TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Thomas Johann Seebeck Department of Electronics

IEE70LT

Arbind Kumar Rimal IVEM144955

POWER ESTIMATION OF FPGA ARCHITECTURES FOR A WIRELESS HEART-RATE MONITORING SYSTEM BASED ON COMPRESSED SENSING

Master’s Thesis

Supervisor: Yannick Le Moullec
PhD
Senior Researcher

Tallinn 2016

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

IEE70LT

Arbind Kumar Rimal IVEM144955

FPGA ARHITEKTUURIDE VÕIMSUSTARBE HINDAMINE HÕREDAL SEIREL PÕHINEVA SÜDAMELÖÖGISAGEDUSE JUHTMEVABA SEIRESÜSTEEMI JAOKS

Magistritöö

Juhendaja: Yannick Le Moullec
Doktorikraad
Vanemteadur

Tallinn 2016

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Arbind Kumar Rimal

19.01.2017

ACKNOWLEDGEMENT

I would first like to thank my thesis supervisor Yannick Le Moullec, PhD, Senior Researcher of Faculty of Information Technology/Thomas Johan Seebeck Department of Electronics at Tallinn University of Technology. The door to Prof. Moullec office was always open whenever I was into a trouble or had any question about my writing or practical work; he helped me through out and guided me in right direction whenever he thought I needed it. His guidance, words of encouragement and positive attitude and consultation were inspiring. You will always be greatly appreciated.

I would like to display my appreciation to all my friends who supported and incented me to strive towards my goal.

Finally, a special thanks to my parents. Words cannot express how grateful I am to them for all their support morally and sacrifices that they made on my behalf. Your love prayers and supports for me was what sustained me thus far. I am extremely grateful for the opportunities they have given me to further my education. May God blessing in their life never cease.

Abstract

Developments in wireless sensor networks have helped healthcare medical systems to be advanced with many new opportunities in sports training, smart hospitals and nursing homes, medical surveillance for aged people and many more. The application considered in this MSc thesis is a wireless heart rate monitoring system. The focus is on the evaluation of the power consumption of various architectural design implemented on an FPGA platform.

Firstly, this thesis introduces an existing system for such an application. An analysis is performed, including 1) algorithmic aspects, i.e. QRS detection on the electrocardiogram signal and compressed sensing that allows reducing the amount of data to be transmitted wirelessly (to reduce radio activity and thus energy consumption), and 2) architectural aspects, i.e. different structures with various parallelism levels for the compressed sensing block.

Then, this thesis discusses elements of power consumption on FPGA as well as a method for estimating the power consumption of the existing architectures on a Cyclone IV FPGA platform by means of Altera Quartus and ModelSim tools.

Subsequently, various architectures are synthesized for the Cyclone IV FPGA and the synthesis results are presented. Finally, power estimation is performed for three different example architectures (full_para_N4_M2, para_4_2_wo_pipe, and Generic_semi_para). The power estimates are then compared; the results show that one of the architectures has a lower power consumption than the others but requires many more resources. The results also show that pipelining allows higher maximum frequencies but requires more resources and at the same time does not affect power consumption significantly.

This thesis is written in English and is 56 pages long, including 6 chapters, 16 figures and 4 tables.

Annotatsioon

FPGA arhitektuuride võimsustarbe hindamine hõredal seirel põhineva südamelöögisageduse juhtmevaba seiresüsteemi jaoks

Arengud traadita andurite võrkude osas on aidanud tervishoiu meditsiinilise süsteemide arendamist ja pakkunud palju uusi võimalusi sportliku treeningu, tarkade haiglate ja hooldekodude, eakate meditsiinilise järelevalve jne osas. Käesolev magistritöö on juhtmeta südame löögisageduse seiresüsteemist.Keskendutakse erinevate FPGA-põhaliste arhitektuuride hindamisele energiatarbe järgi.

Töö alguses tutvustakse ühte olemasolevat süsteemi. Analüüsitud on muuhulgas:1) algoritmilisi aspekte, st QRS avastamise elektrokardiogrammi signaali ja tihendatud sensori-signaale, mis lubab vähendada juhtmevabalt edastatavate andmete hulka (vähendada raadio aktiivsust ja seega energiakulu) ja 2) arhitektuurilisi aspekte, st erinevate struktuuride erinevaid parallelismi tasemeid kokkusurutud kaugseire blokeerimiseks.

See väitekiri käsitleb ka FPGA elementide voolutarbimist, samuti energiatarbe hindamist olemasolevate struktuuride kohta Cyclone IV FPGA platvormi abil Altera Quartus ja Modelsim tööriistu kasutades. Seejärel sünteesitakse erinevate arhitektuuride jaoks Cyclone IV FPGA. Sünteesi tulemused on esitatud. Lõpuks võimsustarbe hindamine viiakse läbi kolmes erinevas näiids arhitektuurile (full_para_N4_M2, para_4_2_wo_pipe ja Generic_semi_para).Võimsus hinnanguid on võrreldud. Tulemused näitavad, et ühtedel arhitektuuridel on väiksem energiatarve kui teistel, kuid nõuavad palju rohkem ressursse. Tulemused näitavad ka, et konveier lubab kõrgemat maksimaalset sagedust, kuid nõuab rohkem ressursse ja samal ajal ei mõjuta energiatarve oluliselt.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 56 leheküljel, 6 peatükki, 16 joonist, 4 tabelit.

List of abbreviations and terms

CS	Compressed Sensing
ECG	Electrocardiogram
FPGA	Field Programmable Gate Array
FSMD	Finite State Machine With Data Path
HR	Heart Rate
HRMs	Heart Rate Monitors
HRV	Heart Rate Variability
LEs	Logic Elements
PC	Personal Computer
RTL	Register Transfer Level
WSN	Wireless Sensor Network

Table of contents

1 Introduction 12
    1.1 Starting Point of this MSc Thesis 14
    1.2 Problem Statement 15
2 Background 17
    2.1 Electrocardiogram 17
    2.2 The Need for Compressing the ECG Signal 19
3 Analysis of the Existing System 21
    3.1 System Overview 21
    3.2 Analysis of the Transmitting Side 22
        3.2.1 Pan and Tompkins QRS Detection Algorithm 23
        3.2.2 Noises Types 25
        3.2.1 Detection of the QRS 26
        3.2.2 Compressed Sensing 26
        3.2.3 Sparsity and Compressible Signals 27
    3.3 Heart-Rate Monitoring Examples for Reconstruction of Sparsified ECG Test Signals 29
    3.4 Analysis of the Existing Architectural Design 30
    3.5 RTL Implementation Results 32
4 Method for Estimating the power Consumption on FPGA 35
4.1 Power Overview 37
    4.2 Standby Power 38
    4.3 Dynamic Power 39
    4.4 I/O Power 39
    4.5 PowerPlay Early Power Estimator Overview 40
    4.6 Estimating Power Consumption 40
        4.6.1 Pros and Cons of Power Estimation before Design of FPGA 40
        4.6.2 Estimating Power Consumption While Creating the FPGA Design 41

4.6.3 Pros and Cons of Power Estimation if the Design of FPGA is partially complete 41
4.6.4 Estimating Power Consumption after Completing the FPGA Design 41
4.7 Thermal Power 42
5 Result and analysis 43
5.1 High level synthesis results of different architecture (with the various levels of parallelism) and their comparisons.. 43
6 Conclusion.. 52
References 54

List of figures

Figure1. System overview of many-to-one HRM [3]..14

Figure 2. A healthy ECG and its characteristics [7]...18

Figure 3. Overview of transmitter and receiver end of the HR monitoring [2].......................21

Figure 4. Overview of the inputs, outputs and contents of the digital signal processing part of the transmitting side [3]…………………………………………………………………....22

Figure 5. Steps within the R waves extraction algorithm [3]..24

Figure 6. Three examples of noises on the ECG signal [13]..25

Figure 7.Graphical representation of the CS [3]..28

Figure 8. Uncompressed, compressed and reconstructed version of the sparsified ECG test signals [3]...29

Figure 9. Total execution time for the different schedules as a function of N [3]………..…31

Figure10.The RTL implementation of the modified semi-parallel architecture [3]…………33

Figure 11. The RTL implementation for the full-parallel architecture [3]…………………...34

Figure 12. Method for FPGA power estimation with Altera tools..36

Figure 13. Relationship between junction temperature and standby power [6].......................38

Figure 14. Overview of the RTL generic semi parallel architecture.......................................45

Figure 15. RTL implementation of generic semi parallel architecture...................................46

Figure 16. RTL implementation of full parallel architecture..47

List of tables

Table 1: Synthesis results for different types of functional units [3]…………………..31

Table 2: FPGA synthesis results and comparison of different degrees of parallelism....44

Table 3: Power estimates for different architecture in power aggressive mode at 100 MHz…………………………………………………………………………………48

Table 4: Power estimates for different architecture in power aggressive mode at 100 MHz………………………………………………………………………………….50

1 Introduction

Over the past 20 years, the development of heart rate monitors (HRMs) has evolved rapidly. Heart rate monitoring has wide range of application, i.e. sports, biomedical, self-monitored health and fitness for common people, emergency condition monitoring on hospitals, health monitoring for elderly peoples at home and many more. The development of new HRMs makes that they are increasingly used in sports as a training tool for the players to monitor and increase their strength, stamina and intensity of exercise. In addition with the increasing demands on sports, HRM research has recently focused on heart rate variability (HRV). HRV is actually evaluated by examining beat-to-beat variations in normal R-R intervals [1]. Increases in HRV is directly dependent with lower morality rate and thus affects human regardless of age and sex. HRM is also being used by medical experts in hospitals to diagnosis the patient mainly for the cardiac diseases which are dependent on HRV and which can be diagnosed from heart rate (HR) measurements [2].

In sports, to obtain the maximum benefit from training and to restrict the over training, it is needed to measure the intensity that an individual puts in during their exercise. In racing, only speed is not the accurate factor of exercise intensity so there should be some alternative to find the intensity in training or competition. The power output generated during exercise may be the direct factor but instead of calculating the consumption of power output, heart rate is easier to monitor and measure comparatively [26].

It is noticed that during graded exercise, the studies [1] and [27] show that the HRV decreases constantly till reasonable intensities and after that it stabilises, which shows that the trained individuals have higher HRV. So the HR sensing is employed on professional sport players, especially athletes, to determine their exercise and workout intensity. Thus the heart rate monitors are used for maximum utilization of their training session for the proper preparation of professional and maximum benefits is achieved [1]. In these days heart rate monitors are mainly referred by the medical professionals to monitors their patients so that the person knows they are within their target heart rate zone.

Because of the high expansion of the use of heart rate monitors, several products (e.g. wearable smart shirt, AMON: a wearable multiparameter medical monitoring and alert system) are available on the market using wireless technology for the transmission of data collected by sensors on the human body and to analyses the data. However, this kind of system is generally limited to one-to-one monitoring. This means a single communication channel system at a time i.e. one person at a time is connected to one computing device which can be on a PC or a smartphones. And in the case of multiple persons (e.g. a group of people in a team) that need simultaneous supervision of a heart rate at a time, a wireless sensor network (WSN) approach can be deployed. The data transmitted from the persons can be collected through the series of sensor nodes which can be acquired and further processed through means of transmission protocol. The data can be analysed at real-time and monitored on PC screens.

An illustration of the concept can be seen in Figure 1[3].The data is collected from the series of sensor nodes (each sensor nodes is related with a specific person profile) and further processed into the signal processing block and for this case it is assumed to be black box. After that the collected data is transmitted to the centralized server which is connected to WSN- nodes through the means of transmission protocol. Then, the real time monitoring of HR can be received on a monitoring screen [1].

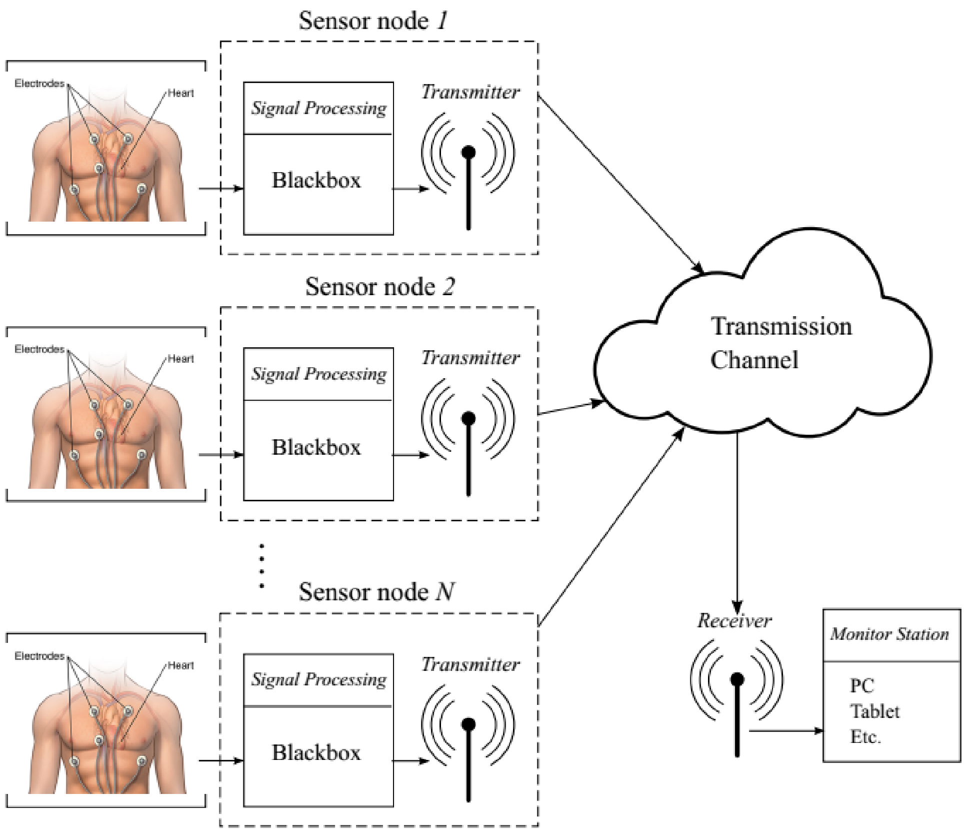


Figure 1. System overview of many-to-one HRM [3].

1.1 Starting Point of this MSc Thesis

One such above-mentioned WSN-based, multi-user HRM system has been previously designed and explored in the project “Architectural Design Space Exploration of an FPGA-based Compressed Sampling Engine: Application to Wireless Heart-Rate Monitoring”- by Mohammad EI-Sayed and Soren Lund [3] under the supervision of Peter Koch (Department of Electronics Systems, Aalborg University, Denmark) and Yannick Le Moullec, T.J. Seebeck Department of Electronics, Tallinn University of Technology.

In [3], two important blocks in the WSN- node i.e., the so-called QRS detection in the electrocardiogram (ECG) and compressed sensing (CS) encoder and its corresponding reconstruction decoder have been designed and simulated for the Altera Cyclone III FPGA platform using the-Quartus design tool. Furthermore, various hardware architectures for the CS engine have been developed at the register transfer level (RTL).

The main aim in [3] was to explore the design and show how parallelism affects execution time at RTL level. As a result, a prototyping solution (partial and full-parallel architectures) has been simulated onto the above-mentioned FPGA platform.

