REAL-TIME VISUALIZATION OF MULTICHANNEL ECG SIGNALS USING THE PARALLEL CPU THREADS

Peter Kaľavský, Milan Tyšler

Institute of Measurement Science, Slovak Academy of Sciences, Bratislava, Slovak Republic

Abstract

In this paper the concept of real-time visualization of multichannel ECG signals is introduced. The visualization of more than one hundred ECG signals per screen is achieved through the parallel execution of two CPU threads. The main thread of the application handles the GUI activity and the visualization of processed ECG signals, the worker thread is responsible for handling the data acquisition, the dataflow formatting and computation of ECG leads and processes the data for ECG signal visualization. For proper reconstruction of the signal shape in real-time a peak detection algorithm is used. The application software is written in a cross-platform application and UI framework named Qt. From the parallel execution point of view the application software uses task-parallelism. For the inter-thread communication the Qt event system together with queued signals and slots mechanisms is used.

Keywords

multichannel ECG, real-time visualization, parallel CPU threads, real-time signal processing, body surface potential mapping

Introduction

Body surface potential (BSP) mapping is a non-invasive electrocardiographic (ECG) method enabling more precise diagnostics of cardiac diseases than the commonly used 12-lead ECG. It is widely accepted that 12-lead ECG may not be optimal for diagnostic assessment of certain cardiac diseases like acute cardiac ischemia or myocardial infarction since the coverage of the standard precordial leads over the thorax is limited. BSP, on the other side, enables detailed registration of surface cardiac potentials using tens to hundreds of sensing electrodes and consequent ECG signal analysis offers much more information on the physiological state of the heart [1]. Moreover, many studies proved that BSP maps together with information on torso geometry obtained from imaging techniques such as MRI or CT can be used for more advanced diagnostic methods based on inverse solutions and enable non-invasive model based assessment of abnormal electrical sources in the heart [2].

However, the processing of a large number of measured ECG signals imposes increased performance requirements on the computing system. The growing number of processed signals not only increases the computing demands on the basic ECG signal processing chain (see Fig. 1), but also complicates the possibility to implement advanced methods for ECG signal processing in real-time. The mentioned obstacles could be overcome by the use of parallel processor systems and parallel programming models. If it is possible and suitable, the traditional single threaded application can be parallelized in order to exploit the processors capabilities of current heterogeneous CPU-GPU based systems [3].

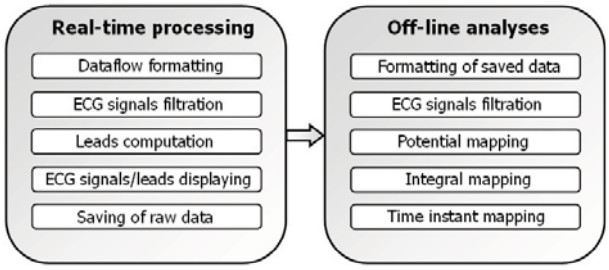

Fig. 1: Block diagram showing basic program modules of the BSP mapping system.

In this paper possible concept of the real-time multichannel ECG signal processing and visualization

together with the application software supporting parallel execution of two CPU threads is introduced.

Methods

Real-time visualization of ECG signals

The visualization of ECG signals plays important role mainly during the heart rhythm monitoring of patients with critical diseases. Such monitoring requires real-time ECG signal processing. The term "real-time processing" can be used either to identify the processing of each individual signal sample before the next signal sample will arrive, or to identify the processing of more samples from one signal episode, like a heart beat, before the samples from the next signal episode are at disposal [4]. If we consider sampling frequencies used in high-resolution electrocardiography and the resolution of common PC monitors, the visualization of several heart beats on the screen requires a compression of acquired samples. Thus the second above mentioned real-time processing approach should be used. To ensure proper reconstruction of the signal shape in real-time without repeated plotting of redundant points or discontinuities in signal tracings an appropriate method has to be chosen. The best results could be achieved by the use of sinx/x interpolation [5], however this method is applicable only in situations where the time is not critical.

