Abstract
In today’s research and market, IT applications for health-care are gaining huge interest of both IT and medical researchers. Cardiovascular diseases (CVDs) are considered the largest cause of death for both men and women regardless of ethnic backgrounds. More efficient treatments and most importantly efficient methods of cardiac diagnosis that examine heart diseases are desired. Electrocardiography (ECG) is an essential method used to diagnose heart diseases. However, diagnosing any cardiovascular disease based on the 12-lead ECG printout from an ECG machine using human eye might seriously impair analysis accuracy. To meet this challenge of today’s ECG analysis methodology, a more reliable solution that can analyze huge amount of patient’s data in real-time is desired. The software solution presented in this article is aimed to reduce the risk while diagnosing cardiovascular diseases (CVDs) by human eye, computation of large-scale patient’s data in real-time at the patient’s location and sending the required results or summary to the doctor/nurse. Keeping in mind the importance of real-time analysis of patient’s data, the software system has built upon small individual algorithms/modules designed for multi-core architecture, where each module is supposed to be processed by an individual core/processor in parallel. All the input and output processes to the analysis system are made automated, which reduces operator’s interaction to the system and thus reducing the cost. The outputs/results of the processing are summarized to smaller files in both ASCII and binary formats to meet the requirement of exchanging the data over Voice and Data Networks.

**Keywords:** Electrocardiogram (ECG) analysis, parallel algorithms/modules, multi-core architecture, real-time analysis, health-care applications, 12-lead ECG.
Table of Contents

1. INTRODUCTION ........................................................................................................................................ 1

2. BACKGROUND AND LITERATURE REVIEW .................................................................................. 4
   2.1 BIOMEDICAL LITERATURE REVIEW ............................................................................................... 4
       2.1.1 Electrocardiogram (ECG) ......................................................................................................... 4
       2.1.2 The 12-lead ECG ...................................................................................................................... 5
   2.2 PARALLELING SOFTWARE ............................................................................................................. 6
   2.3 MULTI-CORE HARDWARE ........................................................................................................... 6
   2.4 PREVIOUS WORK .......................................................................................................................... 6

3. PARALLEL ELECTROCARDIOGRAM (ECG) ANALYSIS ALGORITHMS ...................................... 8
   3.1 FLOW CHART .................................................................................................................................... 9
   3.2 MAIN MODULE .................................................................................................................................. 9
   3.3 LEAD MODULE .................................................................................................................................. 10
   3.4 ARITHMETIC MODULE .................................................................................................................. 11
       3.4.1 Two numbers of fixed Point 'Q' format .................................................................................... 11
       3.4.2 Fixed point 'Q' format and integer ......................................................................................... 11
       3.4.3 Conversion between fixed point 'Q' formats ............................................................................ 12
       3.4.4 Fixed point 'Q1' and 'Q2' format ............................................................................................ 12
       3.4.5 Conversion integer to/from float ............................................................................................ 12
   3.5 AUTOCORRELATION AND PERIOD MODULE ............................................................................. 13
       3.5.1 Difference vector ...................................................................................................................... 13
       3.5.2 Autocorrelation ........................................................................................................................ 13
       3.5.3 Signal period ............................................................................................................................ 13
   3.6 ECG PEAKS METHOD1 MODULE ................................................................................................. 14
       3.6.1 Maximum peak (Threshold) ..................................................................................................... 15
       3.6.2 'R' peaks .................................................................................................................................... 15
       3.6.3 Peaks 'P' and 'T' ...................................................................................................................... 15
       3.6.4 Minimum peaks/values .......................................................................................................... 15
   3.7 ECG PEAKS METHOD2 MODULE ................................................................................................. 16
       3.7.1 Minimum Peak (Threshold) ...................................................................................................... 16
       3.7.2 'R' peaks .................................................................................................................................... 16
       3.7.3 Peaks 'P' and 'T' ...................................................................................................................... 17
       3.7.4 Maximum peaks/values .......................................................................................................... 17
   3.8 READ WRITE MODULE .................................................................................................................. 17
       3.8.1 Reading Input Files .................................................................................................................. 17
       3.8.2 Sampling Interval ..................................................................................................................... 18
3.8.3 Writing Headers ........................................................................................................ 18
3.8.4 Saving Results .......................................................................................................... 18

4. RESULTS AND PERFORMANCE .............................................................................. 19
   4.1 INPUT AND OUTPUT PROCESSES AUTOMATION ............................................. 19
   4.2 OUTPUT FILES FORMATS (BINARY AND ASCII) ........................................... 20
   4.3 RESULT/OUTPUT FILES .................................................................................... 20
   4.4 LOG COMPONENT .............................................................................................. 23
   4.5 PERFORMANCE EVALUATION .......................................................................... 24

5. CONCLUSION AND FUTURE WORK ........................................................................ 28
   5.1 CONCLUSION ..................................................................................................... 28
   5.2 FUTURE WORK .................................................................................................. 28
   ACKNOWLEDGMENTS ............................................................................................. 29

6. REFERENCES .............................................................................................................. 30

Table of Figures

Figure 1: Electrocardiogram (ECG) Analysis based on Autocorrelation Function ................ 7
Figure 2: Flow Chart Electrocardiogram (ECG) Parallel Algorithms .................................. 9
Figure 3: Autocorrelation Function (ACF) of Lead-1 ......................................................... 21
Figure 4: Difference vector of Lead-I ................................................................................ 21
Figure 5: Detailed results of Lead-I Sampling interval ‘0-5’ .............................................. 22
Figure 6: Summarized and average results ..................................................................... 23
Figure 7: Average of heart period and heart rate .............................................................. 23
Figure 8: Electrocardiogram (ECG) analysis system log component .............................. 24
Figure 9: Performance comparison of old and new versions of Electrocardiogram (ECG) analysis system ........................................................................................................ 26
Figure 10: Electrocardiogram (ECG) Analysis based on Autocorrelation Function .......... 27
Chapter-1

1. Introduction

Despite the ongoing advances in heart treatment, cardiovascular diseases (CVDs) are the number one cause of death globally. CVDs kill more people annually as compared to any other disease. According to World Health Organization (WHO) report, an estimated 17.1 million people died due to CVDs in 2004, which represent 29% of all global deaths [1]. In the United States [2] and Canada [3] as well as in many other countries, the various forms of cardiovascular disease (CVD) and stroke remain by far the top most cause of death for both men and women regardless of ethnic backgrounds. By 2030, an estimated 23.6 million people will die from CVDs, mainly from heart diseases and strokes and thus CVDs are projected to remain the single leading causes of death [1]. To meet the challenge, more efficient treatments and most importantly efficient methods of cardiac diagnosis that examine heart diseases are desired. However, the current available biomedical equipments for heartbeat sensing and monitoring lacks the ability of providing large-scale analysis and remote, real-time computation at the patient’s location. The project attempts to address these issues and propose a system that could provide large-scale analysis and remote, real-time computation at the patient’s location. This way, the system would reduce the cost of hospitalization by sensing heartbeats; processing the data in real-time at patient’s location and sending the required results to the doctor/nurse. The system is aimed to replace the current biomedical machines, reduce the doctor’s and nurse’s work-load and improve the quality of CVDs diagnoses.

