Microelectronics System Design Research Group

School of Electrical, Electronic and Computer Engineering

Electrocardiogram Monitor Design from Concept to Silicon

Basel Halak,

Julian Murphy, Alex Yakovlev

Technical Report Series

NCL-EECE-MSD-TR-2011-168

Contact: b.halak@ncl.ac.uk

Electrocardiogram Monitor Design From Concept to Silicon

Basel Halak
Julian Murphy, Alex Yakovlev
School of EECE
Newcastle University
Newcastle upon Tyne, United Kingdom

Abstract. **The proliferation of chronic diseases such as heart failure due to the aging population is causing an increasing pressure on healthcare systems; this makes it extremely difficult to maintain the quality of care for patients without significant increase of budget. Switching resources from crisis management by hospitalizing patients to health maintenance through home tele-monitoring is promising solution that can help achieve high quality of care at an affordable cost. To make this vision a reality, there is a strong need for portable diagnostic systems. Such devices are expected to battery operated and to communicate wirelessly, therefore they have stringent power consumption constraints. This work investigates the design and implementation of electrocardiogram monitor used for the detection and classification of the heart's electrical signal. This device is an essential part of chronic heart failure diagnostic equipments and implantable pacemakers. The design process of such a system is described from initial specifications to final implementation and testing on field programmable gate array (FPGA). Power-saving methodologies employed at every stage of development are also presented.**

1. Introduction

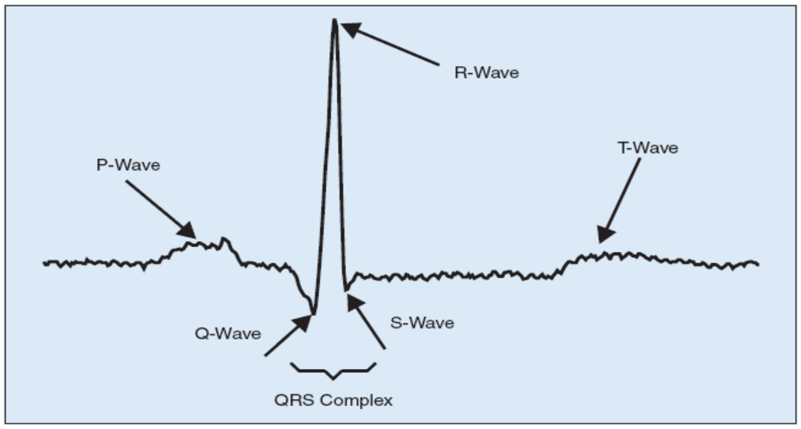
Chronic heart failure (CHF) is a complex, debilitating syndrome due to cardiac dysfunction that impairs the ability of the ventricle to fill with, or eject, blood. As a result, typical symptoms such as dyspnoea and fatigue occur at rest or with reduced physical effort. As the prevalence of CHF increases with the ageing of populations internationally, it is becoming increasingly difficult to maintain chronic disease patient quality-of-care without new solutions. Switching resources from crisis management (by hospitalizing patients) to health maintenance (e.g. through home telemonitoring) may be an affordable method to maintain and improve the quality of care for CHF. Recent Study showed that remote monitoring of patients with CHF, reduces deaths and hospitalizations and may provide benefits on health care costs and quality of life [1].

A monitoring system can be envisaged to constitute of a biomedical sensor and a computerised diagnostic system. The former can be described as mobile electronic devices that can be unobtrusively embedded in the user's outfit as part of the clothing or an accessory. Such a device will perform several measurements (i.e. heart signal, blood pressure and SpO2). These data will be initially examined in the sensor and then sent wirelessly to the diagnostic software package. The latter conducts comprehensive analysis on the sensor data, then, it carries out necessary measures based on the state of the patients. Examples of such devices are many in the literature [1-4].
The electrocardiogram (ECG) signal that is an electric activity generated by a heart is typically used as a primary method for diagnosing the abnormal condition of a heart.

The QRS complex is the most striking waveform within the (ECG). Since it reflects the electrical activity within the heart during the ventricular contraction, the time of its occurrence as well as its shape provide much information about the current state of the heart. Due to its characteristic shape (see Fig. 1) it serves as the basis for the automated determination of the heart rate, as an entry point for classification schemes of the cardiac cycle, and often it is also used in ECG data compression algorithms. In that sense, QRS detection provides the fundamentals for almost all ECG analysis algorithms. Therefore, QRS detection circuitry is an essential component of ECG monitoring systems (e.g. implantable pacemaker, wearable heart monitor).

This work describes the design process of an ECG monitoring system from concept to hardware implementation on FPGA, with a special emphasis on the QRS detection circuit.

The structure of the report is as follows, section 2 discusses in depth the stages of the design process, it starts with a description of the specification of the system, followed by an in-depth analysis of the algorithm used to achieve the required behaviour, finally the hardware implementation is presented along with a verilog models of different part of the design. Section 3 briefly outlines the power reduction techniques employed in this project. A summary is presented in section 4.



Figurer 1: The QRS complex within the ECG signal

2. QRS Detection (ECG) Design Flow

An ECG monitoring system is envisaged in figure 2. The electrical signal of the heart is captured and sampled by the data acquisition block. These digital samples are applied to the data processing block which implements the main QRS detection algorithm. The interface stage is responsible for connecting the output of the detection algorithm to other systems. The design of this stage will vary depending on the context within which this system is used. For clinical ECG monitor the interface needs to condition the signal to be displayed on a screen monitor. For implantable ECG devices (e.g. pacemakers), this interface might need to transmit the signal wirelessly to an outside system.

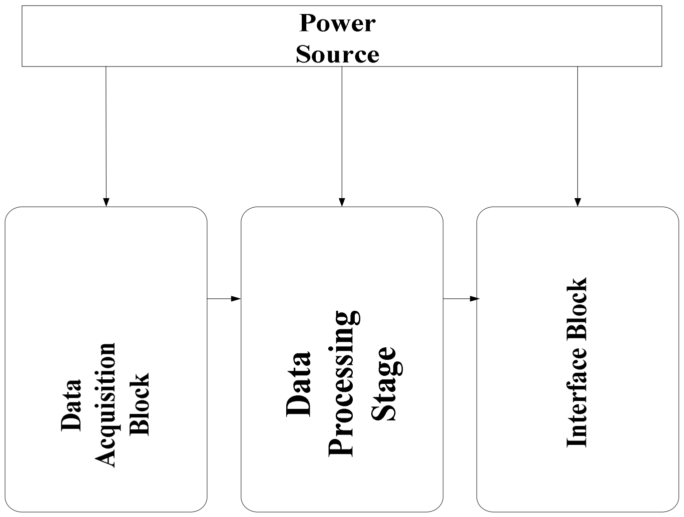


Figure 2: Generic Functional of QRS Detection System

The design and implementation of this system is detailed in this section, for the purpose of completeness a summary of the design flow is presented in the first subsection.

