, it is stored in an array that will be used in the Principal Component Analysis block. This block is enabled by a flag at the Done state, which then reverts back to the Wait state, unless the system Finish flag is on, where it then reinitializes the
5.4.6 Principal Component Analysis

The implementation of the PCA in this architecture considers that a principal components coefficients matrix has already been obtained previously to the processing when generating the model. Therefore, this PCA block puts the morphological features through the matrix in order to obtain its principal components. After the Initial system Reset wait, the state-machine (Figure 5.14) is immediately put on hold until the features are made available by the Morphological Features Extraction block. The coefficients are not loaded into any array because since each coefficient is used only once for each segment and in a sequential order, it is straightforward to read them directly from memory with no loss of performance and thus saving area utilization. When the state-machine enters the Wait state, the three dynamic features are copied to the last three positions of a final features array.

When Process state is set it activates the PCA Multiply Accumulate Datapath sub-unit, which accumulates the inner product of the features array and each principal component coefficients matrix row. For every inner product accumulated, the value is outputted to the final features array at the Process Accumulate state. The Process Clean state resets the Accum register of the Datapath sub-unit to zero, and if there are still principal components to be calculated it reenters the Process state. If not, this means the final features array is complete and at the Done state the Classification block is

![Figure 5.13: DWT Filtering Datapath Sub-Unit.](image)

![Figure 5.14: Flowchart of the Principal Component Analysis state-machine.](image)
flagged for activation. Until the system Finish flag is on, where the Initial state would be restored, the state-machine always reverts to the Waiting state. For this stage the number of principal components to be calculated is defined by a generic before synthesis.

![PCA Multiply Accumulate Unit](image)

**Figure 5.15:** Principal Component Analysis Multiply Accumulate Datapath sub-unit.

### 5.5 Classification

The final block of this architecture is the heart disease classifier. As in the software implementation, a [PNN](#) was developed for this purpose. One particularity of this Neural Network is that it does not need a back-propagation to generate the classifier weights. These are the feature vectors of the training set and, therefore, the size of the classifier depends directly on its amount of training vectors. Since this amount can be extremely variable, a scalable architecture was designed in order to allow the usage of a training set of a wide range of sizes. For the classification process, the features to be classified are compared with the training features, processed and accumulated in class specific registers. This means that most of the classification process is done independently for each class. Therefore, a generalist Datapath sub-unit was designed to process each class, having a corresponding memory that contains the training set for that class. It allows for full scalability with the performance to be dependent mostly on the number of training vectors, which must be the same for all classes if a balanced neural network is intended.

The classification process requires the usage of three memory blocks: for the input weights, for the bias coefficients and for the exponential function. Since classes are separated, it was established that each class had a distributed memory block of its own for the input weights. Since the exponential function is needed for negative values, implementing a Taylor-series based calculation would require the usage of arithmetic divisions to obtain its inverse. Hence it was implemented through a table stored in a [BRAM](#). This memory is accessed sequentially by each class’s Datapath sub-unit. That decision can compromise a bit of the performance if there are several classes, but having one exponential table per class would impact the area utilization tremendously. It was defined that the bias coefficient memory had just one value since the bias value is always constant in each of the attempted, and also in most practical usages of neural networks in general. In any case, the functioning of the control unit block for classification supports variable bias coefficients being loaded from a memory block.

The control of the classification process is done by a state-machine as in Figure 5.16.
Considering that all coefficients are read from memory, no loading stage is needed. Therefore, from the Initial state, which waits for the system Reset to be disabled, the state-machine goes directly to Waiting for a new feature array to begin the classification.

The Datapath sub-unit generalist block for each is the "Neuron" illustrated by Figure 5.17 was developed. It is structured as a 4-level pipeline with one more register for selection purposes. The arrival of new features enables the Processing stage. For each training features array, the City-block (Manhattan) distance is calculated and, in this methodology, divided by four through a 2-bit arithmetic shift-right operation in order to reduce the order of the distances. This step was introduced in order to reduce the dynamic range of the distances. The distances are then multiplied by the bias, and accumulated on a register. When the accumulation is finished for that training array, the Exponential state ensures that the accumulated value is outputted by the Neuron as an address to the exponential memory. The fact that the distances were divided by four means that more of the accumulated values are inside the bounds of the values of the exponential table. If this was not done, too many values would be over that value, for which the result of the exponential is considered to be zero. These two states are repeated until there are no more training features arrays, and every exponential value that resulted from them is accumulated in a register of the Neuron.

After a clock cycle of Waiting to make sure all accumulated values are ready to be outputted, the Classification stage will verify which of the Neurons contains the biggest accumulated value. The maximum of this process is considered to be the class of the segment and is outputted to a BRAM. This conclusion will make the PNN reset its support variables, flag the Baseline Filtering block to resume its activity, and then return to waiting for a new feature set. However, if the system Finish flag is activated, it reverts to the Initial state.
5.6 Time Constraints

As seen throughout this chapter, the processing of the signal is not done through a continuous stream but through batches of data. For different systems, there can be both power and performance limitations, and the frequency of the main clock greatly impacts both parameters. Therefore, a general guideline for the choice of the clock frequency is presented.

The clock frequency takes several parameters into consideration: the execution time of the baseline \( t_{MA} \) and low-pass \( t_{IIR} \) filtering, template matching \( t_{tempMatch} \) and peak detection \( t_{peakDet} \) for each sample, the processing and classification of a detected peak/match, the sampling frequency and the heart rate. When no peak/match is detected and the samples are being processed, every sample should be processed in \( 1/Fs \) seconds. This translates as

\[
t_{MA} + t_{IIR} + t_{tempMatch} + t_{peakDet} < \frac{1}{Fs}. \tag{5.6}
\]

However, when a detected peak/match is being processed, no new sample is being read from memory, thus having a lag. This lag depends on the parameters of each of feature extraction and classification block. This creates a dependency on the heart rate. If no sample is read during the peak/match processing, then after the processing finishes the system must ensure that all stored samples are read before the next peak. In case this condition is not verified, the lag of the processing will consistently increase. Therefore, between two RR peaks at a certain heart rate the condition

\[
t_{peakProcess} + t_{RR}Fs(t_{MA} + t_{IIR} + t_{tempMatch} + t_{peakDet}) < t_{RR}, \tag{5.7}
\]

with

\[
t_{RR} = \frac{60}{\text{heartRate}}, \tag{5.8}
\]

should be verified. However, except for \( t_{RR} \), all times in Equation 5.7 depend on the systems clock frequency. Therefore, the equation can be written in function of clock frequency as:

\[
\frac{S_{peakProcess} + t_{RR}Fs(S_{MA} + S_{IIR} + S_{tempMatch} + S_{peakDet})}{F_{c}} < t_{RR}, \tag{5.9}
\]

where all \( S \) parameters are the number of samples of execution of each block and \( F_{c} \) is the clock frequency. Therefore, the clock frequency should verify

\[
F_{c} > \frac{1}{t_{RR}}S_{peakProcess} + Fs(S_{MA} + S_{IIR} + S_{tempMatch} + S_{peakDet}). \tag{5.10}
\]