A solution for analyzing Electrocardiogram (ECG) data had been proposed in “A Multiprocessor System-on-Chip for Real-Time Human Heart Monitoring and Analysis” [4]. The algorithms presented in this previous work are based on autocorrelation that calculate the parameters i.e. heart period, peaks (P, Q, R, S, T, and U) in a given sampling interval and inter-peak time spans of an ECG signal [5]. These are parallel modules, where each module is supposed to process and generate the results for a single lead. As a result there are 12 modules to process and analyze all the 12 leads of an ECG signal. The system reads the data from input files in chunk of 4 seconds and analyzes it then goes to the next chunk of 4 seconds [4]. The system had low performance and the generated output files were too heavy to be sent through a network for further diagnosis by the doctors. This project is intended to further
modularize the system and build new parallel algorithms in a way where each module is to be processed by an individual core/processor. As the system has already different modules for different ECG leads, the aim of this project is to group the common functions i.e. autocorrelation, method of finding peaks, finding periods of the signals etc from different leads and make separate modules that are supposed to be processed by individual cores/processors. The system is supposed to be extended to generate the patient's output files in a format that is suitable to be transferred through a data network and automate the input and output processes to the system.

The new proposed system presented in this article has been divided and programmed in more smaller individual and paralleled algorithms/modules. The system has both paralleled and individual modules based on the ECG leads and based on the functionality i.e. autocorrelation, finding peaks (P, Q, R, S, T, and U), periods and reading/writing of the results/output files etc. The system is supposed to process each individual module on a single core/processor using multi-core architecture. The performance of the system has much increased due to modularizing the system further in new modules because of the fact that each module is supposed to be processed by an individual core/processor. The methods used for buffering and output operations while writing results to the output files have been changed in the new system, which results in approximately more than 4 times (403%) increase in performance. The entire input and output processes to the Electrocardiogram (ECG) analysis system have been automated. In addition, the system takes care of downloading the patients input data/files from one or several online databases at the same time, changing the data files to a format suitable for system's input and giving the data as input to the system. The system reads the input and output paths including the path(s) to online databases from the configuration files given by the user. New output files have been introduced to save the summary of the results, average of heart periods and heart rates, which are much smaller in sizes compared to the output files by the old system. These new output files are supposed to be transferred through networks for further diagnosis by the doctors. The system is upgraded to give the user flexibility to either save the output files in ASCII or binary format. A log component has been added to the ECG system that calculates the processing/execution time taken by each lead/patient in micro-seconds.

In chapter-2 of the article, we present biomedical literature review, where it discusses the Electrocardiogram (ECG) signal in general and then presents the twelve different leads of the ECG signal. It also presents an overview of paralleling software and multi-
core hardware. In the end of the chapter we discuss and present a short overview of the previous work in the area. Chapter-3 presents paralleled ECG analysis algorithms designed as a major task of the project. Chapter-4 discusses the results of the electrocardiogram (ECG) analysis system and performance improvements. In the end of the article we present the conclusion and future work.
Chapter-2

2. Background and Literature Review

This chapter presents the necessary background needed to perform the assigned task. The first phase of the project was dedicated to read the materials relevant to the assigned task and to understand the contributed work by previous researchers on the topic.

The chapter is organized as first presenting the biomedical background needed to know what the Electrocardiogram (ECG) is, to understand how it is interpreted and how it operates. A discussion about paralleling software and multi-core hardware is given in the second and third section of the chapter, respectively.

2.1 Biomedical Literature Review

This section presents the literature review performed to get biomedical background needed before starting the actual work. The major tasks here were to understand the Electrocardiogram (ECG), how the different leads of an ECG signal are obtained and how they are interpreted. After understanding Electrocardiogram, the algorithms that are used in processing the leads of an ECG signal were studied [4][5].

In the below sub-sections a detailed overview of Electrocardiogram and the algorithms that are used to process the ECG leads and generate the required results are given, respectively.

2.1.1 Electrocardiogram (ECG)

Electrocardiogram is the graphical representation of the electrical activity of the heart [9]. The two terms ‘EKG’ and ‘ECG’ are synonymous and are used interchangeably, the first term is based on the original word ‘Electrokardiogram’ and was first introduced by the famous Dutch physiologist, Willem Einthoven in the 1890s and early 1900s [7], the latter term is derived from the English version ‘Electrocardiogram’, which is the most common, used today. The first introduced ‘ECG’ consisted of only three limb leads that are considered the main limb leads, in 1930’s the six chest leads were added and then in 1942 the three augmented limb leads were added to the complex, which resulted in the 12 leads ECG that is used today [7].
The electrical activity of the heart is sensed and picked up by the electrodes that are placed on the surface of the body. In total, there are ten electrodes used to pick up the heart activity and are then transmitted to a recording machine for generating 12 leads. The basic four limb electrodes are placed one each on both arms and one each on both leg. The other six electrodes are placed on the chest. The electrode used to place on the right leg is in ground or neutral [8]. The different leads of ECG are discussed in the next sub-section.