2.1 . Design Flow of Systems

The overall process of a system design such an ECG monitor begins with identifying the system requirements. They are the required functions, performance, power, cost, reliability, and development time for the system. These requirements form the preliminary specifications often produced by the development teams and marketing professionals. Figure 3 shows a generic design methodology flow at high level.

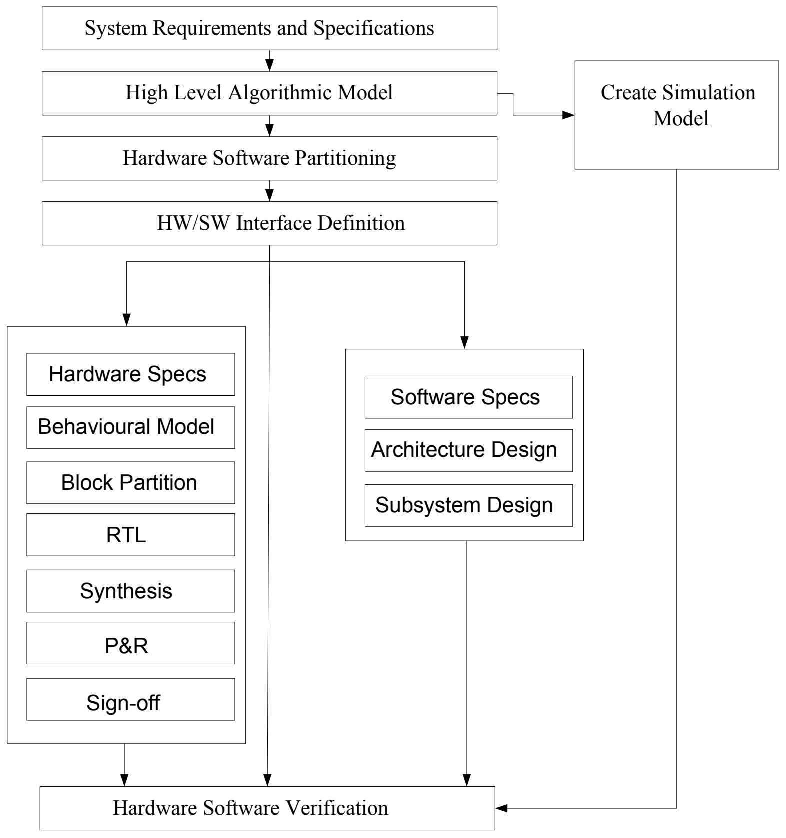

System Requirements and Specifications
High Level Algorithmic Model
Create Simulation Model
Hardware Software Partitioning
HW/SW Interface Definition
Hardware Specs
Behavioural Model
Block Partition
RTL
Synthesis
P&R
Sign-off
Software Specs
Architecture Design
Subsystem Design
Hardware Software Verification

Figure 3: General Design Methodology of Systems

For a specific design, some of these steps may not be used or the flow may be somewhat modified. Some form of validation and analysis is necessary at every step in order to reduce the risk of errors.

As the high-level model begins to finalize, a model of the system is created in a programming language such as C++. The system architect then decides on the software and hardware partitions to determine what functions should be done by hardware and what should be achieved by software applications. Partitioning the software and hardware subsystems is currently a manual process that requires experience and a cost/performance trade-off. This is however an active area of research, where new methods are being developed to automate this process, [5, 6], this is however beyond the scope if this work. During the partitioning step, the interface and protocols between hardware and software should be defined; in addition detailed specs on individual partitions of both software and hardware. Once the hardware and software partitions have been determined, a behavioral model of the hardware is created together with a working prototype of the software. The co-simulation of hardware and software allows these components to be refined and to develop an executable model with fully functional specs. These refinements continue throughout the design phase. Some of the major hardware design considerations in this process are clock tree, clock domains, layout, floor planning, buses, verification, synthesis, and interoperability issues. In addition, the entire project should have consistent rules and guidelines clearly defined and documented, with additional structures to facilitate silicon debugging and manufacturing tests. If the hardware behavior is assigned to a standard processor, it will be fed into the compiler of this processor. This compiler should translate the design description into machine code for the target processor. If it is to be mapped into an ASIC, a high-level synthesis tool can synthesize it. The high-level synthesizer translates the behavioral design model into a netlist of RTL library components. This concludes this section that provided a summary of the design flow of systems. The remaining subsections will detail the stages of QRS detector design process.

2.2. System Requirements and Specifications

The goal of this of the project is to develop a hardware prototype of an ECG monitor, which accepts as an input digitized ECG recordings from the MIT-BIH arrhythmia database. The latter is the most widely used set of standard test material for evaluation of arrhythmia detectors[7].

First, an interface between the MIT-BIH library and the rest of the system is needed; this interface is going to be built in software using MATLAB and will be run on a desktop computer.

The QRS detection algorithm is going to be implemented on a dedicated external device, therefore, each ECG sample whose size is at least 8 bits needs to be transferred form the computer to the external hardware. For this purpose we are going to use the serial port, which uses RS-232 communication protocol, to transmit data.

The QRS detection logic expects to receive a complete ECG sample i.e. parallel data, therefore there is a need for a piece of logic that can translate the serial data received from the RS_232 port into a parallel form. For this part of the system, a universal asynchronous receiver (UAR) will be used.

The output of the detected signal needs to be displayed on a monitoring screen, therefore, the data should be transferred back to the computer through the RS-232 port, for this purpose, a universal asynchronous transmitter (UAT) is employed. To display the detected samples, a piece of software is going to be developed using MATLAB.

A complete diagram of the system is shown in figurer 4. In comparison with figure 2, the S/H interface and UART receiver blocks comprise the data acquisition stage, the QRS detector comprises the data processing block and the UART transmitter and H/S interface blocks comprise the interface stage.

QRS detector and UART block will both be implemented in hardware, while the H/S interface block will be implemented in software.

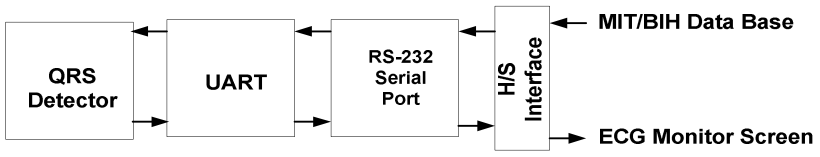


Figure 4: System Specifications Diagram

2.3. High Level Algorithmic Model

Figure 4 showed a high level abstraction of the system, this section will descries the algorithm for each of these blocks.

