electronics

MDPI

Article

A Low-Latency, Low-Power FPGA Implementation of ECG Signal Characterization Using Hermite Polynomials

Madhav P. Desai [1], Gabriel Caffarena [2,*], Ruzica Jevtic [2], David G. Márquez [2] and Abraham Otero [2]

[1] Indian Institute of Technology (Bombay), Mumbai 400076, India; madhav@ee.iitb.ac.in
[2] Department of Information Technologies, University CEU San Pablo, 28003 Madrid, Spain; ruzica.jevtic@ceu.es (R.J.); david.gonzalez.marquez@gmail.com (D.G.M.); aotero@ceu.es (A.O.)
* Correspondence: gabriel.caffarena@ceu.es

Abstract: Automatic ECG signal characterization is of critical importance in patient monitoring and diagnosis. This process is computationally intensive, and low-power, online (real-time) solutions to this problem are of great interest. In this paper, we present a novel, dedicated hardware implementation of the ECG signal processing chain based on Hermite functions, aiming for real-time processing. Starting from 12-bit ADC samples of the ECG signal, the hardware implements filtering, peak and QRS detection, and least-squares Hermite polynomial fit on heartbeats. This hardware module can be used to compress ECG data or to perform beat classification. The hardware implementation has been validated on a Field Programmable Gate Array (FPGA). The implementation is generated using an algorithm-to-hardware compiler tool-chain and the resulting hardware is characterized using a low-cost off-the-shelf FPGA card. The single-beat best-fit computation latency when using six Hermite basis polynomials is under 1 s with a throughput of 3 beats/s and with an average power dissipation around 28 mW, demonstrating true real-time applicability.

Keywords: ECG; real-time; FPGA; HLS; AHIR; Hermite; low power

Citation: Desai, M.P.; Caffarena, G.; Jevtic, R.; Márquez, D.G.; Otero, A. A Low-Latency, Low-Power FPGA Implementation of ECG Signal Characterization Using Hermite Polynomials. *Electronics* **2021**, *10*, 2324. https://doi.org/10.3390/electronics10192324

Academic Editor: Stefano Ricci

Received: 27 July 2021
Accepted: 16 September 2021
Published: 22 September 2021

1. Introduction

Cardiovascular disease is the number one cause of death worldwide [1]. An electrocardiogram (ECG) registers the electrical activity of a heart, and it stands as a valuable diagnostic tool. However, in clinical routines, ECG analysis is performed as a visual inspection by a cardiologist, which is a tedious task, further aggravated in the case of long-term ECG. For instance, 24 h of Holter recordings contains around 100,000 heartbeats.

Figure 1 depicts the main components of the ECG, with the most important for diagnosis being the waves P, Q, R, S and T. The Q, R and S waves are normally studied together as the QRS complex. The P wave represents the moment when the auricles contract to send blood to the ventricles, and at the end of the PR segment, the ventricle is full. During the QRS complex, the ventricle expels their contents and are fully emptied at the end of the ST segment. The T wave indicates that the heart is at rest.

Developing efficient techniques to automate ECG analysis is instrumental in helping a cardiologist with their diagnosis. The detection of arrhythmias is of special interest [2]. The QRS complexes of heartbeats can be successfully used to identify most arrhythmia types [3–5]. The T wave does not contribute to the identification process [6] and the P wave, even though it provides relevant information about arrhythmias, possesses a low signal-to-noise ratio (SNR), so it is not reliable [7,8].

ECG analysis starts with the detection and characterization of the beats [9]. The detection of the QRS complex is carried out with a high accuracy; a 99.7% detection accuracy was reported in [10]. As for the characterization of the beat, among the different methods [6,11,12], the use of a function space based on Hermite polynomials has many advantages [3,10,13]: dimensionality reduction, low noise sensitivity, etc. The ECG samples are fitted with a linear combination of basis functions, and the coefficients of this linear

combination are used as features for representing heartbeats. As an example of the resulting dimension reduction, the 144-sample QRS complex obtained at a rate of 360 sps can be reasonably characterized with 6 or 7 parameters [14]. Regarding the average classification error, values as small as 0.34% are reported in [15], thus supporting the development of new classifiers based on Hermite functions as well as hardware implementations able to provide high-quality real-time heartbeat analysis.

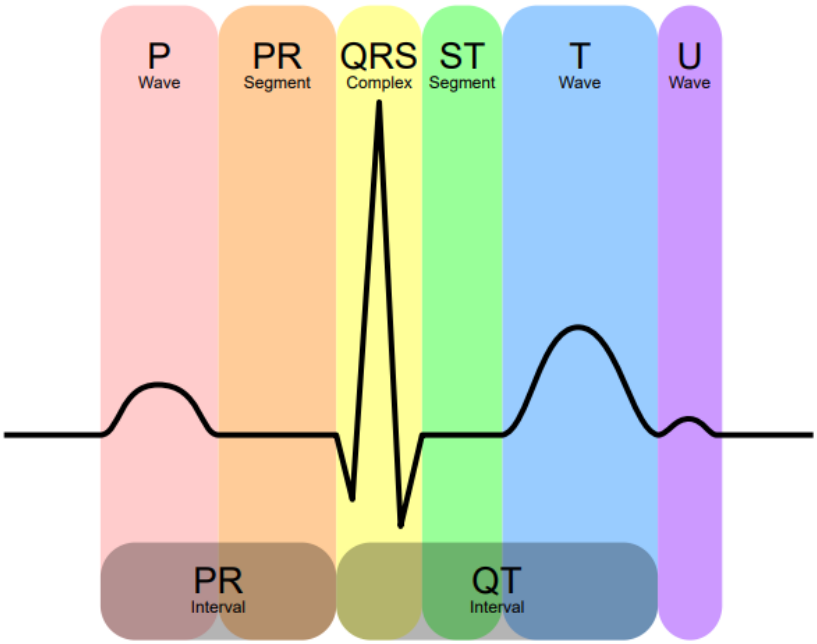


Figure 1. The main components of an electrocardiogram (author: Hank van Helvete; derivative: Rehua; source: https://commons.wikimedia.org/wiki/File:EKG_Complex_en.svg; accessed date: 1 September 2021).

One disadvantage of the Hermite representation is that it is computationally demanding. There are some approaches addressing this problem. In [16], graphics processing units (GPU) are used to accelerate the offline processing of Hermite fitting of heartbeats. The use of Field-Programmable Gate Array (FPGA) devices is supported in [17]; in this paper, the results of an FPGA-based implementation aiming at wearable systems are presented. Reconfigurable devices (i.e., FPGA) allow for developing a custom architecture that can be adjusted to the different levels of computation performance and energy efficiency. Moreover, they can be used to prototype a system before being implemented as an application-specific integrated circuit (i.e., ASIC), which can achieve even better computation and electrical consumption performance. The developing times required for both FPGA and ASIC is quite long and complex in comparison with the traditional software approach (i.e., microprocessor-based or GPU-based), and high-level synthesis (HLS) tools have thrived in the last few years [18,19]. In this work, the HLS tool AHIR [20–22] has been used. AHIR is an open-source alternative to proprietary products that allows us to generate RTL descriptions from C language with reduced development times.

