Optimal Parallel Wavelet ECG Signal Processing

Ervin Domazet
Ss. Cyril and Methodius University
Faculty of Computer Science and Engineering
1000, Skopje, Macedonia
Email: ervin_domazet@hotmail.com

Marjan Gusev
Ss. Cyril and Methodius University
Faculty of Computer Science and Engineering
1000, Skopje, Macedonia
Email: marjan.gushev@finki.ukim.mk

***Abstract*—Real time detection of heart abnormalities can prevent serious health problems. This requires real time processing of ECG data by a corresponding web service. Considering the case of wearable devices to collect ECG data, the signal is actually contaminated by noise. Noise can seriously change the ECG signal and occur in the form of a baseline drift representing various physical movements and breathing. Unless it is removed, correct analysis on ECG data is impossible. Being characterized by very low frequencies, its elimination can not be efficiently realized by simple DSP filters, such as Finite Response Filters (FIR) or Infinite Response Filters (IIR).**

Wavelet Transformation is a promising technique to eliminate the noise with very low frequencies, and its digital version (DWT) is capable of efficient removing the ECG baseline drift. In this paper, we set a research question to investigate the dependence between the nodes in the DWT implementation (and therefore to their corresponding threads) and the available number of cores that can execute the code. This analysis leads to valuable conclusions that will allow construction of even better optimizations. Results indicate that proper allocation of cores can yield faster code.

***Index Terms*—Wavelet Transformation, ECG, Heart Signal, Parallelization, OpenMP**

I. Introduction

Information and Communication Technologies (ICT) is an emerging field, which stimulates innovative solutions in the domain of healthcare. In this paper, we analyze solutions based on wearable Electrocardiogram (ECG) sensors that continuously stream data to the server and huge data quantities are being processed by a corresponding web service [1].

It is scientifically proven that the detection of heart abnormalities can prevent serious health problems [2], [3]. This requires real time processing of ECG data by a corresponding web service. Considering huge data quantities coming in a certain velocity, optimization is inevitable.

The pre-processing phase in processing of ECG signals is mainly responsible to eliminate the noise stemming from different sources and DSP filters are primary used tools.

Noise can seriously change the ECG signal and occur in the form of a baseline drift representing various physical movements and breathing. Unless the noise is removed, correct analysis on ECG data is impossible. This noise is characterized by very low frequencies, and its elimination can not be efficiently realized by simple DSP filters, such as Finite Response Filters (FIR) or Infinite Response Filters (IIR). Wavelet Transformation is a promising technique to eliminate the noise with very low frequencies, and its digital version (DWT) is capable of efficient catching and removing the ECG baseline drift. Additionally, DWT is also used in Feature Space Reduction phase in order to locate the QRS characteristics of the ECG signal.

Milcheski and Gusev [4] propose a new version of DWT implementation using a circular buffer and obtain a significant speedup. In our previous study [5], we have optimized this implementation of the DWT algorithm by optimizing the Initialization part for additional 20% faster code using OpenMP. However, the problem of synchronization between different iterations prevents even higher speedup.

In this paper, we set a research question to investigate the dependence between the nodes in the DWT implementation (and therefore to their corresponding threads) and the available number of cores that can execute the code. This analysis leads to valuable conclusions that will allow construction of even better optimizations. We give a detailed analysis and also realize experimental testing to analyze the practical implementations. Evaluation of the results are compared with the results of previously parallel code.

The paper is organized as follows, Section II presents background information and shortly the previous study. In Section III, the optimization approaches are listed. Section IV gives the details about conducted tests. Evaluation and discussion regarding the results are presented in Section V. Related work is given in Section VI and, lastly, in Section VII, the paper is concluded with future considerations.

II. Background

ECG holds vital information related to the cardiovascular condition of a living person. This section gives a general overview of ECG signal processing and briefly explains Wavelet Transformation and its importance in ECG feature reduction and extraction.

A. ECG signal processing

Methodologies for processing and analyzing ECG signal consist of three stages: data pre-processing, feature space reduction and feature extraction [3].