Our application utilizes the method that is similar to the method for peak detection used in some digital oscilloscopes (Tektronics or LeCroy, [6]). The method determines the maximum and minimum value among the samples from one acquisition interval Δt and then displays the values and all the values between them to one vertical line that has the width of one pixel on the display (see Fig. 2). In this way the compression of the time axis is achieved without missing any extremes that might occur during the acquisition interval [7].

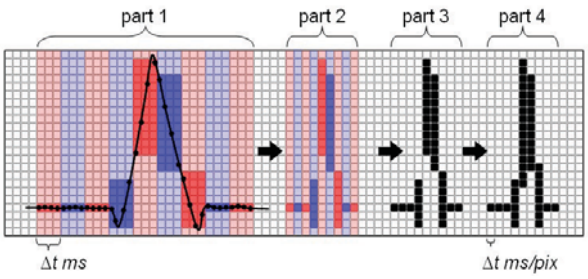

Fig. 2 Peak detection and time compression during signal visualization.

From Fig. 2 another problem of the visualization is apparent. Rapid changes of signal amplitudes can lead to broken tracing lines on the screen as shown in Fig. 2 - part 3. This occurs usually on the onset and offset of the R wave of the ECG curve. To solve this problem, connections between vertical lines of previous and following processed intervals of the visualized signal need to be added as depicted in the part 4 of the same figure.

The value of Δt has to be selected with respect to the character of the displayed signal. The shortest component of the ECG signal is represented by the Q wave with the duration of 30 ms (see Tab. 1). In order to ensure reliable detection of all ECG waves and to eliminate their merging, the value of Δt should not be greater than 30 ms.

Tab. 1: Durations of ECG waves.

ECG curve component	Boundary points	Duration [ms]
P wave	P_o - P_e	60 - 110
PQ segment	P_e - Q_o	60 - 90
PQ interval	P_o - Q_o	120 - 200
Q wave	Q_o - Ro	< 30
QR interval	Q_o - R	< 30 - 50
QRS complex	Q_o - S_e	< 100
T wave	T_o - T_e	< 160
ST segment	S_e - T_o	< 120
QT interval	Q_o - T_e	200 - 400
RR interval	R - R	830 (at f_s = 72/min)

With respect to the above mentioned method that requires at least two samples (minimal and maximal) for signal shape reconstruction, another factor influencing Δt is the sampling frequency used during the ECG monitoring. Our multichannel mapping system ProCardio8 enables to set sampling frequency from 125 to 2000 Hz. It implies that the value of Δt needed to acquire two samples lies in the range between 16 and1 ms. For our application we have decided to define common acquisition interval for all possible sampling frequencies, thus we have selected Δt = 16 ms. Value of Δt defines also the time base rate for visualization of ECG signals during the monitoring.

When very large number of signals is recorded simultaneously, the multichannel ECG monitoring should enable also visualization of lower number of ECG signals on the screen suitable for human perception. For this reason our application uses the concept of virtual screens (see Fig. 3). Each individual virtual screen can visualize only an n-tuple of measured signals and during ECG monitoring it is possible to switch between individual virtual screens.

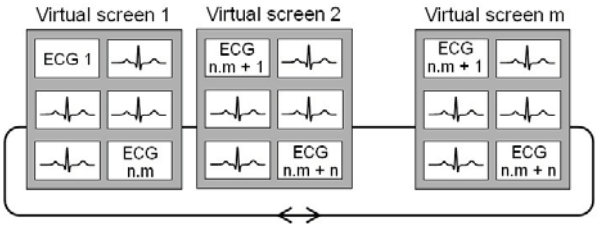

Fig. 3: Concept of virtual screens.

To further speed up the signal displaying, the time consuming complete erasing of the virtual screen is avoided during the signal monitoring, if possible. Thus the application draws only additional vertical lines of pixels corresponding to the actual acquisition interval Δt, without redrawing the whole virtual screen.