The central contribution of this paper is the design and implementation of a novel hardware module able to characterize heartbeats in real time by means of Hermite functions. This module can be used as the input to systems to compress the ECG data as well as to classifiers. Despite the interest in producing hardware systems for real-time processing of ECG signals [23–26], to the best of our knowledge, this is the first time that Hermite function fitting with a complete preprocessing chain is implemented in hardware for ECG processing. The main contributions of this paper are as follows:

- Novel hardware implementation of full processing chain for real-time ECG characterization based on Hermite functions;

- Introduction to the AHIR HLS tool;
- Implementation of the system in a low-cost FPGA-based board; and
- On-board power consumption measurements.

The paper is organized as follows: Section 2 elaborates on the Hermite fitting of heartbeats; in Section 3, the AHIR tool is presented; Section 4 describes the system implemented on an FPGA device; the implementation results are in Section 5, and they are analysed in Section 6; and, finally, the conclusions are drawn in Section 7.

2. Estimation of the QRS Complex with Hermite Polynomials

As mentioned in Section 1, QRS complexes are employed for arrhythmia detection and the use of Hermite functions allows us to reduce the number of dimensions involved in the ECG classification, without sacrificing accuracy [3,10] as well as enabling the transmission of ECG compressed data [27]. Moreover, Hermite fitting representations are robust in the presence of noise.

The MIT-BIH arrhythmia database [28] is used as a benchmark in this work. It contains 48 2-channel ECG recordings, sampled at a frequency of 360 Hz and with a duration of approximately 2000 beats (half an hour). Each beat has been manually annotated by at least two cardiologists, so it can be used to check the outcome of ECG automatic classification. The database includes an extended set of arrhythmias, and it has been extensively used in automatic arrhythmia classification [4,10,29].

Prior to QRS characterization, the ECG signal must be processed to remove the baseline drift and high-frequency noise [30]. The QRS complexes have a length of 70–100 ms; therefore, extracting a window of 200 ms around the peak (i.e., R wave) of the beat ensures that we acquire the complete complex while leaving the T and P waves out. The QRS window is expanded up to 400 ms by means of zero padding the extremes of the 200-ms window since the Hermite functions converge to zero in $\pm\infty$. Thus, the QRS beat data used as an input to the Hermite polynomial approximation consists of a 144-sample vector $\vec{x} = \{x(t)\}$ that can be estimated with a linear combination of N Hermite basis functions ϕ_n by means of coefficients c_n (Equation (1)). In this work, we use $N = 6$, which provides a good compromise between having a compact representation and having a good accuracy in the representation of the beat [14].

The aim of the Hermite fitting is to find the approximation to the QRS complex $\{x(t)\}$ with the best minimum-mean-square-error (MMSE). The approximation of $x(t)$ is expressed as

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

with

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

where $H_n(t/\sigma)$ is the n^{th} Hermite polynomial. The Hermite polynomials can be computed recursively as

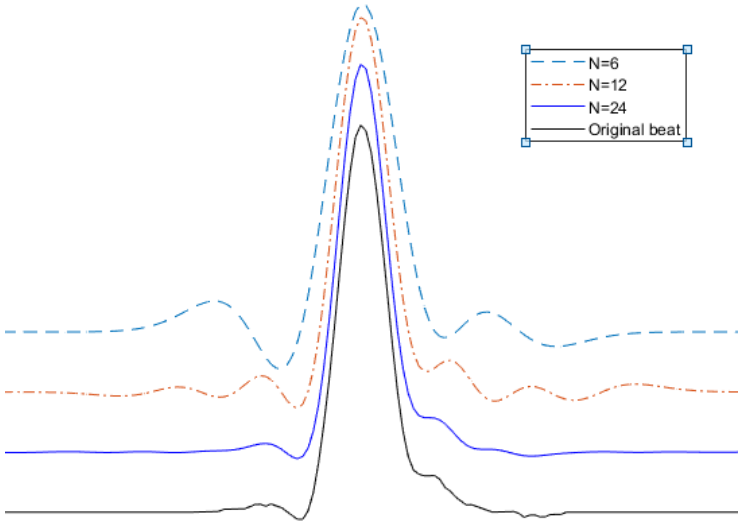
$$H_0(x) = 1 \tag{3}$$
$$H_1(x) = 2x \tag{4}$$
$$H_n(x) = 2x \cdot H_{n-1}(x) - 2(n-1)H_{n-2}(x) \tag{5}$$

The parameter σ is a time-scaling factor in the polynomials that adjusts the width of the Hermite functions to the one of the actual QRS complexes. The maximum value of σ for a given order n is studied in [3].

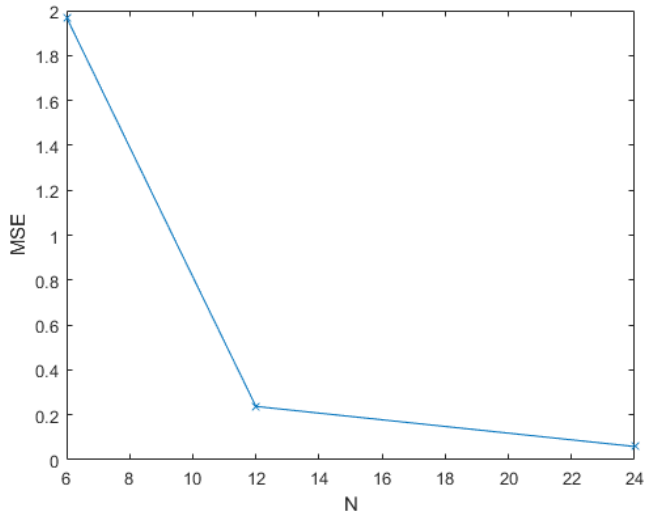
Give σ, the orthogonality of the Hermite basis function allows us to find the optimal coefficient—those that minimize the square error—as

$$c_n(\sigma) = \sum_t x(t) \cdot \phi_n(t, \sigma) \tag{6}$$

In order to find the best fit, the MMSE approximation for each σ is obtained, and the one with the smallest value is kept. As a result, each heartbeat is represented by a set composed of the best σ and the corresponding fit coefficients $\vec{c} = \{c_n(\sigma)\}$ $(n \in [0, N-1])$ and it is possible to use only these parameters to perform the morphological classification of the heartbeats [3,29]. Figure 2 depicts the effect of increasing the number of Hermite function in the beat estimation. Figure 2a shows the original beat (in black) and the estimation with $N \in \{6, 12, 24\}$. It can be seen that, as long as the value of N is increased, the estimation captures the variations of the heartbeat in more detail. Figure 2b shows the decreasing trend of the minimum square error (MSE) for each estimation.



