Dataflow DSP Filter for ECG Signals

Ervin Domazet, Marjan Gusev and Sasko Ristov
Ss. Cyril and Methodius University, Faculty of Computer Science and Engineering,
1000 Skopje, Macedonia
e-mail: ervin_domazet@hotmail.com, {marjan.gushev, sashko.ristov}@finki.ukim.mk

***Abstract*—Electrocardiogram (ECG) signal analysis and interpretation is achieved by Digital Signal Processing (DSP). ECG signals usually are accompanied by a lot of noise coming from several sources, including the noise from environment (electrical switching power or other related sources) or the internal noises generated by the human breathing physical movement or similar sources. DSP filters are essential in eliminating the noise and extracting the essential characteristic signal. Afterwards, the main ECG features can be detected and analyzed for further determination of the complex heart condition. This processing of the ECG signals is based on detecting the hidden information and the subtle deviation of the heart rhythm to alternating changes of the wave amplitude. In some specific cases, real time analysis of heart signals can save lives. Due to intensive data processing, sequential algorithms are insufficient to run in real-time, especially when a cloud data server processes thousands of data streams coming from remote wearable ECG sensors.**

In this research, we focus on parallelizing the sequential DSP filter for processing of heart signals on dataflow cores. Dataflow Computing is a completely different paradigm of computing than conventional CPUs, where instructions are parallelized across the available space, rather than time. It is a revolutionary way for High Performance Computing (HPC) solutions. Data streams are optimized by utilizing thousands of dataflow cores, providing order of magnitude speedups. We consider using Maxeler Systems for dataflow computing. The performance of the parallelized code will be compared to that of the sequential code. Our analysis shows speedups linear to the kernel size of the filter.

***Index Terms*—DSP, ECG, Heart Signal, Parallelization, Dataflow Computing, Maxeler.**

I. Introduction

Electrocardiogram (ECG) is a stream of electric impulses generated by the beating heart muscle. They are detected by electrodes placed on human skin, by measuring the electric potential that reaches the skin surface [1]. ECG stream holds cardiovascular condition of the patient. Figure 1 reveals general representation ECG signal with its representative P, QRS and T waves.

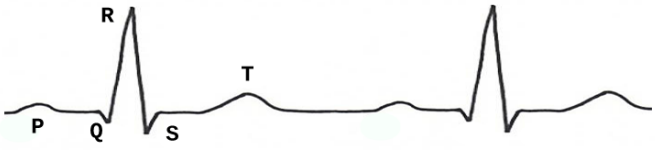

Fig. 1. General Representation of ECG Signal.

Interpretation of an ECG stream is essential for a better quality of life. Interpretation along with signal analysis is achieved by Digital Signal Processing (DSP) [2], [3]. Nevertheless, ECG signals are exposed to noise that stem from several sources, varying from environment (electrical switching power, radio waves or other related sources) to the internal noises generated by the human breathing physical movement or similar sources.

Precise interpretation and analysis of the ECG signal can be achieved by eliminating the noise. Essential data preprocessing phase is conducted by the DSP filters. Hereinafter, the main ECG features can be detected and analyzed for further determination of the complex heart condition. Processing of the signal is based on detecting hidden information and the subtle deviation of the heart rhythm to alternating changes of the wave amplitude [4].

Lugovaya [5] had focussed on revealing the efficiency of an ECG signal for identification, when compared to the three efficient biometric methods, i.e identification based on fingerprints, iris or retina, and face. Her experimental results showed that the rate of correct identification was 96%, which gave an insight for considering ECG signal as a new biometric characteristic. What is more important, she was successful in showing the persistence of an individual's ECG characteristics over time (slow

and gradual variations on ECG signal). This in turn makes it possible to detect subtle deviations of the heart rhythm and alternating changes of the wave amplitude.

In some specific cases real time processing of the ECG signal can save lives. Due to intensive data processing, sequential algorithms are insufficient to run in real-time, especially when a cloud data server processes thousands of data streams coming from remote wearable ECG sensors.

In this research, we focus on parallelizing the sequential DSP filter for processing of the heart signals on dataflow cores. The DSP filter is used for preprocessing of the ECG data, in order to eliminate noise from the ECG signal. Based on the noise components of the ECG signal, several filtering methods are available, such as Low pass, High bass and Bandpass filter.