Parallel processing

The current trend in CPU design moves towards several cores. A typical single-threaded application can exploit only one core at a time. However, a program with multiple threads can be assigned to multiple cores, making threads running in a concurrent way. As a result, the distribution of the work to more than one thread can make the program running much faster on multicore CPUs [3, 8].

The basic idea of writing a parallel program is the partitioning of the work among several cores. There are two widely used approaches: task-parallelism and data-parallelism. In task-parallelism, various tasks carried out to solve the problem are partitioned among the cores. In data-parallelism, data used in solving the problem are partitioned among the cores and each core carries out more or less similar operations on its part of the data [3, 8].

When the cores can work independently, writing a parallel program is much the same as writing a serial program. Things get more complex when the cores need to coordinate their work. The coordination usually involves communication among the cores, load balancing, and synchronization of the cores.

Application software

Since the real-time processing and visualization of multichannel ECG signals is computationally highly demanding, we proposed an application taking the advantage of multicore CPUs. The application software is written in Qt – a cross-platform application and user interface framework for developers using C++.

The application software uses the task-parallelism. It consists of the main thread and the worker thread (see Fig. 4). The main thread handles the activity of the graphical user interface (GUI) and thus also the visualization of processed ECG signals. The worker thread is used to offload the data processing work from the main thread. It is responsible for managing data acquisition, dataflow formatting, leads computation and the processing of data for ECG visualization.

For the inter-thread communication, the Qt's event system together with queued signals and slots mechanisms is used. The signal is a method that is emitted rather than executed when called. The slot is a member function that is called as a result of signal emission [9].

Every thread has its own event loop. It waits in the event loop for a specific event to occur. To place an invocation in an event loop, a queued signal-slot connection is made. Queued connection type determines that the slot is executed in the receiver's thread when the control returns to the event loop of the receiver's thread. In other words, to call a slot in another thread, that call is placed in the receiver's thread event loop. This lets the receiver thread to finish its current task before the slot starts running, while the sender thread continues running in parallel. Such approach enables to obtain the worker thread's results without blocking the main thread run.

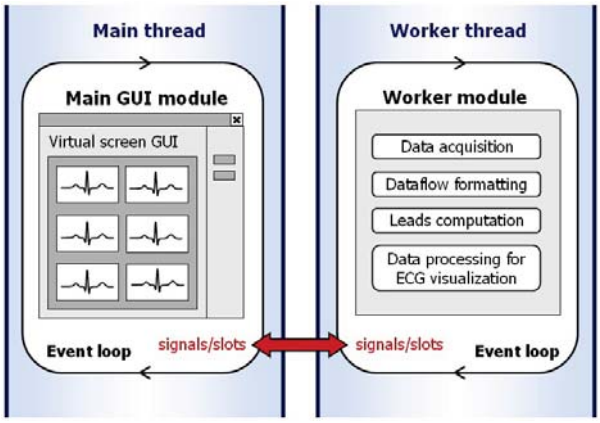

Fig. 4: Parallel threads and program modules of the proposed application.

For the sake of simplicity, we shall consider only two programming modules that will be distributed among the threads: the main GUI module and the worker module (see Fig. 4). The main GUI module incorporates all GUI modules including the virtual screen GUI. The worker module integrates all additional modules used in the application.

Results

When our application supported only one execution thread, the main GUI module and the worker module had to be processed within the timeout interval of 16 ms. (A timer with a timeout interval of 16 ms periodically forced the thread event loop to acquire and process new data). When we gradually increased the number of measured and visualized signals, we reached

a point when the sum of the time needed to update the main GUI module and the execution time of the worker module exceeded the set timeout interval. Such behaviour naturally led to the overflow of the device circular buffer with acquired data samples.