(**a**)



(**b**)

Figure 2. Effect of polynomial order in the quality of the estimation of one heartbeat ($N \in \{6, 12, 24\}$): (**a**) original heartbeat and different estimations; (**b**) MSE.

3. From Algorithm-to-Hardware Using AHIRV2, a C-2-VHDL Compiler

The AHIRV2 compiler tool-chain [20–22] provides a pathway from a C-program to actual synthesizable hardware. The tool-chain takes a description of an algorithm (described in C) and produces a VHDL logic circuit description that is equivalent to the algorithm.

The AHIRV2 compiler starts with a C program and produces VHDL. The clang 2.8 compiler (www.clang.org; accessed on 1 September 2021) acts as the C front-end and is used to emit LLVM byte-code (www.llvm.org), which is then converted to VHDL using the following transformations:

1. The LLVM byte-code is translated into an internal intermediate format, which is itself a static-single assignment centric control-flow language (named Aa), which allows for the description of parallelism using fork-joined structures as well as arbitrary branching;
2. The Aa description is translated to a virtual circuit (the model is described in the next subsection). During this translation, the following major optimizations are performed: declared storage objects are partitioned into disjoint memory spaces using pointer reference analysis, and dependency analysis is used to generate appropriate sequencing of operations in order to maximize the parallelism. Inner loops in the Aa code are pipelined so that multiple iterations of a loop can be executed concurrently;
3. The virtual circuit is then translated to VHDL. At this point, decisions about operator sharing are taken. Concurrency analysis is used to determine if a shared hardware unit needs arbitration. Optimizations related to clock-frequency maximization are also carried out here. The generated VHDL uses a pre-designed library of useful operators ranging from multiplexors and arbiters to pipelined floating point arithmetic units (arbitrary precision arithmetic is supported, and in particular, there is full support for IEEE-754 single precision and double precision add/multiply with all rounding modes).

3.1. An Illustration of the Virtual Circuit Generated by the AHIRV2 Compiler

The virtual circuit generated by the AHIRV2 compiler consists of three cooperating components: the control path, the data path and the storage system [21,22].

To illustrate the model, we consider a simple example.

```
float a[1024], b[1024];
float dotp = 0.0;
for(i=0; i < 1024; i++)
{
dotp += a[i]*b[i];
}
```

The AHIRV2 tool-chain transforms this program to produce a virtual circuit, which is depicted in Figure 3. The virtual circuit is then translated to synthesizable VHDL. The virtual circuit consists of three components independent parts, namely the data path, the storage subsystem and the control path.

3.1.1. The Data Path

The data path is a directed hyper-graph with nodes being operations and arcs being nets (shown as ovals). Each net has at most one operation that drives it. Furthermore, most operations have a split protocol handshake with the control path: two pairs of request/acknowledge associations ($*sr/*sa$ for sampling the inputs and $*cr/*ca$ for updating the outputs). The operation samples its inputs upon receiving the *sr* request symbol and acknowledges the completion of this action by emitting the *sa* acknowledge symbol. After receiving the *cr* symbol, the operation updates its output net using the newly computed value. The sequencing is as follows:

```
sr -> sa -> cr -> ca
```

Note that an operation can be re-triggered while an earlier edition of the operation is in progress (this is important if the operation is implemented using a pipelined operator).

Some data-path operations (such as the multiplexor shown on the top and the decision operation shown at the bottom left in Figure 3) follow a simpler protocol. The multiplexor has a pair of requests and a single acknowledge, with the condition that at most one of the requests can be received at any time instant. The input corresponding to the request is then sampled and stored in the output net of the multiplexor. The decision operation has a single request and two acknowledges. Upon receipt of the request symbol, the decision operation checks its input net and emits one of the two acknowledges depending on whether the input is zero or nonzero.

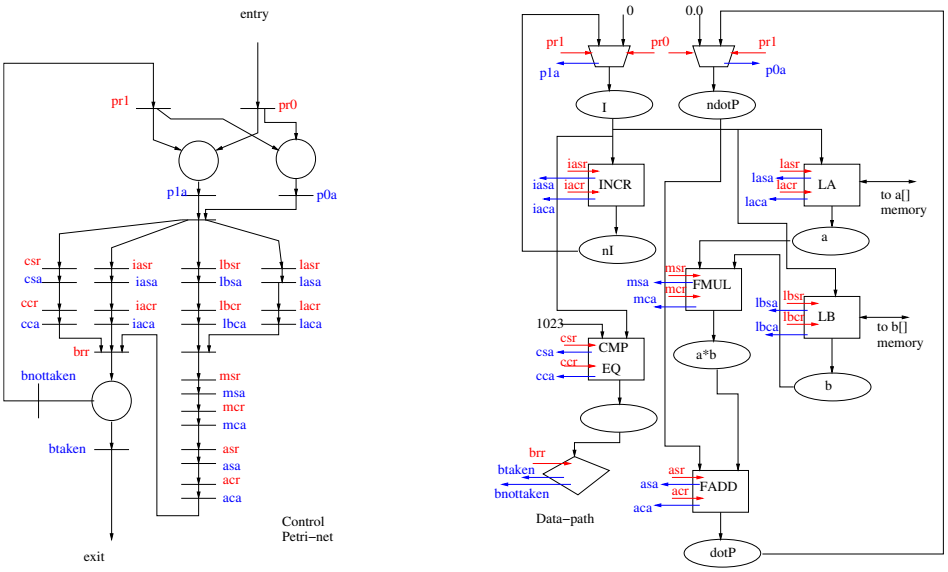


Figure 3. Control data storage virtual circuit model.

In Figure 3, the following data-path operations are instantiated:

```
mI, mdotP  multiplexors for I, dotP.
INCR       increment for I++
LA         load for a[I]
LB         load for b[I]
FMUL       multiply for p=a[I]*b[I]
FADD       add for dotP += a*b
CMP EQ     compare for COND=(I==1023)
D          decision  COND?
```

Note that the data path only shows the operations and their interconnection. When the data path is implemented as hardware, multiple operations may be mapped to a single operator depending on cost/performance trade-offs. In this case, multiplexing logic is introduced in the hardware. These decisions and manipulations are performed in the compiler stage, which is responsible for transforming the virtual circuit to VHDL.

3.1.2. Storage Subsystem

The load and store operations in the data path are associated with memory subsystems. In general, there can be multiple disjoint memory subsystems inferred by the compiler. In this particular case, the arrays $a[]$ and $b[]$ are mapped to disjoint memories, due to which

the two loads are allowed to proceed in parallel (the relaxed consistency model is enforced). In order to maintain the relaxed consistency model, the memory subsystems are designed to use a time-stamping scheme, which guarantees first-come-first-served access to the same memory location.

