A Real-Time FPGA-Based QRS Detector Using Adaptive Threshold with the Previous Smallest Peak

El Hassan El Mimouni, Mohammed Karim
University Sidi Mohammed Ben Abdellah
Fes, Morocco
elmihass@gmail.com, karim_lessi@yahoo.fr

ABSTRACT: *The QRS complex of the ECG signal is the reference point for the most ECG applications. In this paper, we aim to describe the design and the implementation of an embedded system for detection of the QRS complexes in real-time. The design is based on the notorious algorithm of Pan & Tompkins, with a novel simple idea for the decision stage of this algorithm. The implementation uses a circuit of the current trend, i.e. the FPGA, and it is developed with the Xilinx design tool, System Generator for DSP. In the authors' view, the specific feature, i.e. authenticity and simplicity of the proposed model, is that the threshold value is updated from the previous smallest peak; in addition, the model is entirely designed simply with MCode blocks. The hardware design is tested with five 30 minutes data records obtained from the MIT-BIH Arrhythmia database. Its accuracy exceeds 96%, knowing that four records among the five represent the worst cases in the database.*

Keywords: Automated Diagnosis, Adaptive thresholding, ECG, FPGA, DSP, QRS detection

Received: 16 June 2012, Revised 2 August 2012, Accepted 4 August 2012

1. Introduction

The electrocardiogram (ECG) signal is one of the most commonly used in medical practice because of its noninvasive nature, simple acquisition process and the information it contains; the analysis of such information permits to evaluate the pathophysiological condition of the heart. Several systems of ECG recordings and analysis have been developed for over a century. The first systems include ECG recording and printing of the signal. Modern systems use computer technology to provide automated diagnosis, contributing to reduce the subjectivity of manual measurements and visual assessments. This is an important area of research and many methods and approaches have been proposed for the detection of ischemia, detection and classification of arrhythmia, diagnosis of the chronic diseases of the myocardium, analysis of Heart Rate Variability (HRV) increasingly studied in the recent years. These studies have for purpose the remove noise and artefacts, the extraction of information characterizing certain pathologies, and analysis of such features for decision making. The analysis is generally based on classical digital signal processing (DSP), wavelet transform, artificial neural networks; it is supported by the opinion of medical experts, who has the responsibility to decide for the final diagnosis.

These systems of analysis are mainly evaluated by using MIT-BIH arrhythmia database; then after, they are introduced into the clinical setting to be validated with the real data [1]-[5].

The present work is within the scope of a system of automated diagnosis of ECG. The block diagram shown in figure 1, illustrates

the operation of such medical data analysis system:

• ***QRS detection:*** By nature, the QRS complex is the most prominent wave in the ECG cycle; so, it is the starting point for almost all the ECG applications. The QRS detection is the subject of this paper,

• ***Features extraction:*** At this stage, accurate recognition of ECG characteristics must be realized, such the different waves in ECG (P, QRS complex and T) and a set of amplitudes and time durations and intervals,

• ***Decision making:*** This process is based on a set of rules (medical knowledge) that operate on values of features recognized in previous stage.

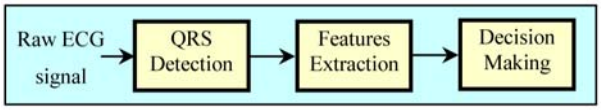

Figure 1. The stages in ECG automated diagnosis

2. QRS Detection

2.1 The Electrocardiogram (ECG)

The ECG signal is the measure, via electrodes acquiring a voltage (potential difference), on the surface of the skin, of the electrical potential generated by the heart's electrical activity. Reading it allows an accurate evaluation of the performances of the heart. As shown in the figure 2, the ECG is characterized mainly by 5 waves reflecting the activity of the heart during a cardiac cycle (R-R Interval); these waves are called by the successive letters of the alphabet P, Q, R, S and T; the Q, R, and S waves are treated as a single composite wave known as the QRS complex [5].

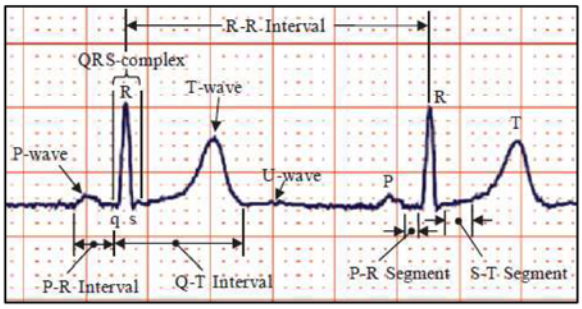

Figure 2. Typical ECG of a healthy person

The placement of the electrodes, voltage measure, in the body determines the viewpoint of the heart. Each viewpoint is called "*lead* ". The most popular clinical standard is the 12-lead ECG, which is made up of the three standard limb leads (I, II and III), three augmented limb leads (aVR, aVL and aVF) and the six precordial leads (V1, V2, V3, V4, V5 and V6) [5]. The figure 3 illustrates the using of the leads I and II.

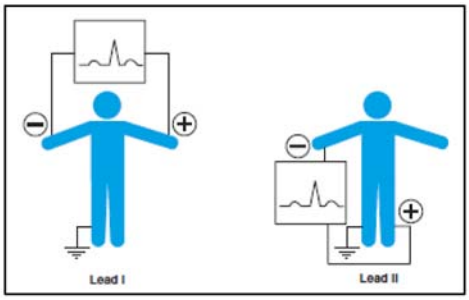

Figure 3. The using of the leads I and II

As we can see in the figure 2, the QRS complex is the most marked feature in the ECG signal, with its narrow shape and its dominant amplitude. It corresponds to the a prominent phase of the cardiac cycle, that is, the depolarization of the right and left ventricles of the human heart; in other words, acquiring the oxygen-enriched blood from the lungs, the ventricles contract and pump blood to the entire body. So, the QRS characteristics such as shape, duration and amplitude provide physicians meaningful information for evaluating the state of heart, particularly the nature of diseases afflicting it. Besides this, the QRS complex has a status of reference point for almost all ECG signal treatments [5].