On another hand, the functioning of the architecture when implemented is not instantaneous, having both logic and path delays. This limits the speed at which the operations are performed, thus setting a top boundary to the clock frequency. These delays are dependent on the implementation specificities, therefore an analysis of the critical path should also be performed in order to choose the most suitable clock frequency. The clock frequency is then bound as in

\[
\frac{1}{t_{RR}}S_{peakProcess} + Fs(S_{MA} + S_{IIR} + S_{tempMatch} + S_{peakDet}) < F_{c} < \frac{1}{t_{criticalPath}}. \tag{5.11}
\]
5.7 Summary

This chapter presented an analysis of the different blocks that make up the implemented architecture. Each subsection contains an overview of their functioning and the reasoning behind several design decisions. The architecture is mainly divided in Control and Datapath units, with input and output memory structures. All blocks have their processes controlled by a state-machine and most of them have a Datapath sub-unit where most arithmetic and logic operations were performed. The blocks follow a defined sequence, which is represented by Figure 5.1 and they interact with flagging signals for activation and deactivation. The blocks are independent and most of their parameters are fully programmable, from the size and type of the filters in different blocks, to the number of principal components to be used in classification, or even the number of classes and training arrays to be used in the PNN classifier. This programming is done prior to synthesis by changing the values in a Settings library.

Since the architecture has been described, the following chapter will present and analyze the obtained results.
Experimental Results

Having the software and hardware methodologies been described in the previous chapter, a particular case of their implementation is here described. Firstly, an overview of the standard databases and classifier evaluation parameters are described in Section 6.1. That section contains also an analysis of the database that was used throughout this work. From the standard databases, this was the only one that had annotated Lead 1 recordings, hence being the only option to validate that the architecture is fit for Lead 1-only processing, as initially proposed. However, both the software and hardware methodologies are lead-agnostic, as nothing binds them to be used only with Lead 1, any other lead could be used. The software results are described in Section 6.2 with the most important test results at each stage and their repercussions on the final classification. Section 6.3 contains an analysis of the final architecture’s data I/O format, the specificities of its structure, its time constraints, power consumption, hardware utilization and signal classification quality for the chosen database.

6.1 Databases and Evaluation Standards

In order to provide a standardized data workspace and methodology for a fair comparison of ECG processing methods, the Association for the Advancement in Medical Instrumentation (AAMI) developed the ANSI/AAMI EC57:1998/(R)2008 protocol [32]. This protocol starts by recommending the usage of five databases: The Massachusetts Institute of Technology - Beth Israel Hospital Arrhythmia Database (MIT-BIH) (48 records of 30 min each); The European Society of Cardiology ST-T Database (EDB) (90 records of 2 h each); The American Heart Association Database for Evaluation of Ventricular Arrhythmia Detectors (AHA) (80 records of 35 min each); The Creighton University Sustained Ventricular Arrhythmia Database (CU) (35 records of 8 min each); The Noise Stress Test Database (NST) (12 records of ECG of 30 min each, plus 3 records with noise excess). The common de-
nominators to the usage of these databases are the absence of patients with pacemakers and that segments with either ventricular flutter or fibrillation should be dismissed. From these databases, the MIT-BIH comes as the most representative and utilized when comparing algorithm performance. The protocol also establishes 15 types of arrhythmias to be detected by devices, all present in the MIT-BIH database, which can be clustered into 5 superclasses: Normal (N), Supraventricular Ectopic Beat (SVEB), Ventricular Ectopic Beat (VEB), Fusion Beat (F) and Unknown Beat (Q). The composition of each cluster is explicit in Table 6.1.

Table 6.1: Superclasses and classes of heartbeats recommended for detection by AANSI.

<table>
<thead>
<tr>
<th>Group</th>
<th>Symbol</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>Normal Beat</td>
</tr>
<tr>
<td>Any heartbeat not categorized as SVEB, VEB, F or Q</td>
<td>L</td>
<td>Left bundle branch block beat</td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>Arial escape beat</td>
</tr>
<tr>
<td></td>
<td>j</td>
<td>Nodal (junctional) escape beat</td>
</tr>
<tr>
<td>S</td>
<td>A</td>
<td>Atrial premature contraction</td>
</tr>
<tr>
<td>Supraventricular Ectopic Beat</td>
<td>a</td>
<td>Aberrated atrial premature contraction</td>
</tr>
<tr>
<td></td>
<td>J</td>
<td>Nodal (junctional) premature beat</td>
</tr>
<tr>
<td>VEB</td>
<td>S</td>
<td>Supraventricular premature beat</td>
</tr>
<tr>
<td>Ventricular Ectopic Beat</td>
<td>V</td>
<td>Premature ventricular contraction</td>
</tr>
<tr>
<td>Fusion Beat</td>
<td>E</td>
<td>Ventricular escape beat</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Fusion of ventricular and normal beat</td>
</tr>
<tr>
<td>Q</td>
<td>P</td>
<td>Paced beat</td>
</tr>
<tr>
<td>Unknown beat</td>
<td>f</td>
<td>Fusion of paced and normal beat</td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>Unclassifiable beat</td>
</tr>
</tbody>
</table>

In order to evaluate the quality of segmentation methods, the following measurements should be taken: Sensitivity, Positive Predictivity Rate (PPR) and False Positive Rate (FPR). These metrics are calculated for each class as

\[
\text{Sensitivity} = \frac{TP_{seg}}{\sum_{seg}},
\]

\[
\text{PPR} = \frac{TP_{seg}}{TP_{seg} + FP_{seg}},
\]

\[
\text{FPR} = \frac{FP_{seg}}{TP_{seg} + FP_{seg}},
\]

where \(TP\), \(FP\) and \(FN\) represent, respectively, the number of heartbeats correctly segmented, the number of segmentations that do not correspond to annotated heartbeats, and the number of annotated heartbeats that were not detected.

Likewise, to evaluate the quality of classification methods, the previous three metrics are calculated for each class as

\[
\text{Sensitivity} = \frac{TP_{class}}{\sum_{class}},
\]

\[
\text{PPR} = \frac{TP_{class}}{TP_{class} + FP_{class}},
\]

\[
\text{FPR} = \frac{FP_{class}}{TP_{class} + FP_{class}},
\]
FPR = \frac{FP_{\text{class}}}{TP_{\text{class}} + FP_{\text{class}}}, \quad (6.6)

and the Accuracy, which is a global metric, as

\text{Accuracy} = \frac{\sum TP_{\text{class}} + \sum TP_{\text{class}}}{\sum \text{Total}}. \quad (6.7)

However, for the present work neither of the AAMI standard databases can be used, as none contains recordings of Lead I. Thus, another database had to be used. The AAMI standard databases are contained by a larger open source project on data collection of complex physiological signals through standard methods called PhysioNet [48]. In this project, other databases were available, but only one had both Lead I recordings and heartbeat annotations of different diseases: the InCarTDb. Therefore, in the context of this work all of the seventy-five 30 minutes recordings at 257 Hz from 32 different patients were used. These recordings contain, in total, the classes of heartbeats and their number of occurrences described in Table 6.2.

