Universitat Oberta de Catalunya (UOC)

Máster Universitario de Ingeniería de Telecomunicaciones

TRABAJO FINAL DE MÁSTER

Área: Sistemas de comunicación

Detección de arritmias mediante procesado de señales ECG a través de FPGAs

---

Autor: Gonzalo Morente Terrado

Tutor: David Naranjo Hernández

Profesor: Carlos Monzo Sánchez

Madrid, 20 de abril de 2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Detección de arritmias mediante procesado de señales ECG a través de FPGAs*
Nombre del autor:	*Gonzalo Morente Terrado*
Nombre del colaborador/a docente:	*David Naranjo Hernández*
Nombre del PRA:	*Carlos Monzo Sánchez*
Fecha de entrega:	01/2022
Titulación o programa:	*Máster de Ingeniería de Telecomunicaciones*
Área del Trabajo Final:	*Sistemas de Comunicación*
Idioma del trabajo:	*Español*
Palabras clave	*FPGA, VHDL, ECG, Complejo QRS y Arritmia*

Resumen del Trabajo:

La monitorización del pulso cardiaco de los pacientes requiere sistemas y algoritmos capaces de detectar trastornos cardiacos como las arritmias. Este proyecto tiene como objetivo la creación de un algoritmo que permita detectar complejos QRS y arritmias mediante el procesado de señales de electrocardiograma (ECG) a través de FPGAs. El método propuesto está basado en el algoritmo de Pan y Tompkins pero empleando una nueva técnica de detección de complejos QRS. Primeramente, se realiza un filtrado de la señal original a través de Matlab debido a la complejidad de la implementación del filtrado en VHDL. Tras el filtrado de la señal, ésta será procesada por la FPGA mediante el algoritmo implementado. Por último, tras la detección de complejos QRS, comienza la fase de toma de decisiones en donde la FPGA decide si es ECG con ritmo normal o arritmia. Para validar el rendimiento y fiabilidad del algoritmo se van a procesar diferentes señales ECG obtenidas a través de la base de datos MIT-BIH de arritmias, que contiene 48 ECGs de media hora cada uno..

Abstract:

Cardiac pulse monitoring of patients requires systems and algorithms capable of detecting cardiac disorders such as arrhythmias. This project aims to create an algorithm to detect QRS complexes and arrhythmias by processing electrocardiogram signals (ECG) through FPGAs. The proposed method is based on Pan and Tompkins algorithm but employs a new technique for detecting QRS complexes. First, original signal filtering is processed through Matlab due to the complexity of VHDL filtering implementation. After filtering the signal, it will be processed by the FPGA using the implemented algorithm. Finally, after the detection of QRS complexes, the decision-making phase begins where the FPGA decides whether it is ECG with normal rhythm or arrhythmia. To validate the performance and reliability of the algorithm, different ECG signals obtained through the MIT-BIH arrhythmia database, which contains 48 half-hour ECGs, will be processed.

Cita

"Donde hay voluntad, hay fuerza".

Agradecimientos

Agradecer a mi padre, mi madre, mi hermana y mi abuela por la ayuda y los ánimos recibidos para realizar este proyecto de fin de máster. Agradecer a todas las personas que me han ayudado a terminarlo.

Pero también quiero agradecerme a mí mismo por creer en mí, quiero agradecerme por hacer todo este trabajo duro, quiero agradecerme por no tener días libres. Quiero agradecerme por haber siempre estado para los demás tratando de dar más de lo que recibo. Quiero agradecerme por tratar de hacer más bien que mal. Quiero agradecerme por ser sólo yo en todo momento.

Resumen

La monitorización del pulso cardiaco de los pacientes requiere sistemas y algoritmos capaces de detectar trastornos cardiacos como las arritmias.

Este proyecto tiene como objetivo la creación de un algoritmo que permita detectar complejos QRS y arritmias mediante el procesado de señales de electrocardiograma (ECG) a través de *FPGAs*. El método propuesto está basado en el algoritmo de *Pan y Tompkins* pero empleando una nueva técnica de detección de complejos QRS.

Primeramente, se realiza un filtrado de la señal original a través de *Matlab* debido a la complejidad de la implementación del filtrado en *VHDL*. Tras el filtrado de la señal, ésta será procesada por la *FPGA* mediante el algoritmo implementado. Por último, tras la detección de complejos QRS, comienza la fase de toma de decisiones en donde la *FPGA* decide si es ECG con ritmo normal o arritmia.

Para validar el rendimiento y fiabilidad del algoritmo se van a procesar diferentes señales ECG obtenidas a través de la base de datos *MIT-BIH* de arritmias, que contiene 48 ECGs de media hora cada uno.

Palabras clave: *FPGA, VHDL, ECG, Complejo QRS y Arritmia.*

Abstract

Cardiac pulse monitoring of patients requires systems and algorithms capable of detecting cardiac disorders such as arrhythmias.

This project aims to create an algorithm to detect QRS complexes and arrhythmias by processing electrocardiogram signals (ECG) through *FPGAs*. The proposed method is based on Pan and Tompkins algorithm but employs a new technique for detecting QRS complexes.

First, original signal filtering is processed through *Matlab* due to the complexity of *VHDL* filtering implementation. After filtering the signal, it will be processed by the *FPGA* using the implemented algorithm. Finally, after the detection of QRS complexes, the decision-making phase begins where the *FPGA* decides whether it is ECG with normal rhythm or arrhythmia.

To validate the performance and reliability of the algorithm, different ECG signals obtained through the *MIT-BIH arrhythmia database*, which contains 48 half-hour ECGs, will be processed.

Keywords: *FPGA, VHDL, ECG, QRS Complex and Arrhythmia.*

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

Este proyecto consiste en el desarrollo e implementación de un algoritmo capaz de detectar complejos QRS y arritmias. Para ello se usará la tecnología de *FPGAs* debido a su gran versatilidad y capacidad de procesado. El desarrollo se realizará con la herramienta de *Modelsim.*

1.1. Contexto y justificación del Trabajo

La arritmia es un trastorno de la frecuencia cardiaca o del ritmo cardiaco, que se produce porque el pulso eléctrico generado por el corazón presenta algún tipo de irregularidad. Para detectarlas se analizan las señales ECG mediante el procesado de señales. Gracias a ello, se detecta si el corazón late más rápido de lo normal (taquicardia), más lento (bradicardia) o de forma irregular. La arritmias pueden no causar ningún tipo de daño, pero también puede indicar algún tipo de problema cardiovascular y en algunas situaciones puede ser peligroso para la salud.

Las enfermedades cardiovasculares son la primera causa de muerte en España alcanzando un 28.30 % del total de muertes en 2018, 120.859, por tanto la detección de arritmias y monitorización de señales ECG se ha convertido en un objetivo prioritario [10]. Las arritmias pueden provocar insuficiencia cardiaca, infarto cerebral o incluso la muerte, como la muerte súbita.

Las arritmias pueden detectarse mediante múltiples técnicas como:

1. **Electrocardiograma (ECG)**: es el más común porque es el más fiable. Registra la actividad eléctrica del corazón. Puede suceder que la arritmia sea intermitente y que no se detecte en el momento de la prueba del electrocardiograma.

2. **Monitor Holter**: consiste en una lectura durante 24h del ritmo cardiaco del paciente. Todo el ritmo cardiaco queda registrado. En caso de que sea necesario una lectura de más de 24h existen dispositivos que graban hasta tres semanas. Gracias a esta técnica, se pueden detectar arritmias intermitentes.

3. **Grabador de episodios**: es un dispositivo portátil que se activa cuando se presiona un botón si se detectan los síntomas. Se usa para detectar arritmias esporádicas.

4. **Ecocardiograma**: es un dispositivo que realiza una ecografía para visualizar el tamaño, estructura y movimiento del corazón.

5. **Grabador de bucle implantable**: hay eventos que ocurren muy esporádicamente, por tanto, se puede implementar bajo la piel estos dispositivos que pueden durar hasta varios años.

Este trabajo se centrará en el análisis de señales ECG: señales que miden el tiempo y la duración de cada fase eléctrica de los latidos del corazón, a través de unos electrodos adheridos al pecho y en algunas ocasiones a los brazos o piernas [7].

El algoritmo y la detección de este trastorno se implementará en una *FPGA* (*Field Programmable Gate Array*). Este dispositivo es un circuito integrado reconfigurable formado por puertas lógicas, que se pueden interconectar para formar diferentes circuitos electrónicos, por ejemplo, un procesador de señales. A diferencia de un procesador o microprocesador, los recursos de una *FPGA* se pueden configurar y crear hardware específico para lo que se está desarrollando, es decir, se puede adaptar la *FPGA* al hardware específico que se desea. Las *FPGAs* se han convertido en uno de los dispositivos más usados de las recientes tecnologías. Debido a su gran capacidad de procesado, cálculos complejos en tiempo real, bajo coste y reconfiguración es la herramienta perfecta para la biomedicina. Hoy en día las dos empresas más grandes de *FPGAs* son *Xilinx* y *Altera*. Las *FPGAs* son configuradas mediante un lenguaje de descripción de hardware (*HDL*), como *VHDL* (*Very High Speed Integrated Circuit Hardware Description Language*) o *Verilog*, en este proyecto se usará *VHDL* para el desarrollo.

El algoritmo desarrollado de este proyecto detecta complejos QRS y tras ello detecta si se está ante un caso de bradicardia, taquicardia o ECG normal.

1.2. Objetivos del Trabajo

El objetivo principal de este proyecto es desarrollar un algoritmo en *VHDL* adecuado para su implementación en *FPGAs* que procese señales ECG con el objetivo de detectar arritmias. Para la consecución del objetivo principal, se buscarán alcanzar los siguientes puntos:

1. Investigar y analizar algoritmos ya existentes que detecten complejos QRS y arritmias mediante el procesado de señales ECG.

2. Desarrollar el algoritmo basándose en los estudiados e investigados anteriormente.

3. Implementar el algoritmo en *VHDL* siguiendo un esquema modular.

4. Verificar el diseño *VHDL* mediante simulaciones a través de *Modelsim*.

1.3. Enfoque y método seguido

Para alcanzar el objetivo del proyecto la estrategia que se llevará acabo será la siguiente:

1. **Búsqueda de información e investigación**: se realizará un estudio e investigación, recopilando información de diferentes artículos científicos y técnicos sobre la detección de complejos QRS y arritmias en *FPGAs* con *VHDL*. Tras ello, se analizarán los diferentes algoritmos empleados y se creará uno nuevo, adaptando los ya existentes, mejorándolos o uniéndolos en uno más potente.

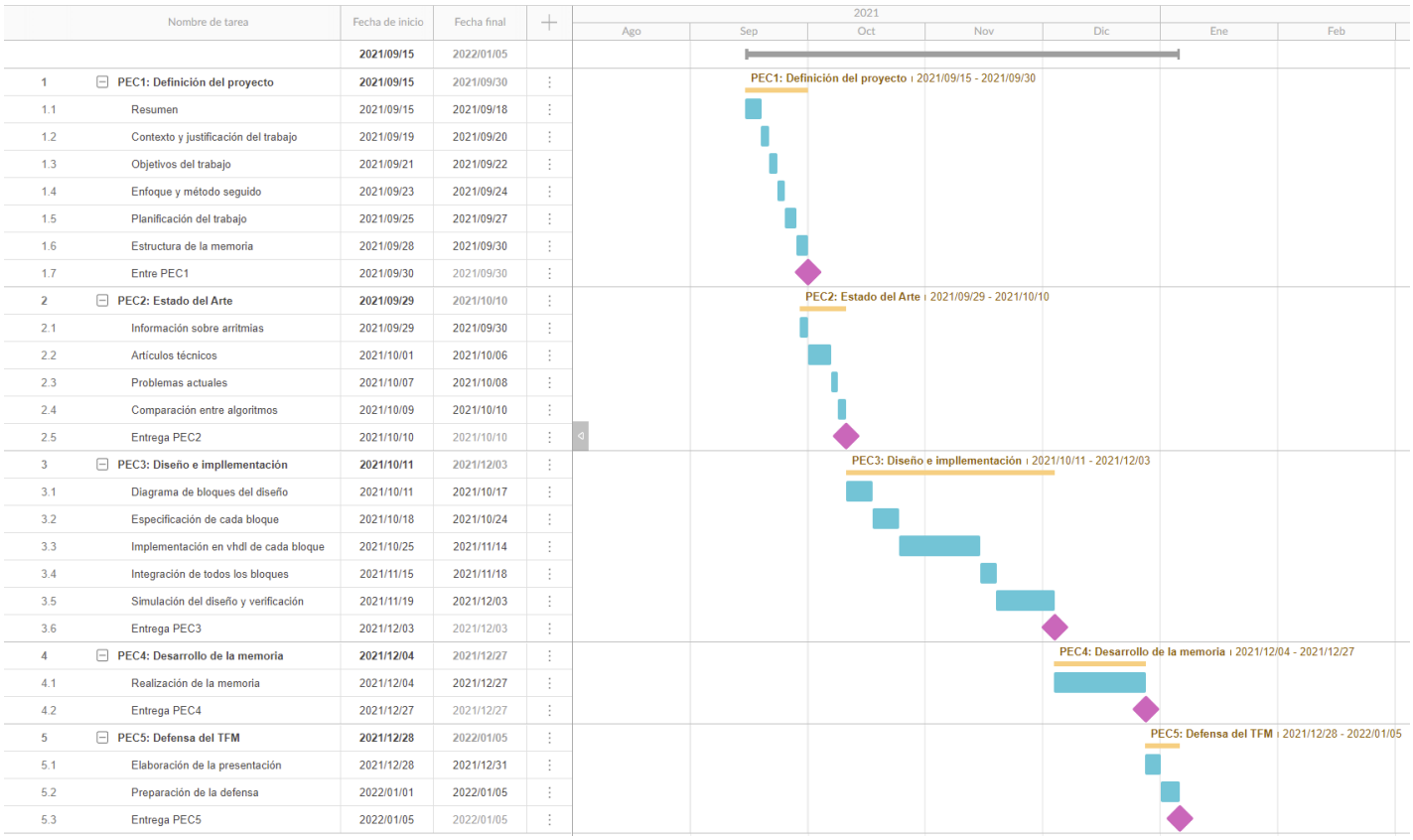
2. **Estructura del diseño**: una vez elegido el algoritmo que se va a emplear para el tratado de señales ECG, se procederá con la estructura del diagrama de bloques que tendrá el diseño *VHDL*: cuántos bloques contendrá, qué función realizará cada bloque, el porqué, cómo funcionará el sistema y cuáles serán sus entradas y salidas. Esta parte será realizada a alto nivel.

3. **Desarrollo en *VHDL***: tras conocer la estructura del diseño, el siguiente paso es comenzar con el desarrollo en *VHDL*. La metodología que se va a seguir en este apartado es la siguiente: se desarrollará cada bloque de forma independiente, se integrarán todos los bloques en el diseño estructural y se procederá con la verificación a nivel de sistema. Las pruebas de sistema se realizarán a través de *testbenches* con la herramienta de *Modelsim*.