2.2 The Design Structure

Almost all ECG analysis applications require an accurate detection of the QRS complex. Software QRS detection is research area since more than 30 years and the algorithmic evolution is marked by the developments in the computer sciences [6], [7]. The current trend is FPGA-based designing; indeed, platform FPGAs has become key components for implementing high-performance digital DSP systems; this is because microprocessor-based systems continuously increase in computing power and memory while decreasing in size, cost, and power consumption [4]. According to the authors' knowledge, there are few articles about implementing QRS detection in FPGA, likely due to its relative novelty. References [8]-[10] implement Pan- Tompkins algorithm, while [11]-[13] use the wavelets. For our model, we have adopted the algorithm developed by Pan and Tompkins [14], [15]; it is the most widely used and highly acknowledged algorithm, for its real-time aspect, robustness and efficiency; its structure (Figure 4) is shared by many algorithms in the recent years: the pre-processing stage have for purpose enhancing QRS complex with noise suppressing and artefacts; based on a set of rules, the decision stage determines the assumed QRS candidates. Our work aims to contribute to improve the structure proposed by Shukla [8], [9], with using the MCode blocks of Xilinx System Generator for DSP tool and with novel idea consisting of updating thresholds with the smallest previous detected peak of QRS complex. In fact, in several cases, for example during dynamic exercise, in pericardial effusion, etc., a sudden drop in the amplitude of the R wave is observed; in such cases, if the thresholds are constant, the QRS detection may fail. The adaptive thresholding is used to resolve this difficulty. In the Pan and Tompkins algorithm the thresholds are determined with the mean or median of a specified number of past peaks. We propose adjusting thresholds with the minimum of the last detected peaks.

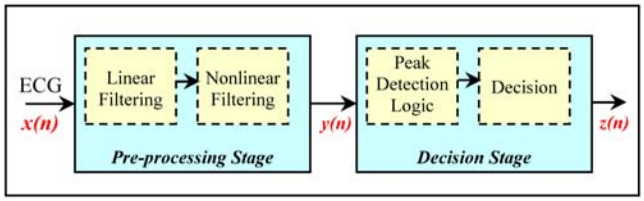

Figure 4. The general QRS detector structure

2.2.1 Pre-processing Stage

The pre-processing stage of the algorithm recognizes QRS complexes based on analysis of the slope, amplitude and width (Figure 5).

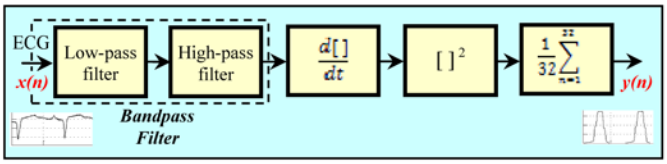

Figure 5. The filter stage of the QRS detector

The shape of the QRS complex (Figure 2) and the power spectrum of the ECG signal (Figure 6) show that the QRS complex has the largest slope; in other words, it has the higher spectrum, centered around 10 Hz.

So, the derivative operation would be the most logical starting point for the algorithm; but the ECG signal is normally subject to several sources of noise: bad contacts and movement of the electrodes, contraction muscles, and 50 Hz or 60 Hz interference power line. Notice that in this context, the P and T waves are considered as a noise; the figure 7 gives an example of 2 cardiac

cycles with tall T wave, where detecting QRS complex with simple thresholding operation is not convenient, because the peak of the T wave is higher than that of the R wave.

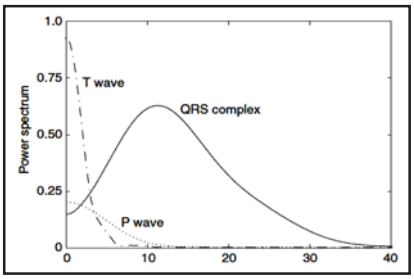

Figure 6. The spectrum of the P wave, QRS complex, and T wave

Thus, this harmonic aspect of the QRS complex and the effect of noise, show the need of the pre-processing stage.

We present, for each pre-processing stage filter, a small description, while giving the transfer function and difference equation:

• **Bandpass filter:** This filter preserving the frequency content of the QRS complex (5 to 15 Hz), is designed from a special class, that requires only integer coefficients, suitable for DSP systems; so, with this approach, the filter consists of cascaded low-pass and high-pass filters.

- The low-pass filter is characterized by:

$$H(z) = \frac{(1 - z^{-6})^2}{(1 - z^{-1})^2} \tag{1}$$

$$y(n) = 2y(n-1) - y(n-2) + x(n) - 2x(n-6) + x(n-12) \tag{2}$$

- The high-pass filter is implemented by subtracting a low-pass filter from all-pass filter with delay (z^{-16}):

$$H(z) = \frac{1}{32} \frac{(1 - z^{-32})}{(1 - z^{-1})} \tag{3}$$

$$p(n) = x(n-16) - \frac{1}{32}[y(n-1) + x(n) - x(n-32)] \tag{4}$$

Figure 7. ECG with tall T wave

• **Derivative filter:** This processing step is differentiation, a standard technique for finding slope information that provides an easier discrimination of QRS complexes among the other ECG waves.

$$H(z) = 0.1(2 + z^{-1} - z^{-3} - 2z^{-4}) \tag{5}$$

$$p(n) = y(n) = \frac{1}{8}[2x(n) + x(n-1) - x(n-3) - 2x(n-4)] \tag{6}$$

• **Squaring function:** This nonlinear transformation serves to make the entire data positive better suited for subsequent operation, i.e., integration.

$$y(n) = [x(n)]^2 \quad (7)$$

• **Moving window integral filter:** This is the latest stage of the process; besides the slope feature highlighted by the previous phases, this stage adds more discrimination effect for the QRS complex, especially for the abnormal complexes. For frequency sampling f_s = 200 Hz, the window width is typically chosen equal to 32. This operation is characterized by:

$$y(n) = \frac{1}{32}\sum_{k=1}^{32} x(n-k) \quad (8)$$

2.2.2 Pre-processing Stage Implementation

For all the implementation, we used the Xilinx System Generator for DSP tool; this provides system modeling and automatic HDL code generation from Simulink and MATLAB [16], [17]. Our implementation is tested using the MIT-BIH database. This contains 48 half-hour excerpts of twochannel ambulatory ECG recordings, obtained from subjects who suffer from various known heart conditions, as well as examples of healthy ECGs. The recordings were digitized at 360 samples per second. The two leads of ECG used are lead I (which gives a morphology similar to lead V5) and lead II. These records have been annotated by clinicians and thus can be used to develop diagnostic software [18]. These valuable resources can be downloaded from the free-access website (physionet.org) [18].