DSP filters are generally used in the data pre-processing phase. Low pass filters are usually used to eliminate the noise with high frequencies, such as the electrical switching and radio waves. High pass filters eliminate the noise initiated

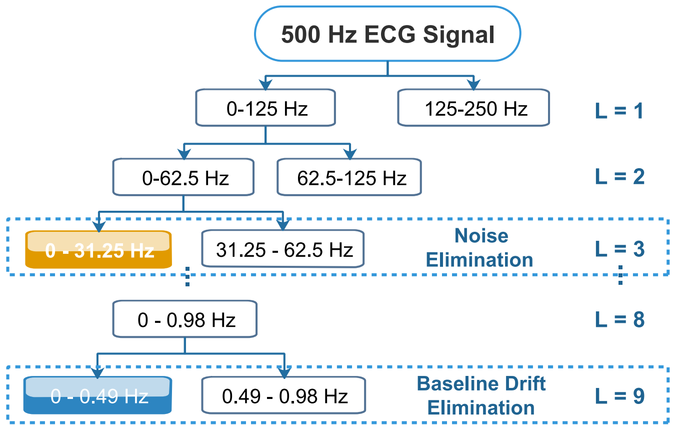

Fig. 1. Wavelet decomposition tree. High pass filter is used to eliminate baseline drift and low pass filter for noise elimination.

by physical movement and breathing, mainly interpreted as baseline drift elimination. Bandpass filters, as a combination of high pass and low pass filters are considered as effective DSP tools for noise elimination. Although, DSP filters eliminate the noise to a certain extent, they provide a relatively clear signal, which can be further processed for feature extraction.

In the feature space reduction phase, the signal is analyzed by detecting the peaks of QRS complexes and locating the peaks of individual P and T waves. A QRS complex is used as the starting point for further analysis, and, therefore, it's exact detection is of a high importance [6]. For example, a Wavelet transformation can be used for baseline drift elimination in this stage. In the final phase, QRS features are extracted, and the ECG signal precisely characterized.

The quality of extracted features, is directly dependent on the correct rate of eliminated baseline drift. Thus, focusing on this step is vital.

Digital filtering is essential for both the first two steps of the ECG signal processing. Wavelet Transformation is an efficient method used in both the elimination of baseline drift and QRS complex extraction.

B. *Baseline drift and noise removal of an ECG signal*

An important fact about Wavelet Transform is that the number of iterations needed to decompose the signal into smaller frequency bands is higher for smaller bands. In the case of ECG, a smaller number of iterations (and therefore time) are needed to analyze higher frequencies, and the opposite for lower frequencies. To eliminate the baseline drift one needs to deal with very low frequency bands and filter the lower frequency components.

An example of an implementation of a DWT algorithm that eliminates the baseline drift of an ECG signal is given next.

In case of a ECG sensor functioning at a 500Hz sampling frequency, it is known that spectrum of a real valued signal is symmetric [7]. Since the symmetric part is the mirror image of the first half, it does not provide additional information.

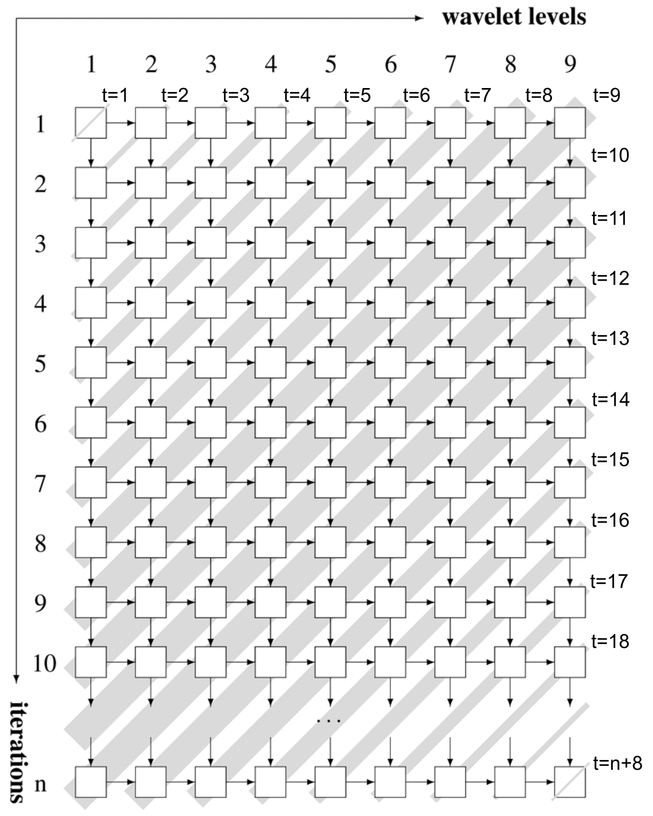

Fig. 2. Simultaneous Execution of nodes on the Processing phase of DWT code.