After the implementation of two threads, the worker thread offloaded the processing work from the main thread and contributed to keeping the main thread GUI responsive. The execution of the worker module has become independent from the main GUI module updates, what helped to stabilize the periodicity of the data acquisition and the data processing. In addition, an idle time interval has also arisen in the worker thread

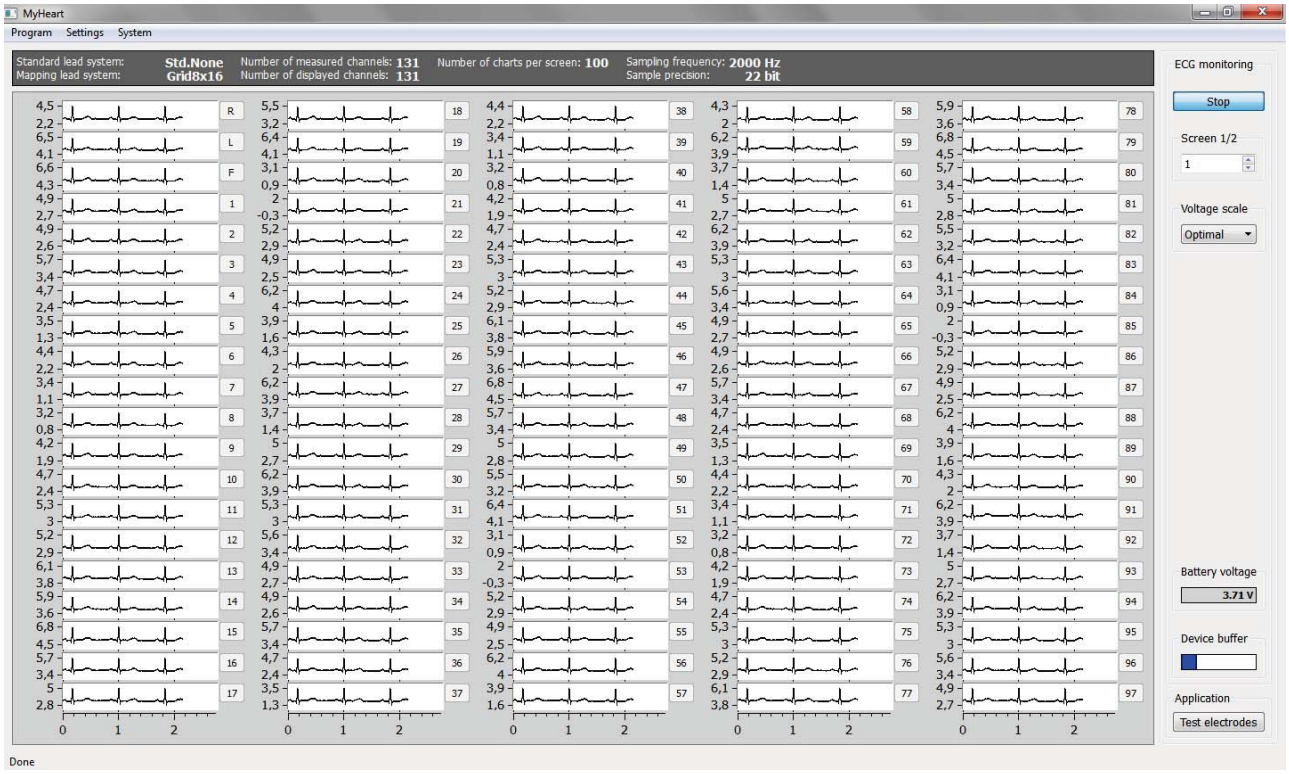

Fig. 5: The main window of the designed application software.

event loop. This idle time interval can be used for implementation of additional program modules into the worker module. In an ideal case both, the runtime of the main GUI module and also the runtime of the worker module should not exceed the timer timeout interval. Anyway, if the number of visualized ECG signals per virtual screen increases, the time needed to manage and update corresponding pixels in signal windows on the screen will also increase and can exceed the timer timeout interval. The most critical part represents the switching between virtual screens, when the erasing of the whole virtual screen is necessary. However, thanks to the queued signal/slot connections our application enables continuous execution of the worker module even if there is a delay in the main GUI module.

The extra samples prepared for ECG visualization in the worker thread are simply stored in the main thread event queue for later dispatching. Queued ECG samples are consequently visualized within a few tenths of milliseconds, without noticeable perception of the user.