Dataflow Computing is a completely different paradigm of computing than traditional CPUs. Instructions are parallelized across the available space, rather than time. It is a revolutionary way for High Performance Computing (HPC) solutions[6], [7]. Data streams are optimized by utilizing thousands of dataflow cores, providing order of magnitude speedups. Maxeler systems are used for dataflow computing [8]. The performance of the parallelized code is compared to that of the sequential code. Our analysis shows speedups linear to the kernel size of the filter.

The paper is organized as follows Section II presents DSP filters and convolution operators. In Section V, we explain the way we have parallelized the DSP filter by using Dataflow computing, namely Maxeler systems. Results of timings and performance figures of the parallelized code versus the original sequential code are presented in Section IV. Finally, the paper is concluded with a discussion and future work in Section V.

II. Digital Signal Processing

Digital Signal Processing (DSP) is the act of manipulating signals with intention varying from filtering, measuring to producing or compressing analog signals. As the power of computers radically increased during the last decades, so does the power of the DSP [3]. DSP had made a tremendous impact on science and engineering, by providing methodologies to deal with the most powerful technologies.

DSP had revolutionized many fields in science and engineering. There are many industrial sectors benefiting from the advancements on the DSP field such as Medical, Military, Space and Telephone. Electrocardiogram analysis, diagnostic imaging, voice and data compression, radars, secure communication, telephone signal filtering are among the range of revolutionized fields.

The focus of our research is on ECG signal filtering, with the intention to remove the noise that stem from several sources. The commonly used method in DSP filtering is the *convolution*, as one of the most important techniques of signal processing. It is defined as a mathematical operation that combines the *input stream* and the *impulse response* in order to generate a new *output stream*. In case of a filter, the impulse response is known as a *filter kernel*.

Each value of the output stream in digital signal convolution is represented as the sum of input stream multiplied by set of weight coefficients, which define the *impulse response*. The impulse response is the signal that results when a delta function (unit impulse) is the input in the DSP filter.

Denote by $f(i)$ the weight (filter kernel) coefficients in the range $i \in \{-\infty, +\infty\}$ if it is an infinite response filter. We will use finite response filters and the weight coefficients $h(i)$ in the range $i \in \{0, \ldots, M-1\}$, where M is the filter length. Let the input stream consists of elements $x(i)$ and the output stream of elements $y(i)$, for $i = 0, \ldots$. The convolution, as a mathematical operation can be expressed by (1).

$$y(i) = \sum_{j=0}^{M-1} h(j)x(i-j) \tag{1}$$

During this research, we have used the three classic filters to eliminate the noise: Low-Pass, High-Pass and Band-Pass filters.

A. Low-Pass Filter

Low-pass filters are designed to thoroughly weaken all the frequencies above the cutoff frequency, known as a *stopband*, while passing all frequencies below the *passband* [3]. These filters are composed of stream of data items. All samples of

the output stream are in fact a weighted average of the input with the adjacent points of low pass filter. A simple low-pass filter is presented in Figure 2.

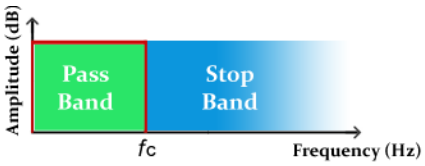

Fig. 2. Frequency response of a simple Low-pass filter.

B. High-Pass Filter

A high-pass filter has opposite characteristics of the low-pass filter. The effect of the filter is to weaken the frequencies below the *cutoff frequency* whereas passing all frequencies above the cutoff frequency.

As in the case of Lowpass filter, the output is generated with a weighted average of the adjacent input stream. The response characteristics of a simple high-pass filter is presented in Figure 3.

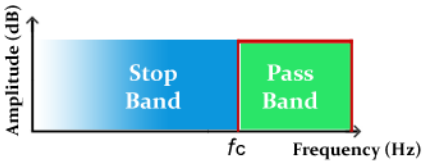

Fig. 3. Frequency response of a simple High-pass filter.

C. Band-Pass Filter

A band-pass filter is a composition of the high-pass and low-pass filters. This type of filter passes certain ranges of frequencies and rejects the frequencies of the remaining region. The frequency response of a simple band-pass filter is shown in Figure 4.

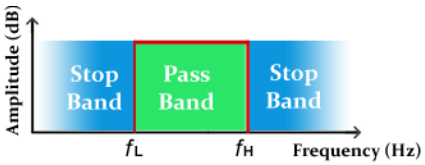

Fig. 4. Frequency response of a simple Band-pass filter.