3.1.3. Control Path

The control path in the virtual circuit encodes all of the sequencing that is necessary for correct operation of the assembly. The control path (shown on the left in Figure 3) is modelled as a Petri-net with a unique entry point and a unique exit point. The Petri-net is constructed using a set of production rules, which guarantee liveness and safeness [21]. Transitions in the Petri-net are associated with output symbols to the data-path (these can be described by the regular expressions $*sr$ and $*cr$) and input symbols from the data path (these are of the form $*sa$ and $*ca$). The $*sr$ symbols instruct an element in the data path to sample its inputs and the $*cr$ symbols instruct an element in the data path to update its outputs (all outputs of data path elements are registered). The $*sa$ and $*ca$ symbols are acknowledgements from the data path, which indicate that the corresponding requests have been served.

The following classes of dependencies are encoded in the control Petri-net:

- Read-after-write (RAW): If the result of operator A is used as an input to operator B, the sr symbol to B can be emitted only after the ca symbol from A has been received;
- Write-after-read (WAR): If B writes to a net in which the value needs to have been used by A earlier, for example, as in

```
a = (b+c) -- operation A reads c
c = (p*q) -- operation B writes to c
```

  where there is a WAR dependency through c, then the cr request to B can be issued only after the sa acknowledge from A has been received;
- Load–Store ordering: If P, Q are load/store operations to the same memory subsystem, and if at least one of P, Q is a store, and if P is supposed to happen before Q, then the sr request to Q must be emitted only after the sa acknowledge from Q has been received. The memory subsystem itself guarantees that requests finish in the same order that they were initiated. This takes care of WAR, RAW and WAW memory dependencies.

The control path in Figure 3 shows the sequencing generated by these rules. When pipelining an inner loop, the execution of an operation in a particular iteration is enabled as soon as its dependencies on results from previous iterations are satisfied.

4. Implementation of the System

The analysis of an ECG signal received from a sensor goes through the following steps:

1. Initial signal filtering to remove noise and drift;
2. ECG beat recognition and identification of the QRS complex;
3. ECG beat feature extraction: this can be performed in various ways. We look at the use of Hermite polynomials for the same;
4. ECG classification: based on the beat features, classify the beat as normal or anomalous. This last step is not part of the current work.

We have implemented a signal chain that integrates the first three steps in the list above. Our main contribution is that we have built a custom hardware implementation of the entire signal flow up to Hermite classification, and demonstrated that sophisticated low power, real time ECG analysis is possible in hardware and that high level algorithm to hardware design techniques offer a practical pathway to such realizations.

The incoming ECG signal is assumed to be generated by an 11-bit ADC with a sampling rate of 360 Hz. For all experiments described in this report, we used 11-bit

sampled data from the MIT arrhythmia reference database [28]. The initial signal processing such as the band-pass filter characteristics and the algorithm for QRS detection have been well studied in the literature [30]. The use of Hermite polynomials to extract features from the ECG signal has also been studied extensively [3,10,29].

The entire signal chain is illustrated in Figure 4. In our implementation, the signal chain is divided into two stages. The first stage (the front-end) is responsible for the signal filtering and the QRS peak detection. The second stage takes the identified beats and calculates a best Hermite-polynomial fit for the identified beat. We illustrate this division in Figure 5. All the elements of the signal chain are explained in Sections 4.1 and 4.2. Section 4.3 elaborates on the final system architecture included the signal chain as well as the control block and communications interfaces.

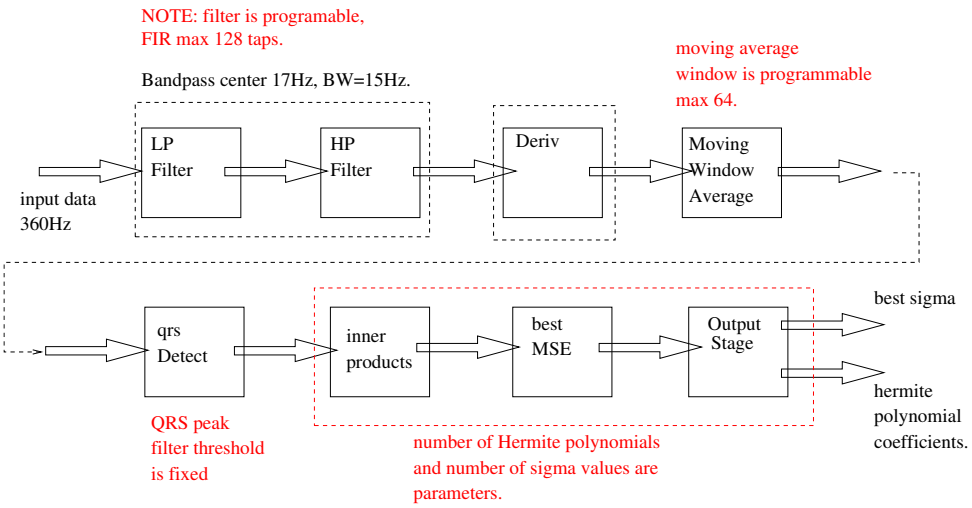


Figure 4. Complete signal chain of the system.

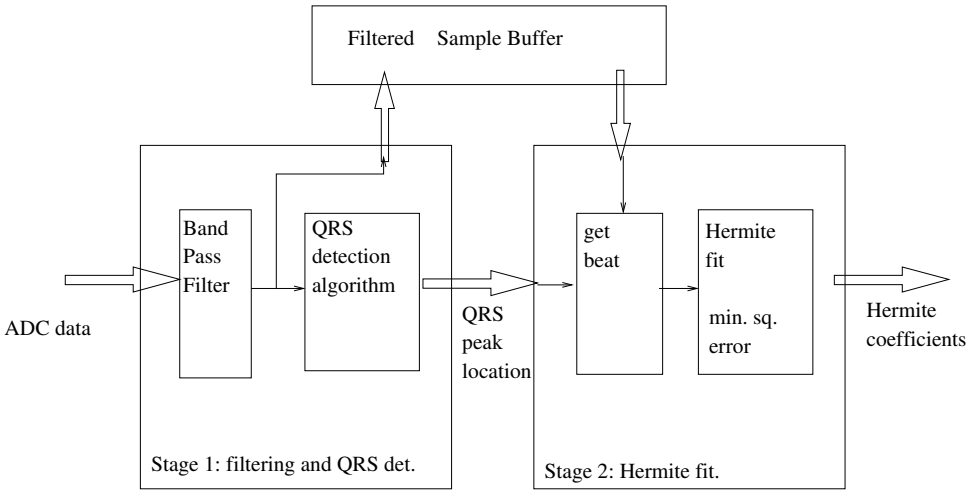


Figure 5. Detail of the two stages composing the system.

4.1. Algorithmic Description of the First Stage

The first stage is responsible for the filtering and QRS peak detection, and the sequence followed is shown in Listing 1.

4.1.1. The Band-Pass Filter

The bandpass filter used is a 99-tap FIR filter with 16-bit taps. The pass-band is set between 6 Hz and 28 Hz. The stop-band attenuation is chosen to be −40 dB. We acknowledge the use of an online filter design tool (http://t-filter.engineerjs.com) [31].

The band pass filter is programmable and can have a maximum of 128 16-bit taps. The implementation of the band-pass filter is shown in Listing 2.

