QRS DETECTION USING FPGA ACCELERATION IN ELECTROCARDIOGRAMS

LUBOMIR BOGDANOV* AND NIKOLAY KADIEV

Department of Electronics, Technical University of Sofia
„St. Kliment Ohridski" № 8 Blvd., 1000, Sofia, Bulgaria
e-mails: lbogdanov@tu-sofia.bg, nkadiev@tu-sofia.bg

Abstract. The following paper presents a comparison of four methods for QRS detection in electrocardiograms with the help of a field programmable gate array (FPGA). The first method uses a hard-wired microprocessor and external DRAM, while the second, third, and fourth methods use custom IP blocks that are hand-written and automatically generated, respectively. The hardware description language is chosen to be Verilog, and the controlling firmware is written in C algorithmic language.

Keywords: ECG detection; embedded systems; FPGA; high-level synthesis; power consumption.

1. INTRODUCTION

The electrocardiogram (ECG) monitoring in patients is one of the most important procedures in modern-day medicine. It presents a graph of the voltages produced by the human heart while beating. These voltages are sensed by electrodes attached to the skin, and the expected amplitudes are in the millivolt range (up to 3.0 mV). An analog front-end is usually connected in series and aims at amplifying the signal to a scale that is suitable for the subsequent analog-to-digital conversion (ADC). The sample rate is low, in the order of 250 – 360 samples per second.

A typical ECG signal is shown in Fig. 1. Six distinctive regions can be seen: QRS complex representing the ventricular contraction, P-wave – depolarization of the atrium, T-wave – repolarization of the muscle, U-wave – cell recovery before the next beat period. The information carried by the signal is distorted by external interference sources [1] such as the power line in the hospital's rooms, devices using radio for communication (such as Wi-Fi and Bluetooth-enabled devices [2]), various medical equipment, etc. To filter the unwanted noise, hardware or software methods could be used. The filtered signal is then supplied to the QRS detection hardware or, if no

*Corresponding author
http://doi.org/10.7546/EngSci.LXII.25.04.03

specialized accelerators are used, a software algorithm. The output of the processing is simply a binary array containing true and false values for each ADC sample.

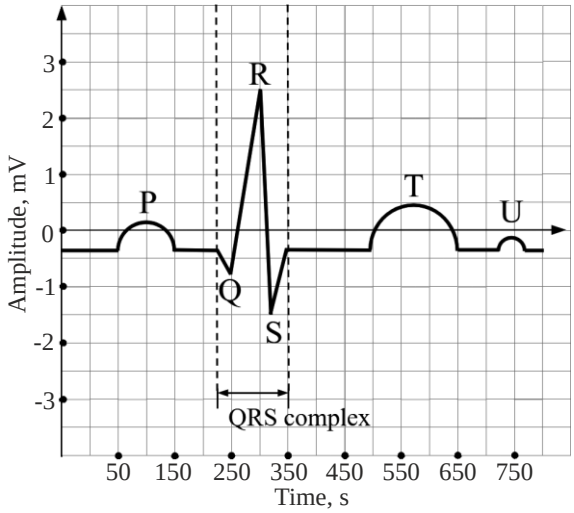

Fig. 1. Typical ECG signal with its phases.

A standard approach for processing the ECG signal includes a microcontroller with an integrated or external ADC module. The processing resource of any modern, mid-range microcontroller is enough to cope with a single-channel ECG. However, in multi-channel measurements, the device may slow down its response due to the higher workload. The problem exacerbates further if multiple patients have to be monitored by a single apparatus. So, to speed up the calculations, an FPGA-based designs are presented in the paper. This type of chip is intended for streaming applications that have to perform multiple parallel operations on many data arrays [3, 4].

A simplified block diagram of the selected FPGA is given in Fig. 2. It contains a hard-wired microprocessor ARM Cortex A9 with dual cores. Its maximum operating frequency depends on the FPGA version and, for the current experiments, is 667 MHz. The FPGA chip can be divided into two regions – a processing system (PS) containing the CPU along with various peripherals (UARTs, timers, SPI, ADC, DAC, etc) and a programmable logic (PL) containing complex logic blocks (called slices) where the Verilog functions are implemented.

The FPGA chip cannot be used alone, and it requires external DRAM and a flash chip/SD card to operate correctly. The DRAM is of type DDR3 with 1 GB capacity. It stores the firmware for the hard-wired processor or for the soft cores (if there are any in the design). The flash memory is a QSPI memory with a capacity of 16 MB and is used for loading the PL configuration (derived from the Verilog description).