2.1.2 The 12-lead ECG

A lead is basically the voltage difference sensed by two different electrodes. The 12 leads of ECG are divided into three categories, bipolar limb leads, uni-polar augmented leads and uni-polar chest leads. The leads in different categories are discussed below,

**Bipolar Limb Leads**

- Lead-I: Lead-I represents the difference of the voltages obtained by the (positive) left arm (LA) electrode and right arm (RA) electrode [10]
- Lead-II: Lead-II represents the difference of the voltages obtained by the (positive) left leg (LL) electrode and right arm (RA) electrode [10]
- Lead-III: Lead-III represents the difference of the voltages obtained by the (positive) left leg (LL) electrode and left arm (LA) electrode [10]

**Uni-Polar Augmented Limb Leads**

- Augmented Vector Right (aVR): The electrodes placed on right arm (LA) and left leg (LL) are connected together by two resistors; the lead is the difference in the voltages between the midpoint of resistors and left arm (RA) electrodes [10]
- Augmented Vector Left (aVL): The electrodes placed on right arm (RA) and left leg (LL) are connected together by two resistors; the lead is the difference in the voltages between the midpoint of resistors and left arm (LA) electrodes [10]
- Augmented Vector Foot (aVF): The electrodes placed on right arm (LA) and left arm (LA) are connected together by two resistors; the lead is the difference in the voltages between the midpoint of resistors and left leg (LL) electrodes [10]
Uni-Polar Chest Leads

- There are six leads in this category that are consist of V1 to V6 and are obtained by measuring the difference in voltages of each chest electrode with the electrode on the right leg (RL) which is set to ground or neutral [10].

2.2 Paralleling Software
Traditionally, computer programs have been written for processing sequentially. The instructions are executed by central processing unit (CPU) on one computer, where the compiler may process the instructions in a program one after the other. This kind of approach will be very slow for an application like Electrocardiogram, where it has to process algorithms for all the 12 leads of an ECG signal. To overcome this problem the ECG processing application had been divided in to twelve different modules where each module consisted of one lead and was supposed to be processed on one core or processor at the same time [4][5]. But, while processing hundreds or thousands of patients the current application was not appropriate and a more modular approach was needed, where each lead could assign its different tasks to different cores or processors and then they run the tasks simultaneously.

2.3 Multi-Core Hardware
In order to achieve the parallel execution of computer programs or different modules of a computer program as discussed in the sub-section ‘Paralleling Software’, we need a hardware platform that can execute multiple programs or modules simultaneously [11]. The target application is assumed to be processed on Multi Processor System on Chip (MPSoC) [5].

2.4 Previous Work
This section presents the work that had done in the area before starting the project. A thorough study of the previous work was needed in order to continue developing and enhancing the algorithms and the code.

The project is the continuation of work published in “A Multiprocessor System-on-Chip for Real-Time Human Heart Monitoring and Analysis” [4]. The algorithms in the publication are based on autocorrelation that calculate the parameters i-e heart period, peaks (P, Q, R, S, T, and U) in a given sampling interval and inter-peak time spans of an ECG signal [5]. These are parallel modules, where each module is supposed to process and generate the results for a single lead, so there are 12 different modules to process and analyze all the 12 leads of an ECG signal. The system could be easily
extended for 15 leads ECG by just adding three more modules to the system for analyzing new added modules. The system reads the data from input files in chunk of 4 seconds and analyzes it then goes to the next chunk of 4 seconds [4].

The system based on autocorrelation is summarized in Figure 1 [5], where the system reads the input file in chunks of 4 seconds and then two processes are used. The first process finds ‘Heart Period’ and another process finds the peaks in a given sampling interval. To calculate Heart Period first the process calculates the discrete derivative of the ECG function then shifts the discrete derivative towards right by 25% and correlates the shifted version with the original counterpart. While, the other process finds the maximum value in a sampling interval, estimates a safe threshold and then finds the rest of the peaks i-e ‘P’, ‘Q’, ‘S’ and ‘T’.

Figure 1: Electrocardiogram (ECG) Analysis based on Autocorrelation Function
Chapter-3

3. Parallel Electrocardiogram (ECG) Analysis Algorithms
This chapter presents parallel electrocardiogram (ECG) analysis algorithms. An introduction to electrocardiogram (ECG) and how the different leads are interpreted is presented in chapter-2 ‘Background and Literature Review’. The parallel electrocardiogram (ECG) analysis algorithms are based on the work published in ‘MPSoC ECG Biochip: A Multiprocessor System-on-Chip for Real-Time Human Heart Monitoring and Analysis’ [4]. In the previous work, there were twelve different files/parallel algorithms, each processing one lead. In order to minimize the processing time, we separated and grouped the function that were common among all the leads or some leads of the ECG analysis code wherever possible and made them individual modules. The individual modules are supposed to be processed by individual cores/CPUs each as a separate algorithm.

The chapter is organized as first presenting the flow chart that explains how the different algorithms process the input data how they interact with each other in order to send/receive output/input. In the later sections, we present an overview of each algorithm and how it analyses the patient’s data to generate the required results.
3.1 Flow Chart
This section presents the flow chart of the electrocardiogram (ECG) analysis system. It also presents how the different modules in the system process the patient’s input data, interact with other modules and generate the required results. The flow chart of the system is given in Figure 2.

Figure 2: Flow Chart Electrocardiogram (ECG) Parallel Algorithms

3.2 Main Module
This section presents the main module of the electrocardiogram (ECG) analysis system. Main module consists of main function where all the headers are included to import other different modules. All the other lead modules are called from the main module where each lead module calls other sub modules to process the patient’s data and generate results. These sub modules are discussed later in this chapter. Each lead module and sub modules are supposed to be processed on an individual core/CPU.

A log system has been introduced in the new electrocardiogram (ECG) analysis system. When the system starts to process the patients input data it creates a log file and
saves the processing/execution time for each lead then calculates the processing/execution time patient-wise and in the end it calculates the time taken to process all the patients. These processing/execution times are saved in the log file for further analysis. All the processing/execution times are saved in microseconds.

The main module is also responsible for automation of input and output processes to the system and creates directories that are needed for saving the output files. In addition, the main module takes care of whether the output files are supposed to be saved in binary or ASCII. These functionalities are discussed in more detail later in chapter-4 ‘Results and Performance’

### 3.3 Lead Module

These are the modules that are responsible for processing the twelve leads of the electrocardiogram (ECG). We have twelve modules and each is responsible for processing one lead. Here we divided the twelve leads in two different categories on the basis of their characteristics. The leads with upwards peaks are grouped in one category and the leads with downwards peaks are grouped in the other category. The group of leads with upwards peaks consists of Augmented Vector Foot (aVF), lead-1, lead-2, lead-3, V5 and V6 and the group of leads with downwards peaks consists of Augmented Vector Right (aVR), Augmented Vector Right (aVL), V1, V2 and V3. The first group of leads finds their peaks on the basis of algorithms in ‘ECG Peaks Method1’ and the second group finds peaks on the basis of algorithms in ‘ECG Peaks Method2’ as shown in Figure 2.