Thus, working on a 250 Hz frequency is enough to extract data from the signal.

DWT is based on decomposition, and reconstruction of the signal [4]. In each step, a signal is decomposed into high pass and low pass coefficients, from which approximation and detail coefficients are calculated. In order to eliminate the baseline drift of 0.5 Hz, it requires at least $L = 9$ wavelet levels. Noise can also be canceled with an additional $L = 3$ wavelet levels and a delay of 6. Details are presented on our previous work [5].

C. *Discrete Wavelet Transform Analysis*

Our analysis on the previous study [5], showed that DWT algorithm contains two bottlenecks, exposed in the *Initialization* and the *Processing* phase.

Observation is that the former phase does not include data dependencies between iterations. This was a vital information for pure parallelization. Though, the latter phase is highly dependent, preventing direct parallelization. This is presented in Fig. 2 where data dependence is vizualised as A $\rightarrow$ B, with the meaning *B depends on A*.

D. *Previous Parallel Algorithm*

Our previous parallel algorithm [5] was based on optimizing both of the bottlenecks. The *Initialization* phase was parallelized by a straightforward approach. Nevertheless, high data dependency on the *Processing* phase required re-arrangement on the nodes for a concurrent computation.

Computation waves can flow with 45 degrees to axes where each wave contains independent computations and can be executed simultaneously at a given time stamp. This ensures that previous nodes (found on the left) are already calculated. Due to this pipelined structure, the first output will be ready after L iterations, where L is the number of wavelet levels.

The proposed implementation requires that each block of independent nodes to be synchronized between iterations. However, this is a costly operation and prevents theoretical speedup of L, when executed on L cores.

Next section gives further optimization strategies, in order to achieve the best efficiency through the parallel algorithm.

III. Optimization Approaches

The methodology for testing the parallel algorithm on the previous study [5] was based on executing both the bottlenecks on the same number of cores.

The algorithmic and storage complexity of the DWT is $O(L{*}2^L)$, making it nearly hard to increase the Wavelet levels.

One interesting approach is to keep the core numbers for *Initialization* phase high. This would increase the efficiency, simply because the data independent iterations.

In the *Processing* phase the maximum available nodes that can concurrently be processed is restricted to the number of wavelet levels. Thus, increasing the core numbers, will only increase the number of idle cores. However, executing this region with less number of cores can decrease the burden of barrier synchronization.

Our previous work did not address the effect of filter length. Theoretically, increasing the filter length will directly increase the percentage of processing compared to the percentage required to synchronize iterations.

Moreover, OpenMP provides built in optimization strategies [8]. Previous study did not considered using them. It would be interesting to test their effect on the barrier synchronization.

IV. Testing Methodology

Let the response time required to process the parallel algorithm be denoted by T_p, and the response time required to process the optimized parallel algorithm with p cores, be denoted by T_{op}. Then, the speedup is defined as the ratio of the execution times by (1).

$$S_{OP} = \frac{T_p}{T_{op}} \tag{1}$$

The proposed optimization approaches are tested on an Amazon C3 c3.8xlarge instance. It consists of a high-frequency Intel Xeon E5-2680 v2 (Ivy Bridge) Processor with 32 cores, 60GB of memory. OpenMP library is used for testing the proposed optimizations.

The following optimisation approaches will be tested:

OA1: Using more core numbers for Initialization phase.
OA2: Using less core numbers for Processing phase.
OA3: Increasing the filter length.
OA4: Using compiler optimizations.
OA5: Combined Effect.

On the previous study, we observed the effect of input size is negligible as the wavelet levels increase. Due to this, the input size will be fixated to 10.000 sample length ECG signal. Throughout the tests, wavelet levels vary from 3 up to 24, with incremental steps of 1 level.