1.2 Problem Statement

Gathering the information from [3] and the preliminary literature survey, a topic is put forward in order to guide the direction of this project.

In particular, it is worth noting that whereas minimizing power consumption has been taken care of during the design of the architectures proposed in [3], no evaluation regarding the power consumption on the FPGA implementation has been reported. Given that power consumption is critical in wearable WSNs (that are typically battery-powered), this MSc thesis can be seen as a continuation of the above work, the overall purpose being to estimate and analyse the power consumption of the existing architectures.

The problem statement for this MSc thesis is two-fold and is formulated as follows:

"How to estimate the power consumption of FPGA-based HRM nodes and what are those estimates for the full parallel, full parallel without pipeline and generic semi parallel architectures proposed in [3]?".

To address this problem, I have performed the following tasks:

- Analysed the proposed overall system, including gaining a basic understanding of how an HR signal can be pre-processed and how to reduce the amount of data to be transmitted by means of CS;
- Analysed the existing architectures, including gaining an understanding of how parallelism affects execution time;
- Prepared a method for estimating the power consumption, including performing high-level synthesis, implementing a method for power estimation (configuration of the needed tools, test-bench creation, etc.);

- Performed the experiments to obtain power consumption estimates of the HRM for full parallel, full parallel without pipeline and generic semi parallel architectures on the Cyclone IV;

The rest of this MSc thesis is organized as follows: Chapter 2 presents some background information about the electrocardiogram (ECG) and the need for compressing the ECG signal. Chapter 3 analyses the existing system and architectures. Chapter 4 presents the method used to estimate the power consumption of the architectures on the Cyclone IV FPGA. Chapter 5 presents the result and their analysis. The final chapter concludes the work.

2 Background

2.1 Electrocardiogram

ECG is the most commonly used process for heart rate monitoring. ECG records the electrical activity of the heart through electrodes that are placed on the skin over a period of time, and from it the contraction (depolarization) and relaxation (repolarization) of the heart can be seen and measured in the form of waves [6]. ECG is widely used in the field of medical industry around the world for identifying and continues monitoring of the several heart diseases and disorders.

The overall objective of ECG is to obtain information about the function, structure and condition of the heart. It helps in the detection of several diseases such as myocardial infarction (heart attack), suspected pulmonary embolism, a cardiac murmur, cardiac stress testing and so on. The normal ECG signal reading can be observed in Figure 2[7]. The ECG is explained in terms of five intervals; P, Q, R, S and T. These intervals describes a deflection which means heart rate, rhythm and morphology. A normal ECG wave of general heart beat consists of P wave, a QRS complex and T wave. P wave describes the sequential depolarization of the left and right atria. The QRS reflects to depolarization of left and right ventricles. It lasts till 70- 110 milliseconds in general; the heartbeat has the largest amplitude of the ECG waveform which can be clearly noticed in Figure 2. The T wave represents the ventricular repolarization and about 300 milliseconds extension after QRS wave complex. The positioning of T wave is mainly dependent on the heart rate [8].

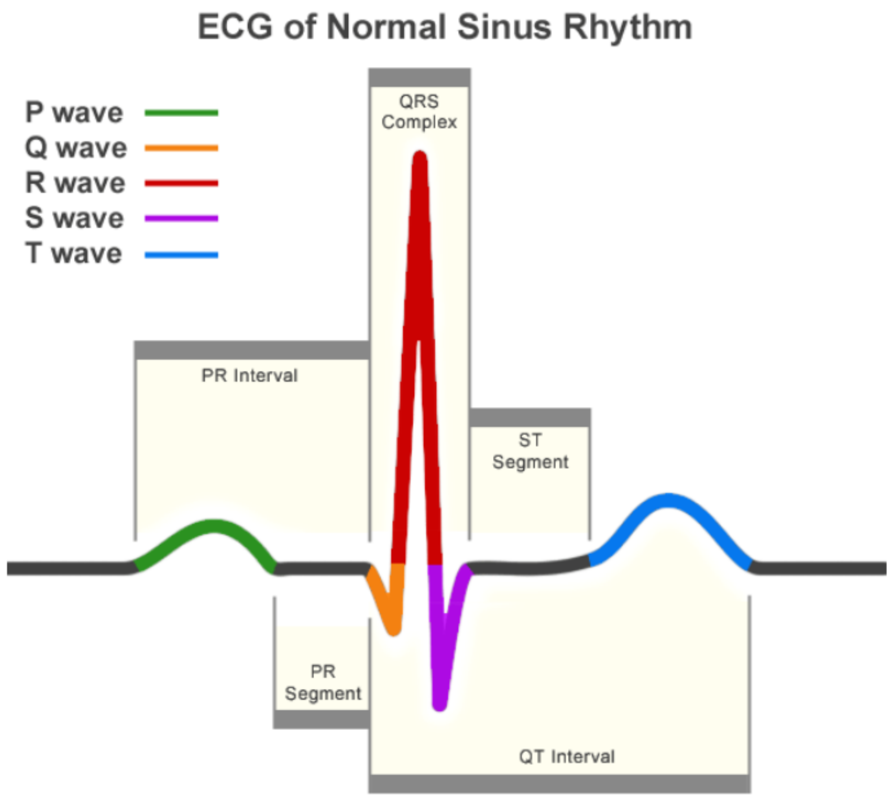


Figure 2. A healthy ECG and its characteristics [7].

In the project that this MSc thesis expands, the main task was to measure the heart rate of an individual but not to detect the different irregularities within the ECG itself. As explained in [3], this means that the detection of the heart rate is possible without cardiac arrhythmias and others different conditions which causes an irregular heartbeat. Few examples of irregularities are variation(s) in the beat-to-beat interval, where beats are completely bypassed, small or great morphing of the P, Q, R, S and T intervals or heart beats where one or more of intervals are not present [6][9].

As also explained in [3], the major task was to measure the heart rate (HR). Reading several articles and relevant literature which study about the methods of measuring HR, it is noticed that QRS complex interval of the ECG is mostly used [10] [4] [11]. This is because, QRS has a unique appearance and is easily differentiated from other intervals in the ECG because of its high amplitude. Others intervals have generally low amplitude. Generally the overall amplitude of the QRS interval will not change drastically when compared to the rest of the ECG intervals, even in case of heart disease is detected or present. This characteristic of QRS intervals is a suitable selection for tracking the HR, because of its higher amplitude to the remaining peaks, which is easier to detect and the rate of depolarization is also a direct measure of the HR. When the QRS intervals are detected, from here R waves can be extracted, and the rate between them can be directly calculated into the measure of HR, normally represented in Beats-per-minute (BPM) [1].

2.2 The Need for Compressing the ECG Signal

Designing a system such as shown in Figure 1 is a difficult task to perform efficiently and emphasis on the inter-related parameters of energy consumption and bandwidth is required. Power (and energy) consumption is a main aspect in such a system because the sensor nodes are mainly operated by batteries and if they consume a lot of energy, the batteries need to be recharged on regular basis by the user or network provider.

The Wireless technologies such as Wi-Fi support high data-rates but are quite energy-hungry, so it is preferable to rely on lower power, low-data-rate technologies such as 6LoWPAN or Bluetooth [28]. Because of this low data-rate, it is needed to consider the different ways of reducing the amount of data to be transmitted. Therefore, it has been proposed to create a sparse representation of the sampled HR signal and further process the data with a compression technique which can result in creating a low energy system.

Compressing the data which are transmitted into the system could help to deal with the above-mentioned issue which results in less radio activity on the sensors and the traffic gets decreased in the network.

To achieve this, it has been proposed to use CS, as mentioned in Section 1.1. CS is used for data reduction which makes it possible to under sample signals at frequencies below the Nyquist-Shannon rate [3]. For heart rate monitoring, the QRS interval (normally high in amplitude) is mainly extracted during the processing because this interval helps to identify the heart rate and because its nature makes it differ from the other and is easily recognisable. Firstly, the QRS detection algorithm is used for the detection of the R waves. These R waves can be represented as sparse which can be further applied for compressed sensing. Secondly, CS helps to reduce the number of samples that are needed to represent the signal. The amount of the data is reduced by this process which helps in reduction of power consumption during the transmission of data [1]. The principle of CS is explained in Section 3.2.4.

Pan and Tompkins QRS detection algorithm (introduced in Section 3.2.1) can be applied to FPGA based hardware implementation. This algorithm is widely used in biomedical system. There are several uses for a dependable QRS recognition algorithm among which computer interpretation of the 12-lead ECG is the most accepted technique. Coronary care units use arrhythmia monitors which are currently under development for ambulatory patients which analyses the ECG in real time [31]. The other widely used one is Holter tape recording which need a Holter scanning instrument that includes a QRS detector to analyse the tapes more efficiently than real time. In case of false detection, it results in unnecessarily transmission of data or requires extremely large memory to store all ECG segments which are captured unnecessarily [4].

So, an accurate QRS detector is the vital function of ECG devices. QRS detection is a difficult task to perform because of the physiological variability of the QRS complexes and also because of the different noises which are present in ECG signals. An algorithm for the detection of QRS is illustrated and explained in Sections 3.2.1 and 3.2.3**.** This processing uses different blocks; i.e., filtering, integration, differentiation and intelligent thresholding. This algorithm’s input is the ECG signal and it results in a sparse representation of the ECG indicating the position of the R waves which is also explained briefly in Section 3.2.5[4][5].

3 Analysis of the Existing System

The purpose of this chapter is to analyse and understand the algorithmic and architectural design proposed in [3] for the real time wireless heart rate monitoring system which compresses the heart rate signal by means of Pan and Tompkins QRS detection and CS. For this, an understanding of how to create a sparse representation of the sampled HR signal is needed as well as an understanding of the CS for further compression, which can result in creating a low energy system.

3.1 System Overview

In this section an overview of the HR monitoring system proposed in [3] is presented in Figure 3. The analysis of the system is performed in order to get a better understanding of what kind of pre-processing is required, depending on the various kind of noise that occur in the ECG. In addition, the QRS detection technique is used for the extraction of the R wave's location, which is discussed afterwards on the introduction to CS section.

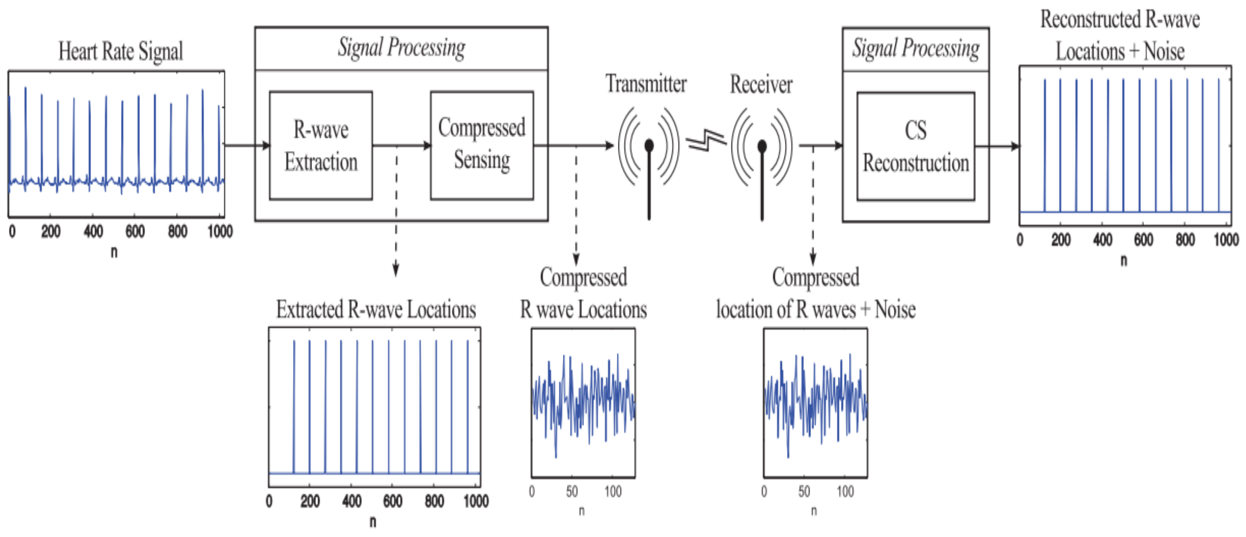

Heart Rate Signal
Signal Processing
R-wave Extraction
Compressed Sensing
Transmitter
Receiver
Signal Processing
CS Reconstruction
Reconstructed R-wave Locations + Noise
Extracted R-wave Locations
Compressed R wave Locations
Compressed location of R waves + Noise
n

Figure 3. Overview of transmitter and receiver end of the HR monitoring [2].

The input in this case is the ECG signal which is as shown in Figure 2. The input ECG signal is passed through the signal processing block, where the extraction of R waves is performed. In this system, the signal processing block of a WSN- node consists of two major blocks; R-wave extraction and CS; its target is to facilitate the system with energy-efficient hardware realization.

The main concepts of extraction of R waves is to extract the location of the R waves, so that the extracted R waves can be represented as a sparse signal. This enables the sparse signal to be later-on subject to CS, which is the next task [2]; the functionality of CS is described Section 3.2.4.

The CS block uses the extracted R waves signal as an input and decreases the number of samples needed to represent the signal. It is performed for decreasing the amount of signal data which should be transferred wireless in order to reduce the consumption of power during the process of transmission.

3.2 Analysis of the Transmitting Side

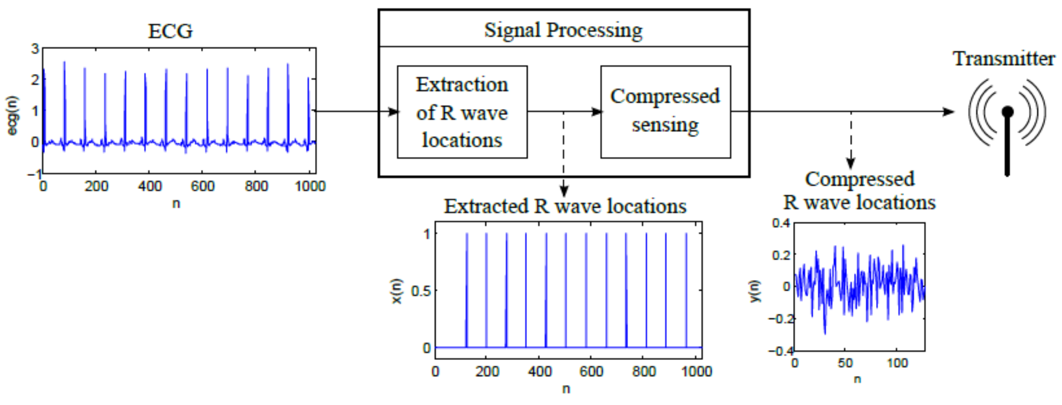

ECG
ecg(n)
n
Signal Processing
Extraction of R wave locations
Compressed sensing
Transmitter
Extracted R wave locations
x(n)
Compressed R wave locations
y(n)

Figure 4. Overview of the inputs, outputs and contents of the digital signal processing part of the transmitting side [3].

In this section the analysis of the system is carried out with the understanding of pre-processing needed into the system based on the noises types which may occurs with the ECG. The major concept of QRS detection algorithm to extract the location of the R waves is discussed. Afterwards the concept of CS is explained, including a discussion of how CS relies on the given signal which is represented as sparse.