<table>
<thead>
<tr>
<th>Class</th>
<th>Annotation Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Beat</td>
<td>N</td>
<td>150409</td>
</tr>
<tr>
<td>Premature Ventricular Contraction</td>
<td>V</td>
<td>20013</td>
</tr>
<tr>
<td>Right Bundle Branch Block Beat</td>
<td>R</td>
<td>3174</td>
</tr>
<tr>
<td>Atrial Premature Contraction</td>
<td>A</td>
<td>1944</td>
</tr>
<tr>
<td>Fusion of Ventricular and normal beat</td>
<td>F</td>
<td>219</td>
</tr>
<tr>
<td>Nodal (Junctional) Escape Beat</td>
<td>j</td>
<td>92</td>
</tr>
<tr>
<td>Supraventricular Escape Beat (atrial or nodal)</td>
<td>n</td>
<td>32</td>
</tr>
<tr>
<td>Supraventricular Premature Beat</td>
<td>S</td>
<td>16</td>
</tr>
<tr>
<td>Rhythm change</td>
<td>+</td>
<td>12</td>
</tr>
<tr>
<td>Unclassifiable beat</td>
<td>Q</td>
<td>6</td>
</tr>
<tr>
<td>Bundle branch block beat</td>
<td>B</td>
<td>1</td>
</tr>
</tbody>
</table>

For the classification models in this work, only the first six classes from Table 6.2 were selected, since the remaining classes did not have enough occurrences. Average templates of the heartbeats of these classes were generated from the 75 recordings and are shown in Figure 6.1.

Figure 6.1: Templates of the selected heartbeat classes.
6.2 Evaluation of the heart disease identification procedure

For the validation of the heart disease identification methodology presented in Chapter 4, the software implementation took the 75 recordings and their corresponding annotations of the heartbeats as a starting point. The process described by Algorithm 2 is then performed sequentially. Steps 1-7 are performed once for each recording, having the features being saved in a matrix for later use. No peak detection is applied here, only the annotations of the heartbeats are used. Since only peaks over 0.5 mV and VPCs with a certain cross-correlation value are to be detected by the architecture, modelling a classifier that considers heartbeats outside these bounds makes no sense and would worsen the classification. Therefore, the morphological features are only extracted for the peaks that are compliant with these parameters, being the final number of occurrences for each of the six selected classes the ones in Table 6.3. Appendix A.1 contains a table with the number of occurrences of each heartbeat class discriminated by patient.

Table 6.3: Classes of heartbeats and number of occurrences annotated in InCarTDb after normalization and amplitude thresholding.

<table>
<thead>
<tr>
<th>Class</th>
<th>Annotation Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Beat</td>
<td>N</td>
<td>127639</td>
</tr>
<tr>
<td>Premature Ventricular Contraction (VPC)</td>
<td>V</td>
<td>12195</td>
</tr>
<tr>
<td>Right Bundle Branch Block Beat (RBBB)</td>
<td>R</td>
<td>3146</td>
</tr>
<tr>
<td>Atrial Premature Contraction (APC)</td>
<td>A</td>
<td>1819</td>
</tr>
<tr>
<td>Fusion of Ventricular and normal beat (VFN)</td>
<td>F</td>
<td>148</td>
</tr>
<tr>
<td>Nodal (Junctional) Escape Beat (NEB)</td>
<td>j</td>
<td>92</td>
</tr>
</tbody>
</table>

Subsections 6.2.1 and 6.2.2 show how the accuracy of the classification varies when different parameters are changed. For every parameter to be analyzed, the remaining parameters are the ones used for the final implementation. Hence, for every step analyzed it is shown how the changes affect the final accuracy.

After the features are calculated and separated by patient, the training and test sets selection ensues. There are several ways to use the data to model a classifier. However, the most reliable method in terms of generalization is based on the separation of subjects for training and testing. Since it would not be viable to test all possible combinations of recordings in the training and testing sets, a number of training recordings is set and 100 random combinations tests are performed every time a parameter needs to be validated. This way every recording is used for training and testing several times, more so depending on the number of training recordings. The results of this analysis are presented in Subsection 6.2.3. This subsection also contains the classification results per recording and for 5 and 6 different classes. The tests in Subsections 6.2.1 and 6.2.2 also apply the 100-fold random combination testing method.

6.2.1 Pre-Processing

For the pre-processing, the effect of each stage was carefully analyzed as to choose the most fitting setup. The first approach was mostly by visual analysis, as described in Section 4.2. As the
whole system was implemented, most settings were analyzed by verifying their impact on the final accuracy of the classification.

The first parameters to be analyzed were the baseline filter window size and the IIR filter cutoff frequency. These parameters have great impact on the quality of the filtered signal, thus this joint validation aimed for the best combined result of the both of them. Table 6.4 presents the accuracy of the classifier for several different combinations, with a maximum for the combination of 45 Hz of cutoff frequency and 500 ms for the baseline filter window size. In this case, the visual analysis presented in Section 4.2 is corroborated.

Table 6.4: Average accuracy of the classifier for different values of the baseline filter window size and the IIR filter cutoff frequency.

<table>
<thead>
<tr>
<th>IIR Filter Cutoff Frequency (Hz)</th>
<th>Window Size (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>125</td>
</tr>
<tr>
<td>25</td>
<td>71.66</td>
</tr>
<tr>
<td>35</td>
<td>83.96</td>
</tr>
<tr>
<td>45</td>
<td>77.85</td>
</tr>
<tr>
<td>55</td>
<td>77.71</td>
</tr>
<tr>
<td>65</td>
<td>72.56</td>
</tr>
<tr>
<td>75</td>
<td>74.37</td>
</tr>
</tbody>
</table>

The order of the filter is a parameter that is expected to improve the filtering quality as it goes up. Hence, for every subsequent order increment, it becomes a trade-off between accuracy improvement and computational complexity increase. From the results in Table 6.5 combined with the visual analysis, it was decided that the 8th order was the best compromise between these two considerations.

Table 6.5: Average accuracy of the classifier for different orders of the IIR filter.

<table>
<thead>
<tr>
<th>IIR Order</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>92.903</td>
<td>94.245</td>
<td>94.346</td>
<td>94.333</td>
<td>94.262</td>
<td>94.521</td>
</tr>
</tbody>
</table>

Not being prioritized in the initial design planning stages, the normalization stage became a key piece in this architecture. For the InCarTDb recordings, no normalization means having an architecture that is almost useless, as seen in the results of Table 6.6. The selection of some parameters and boundaries of this process have already been described in Section 4.2.

Table 6.6: Average accuracy of the classifier with and without the normalization stage.

<table>
<thead>
<tr>
<th>Normalization</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>94.25</td>
<td>74.97</td>
</tr>
</tbody>
</table>

6.2.2 Features Extraction

The dynamic features were implemented as-is from the work of Ye et al. [34], and their suitability is going to be discussed in Subsection 6.2.3.