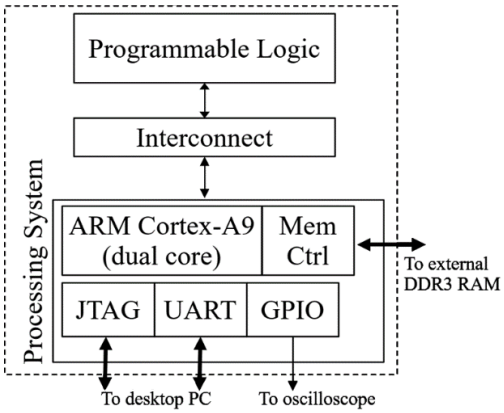

Fig. 2. Simplified internal block diagram of Xilinx XC7Z010 FPGA (used modules only).

2. ECG ALGORITHM

The ECG detection algorithm is tested with the steps shown in Fig. 3. It consists of two parts: a digital filter to remove the power line noise and the DC offset [5, 6] and a QRS detector [7].

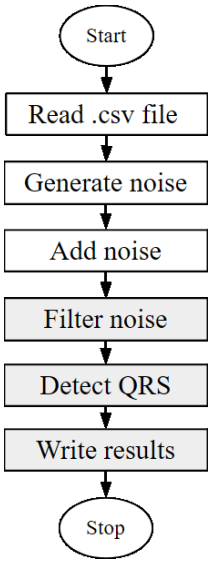

Fig. 3. Steps performed to test the QRS detection algorithm.

For the purpose of this research, the MIT-BIH database with reference ECG samples has been used [8]. To work with this database, a software package has to be installed on Linux (assuming Linux Ubuntu) – the WFDB package. The respective package dependencies can be installed with the command:

─ apt install gcc libcurl4-openssl-dev libexpat1-dev

Then, a single file from the database is chosen randomly; in this case, it is the "200.dat" file. This is a binary file that has to be converted to a text file. For convenience, a CSV file format has been chosen. The conversion can be done with the command:

─ rdsamp -c -r mitdb/200 > 200.csv

The sample ECG graph is taken with an 11-bit A/D converter with a bipolar input range of +/- 5 mV. The sample values are in the range of 0 – 2047 (this gives a ~4.8 μV bit weight) and, therefore, the zero voltage is around 1023. The sample rate of the A/D is 360 samples per second (360 Hz). The mains frequency has to be known in advance, and in the "200.dat" file, it is 60 Hz.

The second step of the algorithm includes the development of the actual QRS detection. To do this, initially, the program is written in C on a personal computer. Only integer variables are used in it, so that it can be easily portable to embedded systems. Once it has been verified that it is operating correctly, then it can be ported to the ARM Cortex-A9 processor and, later on, can be developed into a Verilog accelerator. The chosen tools are all open-source – Eclipse is the integrated development environment (IDE), and GCC is the toolchain.

To test the power line interference removal filter, a sinusoidal signal with a frequency of 60 Hz and a large amplitude is superimposed on the MIT reference signal using the formula:

$$PL(T) = \sin\left(2\pi \frac{f_{mains}}{f_{ecg}} T\right).N_{ADC} \tag{1}$$

where PL is the desired mains signal, f_{ecg} is the sample rate of the A/D, N_{ADC} is the resolution of the A/D, T is the A/D sample rate, f_{mains} is the frequency of the power line. Then, a superposition is done by adding the points from the unwanted power noise signal PL(T) to the ECG(T) reference signal:

$$ECG_{PL}(T) = ECG(T) + PL(T) \tag{2}$$

A thorough description of the ECG signal denoising is given in [5]. The filter is named "DxN" and is implemented by using the moving average of N signal samples with a window of D number of samples. It has a comb-frequency response characteristic with a high-pass cut-off frequency defined by N. Its zeroes are defined by the integer ratio of the sample frequency F_S divided by D. To calculate the D and N coefficients, the following formulae are used:

$$D = \left(\frac{f_{ecg}}{f_{mains}}\right) \tag{3}$$

$$N = \frac{3}{4}\left(\frac{f_{mains}}{f_{3db}}\right) \tag{4}$$

Hence, if $D = 6$ and $N = 42$, a cut-off frequency of approximately 1 Hz can be achieved.

The QRS complex detection is described in detail in [9]. It uses an adaptive amplitude and slope criteria for R-peak recognition. Other algorithms are presented in [10, 11].