III. Parallelization for Dataflow Computing

Algorithm 1 presents the sequential version of convolution of a one dimensional input with a kernel. The complexity of the algorithm depends on the input and kernel stream length, i.e $O(nm)$. When run on a CPU, the flow is sequential, meaning that the inner loop length depends on the kernel size. This flow is visualized in Figure 5.

Algorithm 1 Filtering algorithm

```
1:  procedure CONVOLUTION(in, kernel, out)
2:      i ← 0
3:      while i < inputSize do
4:          sum ← 0
5:          j ← 0
6:          while j < kernelSize do
7:              sum ← sum + in[i − j] ∗ kernel[j]
8:              j ← j + 1
9:          out[i] ← sum
10:         i ← i + 1
11:     return out
```

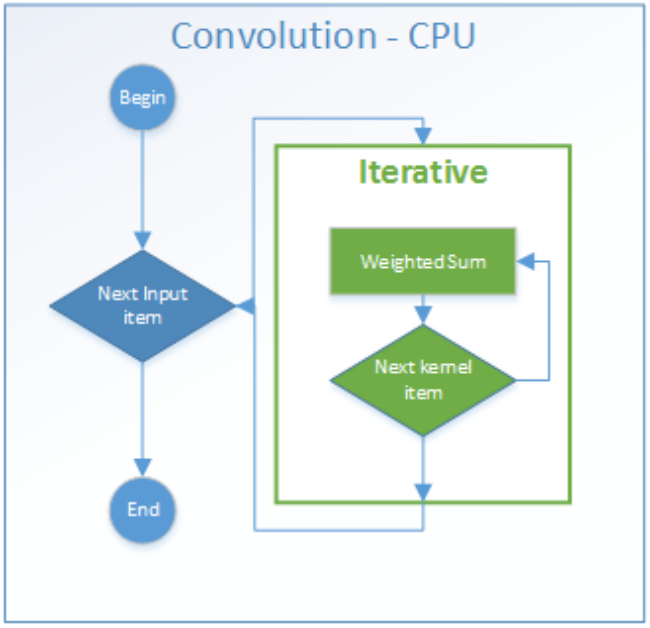

Fig. 5. Flow of sequential filtering algorithm.

In CPU computing, iterations are parallelized across the available time, and performed sequentially.

Dataflow is a completely different computing paradigm compared to the traditional CPUs. Here, the instructions are parallelized across the available space, rather than time. Using this key feature, we have achieved parallelization of kernel computing via Dataflow cores. In this manner, iterative kernel computation is massively parallelized. Depending on the kernel size, up to thousands of dataflow cores can be utilized synchronously, providing a speedup with a higher order of magnitude. The proposed solution is visualized in Figure 6.

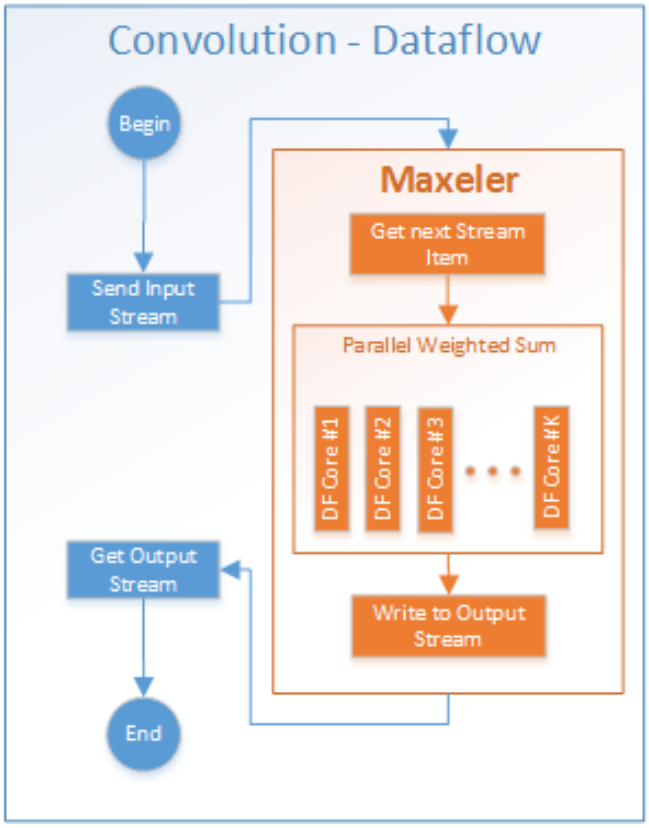

Fig. 6. Parallel Dataflow Computation algorithm.