3.2.1 Pan and Tompkins QRS Detection Algorithm

Pan and Tompkins QRS Detection Algorithm is used for the detection of the QRS interval of an ECG signal. This algorithm is suitable for all kind of ECG signals without the use of manual algorithm measurement. This is possible because of the adaptive behaviour of the detection algorithm. It uses a dual-threshold technique, search-back technique which is used for finding missing peaks and a working RR-interval estimation for the detection of the irregularities [32].

This detection method uses the functionalities of filtering, differentiation, integration and thresholding. The procedure of the steps within the algorithm can be seen in Figure 5.

Preprocessing is done in order to make the processing on QRS complexes detection easier and acknowledged. The blocks used in the Preprocessing of the ECG are contained within the striped box as in Figure 5.

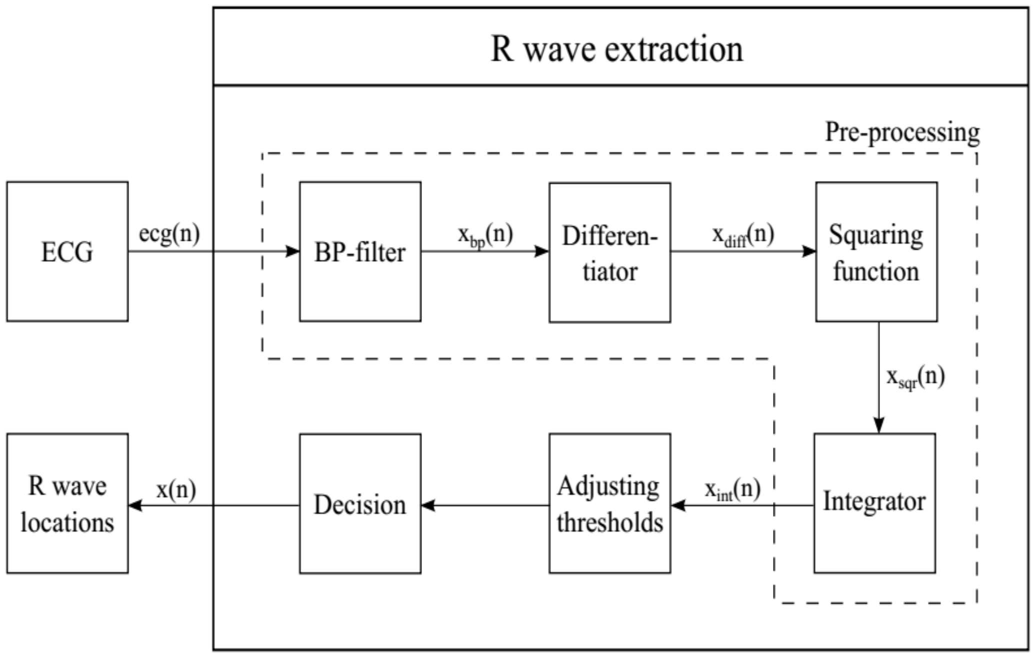


Figure 5. Steps within the R waves extraction algorithm [3].

The main reason for using the band pass-filter is to attenuate the noise that is imposed onthe ECG; it helps to improve the signal-to-noise ratio, and so make it easier to detect the QRS complexes within the signal. The band-pass filter consists of a high-pass filter and low-pass filter in form of cascade. The differential equation of the filters are explained in [11].

The final steps in the preprocessing is integration which results in obtaining the waveform information and the slope of the R waves.. Its mathematical final results can be seen in Equation (1).

$$X_{\mathrm{int}}(n) = \frac{1}{N}\left[X_{sqr}(n-(N-1)) + X_{sqr}(n-(N-2)) + \cdots + X_{sqr(n)}\right] \tag{1}$$

where X is the input signal and N is the size of the sliding window (selected on basis of the sampling rate).

3.2.2 Noises Types

In this section, and also discussed in [3], the different types of noises that occurs during the processing of ECG are studied. This study is mainly based on [13]. Several of the existing ECG analysis system requires noise-free digitized ECG. The data that are corrupted by the relative noise should be filtered or discarded. For the detection of noise, ECG quality assurance not only requires the software noise detection technique or human, but it can result into the major loss of significant data as well. Filtering the data itself can alter the signal and can require considerable computational overhead. These problems are vital for the design study of real-time HRM monitoring applications.

ECG signals can be corrupted by different types on noise. These are:

- Power Lines Interference;
- Electrode Contact Noise;
- Motion Artifacts;
- Muscle Contractions [EMG];
- Baseline Drift and ECG Amplitude Modulation with Respiration;
- Electrosurgical Noise;
- Noise Generated by Electronic Devices Used in Signal Processing;

Three examples diagrams of different types of noise and how they differ from original ECG signal are shown in Figure 6.

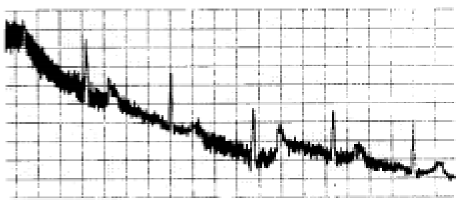


Figure 5(a):Power Lines Interference

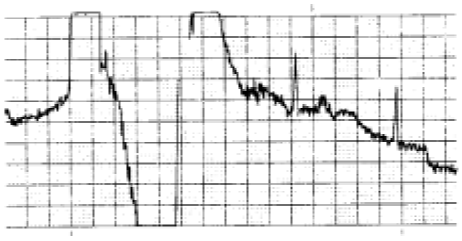


Figure5 (b): Electrode Contact Noise

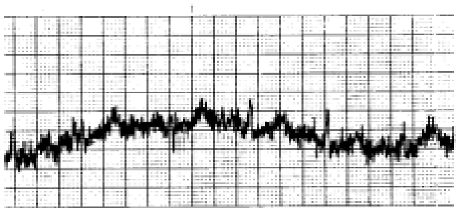


Figure5(c): Muscle Contractions

Figure 6. Three examples of noises on the ECG signal [13].

3.2.1 Detection of the QRS

In this section, the extraction of location of the R waves, as proposed in [3] is discussed. A double thresholding technique is implemented for the extraction of R waves. These different two threshold used for pre-processing are further applied to bandpass filter the ECG signal and the integrations waveform which are performed as shown in Figure 4. The QRS complex peaks should exceed the thresholds in both signals. And in case the peaks are not detected by the thresholds, then it is the noise peak in the given signal.

For maximizing the chance of detection of QRS, the algorithm involves applying two thresholds to both the signals. It includes a low and a high threshold and automatically adjust the level of threshold based on amplitudes of QRS and noise peaks. The higher threshold is used for the analysis of the system and lower threshold for no QRS complexes is detected in certain time interval. And then it goes back in time using a search-back algorithm and search for peaks within that time interval using the lower threshold.

For the case of irregular heart rate detection, the slow adjustment of the threshold is performed when it operates on normal heart rates, but if the heart rates becomes irregular, the algorithm should be able to control that as well. And this can be performed by reducing the thresholds by half and this is done to increase the chance of detection sensitivity and avoid missing beats [3].

3.2.2 Compressed Sensing

CS plays a vital role in the signal processing due to its ability to reconstruct signals from data sampled at sub-Nyquist rate. It has been already applied in medical imaging, communication, MRI, radar imaging, remote sensing, machine learning and so on.

Usually the Shannon Nyquist sampling theorem is strongly followed in system signal processing. As per the theorem, “the sampling frequency of a signal should be at least twice the bandwidth of the signal to avoid aliasing. Signal bandwidth is defined as the difference between highest and lowest frequencies of a signal”. Mathematically, the Nyquist sampling theorem can is expressed as per (2).

$$f_s > 2 f_{\max} ; \qquad (2)$$

Where f_s is the sampling frequency and fmax is the highest frequency occurring in the signal.

With the Shannon-Nyquist sample theorem, aliasing happens when signals are sampled at sub-Nyquist rate or when the desired condition is violated. Because of aliasing, higher frequencies appears as lower frequencies in the sampled signal. As a result, reconstruction of signal cannot be done from the aliased samples.

In this project, this is one of the critical case since one of the main aims is to achieve a compressed representation of the signal to reduce the power consumption of the wireless transfer of the data. So, this method for compressing the signal is important and in this section the CS is explored as a tool for that task [14].

3.2.3 Sparsity and Compressible Signals

"CS relies on the given signal being represented as sparse in a given basis, and its purpose is to reduce the amount of measurements used to represent the signal. It is possible to find such a basis, if it is assumed that natural signals are sparse or compressible in the sense that they have concise representations when expressed in the proper basis" [3]. After all, the physical signals are typically non-sparse by nature. So, it is possible to construct the signal to transform domain in which a sparse representation can be extracted. In this concern, the introduction to the pre-processing stage is imported where a sparse signal representation is derived using the above mentioned Pan and Tompkins QRS detection algorithm. Specially, this algorithm is designed in such a way that it is suitable for any type of ECG signals without the need of manual algorithm calibration. This is basically carried out because of the robust nature of the detection algorithm, which utilizes a dual-threshold technique, search-back for missing peaks, and a running RR-interval estimation for detecting irregularities. Initially, the sparsified ECG signal is compressed and then it is transmitted over a wireless channel or medium. And at the receiver-end the original signal is recovered from the CS samples by solving a convex optimization problem that detects the sparsest solution out of infinitely many possible. As this is often prove to be the correct solution [15]. Applying a sufficient amount of CS samples to the system, this type of algorithm enables (almost) exact signal recovery.

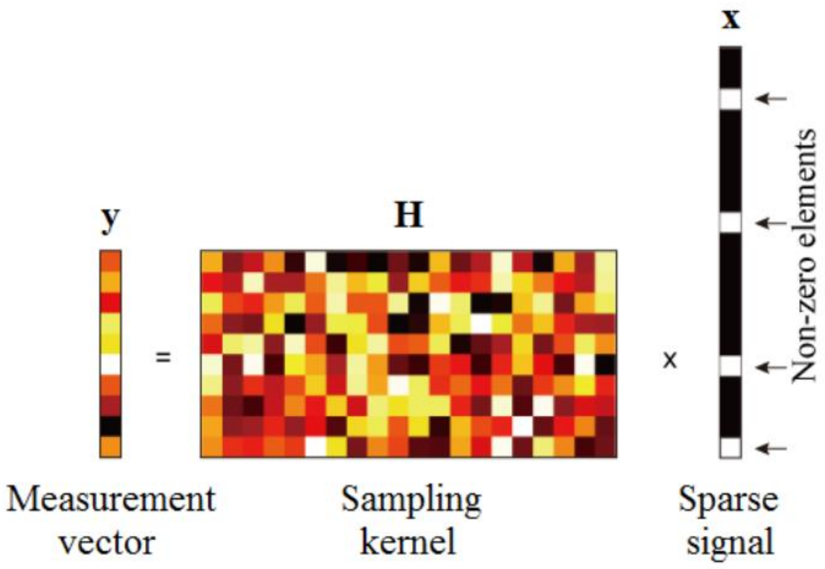


Figure 7.Graphical representation of the CS [3].

CS theory depends first and foremost on the signal of ψ . The theory behind CS is outside the scope of this thesis; the interested reader can refer to [14, 29, 30 from the report] for more information. To summarise, and as illustrated in Figure 7;

The dense signal f had a sparse representation x.

$$f = \psi x = f = \psi x = \sum_{i=1}^{n} x_{i,}, \quad \|x\|_0 < n \tag{3}$$

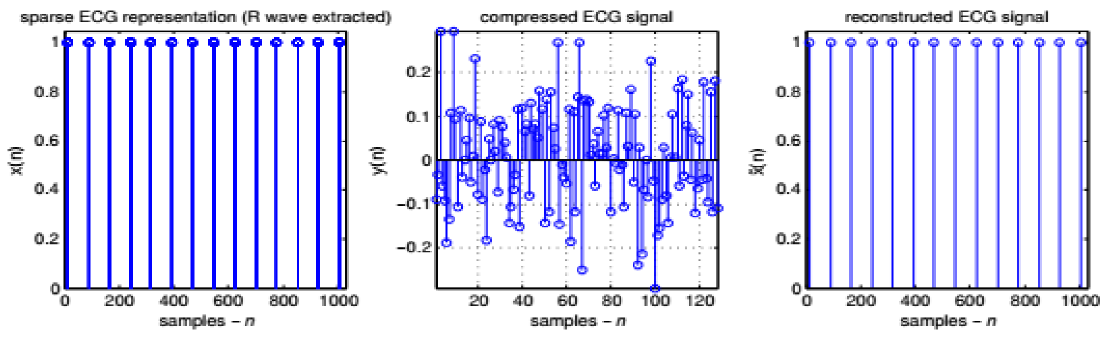
And to compress the signal a sensing matrix is applied.

$$y = \phi^T f = \phi^T \psi x = Hx, y \in R^{m \times 1} \tag{4}$$

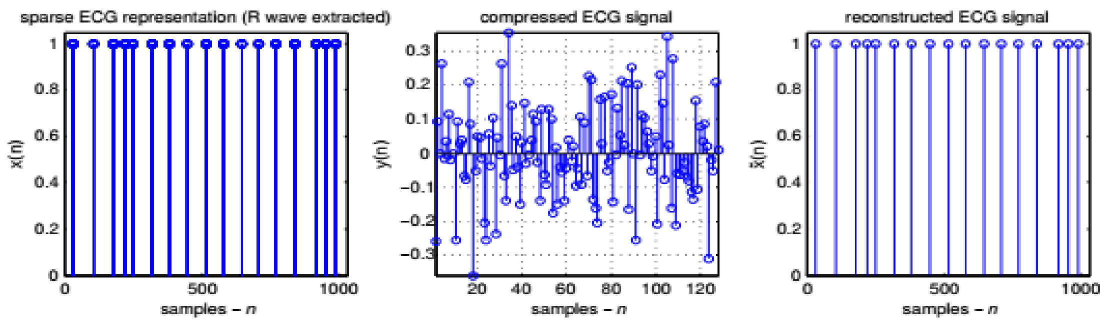
where, $H = \phi^T \psi$

3.3 Heart-Rate Monitoring Examples for Reconstruction of Sparsified ECG Test Signals

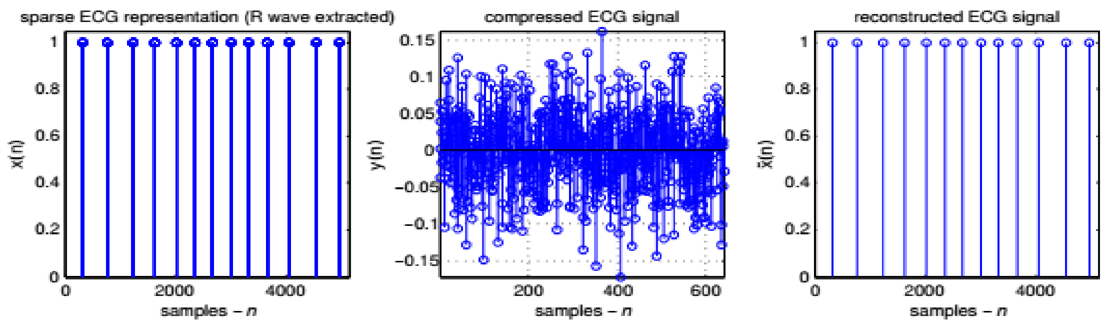
Figure 7[3] shows the uncompressed, compressed and reconstructed forms of the sparsified ECG signal for three instances of test signals by performing the CS simulation. These three test signals are very similar in their pulse streams and the only changed parameters is the intervals between the consecutive R peaks. And such variation is mainly due to the arrhythmic behaviour of the ECG signal.