4. **Verificación del diseño en *VHDL***: para la verificación del diseño y el algoritmo propuesto se realizarán simulaciones a través de *testbenches*. El objetivo de estas simulaciones es comprobar que el diseño funciona correctamente. Para ello, se le inyectará al diseño los diferentes datos de las señales ECG obtenidos a través de la base de datos de *MIT-BIH* para verificar la eficacia del algoritmo propuesto. Éste debe ser capaz de detectar que señales ECG presentan arritmias (taquicardia o bradicardia) y cuáles no. Las simulaciones se ejecutarán a través del simulador de *Modelsim*.

1.4. Planificación del Trabajo

El trabajo está estructurado en cinco hitos, uno por cada entrega. A continuación se muestra un diagrama de *Gantt* donde se estima la duración de cada fase.



	Nombre de tarea	Fecha de inicio	Fecha final
		2021/09/15	2022/01/05
1	**PEC1: Definición del proyecto**	**2021/09/15**	2021/09/30
1.1	Resumen	2021/09/15	2021/09/18
1.2	Contexto y justificación del trabajo	2021/09/19	2021/09/20
1.3	Objetivos del trabajo	2021/09/21	2021/09/22
1.4	Enfoque y método seguido	2021/09/23	2021/09/24
1.5	Planificación del trabajo	2021/09/25	2021/09/27
1.6	Estructura de la memoria	2021/09/28	2021/09/30
1.7	Entre PEC1	2021/09/30	2021/09/30
2	**PEC2: Estado del Arte**	**2021/09/29**	2021/10/10
2.1	Información sobre arritmias	2021/09/29	2021/09/30
2.2	Artículos técnicos	2021/10/01	2021/10/06
2.3	Problemas actuales	2021/10/07	2021/10/08
2.4	Comparación entre algoritmos	2021/10/09	2021/10/10
2.5	Entrega PEC2	2021/10/10	2021/10/10
3	**PEC3: Diseño e impllementación**	**2021/10/11**	2021/12/03
3.1	Diagrama de bloques del diseño	2021/10/11	2021/10/17
3.2	Especificación de cada bloque	2021/10/18	2021/10/24
3.3	Implementación en vhdl de cada bloque	2021/10/25	2021/11/14
3.4	Integración de todos los bloques	2021/11/15	2021/11/18
3.5	Simulación del diseño y verificación	2021/11/19	2021/12/03
3.6	Entrega PEC3	2021/12/03	2021/12/03
4	**PEC4: Desarrollo de la memoria**	**2021/12/04**	2021/12/27
4.1	Realización de la memoria	2021/12/04	2021/12/27
4.2	Entrega PEC4	2021/12/27	2021/12/27
5	**PEC5: Defensa del TFM**	**2021/12/28**	2022/01/05
5.1	Elaboración de la presentación	2021/12/28	2021/12/31
5.2	Preparación de la defensa	2022/01/01	2022/01/05
5.3	Entrega PEC5	2022/01/05	2022/01/05

Figura 1.1: *Diagrama de Gantt.*

1.5. Breve sumario de productos obtenidos

El resultado final de este proyecto es un algoritmo en *VHDL* para detectar complejos QRS y arritmias. Dicho algoritmo está basado en el de *Pan y Tompkins*. En primer lugar se realiza un filtrado y acondicionamiento de la señal original ECG y tras ello, la *FPGA* realiza el procesado de la señal ECG filtrada.

1.6. Breve descripción de los otros capítulos de la memoria

El presente proyecto está estructurado en cinco diferentes capítulos: el primero muestra una introducción, en la cual se expone el contexto en el que se encuentra este trabajo, por qué se

realiza, motivaciones y objetivo principal del mismo.

El segundo capítulo consiste en un estudio del estado del arte donde se explica cómo funciona un latido del corazón, qué es una arritmia y tipos de arritmias. A continuación se realiza una investigación y estudio de diferentes algoritmos de detección de arritmias, mediante el procesado de señales ECG a través de *FPGAs* con *VHDL*. Se destacan los puntos más importantes de cada uno de ellos. Los tres algoritmos que se analizan son: el algoritmo de *Pan y Tompkins* para la detección del complejo QRS, caracterización de las señales ECG a través de los *polinomios de Hermite* y detección del complejo QRS a través del mapeo de coordenadas de retardo. En este proyecto se va a usar el algoritmo de *Pan y Tompkins*.

El tercer capítulo está enfocado en la parte más técnica del proyecto. Se expondrá la estructura del diseño, las distintas etapas que forman el acondicionamiento de las señales ECG (diferentes filtros utilizados), los bloques que lo forman, el porqué y su funcionamiento.

El cuarto capítulo explica, en detalle, el *VHDL* desarrollado para el algoritmo empleado, el cálculo de las pulsaciones por minuto y la explicación, también en detalle, de cada bloque implementado en *VHDL*.

En el último capítulo se realiza una evaluación de los resultados y conclusiones finales, posibles futuros desarrollos, mejoras del proyecto y algoritmo implementado. Se comparan los resultados reales y los resultados obtenidos gracias al algoritmo, es decir, si la señal ECG analizada indica posible arritmia o no. Con este análisis se obtiene la fiabilidad y consistencia del algoritmo, el porcentaje de éxito y en qué situaciones el algoritmo es menos fiable o no es capaz de detectar correctamente el resultado esperado. Los datos de señales ECG se van a obtener de la base de datos pública *MIT-BIH*: https://physionet.org/content/mitdb/1.0.0/

Capítulo 2

Estado del arte

2.1. ECG y arritmia

El corazón es uno de los órganos más importantes del cuerpo humano. Pertenece al sistema circulatorio y se encarga de bombear la sangre hacia el resto del cuerpo a través de impulsos eléctricos. El corazón se divide en corazón derecho e izquierdo. La parte derecha está formada por la aurícula y ventrículo derechos, que están comunicados mediante una válvula denominada válvula tricúspide. Ésta permite el flujo de sangre de la aurícula al ventrículo. Por otro lado, el corazón izquierdo sigue la misma estructura pero la válvula recibe el nombre de válvula mitral. Cada pulso se divide en sístole (contracción) y diástole (relajación). Estos impulsos eléctricos producen una forma de onda muy característica denominada electrocardiograma (ECG o EKG, del alemán *Elektrokardiogramm* [4]).

El médico Willem Einthoven fue el inventor del ECG, quién descubrió que el corazón generaba una actividad eléctrica y que podía representarse mediante un electrocardiógrafo. Gracias a esto recibió en 1924 el Premio Nobel de medicina.

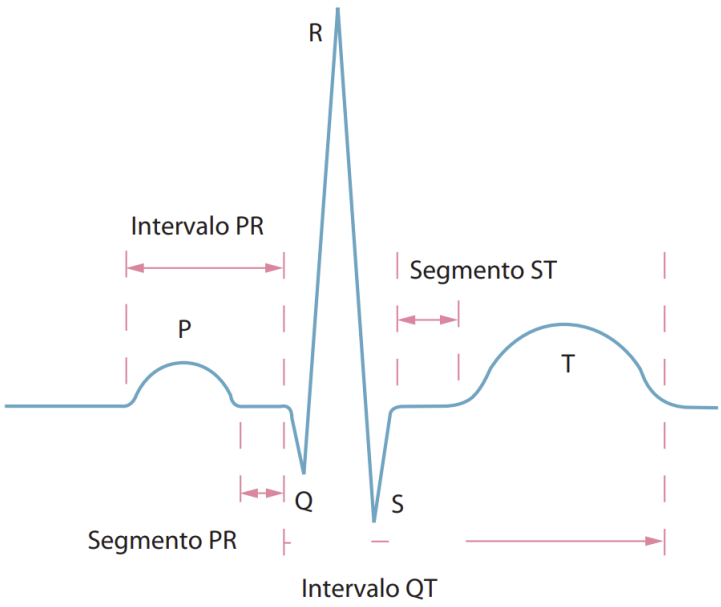


Figura 2.1: *Señal electrocardiograma (Fundación BBVA).*

La señal ECG se divide en varias ondas y segmentos:

1. **Onda P**: la onda P representa el momento en el que la sangre de las aurículas viaja hacia los ventrículos a través de las válvulas que los conectan.

2. **Segmento P-R**: los ventrículos se encuentran llenos de sangre y es el momento previo a la contracción de los ventrículos.

3. **Complejo QRS**: los ventrículos se contraen y expulsan su contenido sanguíneo a través de las arterias.

4. **Segmento S-T**: el segmento S-T se encuentra justo después de la onda S y antes de la onda T. Representa el momento previo al llenado de las aurículas.

5. **Onda T**: el corazón encuentra un momento de relajación con las aurículas llenas y está preparado para un pulso nuevo.

Lo descrito anteriormente es un pulso cardiaco normal, que se encuentra normalmente entre 60 y 100 latidos por minuto. Hay que tener en cuenta diversos factores como: edad, nivel físico, actividad, etc. Cuando se produce una pérdida del ritmo del corazón debido a la alteración de los impulsos eléctricos que controlan el pulso cardiaco, se estará hablando de arritmias. Éstas se dividen en: taquicardias, cuando el corazón supera los 100 lpm, bradicardias cuando las pulsaciones son menores a 60 lpm o latidos irregulares [3].

Figura 2.2: *A: pulso normal. B: arritmia (Fundación BBVA).*

Figura 2.3: *A: taquicardia. B: bradicardia (Fundación BBVA).*

En las figuras anteriores se pueden observar los diferentes ejemplos de arritmias comentados anteriormente.

Las enfermedades cardiovasculares son la primera causa de muerte en España alcanzando un 28.30 % del total de muertes en 2018, 120.859, por tanto la detección de arritmias y monitorización de señales ECG se ha convertido en un objetivo prioritario [10]. Las arritmias pueden provocar insuficiencia cardiaca, infarto cerebral o incluso la muerte, como la muerte súbita.

Para la realización de un ECG se necesitan unos parches (electrodos), que actúan como sensores que se colocan sobre la piel y unos cables conectados al electrocardiógrafo para transmitir los impulsos eléctricos detectados. Este aparato es el encargado de tratar y acondicionar la señal. Estas señales son representadas y lo que se observa es lo que se conoce como ECG.

La detección de trastornos cardiovasculares es una de las prioridades de estudio hoy en día porque es la mayor causa de muerte en adultos. Normalmente, el individuo no es consciente de sufrir una enfermedad cardiovascular hasta que una tragedia ocurre. Por tanto, gracias a la monitorización de pulsos irregulares del corazón se pueden detectar diferentes tipos de arritmias. En algunos casos no presentan mayor complicación pero en otros pueden provocar la muerte. Debido a esto, el desarrollo e implementación algoritmos para su detección puede ser clave para salvar vidas.

2.2. Algoritmos para la detección de arritmias

La presencia de arritmias y trastornos cardiovasculares es uno de los problemas hoy en día debido a la gran cantidad de fallecimientos producidos por éstos. Para detectarlas mediante el procesado de señales ECGs es necesario un algoritmo. A continuación se explicarán en detalle tres de ellos. El algoritmo más usado y conocido es el de *Pan y Tompkins*:

2.2.1. Algoritmo de *Pan y Tompkins*

El algoritmo de *Pan y Tompkins* es un algoritmo utilizado para la detección del complejo QRS de la señal ECG. Como se ha comentado anteriormente, el complejo QRS representa la despolarización de los ventrículos, en donde éstos se contraen y bombean la sangre a través de las arterias. Es útil para medir las pulsaciones por minuto, ya que también está enfocado en la detección de la señal R [13]-[14]. Está formado por varias etapas de procesado:

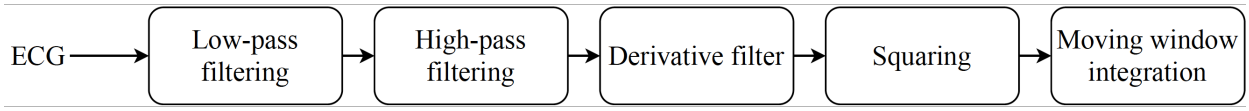


Figura 2.4: *Etapas del algoritmo Pan y Tompkins (diagrama de bloques).*

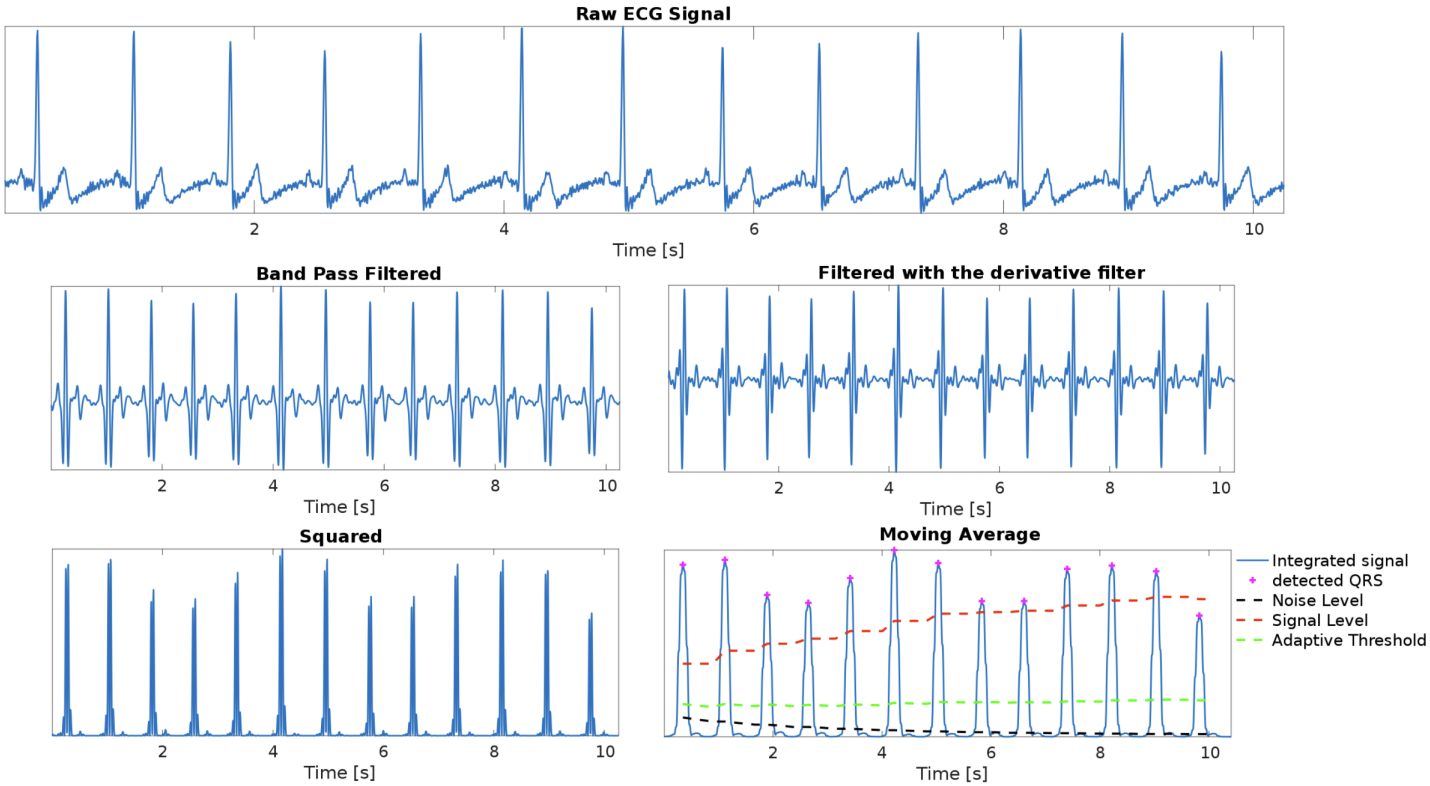


Figura 2.5: *Etapas del algoritmo Pan y Tompkins (representación).*

2.2.1.1. Filtrado

El primer paso del algoritmo es un filtrado para la cancelación de ruido e incremento de la relación *SNB*. Se implementa un filtro paso banda entre 5 y 15 Hz que reduce las falsas detecciones causadas por los diversos tipos de interferencia presentes en las señales de ECG. En este caso se añade un filtro paso bajo y después un filtro paso alto, con ello queda implementado el filtro paso banda [16].