The resampling stage improved the overall performance of the system. Although this improvement was of only 2% (Table 6.7), it is a process that is completely justified to make up for the dynamic
variability of the heart rate. The analysis at this point was not as extensive as for the remaining stages of pre-processing. The choice of the 1.2 and 0.8 Fs resampling factors was based on visual analysis of a set of heartbeats that were going either out of bounds or had adjacent heartbeats in its segment window. Future work should verify a more systematic choice of segmentation parameters.

Table 6.7: Average accuracy of the classifier with and without the resampling stage.

<table>
<thead>
<tr>
<th>Resampling</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>94.245</td>
<td>92.213</td>
</tr>
</tbody>
</table>

As mentioned in Section 4.4, the morphological features were excluded at an early stage of the process as they did not provide better results for the classification. For the DWT based features, the exact same features from the work of Ye et al [34] were used, as described in the previously mentioned section. Therefore, only the most suitable number of principal components needed to be assessed. From Figure 6.2, the 14 first principal components produce the best classifications on average. An analysis of the box plot distribution of the 100-fold inter-patient classification process of Figure 6.3 verifies what is expected as the number of principal components increase: the more features used, the lower the bias and higher the variance of the classification [49]. In this case, the median accuracy plateaus (hence, so does the bias) after the 15th principal component is used while the maximum goes up a bit more. However, lower values of accuracy also become more frequent, which verifies the increase of variance. This decrease is also evident in the decrease of the mean accuracy in Figure 6.2 for more principal components. Therefore, and since a smaller number of principal components renders a better computational performance and area utilization in the architecture, only the first 14 principal components were used for the classification.

Figure 6.2: Average accuracy of the classifier while varying the number of Principal Components used in classification.
Although this selection of morphological features provided good results, future work should verify a systematic approach to the DWT based features selection in order to assess if there is a more suitable combination of approximation and details coefficients, as well as different mother wavelets.

### 6.2.3 Classification

As mentioned in the beginning of the current section, the classification process has had many different formats of training and test and of the classifier itself. The inter-patient analysis was applied in two ways: with the training and test recordings to be used as a whole, being this method used to generate the results present in the accuracy tables of this section; with the training set as whole but with the testing being separate for each subject.

Since the definition of the training set is the same for both methods, a cross analysis of the number of training recordings and the number of heartbeats to be used for training per class was performed. The heartbeat features of the selected training recordings for each model are grouped by the classes they represent, and the training set of each class is randomly selected from those groups. The results of this analysis are shown in Table 6.8 and it is observable that 10 patient recordings and 150 samples for each class of the training set provide the best results. However, to improve the model generalization (which becomes a problem when more samples are chosen for each class) and to reduce the number of samples per class for area utilization reduction, 15 recordings with 100 samples per class were used.

For the first method of testing, the classes of the testing set recordings were also bundled together like the training set. All heartbeats from the testing recordings are used, except for a fraction of the healthy ones. This cropping of the testing set is meant to keep a balance between having a realistic healthy/disease ratio, while insuring that the classification of classes with lower representation also takes a toll in the overall accuracy. Therefore, it was established that the number of healthy heartbeats...
of the testing set is equal to 5 times the number of the most highly represented disease class.

Table 6.8: Accuracy average of the classifier while varying both the number of recordings and the number of samples of each class in the training set used in classification.

<table>
<thead>
<tr>
<th># of Patients in the training set</th>
<th># of Samples per class</th>
<th>25</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>92.18</td>
<td>93.22</td>
<td>94.69</td>
<td>94.78</td>
<td>95.01</td>
<td>94.79</td>
<td>94.71</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>92.22</td>
<td>93.22</td>
<td>94.25</td>
<td>94.43</td>
<td>94.73</td>
<td>94.91</td>
<td>94.81</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>92.23</td>
<td>93.27</td>
<td>93.92</td>
<td>94.25</td>
<td>94.47</td>
<td>94.39</td>
<td>94.77</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>92.99</td>
<td>93.43</td>
<td>93.87</td>
<td>94.36</td>
<td>94.44</td>
<td>94.47</td>
<td>94.78</td>
<td></td>
</tr>
</tbody>
</table>

An analysis of the classes ensued. For two classes, Nodal Ectopic Beat (NEB)s and Right Bundle Branch Block (RBBB)s, inter-patient validation of the model for this class is not possible since they are only verified in one patient, hence intra-patient and inter-recording was attempted. The remaining classes kept the inter-patient method previously described. At first, 6-class classification was attempted. However, there are only 92 NEB occurrences in patient (26). In order to keep a balanced classifier and a considerable testing set, 60 samples per class was considered. For this setup, the maximum of accuracy across 200 different models generated for 6 classes was of 81%, with the mean being at 51%. Aside from the low number of training samples for each class, as seen in Figure 6.1, the NEB is also extremely similar to the healthy beat, so the features used are not sufficient to differentiate. The confusion matrix of the 6-class classification is shown in Figure 6.4. From the matrix is clear that a big portion of the healthy and Ventricular Fusion Beat (VFN) beats are confused by the classifier with NEBs. Although the intra-patient NEB classification provided good results, the low occurrence of NEBs in the recordings and the consequent poor accuracy of the classifier led to disregarding the class.

With VFN having 148 occurrences, having the required 100 samples per class is feasible. However, in this case too the similarities between the VFN and the healthy waveforms is great. Although the healthy heartbeats are only classified as VFNs for about 3.5% of the occurrences, the VFNs are classified as healthy heartbeats about 61% of the times, which makes the classification pointless for this class as well. Figure 6.4 presents the confusion matrix of the 5-class classification as well.

The last model to be generated considers the remaining four classes. Accuracies of over 96% are possible with this model, as well as particularly good classification parameters for the VPC and RBBB classes. However, a great fraction of the APCs are being confused with the Healthy beats, and this was verified for the 4, 5 and 6 class classification. Therefore, a limitation of this model is in the identification of abnormal beats mostly based on rhythm disturbances, such as the case of APCs. For this model, the AAMI parameters for assessing the classification quality described in Section 5.1 were assessed and written down in Table 6.9. More information is shown in the confusion matrix of this 4-class classification presented in Figure 6.4.

The results for the 4-class classifier are close to the multiple-lead classifier developed by Jannah et al [18], which got an accuracy of 98.33%, and performs better than most of the classifiers to which their classifier is compared to in their work.
Table 6.9: Evaluation of the AAMI parameters for a 4-class classification model.