The module creates dynamic paths to the patients input files, open the input files and read the required data from the input files. The program also creates dynamic paths to the output files in this module. In total, there are five output files for each patient per lead. Three output files save the detailed output/results, first holding the final results, second the autocorrelation function and third the difference vector of the signal. There are two new files added in this version of the electrocardiogram (ECG) analysis system. One contains the summary of the file that are holding the final results and the other keeps the average of ‘Heart Periods’ and of ‘Heart Rates’. The latter two files are supposed to be sent for diagnosis and we keep the other three detailed files for future analysis. ‘ifstream’ and ‘ofstream’ are used to provide interface to the input and output files respectively.
The module is also responsible to provide interfaces to communicate with other modules in the systems such as ‘Arithmetic’, ‘ECG Peaks Method1’, ‘ECG Peaks Method2’, ‘Autocorrelation & Period’ and ‘Read Write’.

3.4 Arithmetic Module
Arithmetic module is responsible for handling arithmetic operations. All the arithmetic operations in the electrocardiogram (ECG) analysis system are in fixed point integer format. The numbers that the arithmetic module processes are of 'Q' format where the number of integer bits (optional) and the number of fractional bits are specified. In general, ‘Q’ format numbers are represented in 'Qm.n' format where 'm' represents the integer portion and 'n' represents the fractional portion of the fixed point integer.

The module contains several different functions and each function is responsible for different arithmetic operations. Many of the functions perform addition, subtraction, division and multiplication in different formats of fixed point integers. Some functions are responsible for conversion operations like conversion from one fixed point integer format to another, conversion from floating point to fixed point integer format and vice versa. The arithmetic functions in the system are divided and grouped in different categories based on the nature of operations these functions perform. An introduction to each of these category is given in the below sub sections.

3.4.1 Two numbers of fixed Point 'Q' format
This category consists of functions that perform the basic operations like addition, subtraction, multiplication and division on two numbers i-e 'a' and 'b' of fixed point 'Q' format. The functions take inputs as fixed point integers of 'Q' format and produce results in fixed point integers of 'Q' format.

The addition and subtraction are simple operations in this category but multiplication and division need additional operations/calculations. Arithmetic shift is used both in multiplication and division. The arithmetic shift can be performed in C/C++ by the shift operators '<<' and '>>'. In multiplication, the digits in the result are shifted towards right by a value of 'Q' and in division, the digits in the dividend are shifted towards left by a value of 'Q' and then are divided by the divisor, where ‘Q’ is the format in fixed point integers and can be given as an input to the function.

3.4.2 Fixed point ‘Q’ format and integer
This category represents the functions that perform arithmetic operations when one of the number 'a' is of fixed point 'Q' format and the other number 'b' is an integer. In
addition and subtraction, we need to shift the digits towards left in the integer 'b' by a
factor of 'Q' before adding/subtracting to/from 'a'. No shifting of digits occurs in
multiplication and division where the divisor ‘b’ is of integer format, while the digits in
the divided ‘a’ are shifted towards left by a factor of ‘2*Q’ in the division where the
divisor ‘b’ is of fixed point ‘Q’ format.

3.4.3 Conversion between fixed point ‘Q’ formats
In this category there is one function that is responsible for conversion of fixed point
‘Q’ formats. It converts a number 'a' from 'Q1' to 'Q2' format. This function uses an 'if'
condition and checks if the format ‘Q2’ is greater than ‘Q1’. In case ‘Q2’ is greater than
‘Q1’, the function performs left arithmetic shift otherwise right arithmetic shift.

3.4.4 Fixed point ‘Q1’ and ‘Q2’ format
This category presents the functions that perform arithmetic operations when one
number 'a' is in 'Q1' format and the other number 'b' in 'Q2' format. As both numbers
are in different fixed point integer formats, therefore, first the function calls the
conversion function from one fixed point integer format to the other fixed point
integer format as discussed in the previous sub section 3.4.3 and then performs the
arithmetic operations. All the arithmetic operations i-e addition, subtraction,
multiplication and division are simple operations after conversion.

3.4.5 Conversion integer to/from float
This category of arithmetic operations consists of two functions, one is used to convert
a number from integer to float and the other is used to convert a number from float to
integer format.

To convert a number from fixed point integer of 'Q' format to floating point, the
function first changes ‘a’ from data type ‘int’ to ‘double’ and divide it by ‘1’ with digits
shifted towards right by a value of ‘Q’. Multiplication is the same process but the only
difference is the number is multiplied by ‘1’ with digits shifted towards right by a value
of ‘Q’.

All the functions in the arithmetic operations are of two types, one is the simple
version as we discussed in the above sub sections and the other type contains the
same functions but the inputs to the functions and the outputs of the functions are
'long int' instead of 'int'.
3.5 Autocorrelation and Period Module

This module contains a class named as ‘Autocorrelation & Period’, which is defined as public. The class consists of three functions. The first function is responsible for calculating the difference vector of the given input data, the second function calculates the autocorrelation function of data given the number of lags and the last function finds the period from the Autocorrelation function (ACF) that was generated in the second function.

The purpose of these functions/operations and an introduction to each is given in the below sub-sections.

3.5.1 Difference vector

The function is responsible for calculating the difference vector of the given input data. Here is the equation that is used to calculate the difference vector,

\[ \text{Diff}[i] = x[i+1] - x[i] \]

The program reads the input data from start to end and calculate the required difference vector. For subtraction the programs calls the required arithmetic function. The results of difference vector are saved in a separate output file in floating point format.

3.5.2 Autocorrelation

This function calculates autocorrelation for the patient's input data given the number of 'lags' and stores the results in a separate file 'ACF', the program can store the results both in ASCII and binary depends on the on the user input.

Correlation is a mathematical operation that is very similar to convolution. Correlation takes two signals as input and results in a third signal as output. This output signal is called the cross-correlation of the two input signals. If both the input signals are same then the resulting signal is instead called the autocorrelation [13].

The autocorrelation function of a random signal describes the general dependence of the values of the samples at one time on the values of the samples at another time [14].