2.3.1. QRS Detector

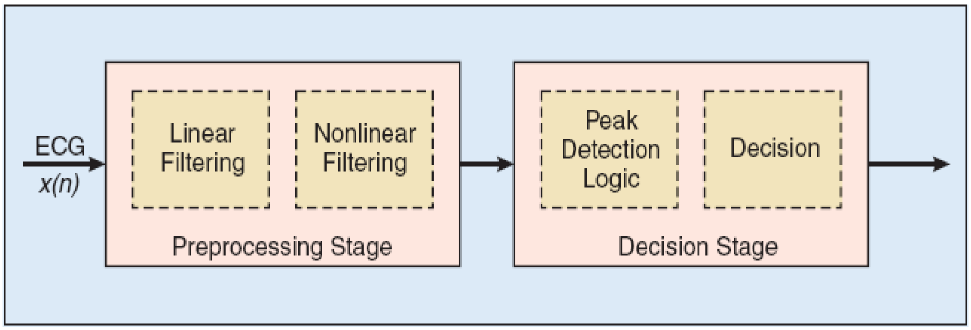


Figure 5: A Common Structure of QRS Detectors[8]

QRS detection is difficult, not only because of the physiological variability of the QRS complexes, but also because of the various types of noise that can be present in the ECG signal. Noise sources include muscle noise, artifacts due to electrode motion, power-line interference, baseline wander, and T waves with high-frequency characteristics similar to QRS complexes. Typical frequency components of a QRS complex range from about 10 Hz to about 25 Hz. Therefore, almost all QRS detection algorithms use a filter stage prior to the actual detection in order to attenuate other signal components, such as P-wave, T-wave, baseline drift, and in-coupling noise. Whereas the attenuation of the P- and T-wave as well as baseline drift requires high-pass filtering, the suppression of in-coupling noise is usually accomplished by a low-pass filter. The combination of low and high pass means effectively the application of a band pass filter, in this case with cut-off frequencies at about 10 Hz and 25 Hz. QRS detection has been a research topic for more than 30 years, a great number of algorithms have been developed with varying performance and complexity. They typically include one or more of three different types of processing steps: linear digital filtering, nonlinear transformation, and decision rule algorithms. A common structure of QRS detector is shown in figure 5 [8]. The choice of a particular algorithm is a design decision, which is made based on the target application. In our case, the application is portable diagnostic systems, therefore, low power, small area and high reliability are the main requirements. The algorithm outlined in [9] is said to have 99.3 detection accuracy which satisfy the reliability requirement. The pre-processing stages are particularly suited for fixed-point arithmetic with a short word length, which helps minimise the silicon area and reduce power consumption. Based on these considerations this algorithm was chosen to be implemented in this project. The algorithm uses the three types of processing steps illustrated in figure 5, Linear processes include a bandpass filter, derivative, and a moving window integrator. The nonlinear transformation used is signal amplitude rectifying. Adaptive thresholds and T-wave discrimination techniques provide the decision part of the algorithm.

The first stage of the algorithm aims to attenuate noise. The signal passes through a digital bandpass filter composed of cascaded high-pass and lowpass filters. Figure 6-b shows the output of this filter. The next process after filtering is differentiation (Figure 6-c) , followed by rectifying (Figure 6 (d)), and then moving window integration (Figure 6-e). Information about the slope of the QRS is obtained in the derivative stage. The squaring process intensifies the slope of the frequency response curve of the derivative and helps restrict false positives caused by T waves with higher than usual spectral energies. The moving window integrator produces a signal that includes information about both the slope and the width of the QRS complex. Figure (6-f) is the same as the original ECG in Figure (6-a) except delayed by the total processing time of the detection algorithm. Figure(6-g) shows the final output stream of pulses marking the locations of the QRS complexes after application of the adaptive thresholds. The requirement and the and the equations of each of these stages are detailed below.

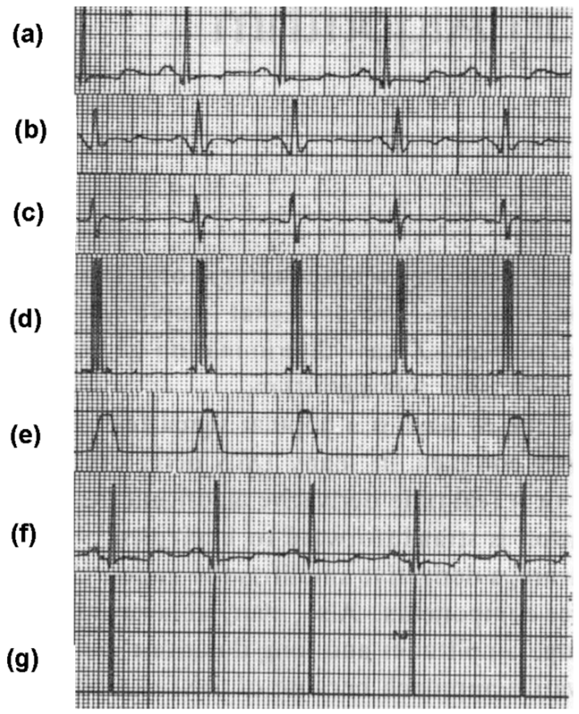


Figure 6: QRS Detection Algorithm Processing Steps for a Normal ECG from MIT/BIH Database.

(a) Original signal. (b) Output of bandpass filter.

(c) Output of differentiator. (d) Output of rectifying process.

(e) Results of moving-window integration.

(f) Original ECG signal delayed by the total processing time. (g) Output pulse stream.