2.2.1.2. Derivador

El filtro derivador se utiliza para obtener información de la pendiente. Este filtro se usa para diferenciar la señal R de la señal T y P, porque su cambio habitualmente es más brusco y de mayor pendiente.

2.2.1.3. Cuadrado e integrador

Se eleva al cuadrado la señal convirtiéndola en positiva y el integrador es usado para detectar los máximos del complejo QRS, en este caso detecta los picos de la señal R.

2.2.1.4. Decisiones y funcionamiento

El algoritmo está basado en una toma de decisiones que dependen de unos *thresholds*. En primer lugar se detectan los picos del complejo QRS buscando cambios bruscos de dirección. Una vez se detecta un pico, se compara a un *threshold* para reducir la probabilidad de error.

2.2.2. Polinomios de Hermite

Los polinomios de Hermite son polinomios ortogonales que se usan en diferentes sectores como procesado de señales, probabilidad o física entre otros. Este algoritmo reconstruye y caracteriza el complejo QRS a través de polinomios de un grado determinado dando lugar a $x(t)$. Estos polinomios se pueden computar recursivamente para una mayor precisión [9]-[11]. Este polinomio se define como:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{\frac{-x^2}{2}}$$

La aproximación de la señal x(t) se expresa como:

$$x(t) = \sum_{n=0}^{N-1} c_n(\sigma)\phi_n(t,\sigma)$$

siendo:

$$\phi_n(t,\sigma) = \frac{1}{\sqrt{\sigma 2^n n! \sqrt{\pi}}} e^{-t^2/2\sigma^2} H_n(t/\sigma)$$

El polinomio de Hermite H_n puede ser computado recursivamente donde n es el grado del polinomio [6]-[15]:

$$\begin{aligned} H_0(t) &= 1 \\ H_1(t) &= 2t \\ H_2(t) &= 4t^2 - 2 \\ H_3(t) &= 8t^3 - 12t \\ H_4(t) &= 16t^4 - 48t^2 + 12 \end{aligned}$$

El parámetro σ es el factor de escala del polinomio que ajusta el ancho de la ventana al ancho del complejo QRS que se puede obtener mediante:

$$c_n(\sigma) = \sum_t x(t) \cdot \phi_n(t,\sigma)$$

La caracterización de las señales ECG a través de los polinomios de hermite se puede ver en la siguiente figura:

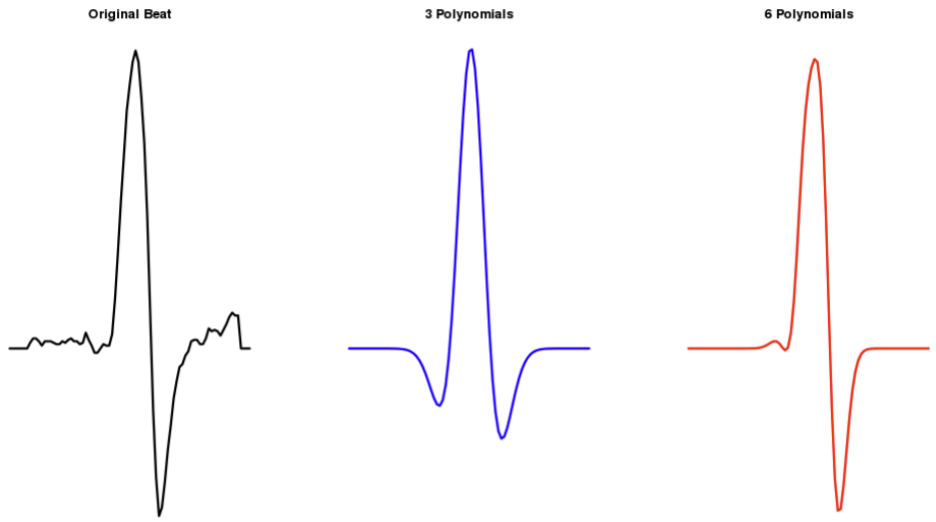
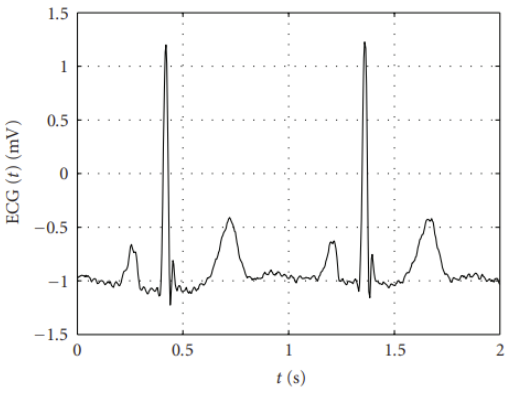


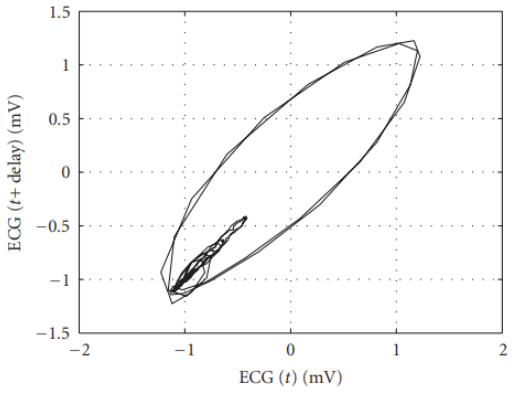
Figura 2.6: *Izquierda: ECG original; centro: polinomio grado 3; derecha: polinomio grado 6.*

2.2.3. Mapeo de coordenadas de retardo

Este algoritmo está basado en la detección del complejo QRS mediante retardos para mapear la señal ECG. La señal ECG y su duplicado retardado crean un retrato del espacio de fases, cuyas propiedades geométricas se explotan para la detección de complejos QRS. Dependiendo del área del polígono delimitada por una serie de puntos de datos consecutivos, se puede encontrar el complejo QRS. Cuando se encuentra un nuevo polígono y su área excede al nivel del umbral establecido, se toma una decisión de si el pico es realmente un complejo QRS y se actualiza el nivel de umbral de detección [8]-[12]. Esta representación se realiza en un plano 2D a partir de la señal original y su duplicado retardado siendo el resultado:



(a) *Señal ECG de entrada.*



(b) *Retrato de espacio-fase de la señal ECG con un retardo de 4 ms.*

Se pueden observar tres diferentes áreas: las dos áreas más pequeñas son las señales P y T, mientras que el área más grande hace referencia al complejo QRS.

2.3. Otros algoritmos

Existen otros algoritmos como el algoritmo basado en la Transformada ondícula, redes neuronales o máquinas de vectores. El primero de ellos se puede usar en *wearables* para la detección de complejos QRS. Esta técnica es ideal para este tipos de dispositivos porque tiene una gran relación señal-ruido, la detección es buena y no necesita muchos coeficientes por lo que reduce los recursos necesarios en el hardware, lo que disminuye el consumo [5].

En segundo lugar, utilizando redes neuronales. Por ejemplo, una red neuronal convolucional de una dimensión. Este método tiene como objetivo extraer automáticamente las características morfológicas del complejos QRS. Éstas características extraídas son procesadas por un perceptrón multicapa, para la detección del complejo QRS [17].

Por último, la máquina de vectores, basada en un aprendizaje supervisado y en problemas de clasificación y regresión. Dicha máquina es entrenada mediante diferentes muestras para construir un modelo que sea capaz de detectar los complejos QRS [2].

Muchos de los algoritmos desarrollados para detección de complejos QRS y arritmias se implementan en *FPGAs*, siendo éstas la mejor opción para aplicaciones biomédicas. Las principales razones son las siguientes: gracias a su arquitectura interconectada y paralela, se pueden procesar datos en paralelo a través de algoritmos complejos en tiempo real con poco coste y bajo consumo. Son reconfigurables, por lo que es muy sencillo realizar modificaciones en el algoritmo y volcarlo en la *FPGA*. Asimismo, las *FPGAs* han incrementado su velocidad y capacidad de procesamiento en los últimos años, siendo ideales para procesamiento de señales gracias a su gran paralelismo. Además, se puede realizar todo el filtrado en la propia *FPGA*. Por contra, no es posible trabajar con números decimales, ya que la descripción del hardware en *VHDL* solo es posible en circuitos digitales, lo que puede provocar una peor precisión. Sin embargo, existen soluciones a este problema como el uso de tipos específicos proporcionados por el propio lenguaje [1].

En este proyecto los datos de las señales ECG se obtendrán de la base de datos *MIT-BIH* de arritmias. El algoritmo que se va a desarrollar estará basado en el algoritmo de *Pan y Tompkins*, ya que el porcentaje de éxito se encuentra cercano al 100 % y su implementación en *VHDL* es menos compleja que otros algoritmos.

Capítulo 3

Estructura del diseño

El diseño cuenta de dos bloques principales y dentro de cada bloque unos subbloques: el primer bloque es un filtrado de las señales ECG a través de *Matlab*, el segundo es el procesado en *VHDL* del ECG filtrado, con el que se detectan los complejos QRS y las pulsaciones por minuto.

3.1. Matlab

En *Matlab* se realiza el procesado del ECG original basándose en el algoritmo de *Pan y Tompkins*. Los datos de los ECG procesados se han obtenido de la base de datos de *MIT-BIH Arrhythmia Database*. Cada segundo contiene 360 muestras con una resolución de 11 bits y un rango de 10 mV. Contiene 48 señales ECG diferentes, aunque en este proyecto no se van a procesar todas.

3.1.1. Filtro paso banda

El primer filtrado que se realiza es un paso banda entre 4 y 15 Hz, con esto se pueden suprimir las pequeñas oscilaciones del ECG o del ruido aumentando la relación S/N de la señal.

3.1.2. Derivador

El derivador se usa para diferenciar correctamente la señal R de las señales P y T, porque la pendiente en el complejo QRS es mayor, sin embargo puede suceder que el valor de las señales P o T sean muy parecidos al valor de la señal R. Si esto sucede puede provocar que el procesado no sea capaz de suprimirlas y como resultado el algoritmo lo detecte como complejo QRS.

3.1.3. Cuadrado

Se eleva al cuadrado la señal para trabajar únicamente con valores positivos.

3.1.4. Integrador

El integrador realiza la media de determinadas muestras, en este caso se realiza la media con una ventana de 150 ms, teniendo en cuenta que cada segundo son 360 muestras, 150 ms

equivalen a:

$$n = 360 \cdot 0{,}15 = 54\ muestras$$

Por lo que la media se realiza cada 54 muestras. Cuanto más grande sea la ventana del integrador más suavizada queda la señal y los picos son menos bruscos. En este último paso también se multiplica la señal por 10^6 para poder trabajar en la *FPGA* con decimales como si fuesen enteros y no con números reales. Es decir, si se tiene una muestra de valor 0,000524 a la *FPGA* le llegará el valor 524.

Si realizamos el procesado de la muestra 100 de la base de datos:

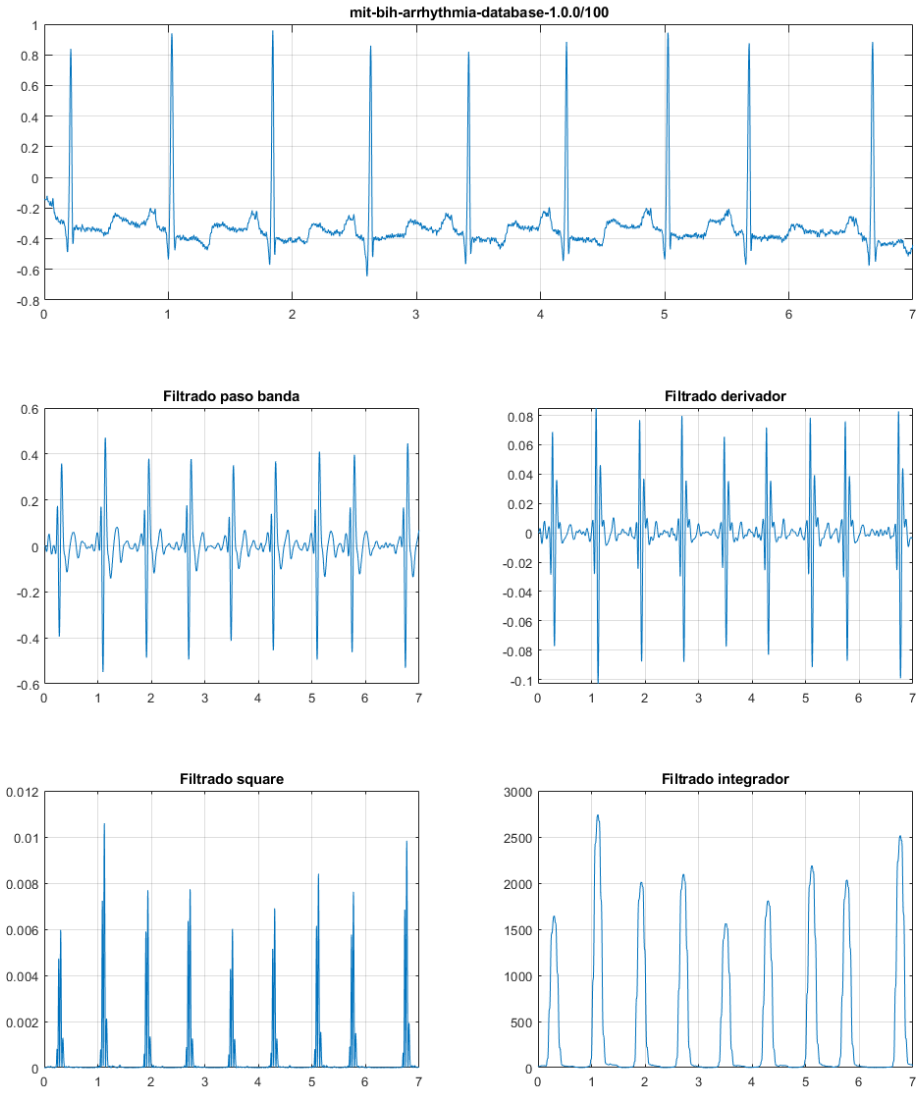


Figura 3.1: *Procesado del ECG número 100 de la base de datos.*

El eje x son segundos y el eje y son mV en las dos primeras figuras (ecg original y el paso banda). Como se puede apreciar en la figura 3.1 los ocho picos quedan muy definidos tras el filtro integrador y gracias a ello el algoritmo implementado en la *FPGA* podrá detectarlos correctamente. Los datos son guardados en ficheros, en este caso con 2520 muestras, que equivalen a siete segundos.