Like in Shukla's work, we used 5 particular data records: 100, 105, 108, 203 and 222. Indeed, Pan and Tompkins report that the record 100 has the best accuracy, while the others were the worst. For each record, we have used data obtained from lead II. These data records were originally sampled at f_s = 360 Hz, while the Pan and Tompkins algorithm has f_s = 200 Hz. Hence the data records are resampled at 200 Hz with the MATLAB function "*resample*". Each record is downloaded from Physionet as data MATLAB file (100m.mat, 105m.mat, etc.). It is resampled at 200 Hz (LeadII_200 = resample(LeadII_360, 200, 360)), saved with the new dimension and used in the Simulink model as workspace by means of the "*from file*" block.

The figure 8 represents the Simulink model of this preprocessing stage that is an exact interpretation of the structure of the figure 4. So, the implementation is a direct transcription, in MCode, of the open source C-Code presented by Tompkins in [4]. Every block encapsulates the corresponding code.

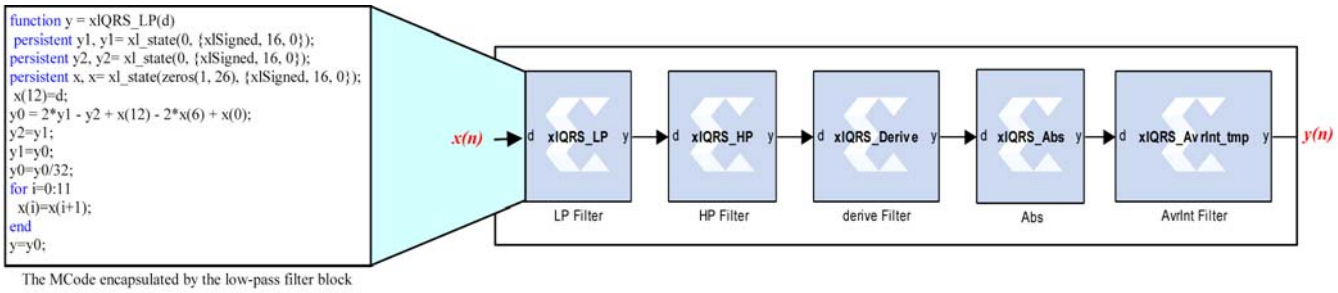

Figure 8. The System Generator pre-processing stage model

For example, we give the corresponding codes for the first and the last blocks; thus, the figures 9 and 10 give, respectively, the C-Code and the MCode for the low-pass filter and the figures 11 and 12 give the same codes for the moving window integral filter. In the MCode, we note particularly the use of the "persistent" MATLAB type equivalent to the "*static*" C. The response of this stage is presented in figure 13 where the output of each filter is shown for a subset of 800 samples of the record 100. Such a response was as expected and widely established in the corresponding literature, particularly in [4].

2.2.3 Decision stage

The peak signal of the moving window integration (MWI) output (figure 13, signal (e)), looks easy to locate by a simple thresholding algorithm; but we could be dealing with some ECGs distorted by noise or by morphological nature due to possible pathologies. Thus, the decision stage consists of 2 phases, peak detection and decision making, using comparison with adaptive thresholds. When the MWI signal crosses a threshold, a flag (QRS Flag) is triggered. The thresholding procedure in the original Pan-Tompkins algorithm adapts to changes in the ECG signal by computing running estimates of signal and noise

peaks; signal peak is defined as the effective QRS complex, while noise peak represents the T wave for example. In this work, we used only one type of peak, that is, the peak of assumed signal; so, the algorithm is based on 3 concepts:

```
int LowPassFilter(int data)
{
static int y1 = 0, y2 = 0, x[26], n = 12;
int y0;
x[n] = x[n + 13] = data;
y0 = (y1 << 1) – y2 + x[n] – (x[n + 6] << 1) + x[n + 12];
y2 = y1;
y1 = y0;
y0 >>= 5;
if(- -n < 0)
n = 12;
return(y0);
}
```

Figure 9. The C-Code for low-pass filter

```
function y = xlQRS_LP(d)
persistent y1, y1= xl_state(0, {xlSigned, 16, 0});
persistent y2, y2 = xl_state(0, {xlSigned, 16, 0});
persistent x, x = xl_state(zeros(1, 26), {xlSigned, 16, 0});
x(12) = d;
y0 = 2*y1 – y2 + x(12) –2*x(6) + x(0);
y2= y1;
y1= y0;
y0 = y0/32;
for i = 0:11
x(i) = x(i  + 1);
end
y = y0;
```

Figure 10. The MCode for low-pass filter

```
int MovingWindowIntegral(int data)
{
static int x[32], ptr = 0;
static long sum = 0;
long ly;
int y;
if(++ptr == 32)
ptr = 0;
sum -= x[ptr];
sum += data;
x[ptr] = data;
ly = sum >> 5;
if(ly > 32400) /*check for register overflow*/
y = 32400;
else
y = (int) ly;
return(y);
}
```

Figure 11. The C-Code for moving window integral filter

```
function y = xlQRS_AvrInt(d)
persistent x, x = xl_state(zeros(1, 32), d);
persistent ly, ly = xl_state(0, {xlUnsigned, 32, 0});
sum = 0;
for i = 0:31
sum = sum + x(i);
end
x.push_front_pop_back(d);
ly = sum + d;
ly = ly/32;
if (ly > 32400)
y = 32400;
else
y = xfix({xlUnsigned, 16, 0},ly);
end
```