On the test environment, the maximum number of available cores is 32. Considering this, the test configuration is presented in Table I.

TABLE I. Test Environment Setup

Optimization Approach	Description of The Testing Methodology
OA1	Core numbers from 2 to 30, incremental steps of 2
OA2	Core numbers from 2 to 10, incremental steps of 2
OA3	Daubechies filters of length 4, 8, 16 , 32 and 64
OA4	OpenMP's built-in O1, O2 and O3 optimizations
OA5	Combination of the most efficient approaches

Each test case was tested ten times and an average value of measured times was calculated and used for further processing. Moreover, functional verification was conducted to verify the functional characteristics of the executions obtain identical results.

V. Evaluatuation and Discussion

Figure 3 presents the speedup values when running the parallelized *Initialization* phase with fixed number of cores. These values are calculated by comparing with the case when running on core numbers equal to Wavelet levels. It is observed that, from wavelet levels ranging from 3 to 10, the fixed 2 core execution is faster by an average speedup of 12%. Similar speedup is obtained between wavelet levels 11 and 20, when running on 4 cores. On wavelet levels higher than 20, the average speedup is calculated to be 14%, though on executions with fix 10 cores.

These results indicate that, running the initialization phase on fixed number of cores yield faster code. It is observed that, as the wavelet increase, using higher number of cores becomes efficient. The speedup which is obtained is nearly 12%.

Next, on the Figure 4, speedup values which correspond to running parallelized *Processing* phase with fixed number of cores. Again, values are calculated by comparing with the case when running on core numbers equal to Wavelet levels. It is observed that this optimization approach is effective especially when the Wavelet level is greater than 13. Fix core 2 case yields the best results. This was expected, since it decreases the negative effect of barrier synchronization. On wavelet levels higher than 13, the average speedup is calculated to be 10%, though on executions with fix 2 cores.

Figure 6 presents the effect of filter length. Filter length has an effect only on the *Processing* phase, thus test is conducted on the complete parallelization case. As expected, increasing the filter length has a direct effect on the speedup of the parallelization. This is primarily due to the fact that, as filter length increase, the effect of synchronization becomes less

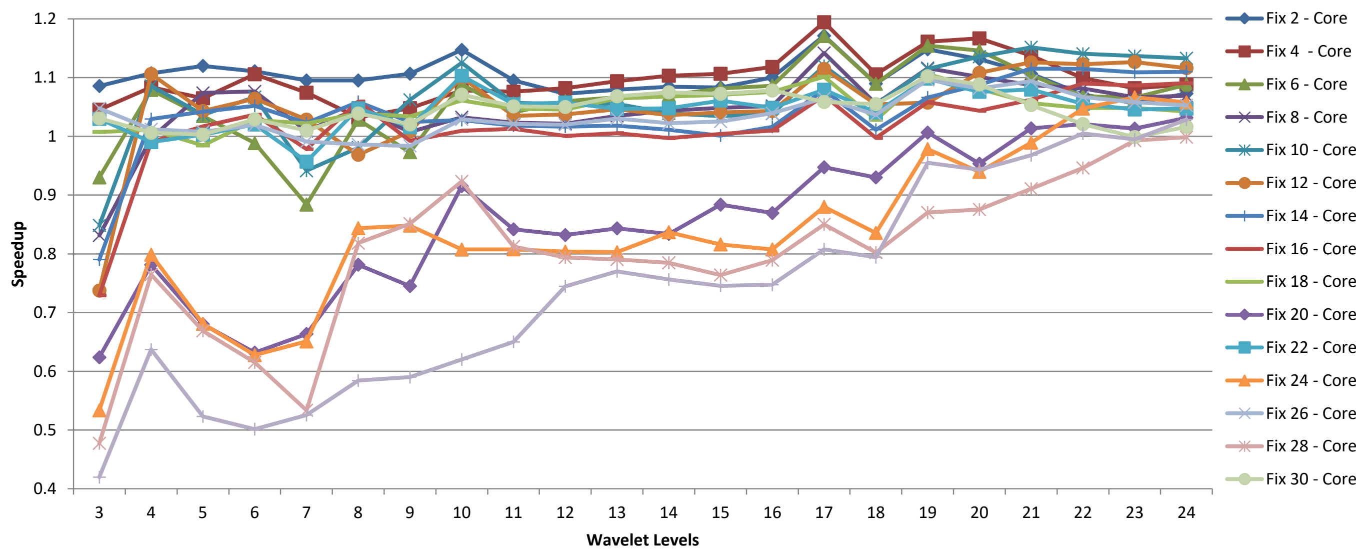

Fig. 3. Speedup of the paralleled Initialization phase with fixed number of cores compared to an implementation with cores equal to the Wavelet levels.