Autocorrelation can be very useful in finding repeated patterns, such as the presence of a periodic signal which has been buried under noise.
The autocorrelation of a continuous-time signal \( y(t) \) can be represented by \( R_{yy}(s) \) and is obtained as

\[
R_{yy}(s) = \frac{1}{T} \int_{-T}^{T} y(t)y(t + s) \, dt
\]

In the above equation, \( 'T' \) is the period of observation.

In case we have a discrete-time signal \( X(k) \), with \( 'N' \) samples, it’s autocorrelation can be obtained as

\[
R_{xx}(m) = \frac{1}{N} \sum_{k=0}^{N-1-m} X(k)X(k+m)
\]

Where \( 'N' \) is the number of sample, \( 'k' \) is the sample number in the original signal and \( 'm' \) is the ‘lag’.

First, the program finds the shifted version of the signal and then it finds the summation as explained above.

### 3.5.3 Signal period

The function is responsible for finding the period of the signal at which the signal repeats itself. It calculates the period from the autocorrelation file 'ACF' that was generated by autocorrelation function. It first finds the threshold that will be use to find the peaks and then the interval between the peaks that will show the rate at which the signal repeats itself.

### 3.6 ECG Peaks Method1 Module

This module contains a public class consists of different functions that are used to find peaks in the electrocardiogram (ECG) signal. Basically, there are two methods of finding peaks. Different leads used different methods to find peaks depend on whether the peaks are upwards or downwards. The first method of finding peaks is based on the upward peaks and is discussed in this section and the second method is based on the downward peaks that has a separate module and is given in the next section named ECG Peaks Method2.

Leads Augmented Vector Foot (aVF), lead-1, lead-2, lead-3, V5 and V6 find peaks based on the first method given in ECG Peaks Method1.
There are several functions in the module that find different kind of peaks; all these functions are discussed in the subsequent sub-sections,

3.6.1 Maximum peak (Threshold)
This function is used to find the maximum Peak value that decides 'Threshold, which is then used in finding R-Peaks. The function also finds the time when the maximum value occurs in a given sample of signal. It initializes a variable and start its value from '0', runs for loop from '0' to the last samples in a given sample of signal to calculate maximum values and the indices when they occur.

3.6.2 'R' peaks
These are highest peaks in an electrocardiogram (ECG) signal. In the previous subsection/function the program defines a value called ‘Threshold’, in this function the program compares all the samples with the value of threshold. If the value of the sample is more than the defined threshold value and is more than the values of its previous and next counterparts then the sample is considered as 'R' peak. To compare all the samples with the value of threshold, the function runs a 'for' loop for all samples and checks if the value of the sample is more than the 'threshold'

All the leads in this module such as Augmented Vector Foot (aVF), lead-1, lead-2, lead-3, V5 and V6 have upward ‘R’ peaks in this module.

3.6.3 Peaks 'P' and 'T'
This function is responsible to find the values and time indices of the 'P' and 'T' peaks. 'T' is the second maximum value after the first maximum peak 'R' and 'P' is the third maximum value in a period of electrocardiogram (ECG) signal.

To find these peaks, the program search for maximum peaks between two ‘R’ peaks. The program start searching for these values '5' samples after the 'R' peak has occurred and stop searching '5' samples before the next 'R' peaks has occurred.

3.6.4 Minimum peaks/values
The last function in the module finds the samples with minimum values in given sampling interval. It finds the values and time-indices of the samples.

The function first sets a value for minimum and then compares the samples with the defined minimum. The sample is considered as the minimum sample if the value sample is less than the value of the given minimum and is less than the value of its
previous and next counterparts. These minimum samples are considered as ‘Q’ peaks of the electrocardiogram (ECG) signal.

3.7 ECG Peaks Method2 Module
This module contains a public class consists of different functions that are used to find peaks in the electrocardiogram (ECG) signal. As discussed in the previous module named ECG Peaks Method1, there are two methods of finding peaks. Different leads used different methods to find peaks depend on whether the peaks are upwards or downwards. The first method of finding peaks is based on the upward peaks and is discussed in its separate section ECG Peaks Method1 and the second method is based on the downward peaks that and is presented in this section.

Leads Augmented Vector Left (aVL), Augmented Vector Right (aVR), V1, V2, V3, and V4 find peaks based on the first method given in ECG Peaks Method2.

There are several functions in the module that find different kind of peaks; all these functions are discussed in the subsequent sub-sections,

3.7.1 Minimum Peak (Threshold)
This function is used to find the minimum Peak value that decides ‘Threshold, which is then used in finding R-Peaks. The minimum peak here is basically the maximum downward value that has a negative value. The function also finds the time when the maximum value occurs in a given sample of signal. It initializes a variable and start its value from '0', runs for loop from '0' to the last samples in a given sample of signal to calculate maximum values and the indices when they occur.

3.7.2 ‘R’ peaks
These are highest downward peaks or the samples with maximum values but with negative sign in an electrocardiogram (ECG) signal. In the previous sub section/function the program defines a value called ‘Threshold’, in this function the program compares all the samples with the value of threshold. If the value of the sample is less than the defined threshold value and is less than the value of its previous and next counterparts then the sample is considered as ‘R’ peak. To compare all the samples with the value of threshold, the function runs a 'for' loop for all samples and checks if the value of the sample is more than the ‘threshold'

All the leads in this module such as Augmented Vector Left (aVL), Augmented Vector Right (aVR), V1, V2, V3, and V4 have upward ‘R’ peaks in this module.
3.7.3 Peaks 'P' and 'T'
This function is responsible for finding the values and time indices of the 'P' and 'T' peaks. 'T' is the second minimum value after the first minimum peak 'R' (first maximum downwards) and 'P' is the third minimum value in a period of electrocardiogram (ECG) signal.

To find these peaks, the program search for minimum peaks between two ‘R’ peaks. The program start searching for these values '5' samples after the 'R' peak has occurred and stop searching '5' samples before the next 'R' peaks has occurred.

3.7.4 Maximum peaks/values
The last function in the module finds the samples with maximum downward values in given sampling interval. It finds the values and time-indices of the samples.

The function first sets a value for maximum and then compares the samples with this defined value. The sample is considered as the maximum sample if the value of the sample is more than the value of the defined maximum and is more than the value of its previous and next counterparts. These maximum downward samples are considered as 'Q' peaks of the electrocardiogram (ECG) signal.