Figure 12. The MCode for moving window integral filter

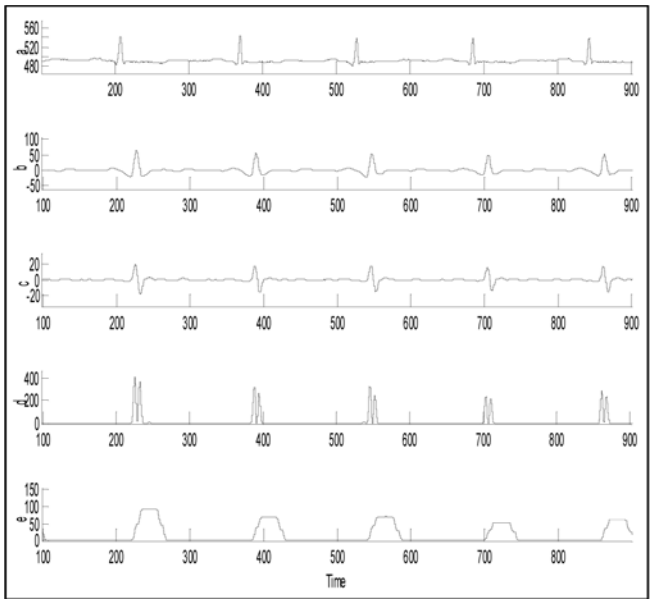

Figure 13. The pre-processing stage response. (a) Original ECG, (b) Bandpass, (c) Derivative, (d) Squaring function (e) Moving Window Integration (MWI)

• Based on the improbability to having 2 QRS so close to each other, the rule of thumb in this case is that the algorithm ignores the assumed noise peak within a delay of 200 ms starting after a detection of a signal peak; this delay equals 40 samples with fS = 200 Hz. The figure 14 illustrates this concept of delay. This implements partially the rule stated in the Hamilton's work, i.e., "*Ignore all peaks that precede or follow larger peaks by less than* 200 *ms*" [19]. During the delay state, noise peak is ignored. So, we can use threshold that is just above the noise peak levels; this improves the detector accuracy.

• The QRS detection is then based on the thresholding principle; we used a concept borrowed from the analog electronics field, that is, the Schmitt trigger, which is defined as a comparator with 2 thresholds possessing memory; the figure 15 shows the structure we have implemented with MCode; th1 and th2 are the trigger thresholds; this avoids to detect falsely multiple peaks due to eventual ripples in the MWI wave peak, as explained by the figure 16. The difference (th1-th2), the trigger hysteresis, determines the noise immunity, that is, its noise rejection ability. Later, in the future work, the fiducial point, that is, the peak of

the R wave will be identified by searching for a maximum within a time interval corresponding to th1 and th2.

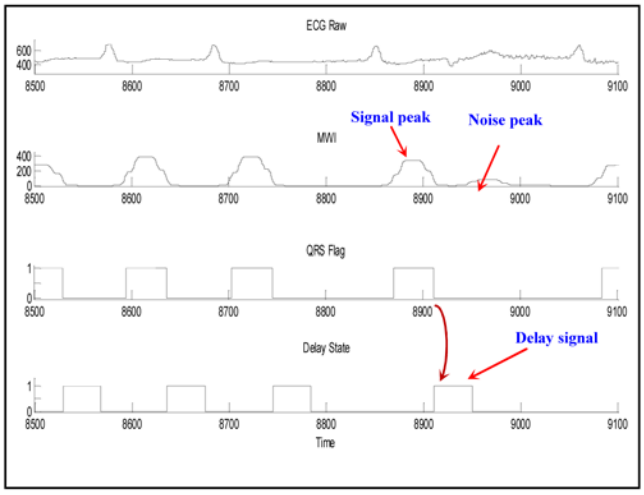

Figure 14. The discarding of the noise peak with delay

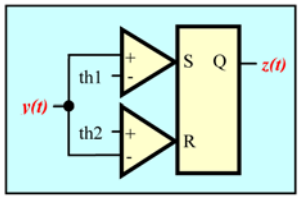

Figure 15. The Schmitt trigger principle

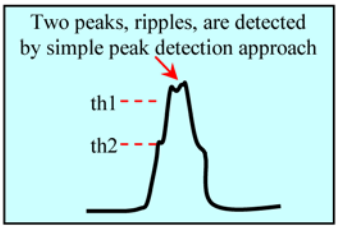

Figure 16. The Schmitt trigger action

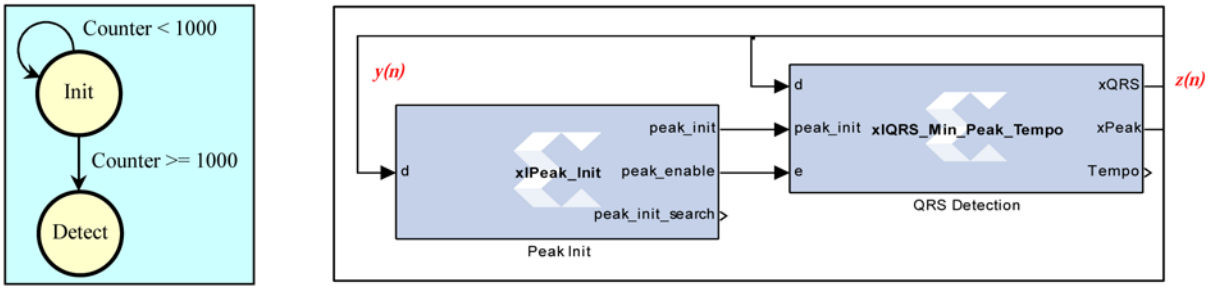

Figure 17. The logic control structure (FSM)

In the Pan and Tompkins algorithm, the thresholds are adapted with the latest peak or with the median of previously detected eight peaks. In our algorithm, "*th*1" and "*th*2" are the trigger thresholds which are updated for every beat from the smallest previous peak; in this way, the algorithm adapts dynamically to changes in the ECG signal from a particular person, improving by this manner once again the accuracy. "*th*1" is chosen sufficiently close to the smaller previous peak (ratio of 0.75), whereas th2 is chosen enough far from zero (ratio of 0.5), because sometimes the MWI signal does not crosses zero; this reduces the false detections due to noise.