(a) signal 1, reconstruction results in double precision float values, where MSE $= 3.1028 \cdot 10^{-19}$.



(b) signal 4, reconstruction results in double precision float values, where MSE $= 1.3395 \cdot 10^{-18}$.



(c) signal 6, reconstruction results in double precision float values, where MSE $= 7.6325 \cdot 10^{-22}$.

Figure 8. Uncompressed, compressed and reconstructed version of the sparsified ECG test signals [3].

The peaks of the reconstructed signals are observed clearly in Figure 8.Similarly, the amplitude of the reconstructed signals are also preserved. The estimation results given based on visual inspection with calculated error metrics i.e. mean squared error (MSE) value between the original R wave and the reconstructed extracted ECG for above Signal 1, Signal 4 and Signal 6 from Figure 7 are $3.1028 \cdot 10^{-19}$, $1.3395 \cdot 10^{-18}$ *and* $7.6325 \cdot 10^{-22}$, respectively [3]. Therefore, its concluded that the reconstruction of the ECG have been performed and achieved in all three cases.

3.4 Analysis of the Existing Architectural Design

The kernel operation in CS is a matrix-vector multiplication which is shown in Equation (5) and Equation (6).

$$y = Hx \tag{5}$$

$$\Downarrow$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} H_{11}x_1 & + & H_{12}x_2 & + & \cdots & + & H_{1N}x_N \\ H_{21}x_1 & + & H_{22}x_2 & + & \cdots & + & H_{2N}x_N \\ & & & \vdots & & & \\ H_{M1}x_1 & + & H_{M2}x_2 & + & \cdots & + & H_{MN}x_N \end{bmatrix} \tag{6}$$

From (5), it can be seen that all the operations can be executed in sequential order. Initially one possible sequence is calculated as y_1 and then y_2 and so on; this means all the operation in the successive matrix rows are executed before any other operations. So, one logical procedure is to reorganize the individual operations in time onto the available hardware units in order to reduce the total execution time.

In [3], the Altera Quartus II design suite has been used [17]. Here, the adders and multipliers are synthesized on the hardware platform FPGA and the information and results collected from the synthesis are presented in Table 1. In this MSc thesis, such synthesis is also conducted (on a different FPGA model) and reported in Section 5.

Table 1: Synthesis results for different types of functional units [3].

Component Type	Number of LEs	$f_{max\ [MHz]}$	$T_{min\ [ns]}$
Adder	16	312.79	3.197
Multiplier (combinatorial)	188	85.37	11.714
Multiplier (embedded)	9	225.33	4.438

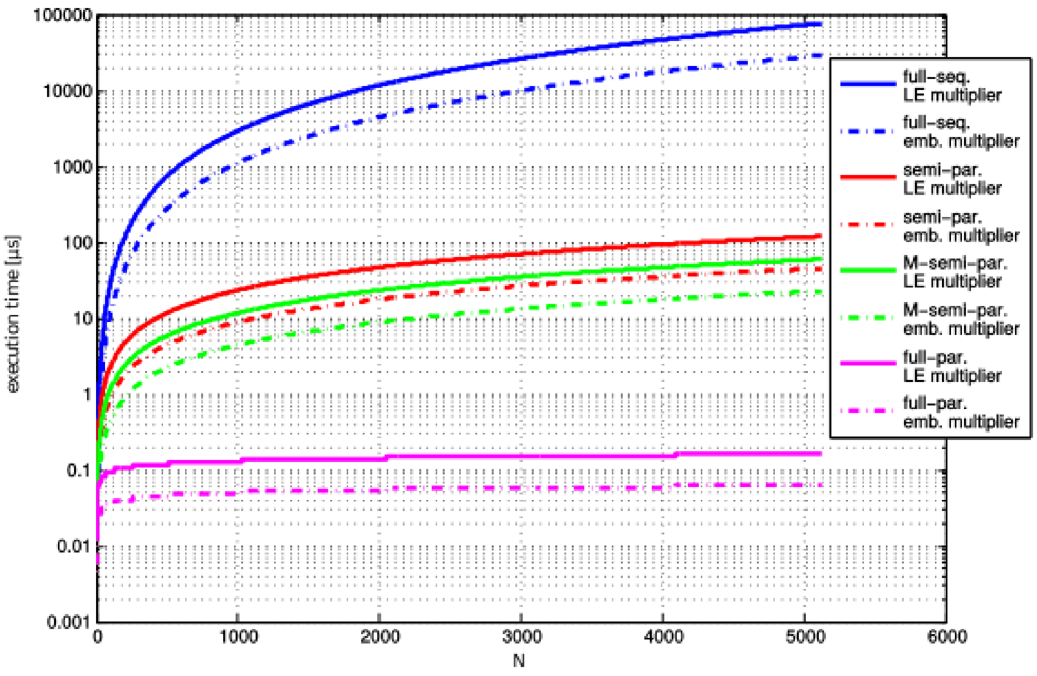


Figure 9. Total execution time for the different schedules as a function of N [3].

Figure 9 [3] describes the total execution time that is estimated as a function N for the multiplier (combinatorial) and embedded multiplier. The different degrees of parallelism are plotted with their respective execution time. From Figure 9, it is observed that the semi-parallel schedule has linear growth in the number of clock cycles as a function of N and for the case of full-parallel scheme, the clock cycles grows logarithmically with N, whereas the full sequential schedule has a non-linear growth with N. The speed-up element is mainly dependent on the type of schedule chosen and the amount of hardware resources.

The full parallel approach is more beneficial with respect to execution time and large number of N values. And for the case of reduction of the usage of hardware resource, semi parallel and the full-sequential can be used. So, this can be a possible trade-off for the designer to consider.

3.5 RTL Implementation Results

In [3], the RTL implementation of the modified semi and full parallel solution are presented. They are explained according to a finite state machine with data path (FSMD) which supports the matrix $H \in R^{2x4}$ and a vector $x \in R^4$.

In the case of the modified semi-parallel architecture (which is illustrated in Figure 10), the multiplication are performed in parallel and a single addition is executed in every control step. The FSM helps to delete the temporary results that are stored in intermediate registers and updates the output. It also reset every time with the internal counter when the latter approaches to N. Moreover, the “clock enables” signal has to be controlled in every state in order to minimize the energy consumption. The input registers and matrix are in registers X_i and $H_{ji,}$ respectively and its temporary results are stored in R_i and the main results in register Y_i. Increasing the numbers of rows and columns in the matrix simultaneously will increase the numbers of FUs in both the cases matching numbers of rows and columns in the respective matrix [12].

And in the case of full-parallel architecture (which is shown in Figure 11), multipliers and adders are also allocated for each operation in the matrix-vector multiplication. The FSM remains the same as for the case of modified semi-parallel architecture. These two architectures are coded in VHDL and further synthesized on the Cyclone III P3C25F324C8 FPGA.

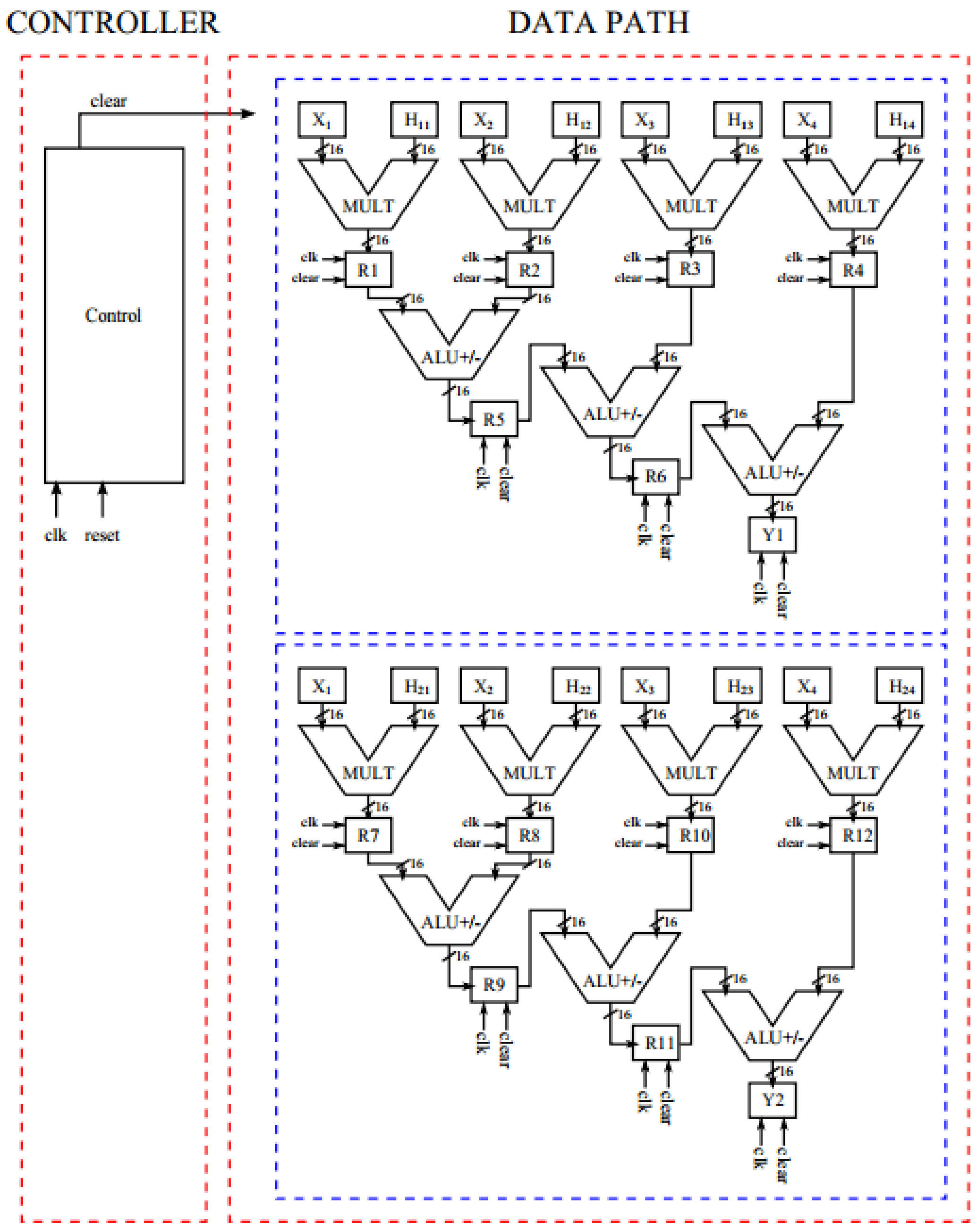


Figure 10.The RTL implementation of the modified semi-parallel architecture [3].

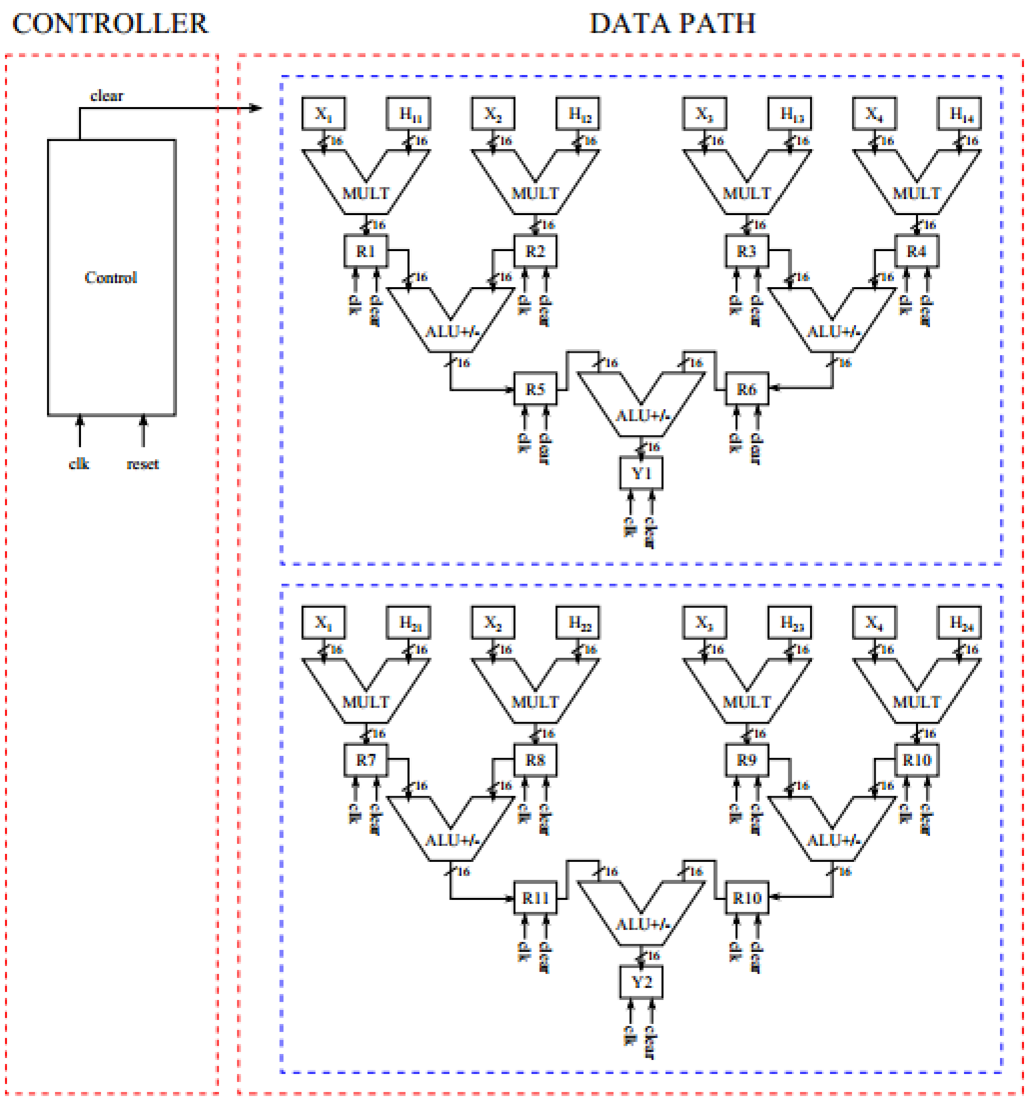


Figure 11. The RTL implementation for the full-parallel architecture [3].

This concludes the analysis of the existing system. The next chapter present the method used to estimate the power consumption of the above architectures on an FPGA platform.

4 Method for Estimating the power Consumption on FPGA

The previous chapter has introduced the essential elements of the existing WSN-based HRM system proposed in [3]. In particular, the signal processing chain (QRS detection and CS) has been presented; furthermore, the corresponding architectures and their FPGA implementation results have been highlighted.

As mentioned in the problem statement (see Section 1), the next natural step (and the main goal of this thesis) is to evaluate the power consumption of these architectures for an FPGA based implementation. This chapter presents the method used to estimate the power consumption of the architectures on the Cyclone IV FPGA. Then, Section 5 presents the corresponding results.