3.8 Read Write Module
This module has a public class and is responsible for reading patients input files and writing to patients output files. The module contains four functions, each has a specific purpose, one is used for reading data from the input files, reading sampling interval, writing headers and writing the actual results to the output files.

A short introduction to each of these module is given here in the below subsections,

3.8.1 Reading Input Files
This function is used to read data from the input files and store the required values in the different variables that are used for calculating the results in other classes/functions.

The function uses a ‘for’ loop to read data from the input file. The ‘for’ reads from the beginning of the file to the end of the file. An array 'x[n]' is used to hold the values that are used in calculating autocorrelation function (ACF). The values holding by ‘x[n]’ are in fixed point integer format.
3.8.2 Sampling Interval
This function is used to read/get the sampling interval. It gets two inputs i.e. 'start' and 'end' which hold the address in the memory location of the start and end of the previous sample. The 'start' sample in the current sampling interval will be the 'end' sample in the previous one. It calls the overloaded arithmetic function that calculates addition. The buffer capacity should be more or equal to '4 seconds' to ensure the correctness of the algorithm.

3.8.3 Writing Headers
This function is responsible to write header for the output files, the files can either be stored in binary or ASCII. The headers are included in the patient’s output files for better understanding.

These headers include maximum number of samples, sampling frequency and buffer capacity.

3.8.4 Saving Results
This function is responsible to save the actual results in the output files. The results include parameters like heart period, heart rate, used threshold and the peaks etc to the output files either in binary or ASCII. ‘ofstream’ is used to provide interface to output files.

There are two new output files added in the electrocardiogram (ECG) analysis system. One contains the summary of the file that are holding the final results and the other keeps the average of ‘Heart Periods’ and of ‘Heart Rates’. These two files are supposed to be sent for diagnosis and we keep the other three detailed files for future analysis.

The method of saving the output/results in the output files are also changed in the new version of electrocardiogram (ECG) analysis system, in the old version the process was flushing the buffer for each line of the output but in the new version we use the full buffer capacity define in ‘ofstream’ and flush the buffer when it is full.
Chapter-4

4. Results and Performance

This chapter presents the performance results and modifications that are made in the input and output processes of the electrocardiogram (ECG) analysis system. In the new system, both the input and output processes are automated and a Linux based script is introduced to automate the process of downloading the patients input data/files from one or more online databases. The system is modified to save the output of the parallel algorithms both in ASCII and binary formats. There are two new output files added in this version of the electrocardiogram (ECG) analysis system. One of the new added output file contains the summary of the old files that are holding the final results and the other keeps the average of ‘Heart Periods’ and of ‘Heart Rates’. Only these two files are supposed to be sent for diagnosis and we keep the other three old detailed output files for future analysis. A log system has been introduced in the new system. When the program runs it creates a log file and saves the processing/execution time lead-wise, patient-wise and in the end it calculates the time taken for processing all the patients in microseconds. The buffering techniques have been changed to reduce the processing/execution time and to get better performance.

The chapter is organized as first presenting the input and output processes automation, how the system saves output files in ASCII and binary, how the output files are organized, log system and increasing performance by modifying the buffering techniques. Examples from real output files and performance results are given wherever is necessary.

4.1 Input and Output Processes Automation

The entire input and output processes to the electrocardiogram (ECG) analysis system have been automated. In addition, a script written in Linux takes care of downloading the patients input files from one or several online databases at the same time. The input and output paths including the path(s) to online databases can be saved in configuration files.

When the program runs, first the Linux script reads the path(s) to online databases from a specific configuration file where these paths are saved. If any path(s) exist, the script downloads the input files from online databases and save them in a temporary memory location on the disk. The script converts the files to .txt format, which is the
only format the program can read the input files. After converting the data files to .txt format, the script saves them to a location specified in input path to the program. Now the files are ready to be read by the program to generate the required results.

The program reads the path specified in the configuration file where the input path is saved. If no path exists or the path to the input files is wrong, the program gives error message and asks to enter the correct path. At this stage, the program gives the user opportunity to either correct the path in the input path configuration file or enter a correct path on the screen. If the input path to the program is correct and exists, the program reads the input files patient-wise and generates the required results. Now the program reads the output path specified in the output path configuration file. If the path to the output directory where the user wants to save the output files does not exists, the program gives error message and asks to enter the correct path. At this stage, the user can correct the path to the output directory. Now the program creates a directory for each patient in the specified parent directory and saves the output files in the relevant patient’s directory.

4.2 Output Files Formats (Binary and ASCII)
The electrocardiogram (ECG) analysis system is upgraded to save the results in both ASCII and binary formats. The binary format is added to make the output files more secure and keeping in mind the transferring of output files through VoIP networks to another location for diagnoses purposes.

When the program runs it asks the user whether to save the output files in ASCII or binary. The program then saves the output files on the location specified in output path configuration file in a format mentioned by the user. All the files are stored in one format either in ASCII or binary.

4.3 Result/Output Files
This section presents the output files of the electrocardiogram (ECG) analysis system. The old system has three output files i.e. One files each for results, autocorrelation and difference vector. There are two new files added in the new system. One contains the summary of the file that are holding the final results and the other keeps the average of ‘Heart Periods’ and of ‘Heart Rates’. These two new files are smaller in sizes and are supposed to be sent for diagnoses and we keep the other three detailed files for future analysis.
Note: All the results given in this document are obtained on a Dell system with Intel® 1.73GHz Core™ 2 Duo processor running Ubuntu 9.10.

Autocorrelation is the cross-correlation of the two same input signals as is explained in details in chapter-3 ‘Parallel Electrocardiogram (ECG) Analysis Algorithms’. The autocorrelation function of the input signal is saved in a separate file. An example from a real output file in ASCII format that contains the autocorrelation function for Lead-I of the ECG signal is given in Figure 3. Each file that contains autocorrelation is approximately 183 KB in size.

![Figure 3: Autocorrelation Function (ACF) of Lead-1](image)

<table>
<thead>
<tr>
<th>ACF</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.59178e</td>
</tr>
<tr>
<td>2.53962e</td>
</tr>
<tr>
<td>2.38856e</td>
</tr>
<tr>
<td>2.15536e</td>
</tr>
<tr>
<td>1.86274e</td>
</tr>
<tr>
<td>1.53435e</td>
</tr>
</tbody>
</table>

There is another output file that contains the difference vector of the input signal or the patients input data. An example from a real output file in ASCII format that contains the difference vector for Lead-I of the ECG signal is given in Figure 4. Each file that contains difference vector is approximately more than 650 KB in size.