2.2.5 Decision Stage Implementation

The figure 17 shows the model of this logic control stage that is a finite state machine (FSM); as shown by this figure, the FSM mainly in 2 MCode blocks representing 2 states of the system: the first one (Init) is a initial state where a starting threshold value is set up; once this threshold is established, the control is never given to it; this diagnostic state takes 1000 samples; the second one (Detect) is the real QRS detector; indeed, this block implements the 3 concepts stated above, that is, the noise peak

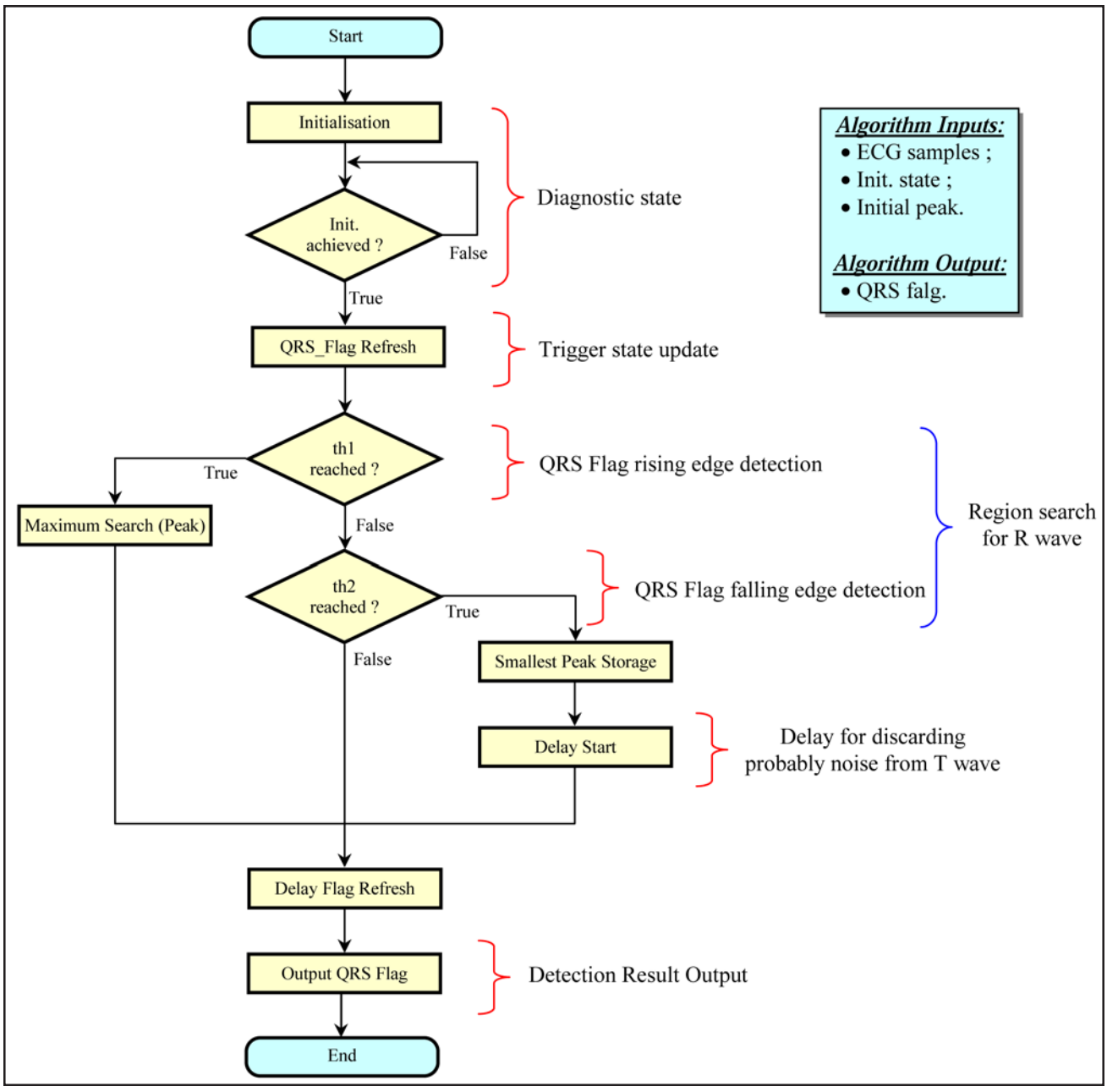

Figure 18. The ASM of the main block of the FSM implementation

discarding with delay, the Schmitt trigger, and the adaptive thresholding with the smallest previous peak. The corresponding algorithmic state machine (ASM) chart of the implementation is shown in figure 18.

For example, we give a MCode sequence relating to the Schmitt trigger function (figure 19); this demonstrates the simplicity of coding adopted along the whole implementation. "*th*1" and "*th*2' are constants and respectively adjusted to 0.75 and 0.5; the variable "*peak*" is the latest and smallest detected R wave value.

The figure 20 represents the complete model of our application.

The figure 21 shows a small episode of the response for the record 100, which is among the cleaner records in the MIT-BIH database, and the figure 22 shows another small episode for the noisy record 108 and among the most difficult to process, where 2 beats are missed (False negative). The figure 23 illustrates with a third sequence the main feature of our algorithm, that is, the

calculation of th1 and th2 based on the smallest previous peak; this decreases the false negative counting and thereby increases the algorithm global accuracy. The figure 24 displays another sequence, with simulated additional noise, that is the 50/60-Hz power line interference, and the baseline wandering.

```
if (d >= th1*peak)
   S = true;
else
   S = false;
end
if (d <= th2*peak)
   R = true;
else
   R = false;
end
  QRS = (S | QRS) & (~R);
```