<table>
<thead>
<tr>
<th>Class</th>
<th>Occurrences</th>
<th>Sensitivity(%)</th>
<th>PPR(%)</th>
<th>FPR(%)</th>
<th>Accuracy(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy</td>
<td>38,130</td>
<td>97.9</td>
<td>97.6</td>
<td>2.4</td>
<td>-</td>
</tr>
<tr>
<td>APC</td>
<td>1543</td>
<td>42.3</td>
<td>50.9</td>
<td>49.1</td>
<td>-</td>
</tr>
<tr>
<td>VPC</td>
<td>7,626</td>
<td>100.0</td>
<td>99.4</td>
<td>0.6</td>
<td>-</td>
</tr>
<tr>
<td>RBBB</td>
<td>3,045</td>
<td>97.5</td>
<td>94.7</td>
<td>5.3</td>
<td>-</td>
</tr>
<tr>
<td>Overall</td>
<td>50,364</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>96.5</td>
</tr>
</tbody>
</table>

Figure 6.4: Confusion Matrix for some of the best models generated for an inter-patient classification process of 4, 5 and 6 classes.

The final analysis of the classification tested the accuracy of the method for each patient. Figure 6.5 contains a trend line of the mean accuracy of each patient. As it is observed, only two recordings go under 80%, with patient 30 being even under 10%. Thus, both patients were analyzed in depth. Patient 30 revealed a great number of healthy peaks being classified as RBBBs. This could be due to three factors: having training RBBB samples from only one patient is insufficient, which is the most likely cause; an error of annotation; a small distortion to the signal imposed by the filtering which lead
to an increase of the width of the QRS peak, one of the most striking characteristics of the RBBB.
Patient 18 revealed the same issue, but with healthy beats being classified as APCs. In this case, the
patient’s recording 41 header indicates that the patient has sinus arrhythmia, which may induce the
dynamic features to lean the classifier towards APC. Since these are errors of healthy beats being
classified as abnormal and are only 2 patients out of 32, patients 18 and 30 were always used on
the training sets, as to not lower the overall accuracy of the classifier with a known bias. Therefore,
all the previous results, aside from the ones in Figure 6.5, were generated with patients 18 and 30
purposefully on the training sets.

![Figure 6.5: Accuracy average of the classifier for each patient every time it is included in the testing set.](image)

6.3 Hardware Implementation

The details of the actual implementation of the architecture for the present work are described in
this section. The choice of Data I/O is justified in Subsection 6.3.1 as are the structural parameters
and configurations of the architecture in Subsection 6.3.2. This architecture was designed to be
low power and low in area utilization, while keeping a time performance good enough for real-time
processing. An analysis of these parameters, as well as the scalability of the architecture for 4, 5 and
6-class classifiers, and the signal classification quality are presented in Subsections 6.3.3 (resources),
6.3.4 (timing), 6.3.5 (power) and 6.3.6 (classification).

6.3.1 Data I/O

To develop an architecture, one of the first considerations made should be the format of the data
to be used. An analysis of the 75 recordings of the InCarTDb reveals that the signal oscillates, at
most, in the interval of [-8; 8] mV. Therefore, the signal requires at least 4 bits to code the integer part
in 2’s complement format. Since ECG processing systems do not require extremely high dynamic
range nor high resolution, the floating-point format was dismissed, with fixed-point format being used
instead. The remaining analysis lies in how many bits are required to code the fractional part. For a
matter of standardization of architecture, we should choose a format with a number of bits equal to a
power of two, so either 8, 16 or 32 bits. 4 bits are obviously not enough, otherwise the signal would just be steps between 16 different values, and 64 is too much since a resolution of $8.67e^{-19}$ mV is excessive, as was for the floating-point format. In order to optimize the area utilization of the FPGA the format should have the least amount of bits possible with no damage to the classification process. The absolute minimum, mean squared and absolute maximum errors of the format conversion of all recordings from double to 8, 16 and 32 bits with 4 integer and signal bits are shown in Table 6.10.

Table 6.10: Average absolute maximum and mean squared errors of format conversions from double to 8, 16 and 32 bits for all the 75 InCarTDb recordings.

<table>
<thead>
<tr>
<th>Data format</th>
<th>8 bits (Q4.4)</th>
<th>16 bits (Q4.12)</th>
<th>32 bits (Q4.28)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Maximum Error (mV)</td>
<td>3.099e-02</td>
<td>1.2104e-04</td>
<td>1.874e-09</td>
</tr>
<tr>
<td>Average Mean Squared Error (mV)</td>
<td>7.435e-02</td>
<td>4.9659e-09</td>
<td>1.155e-18</td>
</tr>
</tbody>
</table>

A segment of a recording before and after the format conversion to 8, 16 and 32 bit fixed-point formats are shown in Figure 6.6. Taking the data from Table 6.10 and verifying that visually it is not possible to distinguish between the signal in double and both 16 and 32 fixed-point formats, it is possible to conclude that the Q4.12 format is enough to be used in the processing stages.

Throughout the development of the architecture in this work, the data was inputted to the processing stages by writing the all the signal data, weights and coefficients in either Q4.12 or Q1.15 (whenever the coefficients were between -1 and 1). It was then stored in hexadecimal format in .coe files and used to program distributed memory and BRAMs before synthesis. At the end of the processing stages, the filtered signal, the peaks locations and respective classifications are outputted to three separate .txt files to be analyzed and processed in Matlab. Due to high simulation times, the input signal files contained 130,000 samples of data at each time, which correspond to files of around 8 minutes and 45 seconds.
6.3.2 Architectural Specifications

Some general parameters considered in Chapter 5 are here specified in order to detail the characteristics of the implemented architecture. As the sampling frequency of the InCarTDb is 257 Hz, most structures considered powers of two in their sizes taking 256 samples as 1 second. This approximation does not affect the processing of the signal and reduces greatly the hardware complexity.

Baseline Filtering

For this stage, the only consideration to be made is related to the \texttt{WindowSize}. Being dependent on the sampling frequency, it becomes that this array comprised 128 samples, thus being the \texttt{WindowSizeBit} arithmetic shift right input equal to 7 bits.

IIR Filtering

Since the IIR filter implemented is of the 8th order, this means that it is composed of four second-order sections. To avoid area utilization problems when the order of the filter scales up, the same Datapath sub-unit is used for all stages sequentially. This is done by running all the samples for each stage and storing them in the exact same register of the array, which justifies the multiplexed writing of the \texttt{FilteredSamplesWindow} array of Figure 5.1. In terms of performance, it would be equally good to put each sample through the four stages of filtering before moving to the next one, but this results in unnecessary switching of the coefficients which would lead to an increase of power consumption.

Peak Detection and Signal Normalization

For the sampling frequency of 257 Hz, the standard processing window requires that the peak is to be placed in the 64th sample of a 160 samples window. The resampling requirements are that of 26 ms before and 62.5 ms after the intended heartbeat segment array, thus an extension of 10 samples before and 16 samples after the intended 160 samples array. Hence the peak is place at 276 ms in the window, and the runs are, at most 437.5 ms long. With these extensions, the peak needs to be stored in the 74th position of a 186 positions array. With this knowledge, the filtering runs will be, at most, 111 samples long as to place the peak in the correct position of the array.

In terms of the normalization process, the fixed threshold for the peak detection was set to 0.5 mV.

Template Matching

For template matching, the only template used was that of a VPC, since it is the only of the considered classes that cannot be detected by the previously defined amplitude thresholding.