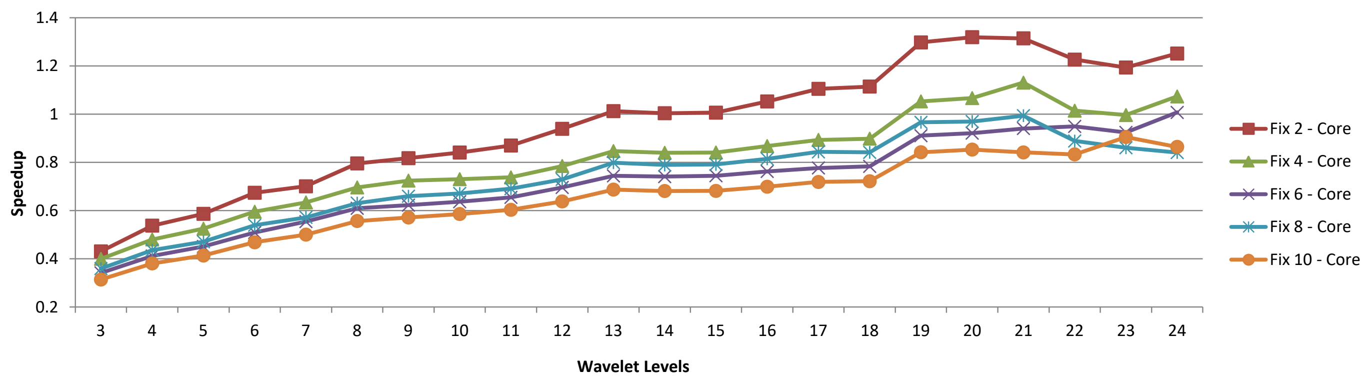

Fig. 4. Speedup of the paralleled Processing phase with fixed number of cores compared to an implementation with cores equal to the Wavelet levels.

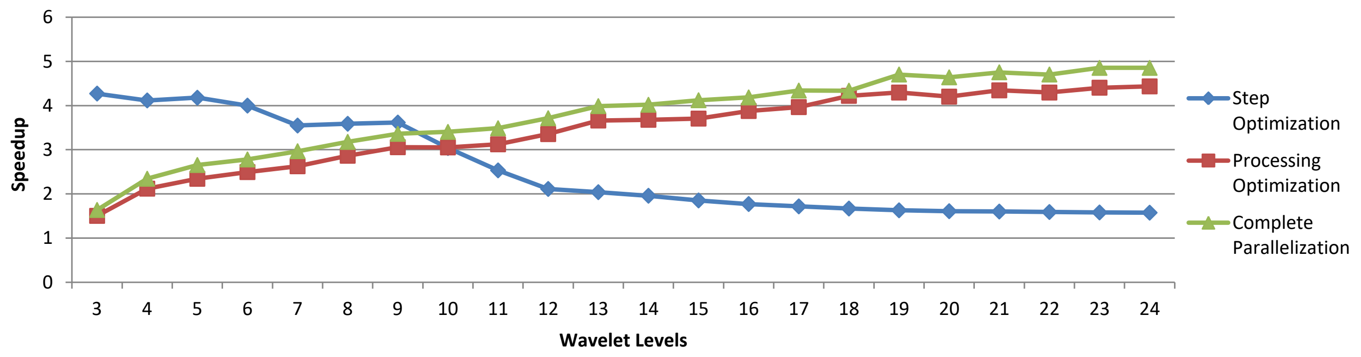

Fig. 5. Speedup of combining the best optimisation approaches compared to an implementation with cores equal to Wavelet levels without optimisations.