3.2. Implementación *VHDL*

El diseño VHDL está estructurado en diferentes bloques:

1. **Bloque principal *top-level struct***: este bloque es el estructural, que contiene los otros bloques más pequeños. Contiene las entradas y salidas del sistema completo.

2. **Bloque *sync***: se encarga de registrar las señales de entrada para evitar problemas de metaestabilidad.

3. **Bloque *fsm***: la máquina de estados se encarga de gestionar en qué parte del procesado se encuentra el diseño a medida que cambian las entradas.

4. **Bloque *fe detection***: se encarga de detectar flancos de bajada de la entrada, que en este caso es la que habilita el procesado.

5. **Bloque *load data***: se encarga de cargar el ECG que se va procesar dependiendo de la entrada correspondiente.

6. **Bloque *process ecg***: se encarga procesar el ECG, detectar sus complejos QRS, así como de los tiempos en los que ha sido detectado cada uno de ellos. Posteriormente calcula las pulsaciones por minuto y activa las salidas de taquicardia, bradicardia o ECG estable y normal.

Capítulo 4

Implementación *VDHL*

La implementación del algoritmo se ha realizado en *VDHL*. Éste es un lenguaje de descripción de hardware y se utiliza para describir circuitos digitales. Sus siglas son la unión de *HDL* (*Hardware Description Language*) y *VHSIC* (*Very High Speed Integrated Circuit*). El diseño se divide en un bloque estructural que conecta todos los diferentes bloques y pequeños submódulos, todos ellos implementados en *VDHL*. En primer lugar se explicará el algoritmo en detalle, posteriormente el cálculo de los latidos por minuto y por último cada bloque *VDHL*:

4.1. Algoritmo de detección de complejos QRS

El algoritmo se basa en *Pan y Tompkins* y se divide en dos fases: la primera consiste en un filtrado usando *Matlab* y la segunda en un procesado a través de la *FPGA*.

La señal ECG a filtrar está formada por 2520 muestras, con una frecuencia de muestreo de 360, por tanto, se procesan siete segundos. En primer lugar, el algoritmo procesa todas las muestras para obtener la de valor máximo. Este valor máximo permite configurar el primer parámetro del algoritmo. Dicho parámetro establece cuál será el valor mínimo que debe tener una muestra para no ser descartada. El valor máximo obtenido anteriormente, se divide por una constante y cualquier valor por debajo de esa división se descarta. Por ejemplo, si el valor máximo del ECG completo es de 5000 y se ha configurado este parámetro con un valor de 10, el algoritmo realizaría la siguiente operación:

$$\frac{5000}{10} = 500$$

Todas las muestras con valor menor a 500 serán descartadas. Este parámetro es interesante y necesario, ya que en determinados casos tras el procesado aparecen picos de tamaño reducido. En caso de no establecerse un mínimo, puede darse el caso de que el algoritmo detecte algunos picos de forma incorrecta como complejos QRS. Cuánto menor sea la constante, mayor serán los picos que se descarten.

Una vez se obtiene la muestra con valor máximo y se configura este parámetro, comienza la detección de complejos QRS. Empezando por la muestra número cero, el algoritmo anali-

za consecutivamente muestras para detectar subidas y potenciales complejos QRS. Para ello, almacena el valor de la muestra actual y la compara con la media de las cuatro próximas muestras contando a partir de la tercera. Es decir, si la muestra actual es la número 10, ésta se compara con la media de la muestras 13, 14, 15 y 16. Si la media de las muestras calculadas es suficientemente más grande que la muestra actual, el algoritmo detectará una subida. Este valor de comparación es configurable y se establece en porcentaje. Por ejemplo: si se establece este parámetro a un 25 % y la media de las cuatro muestras siguientes es un 25 % mayor a la muestra actual, se activará la señal *rising*, que indica una subida. El complejo QRS se detecta cuando la muestra actual es un 25 % mayor que la media de las cuatro muestras siguientes, es decir, se hace a la inversa.

Cada vez que un complejo QRS es detectado, la información relativa a éste se almacena en la señal *ecg_info*, que es un *record*. Dicha señal guarda la siguiente información: número de complejos QRS, valor máximo de cada complejo y el número de la muestra en donde se detectó dicho complejo. Una vez se han procesado todas las muestras, se habilita la señal *trig_dec* para realizar los cálculos de las pulsaciones por minuto, así como la toma de decisiones de taquicardia, bradicardia o ECG normal.

4.2. Cálculo de las pulsaciones por minuto

4.2.1. Cálculo teórico

Para calcular las pulsaciones por minuto se necesita saber: números de complejos QRS detectados y en qué muestra se ha detectado cada uno. Por ejemplo: se va a suponer que se han detectado siete complejos QRS en las muestras 300, 700, 1050, 1400, 1700, 2100 y 2450. Gracias a esta información, almacenada en la señal *ecg_info*, se puede calcular las pulsaciones por minuto. En primer lugar, se calcula el tiempo entre la primera y la última detección:

$$\frac{2450 - 300}{360} = 5{,}972 \approx 6 \; segundos$$

Una vez calculada la duración en segundos entre la primera detección y la última, y conociendo el número de complejos QRS detectados, ya se puede calcular las pulsaciones por minuto:

$$\frac{60}{6} \cdot 7 = 70 \; pulsaciones \; por \; minuto$$

4.2.2. Cálculo en la *FPGA*

Las *FPGAs* no pueden trabajar con números décimales. Por tanto, tras realizar el primer cálculo:

$$\frac{2450 - 300}{360} = 5{,}972 \approx 5 \; segundos$$

La *FPGA* aproxima a 5 segundos porque siempre redondea hacia abajo. Si se realiza la misma operación para la obtención de los latidos por minuto:

$$\frac{60}{5} \cdot 7 = 84 \; pulsaciones \; por \; minuto$$

Esto implica un error del 20 %. Para solucionarlo, se ha incluido en el cálculo dos decimales. Como la *FPGA* no trabaja con decimales, se multiplica por 100 la resta de la muestra máxima y la muestra mínima:

$$100 \cdot \frac{2450 - 300}{360} = 597{,}22 \approx 597$$

Ahora en vez de 5,97 se obtiene tiene 597, en donde el 97 son los dos decimales calculados. Se vuelve a calcular las pulsaciones por minuto:

$$\frac{60 \cdot 100}{597} \cdot 7 \approx 70 \; pulsaciones \; por \; minuto$$

Este caso concreto es correcto y funciona, pero en algunos casos el error puede ser igual de grande que en el caso anterior. Si se realiza la siguiente operación:

$$\frac{60 \cdot 100}{597} = 10{,}05 \; segundos$$

Se aproxima a 10 segundos y el cálculo es correcto, pero ¿qué sucedería si en vez de 5,97 segundos son 5,70?

$$\frac{60 \cdot 100}{570} \cdot 7 = 10{,}53 \cdot 7 \rightarrow 10 \cdot 7 = 70 \; pulsaciones \; por \; minuto$$

Es decir, tanto si es 5.97 segundos como 5.70 las pulsaciones son las mismas, lo que es incorrecto. Para solucionarlo, en vez de multiplicar el 60 por 100, se multiplica por 10 000 y después de ese cálculo se divide todo por 100:

$$\frac{60 \cdot 10000}{597} \cdot 7 \approx 7035 \rightarrow \frac{7035}{100} \approx 70 \; pulsaciones \; por \; minuto$$

$$\frac{60 \cdot 10000}{570} \cdot 7 \approx 7368 \rightarrow \frac{7368}{100} \approx 73 \; pulsaciones \; por \; minuto$$

Se observa que de esta forma el cálculo es más preciso y fiable.

4.3. Estructura de bloques VHDL

4.3.1. Bloque principal *top-level struct*

La estructura del bloque estructural es la siguiente:

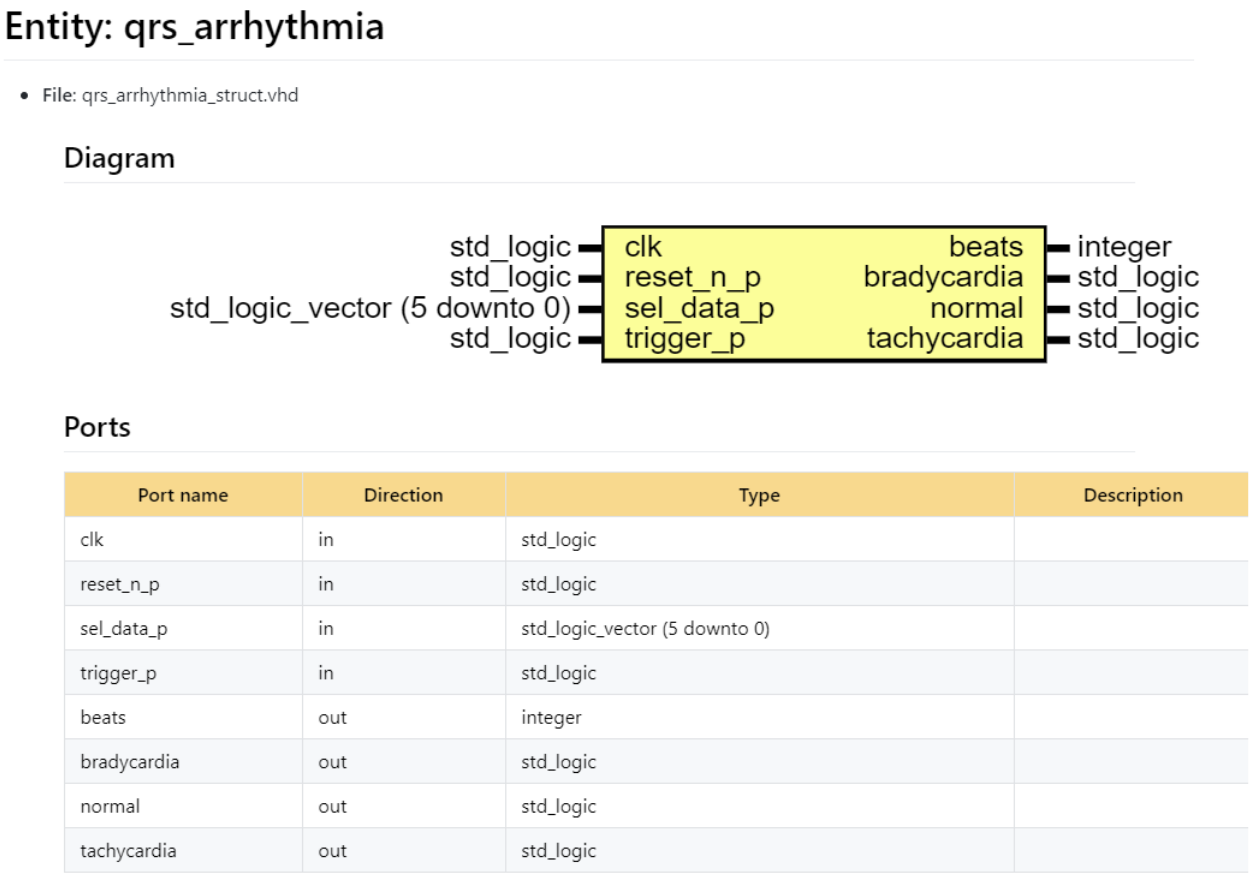


Figura 4.1: *Bloque principal top-level struct.*

Este bloque está formado por las siguientes entradas y salidas:

1. ***clk***: reloj del sistema.
2. ***reset_n_p***: *reset* asíncrono del sistema.
3. ***sel_data_p***: esta entrada selecciona el ECG que se va a procesar, de 0 a 47. Al ser 6 bits, se puede configurar por ejemplo el valor 54, en este caso se selecciona por defecto el ECG número 0.
4. ***trigger_p***: esta entrada acciona el procesado del ECG.
5. ***beats***: esta salida muestra las pulsaciones por minuto una vez el ECG ha sido procesado.

6. ***bradycardia***: esta salida se activa si las pulsaciones son iguales o menores a 60.
7. ***tachycardia***: esta salida se activa si las pulsaciones son iguales o mayores a 100.
8. ***normal***: esta salida se activa si las pulsaciones se encuentran entre 60 y 100.

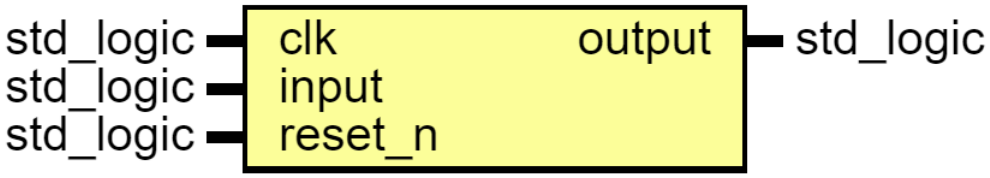
4.3.2. Bloque *sync*

La estructura del bloque *sync* es la siguiente:

Entity: sync

- **File**: sync_rtl.vhd

Diagram



Ports

Port name	Direction	Type	Description
clk	in	std_logic	
input	in	std_logic	
reset_n	in	std_logic	
output	out	std_logic	

Signals

Name	Type	Description
input_d	std_logic	

Processes

- p_sync: (reset_n, clk)

Figura 4.2: *Bloque sync.*

Este bloque se encarga de sincronizar las entradas asíncronas con el reloj de sistema, internamente está formado por dos registros para asegurarse de que no hay problemas de metaestabilidad.

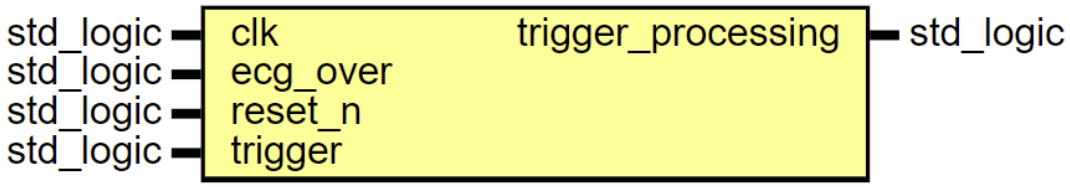
4.3.3. Bloque *fsm*

La estructura del bloque *fsm* es la siguiente:

Entity: fsm

- **File**: fsm.vhd

Diagram



Ports

Port name	Direction	Type	Description
clk	in	std_logic	
ecg_over	in	std_logic	
reset_n	in	std_logic	
trigger	in	std_logic	
trigger_processing	out	std_logic	

Signals

Name	Type	Description
current_state	state_type	
next_state	state_type	

Types

Name	Type	Description
state_type	(ini, processing, finish)	

Processes

- clocked_proc: (clk,
reset_n)
- nextstate_proc: (current_state,
ecg_over,
trigger)
- output_proc: (current_state)

Figura 4.3: *Bloque fsm.*

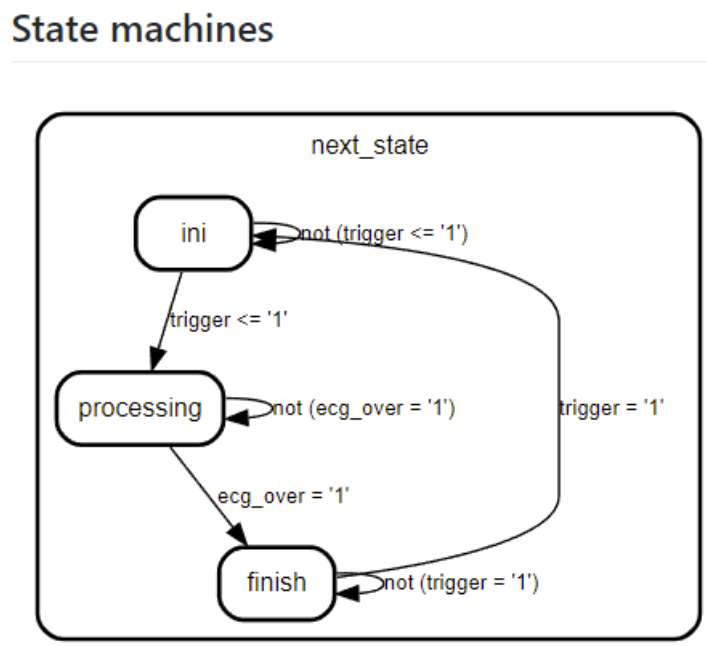


Figura 4.4: *Diagrama de estados del fsm.*

Este bloque se encarga de controlar la máquina de estados del diseño, que está estructurada en tres estados:

1. ***ini***: estado inicial, en este estado se selecciona mediante la entrada *sel_data_p* el ECG que se va a procesar. Una vez se ha seleccionado el ECG que se quiere procesar, se habilita el procesamiento mediante la entrada *trigger*, que provocará el cambio al estado *processing*.