Bandpass Filter

The bandpass filter reduces the influence of muscle noise, 60 Hz interference, baseline wander, and T-wave interference. The desirable passband to maximize the QRS energy is approximately 5-15 Hz. The bandpass filter is designed to be fast, real-time recursive filter in which poles are located to cancel zeros on the unit circle of the z plane. This approach results in a filter design with integer coefficients. Since only integer arithmetic is necessary, a real-time filter can be implemented with a simple logic, which reduces its area and potentially save power. This class of filters having poles and zeros only on the unit circle permits limited passband design flexibility. For our chosen sample rate, it was not possible to design a bandpass filter directly for the desired passband of 5-15 Hz using this specialized design technique. Therefore, we cascaded the low-pass and high-pass filters described below to achieve a 3 dB passband from about 5-12 Hz, reasonably close to the design goal. The low pass filter has a cut-off frequency of 11 Hz and the gain is 36 and a processing delay of six samples. It is described the following difference equations:

$$y(nT) = 2y(nT-T) - y(nT-2T) + x(nT) - 2x(nT-6T) + x(nT-12T) \quad (1)$$

The high band filter has a cut-off frequency of 8 Hz, low cut-off frequency of about 5 Hz and its gain is 32, and the delay is 16 samples. It is described by the following difference equations:

$$y(nT) = 32y(nT-16T) - [y(nT-T) + x(nT) - x(nT-32T)] \quad (2)$$

Derivative

After filtering, the signal is differentiated to provide the QRS complex. A five point derivative is used which can be described with the following difference equation:

$$y(nT) = (1/8T)[-x(nT-2T) - 2x(nT-T) + x(nT+NT) + x(nT+2T)] \quad (3)$$

The frequency response of this derivative is nearly linear between dc and 30 Hz (i.e., it approximates anideal derivative over this range). Its delay is two samples.

Rectifying Function

After differentiation, the signal is rectified point by point. The equation of this operation is as follows:

$$y(nT) = |x(nT)| \tag{4}$$

This makes all data points positive and does nonlinear amplification of the output of the derivative emphasizing the higher frequencies (i.e., predominantly the ECG frequencies).This original algorithm in [9] uses squaring function, however it was found that his operation caused the QRS detector to be somewhat gain sensitive. That is why, in this implementation the absolute value was used, reducing the gain sensitivity and slightly improving the performance of the algorithm.

Moving-Window Integration

The purpose of moving-window integration is to obtain waveform feature information in addition to the slope of the R wave obtained in the derivative stage, because the R-wave slope alone is insufficient for proper QRS detection. The reason behind this fact is that the derivative stage by its very nature, a derivative amplifies the undesirable higher frequency noise components. Also, many abnormal QRS complexes with large amplitudes and long durations are missed in a purely derivative approach because of their relatively low R wave slopes. Thus, To achieve reliable performance, other parameters must be extracted from the signal such as amplitude, width, and QRS energy. The difference equation of a moving window integrator is given as follows.

$$y(nT) = (1/N)\sum_{0}^{N-1} x(nT - NT) \tag{5}$$

where N is the number of samples in the width of the integration window, generally, the width of the window should be approximately the same as the widest possible QRS complex. If the window is too wide, the integration waveform will merge the -QRS and T complexes together. If it is too narrow, some QRS complexes will produce several peaks in the integration waveform. These can cause difficulty in subsequent QRS detection processes. The width of the window is determined empirically. For a sample rate of 200 samples/s used in this project, the window is 32 samples wide.

Peak Detection

Figure 7 shows a typical wave form produced by the time-averaged window for a QRS complex. Although it is easy to visually identify one large peak, simple peak detection algorithms falsely detect multiple peaks due to ripples in the wave. A simple local maxima peak detector has the liability of detecting many small-amplitude peaks. Although both peaks result from the same QRS complex, one peak is classified as resulting from a QRS complex, the other is classified as noise. This can bias the noise level estimate on the high side. In contrast, ripples in the baseline of the signal can bias the noise estimate on the low side. This algorithm eliminates detection of both ripples on large waves and also very small noise peaks. The peak detector finds peaks in the final output of the filtering stages. A detected peak defines an event. The detector algorithm stores the maximal levels encountered

in the signal since the last peak detection. A new peak is defined only after a level is encountered that is less than half the height of the maximal, or peak, level. Detection occurs halfway down the back side of the peak. This approach eliminates multiple detections from ripple around the wave peak.

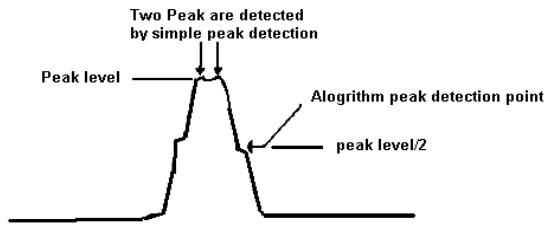


Figure 7: Signal Waveform of the Output of Moving Window Integration Stage

Detection Rules

To reduce false detections and improve accuracy, an additional stage is included which implements a set of detection rules as follows:

1. Ignore all peaks that precede or follow larger peaks by less than 200 ms.
2. If a peak occurs, check to see whether the raw signal contained both positive and negative slopes. If not, the peak represents a baseline shift.
3. If the peak occurred within 360 ms of a previous detection check to see if the maximum derivative in the raw signal was at least half the maximum derivative of the previous detection. If not, the peak is assumed to be a T-wave.
4. If the peak is larger than the detection threshold call it a QRS complex, otherwise call it noise.
5. If no QRS has been detected within 1.5 R-to-R intervals, there was a peak that was larger than half the detection threshold, and the peak followed the preceding detection by at least 360 ms, classify that peak as a QRS complex.

Each time a peak is detected, this stage will classify it as either a QRS complex or noise. This algorithm was described and simulated in C language, it has operated as expected

2.3.2. Universal Asynchronous Transmitter Receiver

UART is an asynchronous serial receiver/transmitter, it is used in this project to as an interface between the RS232 serial port and the QRS detector; it converts data from serial to parallel and vice versa. The asynchronous receiver assembles the individual bits received from the serial port into a complete byte. The asynchronous transmitter takes bytes of data and transmits the individual bits in a sequential fashion to the RS232 serial port. Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver must agree on timing parameters in advance and special bits are added to each word, which are used to synchronize the sending and receiving units. When a word is given to the UART for asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. These two clocks must be accurate enough to not have the frequency drift by more than 10% during the transmission of the remaining bits in the word. After the Start Bit, the individual bits of the word of data are sent, with the Least Significant Bit (LSB) being sent first. Each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits, and the receiver "looks" at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or a 0. For example, if it takes two seconds to send each bit, the receiver will examine the signal to determine if it is a 1 or a 0 after one second has passed, then it will wait two seconds and then examine the value of the next bit, and so on. The sender does not know when the receiver has "looked" at the value of the bit. The sender only knows when the clock says to begin transmitting the next bit of the word. When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity Bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter. When the receiver has received all of the bits in the data word, it may check for the Parity Bits (both sender and receiver must agree on whether a Parity Bit is to be used), and then the receiver looks for a Stop Bit. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be garbled and will report a Framing Error to the host processor when the data word is read. The usual cause of a Framing Error is that the sender and receiver clocks were not running at the same speed, or that the signal was interrupted. Regardless of whether the data was received correctly or not, the UART automatically discards the Start, Parity and Stop bits. If the sender and receiver are configured identically, these bits are not passed to the host. If another word is ready for transmission, the Start Bit for the new word can be sent as soon as the Stop Bit for the previous word has been sent. Because asynchronous data is "self synchronizing", if there is no data to transmit, the transmission line can be idle. The speed of the serial connection is measured in bits-per-second or normally expressed as "baud rate". The duration of a bit is dependent on the baud rate. The baud rate is the number of times the signal can switch states in one second. Thus, if the line is operating at 115200 baud, the line can switch states 115200times per second. This means each bit has the duration of 1/ 115200 of a second or about 100 micro second. In this project the baud rate of UART module is set to 115200.