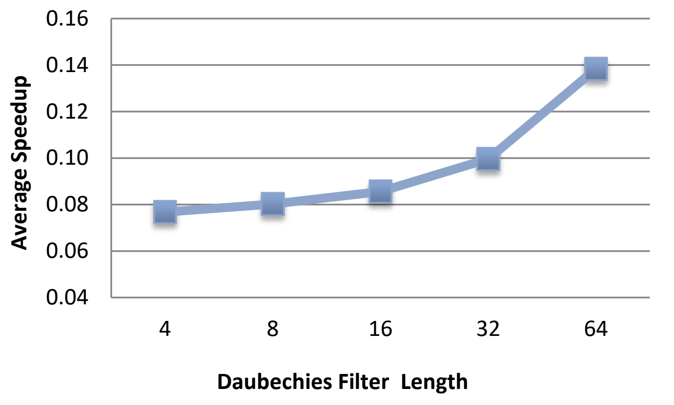

Fig. 6. Average speedup of the paralleled Processing phase with different filter lengths.

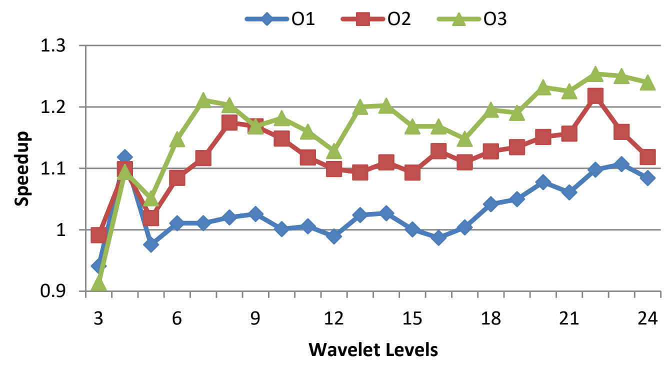

Fig. 7. Speedup of the parallel algorithm with using built-in OpenMP optimization flags.

important. On filters of length 64, the completely parallel algorithm performs 2 times faster.

The effect of compiler optimizations is shown in Figure 7. Results indicate that the build-in **O3** optimization, fastens the completely parallel algorithm by at least 15%.

Figure 5 shows the combined effect of the optimization approaches, where the code is tested on the best configuration, i.e. fix 2 cores, with 64-length Daubechies filter and built-in compiler optimization flag **O3**. Observation is that on Wavelet levels less than 10, the *Step Optimization* algorithm has an average of 4 speedup. What is more interesting, when the Wavelet levels are greater than 10, the optimizations yield a speedup of 4 *Complete Parallelization*, in a scalable nature.

Observation is that proposed optimisation approaches can yield faster codes when run on proper configurations.

VI. Related Work

Previously, we have focused on parallelizing DSP filters on Maxeler dataflow cores [9] and NVIDIA CUDA platform [10], [11]. In both cases, we achieved faster codes with a scalability depending on the number of used cores.

Pan and Tompkins, [12], have presented a real time algorithm for ECG QRS detection. Their algorithm considers the slope, amplitude and with information, and adaptively adjusts to the thresholds and parameters. It uses integer arithmetic in order to operate without requiring much computation power. There are no execution times presented, though their analysis is concentrated in the quality, where their correctness rate 99.3 percent.

An efficient implementation of DWT's in Field Programmable Gate Array(FBGA) devices [13]. They have optimised the power consumption and throughput. Additionally, a three level DWT algorithm with 4 Daubechies length filter is presented.

Milcheski and Gusev [4] have proposed an efficient DWT implementation using a circular buffers. They obtained significant speedups of at least 15. This is further improved in our previous study [5] by 20%. These papers serve as a basis for the current paper.

Several studies in literature addressed the delineation concept of ECG signal. Alfaouri and Daqrouq [14] present algorithms which produces better quality output, however no consideration is made for the performance.

Kayhan and Ercelebi [15], proposed lifting scheme based DWT algorithm for ECG denoising. Tests were conducted with an 360Hz ECG signal with 216.000 samples. Their algorithm provided fast executions where on Daubechies filter of 8, 0.141s execution times were provided.