Figure 19. The MCode Schmitt trigger function

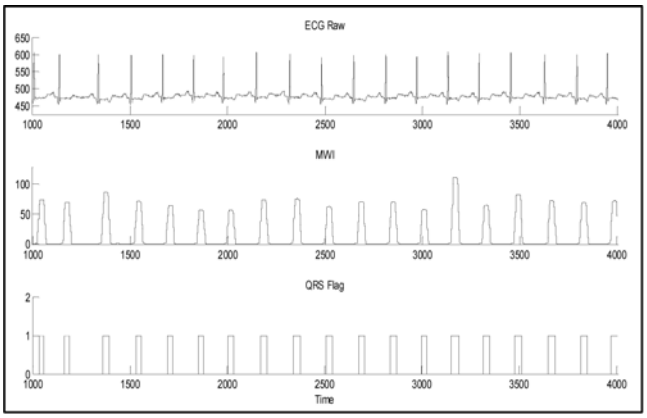

Figure 21. An episode of the record 100

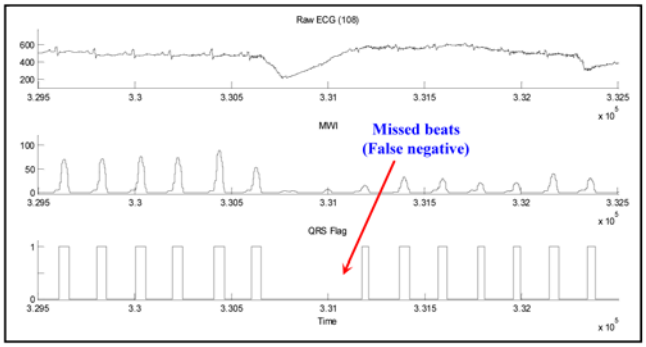

Figure 22. A noisy episode in the record108

3. Evaluation

Due to time constraints, we used mainly a subset of five 30 minutes records of the MIT-BIH arrhythmia database: 100, 105, 108, 203 and 222; the first is the cleaner and the others are clearly noisy; these latter represent the worst case in the Pan and Tompkins results. As described above, the system is implemented in reconfigurable circuit, that is, FPGA representing the strong actual trend for embedded systems designing. Our system is developed in the Xilinx System Generator for DSP, plug-in to Simulink that enables developing high-performance DSP systems for Xilinx FPGAs. The table 1 summarizes the performance of our implementation; the total accuracy exceeds 96%. The evaluation used the following statistical terms:

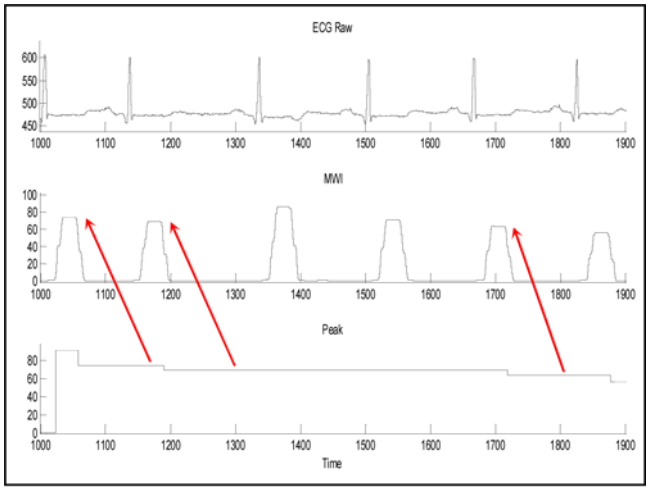

Figure 23. The smallest previous peak following

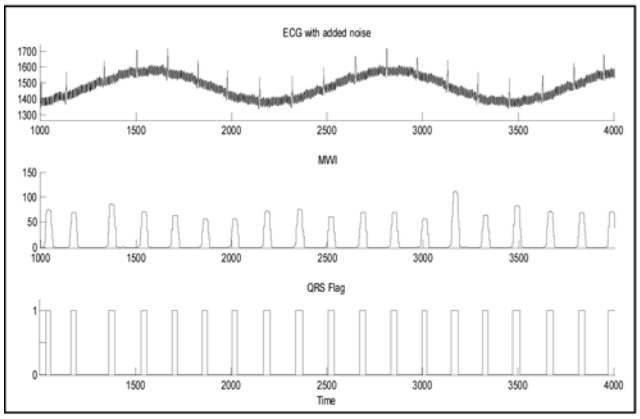

Figure 24. The detector response with additional noise

- False Positive (FP): It represents a peak reported as QRS candidate, while it is none,
- False Negative (FN): It represents a missed peak, when there is in fact a real QRS candidate.

Compared with the works [8]-[13] quoted above, briefly reflecting the state of the art, with emphasing particularly on the good and valuable work of A. Shukla [9], our design is as simple, accurate and efficient as these latter. We could assert also that it brings a little bit more innovation.

The table 2 shows the resources estimation, knowing that we use the Spartan 3e xc3s500e FPGA of Xilinx, that is the core of the development board, Nexys 2 of Digilent [20]. So, this summary reveals that there is enough room for eventual additional functionalities.

The table 3 presents a comparison of our algorithm performances with the well-known original algorithm of Pan and Tompkins, implemented with microprocessorbased system (Z80) and the good work of Shukla implemented with FPGA like in our case.

Due to some difficulties related to Digilent Nexys 2 board JTAG driver, we could not realize, for instant, the hardware co-simulation, which permits implementation verification without need of data acquisition hardware, and hence without risking of hardware loss. But, as good alternative, we have adopted 2 methods for testing our work. In the first method, presented by the figure 25, an excerpt of a MIT-BIH record, one beat corresponding approximately to 200 samples, is loaded in a RAM block and