2.3.3. Software Hardware Interfaces

The purpose software/ Hardware interface is two folds. The first is to send ECG samples from the MIT/BIT data base to the RS-232 serial port. The second is to display the ECG signal together with the detected peaks of the QRS complex on the computer monitoring screen. These two functions were constructed in MATALB, the program is included in the appendixes.

2.4. Hardware Implementation

This section outlines the methodology used to design the hardware parts of the system, namely, the QRS detection logic and the universal asynchronous transmitter and receiver.

The overall design flow is illustrated in figure 8.

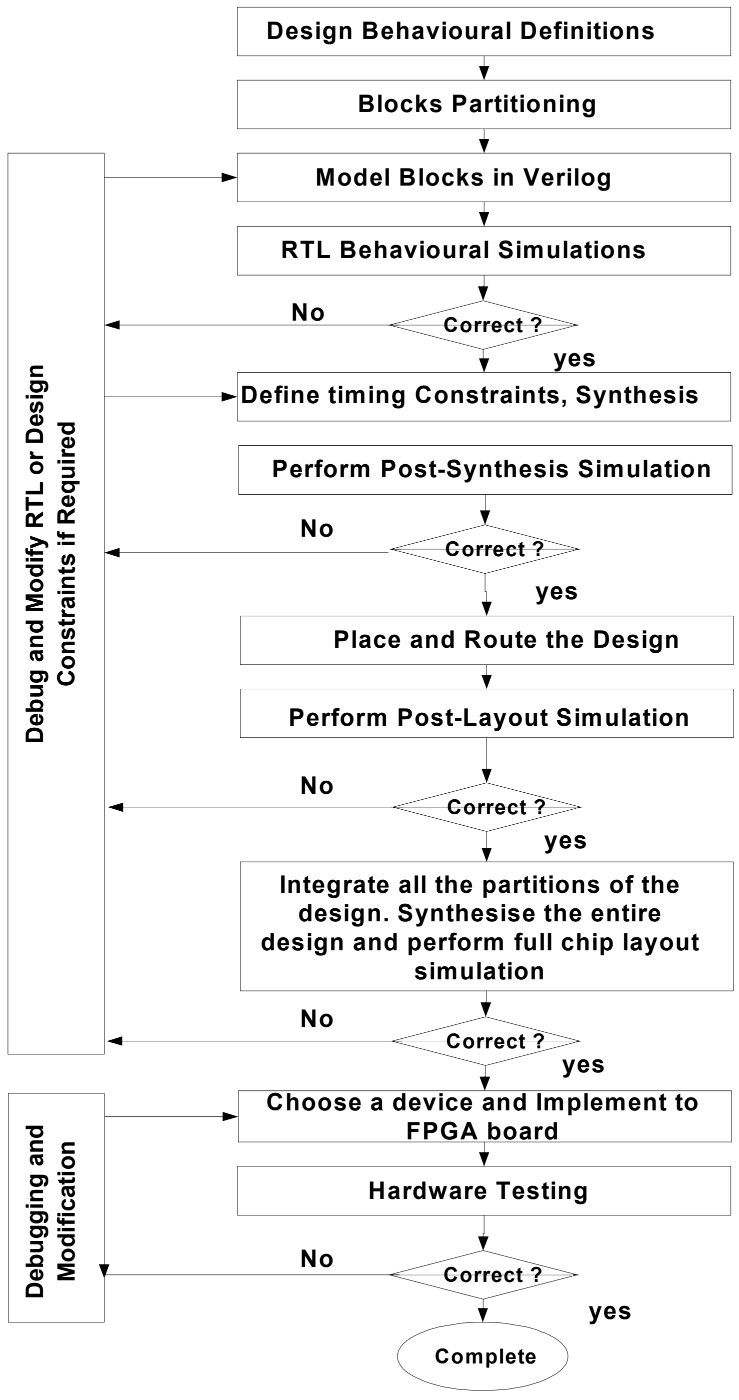

Design Behavioural Definitions
Blocks Partitioning
Model Blocks in Verilog
RTL Behavioural Simulations
Correct ?
No
yes
Define timing Constraints, Synthesis
Perform Post-Synthesis Simulation
Correct ?
No
yes
Place and Route the Design
Perform Post-Layout Simulation
Correct ?
No
yes
Integrate all the partitions of the design. Synthesise the entire design and perform full chip layout simulation
Correct ?
No
yes
Choose a device and Implement to FPGA board
Hardware Testing
Correct ?
No
yes
Complete
Debug and Modify RTL or Design Constraints if Required
Debugging and Modification