Listing 1. First stage algorithm.

```
void controllerDaemon ()
{
initializer();
uint32_t  sample_index = 0;
while(1)
{
int32_t sample =  getAdcSample();
int32_t filtered_sample =
applyBandPassFilter(sample);
// filtered results pushed into buffer (for use by Hermite
//   fitter)
pushIntoFilteredResultBuffer(sample_index, filtered_sample);

int32_t derivative_sample =
applyDerivativeFilter(filtered_sample);
int32_t moving_average_sample =
applyMovingAverageFilter(derivative_sample);

int32_t qrs_peak =
applyQrsDetector(sample_index, moving_average_sample);

if(qrs_peak >= 0)
{
// correction by subtracting insertion delay.
corrected_qrs_peak = qrs_peak - inserted_qrs_delay;
sendToSecondStage(corrected_qrs_peak);
}
sample_index++;
}
}
```

Listing 2. The band-pass filter.

```
int32_t applyBandPassFilter(int32_t sample)
{
pushSample(wp, band_pass_filter);
int32_t ret_val = dotProduct(band_pass_filter);
return(ret_val);
}
```

4.1.2. The QRS Detection Algorithm

The QRS detection algorithm is implemented in three stages:

1. The band-pass filter outputs are sent through a derivative filter. This acts as a high pass filter that identifies the regions of rapid change (including the QRS complex);
2. The output of the derivative filter is rectified and integrated using a moving average filter with 32 taps. The *strong* peaks of the sequence generated by this moving average filter are expected be in correspondence with the peaks of the QRS complex;

3. The output of the moving average filter is analysed by a threshold crossing state machine that attempts to identify the center peaks of the QRS complex.

The threshold crossing state machine is illustrated in Figure 6.

For the sake of brevity, we do not present the entire C code of the finite state machine. However, a summary of the C code is shown in Listing 3. The algorithm gives the position of the QRS peak, and the heartbeat for further analysis consists of 144 samples centered at this peak.

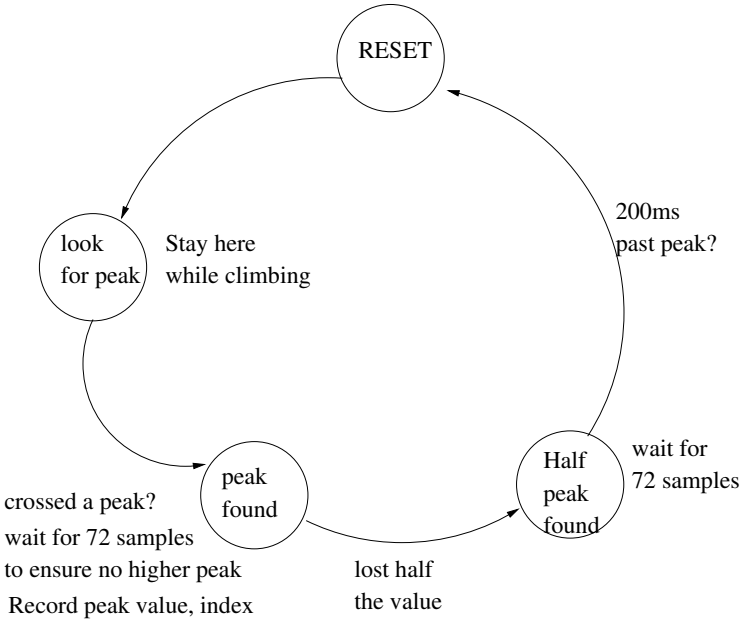


Figure 6. QRS detection finite state machine.

Listing 3. QRS peak detection FSM code outline.

```
// single step of the QRS peak detection FSM.
int32_t applyQrsDetector(uint32_t time_step, int32_t sample)
{
int ret_val = -1;
switch(qrs_state.fsm_state) {
// depending on state,  determine status
// and change state..
case RESET:
....
case LOOKINGFORPEAK:
.....
break;
case PROVISIONALPEAKFOUND:
.....
break;
case HALFPEAKFOUND:
.....
break;
default:
break;
}
return(ret_val);
}
```

4.2. The Second Stage: Calculation of Hermite Polynomial Fits

The first stage in the signal chain provides a QRS peak and a detected heartbeat (post band-pass filtering). Suppose

$$\mathbf{x} \;=\; \{x(k)\}_0^{143}$$

is the detected beat. The Hermite polynomial basis set consists of the first six Hermite polynomials and a scale factor σ. The value of σ ranges between a minimum value of $1/120$ and $1/90$ and is discretized into 10 values. Denote the Hermite polynomial with order N and scale-factor σ as $\mathbf{h}_N(\sigma) = \{h_N(\sigma, k)\}_0^{143}$. We calculate the dot products

$$c_{\sigma_u,N} \;=\; \sum_{k=0}^{143} x(k) \times h_N(\sigma_u, k)$$

as N varies from 1 to 6 and σ_u varies as described above. The dot products are computed using single precision IEEE floating point arithmetic. The Hermite polynomial values are precomputed and stored in the hardware as tables.

The best fit is determined by the value of the scale factor σ_u, which minimizes the mean square error

$$\parallel x \;-\; \sum_{j=0}^{N-1} c_{\sigma,j} \times h_j(\sigma) \parallel_2$$

This value of σ and the corresponding coefficients $c_{\sigma,j}$ are the features of the beat extracted by the Hermite fit. These values are used for further characterization of the beat as normal or anomalous [10,29].

The algorithm used for the second stage is shown in Listing 4.

Listing 4. Second stage.

```
void hermiteFitterDaemon()
{
uint32_t beat_index = 0;
while(1)
{
// get the current beat from
// the filtered sample buffer.
getCurrentBeat();

// compute all the inner products.
ComputeInnerProducts();

// find the best fit.
computeMSE();

// report the best fit.
sendBestFitToOutput();
}
}
```

4.3. System Architecture

The system architecture follows the two stage approach described at the beginning of the section. The architecture is depicted in Figure 7.

A UART is used to configure the system by downloading the pre-calculated Hermite polynomials, the filter coefficients, and some configuration parameters. In this case, there

are sixty distinct Hermite polynomials, each with 144 samples, with each sample being coded in single precision IEEE floating point format (4 bytes per sample).

After the initial configuration, ECG samples are streamed to the hardware, and fit coefficients are extracted for every detected beat. The peak throughput and total latency in the signal chain are characterized.