Dynamic Features

In order to avoid using arithmetic divisions, it was established that the 10 seconds average would use the 16 most recent RR intervals, and the 3 minutes average the most recent 256. Therefore an array of 256 positions was used to store the RR intervals. The average values resulted from the arithmetic shift right by 4 bits for the 10 seconds, and by 8 bits for the 3 minutes.

Resampling

In this work, the trial-by-error sampling rendered the down-sampling to be at 0.8Fs and the up-sampling at 1.2Fs, as seen in Subsection 6.2.2. However, the resampling block allows for the resampling to be done for other frequencies, thus the parameters being defined as generics prior to synthesis. For the up-sampling, 14 samples (\texttt{ResampUpBefore}) are removed from the beginning
and 20 samples ($\text{ResampUpAfter}$) from the end, while the insertion of the average value is performed at every 4th ($\text{ResampUpCount}$) sample. The down-sampling resorts to the extension of 10 samples ($\text{ResampDownBefore}$) before and 16 samples ($\text{ResampDownAfter}$) the array performed in the Peak Detection and Signal Normalization process, while every 6th sample ($\text{ResampDownCount}$) is removed.

**Morphological Features**

The DWT implemented resorted to the DB8 mother wavelet, and that renders two filters, a high-pass and a low-pass, of the 16th order. This results in the mirroring parameter $\text{Mirror}_n$ being equal to 14. Therefore, at every level of the DWT analysis process, the mirroring is performed for the first and last 14 samples. The features to be extracted are the 3rd and 4th order details coefficients and 4th order approximation coefficients. For the sampling frequency of 257 Hz this results in 33 coefficients from the 3rd order, and 24 from both details and approximation coefficients of the 4th order, which gives a total of 81 morphological features.

**Principal Component Analysis**

For the PCA an 81x81 matrix of coefficients is required, being it stored in a BRAM. The number of principal components to be calculated are defined by a generic parameter prior to the synthesis.

**Classification**

A PNN with different characteristics set by generic parameters was defined. The BRAM with the exponential function values table contained 41,000 values ranging from the exponential of 0 to the exponential of -10. The exponential of -10 is close to the limit of resolution for the Q0.16 format, so this was the minimum value considered for the exponential function. Reducing the resolution of the exponential function outputs was attempted and the results were worse, thus this was the final value. The addressing of this memory is done with Q4.12 values from 0 to 10, being the output equal to the exponential of the symmetric address value.

Classifiers with 4, 5, 6, 8 and 16 classes were designed as to study the evolution of resource utilization. However, the model used the for all the remaining tests was the 4-class classifier that rendered the results of Subsection 6.2.3.

### 6.3.3 Physical Resources Usage

After defining all the parameters of the architecture as in Subsection 6.3.2, the architecture was synthesized and implemented in Vivado, and the hardware mapped to a Xilinx Zynq-7 ZC702 Evaluation Board. This board was chosen due to the limitations of resources of its FPGA when compared to other higher grade models, as well as to its availability on the development site of this work.

The different tests for this subsection only took into consideration the assessment of the scalability of the number of classes. The post-implementation resources utilized in the architecture for 4, 5, 6, 8 and 16 classes were tested. Until the 6th class, the six different classes analyzed before were used, but for the remaining classes, the same heartbeat abnormalities were fed into the classifier, as the hardware is independent of the class itself. The resource evolution with the class is represented in Table 6.11 and Figure 6.7.
Table 6.11: FPGA resources utilization table for the architecture with classifiers of 4, 5, 6, 8 and 16 classes.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Available</th>
<th>Utilization</th>
<th>Utilization(%)</th>
<th>Available</th>
<th>Utilization</th>
<th>Utilization(%)</th>
<th>Available</th>
<th>Utilization</th>
<th>Utilization(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT</td>
<td>53200</td>
<td>20648</td>
<td>38.81</td>
<td>21570</td>
<td>40.55</td>
<td></td>
<td>22571</td>
<td>42.43</td>
<td></td>
</tr>
<tr>
<td>LUTRAM</td>
<td>17400</td>
<td>2176</td>
<td>12.51</td>
<td>2576</td>
<td>14.80</td>
<td></td>
<td>2976</td>
<td>17.10</td>
<td></td>
</tr>
<tr>
<td>FF</td>
<td>106400</td>
<td>16542</td>
<td>15.55</td>
<td>16645</td>
<td>15.64</td>
<td></td>
<td>16730</td>
<td>15.72</td>
<td></td>
</tr>
<tr>
<td>BRAM</td>
<td>140</td>
<td>27.5</td>
<td>19.64</td>
<td>27.5</td>
<td>19.64</td>
<td></td>
<td>27.5</td>
<td>19.64</td>
<td></td>
</tr>
<tr>
<td>DSP</td>
<td>220</td>
<td>10</td>
<td>4.55</td>
<td>11</td>
<td>5.00</td>
<td></td>
<td>12</td>
<td>5.45</td>
<td></td>
</tr>
</tbody>
</table>

Default 8-classes

<table>
<thead>
<tr>
<th>Resource</th>
<th>Available</th>
<th>Utilization</th>
<th>Utilization(%)</th>
<th>Available</th>
<th>Utilization</th>
<th>Utilization(%)</th>
<th>Available</th>
<th>Utilization</th>
<th>Utilization(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT</td>
<td>53200</td>
<td>24252</td>
<td>45.59</td>
<td>28371</td>
<td>53.33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUTRAM</td>
<td>17400</td>
<td>3968</td>
<td>22.80</td>
<td>7552</td>
<td>43.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF</td>
<td>106400</td>
<td>16905</td>
<td>15.89</td>
<td>17678</td>
<td>16.61</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRAM</td>
<td>140</td>
<td>27.5</td>
<td>19.64</td>
<td>27.5</td>
<td>19.64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSP</td>
<td>220</td>
<td>14</td>
<td>6.36</td>
<td>22</td>
<td>10.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As expected from the structure of the architecture, most of the utilized resources are LUTs, which implement most functions and logic operations, and FFs, which store the internal data signals. Since no authors described an ECG disease detection dedicated architecture with this order of complexity and published the resource utilization, the comparison will be made only for the classifier. From Table 6.11, it can be derived that each class datapath sub-unit require roughly 645 LUTs, 448 LUTRAMs (LUTs used for data storage), 95 FFs and 1 DSP. These values are extremely inferior to those presented by [5] and described in Subsubsection 3.2.1.D. In that work, just the floating-point classifier requires as much resources as this works full architecture, an about 10 times more DSPs. Hence an implementation of that sort would be unfitting for the current system.
### 6.3.4 Timing Constraints

From Equation 5.9, the lower bound of the clock frequency was considered for the sampling frequency of 257 Hz and a maximum heart rate of 180 bpm as

\[
F_C > \frac{1}{t_{RR}} S_{peakProcess} + F_s (S_{MA} + S_{IIR} + S_{tempMatch} + S_{peakDet}) = \frac{3 \times 8618 + 257(1 + 12 + 160 + 4)}{8618} = 71.343 \text{ kHz},
\]

which corresponds to 14 µs for each clock cycle.