3. THE PARTIAL VERILOG IMPLEMENTATION

Once the C code has been verified on a desktop computer, the next step is to port it to the selected Xilinx FPGA. The easiest solution is tested first - use the embedded ARM Cortex-A9 application processor without any Verilog accelerating blocks. For this case, the input data of 1000 samples is converted to a C header file that, in turn, is made a part of the firmware project. The need for doing so is the fact that the project does not support a file system. A custom desktop program is developed for this purpose, again written in C, that reads a file, in CSV format, with samples and outputs a text in the terminal through the STDOUT stream. This text is formatted in such a way that it complies with the ANSI C standard for declaring an array. It is later copied to a file with the ".h" extension. To measure the execution time of the program, a GPIO module has been used. Additionally, a UART module prints the output data array, so that the correctness of the operation can be verified

The second solution is to add separate intellectual property (IP) blocks, written in Verilog, that would substitute parts of the C code.

An averaging function that has been used by the QRS detection is given below. Since this function is invoked twice in the C code, the Verilog IP blocks are instantiated twice. The IP block contains a control register, an input register, and an output register:

```
always @ (posedge clk)
begin
        if(control_reg[0] == 32'h01)
        begin
                sum <= input_reg[0] + … + input_reg[3];
                output_reg <= sum >> 2;
                status_reg[0] = 32'h01;
        end
end
```

The resulting design is shown in Fig. 4. The processor frequency is kept at its default value of 100 MHz, and all measurements are done with it.

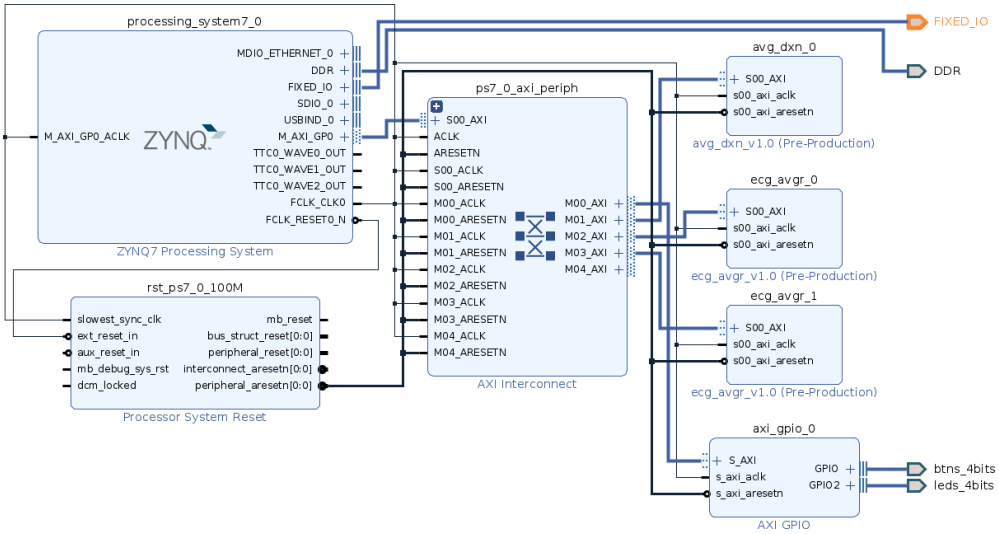

Fig. 4. Block diagram of the partial QRS Verilog implementation.

4. THE SOFT-CORE VERILOG IMPLEMENTATION

An alternative approach to the Cortex-only and the partial Verilog accelerator designs is to use a multiprocessor system where each processor calculates different ranges of the ECG input graph. At the end, all output arrays are merged into a single one. This multi-threaded method is also known as array partitioning. The firmware for the "DxN" and the QRS detection is present on all cores. A block diagram of the design is shown in Fig. 5.

The "DxN" and the QRS detection are present on all cores. A block diagram of the design is shown in Fig. 5 Seven cores in total are able to fit on the chosen FPGA. Their configuration is set to a "microcontroller" mode that saves chip resources. The ARM Cortex-A9 is used to initialize and start the processing of all soft cores. When the processing from all cores has finished, the Cortex asserts a GPIO. By using the UART, the output array (residing in external RAM) is transferred to a desktop computer for verification.

To conform with the previous experiment, the MicroBlaze cores are clocked at 100 MHz. They use 8 control registers and an output register file. Each register's purpose is described below:

– START – a 32-bit register containing a flag that starts the processing of the ECG samples. The initial value is zero. While it is in a zero state, all of the MicroBlaze

cores poll this register for changes. Whenever it changes, the processors start executing their algorithm.