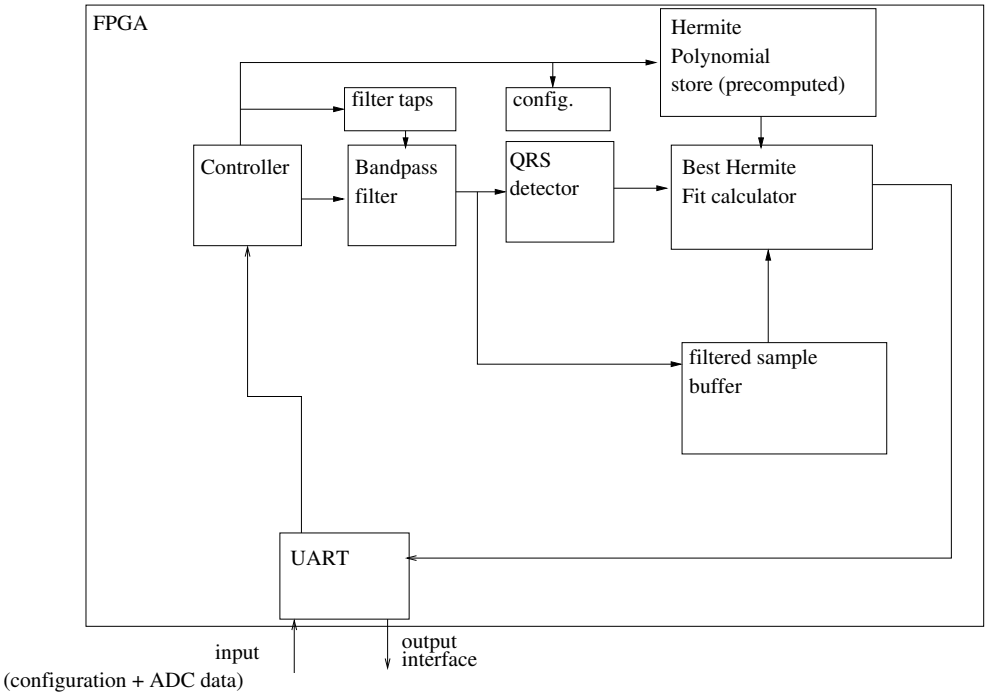


Figure 7. System Architecture.

5. Results

The Xilinx Artix 7 series FPGA xc7a35tcpg236 (Xilinx, San Jose, CA, USA) was used as the platform for the hardware implementation. In particular, we used the BASYS-3 FPGA board from Digilent (Pullman, WA, USA) [32]. For synthesis, we used the Xilinx Vivado 2019.4 tools. The block diagram of the test setup is shown in Figure 8. In this setup, the host computer first uses the UART to download the Hermite polynomial tables and the filter coefficients to the system. After this is performed, ADC samples are streamed to the FPGA over the UART at a baud rate of 115,200. The post Hermite fits and QRS peak locations are monitored by an application on the host computer. It must be stressed that AHIR allows for simulation of the system by using benchmarks written in C. During the simulation, it is possible to select if the simulation is using the compiled C files or if the hardware *functions* are simulated by means of an HDL simulator (i.e., GHDL). In both cases, the input vectors are read from files and the output vectors are stored also in files, so it is possible to check the correctness of the hardware implementation.

For the overall system, we present the

- The hardware utilization;
- The latency through the signal chain;
- The throughput through the signal chain;
- The power dissipation in the system due to the computation activity; and
- The reconstructed waveforms from the calculated Hermite polynomial fits.

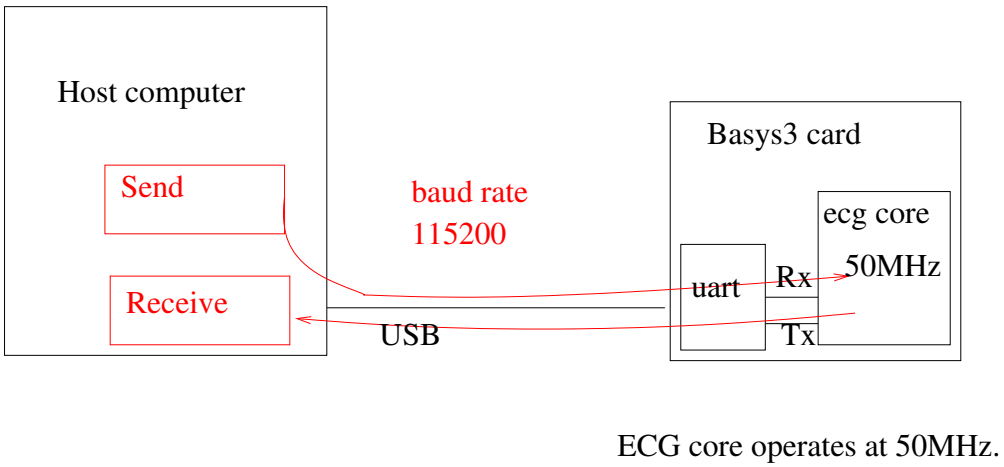


Figure 8. Test setup using the BASYS-3 FPGA card.

The summary of resource utilization is shown in Table 1. For these particular FPGA devices, the limiting factors are the look-up tables (LUT). Thus, devices with more logic resources are required if the order of the polynomial is to be increased.

Table 1. FPGA resource utilization.

Resource	Quantity	Utilization
LUTs	19,663	94.8%
Flip-flops	23,813	57.24%
RAM	50 KB	40.0%
DSP blocks	21	23.33%

To measure the latency in the entire signal chain, we timed the difference between the entry of the first byte of an ECG sample and the exit of the last byte of the Hermite characterization for the corresponding beat. For the throughput, we observed the maximum rate at which beat data could be supplied to the system. For a clock of 50 MHz, the latency and throughputs obtained were 0.82 s and 3 beats/s.

To characterize the power consumption, we observed the difference between the idle current drawn by the FPGA when it was quiescent (unprogrammed) and the current drawn by the FPGA during full speed (maximum throughput) operation. We use the power measurement setup presented in Figure 9.

Basys 3 board features a jumper JP2 that is used as power source select and is located at the entrance of the power supply. It selects whether power supply comes from the USB cable or External power supply. In this work, we use USB power supply of 5 V. We add a shunt resistance over this jumper and use differential probe to measure the voltage over the shunt. Since the resistance is in series with the power supply, we are able to obtain the current that goes to the board from the power supply. By knowing the input voltage and input current, we obtain the power consumed by the board. The resistance value is chosen to ensure the correct functionality of the power supply regulators located on the Basys 3 board, as explained next.

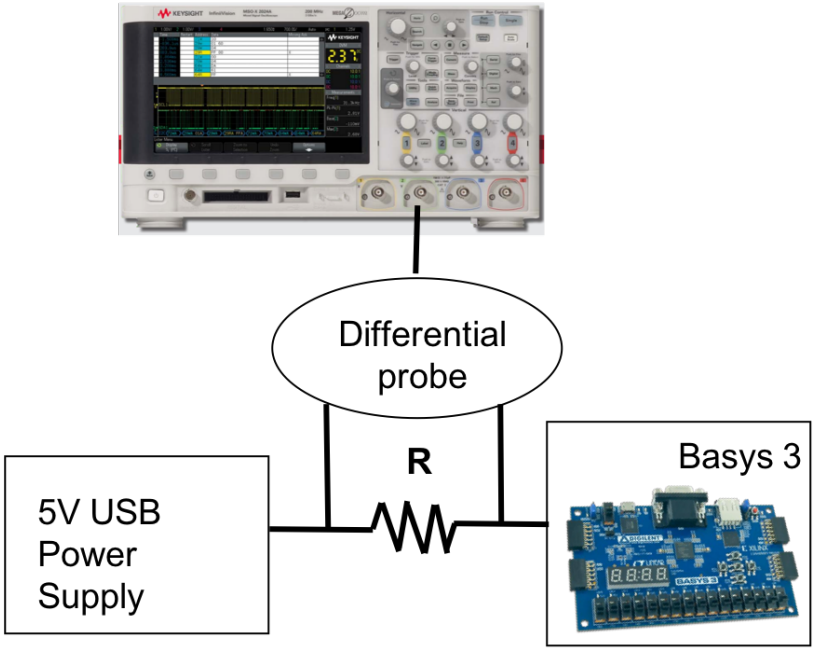


Figure 9. Power measurement setup.