Please note the FPGA used in this thesis is not exactly the same as used in [3]; this is due to the end of support for the Cyclone III by Altera. Also note that this chapter is based on some work that I have carried out during a traineeship at T.J. Seebeck Department of Electronics. Thus, the main body of this chapter is based on material developed during that traineeship with the purpose of supporting the experimental phase of this thesis.

Also note that following the acquisition of Altera by Intel, the tools and supported devices have again been revised; the method described here specifically applies to the versions mentioned later on.

This chapter demonstrates how to prepare a design so that its power consumption can be estimated using Altera Tools. It describes the specified design in VHDL which is to be studied for the power consumption and its estimation using Altera Quartus and ModelSim 10.4b for the Cyclone IVE FPGA.

The method presented is based on following version of software:

- Quartus Prime 15.1 Lite_Edition
- ModelSim 10.4b

There are several possible ways for estimating the power of designs mapped onto FPGA; with all information delivered, estimation becomes easier, understandable and accurate too. The method presented in this chapter is one of the accurate ways of performing power estimation without executing the analog (i.e. hardware-based) simulation. The major steps are explained in the flow chart shown in Figure 12. I have composed it by compiling information from Altera documents [20][21][22].

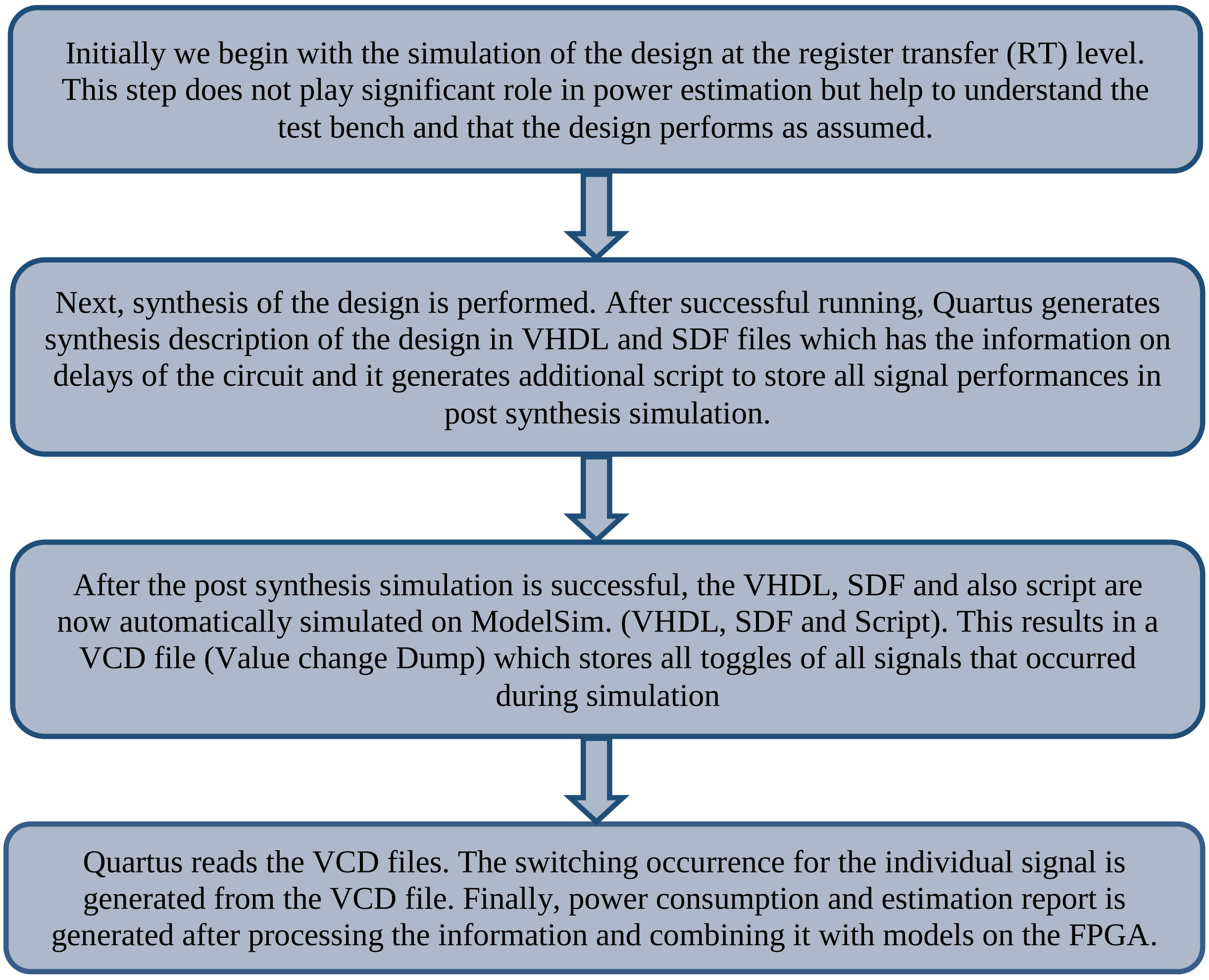

Initially we begin with the simulation of the design at the register transfer (RT) level. This step does not play significant role in power estimation but help to understand the test bench and that the design performs as assumed.
Next, synthesis of the design is performed. After successful running, Quartus generates synthesis description of the design in VHDL and SDF files which has the information on delays of the circuit and it generates additional script to store all signal performances in post synthesis simulation.
After the post synthesis simulation is successful, the VHDL, SDF and also script are now automatically simulated on ModelSim. (VHDL, SDF and Script). This results in a VCD file (Value change Dump) which stores all toggles of all signals that occurred during simulation
Quartus reads the VCD files. The switching occurrence for the individual signal is generated from the VCD file. Finally, power consumption and estimation report is generated after processing the information and combining it with models on the FPGA.

Figure 12. Method for FPGA power estimation with Altera tools.

The following background information has been composed by studying Altera documents [22][23].

4.1 Power Overview

External energy is obtained from external power supplies which is required for operation both internally and externally for the FPGA. Designers should have knowledge of the power needed while implementing power supplies solutions which are called “rail power”. In addition, the designers should acknowledge the amount of power actually to be dissipated by the device which means “Thermal power or dissipated power” as compared with the total amount of power dissipated outside the device such as capacitive external loads and different resistors networks in circuit.

The actual power consumed by an FPGA device, external networks and output consists of three main components:

- Dynamic
- Standby
- I/O

The device in standby mode generates standby power from I_{CCINT} current in the device. Internal switching in the device generates dynamic power through charging and discharging of internal nodes on capacitance which is connected to the device pins, input/output drivers and external network termination. Thermal power is constituent of total power that is dissipated actually occurring inside the device package and others dissipated externally. The real thermal power dissipated inside the device is a major concern for designers when determining if the device intrinsic heat transfer ability is sufficient to sustain internal die-junction temperatures within normal operating specification. For thermal dissipation, it requires a solution such as aluminium heat sinks which are needed for better heat transfer performance. In common, standby power, dynamic power and a section of I/O power will consist of actual thermal power constituent of total power.

4.2 Standby Power

Because of leakage currents the device consumes power in standby mode. The amount of power consumption varies by size, temperature and process variations.

In what follows, optimization of power and performance of a Stratix II device built on a 90 nm technology is used as an example. Comparing with other process technology devices, the 90 nm device dissipates more power because of leakage which is a major constituent of the overall power. Junction temperature is a major focus of designers to keep it to minimum temperature for lowering the standby constituent of overall power. Figure 13 illustrates the relation between junction temperature and standby power.

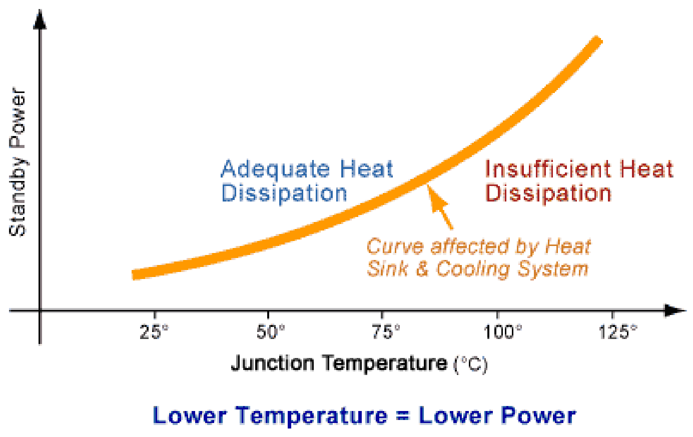


Figure 13. Relationship between junction temperature and standby power [6].

4.3 Dynamic Power

Dynamic power is consumed by internal nodes changing logic levels. The power is required to charge and discharge internal capacitances in the logic array and interconnect networks. Core dynamic power contains routing power and logic element power (LE) for the Stratix II example. Logic element power is absorbed from charging and discharging of internal node capacitance and internal resistive elements. Routing power is generated from charge and discharge of external routing capacitance operated by each logic element. Dynamic power includes architectural resources such as:

- RAM blocks (M512, M4K, and M-RAM)
- DSP-multiplier blocks
- Phase locked loops (PLLs)
- Clock tree networks
- High-speed differential interface (HSDI) transceivers

4.4 I/O Power

V_{CCIO} power is called I/O power, it is absorbed due to charging and discharging of load capacitor. It is connected to the output pins. The output driver circuit is operated mainly in resistive mode and external termination networks (if available).
Devices I/O power is calculated as per Equation (7):

I/O power = (number of active output drivers × power dissipation coefficient) +0.5 × (Sum of die-pad, package trace, pin, and output load cap) ×I/O standard voltage-swing× f_{MAX} × (toggle-factor/100) × V_{CCIO} (7)

4.5 PowerPlay Early Power Estimator Overview

The PowerPlay Early Power Estimator (EPE) [5] supports the Arria II, Arria V, Cyclone III, Cyclone IV, Cyclone V and MAX 10 device families. The user guideline provided by Altera to estimate the power consumption on FPGA design, supports FPGA design at all stage and it also provides detailed features about thermal analysis and the factors that contribute to FPGA power consumption. We can calculate the FPGA power with the Microsoft Excel-based PowerPlay EPE spreadsheet too. For a more accurate power estimation, we should use the PowerPlay Power Analyzer in the Quartus II software.

4.6 Estimating Power Consumption

We can use the PowerPlay EPE spreadsheet to estimate the consumption of power of the design at any stages of design cycle. Even if the design is not started or is incomplete, estimation of power consumption can be performed. The PowerPlay EPE spreadsheet helps to estimates the full design but Altera recommends PowerPlay Power Analyzer in the Quartus II software for accurate information of the design.

4.6.1 Pros and Cons of Power Estimation before Design of FPGA

We can calculate the power estimation before the full FPGA design or even at complete stage which is the positive side.

The disadvantage is precision of the design depends on the inputs and the estimation of the device resources and this may be changed before or after the design is completed. In this case the accuracy of power estimation differs and are less precise. The PowerPlay EPE spreadsheet uses the mean or average but not the real design details implementations. For accessing of the full design details, Altera recommends the PowerPlay power analyser.

4.6.2 Estimating Power Consumption While Creating the FPGA Design

When the FPGA design is completed to a limited extend, we can import the PowerPlay EPE file (early_pwr.csv) which is generated on Quartus software to EPE spreadsheet. We can edit the PowerPlay EPE spreadsheet to demonstrate the resource of device for the final estimation of design.

4.6.3 Pros and Cons of Power Estimation if the Design of FPGA is partially complete

We can perform power estimation in the early stage of FPGA design cycle. It gives the adaptability to consequently fill in the PowerPlay Early Power Estimator spreadsheet taking into account the Quartus II programming aggregation results.
The disadvantage in this case is similar with the previous case.

4.6.4 Estimating Power Consumption after Completing the FPGA Design

When the design of FPGA is fully completed, Altera highly suggest using PowerPlay Power Analyser in the Quartus II software. EPE provides highly accurate result for the device power consumption. In this case EPE uses simulation, user mode or default toggle rate and routing information for the determination of actual power consumption of the complete FPGA design of device.

4.7 Thermal Power

Thermal power is the power scattered in the device. Total thermal power is defined as the total of the thermal power of all the support used in device which includes the maximum power from dynamic power and standby mode. It includes the thermal resources for the I/O parts but it does not include the external power dissipation. Static power (PSTATIC) is the thermal power dissipation on the chip which is independent of user clocks. PSTATIC incorporates the radiation power from the FPGA functional blocks, excluding for I/O DC bias power and transceiver DC bias power and which are considered for the I/O and transceiver section. PSTATIC is the thermal power integral which differs with junction temperature; determine device and characteristics of power processing.

5 Result and analysis

5.1 High level synthesis results of different architecture (with the various levels of parallelism) and their comparisons

The high level synthesis and analysis is performed using Altera Quartus Prime 15.1 Lite Edition for different parallelisms of architectures which are as follows.

- Full parallel
- Full parallel without pipeline
- Generic semi parallel
- Semi parallel with clock enable.

In this thesis, the different architectures (mentioned above) RTL implementations are examined through tests. The VHDL code is synthesized in Altera Quartus IV and tested through simulations. Based on these simulation performed in the program ModelSim-Altera Edition 10.4b, the code is validated with respect of its specifications and requirements. The results are summarised in Table 2.

The different architectural designs were compiled and further, estimation of the power consumption for design full para_4_2, full parallel without pipeline and generic semi parallel was performed. The corresponding power estimation results are shown in Table 3 and Table 4 with two different optimization mode, i.e. Balanced mode and Power (Aggressive) mode, respectively. The RTL implementation of full parallel architecture is shown if Figure 16.

Regarding resource usage, one of the main aspects is the number of embedded multiplier on the FPGA and the reason is the operations within the algorithm which are computed in parallel, so the number of multipliers might be the limiting factor. Furthermore, the logic elements (LEs) are used to implement logics along the multipliers. They can also be used to implement additional multipliers when all embedded multipliers are already used (not the case here).

Table 2: FPGA synthesis results and comparison of different degrees of parallelism.

Architecture	**Full parallel**	**Full Parallel without Pipeline**	**Semi Parallel_clk**	**Generic Semi Parallel**
Top-level Entity Name	para_4_2	Para_4_2_wo_pipe	Seq_8_clken	Mult_Add_example_1
Family	Cyclone IV E	Cyclone IV E	Cyclone IV E	Cyclone IV E
Total Logic elements	100	96	103	384
Total combinational functions	98	96	98	384
Dedicated logic registers	100	32	101	384
Total registers	100	32	101	384
Total I/O pins	130	129	180	92
Embedded multipliers 9-bit elements	8	8	8	25
F_{max} clk frequency (MHz)	299.04	116.9	281.85	282.81

From the synthesis results of the different architectures shown in Table 2, it is observed that the numbers of total logic elements are not much difference (ca 100), except for the generic semi parallel architecture which is 384. This is explained by the fact that the generic semi parallel architecture requires many more blocks than the three other types, as can be observed in Figure 14 (see Figure 14 for some of the details).