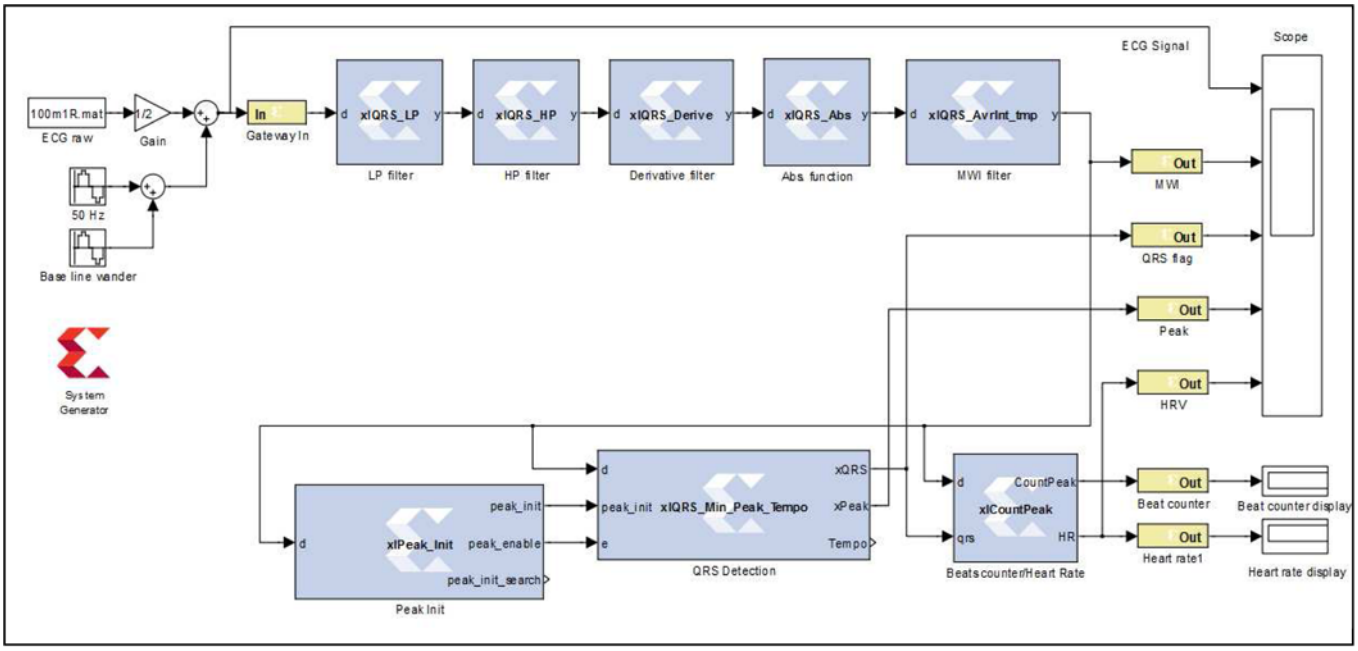

Figure 20. The whole model of the detector

Record (No.)	Total (Nr. Beats)	FP (Beat)	FN (Beat)	Failed detection (%)	Correct detection (%)
100	2266	0	0	0	100
105	2563	26	9	1.37	98.63
108	1757	7	8	0.86	99.14
203	2972	203	2	6.90	93.10
222	2477	162	39	8.12	91.88
Total	***12035***	***398***	***58***	***3.79***	***96.21***

Table 1. Performance of the Implementation

Logic Utilization	Used	Available	Utilization (%)
Number of Slices	2901	4656	28
Number of Slice Flip Flops	2283	9312	14
Number of 4 input LUTs	3443	9312	21
Number of bonded IOBs	18	232	5
Number of MULT18X18SIOs	8	20	40
Number of GCLKs	1	24	4

Table 2. Device Utilization Summary (Estimated Values)

read by the main design at 5 ms sampling interval, corresponding to the sampling frequency of 200 Hz. Thus, a counter addresses the RAM memory which provides the design ECG data raw.

Record (No.)	Original algorithm (%)	Shukla's Implement. (%)	Our Implement. (%)
100	100	100	100
105	96.54	95.60	98.63
108	87.46	91.88	99.14
203	97.22	96.51	93.10
222	92.67	98.95	91.88
Total	***94.78***	***96.59***	***96.21***

Table 3. Performance Comparison

LogicUtilization	Shukla's Implement. (%)	Our Implement. (%)
Number of Slices	76	28
Number of Slice Flip Flops	38	14
Number of 4 input LUTs	54	21
Number of bonded IOBs	1	4
Number of MULT18X18SIOs	45	40

Table 4. FPGA Resources Comparison

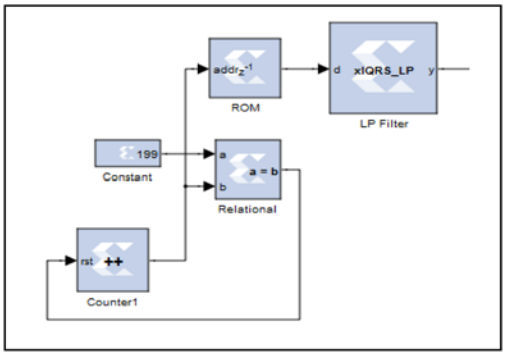

Figure 25. The Design test with ECG raw generated by RAM

The Table 4 presents a comparison with Shukla's work in terms of the resources estimation with the Xilinx FPGA, i.e., Spartan 3e.

In the second method, an excerpt of the same MIT-BIH record, with 10000 samples (50 s), is saved in a text file, read from a simple Visual Basic application and presented to Nexys 2 board, via the parallel port, i.e. LPT1, of a personal computer (PC) at the sampling period mentioned above (5 ms). By convenience, the data is quantized with 8 bits. The figure 26 shows such an experimental setup. The 8-bit output port, having for address 0x378, generates the samples of the ECG to the design, and the eighth bit of the input port, having for address 0x379, receives the QRS flag from the design.

The figure 27 illustrates the global result of the experience with a screenshot, showing the ECG signal sent to the board (top signal) and the QRS flag received from the board (bottom signal). The figure 28 shows an additional module that computes and displays the heart rate (HR). As shown by this figure, this consists mainly of a binary BCD conversion module and a logic control of the multiplexed seven segments display of the Nexys 2 board. The figure 29 presents a captured photo of the