Voltage regulator circuits create the required 3.3 V, 1.8 V and 1 V from the main power supply [32]. A power supply of 1 V is used for an FPGA core; 1.8 V is used for an auxiliary FPGA supply and RAM memory; and 3.3 V is used for IO pins, USB connection, clocks, Flash, etc. Based on typical and maximum current values for each of these supplies, listed in [32], we compute an approximate value for the shunt resistance. According to our estimates, the peak current values for the design should not exceed 80mA, and current demand on the other two supplies should not be extreme either. As a result, when maximum typical current values for the 1.8 V and 3.3 V (150 mA and 1.5 A, respectively) and 80 mA for the 1 V supply are assumed, an approximate value for the resistance is 0.52 Ω. We use 0.47 Ω for our measurements as a value that is close to the estimated one.

Since we are interested in the current consumed by the design only, we first measure the current when the FPGA is programmed and the application is running, i.e., data are sent and received. The measured current is 170.96 mA on average. Then, we subtract the current measured when the FPGA is programmed, but without any data traffic, that becomes 165.36 mA. By subtracting this current, we eliminate the current consumed by other parts of the board as well as the FPGA static current. Consequently, the proposed design consumes 5.6 mA on average. When this current is multiplied by the 5 V input voltage, it results in 28 mW of approximated FPGA dynamic power.

The results are summarized in Table 2. The obtained latency and throughput fits real-time requirements, and the power consumption is low.

Table 2. FPGA implementation metrics.

Latency	0.82 s
Throughput	3 beats per s
Power	28 mW

6. Discussion

The hardware implementation of automatic ECG analysis systems is essential for ambulant monitorization of patients, and there are several examples in the literature for both ASIC [23,24] and FPGA [25,26] implementations. However, to the best of our knowledge, there are no hardware implementations of ECG signal processors that apply the Hermite fit for beat compression or classifications. For example, the work in [26] describes

the implementation of another technique called Empirical Mode Decomposition applied to ECG signals in a Spartan 3E FPGA but does not report power, performance and area metrics. As for the detection performance, the overall accuracy reported is 94.8%, while with Hermite functions, it is possible to achieve 96.66%. The work in [25] is a HW/SW co-design where the QRS complex extraction is implemented in an FPGA and is based on geometrical properties of a two-dimensional phase-space portrait of the ECG signal, while the beat classification is performed by Open Source ECG analysis software. The data are read from and written to the on-board DDR memory, while the data proposed in this work are sent and received by UART, corresponding to a more realistic case, since it could be easily replaced by an ADC interface. Additionally, the pre-processing and pre-partition are performed on the software in [25], so a fair comparison with this work would be difficult to achieve. The authors reported a premature ventricular detection of 92.36%, while with Hermite functions, it is possible to reach 96.86%.

Preliminary results of the proposed design were presented in [17]. Only the Hermite fit process was tackled in our previous work, so the pre-processing chain was neglected. A peak power consumption of 3 W was reported in contrast with the averaged power of 28 mW achieved in the current design. This new version of the circuit can be used to feed a hardware block to perform data compression or classification in real-time with a low power consumption.

The reported performance metrics are promising. The latency is close to a second, which is suitable given that heart rates are commonly between 1 and 2 beats/s; thus, the results of the first beat characterization appear after 1 or 2 beats. The throughput is around 3 beats/s, which covers heart rates of 180 beats/min, which is an extreme situation for a person. Finally, the power consumption is around 30 mW, which is a low value for an FPGA.

Summarizing, the results yield that the system is capable of real-time and low-power processing.

7. Conclusions

In this paper, we presented the design of an FPGA-based system able to perform real-time ECG characterization through Hermite polynomials. The AHIR HLS tool was used to perform the development and testing. The system was successfully implemented on a low-cost board with a latency of less than 1 s, a throughput of 3 beats/s and a power consumption around 28 mW. Hence, we demonstrated that complex low power, real-time ECG analysis is possible through high-level synthesis.

The current design can be easily modified and extended due to the flexibility provided by the AHIR set of tools. On one hand, the number of polynomials used in the estimation (i.e., N) can be increased to improve the accuracy of the estimations. Moreover, a clustering block to help in the classification process can be added [10]. In any case, it is clear that a bigger FPGA device is necessary. Additionally, the throughput can be increased to consider higher heart rates, which involves increasing parallelism and, therefore, increasing the resources demand. All of these new ideas can be easily designed and tested with the HLS approach provided by AHIR.

Author Contributions: M.P.D., G.C., D.G.M. and A.O., design of the signal processing algorithms. M.P.D., G.C. and R.J., conceptualization, implementation and testing of the research. All authors developed the methodology. M.P.D., G.C. and R.J. discussed the basic structure of the manuscript, drafted its main parts, and reviewed and edited the draft. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partially funded by the Spanish Ministry of Science, Innovation and Universities through project RTI2018-095324-B-I00.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ASIC	Application-Specific Integrated Circuit
ECG	Electrocardiogram
FIR	Finite Impulse Reponse
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Units
HLS	High-Level Synthesis
LUT	Look-Up Table
MSE	Mean Square Error
MMSE	Minimum Mean Square Error
QRS	Complex composed of the waves Q, R and S of the ECG
RAW	Read After Write
SNR	Signal-to-Noise Ratio
WAR	Write after Read