The upper bound of the clock frequency was defined by Vivado’s post-implementation timing reports, which indicated that the minimum clock cycle is between 27.5 and 28.5 ns. This indication results from several implementations with the clock cycle at 30 ns which rendered Worst Negative Slack values between 1.5 a 2.5 ns. This value corresponds to the remaining time for logic or path operations to be performed in the busiest clock cycle of the architecture.

Therefore, the clock frequency can be chosen to suit performance and power needs accordingly, as long as the condition

\[
F_C \in [0.071343; 35.714] \text{ MHz}
\]

is verified.

This maximum clock frequency is close to the values that other authors obtained [45, 47], between 22.4 and 50 mHz, and is 500 times higher than the minimum clock frequency. Although it was not tested, this enables the consideration of the processing of all 12 leads of an ECG to be performed sequentially in this architecture, with a fusion method such as the one presented by Ye et al [34] and additional support logic to coordinate the system.

### 6.3.5 Power Analysis

As for the power consumption of the architecture, Vivado’s post-implementation power reports were used as to describe the evolution of power both for different classes and different clock frequencies. For the class analysis, Table 6.12 contains the assessment of the power consumption for the 4, 5, 6, 8 and 16 classes. The static power remains the same, and a fairly linear increase of 2% per class of the dynamic power is verified.

**Table 6.12:** FPGA static and dynamic power consumption table for the architecture with classifiers of 4, 5, 6, 8 and 16 classes.

<table>
<thead>
<tr>
<th></th>
<th>Default 4-classes</th>
<th>Default 5-classes</th>
<th>Default 6-classes</th>
<th>Default 8-classes</th>
<th>Default 16-classes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total On-Chip Power (mW)</strong></td>
<td>192</td>
<td>194</td>
<td>198</td>
<td>202</td>
<td>208</td>
</tr>
<tr>
<td><strong>Static Power (mW)</strong></td>
<td>122</td>
<td>122</td>
<td>122</td>
<td>122</td>
<td>122</td>
</tr>
<tr>
<td><strong>Dynamic Power (mW)</strong></td>
<td>70</td>
<td>72</td>
<td>76</td>
<td>78</td>
<td>86</td>
</tr>
</tbody>
</table>

For the clock frequency analysis, different values were tested: 71.343 kHz, and 1, 10, 20 and 33.3 MHz (Table 6.13). As for the class tests, the static power remained the same as for all values of clock.
However, the dynamic frequency went from less than 1 mW, at the minimum clock frequency, to 70 mW close to maximum frequency.

Table 6.13: FPGA static and dynamic power consumption table for the architecture with different clock frequencies.

<table>
<thead>
<tr>
<th>Clock Frequency (MHz)</th>
<th>Default 4-classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.07143</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>33.3</td>
</tr>
</tbody>
</table>

| Static Power (mW)     | 122               |
| Dynamic Power (mW)    | < 1               |
| Total On-Chip Power (mW) | 122               |

As a comparison, the system from Abburi et al [47] consumes from 2 to 3 times less than this implementation, being, however, separated by less than an order of magnitude.

6.3.6 Testing and Classification Considerations

The quality assessment of the architecture’s signal processing was divided in two parts, segmentation and classification, both characterized as defined in the AAMI standards mentioned in Section 6.1. The classifier was generated with a training set and the analyzed signals of this section all belonged to the patients of the test set. This way the inter-patient paradigm is verified. As mentioned in Subsection 6.3.1, blocks of 130,000 samples from each recording were utilized for the testing. These blocks were carefully selected as to contain occurrences from all classes (although not simultaneously), as well as regions with worse signal quality as to simulate practical noisy scenarios. The results of the behavioral simulation of the architecture for both segmentation and classification considering these blocks as inputs are presented in Table 6.14 and 6.15, respectively.

Table 6.14: Segmentation results of the final architecture for the selected recordings.

<table>
<thead>
<tr>
<th>Recording</th>
<th>Patient</th>
<th>TP</th>
<th>VPC Peak</th>
<th>Other</th>
<th>Total</th>
<th>FN</th>
<th>Small Peak</th>
<th>VPC</th>
<th>Other</th>
<th>Total</th>
<th>Sensitivity (%)</th>
<th>PPR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>368</td>
<td>12</td>
<td>105</td>
<td>117</td>
<td>50</td>
<td>4</td>
<td>18</td>
<td>72</td>
<td></td>
<td>83.64</td>
<td>75.88</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>451</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>14</td>
<td></td>
<td>96.99</td>
<td>98.69</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>396</td>
<td>4</td>
<td>78</td>
<td>82</td>
<td>24</td>
<td>1</td>
<td>3</td>
<td>28</td>
<td></td>
<td>93.40</td>
<td>82.85</td>
</tr>
<tr>
<td>21</td>
<td>10</td>
<td>587</td>
<td>5</td>
<td>16</td>
<td>21</td>
<td>19</td>
<td>0</td>
<td>1</td>
<td>20</td>
<td></td>
<td>96.71</td>
<td>96.55</td>
</tr>
<tr>
<td>75</td>
<td>32</td>
<td>473</td>
<td>143</td>
<td>3</td>
<td>146</td>
<td>82</td>
<td>6</td>
<td>0</td>
<td>88</td>
<td></td>
<td>84.31</td>
<td>76.41</td>
</tr>
</tbody>
</table>

Starting by the last test, for recording 75, from patient 32, the greatest concern lies with the segmentation. An analysis of the segment verified that the majority of the false detections of peaks were caused by most of the VPCs having a P wave peak above the amplitude threshold of peak detection. On the other hand, real heartbeats whose R peak did not reach above the amplitude threshold composed the majority of the undetected heartbeats, with some undetected VPCs as well due to not being above the cross correlation threshold. The false detection problem reveals that a threshold of 0.5 mV may not be ideal for every scenario, while the missed detection problem indicates that the normalization process does not work flawlessly.
Table 6.15: Classification results of the final architecture for the selected recordings.

<table>
<thead>
<tr>
<th>Recording</th>
<th>Patient</th>
<th>TN</th>
<th>FN</th>
<th>Sen (%)</th>
<th>PPR (%)</th>
<th>FPR (%)</th>
<th>TP</th>
<th>FP</th>
<th>Sen (%)</th>
<th>PPR (%)</th>
<th>FPR (%)</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>322</td>
<td>7</td>
<td>89.9</td>
<td>97.9</td>
<td>2.1</td>
<td>2</td>
<td>8</td>
<td>25</td>
<td>20</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>437</td>
<td>0</td>
<td>99.5</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>10</td>
<td>495</td>
<td>31</td>
<td>89.4</td>
<td>94.1</td>
<td>5.9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>32</td>
<td>320</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

This is due to the fact that the normalization uses a window that often encompasses more than one peak. In the raw signal, the peak amplitude can vary considerably for just one peak, then returning to the previous amplitude. This is especially true for noisier acquisition setups.