- MB0_FINISH – MB6_FINISH – these seven registers are 32-bit and contain the finish flags of the cores 0 to 6. They are all checked by the ARM Cortex-A9 to note the end of the processing.
- ECG_PEAKS – a thousand 32-bit registers that hold the output values. For each input value, there is a single output value that holds a binary flag.

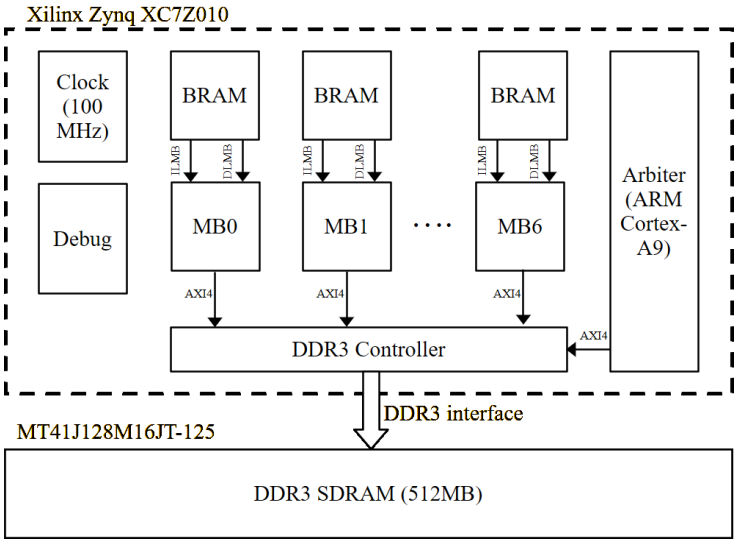

Fig. 5. Multiprocessor parallelization of a QRS algorithm using seven MicroBlaze (MB) soft cores.

5. HIGH-LEVEL SYNTHESIS OF A VERILOG ACCELERATOR

A final option is presented in this section – a single Verilog accelerator. The help of Xilinx Vitis High-Level Synthesis (HLS) has been used [12]. The IDE provides the ability to describe a system as a black box or otherwise - to program its behavior without going into details. The description language to do that is C, C++ or OpenCL. The output language is Verilog or VHDL. However, in order for this "translation" to be correct, certain rules have to be followed. Xilinx recommends the following:

- No dynamic memory allocation – the FPGA designs are fixed and cannot be dynamically extended;
- No file input/output – there is no file system and all of the data has to be received over input/output ports;
- No OS functions – common system API functions like Linux or Windows, getting and setting the time are non-existent.

– No frequency scaling – internal circuits are synchronous, and when a new design is generated, clock uncertainty is taken into account only for a single frequency.

A block diagram of the proposed system is shown in Fig. 6. The accelerator is clocked at 100 MHz, so that it is comparable to the other designs. The input ECG samples are read from the external DDR RAM and, after processing, are written back to that same memory but at a different offset address. The IP core uses local (on-chip) memory (BRAM) to implement the local variables of the C program. So, even though the external RAM is virtually limitless, the complexity of the design is limited by the FPGA's internal BRAM cells.

6. EXPERIMENTAL RESULTS

The firmware and the respective bit-stream files are loaded to the Xilinx FPGA. Note that, for comparison, a low-power microcontroller, NRF52832, has also been added to the experiment. It has an ARM Cortex-M4 microprocessor that operates at 64 MHz (maximum clock rate for this chip). The execution time is measured by toggling a GPIO pin connected to an oscilloscope. The API functions that set and reset the GPIO pin contribute to a 380-nanosecond additive time error. Since most measurements are in the millisecond range, this error is neglected.

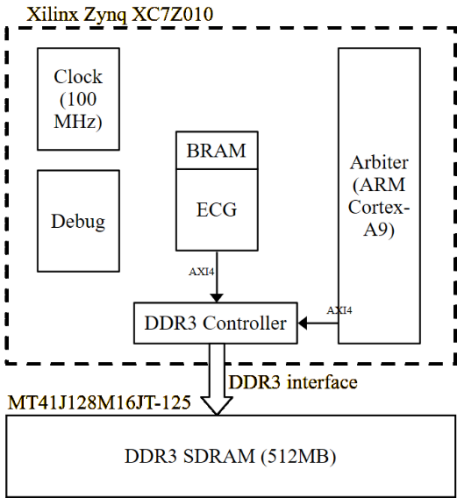

Fig. 6. Vitis HLS-generated Verilog accelerator for QRS detection.