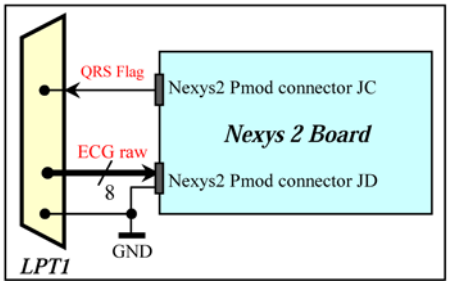

Figure 26. The Design test with ECG raw generated LPT1

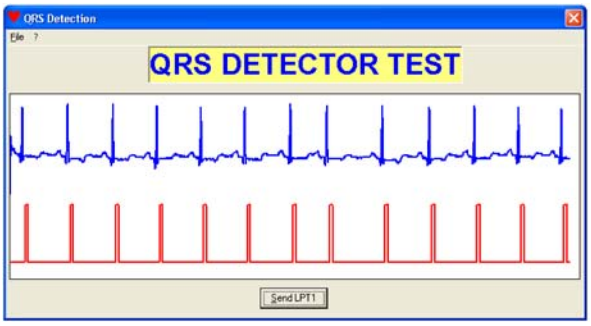

Figure 27. Screenshot of the response of the design fed by LPT1

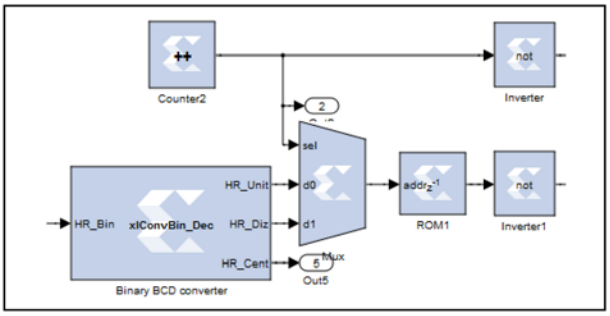

Figure 28. The additional module of heart rate measure

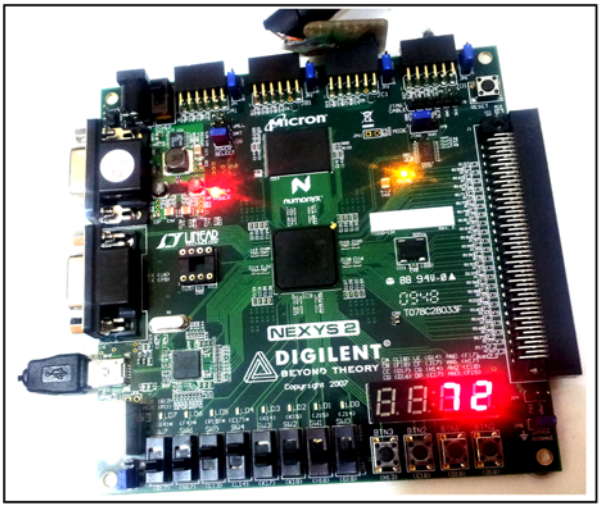

Figure 29. Photo of the design with running program

implementation running the program, with HR equal to 72 bpm (beat per minute) and a blinking LED (LD0 in the board) at measured cardiac rhythm.

4. Conclusions

We have designed a FPGA-based embedded system that implements the most popular algorithm and widely adopted by the patient monitoring industry, that is, the Pan and Tompkins QRS detector algorithm, which is based on slope, amplitude and width information. It was developed in MATLAB/Simulink environment with the Xilinux system Generator for DSP plug-in, using only the MCode block, which provides a convenient and efficient implementing method for algorithmic tasks. The accuracy of the

implementation exceeds 96%, knowing that the tests were done with the worst cases. The design utilizes the parallelism of FPGA; so, the implementation parts are operated in parallel and executed concurrently, making possible a real-time and multitask processing. In this context of parallelism, a first direct application of this QRS detector is the measurement of heart rate, which is ach by the third MCode block of the logic control, same time, allows HRV measure (Figure 14 and figure 28).

The ongoing work is the hardware co-simulation for better verification before real testing with ECG signal acquired from human body; adding to this, the necessity of making the detector more robust by using additional decision rules that obviously requires more algorithmic
complexity.

References

[1] Fotiadis, D. I., Likas, A., Michalis, L., Papaloukas, C. (2006). *Electrocardiogram (ECG): Automated* encyclopedia of biomedical engineering, 2, p. 1259, Wiley.

[2] Sörnmo, L., Laguna, P. (2005). *Bioelectrical Signal Processing in Cardiac and Neurological Applications*, Amsterdam: Elsevier (Academic Press).

[3] Sörnmo, L., Laguna, P. (2006). *Electrocardiogram (ECG) Signal Processing*, Wiley encyclopedia of biomedical, 2, 1298-1313.

[4] Tompkins, W. J., Ed. *Biomedical Digital Signal Processing: C Language Examples and Laboratory Experiments for the IBM P*, Englewood Cliffs, NJ: Prentice

[5] Rajendra Acharya, U., Spaan, A. E., Jasjit Suri, Shankar M. Krishnan. (2007). *Advances in Cardiac Signal Processing*.

[6] Bert-Uwe Köhler, Carsten Hennig, Reinhold Orglmeister. (2002). *Principles of Software QRS Detection,* Society, Jan./ Feb.

[7] Rangaraj M. Rangayyan. (2002). study Approach. IEEE. Press.

[8] Shukla, A., Macchiarulo, L. (2008). *QRS detection System*, 30 Annual International IEEE EMBS Conference,Vancouver, British Co 24,.

[9] Shukla, A. (2008). *Hardware Implementation of Real Time ECG Analysis Algorithms,*M. S. thesis, University of Hawaii at Manoa, Honolulu, Hawaii.

[10] Pavlatos, C., Dimopoulos, A., Manis, G., Papakonstantinou, G. (2005). *Hardware implementation of Pan & Tompkins QRS detection algorithm. In*: Proceedings of the EMBEC'05 Conference, November, Prague, Czech Republic.

[11] Ieong, C. I., Vai, M. I., Mak, P. U. *Complex Detection with Programmable Hardware,*International EMBS Confernce Vancouver, British Columbia, Canada, August 20-24, p. 2920.

[12] Stojanovic, R., Karadaglic, D., Mirkovic, M., Milosevic, D. (2011). FPGA System for QRS Complex Detection Based on Integer Wavelet Transform, *Measurement Science Review*, 11 (4)131-138.

[13] Sasikala, P., Wahidabanu, R. (2010). *Electrocardiogram Using Wavelet Transform* Journal of Advanced Computer Science and Applications IJACSA 1 (6) 48–53.

[14] Pan, J., Tompkins, W. J. (1985). A *Algorithm*., *IEEE Trans. Biomed. Eng.* 236.

[15] Patrick S. Hamilton,Willis J. Tompkins, (1986). Investigation of QRS Detection Rules Using MIT Database, *IEEE Transaction on Biomedical Engineering*, 33 (12), December.

[16] Xilinx System Generator for DSP V12.2, User Guide, July 2010.

[17] Xilinx System Generator for DSP V12.2, Reference Guide, 2010.

[18] http://www.physionet.org/cgi

[19] Hamilton, P. (2002). Open source ECG analysis, *In*: *Cardiology*. Los Alamitos, CA: *IEEE Computer Society Press*, 29, 101–104.

[20] http://digilentinc.com/