![Figure 4: Difference vector of Lead-I](image)

<table>
<thead>
<tr>
<th>DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.00873485</td>
</tr>
<tr>
<td>-0.0217423</td>
</tr>
<tr>
<td>-0.0360133</td>
</tr>
<tr>
<td>-0.0467937</td>
</tr>
<tr>
<td>-0.0519238</td>
</tr>
<tr>
<td>-0.0512037</td>
</tr>
</tbody>
</table>
The old system had one file for final results that contained the detailed results of the electrocardiogram (ECG) analysis system. The file is approximately 5-15 KB long in size and contains both the headers and the actual results. The headers include maximum number of samples, sampling frequency and buffer capacity. The results include other information i.e. heart period, heart rate, threshold used for finding ‘R’ peaks and the peaks in the ECG signal. A sample from an output file for Lead-I starting at sampling interval ‘0’ to ‘5’ in ASCII format is shown in Figure 5.

**Figure 5: Detailed results of Lead-I
Sampling interval ‘0-5’**

<table>
<thead>
<tr>
<th>Max. number of samples</th>
<th>5000 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling frequency (Fs)</td>
<td>1000</td>
</tr>
<tr>
<td>Max. buffer capacity</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Heart period</td>
<td>1.02795</td>
</tr>
<tr>
<td>Heart rate</td>
<td>58.3683</td>
</tr>
<tr>
<td></td>
<td>0.248709</td>
</tr>
<tr>
<td>Number of ‘R’ peaks above threshold</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R Peaks</th>
<th>T Peaks</th>
<th>P Peaks</th>
<th>Q Peaks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Time</td>
<td>Value</td>
<td>Time</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>0.3853</td>
<td>0.191</td>
<td>0.01557</td>
<td>0.213</td>
</tr>
<tr>
<td>0.3841</td>
<td>1.225</td>
<td>0.05468</td>
<td>1.5</td>
</tr>
<tr>
<td>0.4145</td>
<td>2.246</td>
<td>0.01675</td>
<td>2.272</td>
</tr>
<tr>
<td>0.4106</td>
<td>3.274</td>
<td>0.05465</td>
<td>3.298</td>
</tr>
</tbody>
</table>

The new system keeps these three output files shown in table-1, 2 and 3 for future analysis and generates two new output files that are smaller in sizes and that are supposed to be sent through the network. The file that contains the summary of the results shown in Figure 3 is presented in Figure 6 and contains the summary or average of all the information/data that is saved in the detailed output file. The table contains the contents from the output file for lead-I of the ECG signal.
The other new file introduced in the electrocardiogram (ECG) analysis system contains the average of ‘Heart Periods’ and of ‘Heart Rates’ and is given in Figure 7. The table has data from the output file for lead-I of the ECG signal, which is the average of ‘Heart Periods’ and of ‘Heart Rates’ in Figure 6.

### Figure 6: Summarized and average results

<table>
<thead>
<tr>
<th></th>
<th>Max. number of samples</th>
<th>Sampling frequency (Fs)</th>
<th>Max. buffer capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5000 samples</td>
<td>1000</td>
<td>5 seconds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start Sample</th>
<th>Heart Period</th>
<th>Heart Rate (beats/min)</th>
<th>No. of ‘R’ Peaks</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.02795</td>
<td>58.3683</td>
<td>5</td>
<td>0.248709</td>
</tr>
<tr>
<td>5</td>
<td>1.01196</td>
<td>59.2907</td>
<td>5</td>
<td>0.255103</td>
</tr>
<tr>
<td>10</td>
<td>1.034</td>
<td>58.0272</td>
<td>5</td>
<td>0.255103</td>
</tr>
<tr>
<td>15</td>
<td>1.25098</td>
<td>47.9625</td>
<td>5</td>
<td>0.262489</td>
</tr>
<tr>
<td>20</td>
<td>1.05194</td>
<td>57.0374</td>
<td>5</td>
<td>0.262489</td>
</tr>
<tr>
<td>25</td>
<td>1.03894</td>
<td>57.7511</td>
<td>5</td>
<td>0.262489</td>
</tr>
<tr>
<td>30</td>
<td>1.021</td>
<td>58.7661</td>
<td>5</td>
<td>0.262489</td>
</tr>
</tbody>
</table>

### Figure 7: Average of heart period and heart rate

<table>
<thead>
<tr>
<th>Heart Period</th>
<th>Heart Rate (beats/min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.07312</td>
<td>56.1954</td>
</tr>
</tbody>
</table>

### 4.4 Log Component

To evaluate the performance of any system or product, it is always desirable to add a log component to the system. The log component or system makes it easy to improve the performance of the system by first evaluating it and then make the necessary improvements.
A log component has been added in the new electrocardiogram (ECG) analysis system. When the program runs, it creates a log file and saves the processing/execution time lead-wise, patient-wise and in the end it saves the total time taken to process all the patients. The log file can be saved either in ASCII or binary format. All the processing/execution times are saved in micro seconds. When the program processes a lead or a patient, it takes the start time and then note the end time of the lead or patient and calculates the difference. An example of log file for a single patient including all the leads is shown in Figure 8.

**Figure 8: Electrocardiogram (ECG) analysis system log component**

<table>
<thead>
<tr>
<th>Lead/Patient</th>
<th>Processing Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient-1</td>
<td></td>
</tr>
<tr>
<td>aVF</td>
<td>2.80979</td>
</tr>
<tr>
<td>aVL</td>
<td>2.632</td>
</tr>
<tr>
<td>aVR</td>
<td>2.7029</td>
</tr>
<tr>
<td>Lead-I</td>
<td>2.97445</td>
</tr>
<tr>
<td>Lead-II</td>
<td>2.68148</td>
</tr>
<tr>
<td>Lead-III</td>
<td>2.66023</td>
</tr>
<tr>
<td>V1</td>
<td>2.68833</td>
</tr>
<tr>
<td>V2</td>
<td>2.69947</td>
</tr>
<tr>
<td>V3</td>
<td>2.65829</td>
</tr>
<tr>
<td>V4</td>
<td>2.65012</td>
</tr>
<tr>
<td>V5</td>
<td>2.67978</td>
</tr>
<tr>
<td>V6</td>
<td>2.69015</td>
</tr>
<tr>
<td><strong>Total Patient-1</strong></td>
<td><strong>32.8384</strong></td>
</tr>
</tbody>
</table>

**4.5 Performance Evaluation**
The performance evaluation and increasing the performance of the electrocardiogram (ECG) analysis system was the important part of the thesis project. In order to increase the performance two main goals were set. The first was to divide the system in smaller
parallel programs/algorithms in order to be processed on individual cores, which is discussed in details in chapter 3 ‘Parallel Electrocardiogram (ECG) analysis Algorithms’. The second goal was to make the necessary changes in the output system to minimize the processing/execution time of the system. This section presents the changes that are made in saving the results and output files, in order to decrease/minimize the processing/execution time of the system.