The best result yielded the single ARM Cortex A9 core – the processing of 1000 samples took only 1.516 ms. The results are summarized in Table 1 and presented graphically in Fig. 7. The multi-processor design with the MicroBlaze (abbreviated as MB) showed an optimum at a number of cores equal to 5. A greater number of

cores slowed the processing. The reason for this could be the common memory for writing the output data – it creates a bottleneck and could be optimized in the future. The custom IP design, generated by Vitis HLS, is the second-best solution with 1.550 ms.

Table 1. Time of processing 1000 ECG samples

Design type	Number of cores	Texec, ms
ARM Cortex-M4 (NRF52832)	1	31.5
ARM Cortex-A9 processor	1	1.516
MicroBlaze	1	28.8
MicroBlaze	2	14.6
MicroBlaze	3	11.4
MicroBlaze	4	11.2
MicroBlaze	5	6.1
MicroBlaze	6	6.4
MicroBlaze	7	6.96
Cortex-A9 + Custom IP	1	8.652
Custom IP	-	1.55

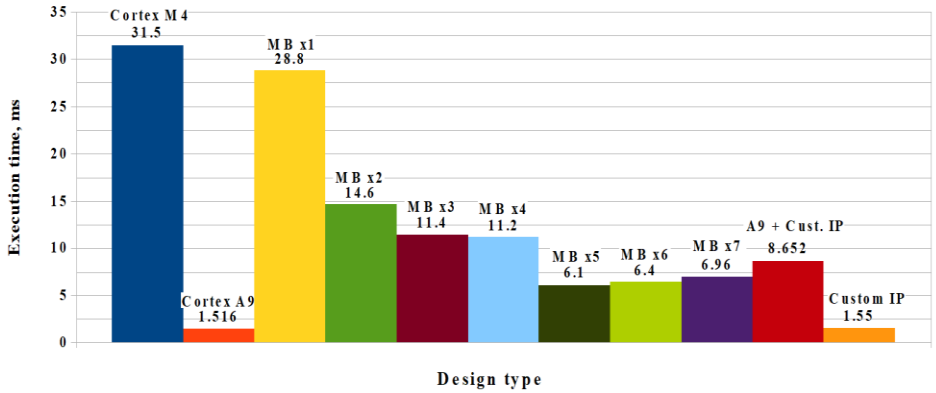

Fig. 7. Execution time of different QRS detector designs.

The multi-processor design with MicroBlaze soft cores is evaluated in terms of energy consumption. To do this, a shunt resistor with a high-bandwidth differential amplifier is connected in series to the power supply of the Zybo demo board. The output of the amplifier is connected to an oscilloscope. The energy is calculated for

the 5-volt power supply and includes all chips on the demo board, including regulators, DDR3 RAM, a debugger, and LEDs. The results are given as a graph in Fig. 8. It can be seen that the optimal, in terms of execution time, 5-core (MB x5) design also yields the best energy results. This can be explained by the fact that the faster the processors do the calculations, the smaller the time they spend in active mode consuming high currents. And vice-versa: the 1-core (MB x1) takes more time to do the calculations, hence it spends more time-consuming high currents.

7. CONCLUSION

The research in this paper presents a comparison of 10 hardware FPGA systems and a single low-power microcontroller system for QRS detection in ECG signals. The FPGA designs include a single embedded processor, a single embedded processor and three custom IP blocks, a multi-processor soft core system with 1 to 7 cores, and a single custom IP. The systems use the same software algorithm and operate at the same clock frequency. The best results were yielded by the single embedded processor and the single custom IP designs. This proves that for benchtop ECG devices, it is better to use hardware acceleration, given that the device is mains-supplied.

Future development of the studies may include a deeper analysis of the multi-processor system by adding multi-port RAM memory to prevent the bottleneck of a single memory interface. This, in theory, should decrease the execution time significantly.

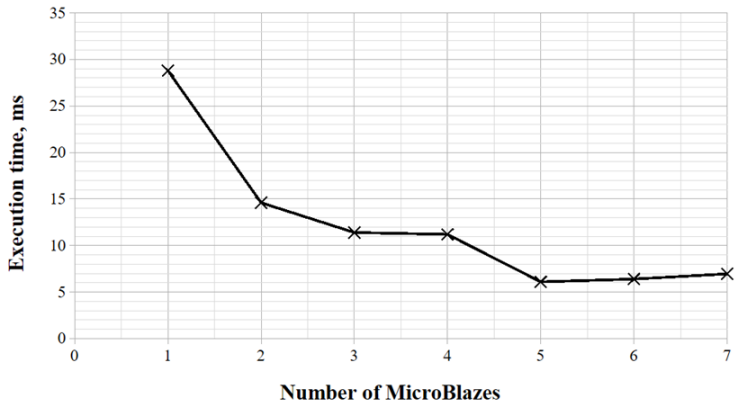

Fig. 8. Multiprocessor's energy consumption versus number of MicroBlaze cores.