Figure 8: Project Hardware Design and Implementation Flow

2.4.1. Implementation of QRS Detection Logic

The design is first partitioned into smaller functional blocks, based on the stages of the QRS detection algorithm, namely: low pass filter, high pass filter, differentiator, rectifier, moving average integrator and peak detector. The first block to design is a low pass digital filter, the specification of which (as described earlier in section 2.3.2) is a cut-off frequency of 11 Hz, and a gain of 32. It is implemented as Infinite-duration Response (IIR) filter, this filter was described in C-language and simulated to verify its functionality. The next step was to convert it into verilog. The input of the filter was chosen to b 8-bit words to make it compatible with the UART block inputs/outputs sizes. This finite word length limits the resolution and the dynamic range that can be represented by the filter, for 8-bits signed integer bits, the dynamic range is [-128 : +128], the MIT/BIT ECG samples used to test the filter was verified to fall into this range. It should be noted here that it is possible to implement the low filter with a larger input size, e.g (16 bits), in which case there is a need for interface logic between the UART and the QRS detector. this logic will collect the 8-bits words received from the UART and form a larger words (e.g. 16 bits words), then apply them to the QRS detector. The choice of 8-bit words was made to facilitate implementation. Another issue that had to be considered is that the arithmetic operation performed by the filter may lead to overflow or underflow errors; this potential problem was solved by accurately sizing the output of the filter. The filter was designed to be fully synchronous with active high synchronous reset. There are various architectures to implement IIR filters, each of which exhibit different requirements for physical resources and different sensitivities to errors caused by finite word length for the data and parameters. The structure adopted in this project is called Type-1 IIR filter, it consists of separate feed forward and feedback blocks implemented as a pair of shift registers-one to hold samples of the input, x[n],and another to hold samples of the output, y[n]. A complete description of the filter in verilog is shown in figure 9.

```
module lowpass (Data_out, Data_in, clock, reset, enable);

  parameter forward_order = 8;

  parameter backward_order = 2;

  parameter word_size_in = 8;

  parameter word_size_out = 2*word_size_in + 2;

  output    signed       [word_size_out -1: 0] Data_out;

  input     signed       [word_size_in-1: 0] Data_in;

  input                  clock, reset,enable;
  parameter b0 = 8'd1;  // Feedforward filter coefficients
  parameter b1 = 0;
  parameter b2 = 0;
  parameter b3 = 0;
  parameter b4 = 8'd2;
  parameter b5 = 0;
  parameter b6 = 0;
  parameter b7 = 0;
  parameter b8 = 8'd1;
  parameter a0 = 8'd2;
  parameter a1 = 8'd1;  // Feedback filter coefficients
 reg signed       [word_size_in-1: 0]     Samples_in [1: forward_order];
 reg signed       [word_size_out-1: 0]    Samples_out [1: backward_order];
 wire signed      [word_size_out -1: 0]   Data_feedforward;
 wire signed      [word_size_out -1: 0]   Data_feedback;
  integer         k,m;
 assign Data_feedforward =      b0 * Data_in
+ b1 * Samples_in[1]
+ b2 * Samples_in[2]
+ b3 * Samples_in[3]
- b4 * Samples_in[4]
+ b5 * Samples_in[5]
+ b6 * Samples_in[6]
+ b7 * Samples_in[7]
+ b8 * Samples_in[8];
  assign Data_feedback = a0 * Samples_out [1] - a1 * Samples_out [2];
  assign Data_out = Data_feedforward + Data_feedback;
  always @ (posedge clock)
  begin
  if (reset)
```

```
       begin
         for (k = 1; k <= forward_order; k = k+1)
             begin
             Samples_in [k] <= 0;
             end
        for (m = 1; m <= backward_order; m = m+1)
             begin
             Samples_out [m] <= 0;
             end
       end
  else if (enable)
       begin
      Samples_in [1] <= Data_in;
      Samples_out [1] <= Data_out;
      for (k = 2; k <= forward_order; k = k+1)
          begin
          Samples_in [k] <= Samples_in [k-1];
           end

      for (m = 2; m <= backward_order; m = m+1)
            begin
            Samples_out [m] <= Samples_out [m-1];
      end
      end
  end
endmodule
```

Figure 9: Verilog Model of Low Pass IIR Filter

This filter was simulated using the Modelsim tool v6.5c from Mentor Graphic, The implemented

modules are tested with a test-bench that applies a 2000 ecg samples to the input of the filter and write the output results to a text file. The response of the filter is then compared against that of the MATLAB model to verify their equivalence. The test-bench is shown in figure 10.

```
module lowpassTest;


  // DUT connections

 parameter W = 7; // Bit width - 1

  reg clk ;

  reg [W:0] data;

  reg reset;

  reg enable;

  wire [2*W+2:0] Q;

  reg [W:0] mem;

  parameter Amax = 2000;

  integer fd,fr ,count,code,i,j;

reg [W:0] ecgsample [0:Amax-1];

// DUT

lowpass (.Data_out(Q), .Data_in(data), .clock(clk), .reset(reset),
.enable(enable));

// clock generator

  always

  begin

    #5 clk = 1;

    #5 clk = 0;

  end
```

```
// Stimulus generator
initial
begin
    fd = $fopen("ecg.dat","r");
    clk = 0;
    data = 0;
    code = 1;
    count=0;
    i=0;
    j=0;
    fr = $fopen("D:\\ecgr.txt");
    while (count < 2000)
    begin
       code = $fread(mem, fd);
       ecgsample [i] = mem;
       i =i+1;
       count =count +1;
      // @(posedge clk);
    end
#200
  reset=1'b1;
    data =0;
#200;
reset=1'b0;
while (j < 2000)
begin
#5000;
     enable = 1'b1;
     data = ecgsample[j];
```

```
      $fwrite(fr, Q[2*W+2]," ",Q[2*W+1:0],"\n");
       j= j+1;
       #30 ;
       enable =1'b0;
 end
    $stop;
 end
endmodule
```

Figure 10: Test Fixture for IIR Low Pass Digital Filter

After the design at RTL level was completed and validated, the low pass filter module was synthesized into gate level netlist using XILINX tools. The Spartan 3E FPGA device was chosen for this purpose. Post-synthesis simulation was aslo performed to validate the functional of the design and check whether or not it meets the timing constraints (e.g. setup and hold times). Simulation was also performed after the place and route stage (P&R). The functionality of the designed was verified at each stage. This concludes the design of the first block in the QRS detector. The remaining blocks were implemented and verified in the same manner.

2.4.2. UART Implementation

The UART is partitioned into two parts, a transmitter and a receiver. For each of which, a verilog description is included in the appendixes. The behavior of these blocks reflects the specifications explained in section 2.3.2. Both expect to send and receive data as 8 data bits, the baud rate used 11500 with a clock of 50MHZ, They also send and receive least significant bit first. They expect to receive at least 1 stop bit and they transmit 2 stop bits. The implementation cycle of the UART was the same as shown in figure 8

2.5. System integration

After all partitions of the design have been individually tested and verified, the whole system was assembled as shown in figure 4. The functionality of the system is tested as follows. The H/S interface, implemented in MATLAB, sends ECG samples to the QRS detector, implemented on Spartan3E FPGA device, through the serial port. The detection signals are sent back through the serial port to be displayed on the screen with the original ECG waveform (as shown in figure 11). The system operated as expected.