2. ***processing***: en este estado se realiza todo el procesado, detectando los complejos QRS, calculando las pulsaciones por minuto y tomando la decisión de si es arritmia o no.

3. ***finish***: este estado indica que ya se ha realizado el procesado y se está esperando a volver al estado inicial para preparar un nuevo procesado.

El estado inicial de la máquina de estados cuando se realiza un *reset* es el estado *ini*.

La máquina de estados está diseñada mediante tres procesos:

1. ***clocked_proc***: este proceso registra el estado, asignando al estado actual el estado calculado en el proceso *nextstate_proc*.

2. ***nextstate_proc***: este proceso calcula el próximo estado, dependiendo de la entrada *trigger*, que decide cuando se va a realizar el procesado.

3. ***output_proc***: este proceso calcula las salidas. Solo hay una salida y es *trigger_processing*. La activación de esta salida indica el procesado del ECG seleccionado mediante la entrada *sel_data*. Cuando el procesado se ha realizado, la salida se desactiva.

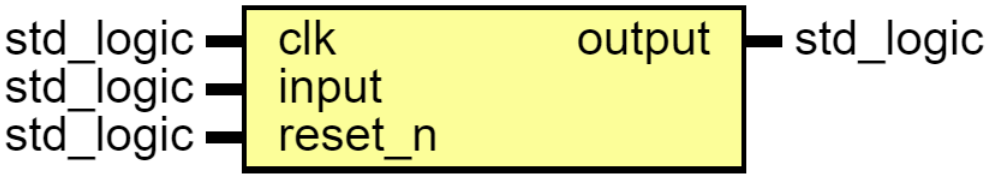
4.3.4. Bloque *falling edge detection*

La estructura del bloque *falling edge detection* es la siguiente:

Entity: fe_detection

- **File**: fe_detection_rtl.vhd

Diagram



Ports

Port name	Direction	Type	Description
clk	in	std_logic	
input	in	std_logic	
reset_n	in	std_logic	
output	out	std_logic	

Signals

Name	Type	Description
input_d	std_logic	

Processes

- p_detect_re_fe: (clk, reset_n)

Figura 4.5: *Bloque falling edge detection.*

Este bloque se encarga de detectar flancos de bajada. Al principio se pensó así simulando la entrada de *trigger* como un pulsador activo a nivel bajo. Cuando se detecta el flanco de bajada, se activa a nivel alto la señal *output* un ciclo de reloj. Esta señal es una entrada del bloque *fsm*.

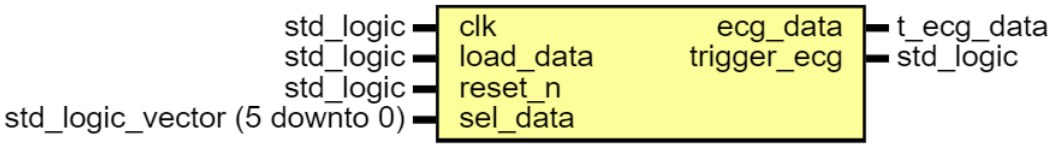
4.3.5. Bloque *load data*

La estructura del bloque *load data* es la siguiente:

Entity: load_data

- **File:** load_data_rtl.vhd

Diagram



Ports

Port name	Direction	Type	Description
clk	in	std_logic	
load_data	in	std_logic	
reset_n	in	std_logic	
sel_data	in	std_logic_vector (5 downto 0)	
ecg_data	out	t_ecg_data	
trigger_ecg	out	std_logic	

Processes

- p_load_data: (reset_n, clk)

Figura 4.6: *Bloque load data.*

Este bloque se encarga de cargar el ECG seleccionado en una señal, que le servirá como entrada al bloque que procesa el ECG. La información de cada ECG está almacenada en una constante. Hay una constante por cada ECG, cada constante es guardada en un array, por tanto, la entrada *sel_data_p* elige la posición del array, que equivale al ECG que se va a procesar. Por último, una vez que el ECG ya ha sido cargado se acciona el procesado en el bloque *process_ecg*.

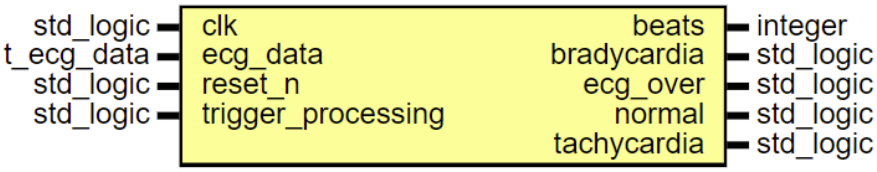
4.3.6. Bloque *process ecg*

La estructura del bloque *process ecg* es la siguiente:

Entity: process_ecg

- File: process_ecg_rtl.vhd

Diagram



Ports

Port name	Direction	Type	Description
clk	in	std_logic	
ecg_data	in	t_ecg_data	
reset_n	in	std_logic	
trigger_processing	in	std_logic	
beats	out	integer	
bradycardia	out	std_logic	
ecg_over	out	std_logic	
normal	out	std_logic	
tachycardia	out	std_logic	

Signals

Name	Type	Description
rising	std_logic	
max_value_relative	integer	
max_value_absolute	integer	
current_value	integer	
compared_value	integer	
trigger_peaks	std_logic	
index	integer	
media	integer	
beats_aux	integer	
ecg_info	t_ecg_info	
trig_decisions	std_logic	

Processes

- p_detect_peaks: (clk, reset_n)
- p_detect_max_value_relative: (clk, reset_n)
- p_detect_max_value: (clk, reset_n)
- p_decision: (clk, reset_n)
- p_calculate_beats: (clk, reset_n)

Figura 4.7: *Bloque process ecg.*

Este bloque es el más complejo: en primer lugar se explicarán las entradas y salidas del bloque, en segundo lugar se explicará qué función realiza cada señal interna y por último, cada proceso qué función tiene dentro del bloque.

4.3.6.1. Entradas y salidas

El reloj y el *reset* asíncrono se van a obviar en la explicación.

1. ***ecg_data***: esta entrada contiene toda la información del ECG que se va a procesar. Es de tipo *t_ecg_data*, que es un array de 2520 muestras de tipo *integer* obtenidas a través del procesado de *Matlab*, que equivalen a siete segundos del ECG.

2. ***trigger_processing***: esta entrada acciona el procesado, es decir, activa el bloque para que comience con el procesado.

3. ***ecg_over***: esta salida se activa cuando todo el procesado del ECG ha sido completado.

4. ***beats***: esta salida muestra las pulsaciones por minuto una vez el ECG ha sido procesado.

5. ***bradycardia***: esta salida se activa si las pulsaciones son iguales o menores a 60.

6. ***tachycardia***: esta salida se activa si las pulsaciones son iguales o mayores a 100.

7. ***normal***: esta salida se activa si las pulsaciones se encuentran entre 60 y 100.

4.3.6.2. Señales internas

1. ***rising***: esta señal se activa cuando se ha detectado una subida en el ECG e indica un potencial complejo QRS.

2. ***max_value_relative***: esta señal almacena el valor máximo del complejo QRS detectado y no del ECG completo.

3. ***max_value_absolute***: esta señal almacena el valor máximo del ECG completo. Este valor es usado para configurar uno de los dos parámetros configurables del algoritmo.

4. ***current_value***: esta señal almacena el valor actual que se está comparando.

5. ***compared_value***: esta señal almacena el valor con el que se está comparando el valor actual.

6. ***trigger_peaks***: esta señal se activa cuando ya se ha calculado el valor máximo absoluto del ECG y comienza la detección de complejos QRS.

7. ***index***: esta señal indica qué muestra se está comparando.

8. ***media***: esta señal almacena la media de cuatro muestras.
9. ***beats_aux***: esta señal almacena las pulsaciones por minuto. Se usa una señal auxiliar porque se va a leer su valor para la toma de decisiones de las señales de salida.
10. ***ecg_info***: esta señal almacena la información relevante del ECG procesado. Guarda el número de complejos QRS detectados, el valor de ese complejo QRS y la muestra en la que se ha detectado. Esta información se usa para calcular las pulsaciones por minuto.
11. ***trig_decisions***: esta señal acciona la toma de decisiones una vez se ha procesado el ECG.

4.3.6.3. Procesos

1. ***p_detect_peaks***: este proceso detecta complejos QRS.
2. ***p_detect_max_value_relative***: este proceso dentro del complejo QRS detectado, calcula el valor máximo.
3. ***p_detect_max_value***: este proceso almacena el valor máximo de las 2520 muestras del ECG.
4. ***p_decision***: este proceso es el encargado de decidir si es una taquicardia, bradicardia o ECG normal.
5. ***p_calculate_beats***: este proceso calcula las pulsaciones por minuto.

Capítulo 5

Resultados y conclusiones

5.1. Simulación completa

Para validar el algoritmo, éste ha sido probado mediante simulaciones (*Testbenches*) con la herramienta de *Modelsim*, una herramienta proporcionada por *Altera (Intel)*. En la siguiente simulación se va mostrar el procesado del ECG número 100 obtenido a través de la base de datos *MIT-BIH* de arritmias:

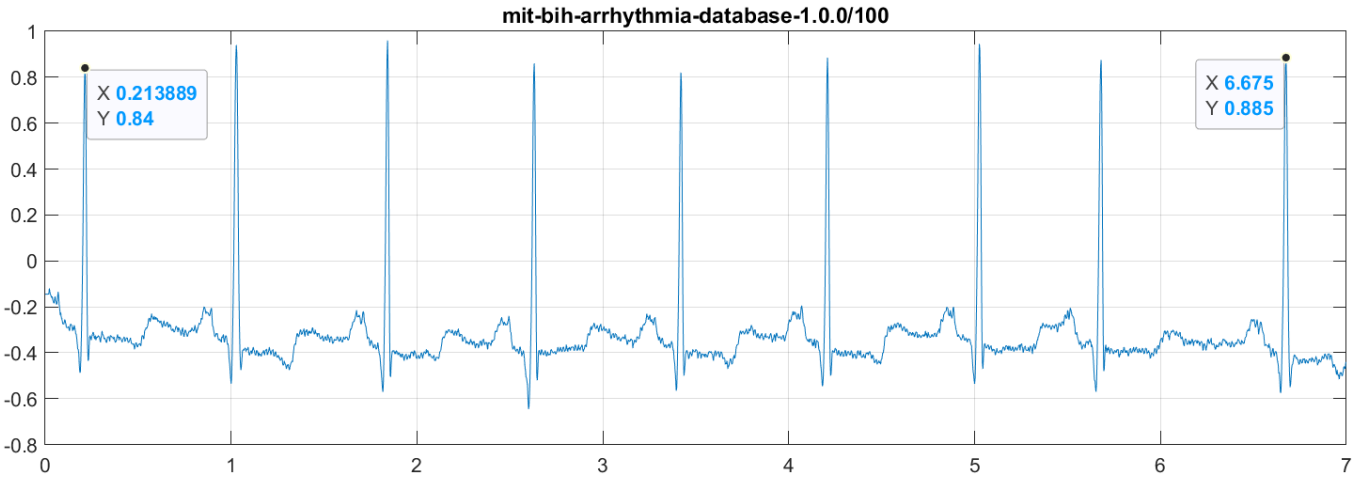


Figura 5.1: *ECG número 100.*

Como se ve en la figura 5.1 la duración es de siete segundos, en donde se aprecian nueve complejos QRS. Para el cálculo de las pulsaciones por minuto se realiza la siguiente operación:

$$9 * \left(\frac{60}{6{,}675 - 0{,}214} \right) = 83{,}58 \approx 83 \; pulsaciones \; por \; minuto$$

El algoritmo tiene una resolución de dos decimales y siempre redondea hacia abajo. A continuación se va a explicar cómo funciona el algoritmo en el diseño *VHDL* apoyándose en la siguiente figura:

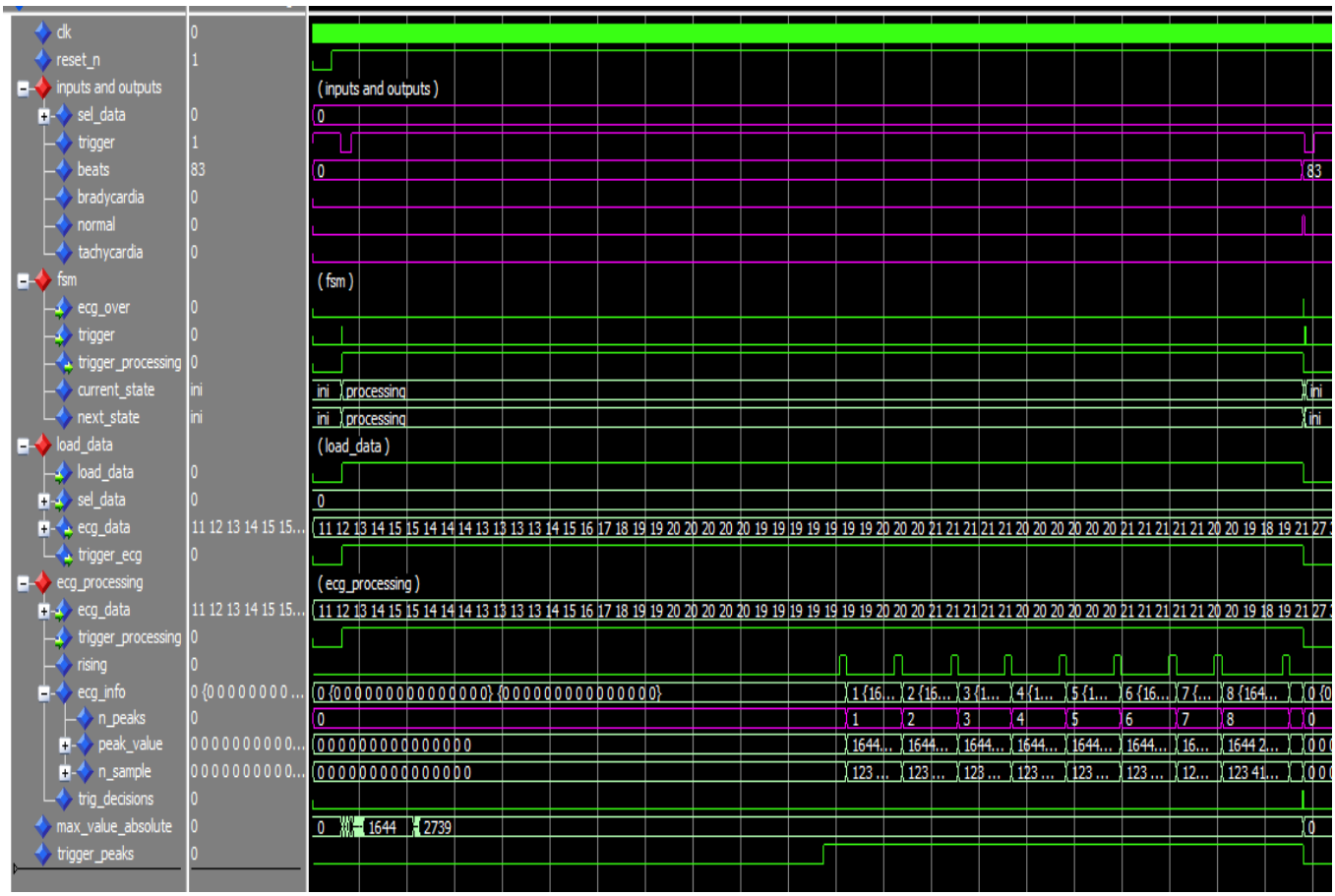


Figura 5.2: *Simulación del ECG número 100.*

Tras el *reset* asíncrono el sistema se encuentra en el estado *ini*, esperando que se active el procesado mediante la señal *trigger*, que se habilita a nivel bajo. Una vez se acciona, se transiciona al estado *processsing* y comienza el procesado activándose la señal *trigger_processing*. En primer lugar, la *FPGA* obtiene el valor máximo de todas las muestras del ECG y lo almacena en la señal *max_value_absolute*. Una vez se han analizado todas las muestras para la obtención de este valor, se habilita la señal *trigger_peaks*, que inicializa el algoritmo.

La señal *rising* se activa cuando se está ante una subida y un posible potencial complejo QRS. Su desactivación implica la detección de un complejo QRS y la información se almacena en la señal *ecg_info*, que almacena número de complejos QRS detectados, valor máximo de ese complejo y en qué muestra se detectó el pico de la señal R. Este proceso es iterativo hasta que se llega a la última muestra y se indica mediante la activación de la señal *trig_decisions*.

Seguidamente, se realiza la toma de decisiones y cálculo de las pulsaciones por minuto. En este caso, como se calculó anteriormente son 83 latidos por minuto y como consecuencia se activa la salida *normal* porque se encuentra entre 60 y 100 pulsaciones por minuto. Por último, se activa la señal *ecg_over* indicando el final del procesado y cambio al último estado *finish*.

Habilitando de nuevo la entrada *trigger*, se transicionará al estado inicial *ini* y se podrá repetir el procesado.

A continuación se prueban las muestras ECG número 213 y 215, obtenidas de la base de datos *MIT-BIH* de arritmias, declaradas como taquicardias ventriculares en la tabla de ritmos de la página web: https://archive.physionet.org/physiobank/database/html/mitdbdir/tables.htm#allrhythms:

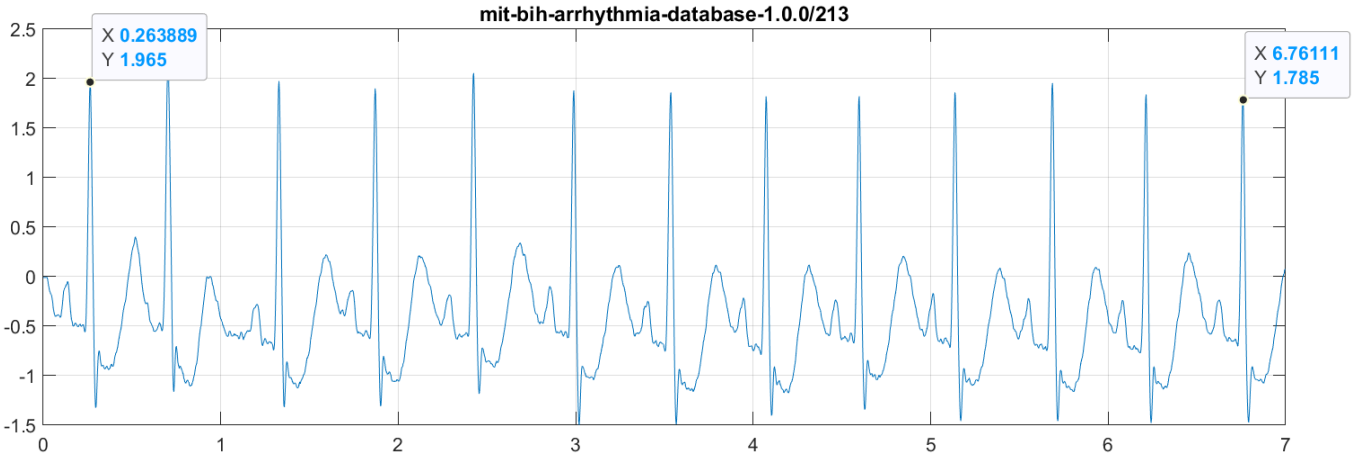


Figura 5.3: *ECG número 213.*

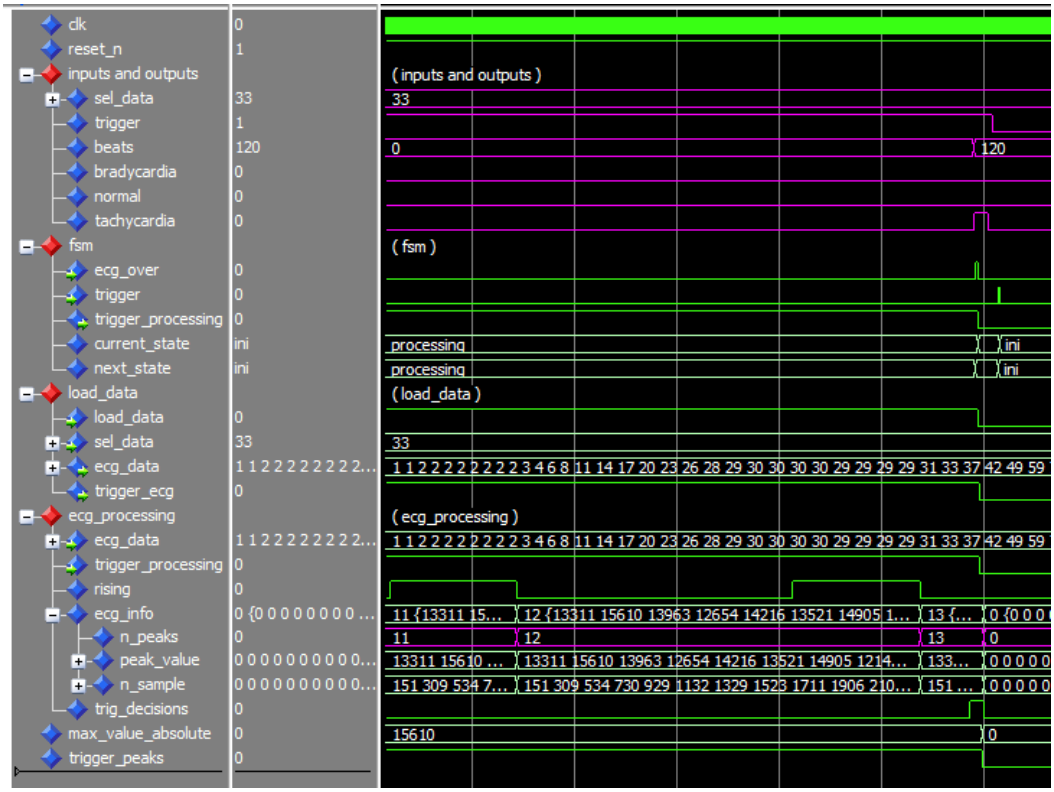


Figura 5.4: *Simulación del ECG número 213.*

Se puede observar que este ECG cuenta con 13 complejos QRS, por tanto el algoritmo los está detectando correctamente. Para calcular las pulsaciones por minuto:

$$13 * \left(\frac{60}{6{,}76 - 0{,}26} \right) \approx 120 \; pulsaciones \; por \; minuto$$

Siguiendo con el ECG número 215 el resultado es similar:

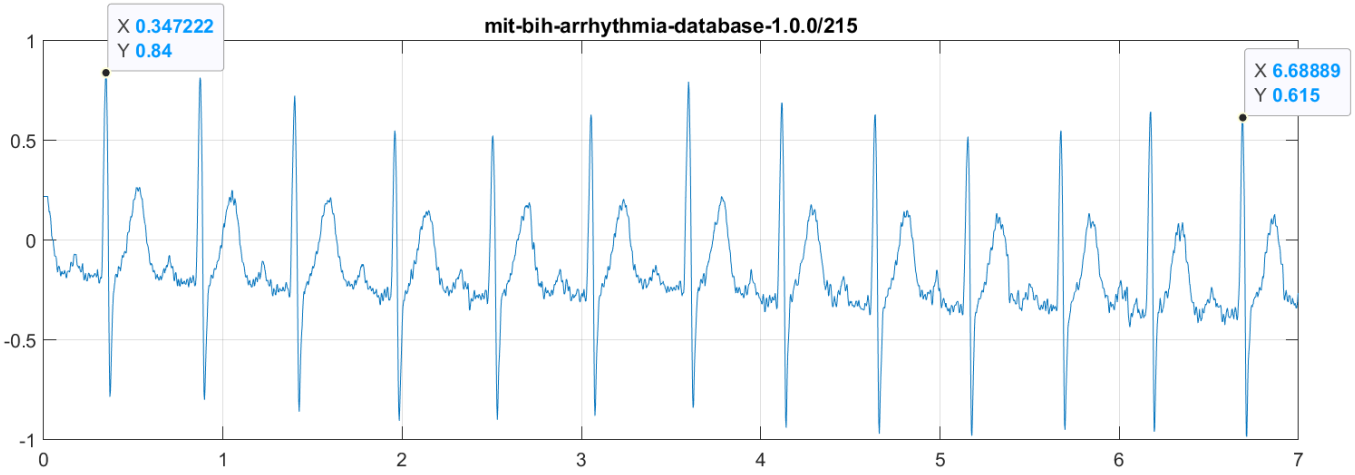


Figura 5.5: *ECG número 215.*

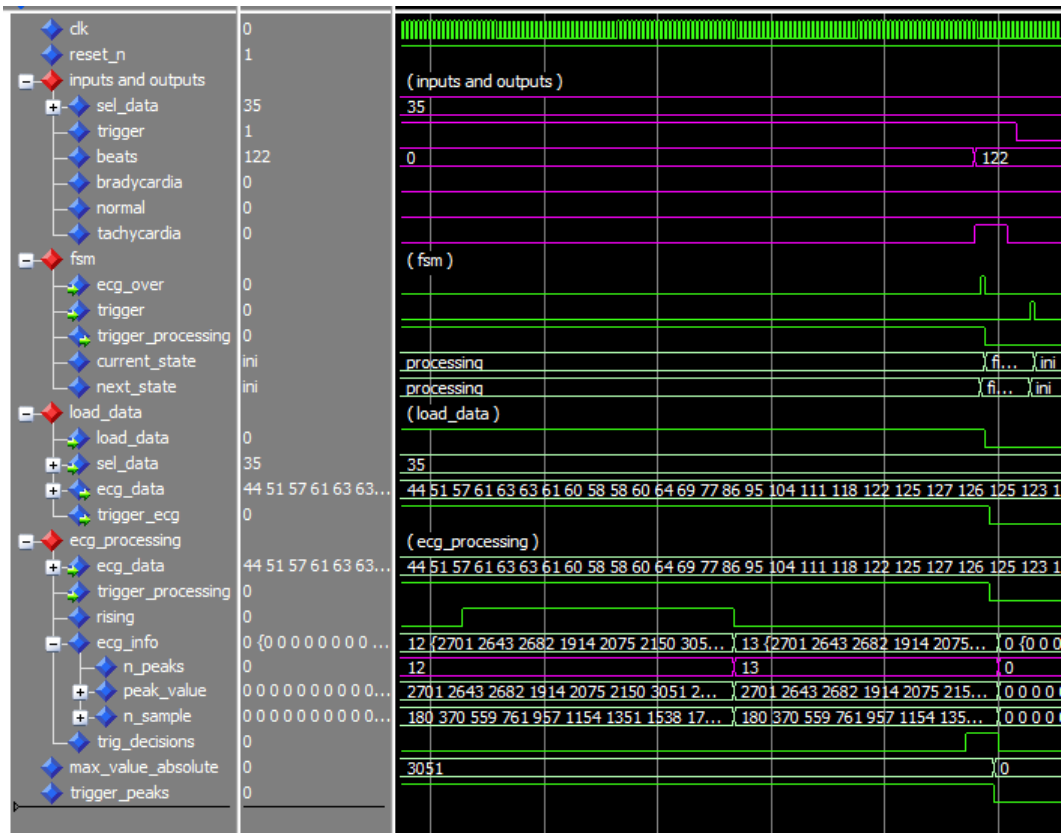


Figura 5.6: *Simulación del ECG número 215.*

Se puede observar que este ECG cuenta también con 13 complejos QRS, por tanto el algoritmo los está detectando correctamente. Para calcular las pulsaciones por minuto:

$$13 * \left(\frac{60}{6{,}68 - 0{,}34} \right) \approx 122 \; pulsaciones \; por \; minuto$$

El sistema habilita la salida de *tachycardia* porque las pulsaciones son superiores a 100 latidos por minuto. Seguidamente se va a probar la muestra ECG número 123 declarada por el algoritmo como bradicardia:

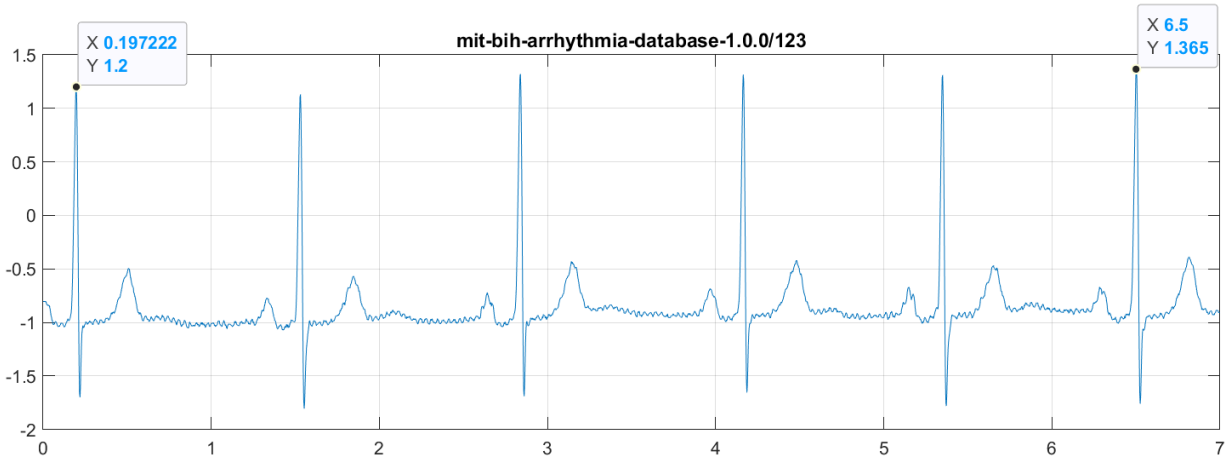


Figura 5.7: *ECG número 123.*

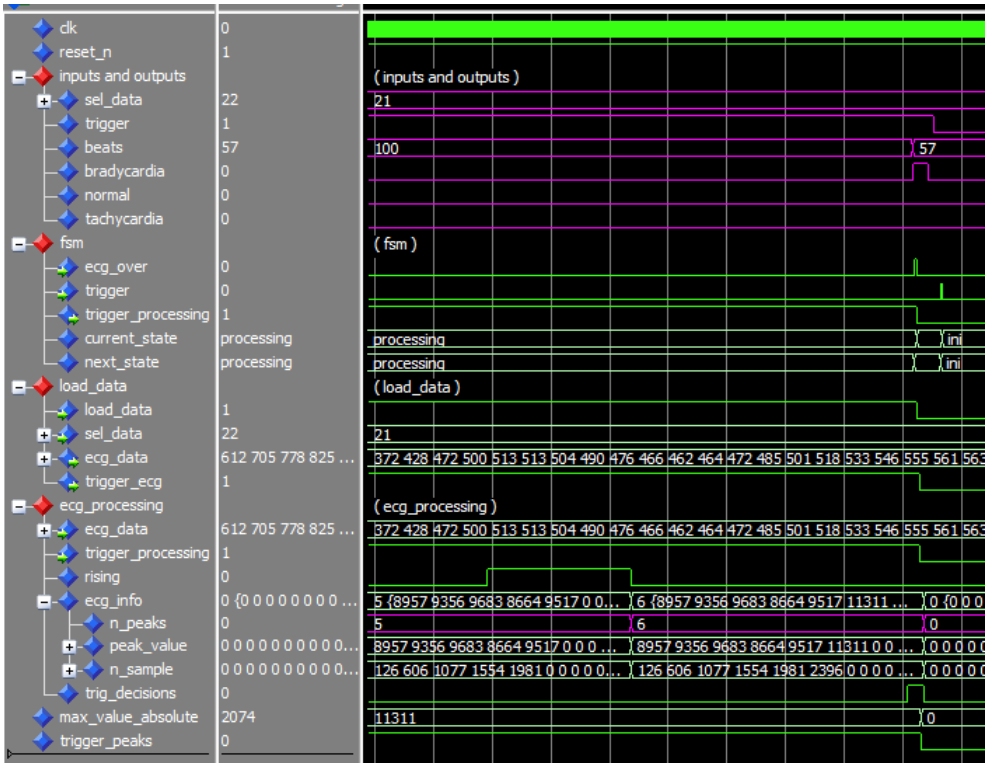


Figura 5.8: *Simulación del ECG número 123.*

Se puede observar que este ECG cuenta con seis complejos QRS, por tanto el algoritmo los está detectando correctamente. Para calcular las pulsaciones por minuto:

$$6 * \left(\frac{60}{6{,}50 - 0{,}19} \right) \approx 57 \; pulsaciones \; por \; minuto$$

El sistema habilita la salida de *bradycardia* porque las pulsaciones son inferiores a 60 latidos por minuto.

5.2. Resultados

ECG	QRS detectados	QRS reales	Pulsaciones por minuto
100	9	9	83
101	8	8	76
102	8	9	83
103	8	8	78
104	9	9	83
105	9	9	93
106	6	6	69
107	9	9	79
108	9	8	82
109	11	11	102
111	8	8	80
112	10	10	95
113	6	6	67
114	13	6	116
115	7	7	69
123	6	6	57
213	13	13	120
215	13	13	122

Cuadro 5.1: *Resultados detección QRS.*

Observando la tabla, los ECGs en los que los complejos QRS no han sido detectados correctamente son: 102, 108, 114. Se va a realizar un análisis de cada caso para detectar el motivo del fallo y si es posible solucionarlo con los parámetros de configuración:

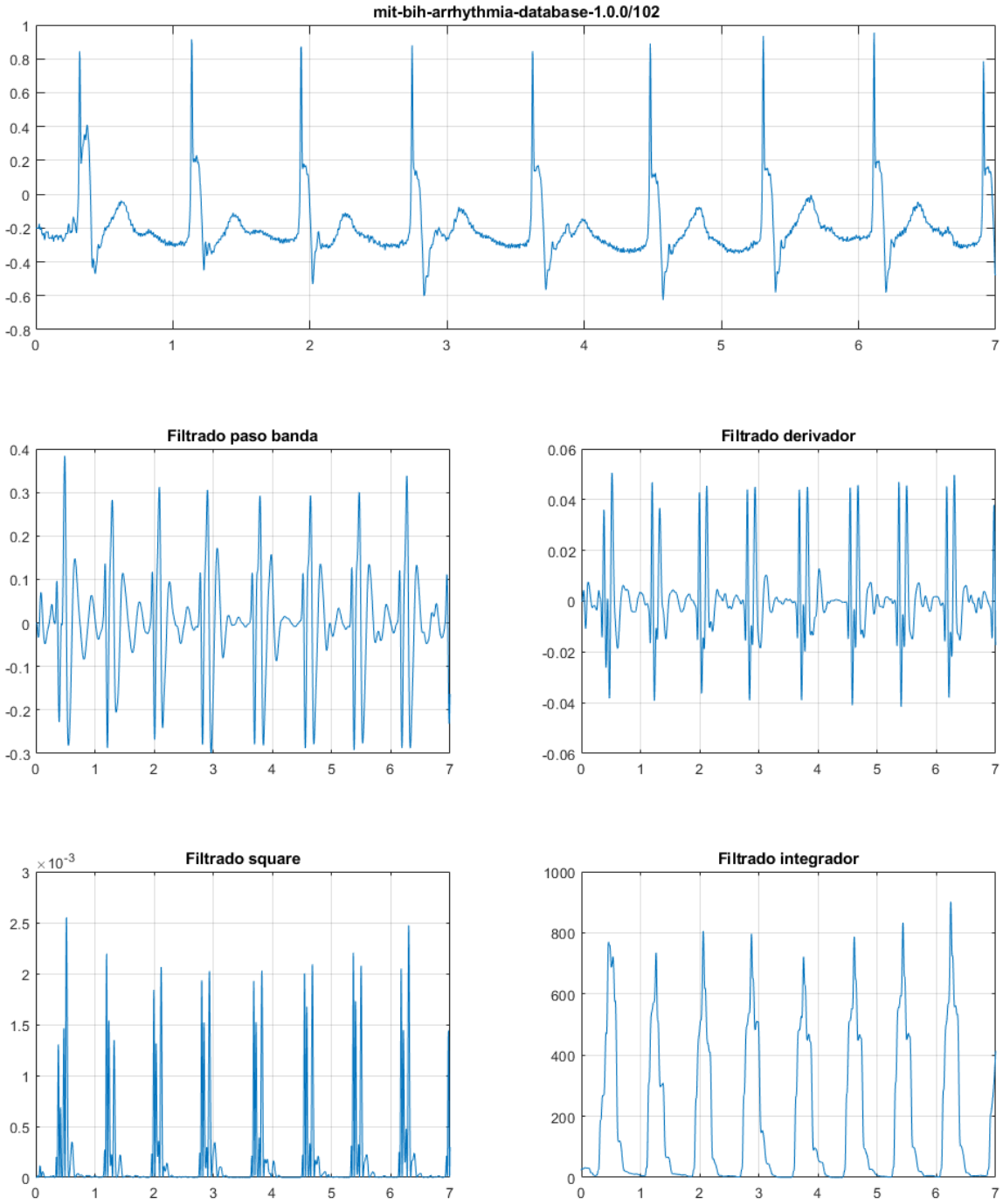


Figura 5.9: *ECG número 102.*

Se puede observar que en el ECG original hay nueve complejos QRS, sin embargo el algoritmo solo está detectando ocho. Si se observa la señal tras el filtro del integrador, el último

pico está recortado, ya que se realiza la media de 54 muestras (una ventana de 150 ms), este caso es correcto y para solucionarlo es suficiente con realizar un procesamiento de más tiempo y así el filtro integrador tendrá tiempo suficiente para mostrar el último pico.

Seguidamente se analizan los ECGs número 108 y 114:

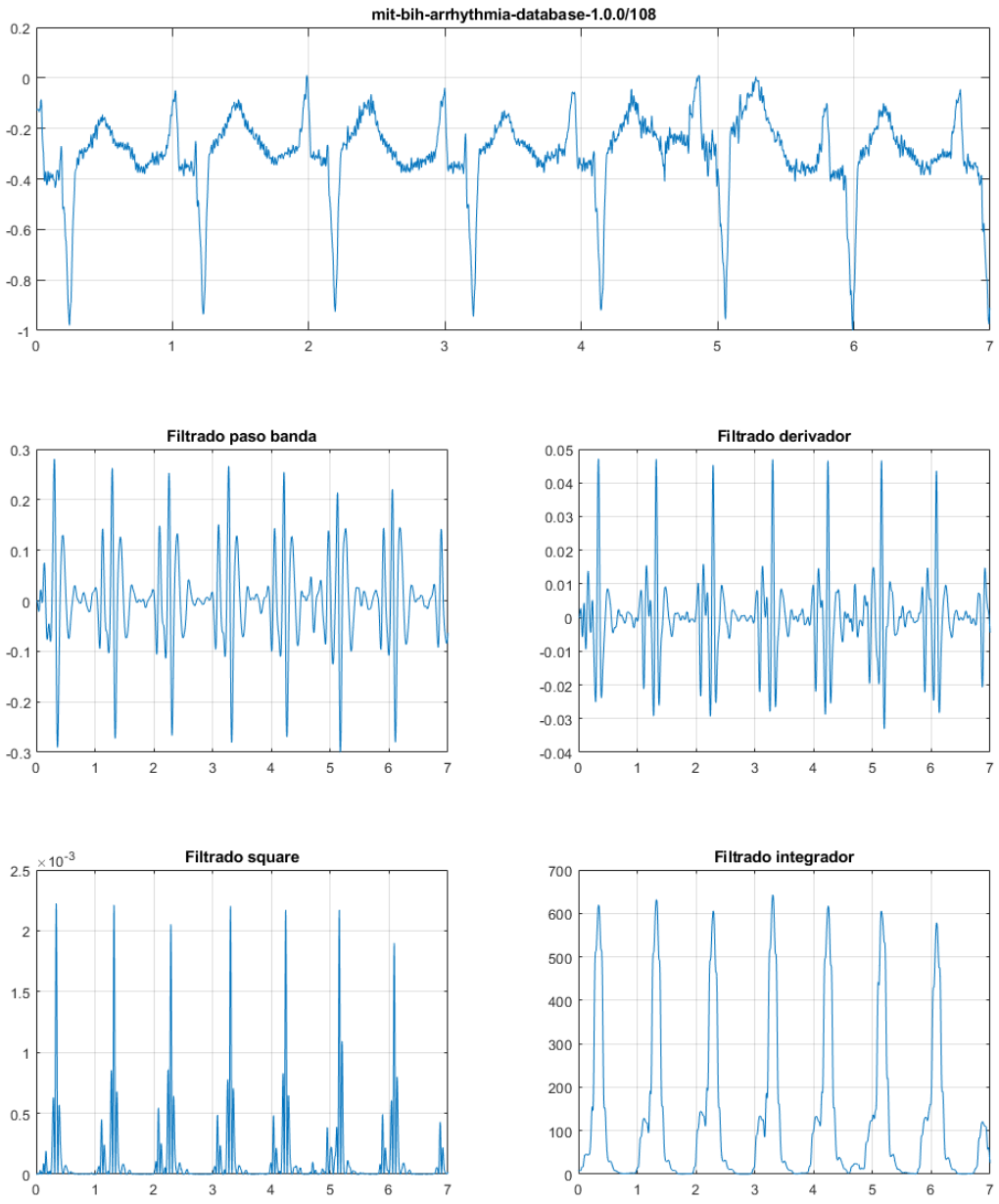


Figura 5.10: *ECG número 108.*

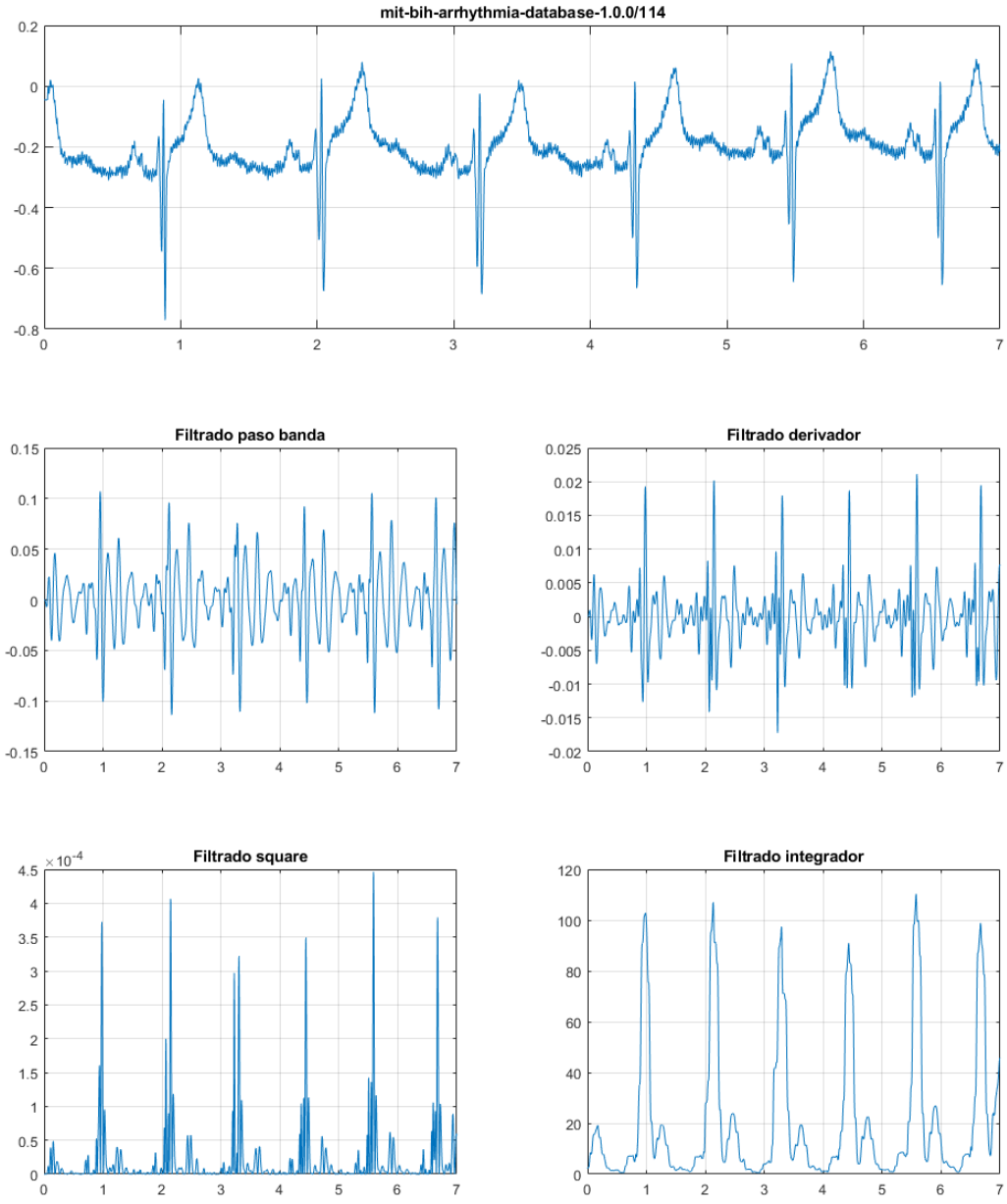


Figura 5.11: *ECG número 114.*

Si se observan las figuras 5.7 y 5.8 está sucediendo lo mismo. Cuando se realiza el filtrado y se observa el resultado final del integrador, se observan pequeños picos, que provocan que el algoritmo los detecte como complejos QRS. Este problema se puede solucionar gracias a uno de los parámetros configurables implementados en el algoritmo. Con aumentar el valor a partir el cuál una muestra es descartada, se van a ignorar esos pequeños picos. En este caso está configurado para que se descarten picos 20 veces más pequeños.