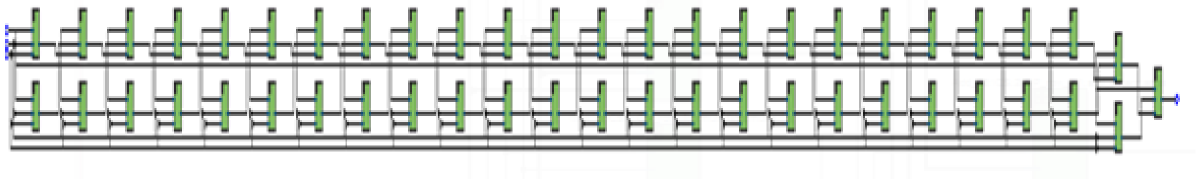


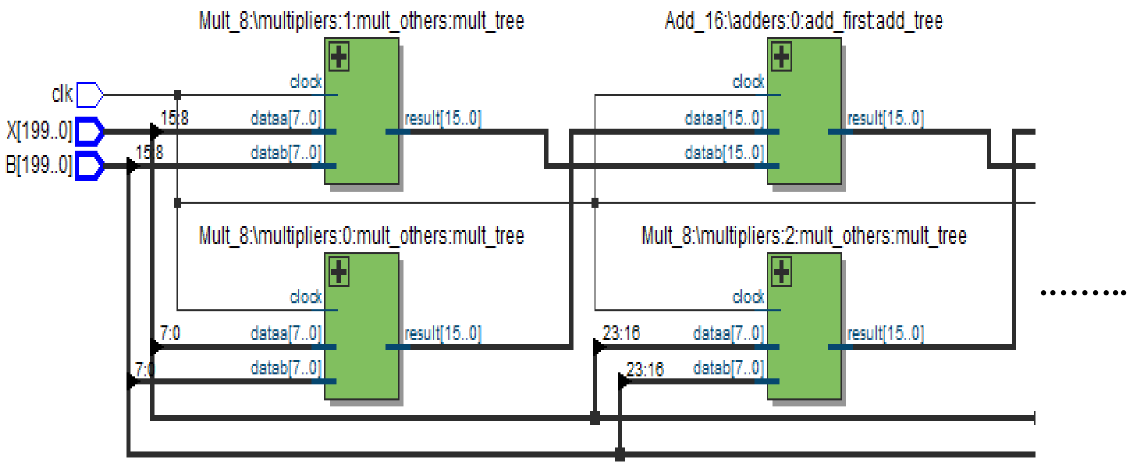
Figure 14. Overview of the RTL generic semi parallel architecture

The total logic elements, total combinational function and embedded multipliers are almost identical for the full parallel, full parallel without pipeline and semi parallel with clock enable architectures because the design has small amount of variation around the main basic structure. This could be explained by the fact that the synthesis tool has not promoted hardware reuse in the more sequential architectures, resulting in the same amount of required resources as for the more parallel ones.

And for the case of generic semi parallel, the number of the embedded multiplier is quite higher- 25, as compared to other architectures with identical low-8.This is also because of the internal organization design which is quite different than the others. The RTL design of generic semi parallel is shown in Figure 14 and Figure 15. It gives a clear understanding how this architecture differs from other. Even though it is semi parallel, the design remains quite sequential because the parallelism is limited with the first two blocks and the other blocks are designed sequentially.

Finally, the effect of enabling or not a pipeline can also be noted by comparing the results for para_4_2 and para_4_2_wo_pipe. For the latter, the maximum frequency is 116.9 MHz, whereas for the former it can be increased to 299.04 thanks to the pipeline (i.e. decomposing long combinatorial blocks is smaller ones). The price to pay is the increased number of registers (increase from 32 to 100) which stores data between the pipeline stages.

Input side of generic semi parallel architecture



Output end of generic semi parallel architecture

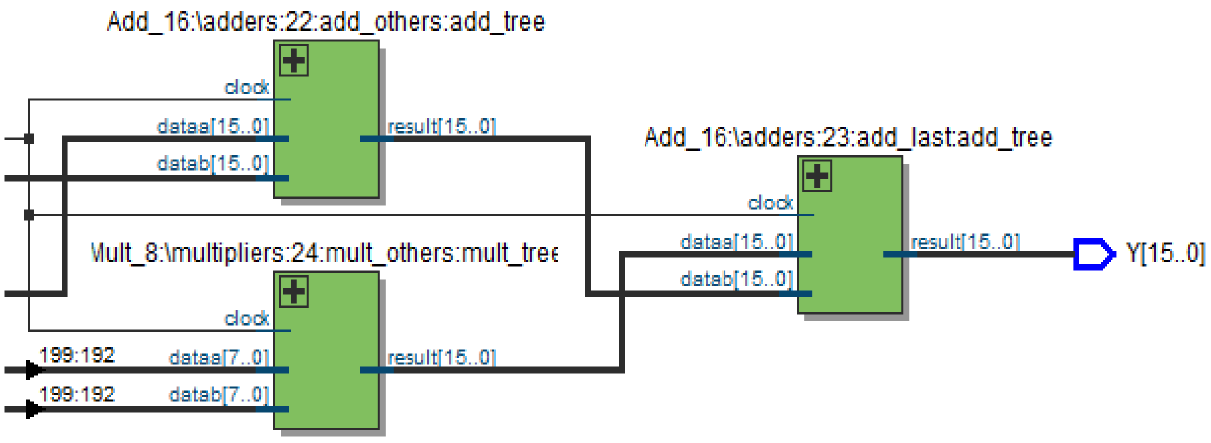


Figure 15. RTL implementation of generic semi parallel architecture.

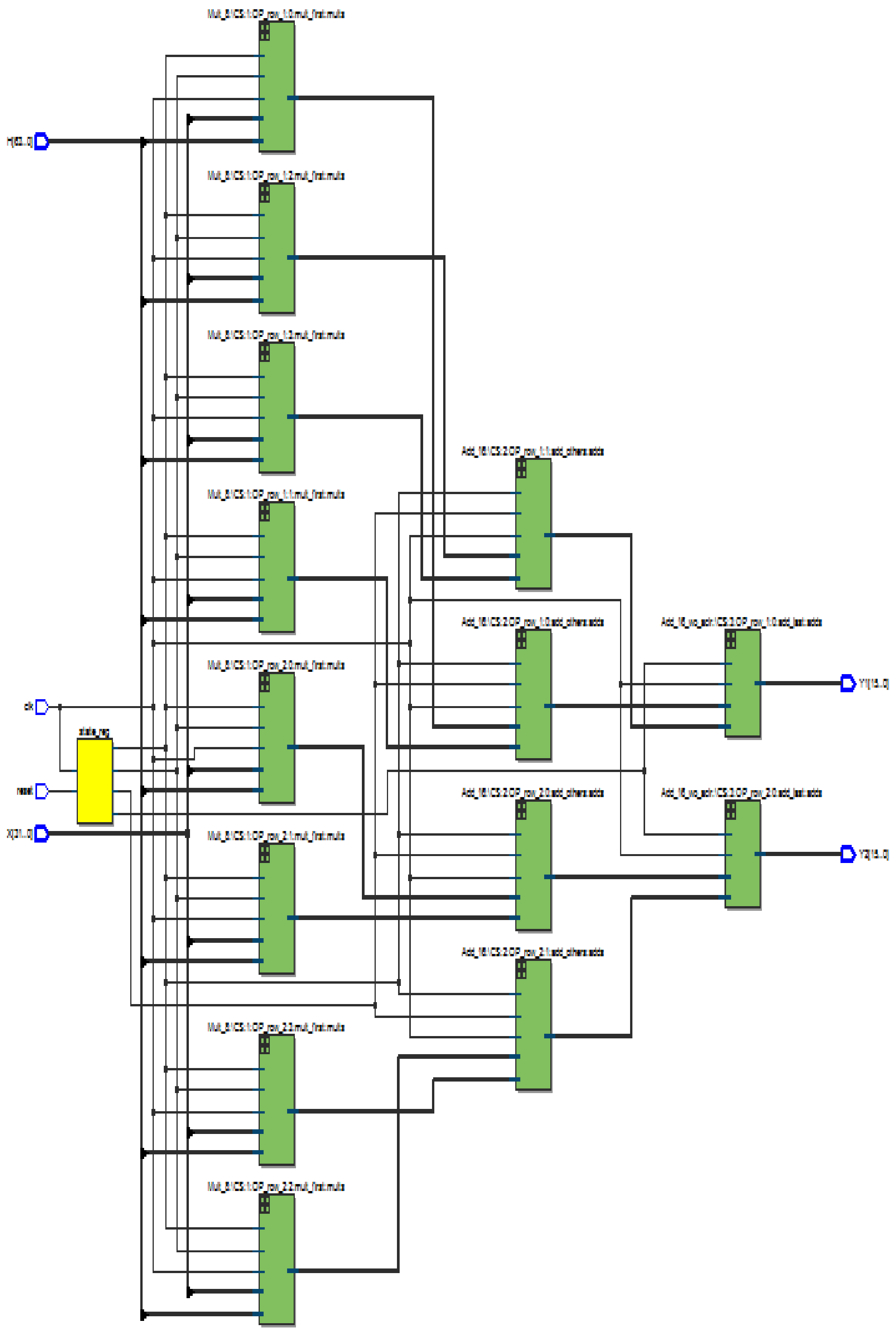

Mult_8\CS:1:OP_row_1:0.mult_first.mults
Mult_8\CS:1:OP_row_1:2.mult_first.mults
Mult_8\CS:1:OP_row_1:3.mult_first.mults
Mult_8\CS:1:OP_row_1:1.mult_first.mults
Mult_8\CS:1:OP_row_2:0.mult_first.mults
Mult_8\CS:1:OP_row_2:1.mult_first.mults
Mult_8\CS:1:OP_row_2:3.mult_first.mults
Mult_8\CS:1:OP_row_2:2.mult_first.mults
Add_16\CS:2:OP_row_1:1.add_others.adds
Add_16\CS:2:OP_row_1:0.add_others.adds
Add_16\CS:2:OP_row_2:0.add_others.adds
Add_16\CS:2:OP_row_2:1.add_others.adds
Add_16_wo_adr\CS:3:OP_row_1:0.add_last.adds
Add_16_wo_adr\CS:3:OP_row_2:0.add_last.adds
H[63..0]
clk
state_reg
reset
X[31..0]
Y1[15..0]
Y2[15..0]

Figure 16. RTL implementation of full parallel architecture.

Table 3: Power estimates for different architectures in power balanced mode at 100MHz.

Component	**Architectures**		
	full_para_N4_M2	**Para_4_2_wo_pipe**	**Generic_semi_para Mult_Add_example_1**
Power estimates clock	100 MHz	100 MHz	100 MHz
Optimization mode	Balanced	Balanced	Balanced
Total Thermal Power Dissipation	85.97 mW	83.31 mW	69.98 mW
Core Dynamic Thermal Power Dissipation	3.59 mW	2.66 mW	11.53 mW
Core Static Thermal Power Dissipation	42.588 mW	42.87 mW	42.91 mW
I/O Thermal Power Dissipation	39.50 mW	37.77 mW	15.53 mW
Power Estimation Confidence	High: user provided sufficient toggle rate data	High: user provided sufficient toggle rate data	Medium: user provided moderately complete toggle rate data

From Table 3, it is observed that lowest total thermal power dissipation-69.98 mW using the balanced optimization mode is achieved for the generic semi parallel architecture. Although it requires more resources, the architecture design and the way its operations are executed, results in a lower power consumption. As can be seen in Table 2, the number of I/Os-92 is lower than that of the other architectures and are not read all simultaneously but rather sequentially. This is reflected in Table 3, where the I/O power- 15.33 mW is quite lower than that of the other architectures (39.50 mW and 37.77 mW).

Note, however, that the power estimation confidence is only ‘medium’ for the generic semi parallel architecture (it is ‘high’ for the other ones). Although I have experimented with the various input signals and internal configuration, it was not possible to increase the confidence level. This means that no firm conclusion can be made when comparing the power consumption of this architecture as compared to the other ones. Nevertheless, the results still indicates that this architecture may have the potential to be lower power than the other ones (at the cost of the higher resource usage).

Table 4: Power estimates for different architecture in power aggressive mode at 100 MHz.

Component	**Architectures**		
	full_para_N4_M2	**Para_4_2_wo_pipe**	**Generic_semi_para Mult_Add_example _1**
Power estimates clock	100 MHz	100 MHz	100 MHz
Optimization mode	Power (Aggressive)	Power (Aggressive)	Power (Aggressive)
Total Thermal Power Dissipation	83.50 mW	81.28 mW	69.59 mW
Core Dynamic Thermal Power Dissipation	2.80 mW	2.59 mW	11.16 mW
Core Static Thermal Power Dissipation	42.87 mW	42.88 mW	42.91 mW
I/O Thermal Power Dissipation	37.82 mW	38.82 mW	15.53 mW
Power Estimation Confidence	High: user provided sufficient toggle rate data	High: user provided sufficient toggle rate data	Medium: user provided moderately complete toggle rate data

Finally, the illustration of the power consumed by the three different architectures in power (aggressive) optimization mode can be observed in Table 4. The designer can use Power (aggressive) optimization mode if it is needed to further minimize the power consumption.

For all three architectures, the power consumption decreases as compared to the balanced mode. The total thermal power dissipation for full parallel, full parallel without pipeline and generic semi parallel architecture is decreased by 2.47 mW, 2.03 mW and 0.39 mW respectively. It can also be noted that the aggressive power optimization mode affects the three power components, but in small amounts. As for the results with the balanced optimization, the power estimation confidence is ‘high’ for the full parallel and full parallel without pipeline architecture and ‘medium’ for generic semi parallel architecture.

Based on the above experiments and comparisons, system designers would have to make trade-off between resource usage, speed, and power consumption.

The full parallel without pipeline architecture requires less resources which can be seen from Table 2, but its maximum speed is also twice less than the full parallel. So, for example, depending on the requirement of the application, if designers need a fast architecture they should choose the full parallel. But, if speed is not a major problem then the full parallel without pipeline architecture can be used as it lowers the speed but is less resources can be hungry. At the same time, the power consumption between the two architectures is quite similar, so this cannot be used as a comparison parameter between them.

Based on the above results, it is also clear that the generic semi parallel with clock enable architecture provides the lower power consumption; but at the same time it requires much more resources. So again, the choice depends on the application requirements and designers have to trade-off one performance metric for another one.

6 Conclusion

The problem statement for this MSc thesis was formulated as follows:

"How to estimate the power consumption of FPGA-based HRM nodes and what are those estimates for the full parallel, full parallel without pipeline and generic semi parallel architectures proposed in [3]?"

I have performed a number of tasks to address this problem. Firstly, I have analysed the proposed overall system, illustrating how QRS extraction and CS can be performed to reduce the amount of data to be transmitted. Then I have analysed existing architectures for the CS hardware engine, illustrating how parallelism is related to execution time. I then described the method for estimating power consumption using Altera tools. Next, I applied this method on four architectures for the hardware synthesis and three selected architectures for the power estimation. Finally, both the synthesis results and power estimation results have been discussed.

The main findings are:

- Enabling pipelining (full parallel architecture) results in higher possible maximum frequency at the cost of increased required number of registers; at the same time, pipelining does not affect power consumption very much;
- The generic semi parallel architecture has the lowest power consumption but requires many more resources as compared to the other ones (put some numbers (main difference power consumption));
- For the tested architectures, using the aggressive power optimization mode brings a moderate reduction in terms of power consumption (key number).