IV. Tests and Results

The sequential code is tested on an 8-core Intel(R) Xeon(R) X5647, 2.93 Ghz system with 12GB of memory. On the other hand, parallelized code is tested on a Maxeler simulator. Five different kernel sizes are tested, and vice versa, for various length ECG input signals.

A. Functional Verification

To verify the functional characteristics of the execution of the sequential and parallel algorithms we have provided several experiments. The input was a short sequence of 500 samples of an ECG signal with all characteristic P, Q, R, S and T waves, as presented in Figure 7.

Figure 8 presents the effect of applying a low pass filter on the ECG signal. One can notice that the 50Hz noise is eliminated.

The effect of the high pass filter is presented in Figure 9. The effect of this filter is elimination of the baseline drift, caused by breathing and other physical movements.

The effect of the band pass filter as a combination of a low pass and high pass filter is presented in Figure 10. This filter eliminates the baseline drift, caused by breathing and other physical movements and also the 50Hz noise caused by the electricity.

We have verified both the sequential and parallelized solution and obtain identical results.

B. Speed-up Analysis

Speed-up is calculated by (2), where T_s is the time required to process the sequential algorithm, and T_p is the time required to process the parallel algorithm with p cores. Since the system clock on the sequential machine is much higher than the system clock on the parallel machine, we will compare the number of sequential steps N_s (operations required by the sequential algorithm) and the number of processing steps N_p (operations required by the parallel algorithm), by considering the sequential system clock C_s and parallel device's system clock C_p.

$$S_P = \frac{T_s}{T_p} = \frac{N_s}{N_p}\frac{C_p}{C_s} \tag{2}$$

Note that for each input signal, the speedup values are calculated by using T_s and T_P values for the same input configuration. Hence, the main reason for using these values is to compare the performance of the sequential code on CPU and the parallelized code on Maxeler Dataflow Engine (DFE).

Sequential running code has mainly two phases: the initialization and processing phases. Let the input stream contains N elements and the filter kernel M elements. On initialization, the input stream and filter kernel are transfered from the memory to the CPU by a total of $N + M$ memory access

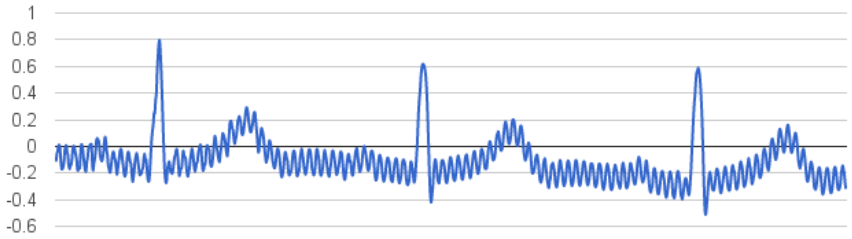

Fig. 7. A segment of an ECG signal with several QRS complexes.

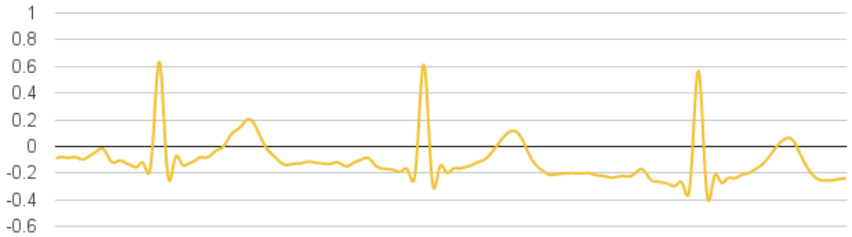

Fig. 8. The ECG signal filtered with a low pass filter of 30Hz.

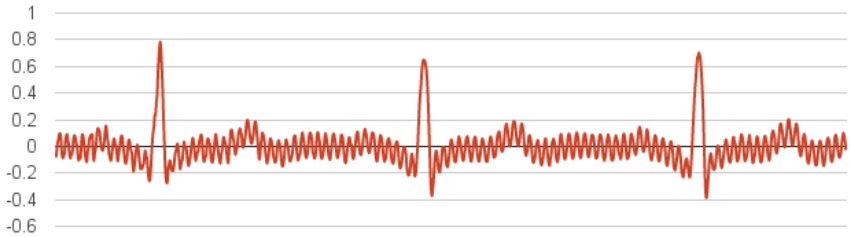

Fig. 9. The ECG signal filtered with a high pass filter of 0.5Hz.