Se puede observar en estos dos ECG que los picos máximos tienen valores muy pequeños del orden de 100-600, por tanto, si se configura para descartar picos dos veces más pequeños, es decir, la mitad del valor máximo, el algoritmo detecta correctamente los complejos QRS. Sin embargo, se está ante dos casos de arritmia, en donde la onda T es mayor al complejo QRS.

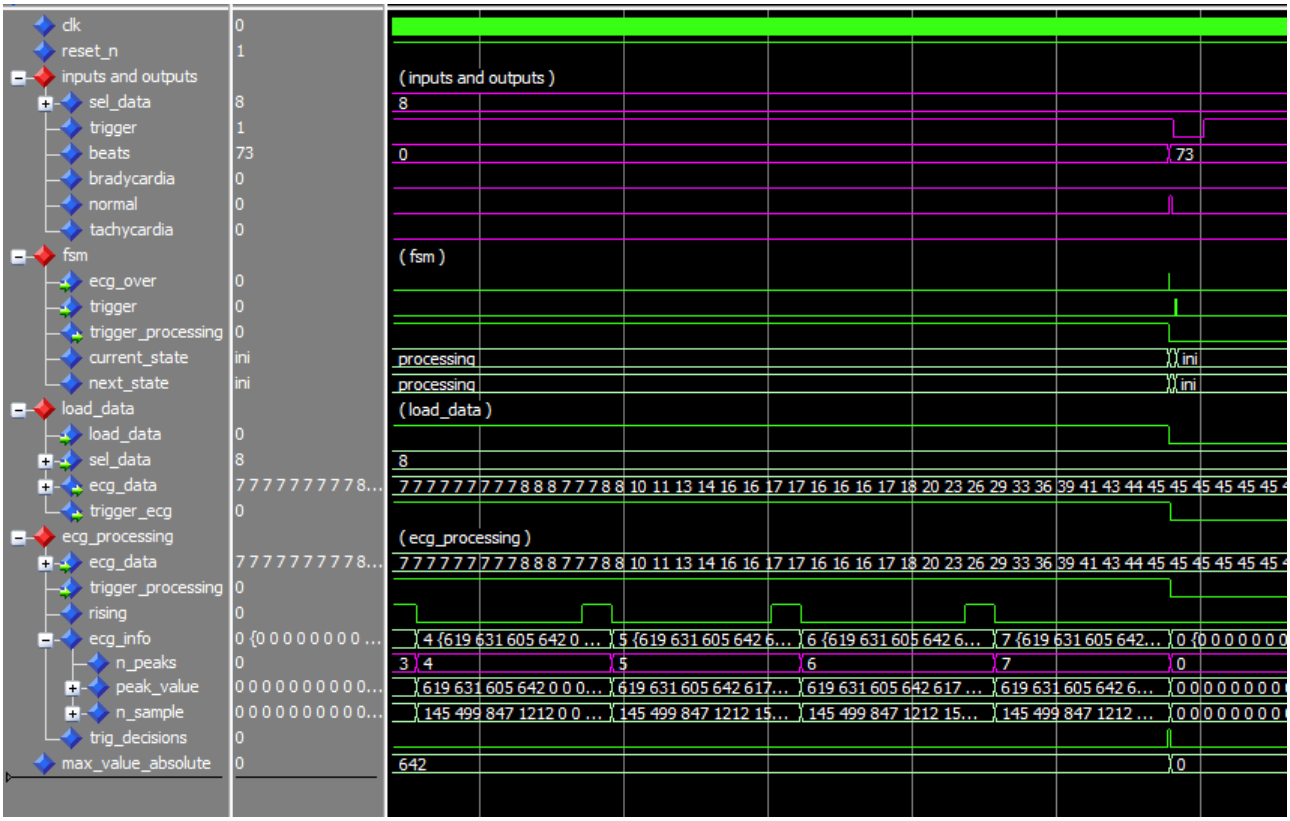


Figura 5.12: *Simulación del ECG número 108.*

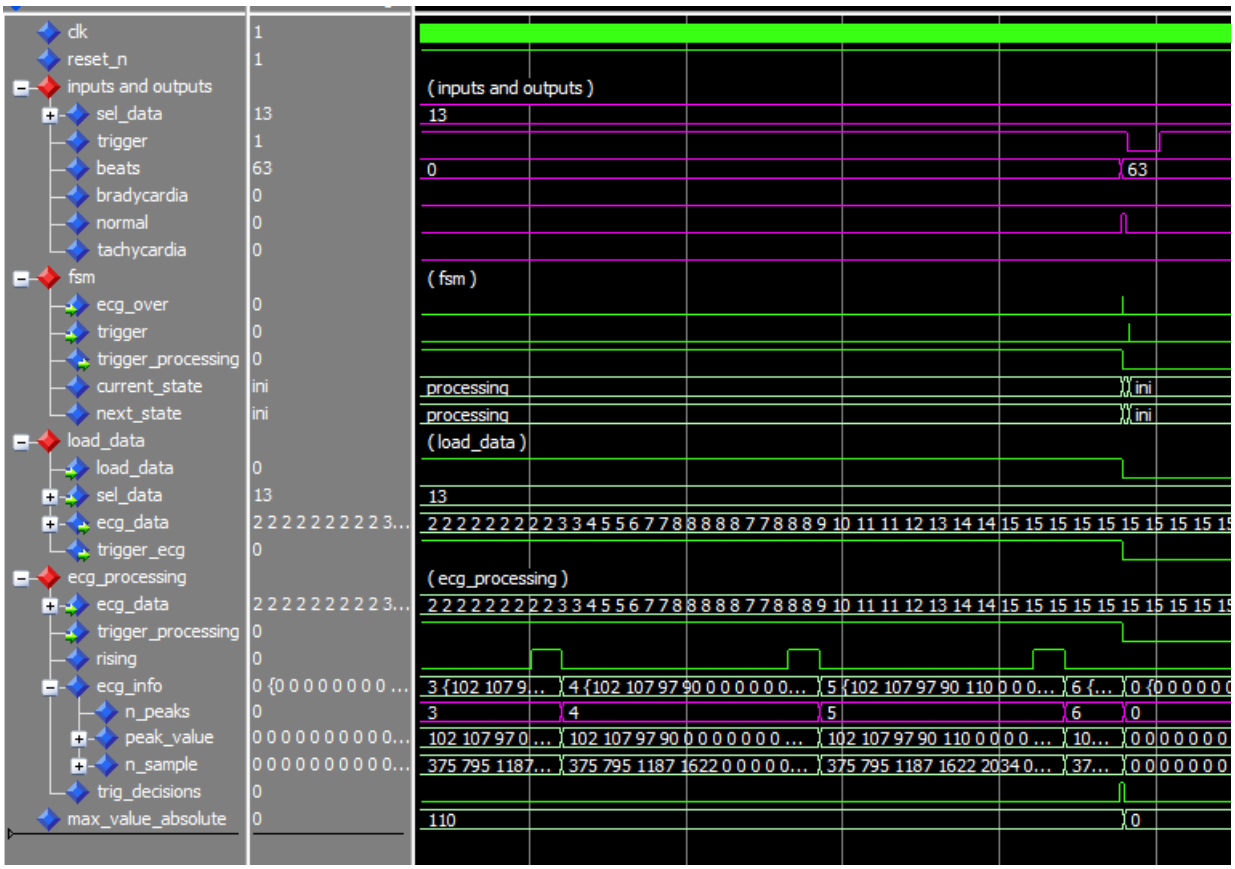


Figura 5.13: *Simulación del ECG número 114.*

La detección de complejos QRS es correcta. Siete complejos para el 108 (el último complejo QRS no es posible detectarlo debido al integrador, por lo que es correcto) y seis para el 114.

5.3. Conclusiones

El mundo de la biomedicina es muy amplio e interesante. Se ayuda en gran medida de diferentes métodos y técnicas de procesado así como también de herramientas y máquinas para detectar trastornos de cualquier tipo en los pacientes. El procesado de señales es una de las técnicas más usadas y útiles en el campo de la biomedicina.

Respecto al algoritmo, éste funciona correctamente en la detección de complejos QRS, calculando los latidos por minuto y con ello la clasificación en bradicardia, taquicardia o ECG normal. Las principales ventajas son: fiabilidad, consistencia y versatilidad, ya que es configurable mediante dos parámetros. Como se observó en los resultados, de 18 señales ECG analizadas el algoritmo detectó correctamente 15, con un porcentaje de acierto del 83.33 %. Sin embargo, tras configurar correctamente los parámetros del algoritmo, la fiabilidad alcanzó el 100 %, detectando correctamente todos los complejos QRS.

Por otro lado, el principal inconveniente o problema se produce en los extremos del procesado. Cuando existe un complejo QRS en uno de ellos, el algoritmo no es capaz de detectar dicho complejo, debido al procesado y acondicionamiento de la señal tras el filtro integrador. Otro de los problemas encontrados durante el desarrollo fue la detección de complejos QRS. Debido a este problema, se incluyó uno de los dos parámetros de configuración. Dicho parámetro establece el valor mínimo que debe tener una muestra del ECG para no ser descartada.

De cara a futuras lineas de trabajo, se propone lo siguiente: implementar el filtrado en *VHDL*, así como también la detección de arritmias irregulares por el algoritmo. Por último, la posibilidad de auto configuración del propio algoritmo. En otras palabras, que el algoritmo sea capaz de configurarse así mismo y descartar picos automáticamente.

Glosario

ECG Electrocardiograma

FPGA Field Programmable Gate Arrays

HDL Hardware Description Language

VHDL Very High Speed Integrated Circuit Hardware Description Language

VHSIC Very High Speed Integrated Circuit

Referencias

[1] Mariel Alfaro-Ponce, Isaac Chairez y Ralph Etienne-Cummings. *Automatic detection of electrocardiographic arrhythmias by parallel continuous neural networks implemented in FPGA*. Feb. de 2019, págs. 363-375. DOI: 10.1007/s00521-017-3051-3.

[2] Jagadeeswara Rao Annam y Bapi Raju Surampudi. *Inter-patient heart-beat classification using complete ECG beat time series by alignment of R-peaks using SVM and decision rule*. Feb. de 2017. DOI: 10.1109/ICONSIP.2016.7857480.

[3] Dra Tamara Archondo Arce y Julián Pérez-Villacastín. *Qué es una arritmia y cómo funciona un marcapasos*. 2009. URL: https://www.fbbva.es/microsites/salud_cardio/mult/fbbva_libroCorazon_cap45.pdf.

[4] Luis Azcona. *Estructura del corazón Capítulo 4 El electrocardiograma*. 2009. URL: https://www.fbbva.es/microsites/salud_cardio/mult/fbbva_libroCorazon_cap4.pdf.

[5] Deepak Berwal, Ashish Kumar y Yogendera Kumar. *Design of high performance QRS complex detector for wearable healthcare devices using biorthogonal spline wavelet transform*. Oct. de 2018, págs. 222-230. DOI: 10.1016/J.ISATRA.2018.08.002.

[6] José Rogan C. *Capítulo 12. Polinomios de Hermite*. URL: https://macul.ciencias.uchile.cl/~jrogan/cursos/mfm2p00/cap12.pdf.

[7] Mayo Clinic. *Arritmia cardíaca - Diagnóstico y tratamiento - Mayo Clinic*. URL: https://www.mayoclinic.org/es-es/diseases-conditions/heart-arrhythmia/diagnosis-treatment/drc-20350674.

[8] Matej Cvikl, Franc Jager y Andrej Zemva. *Hardware Implementation of a Modified Delay-Coordinate Mapping-Based QRS Complex Detection Algorithm*. 2007, pág. 13. DOI: 10.1155/2007/57286.

[9] Madhav P. Desai y col. *A low-latency, low-power FPGA implementation of ECG signal characterization using hermite polynomials*. Oct. de 2021. DOI: 10.3390/electronics10192324.

[10] Instituto Nacional de Estadística. *Mortalidad cardiovascular en España en 2018 - Sociedad Española de Cardiología*. 2019. URL: https://secardiologia.es/publicaciones/infografias/11527-mortalidad-cardiovascular-en-espana-en-2018.

[11] Alberto Gil y col. *Hermite Polynomial Characterization of Heartbeats with Graphics Processing Units*. 2014. URL: http://iwbbio.ugr.es/2014/papers/IWBBIO_2014_paper_60.pdf.

[12] Jeong Whan Lee y col. *A real time QRS detection using delay-coordinate mapping for the microcontroller implementation*. 2002, págs. 1140-1151. DOI: 10.1114/1.1523030.

[13] Karim Meddah y col. *FPGA-based system for heart rate monitoring*. Sep. de 2019, págs. 771-782. DOI: 10.1049/iet-cds.2018.5204.

[14] Karim Meddah y col. *FPGA implementation system for QRS complex detection*. Feb. de 2020. DOI: 10.4015/S1016237220500052.

[15] Diego Peluffo, Jose Luis Rodriguez-Sotelo y German Castellanos-Dominguez. *Metodología para la reconstrucción y extracción de caracaterísticas del complejo QRS basada en el modelo parmétrico de Hermite*. 2008. URL: https://www.researchgate.net/publication/303721985.

[16] Willis J Tompkins. *A Real-Time QRS Detection Algorithm*. 1985.

[17] Yande Xiang, Zhitao Lin y Jianyi Meng. *Automatic QRS complex detection using two-level convolutional neural network*. 2018. DOI: 10.1186/s12938-018-0441-4. URL: https://doi.org/10.1186/s12938-018-0441-4.