References

1. Roth, G.; Mensah, G.; Fuster, V. The Global Burden of Cardiovascular Diseases and Risks: A Compass for Global Action. *J. Am. Coll. Cardiol.* **2020**, *76*, 2980–2981. [CrossRef]
2. Kiranyaz, S.; Ince, T.; Pulkkinen, J.; Gabbouj, M. Personalized long-term ECG classification: A systematic approach. *Exp. Syst. Appl.* **2011**, *38*, 3220–3226. [CrossRef]
3. Lagerholm, M.; Carsten, P.; Braccini, B.; Edenbr, L.; Sörnmo, L. Clustering ECG complexes using Hermite functions and self-organizing maps. *IEEE Trans. Biomed. Eng.* **2000**, *47*, 838–848. [CrossRef]
4. de Chazal, P.; O'Dwyer, M.; Reilly, R. Automatic classification of heartbeats using ECG morphology and heartbeat interval features. *IEEE Trans. Biomed. Eng.* **2004**, *51*, 1196–1206. [CrossRef] [PubMed]
5. Yochum, M.; Renaud, C.; Jacquir, S. Automatic detection of P, QRS and T patterns in 12 leads ECG signal based on CWT. *Biomed. Signal Process. Control* **2016**, *25*, 46–52. [CrossRef]
6. Sörnmo, L.; Laguna, P. *Bioelectrical Signal Processing in Cardiac and Neurological Applications*; Elsevier: Amsterdam, The Netherlands, 2005.
7. Swan, M. Sensor Mania! The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0. *J. Sens. Actuator Netw.* **2012**, *1*, 217–253. [CrossRef]
8. Villegas, A.; McEneaney, D.; Escalona, O. Arm-ECG Wireless Sensor System for Wearable Long-Term Surveillance of Heart Arrhythmias. *Electronics* **2019**, *8*, 1300. [CrossRef]
9. Martis, R.; Acharya, U.; Adeli, H. Current methods in electrocardiogram characterization. *Comput. Biol. Med.* **2020**, *48*, 133–149. [CrossRef] [PubMed]
10. Márquez, D.; Otero, A.; García, C.; Presedo, J. A study on the representation of QRS complexes with the optimumnumber of Hermite function. *Biomed. Signal Process. Control* **2015**, *22*, 11–18. [CrossRef]
11. Young, T.; Huggins, W. On the representation of electrocardiograms. *IEEE Trans. Biomed. Electron.* **1963**, *10*, 86–95.
12. Homaeinezhad, M.; Erfanianmoshiri-Nejad, M.; Naseri, H. A correlation analysis-based detection and delineation of ECG characteristic events using template waveforms extracted by ensemble averaging of clustered heart cycles. *Comput. Biol. Med.* **2014**, *44*, 66–75. [CrossRef]
13. Laguna, P.; Jané, R.; Olmos, S.; Thakor, N.; Rix, H.; Caminal, P. Adaptive estimationof QRS complex wave features of ECG signal by the Hermite model. *Med. Biol. Eng. Comput.* **1996**, *34*, 58–68. [CrossRef]
14. Márquez, D.G.; Otero, A.; Félix, P.; García, C.A. On the Accuracy of Representing Heartbeats with Hermite Basis Functions. In Proceedings of the International Conference on Bio-Inspired Systems and Signal Processing (BIOSIGNALS 2013), Barcelona, Spain, 11–14 February 2013; pp. 338–341.
15. Márquez, D.G.; Félix, P.; García, C.A.; Tejedor, J.; Fred, A.L.; Otero, A. Positive and Negative Evidence Accumulation Clustering for Sensor Fusion: An Application to Heartbeat Clustering. *Sensors* **2019**, *19*, 4635. [CrossRef] [PubMed]
16. Gil, A.; Márquez, D.; Caffarena, G.; Iriarte, A.; Otero, A. GPU-Based Acceleration of ECG Characterization Using High-Order Hermite Polynomials. *Curr. Bioinform.* **2016**, *11*, 430–439. [CrossRef]
17. Lakhotia, K.; Caffarena, G.; Gil, A.; Márquez, D.; Abraham, O.; Desai, M. Low-Power, Low-Latency Hermite Polynomial Characterization of Heartbeats Using a Field-Programmable Gate Array. In Proceedings of the International Work-Conference on Bioinformatics and Biomedical Engineering, Granada, Spain, 7–9 April 2014; pp. 266–276.
18. Xilinx. Vitis High-Level Synthesis. 2021. Available online: https://www.xilinx.com/products/design-tools/vivado/high-level-design.html (accessed on 10 July 2021).
19. Intel. Intel HLS Compiler. 2021. Available online: https://www.intel.la/content/www/xl/es/software/programmable/quartus-prime/hls-compiler.html (accessed on 10 July 2021).

20. Sahasrabuddhe, S. A Competitive Pathway from High-Level Programs to Hardware. Ph.D. Thesis, IIT Bombay, Bombay, India, 2009.
21. Sahasrabudhe, S.D.; Subramanian, S.; Ghosh, K.; Arya, K.; Desai, M.P. A C-to-RTL flow as an energy efficient alternative to the use of embedded processors in digital systems. In Proceedings of the 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, Lille, France, 1–3 September 2010; pp. 147–154.
22. Rinta-Aho, T.; Karlstedt, M.; Desai, M. The ClickToNetFPGA Tool-chain. In Proceedings of the USENIX ATC-2012, Boston, MA, USA, 13–15 June 2012; USENIX Association: Berkeley CA, USA, 2012.
23. Chen, Z.; Luo, J.; Lin, K.; Wu, J.; Zhu, T.; Xiang, X.; Meng, J. An Energy-Efficient ECG Processor with Weak-Strong Hybrid Classifier for Arrhythmia Detection. *IEEE Trans. Circ. Syst. II Express Briefs* **2018**, *65*, 948–952. [CrossRef]
24. Wu, J.; Li, F.; Chen, Z.; Pu, Y.; Zhan, M. A Neural Network-Based ECG Classification Processor with Exploitation of Heartbeat Similarity. *IEEE Access* **2019**, *7*, 172774–172782. [CrossRef]
25. Cvikl, M.; Zemva, A. FPGA-oriented HW/SW implementation of ECG beat detection and classification algorithm. *Digit. Signal Process.* **2010**, *20*, 238–248. [CrossRef]
26. Kumari, L.R.; Sai, Y.P.; Balaji, N.; Viswada, K. FPGA Based Arrhythmia Detection. *Procedia Comput. Sci.* **2015**, *57*, 970–979. [CrossRef]
27. Sandryhaila, A.; Saba, S.; Puschel, M.; Kovacevic, J. Efficient Compression of QRS Complexes Using Hermite Expansion. *IEEE Trans. Signal Process.* **2012**, *60*, 947–955. [CrossRef]
28. Moody, G.B.; Mark, R.G. The impact of the MIT-BIH arrhythmia database. *IEEE Eng. Med. Biol. Mag.* **2001**, *20*, 45–50. [CrossRef] [PubMed]
29. Tejedor, J.; Marquez, D.G.; Garcia, C.A.; Otero, A. A Tandem Feature Extraction Approach for Arrhythmia Identification. *Electronics* **2021**, *10*, 976. [CrossRef]
30. Bronzino, J.D. (Ed.) *Medical Devices and Systems: Biomedical Engineering Handbook*, 3rd ed.; CRC Press: Boca Raton, FL, USA, 2007; 1400p.
31. MobilECG Laboratories Kft. T-filter Design Tool. 2021. Available online: http://t-filter.engineerjs.com/ (accessed on 30 June 2021).
32. Digilent. Basys-3 Reference Manual. 2021. Available online: https://reference.digilentinc.com/programmable-logic/basys-3/reference-manual (accessed on 10 July 2021).