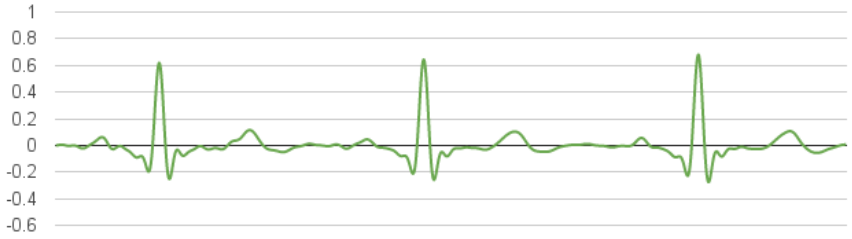

Fig. 10. The ECG signal filtered with a band pass filter between 0.5Hz and 30Hz.

operations and the N output elements are written in the memory. Processing phase requires $N * M$ multiplications and $N * M$ additions. Assuming that each memory access, multiplication and addition requires 1 processing step, the relation that shows the total number of processing steps is presented in (3).

$$N_s = 2NM + 2N + M \tag{3}$$

The number of operations for the parallel algorithm is calculated differently. In addition to processing, the dataflow engine needs to transfer data from memory to device and return the results back, which is equal to a total of $N + N + M$ memory access operations for the input and output stream, and the filter kernel. The dataflow engine performs operations on N samples concurrently in a pipelined manner, so the processing takes N processing steps plus the pipeline length of the number of operations, which is equal to the kernel length M. Note that the summation can be realized in a tree parallel organization, which will take only $log_2 M$ steps, but since we expect that $N >> M$ it will not affect the final result. So the total processing steps is expressed by (4).

$$N_p = 2N + M + N + M = 3N + 2M \tag{4}$$

Table I presents the number of operations and calculated speedup for our experiments where the input contains 100.000 samples and kernel length was 100, 500, 2000, 5000 and 7500. The sequential machine clock was 2.93 GHz and the dataflow Maxeler device system clock 400 Mhz. The speedup increases with the length of the kernel size.

TABLE I. Speed-up analysis as the kernel size increase.

No.	Platform	Input Size	Kernel Size	Num. of Opers.	S_P
1	CPU	100000	100	20200100	9.19
	DFE			301000	
2	CPU	100000	500	100200500	45.45
	DFE			301000	
3	CPU	100000	2000	400202000	179.72
	DFE			304000	
4	CPU	100000	5000	1000205000	440.47
	DFE			310000	
5	CPU	100000	7500	1500207500	650.18
	DFE			315000	

V. Discussion and Conclusions

This work contributes Maxeler Dataflow parallelization for noise filtering of ECG heart signals. The provided parallel solution takes advantage of thousands of Dataflow cores. Their size is increasing linearly (according to the maximum number of available space) depending on the kernel size used.

Results obtained by executing the sequential algorithm and the parallel dataflow algorithm show that the obtained results are identical for low-pass, high-pass and band-pass filters.

The analysis shows that the speedup is proportional to the filter length. In this research we have experimented with the Hamming window and the Blackman window with length of 100 and 200 elements to obtain relatively good results.

This research is the first step of the ECG signal processing with the intention to extract hidden information. As the next work we plan to parallelize the Wavelet Transformation sequential code for Maxeler Dataflow.

As future work, we plan to carry out more tests on Maxeler DFE with higher available space. We believe that the parallelized code will scale linearly with increasing filter size.

References

[1] P.D.Khandait, N. Bawane, and S.S.Limaye, "Article: Features extraction of ecg signal for detection of cardiac arrhythmias," *IJCA Proceedings on National Conference on Innovative Paradigms in Engineering and Technology (NCIPET 2012)*, vol. ncipet, no. 8, pp. 6–10, March 2012, full text available.

[2] A. Antoniou, *Digital signal processing*. McGraw-Hill Toronto, Canada:, 2006.

[3] S. W. Smith, *Digital signal processing: a practical guide for engineers and scientists*. Newnes, 2003.

[4] R. Acharya, S. M. Krishnan, J. A. Spaan, and J. S. Suri, *Advances in cardiac signal processing*. Springer, 2007.

[5] T. S. Lugovaya, "Biometric human identification based on ecg," 2005.

[6] J. A. Sharp, *Data flow computing: theory and practice*. Intellect Books, 1992.

[7] M. J. Flynn, O. Pell, and O. Mencer, "Dataflow supercomputing," in *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*. IEEE, 2012, pp. 1–3.

[8] "Maxeler dataflow computing," https://www.maxeler.com/technology/dataflow-computing/, last visited on 26.03.2016.