Another improvement may include a pipelined ECG detection with a custom Verilog IP core where separate hardware stages perform different calculations. The final period will be shortened due to the better utilization of these stages.

When the best hardware/software combination is found, in-field tests with real patients should be performed to verify the effectiveness of the proposed method. This will require a logging system that is to be developed. Its responsiveness will be critical for the experiment, so a fast-computing device must be considered.

ACKNOWLEDGMENT

The authors would like to thank the "Research and Development" sector at the Technical University of Sofia for the financial support under project 242ПД0004-03.

REFERENCES

[1] G. BRINDHA AND J. MANJULA, FPGA-Based ECG Signal Analysis for Arrhythmia Detection System Using SVM Classifier, in: Proc. International conference on biomedical engineering and computing technologies (ICBECT'22), 21–25 March 2022, **2603** (1), online: https://doi.org/10.1063/5.0126540.

[2] L. KUMARI, Y. SAI, N. BALAJI AND K. VISWADA, FPGA-Based Arrhythmia Detection (2015) *Procedia Computer Science*, **57** 970-979, online: DOI: 10.1016/j.procs.2015.07.495.

[3] MADHAV P. DESAI, G. CAFFARENA, R. JEVTIC AND ET. AL., A Low-Latency, Low-Power FPGA Implementation of ECG Signal Characterization Using Hermite Polynomials (2021) *Electronics – Special Issue FPGA/GPU Acceleration of Biomedical Engineering Applications*, **10** (19), 2324, online: DOI: 10.3390/electronics10192324.

[4] MOHAMED G. EGILA, MAGDY A. EL-MOURSY, ADEL E. EL-HENNAWY AND ET. AL., FPGA-Based Electrocardiography (ECG) Signal Analysis System Using Least-Square Linear Phase Finite Impulse Response (FIR) Filter (2016) *Journal of Electrical Systems and Information Technology*, **3** 513–526, online: DOI: 10.1016/j.jesit.2015.07.001.

[5] I. ILIEV, S. TABAKOV AND V. KRASTEVA, Combined High-Pass and Power-Line Interference Rejecter Filter for ECG Signal Processing (2008) *Proceedings of the Technical University – Sofia*, **58** (2) 7-13, ISSN 1311-0829.

[6] D. DOBREV, T. NEYCHEVA AND N. MUDROV, Simple High-Q Comb Filter for Mains Interference and Baseline Drift Suppression (2009) *Annual Journal of Electronics*, **3** 50-52, ISSN 1313-1842.

[7] V. KRASTEVA AND I. ILIEV, Automatic Analysis and Visualization of Multilead Long-Term ECG Recordings (2007) *Proceedings of the Technical University of Sofia,* **57** (2) 106-112, ISSN 1311-0829.

[8] G. MOODY AND R. MARK, The Impact of the MIT-BIH Arrhythmia Database (2001) *IEEE Engineering in Medicine and Biology Magazine*, **20** (3) 45-50, PMID: 11446209, online: DOI: 10.1109/51.932724.

[9] I. ILIEV, V. KRASTEVA AND S. TABAKOV, Real-Time Detection of Pathological Cardiac Events in the Electrocardiogram (2007) *Physiological Measurement*, **28**, 259-276, online: DOI: 10.1088/0967-3334/28/3/003.

[10] L. KUMARI, Y. SAI, N. BALAJI AND K. VISWADA, FPGA-Based Arrhythmia Detection, in: 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015), 970-979, online: DOI: 10.1016/j.procs.2015.07.495.
[11] F. KARATAŞ, İ. KOYUNCU, M. TUNA AND ET. AL., Design and Implementation of Arrhythmic ECG Signals for Biomedical Engineering Applications on FPGA (2022) *Eur. Phys. J. Spec. Top*, **231** 869-884, online: DOI: 10.1140/epjs/s11734-021-00334-3.
[12] VITIS HIGH-LEVEL SYNTHESIS USER GUIDE, UG1399, AMD Adaptive Computing, v2023.2, 2023.

Received October 4, 2025