Given the above described method and corresponding results, it can be said that the problem statement has been addressed adequately. In what follows, a possible extension of this work is discussed.

In this thesis the focus was on power. A natural continuation of the work would be to investigate energy consumption; this would help estimating e.g. the time a node can operate depending on its battery capacity.

Generally speaking, energy is the product of power and execution time. In the above work, we have estimated the power consumption, so what would be needed is the execution time of the various architectures. Although the synthesis results already give an indication of the maximum frequency, they do not include the execution times for the individual computations. This would require to run detailed timing simulations, which could be a task for future work.

Another possible future work would be to port the architectures onto ultra-low power FPGAs such as Microsemi’s IGLOO nano low-power FPGAs. On the one hand, porting the architectures onto this other FPGA family is deemed relatively uncomplicated (the operations being mostly additions and multiplications). On the other hand, such FPGAs are much more resource limited than the Cyclone IV used in this thesis; thus, it remains to be explored which of the architectures might possibly fit onto the IGLOO FPGAs. This might also call for further modification of the digital signal processing algorithms, such as reducing their numerical precision.

References

[1] Juul Achten.;Asker.Jeukendrup."Heart Rate Monitoring Application and Limitations". A.E. Sports Med,2003.

[2] Mohammad EI-Sayed.; Peter Koch.; Yannick Le Moullec "Architectural Design Space Exploration of an FPGA-based Compressed Sampling Engine: Application to Wireless Heart-Rate Monitoring". Aalborg University, IEEE Conference Publication, 2015.

[3] Mohammad EI-Sayed.; Soren Lund.; "An FPGA-friendly Compressed Sampling Engine for WSN-based Heart Rate Monitoring" Project Work, Department of Electronic System, Aalborg University, 2015.

[4] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm", IEEE, pp. 230-236, 1985.

[5] C. Pavlatos, et al., "Hardware Implementation of Pan & Tompkins QRS Detection Algorithm", 2003.

[6] R. Matos and R. Pereira, "Electrocardiogram", UCIP, pp. 825-831, 2012.

[7] Small Animal Cardiology, "Electrocardiogram", http://research.vet.upenn.edu/smallanimalcardiology/ECGTutorial/tabid/4930/Default.aspx, visit date: 26/02-2015.

[8]YongijinYongjin Wang, Foteini Agrafioti, Dimitrios Hatzinakos, and Konstantinos,"Analysis of Human Electrocardiogram for Biometric Recognition" http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.450.8148&rep=rep1&type=pdf

[9] Practical Clinical Skills, “Arrhythmia Reference Guide”, http://www.practicalclinicalskills.com/ekg-reference-guide

[10] N. V. Thakor, J. G. Webster and W. J. Tompkins, “Optimal QRS detector”, pp.343-350, 1983.

[11] C. Pavlatos, et al., “Hardware Implementation of Pan & Tompkins QRS Detection Algorithm”, 2003.

[12] Marjan Karkooti and Joseph R. Cavallaro, “Semi-Parallel Reconfigurable Architectures for Real-Time LDPC”, Department of Electrical and Computer Engineering-Rice University.

[13] G. M. Friesen, et al., “A Comparison of the Noise Sensitivity of Nine QRS Detection Algorithms”, IEEE, pp. 85-98, Feb. 1990.

[14] E. Cándes and M. B. Wakin, “An introduction to compressive sampling”, IEEE Signal Processing Magazine 25 (2), pp. 21-30, 2008.

[15] F. Chen, A. Chandrakasan, and V. Stojanovic, “A Signal-Agnostic Compressed Sensing Acquisition System for Wireless and Implantable Sensors,” in Custom Integrated Circuits Conference (CICC), 2010 IEEE, Sept 2010, pp. 1–4.

[16]MathWorks, “Eps: Floating-point relative accuracy”, http://se.mathworks.com/help/matlab/ref/eps.html, visit date: 01/04-2015.

[17] Altera, Quartus II v11.1. Documentation and Download, https://w1.altera.com/download/software/quartus-ii-se/11,1,2011,download date:2/4-2015.

[18] Altera, “Cyclone III Device Handbook”, publication date: 01/08-2012.

[19]Altera,2 Cyclone III FPGA Starter Board Reference Manual”, publication date’ 2007.

[20]http://wwwhome.cs.utwente.nl/~molenkam/ods/low_power_exercise/dds-power.pdf.

[21] http://files.chinaaet.com/files/group/2010/09/14/7822031785966.pdf.

[22]https://www.altera.com/support/support-resources/operation-andtesting/power/pow-overview.html

[24]https://www.altera.com/support/support-resources/operation-and-testing/power/pow-overview.html.

[25]https://www.altera.com/content/dam/alterawww/global/en_US/pdfs/literature/ug/ug_epe.pdf.

[26]Nate Brookreson, “Using Heart Rate Monitoring for Personal Training” American college of sports medicine-july-2015.

[27]K.C.Chua, V.Chandran, U.R. Acharya and C. M. Lim, “Cardiac state diagnosis using higher order spectra of heart rate variability”, Journal of Medical Engineering and Technology 32 (2),pp. 145-155,2008.

[28]Mohmoud Shuker Mahmoud, Auday A.H. Mohamad “A study of Efficient power Consumption Wireless Communication Techniques/ Modules for Internet of Things (loT) Applications”- http://file.scirp.org/pdf/AIT_2016042217163795.pdf.

[29] A. Y. Carmi, et al., “Compressed Sensing & Sparse Filtering”, Springer, 2014.

[30] Y. C. Eldar and G. Kutynoik, “Compressed Sensing - Theory and Applications”, Cambridge University Press, 1 edition, 2012.

[31] JIAPU PAN and WILLIS J.TOMPKIN,“A real-Time QRS Detection Algorithm”,JPAN-1985.

[32] Nehemiah T.liu, Jose Salinas, “Peak Detection System and Method for Calculation of Signal-Derived”,2016.

Appendix 1

This appendix presents examples of VHDL source code of the testbench used for simulating the architectures, as described in Chapter 4.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% vhdl file for test bench of full_para_N2_M4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use std.textio.all; --to read from and write to files
entity TB_para_N4_M2 is
--Testbench, thus no ports
end entity TB_para_N4_M2;
architecture MyTB of TB_para_N4_M2 is
component para_4_2 is
        generic (
        M: integer := 2;        -- 2^M   Number of And's
        b_m: integer := 8
        );

        port (
        clk:            in STD_LOGIC;
        reset:  in      std_logic := '0';
        X:              in    STD_LOGIC_VECTOR(b_m*(2**M)-1    downto    0)    :=
        "00000010000000100000001000000010";
        H:              in   STD_LOGIC_VECTOR(b_m*(2**M)*2-1   downto   0)   :=
"0000001000000010000000100000001000000010000000100000001000000100";
        Y1:             out STD_LOGIC_VECTOR((b_m*2)-1 downto 0);
        Y2:             out STD_LOGIC_VECTOR((b_m*2)-1 downto 0)
        );
end component para_4_2;
constant M: integer := 2;       -- 2^M   Number of And's
constant b_m: integer := 8;
signal TB_clk : std_logic := '0';
signal TB_reset : std_logic := '0';
signal     TB_X     :     STD_LOGIC_VECTOR(b_m*(2**M)-1     downto     0)     :=
        "00000010000000100000001000000010";
signal     TB_H:     STD_LOGIC_VECTOR(b_m*(2**M)*2-1     downto     0)     :=
"0000001000000010000000100000001000000010000000100000001000000100";
signal TB_Y1: STD_LOGIC_VECTOR((b_m*2)-1 downto 0);
signal TB_Y2: STD_LOGIC_VECTOR((b_m*2)-1 downto 0);

signal TB_reset_done : std_logic := '0';

-- Change this to control the clock frequency !!!
constant clk_period : time := 10 ns; --100 MHZ, as in .sdc file

begin
uut: para_4_2 PORT MAP (
--M => TB_M,
--b_m => TB_b_m,
clk => TB_clk,
reset => TB_reset,
X => TB_X,
H => TB_H,
Y1 => TB_Y1,
Y2 => TB_Y2
);
-- Reset process
        TB_para_N4_M2_GEN_RESET: process (TB_clk) is
        begin
        -- Change this to control reset duration
        if (TB_reset_done = '0') then
TB_reset <= '1', '0' after 10 ns;
 end if;
TB_reset_done <= '1';                    --not clean style
```

```
end process TB_para_N4_M2_GEN_RESET;

-- Clock process definitions(clock with 50% duty cycle)
   TB_para_N4_M2_CLK: process
   Begin
TB_clk <= '0';
    wait for clk_period/2;
    TB_clk <= '1';
    wait for clk_period/2;
    end process TB_para_N4_M2_CLK;

-- Stimuli process
   TB_para_N4_M2_GEN_STIMULI: process (TB_clk)
   file infile : text is in  "D:\CD_appendix\3. VHDL code\full_para_N4_M2\ECG.txt";   -
-declare input file  --path needed?
   variable inline : line; --line number declaration
   variable dataread : integer;
            begin
                        if (TB_clk = '1' and TB_clk'event and TB_reset_done = '1') then
                        if (not endfile(infile)) then   --checking the "END OF FILE" is not reached.
                        readline(infile, inline);       --reading a line from the file.
                        read(inline, dataread); --reading the data from the line and putting it in an
integer type variable.
                        TB_X <= std_logic_vector(to_unsigned(dataread, 32));

                        end if;
                        end if;
   end process TB_para_N4_M2_GEN_STIMULI;end architecture MyTB;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% vhdl file for testbench of generic semi parallel architecture
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
library IEEE;
```

```
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use std.textio.all; --to read from and write to files

entity TB_generic_semi_para is
--Testbench, thus no ports
end entity TB_generic_semi_para;

architecture MyTB of TB_generic_semi_para is

component mult_Add_example_1 is
        generic (
        N: integer :=25;        -- Antal multipliers
        M: integer :=8   -- Bit multiplier
        );

        port (
        clk:    in STD_LOGIC ;
        X:      in STD_LOGIC_VECTOR(M*N-1 downto 0);
        B:      in STD_LOGIC_VECTOR(M*N-1 downto 0);
        Y:      out STD_LOGIC_VECTOR((M*2)-1 downto 0)
        );
end component mult_Add_example_1;

constant N: integer := 25;
constant M: integer := 8;
constant Z: integer := M*2;    -- Bit adder
signal      mult_con:      STD_LOGIC_VECTOR(Z*N-1      downto      0)      :=
"1111111111111111100110111111111111111111100110111111111111111111111001
1011111111111111111110011011111111111111111110011011111111111111111
1100110111111111111111111100110111111111111111111100110111111111111
1111111001101111111111111111111001101111111111111111111001101111111
1111111111110011011111111111111111110011011111111111111111110011011
11111111111111111001101111111111111111111100110111";
```

```
signal add_con: STD_LOGIC_VECTOR(Z*(N-2)-1 downto 0) :=
"11111111111111111001101111111111111111111110011011111111111111111111001
1011111111111111111111001101111111111111111111110011011111111111111111
110011011111111111111111111001101111111111111111111110011011111111111111
111111100110111111111111111111111001101111111111111111111110011011111111
1111111111111111111111111111110011111111111111111111111111111111111111111
111111111111111111";

signal TB_clk : std_logic := '0';
signal TB_X : STD_LOGIC_VECTOR(M*N-1 downto 0) :=
"11111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111"; --
Vector Input

signal TB_B: STD_LOGIC_VECTOR(M*N-1 downto 0) :=
"11111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111"; --
Matrix Input
signal TB_Y: STD_LOGIC_VECTOR((M*2)-1 downto 0);

-- Change this to control the clock frequency !!!
constant clk_period : time := 10 ns; --100 MHZ, as in .sdc file

begin

        uut: mult_Add_example_1 PORT MAP (
        --M => TB_M,
        --b_m => TB_b_m,
        clk => TB_clk,
        X => TB_X,
        B => TB_B,
        Y => TB_Y
```

```
          );


-- Clock process definitions(clock with 50% duty cycle)
   TB_generic_semi_para_CLK: process
   begin
TB_clk <= '0';
wait for clk_period/2;
TB_clk <= '1';
wait for clk_period/2;
end process TB_generic_semi_para_CLK;
-- Stimuli process
   TB_generic_semi_para_GEN_STIMULI: process (TB_clk)
   file infile : text is in  "D:\CD_appendix\3. VHDL code\full_para_N4_M2\ECG2.txt";
--    declare input file  --path needed?
   variable inline : line; --line number declaration
   variable dataread : integer;
          begin