C++ objects ‘ostream’ are stream objects and are used to write and format output as sequences of characters [15]. In the electrocardiogram (ECG) analysis system, the ‘ofstream’ has been used as the output stream to write to the output files. When a C++ program writes to an output file, first it saves the contents to an output stream buffer and then writes to the output files what is written on the buffer. There are two methods to deal with output stream buffers in C++. The first method is to use the public member ‘flush’ and the second method is the use of ‘endl’. ‘flush’ is used to synchronize the buffer associated with the stream to its controlled output sequence. This effectively means that all unwritten characters in the buffer are written to its controlled output sequence as flushed [16].

The second method of flushing the output stream buffer is the use of ‘endl’. ‘endl’ inserts a new-line character and additionally, for buffered output streams, it also flushes the buffer and writes all unwritten characters in the buffer to the output file [17].

As the output files in the electrocardiogram (ECG) analysis system are written in a way that there are thousands of output lines specifically in the output files where the autocorrelation function and difference vector are saved. In the old ECG analysis system, the program was flushing the buffer after writing each line of output to the output stream buffer, which was adding loads of interaction between output stream buffer and the output files and in the result the system was very slow. The new system has updated to write the contents to the output stream buffer and flush the buffer whenever the buffer is full.

Here in Figure 9, we present the performance comparison of the old and new versions of electrocardiogram (ECG) analysis system. The table contains data from log files for a single patient with all the twelve ECG leads.
Figure 9: Performance comparison of old and new versions of Electrocardiogram (ECG) analysis system

<table>
<thead>
<tr>
<th>Lead/Patient</th>
<th>Processing Time (seconds)</th>
<th>Performance Improvement (x times)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>New System</td>
<td>Old System</td>
</tr>
<tr>
<td>aVF</td>
<td>2.63672</td>
<td>10.0711</td>
</tr>
<tr>
<td>aVL</td>
<td>2.64586</td>
<td>10.7307</td>
</tr>
<tr>
<td>aVR</td>
<td>2.66034</td>
<td>11.0048</td>
</tr>
<tr>
<td>Lead-I</td>
<td>2.97445</td>
<td>11.0395</td>
</tr>
<tr>
<td>Lead-II</td>
<td>2.68148</td>
<td>10.9318</td>
</tr>
<tr>
<td>Lead-III</td>
<td>2.66023</td>
<td>10.8102</td>
</tr>
<tr>
<td>V1</td>
<td>2.67422</td>
<td>10.7901</td>
</tr>
<tr>
<td>V2</td>
<td>2.65131</td>
<td>10.8077</td>
</tr>
<tr>
<td>V3</td>
<td>2.66307</td>
<td>10.8278</td>
</tr>
<tr>
<td>V4</td>
<td>2.65373</td>
<td>10.867</td>
</tr>
<tr>
<td>V5</td>
<td>2.67059</td>
<td>10.9253</td>
</tr>
<tr>
<td>V6</td>
<td>2.658</td>
<td>10.9672</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>32.23</strong></td>
<td><strong>129.775</strong></td>
</tr>
</tbody>
</table>

Figure 10 shows the performance comparison of the old version of Electrocardiogram (ECG) analysis system and the new version. The graph shows that the performance of the new system has been increased by approximately more than 4 times (403%), which means the execution/processing time of the new electrocardiogram (ECG) analysis system has been decreased by approximately one-fourth.
Figure 10: Electrocardiogram (ECG) Analysis based on Autocorrelation Function
Chapter-5

5. Conclusion and Future Work

5.1 Conclusion
An Electrocardiogram (ECG) system with paralleled algorithms designed for multi-core architecture has been presented. The system has smaller individual and paralleled algorithms/modules. Each module is supposed to perform its own task and is supposed to be processed by an individual core/processor in a multi-core environment. The system provides remote and real-time computation of Electrocardiogram (ECG). Thus, it reduces the doctor’s/nurse’s work-load and the cost of hospitalization by sensing heartbeats, processing the data in real-time at patient’s location and sending the required results to the doctor/nurse. All the input and output processes to the system are made automated, thus reducing operator’s work-load. The system reads link to the patient’s input files from a configuration file and saves the results on a specified location. The system takes care of downloading patient’s input files from one or several online databases and converts them to a format suitable for system input. The system gives the user flexibility to either save the results in ASCII or binary formats. A log component adds the functionality to calculate the processing/execution time both per lead and per patient in micro-seconds. By upgrading the buffering techniques used in the older version of ECG analysis system [5], the performance of the system has been increased by approximately more than 4 times (403%) and thus the new system provides real-time processing of Electrocardiogram (ECG) signal.

5.2 Future Work
The Electrocardiogram (ECG) analysis system presented in this article is based on Autocorrelation function, a similar solution based on Fast Fourier Transform (FFT) is desirable in order to compare the performance and correctness of the results generated by both system. It is desirable to test the presented software solution on a real multi-core environment, which was not possible to accomplish due to the scope of the project and time limitation.
Acknowledgments

I am heartily thankful to my supervisor, Dr. Iyad Al Khatib, whose guidance and support from the initial to the final level enabled me to accomplish the project.

I also offer my regards to all of those who supported me in any respect during the completion of the project specially my friend, Mr. Ejaz Ali, for his constant encouragement and motivation.
6. References


[7]. James W. Grier: eHeart: An Introduction to ECG/EKG: Department of Biological Sciences, North Dakota State University, Last accessed at: http://www.ndsu.edu/pubweb/~grier/eheart.html, 08 June 2011

[8]. James W. Grier: A brief introduction to the standard 12-lead ECG (EKG): Department of Biological Sciences, North Dakota State University, August 2008, pp.4-20