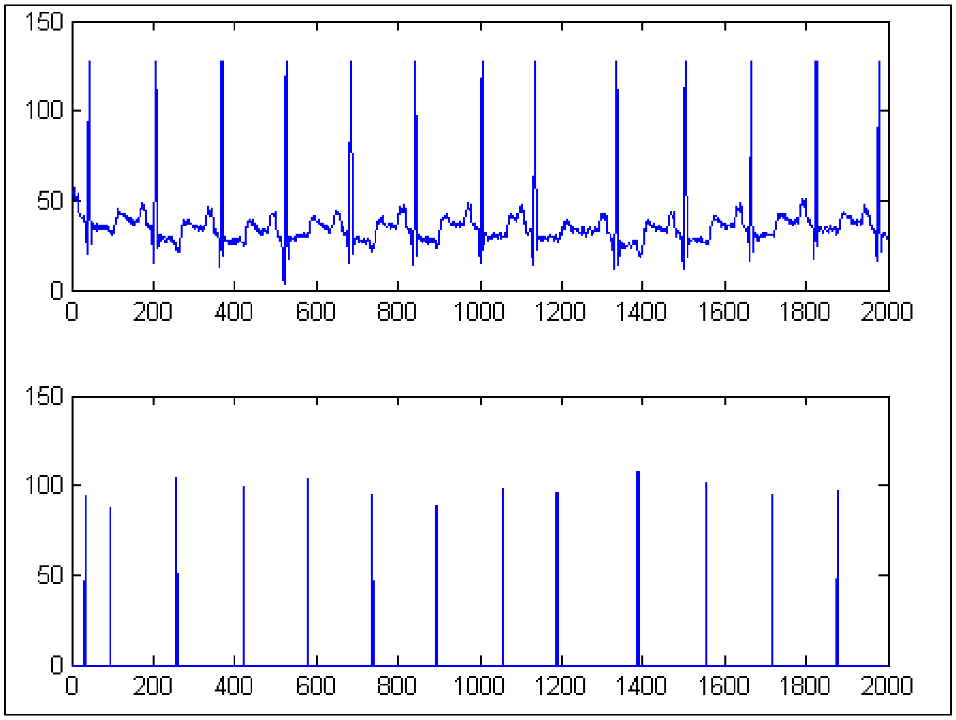


Figure 11: The Outputs of the FPGA Implementation of the QRS Detection Algorithm

3. Design Optimization for Low Power

QRS detection circuits is a indispensable part of wearable heart monitoring devices and pacemakers [1-4], these devices feature more and more complicated functions, they also constantly evolve to meet the increasing requirement of patients. Multiplying functionality means increasing power consumption from the aspect of circuit design. On the other hand, the nature of choric use of such devices places stringent constraints on the amount of power that they can consume. This is mainly due to the fact that the most majority of these devices are battery operated or in some cases implement energy scavenging mechanism [10-13]. There are also additional constraints with regard to power in biomedical implants such as pacemakers; this includes the heat dissipated from using the power. Depending on where in the body the sensor is placed, the allowable amount of heat dissipated varies. Chronic implantation requires much lower dissipation, so as to not damage the tissue surrounding the sensor.

Therefore, finding viable solutions in order stringent power requirements are indispensable for wearable and implantable ECG monitoring devices. This section summarizes the techniques to reduce the power dissipated by the QRS detector throughout its design process.

3.1. Power Consumption of CMOS Circuits

Power dissipation of a CMOS circuit is attributed to two major sources, namely: Leakage, and switching of transistors. The first component is caused by leakage current, which can arise from substrate injection and sub-threshold effects. It is primarily determined by fabrication technology considerations. Switching of transistor can lead to power dissipation into two form; the first is short-circuit which is due to the direct-path short circuit current which arises when both the NMOS and PMOS transistors are simultaneously active, conducting current directly from supply to ground. The second is dynamic power; it is due to the charging of parasitic capacitances at the output of circuit nodes. It is proportional to the load capacitances of circuits' nodes, the operating frequency (i.e. clock frequency) and to the supply voltage. Optimization methods applied in this project mainly target the switching component of the dissipated power, as it is the prominent component in our case.

3.2. Techniques to Reduce Power consumption of QRS Circuits

Low power design consists of a variety of approaches at each level of abstraction i.e. at algorithm level to circuit level, as shown in figure [14]. Each design level has number of low power techniques which may result in considerable reduction of power consumption. To fully manage and optimize the power consumption of any system, the designers have to follow the power optimization methods at each design level.

At the architecture level, one methodology that was used in this project is the optimisation the processing algorithm to simplify computation complexity. In this work the division and multiplication operations were replaced by shift operation where possible. This has helped reduce the total area of the circuit, i.e. the number of transistor hence, power consumption. At the gate level, two techniques were applied, operand isolation and clock gating. The former reduces dynamic power by preventing switching activity when the associated outputs are not used. Operand isolation can by inserting special isolation circuitry at the inputs of the operating units. In this way, unnecessary power consumption caused by redundant operations can be avoided.

Clock gating technique was applied to register banks which share the same clock and enable signal to reduce the clock load. In this condition, the local fan-out of the clock is cut down from numerous register banks to only one clock gating cell. When the enable signal of the register banks is long ineffective and the register number is large enough, which is the case in the QRS detector, the clock power can be dramatically reduced.

At the circuit level (i.e. RTL level), one of the most attractive solutions to reduce power consumption is the use of asynchronous design methodology [15]. Most digital circuits designed and fabricated today are" synchronous." In essence, they are based on two fundamental assumptions that greatly simplify their design, namely: all signals are binary and all components share a common and discrete notion of time, as defined by a clock signal distributed throughout the circuit. Asynchronous circuits are fundamentally different; they also assume binary signals, but there is no common or discrete time. Instead the circuits use handshaking among their components in order to perform the necessary synchronization, communication, and sequencing of operations. This difference gives asynchronous circuits inherent properties of lower power consumption, less emission of electromagnetic interference and robustness toward variations in supply voltage, temperature[16, 17]. Such advantage makes asynchronous design especially attractive for real time implantable biomedical sensors such as pacemakers. This project investigated the use of an asynchronous design technique called pausible clocks [18]. In this approach, each block in the design generates its own clock with a ring oscillator. Each ring oscillator's period is set according to the speed requirements of the block it drives. The local clock of each block is only activated, when the output of the corresponding block is needed, otherwise the clock is stopped which prevents unnecessary consumption of switching power. An example of the implementation of this technique is shown in figure 12, which depicts the current drawn form two silicon chips, both implement cryptographic cores. The upper diagram corresponds to the chip with puasible clocking, the lower corresponds to the chip with a synchronous clock, it can be clearly seen that significant saving can be achieved using plausible clocking techniques.

In the case an ECG system such as shown in figure 2, the QRS block would have its own local clock, which is only activated when receiving ECG samples form the UART. Pausing the clock of the QRS block awaiting communication prevents that block from dissipating dynamic power. VDD can also be lowered during prolonged stalls to reduce static power as well. Additional interface logic is required for each block in the design to implement the local clack and to synchronise with other blocks, however, the area overheads of such interface is small compared to the total area of the system.

To quantify the amount of power saving that can be achieved using this methodology, power consumption figures were compared of a two version of QRS detector, the first is synchronous and the second incorporated with a plausible clock. It was found that up to 25% power reduction can be achieved using this method.