Recording 4, from patient 2, is particularly noisy. This lead to a series of false detections of all classes, having 32 healthy beats, 30 APCs, 50 VPCs, and 5 RBBBs, aside from the VPC peaks as seen in recording 75. Adding a several healthy beats below the amplitude threshold, among others, and the segmentation quality was not great. Also probably due to the noise, several healthy segments were classified as abnormal, most of the times as RBBBs. Corroborating the results from the software implementation, the classifier often fails to detect APCs correctly.

Recording 9, from patient 5, verified a great quality of both segmentation and classification. The false detections were caused by peaks in slightly noisier areas that did not qualify as heartbeats. The missed detections verified the exact same problem as recording 75, with the normalization failing for some peaks. In the classification, most peaks were healthy and their vast majority were correctly classified. This block of signal illustrates the quality of the setup for a less noisy raw signal with little heartbeat abnormalities.

Recording 16, from patient 8, is one of the recordings that does not verify the inter-patient paradigm, as described in Subsection 6.2.3. However, the intra-patient paradigm can still be assessed for this class. Out of the missed detections, 85% were RBBBs whose peak was below the amplitude threshold. The failed the detections were verified 70% of RBBBs being incorrectly identified. However, for the true segmentations, the classification reached an accuracy of 99.2%, hence this classifier is suitable for application in this paradigm.

Recording 21, from patient 10, contains several noisy segments, as in recording 4. In this case, the segmentation performed particularly well in this case, being most of the missed heartbeats the ones below the amplitude threshold. However, the classification verified several healthy beats being classified as both VPCs and RBBBs, which verifies insufficient denoising.

The overall accuracy values for the correctly segmented heartbeats of each patient for the architecture are inline with the ones obtained from the software implementation (Figure 6.5).
Conclusion

This work aimed for the development of a customizable, scalable, low-power and low area architecture for real-time ECG processing. Particular emphasis was made on Lead 1 systems, because Lead 1-only off-the-person electrocardiography is taking its first steps [10, 11], and a specific hardware architecture for disease detection in these conditions has not been described in literature yet.

Due to the lack of gold standard Lead 1 recordings, a less common database (InCarTDb) was used for this development. In terms of software, the main challenge was to assess the validity of the state-of-the-art ECG processing and heart disease detection methodologies for this database and understand its limitations. The system developed verified a signal denoising and artifact removal stage, which considered baseline removal by subtracting a moving average window and low-pass filtering with an IIR filter. The filtered signal then has its amplitude normalized to 1 and, from the annotated peaks location, windows around them are taken to generate the heartbeat segments. Downsampling if the heart rate is below 60 bpm and upsampling if it is above 133 bpm, the dynamic features of the heartbeat are extracted and concatenated with PCA-reduced DWT-based morphological features. A training and a test set are generated from the features of all the heartbeats, having the training cluster being fed into a PNN classifier, and the testing cluster verifying its performance.

This was attempted and a 4-class classifier with accuracies up to 96.5% in a mostly inter-patient paradigm was obtained.

Having the software implementation as a starting point, the hardware architecture was developed and simulated. Independent blocks for each stage of the processing were interconnected in a multiple-cycle architecture. All these blocks have their own Control Unit state-machine, and most of them have also a Datapath sub-unit to aid in the arithmetic operations. The architecture is customizable through generic parameters defined prior to synthesis, which can be used to change from the input format of the signal data to the specific characteristics of each processing block. Being mapped to a low-end
FPGA evaluation board, the minimum power consumption of the architecture is of 122 mW, which is sufficiently good for integration in portable devices. The clock frequency can be any value between 71.343 kHz and 35.714 MHz, which is within the standard working frequencies of different authors [46]. In terms of resource utilization, this system did not require more than 20% of any available resources of the evaluation board except for the LUTs, where 38% of them were required. For the signal classification quality, accuracies from 88.3% to 100% were verified for different patients.

7.1 Future Work

The ground work this work set makes room for an immense amount of improvements. The noise in ECG systems is a hinderance, particularly for off-the-person acquisitions, where sometimes the signals are rendered useless. Therefore, for more filters and noise removal methodologies, such as entropy based thresholding, could be applied and tested, as well as some of the QRS detection methods presented in Subsection 3.1.2.

Both in feature extraction and classification, although the selected methods provided good results, other features and other classifiers could be assessed. In particular for the dynamic features, different ways to integrate them in the classification process should be considered as to improve the poor performance of the classifier for the APCs. Aside from this, more complex methods such as fuzzy logic or genetic algorithms, or even deep learning methods could be assessed. This would be particularly interesting if the architecture was to be implemented on an ASIC, which would allow for more computational power for low power consumption and area utilization.

Particularly for this architecture, the next step would be to effectively program a device with it, develop the interface for an acquisition setup and storage, and test it in a physical context. Besides that, for the highest values of the architecture’s clock frequency, it can be considered the extensibility of the architecture to process more than one lead of ECG sequentially. This possibility would expand its range of applications, as well as the robustness of the classification.

However, for the particular case of the Lead 1-only application, the first and most important step is to work towards the development of a gold-standard Lead 1 annotated database with more occurrences of all of the AAMI standard classes. This way, new methodologies and architectures can be developed and compared for this application.
Bibliography


## A.1 Heartbeats occurrences per patient

**Table A.1:** Discriminated heartbeats occurrences for each patient after the normalization and pruning.

<table>
<thead>
<tr>
<th>Patient Number</th>
<th>Heartbeat Type</th>
<th>Healthy</th>
<th>APC</th>
<th>VPC</th>
<th>RBBB</th>
<th>VFN</th>
<th>NEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4853</td>
<td>0</td>
<td>74</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>6068</td>
<td>17</td>
<td>396</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5068</td>
<td>111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>766</td>
<td>0</td>
<td>81</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>8573</td>
<td>0</td>
<td>78</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>4120</td>
<td>0</td>
<td>274</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1269</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>3145</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1312</td>
<td>0</td>
<td>19</td>
<td>0</td>
<td>41</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>6605</td>
<td>349</td>
<td>233</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>3150</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>3171</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1588</td>
<td>0</td>
<td>707</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>3927</td>
<td>0</td>
<td>2096</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>2648</td>
<td>1105</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>7596</td>
<td>0</td>
<td>1244</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>3099</td>
<td>0</td>
<td>800</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>4171</td>
<td>11</td>
<td>92</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>2580</td>
<td>0</td>
<td>843</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>5304</td>
<td>9</td>
<td>1536</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>3972</td>
<td>3</td>
<td>305</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>5106</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>4632</td>
<td>3</td>
<td>415</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>5110</td>
<td>1</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>3225</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>5874</td>
<td>1</td>
<td>70</td>
<td>0</td>
<td>0</td>
<td>92</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>2546</td>
<td>4</td>
<td>488</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>6500</td>
<td>1</td>
<td>746</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>4467</td>
<td>2</td>
<td>319</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>3163</td>
<td>155</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>3763</td>
<td>40</td>
<td>429</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>3413</td>
<td>0</td>
<td>883</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>