if (not endfile(infile)) then   --checking the "END OF FILE" is not reached.
readline(infile, inline);       --reading a line from the file.
read(inline, dataread); --reading the data from the line and putting it in an integer type
variable.
TB_X <= std_logic_vector(to_unsigned(dataread, 200));
          end if;
end process TB_generic_semi_para_GEN_STIMULI;
end architecture MyTB;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% end of TB files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
```

Appendix 2

This appendix 2 presents the input parameters needed to be considered for determination of power consumption.

The parameters determine on if the junction temperature is manually entered or auto computed. The list of parameters is given in Table 4 [25].

Table 4. Information about the Input Parameters [25].

Input Parameter	**Description**
Family	Device family is selected.
Device	Device is selected by you. Bigger device consumes for static power and clock dynamic power is high. Nothing more is affected (power components) by the use of device.
Package	Choose the package that is implemented. Larger scale of packages provides more cooling surface contact points to the circuit board. Dynamic power is not affected.
Power Characteristic	Theoretical or typical worst case silicon process is selected. This influence the static power consumption. Maximum power characteristic is recommended by Altera for the use of power estimation. It provides output that line up with worst-case device measurement.

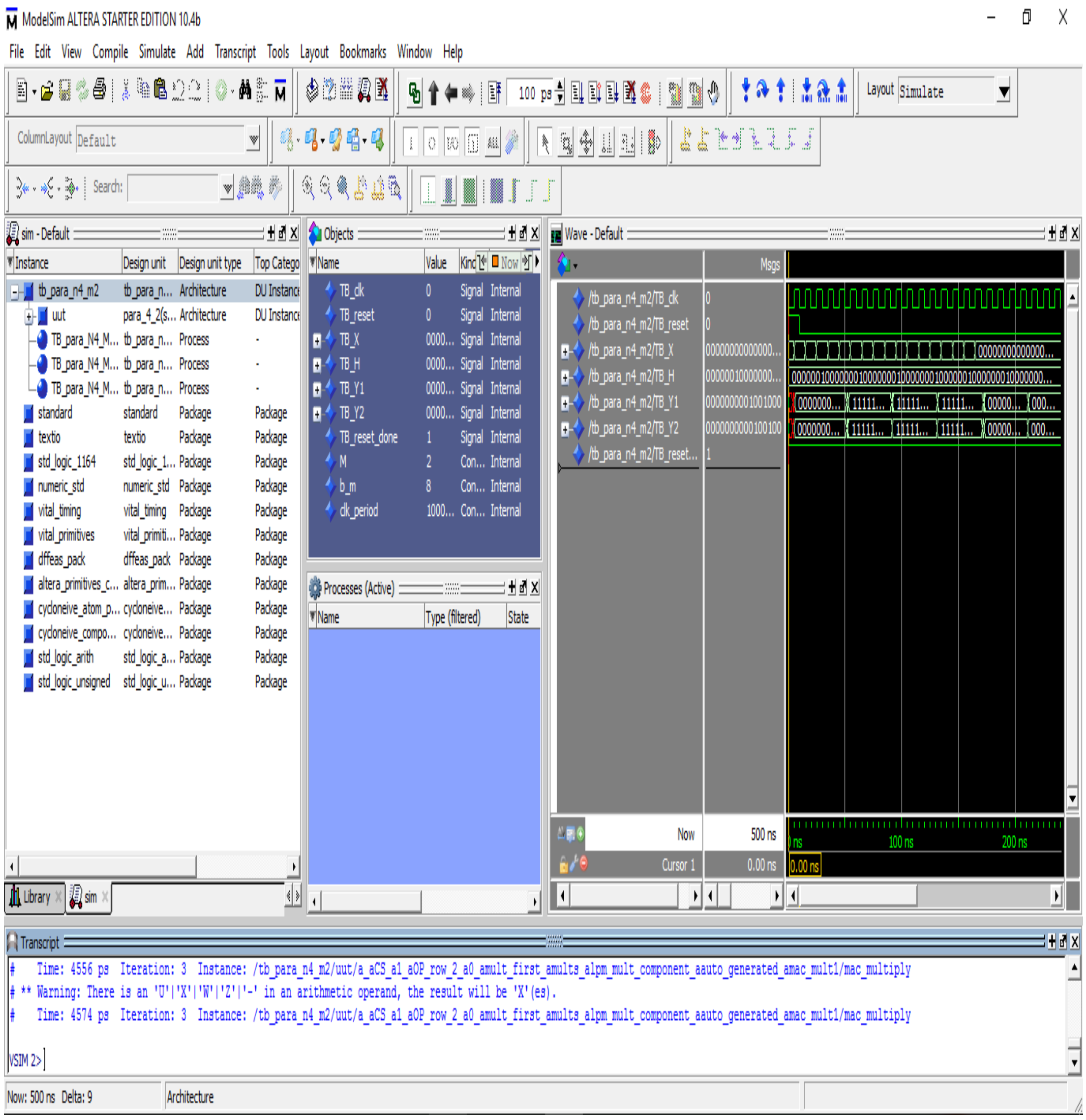
VCCINT Voltage (V)	Select VCCINT voltage for Cyclone IV E devices. Devices with different speed grades C8L, C9L and I8L, VCCINT is set to 1.0V. Devices with speed of grade C6, C7, C8, I7 and A7, VCCINT is set to 1.2V.
VCC_ONE Voltage (V)	VCC-ONE voltage is set for maximum of 10 devices with 3.0V or 3.3V. The internal voltage is adjusted to 1.2V to power supply at the periphery.
Power Model Status	The power model for the devices is suitable only for EPE 14.0 moving.
VCCL Voltage (V)	Devices with the speed grade of -4L the voltage can be 0.9V or 1.1V and for the other speed devices grades, the voltage is set to 1.1V.
Junction Temp, TJ (°C)	The junction temperature is available if we login on the User Entered TJ option. The junction temp is not studied related to the thermal information that is provided. For the highest value of selected temperature grade, Altera recommends setting junction Temp TJ(°C).
Ambient Temp, TA (°C)	When the air temperature near the device entered, this can range from –40°C to 125°C. This field is suitable if we turn on Auto Computed TJ option. If the Estimated Theta JA option is On then it results to compute the junction

	temperature which are always related with power dissipation and thermal resistance (heat sink) and circuit board. In the case of Custom Theta JA option in On then there is computation of junction temperature related with power dissipation and custom θJA entere
Custom θJA (°C/W)	Junction to ambient thermal resistance is entered between device and ambient air (in °C/W). This area is only suitable if the below option are turned On. • Auto Computed TJ • Estimated Theta JA • Heat Sink parameter is custom
Board Thermal Mode	The board is selected which is conventional for thermal analysis which can be typical board, none Conservative or JEDEC (2s2p). It works only when Auto computed TJ is on and estimated Theta JA options. It we select none conservative, the thermal model pretends there is no heat dissipated from the device board which results on calculated junction temperature. This alternative is not accessible if the Heat Sink choice is set to None. If we choose Typical Board then device and package is selected based on the characteristics of a typical customer board stack.

Airflow	The ambient airflow is selected in linear feet per minute (lfm) or meters per second (m/s).The resulted values are 100(lfm) which is 0.5 m/s, 200 lfm (2.0m/s) amd for 400 lfm (2.0m/s).
Heat Sink	Heat sink is selected that is used. We can select one of below mentioned. • No Heat Sink • A custom Solution (Custom) • Heat sink parameters (15 mm– Low Profile, 23 mm– Medium Profile, or 28 mm–High Profile) This subject is accessible if we turn On the Auto computed TJ and Estimated Theta JA options. If neither of this option is selected, custom θSA value is updated by heat sink and we can read the value in Custom θSA (°C/W) parameter.

Appendix 3

This screenshot shows the output waveform for the semi parallel with clock enable architecture.



Instance	Design unit	Design unit type	Top Catego
tb_para_n4_m2	tb_para_n...	Architecture	DU Instanc
uut	para_4_2(s...	Architecture	DU Instanc
TB_para_N4_M...	tb_para_n...	Process	-
TB_para_N4_M...	tb_para_n...	Process	-
TB_para_N4_M...	tb_para_n...	Process	-
standard	standard	Package	Package
textio	textio	Package	Package
std_logic_1164	std_logic_1...	Package	Package
numeric_std	numeric_std	Package	Package
vital_timing	vital_timing	Package	Package
vital_primitives	vital_primiti...	Package	Package
dffeas_pack	dffeas_pack	Package	Package
altera_primitives_c...	altera_prim...	Package	Package
cycloneive_atom_p...	cycloneive...	Package	Package
cycloneive_compo...	cycloneive...	Package	Package
std_logic_arith	std_logic_a...	Package	Package
std_logic_unsigned	std_logic_u...	Package	Package

Name	Value	Kind	Now
TB_clk	0	Signal	Internal
TB_reset	0	Signal	Internal
TB_X	0000...	Signal	Internal
TB_H	0000...	Signal	Internal
TB_Y1	0000...	Signal	Internal
TB_Y2	0000...	Signal	Internal
TB_reset_done	1	Signal	Internal
M	2	Con...	Internal
b_m	8	Con...	Internal
clk_period	1000...	Con...	Internal

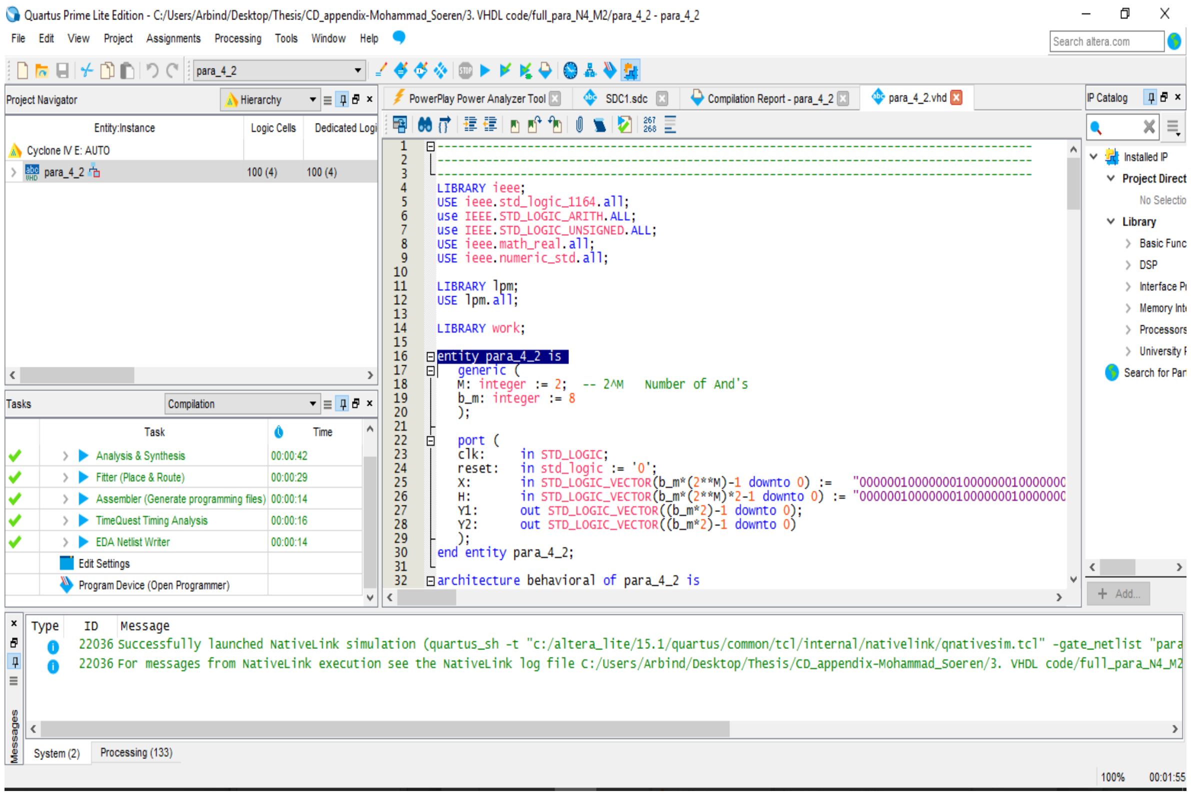
	Msgs
/tb_para_n4_m2/TB_clk	0
/tb_para_n4_m2/TB_reset	0
/tb_para_n4_m2/TB_X	00000000000000...
/tb_para_n4_m2/TB_H	00000010000000...
/tb_para_n4_m2/TB_Y1	0000000001001000
/tb_para_n4_m2/TB_Y2	0000000001001000
/tb_para_n4_m2/TB_reset...	1

```
#    Time: 4556 ps  Iteration: 3  Instance: /tb_para_n4_m2/uut/a_aCS_a1_aOP_row_2_a0_amult_first_amults_alpm_mult_component_aauto_generated_amac_mult1/mac_multiply
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 4574 ps  Iteration: 3  Instance: /tb_para_n4_m2/uut/a_aCS_a1_aOP_row_2_a0_amult_first_amults_alpm_mult_component_aauto_generated_amac_mult1/mac_multiply

VSIM 2>
```

Now: 500 ns Delta: 9   Architecture

Appendix 4

This screenshot show the synthesis of the full parallel architecture.



Quartus Prime Lite Edition - C:/Users/Arbind/Desktop/Thesis/CD_appendix-Mohammad_Soeren/3. VHDL code/full_para_N4_M2/para_4_2 - para_4_2

Entity:Instance	Logic Cells	Dedicated Logi
Cyclone IV E: AUTO		
para_4_2	100 (4)	100 (4)

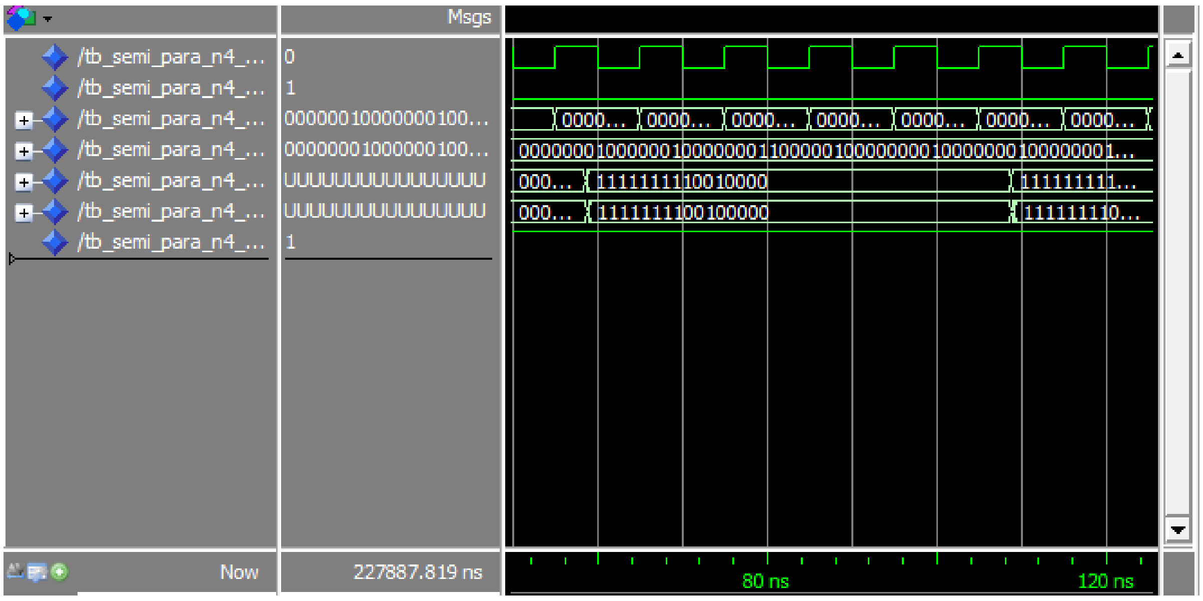
	Task	Time
✓	Analysis & Synthesis	00:00:42
✓	Fitter (Place & Route)	00:00:29
✓	Assembler (Generate programming files)	00:00:14
✓	TimeQuest Timing Analysis	00:00:16
✓	EDA Netlist Writer	00:00:14
	Edit Settings	
	Program Device (Open Programmer)	

```vhdl
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
USE ieee.math_real.all;
USE ieee.numeric_std.all;

LIBRARY lpm;
USE lpm.all;

LIBRARY work;

entity para_4_2 is
   generic (
   M: integer := 2;  -- 2^M   Number of And's
   b_m: integer := 8
   );

   port (
   clk:      in STD_LOGIC;
   reset:    in std_logic := '0';
   X:        in STD_LOGIC_VECTOR(b_m*(2**M)-1 downto 0) :=   "00000010000000100000001000000
   H:        in STD_LOGIC_VECTOR(b_m*(2**M)*2-1 downto 0) := "00000010000000100000001000000
   Y1:       out STD_LOGIC_VECTOR((b_m*2)-1 downto 0);
   Y2:       out STD_LOGIC_VECTOR((b_m*2)-1 downto 0)
   );
end entity para_4_2;

architecture behavioral of para_4_2 is
```

```
Type  ID    Message
      22036 Successfully launched NativeLink simulation (quartus_sh -t "c:/altera_lite/15.1/quartus/common/tcl/internal/nativelink/qnativesim.tcl" -gate_netlist "para
      22036 For messages from NativeLink execution see the NativeLink log file C:/Users/Arbind/Desktop/Thesis/CD_appendix-Mohammad_Soeren/3. VHDL code/full_para_N4_M2
```

Appendix 5

This screen shot shows the output waveform for the semi_para_N4_M2_w_clken architecture.



```
                               Msgs
/tb_semi_para_n4_...    0
/tb_semi_para_n4_...    1
/tb_semi_para_n4_...    000000100000000100...
/tb_semi_para_n4_...    0000000100000000100...
/tb_semi_para_n4_...    UUUUUUUUUUUUUUUU
/tb_semi_para_n4_...    UUUUUUUUUUUUUUUU
/tb_semi_para_n4_...    1

Now                     227887.819 ns
```

%%

%%%%end of simulation screenshots

%%