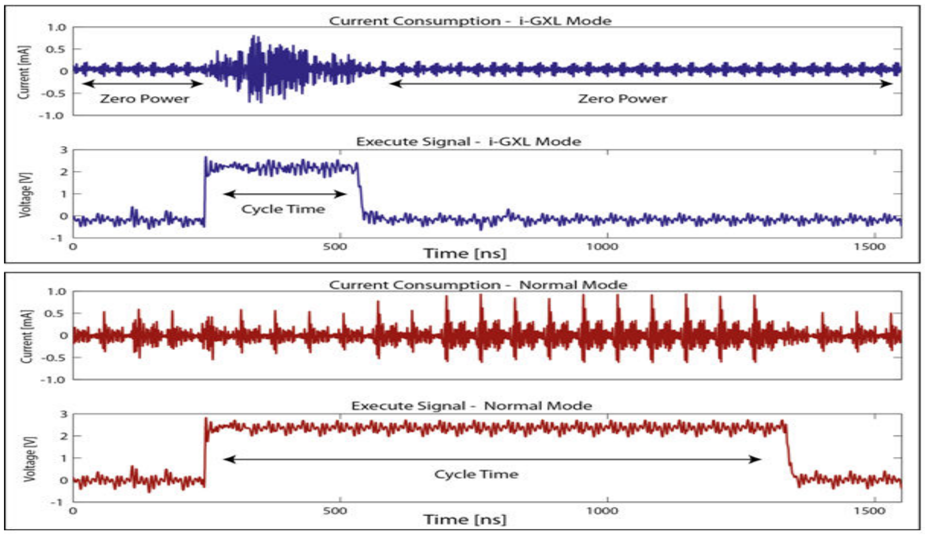


Figure 12: Current Consumption of cryptographic core chips (Pausible clocking(upper), synchronous(lower))

4. Summary

Home monitoring of patients is going to play essential role in reducing the mounting pressure on the health care systems due to the proliferation of chronic diseases. It will help maintain the quality of care for patients and reduce the cost of hospitalization. To make this vision a reality, portable low power diagnostic devices are needed. This paper has described the design and implementation of an electrocardiogram device, a crucial part of diagnostic equipments. The system has been implemented and tested on field programmable array. The stages of development of this system have been discussed in details. Power reduction methodologies have been implemented and validated at different level of the design process.

Acknowledgements

We would like to thank Prof. Mike Cat and Dr Andrew Owens for their invaluable advice and guidance. This work has been supported by EPSRC knowledge transfer grant (RES/0560/7345).

References

[1] Inglis SC., Clark RA, McAlister FA., Ball J., Lewinter C., Cullington D., Stewart S., and C. JGF, "Structured telephone support or telemonitoring programmes for patients with chronic heart failure," The Cochrane Collaboration. JohnWiley & Sons, Ltd., 2010.

[2] U. W. Anliker, J.A.; Lukowicz, P.; Troster, G.; Dolveck, F.; Baer, M.; Keita, F.; Schenker, E.B.; Catarsi, F.; Coluccini, L.; Belardinelli, A.; Shklarski, D.; Alon, M.; Hirt, E.; Schmid, R.; Vuskovic, M., "AMON: a wearable multiparameter medical monitoring and alert system," Information Technology in Biomedicine, IEEE Transactions on vol. 8, pp. 415-427, 2004.

[3] C. G. Bilich, "Bio-Medical Sensing using Ultra Wideband Communications and Radar Technology: A Feasibility Study," Pervasive Health Conference and Workshops, pp. 1-9, 2006.

[4] J. A. Potkay, "Long term, implantable blood pressure monitoring systems," Biomedical Microdevices, vol. 10, pp. 379-392, 2008.

[5] B. S. and N. Dutt, "Efficient search space exploration for HW-SW partitioning" Hardware/Software Codesign and System Synthesis International Conference, pp. 122-127, 2004.

[6] S. Areibi, "Recursive and flat partitioning for VLSI circuit design," Microelectronics International Conference pp. 237-240, 2001.

[7] http://ecg.mit.edu/.

[8] C. H. Bert-Uwe Köhler and R. Orglmeister, "The principles of software QRS detection," Engineering in Medicine and Biology IEEE Magazine, vol. 21, pp. 42-47, 2002.

[9] J. Pan and W.J. Tompkins, "A real-time QRS detection algorithm," IEEE Transection on Biomedical Engineering, vol. 32, pp. 230-236, 1985.

[10] L. T. Ee, B. D. Pereles, B. Horton, R. Shao, M. Zourob, and G. O. Keat, "Implantable biosensors for real-time strain and pressure monitoring," Sensors, vol. 8, pp. 6396-6406, 2008.

[11] P. Cong, W. H. Ko, and D. J. Young, "Wireless implantable blood pressure sensing microsystem design for monitoring of small laboratory animals," Sensors and Materials, vol. 20, pp. 327-340, 2008.

[12] M. Porter, P. Gerrish, L. Tyler, S. Murray, R. Mauriello, F. Soto, G. Phetteplace, and S. Hareland, "Reliability considerations for implantable medical ICs," IEEE International Reliability Physics Symposium, pp. 516-523, 2008.

[13] X. Zou, X. Xu, L. Yao, and Y. Lian, "A 1-V 450-nW Fully Integrated Programmable Biomedical Sensor Interface Chip," Solid-State Circuits, IEEE Journal of, vol. 44, pp. 1067-1077, 2009.

[14] Z. Wang, S. Mai, C. Zhang, and H. Chen, "Design Practice of Power-oriented Integrated Circuits for Biomedical Implant Systems," IEEE International Conference on Electronics, Circuits and Systems, pp. 78-81, 2007.

[15] J. Sparsø, "Asynchronous Circuit Design - A tutorial," Kluwer Acadamic Publishers, 2001.

[16] N. L.S. and S. J., "Designing asynchronous circuits for low power: an IFIR filter bank for a digital hearing aid," Proceedings of the IEEE vol. 87, pp. 268-281, Feb 1999.

[17] A. M. Simonson, Oxenham, A.J., Faltys, M.A., Sarpeshkar, R., "A low-power asynchronous interleaved sampling algorithm for cochlear implants that encodes envelope and phase information," IEEE Transactions on Biomedical Engineering, vol. 54, pp. 138-149, 2007.

[18] P. Teehan, Greenstreet, and G. M. Lemieux, "A Survey and Taxonomy of GALS Design Styles," Design & Test of Computers, IEEE vol. 24, pp. 418-428, 2007.