Regarding the maximal data throughput of the ProCardio8 mapping system, (up to 144 measured channels, 2000 Hz sampling frequency and 22-bit sample resolution), it is possible to reliably display up to one hundred ECG signals per one virtual screen (see Fig. 5).

Discussion

Parallel hardware became ubiquitous in today's computers. It's difficult to find a laptop, desktop, or server that doesn't use a multicore processor. In situation of stagnating single processor capabilities and increasing parallelism, a growing audience of scientists and engineers is concerned with performance and scalability.

The proposed application software takes the advantage of multiprocessor machines. Currently only two execution threads are implemented. However, the presented approach enables to add additional parallel

threads, if necessary. Likewise, we can use the task-parallelism not only to distribute independent program modules between different threads, but also to support parallel execution of tasks inside individual program modules. Because the processing of multichannel ECG signals shows considerable data-parallelism, besides the task-parallelism, also data-parallel algorithms can be utilized in the application, e.g. for parallel multichannel ECG signal filtration [10], or for the BSP map computation and visualization [11]. Thanks to the current heterogeneous CPU-GPU based system and more friendly software support for the general purpose computations on GPUs, the data parallel task can be now effectively processed directly on the GPU. It is possible to exploit the different capabilities of mutli-core CPUs and many-cores GPUs according to their suitability and computational demands. The real-time processing and visualization of multichannel ECG offers a great space for further investigation of its possible improvement by the use of the parallel programming approach.

Acknowledgement

The work was supported by research grant 2/0131/13 from the VEGA Grant Agency and by grant APVV-0513-10 from the Slovak Research and Development Agency.

References

[1] Tyšler, M. et all. *Non-invasive Assessment of Local Myocardium Repolarization Changes using High Resolution Surface ECG Mapping*. Physiological Research, 2007, vol. 56, suppl 1, S133-S141.

[2] Hanninen, H. et all. *ST-T integral and T-wave amplitude in detection of exercise-induced myocardial ischemia evaluated with body surface potential mapping.* J. of Electrocardiology, 2003, 36, 89.

[3] Pacheco, S. P. *An Introduction to Parallel Programming.* Morgan Kaufmann Publishers, Burlington, 2011. ISBN 978-0-12-374260-5.

[4] Rangayyan, R. M. *Biomedical signal analysis: a case study aproach.* Wiley-IEEE Press, New York, 2002. ISBN 0-471-20811-6.

[5] Havlík, L. *Oscilloscopes and their using.* Sdelovaci technika, Praha, 2002. ISBN 80-901936-8-4.

[6] *Tektronix Acquisition Modes.*, 2004, [online], [2014/05/09], http://anlage.umd.edu/.

[7] Karas, S., Tyšler, M. *Matlab Software for High-Resolution Multichannel ECG Measurement.* Instrumentation for the ICT Era. Proceedings of the 17th Symposium IMEKO TC 4, Košice, Slovak Republic, 8.-10. 9. 2010, p. 586–590.

[8] Hager, G., Wellein, G. *Introduction to High Performance Computing for Scientiststst and Engineers.* CRC Press, Boca Raton, 2011. ISBN 978-1-4398-1192-4.

[9] Thelin, J. *Foundations of Qt Development.* Apress, New York, 2007. ISBN 978-1-159059-831-3.

[10] Kaľavský, P., Tyšler, M. *Real-time processing of multichannel ECG signals using graphic processing units.* Clinician and Technology, 2012, vol. 42, no. 2, p. 27–30.

[11] Muzik, J. et all. *ProCardio8 – the 8th generation of the high resolution ECG mapping system.* XIX IMEKO World Congress, Lisbon, Portugal, 6.-11.9.2009, p. 1689-1694.

Ing. Peter Kaľavský
Department of Biomeasurements
Institute of Measurement Science
Slovak Academy of Sciences
Dúbravská cesta 9, 841 04 Bratislava

E-mail: peter.kalavsky@savba.sk
Phone: +421 259 104 551