An interesting approach was reported by Rajmic and Vlach [16], where a segmented wavelet transform analysis was presented. This approach is very attractive in the real-time case, however the authors have only published the concept.

VII. Conclusions

This work contributes OpenMP optimization for the baseline drift elimination of ECG heart signals. Totally five optimization approaches were proposed. Results indicate that, each of them can yield faster codes on proper configuration.

Approaches *OA1* and *OA2* yields speedup values of at least 10%. On the other hand, results showed that as filter length increases, the proposed parallel algorithm's efficiency increases. To be more specific, the *OA3* approach speeds up the parallel code by a factor of 2, when the filter length is increased from 4 to 64.

The effect of compiler's built-in optimization strategies were tested. The outcome of this *OA4* approach was that the *O3* flag performs best, with at least a 15% performance gain.

Lastly the combined effect was tested as the proposed *OA5* approach. Observation was that the combined effect yields a speedup of 4.

As a future work, we plan to further optimize the DWT by considering real-time segmented wavelet transform analysis concept. Additionally, it would be interesting to port the code to dataflow engine and test the parallel DWT algorithm.

References

[1] M. Gusev, A. Stojmenski, and I. Chorbev, "Challenges for development of an ecg m-health solution," *Journal of Emerging Research and Solutions in ICT*, vol. 1, no. 2, pp. 25–38, 2016.

[2] P. Laguna, N. V. Thakor, P. Caminal, R. Jane, H.-R. Yoon, A. Bayés de Luna, V. Marti, and J. Guindo, "New algorithm for qt interval analysis in 24-hour holter ecg: performance and applications," *Medical and Biological Engineering and Computing*, vol. 28, no. 1, pp. 67–73, 1990.

[3] T. S. Lugovaya, "Biometric human identification based on ECG," 2005.

[4] A. Milchevski and M. Gusev, "Improved pipelined wavelet implementation for filtering ECG signals," University Sts Cyril and Methodius, Faculty of Computer Sciences and Engineering, Tech. Rep. 27/2016, 2016.

[5] E. Domazet and M. Gusev, "Parallelization of digital wavelet transformation of ecg signals," in *MIPRO, 2017 Proceedings of the 40th Jubilee International Convention*. Opatija, Croatia, in press: IEEE, 2017.

[6] P. Mehta and M. Kumari, "Qrs complex detection of ECG signal using wavelet transform," *International Journal of Applied Engineering Research*, vol. 7, no. 11, pp. 1889–1893, 2012.

[7] R. Polikar, "The wavelet tutorial," 1996.

[8] Intel, "Quick-reference guide to optimization with intel compilers version 12," 2010, https://software.intel.com/sites/default/files/compiler_qrg12.pdf.

[9] E. Domazet, M. Gusev, and S. Ristov, "Dataflow DSP filter for ECG signals," in *13th International Conference on Informatics and Information Technologies*, in press, Bitola, Macedonia, 2016.

[10] E. Domazet, M. Gusev, and S. Ristov, "CUDA DSP filter for ECG signals," in *6th International Conference on Applied Internet and Information Technologies*, in press, Bitola, Macedonia, 2016.

[11] E. Domazet, M. Gusev, and S. Ristov, "Optimizing high-performance CUDA DSP filter for ECG signals," in *27th DAAAM International Symposium*. in press, Mostar, Bosnia and Herzegovina: DAAAM International Vienna, 2016.

[12] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," *IEEE transactions on biomedical engineering*, no. 3, pp. 230–236, 1985.

[13] D. Shah and C. Vithlani, "Efficient implementations of discrete wavelet transforms using fpgas," *International Journal of Advances in Engineering & Technology*, vol. 1, no. 4, pp. 100–111, 2011.

[14] M. Alfaouri and K. Daqrouq, "ECG signal denoising by wavelet transform thresholding," *American Journal of applied sciences*, vol. 5, no. 3, pp. 276–281, 2008.

[15] S. Kayhan and E. Ercelebi, "Ecg denoising on bivariate shrinkage function exploiting interscale dependency of wavelet coefficients," *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, vol. 19, no. 3, pp. 495–511, 2011.

[16] P. Rajmic and J. Vlach, "Real-time audio processing via segmented wavelet transform," in *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07), Bordeaux, France*. Citeseer, 2007.