FPGA IMPLEMENTATION OF EEG CLASSIFIER USING LDA

Nadun Manohara Ellawala

148455A

Degree of Master of Science

Department of Electronic and Telecommunication Engineering

University of Moratuwa

Sri Lanka

January 2019

FPGA IMPLEMENTATION OF EEG CLASSIFIER USING LDA

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science in Electronics and Automaton Engineering

Nadun Manohara Ellawala

148455A

Degree of Master of Science

Department of Electronic and Telecommunication Engineering

University of Moratuwa

Sri Lanka

January 2019

Declaration by candidate

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: …… Date: ……

Declaration by supervisor

The above candidate has carried out research for the Masters thesis under my supervision.

Name of the supervisor: ……

Signature of the supervisor: …… Date: ……

ABSTRACT

FPGA implementation of EEG classifier using LDA

Supervised by: Dr. S. Thayaparan

Key words: *LDA, EEG, FPGA, DWT, HDL*

Design and implementation of feature classification in Electroencephalography (EEG) signal processing system on Field Programmable Gate Array (FPGA) hardware platform is presented in this thesis. Today there is a growing demand for medical devices which process EEG signals, for which, it is important to implement the EEG processing system in hardware instead of software. Processing of EEG signals consist of extracting features from EEG signal and then processing those features to classify the signals. As of today, in most of EEG processing systems, classification part is done on software platform even though the feature extraction is done on hardware. In this project, classification is done with Linear Discriminant Analysis (LDA), based on the features extracted using Discrete Wavelet Transform (DWT), for EEG signals obtained through PhysioNet website. The hardware implementation was done on Field Programmable Gate Array (FPGA) platform using SystemVerilog Hardware Description Language (HDL). Final design has minimum resource utilization, hence is able implement on Basys 3 Artix-7 FPGA Trainer Board with the accuracy of 80%. Therefore, it is concluded this design is suitable for developing low cost, marketable products like sleep detectors for automobile divers. Nevertheless, ultimate goal is to design a simple Application Specific Integrated Circuit (ASIC) chip, which can extract features and classify EEG, so that the full system can be implemented on a portable mobile device without using software platform.

To beloved Parents and Teachers

ACKNOWLEDGEMENT

It is great pleasure to use this opportunity to complete my responsibility by rewarding the gratitude to all the key personals who were involved in the process of making this project a success.

My sincere gratitude always goes to my project supervisor, Dr. S. Thayaparan for the immense support he gave with continuous guidance and kind advice. Further I am thankful to Dr. Anjula De Silva, Dr. Chamira U. S. Edussooriya, Dr. Upeka Premaratne and Dr. Jayathu Samarawikrama for his valuable instructions. Also, this project would not have become successful without the immense support extended by other academic staff members of the Department of Electronic and Telecommunication Engineering University of Moratuwa, who were always willing to share their ideas and experience, which was invaluable.

I also pay my gratitude to the non-academic staff of the Department of Electronic and Telecommunication Engineering University of Moratuwa for the immense support they have shown in utilizing laboratories and laboratory equipment. My special thanks go to Mr. Chinthaka, Technical Officer, Department of Electronic and Telecommunication Engineering University of Moratuwa for allowing me to use the Post Graduate laboratory of the department when required.

Further I appreciate the supportive and enthusiastic batch mates who motivated me to achieve success while working on their research projects.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
ASIC	Application Specific Integrated Circuit
ATM	Automated Teller Machine
CWT	Continuous Wavelet Transform
BCI	Brain Computer Interfaces
Db4	Daubechies 4
DWT	Discrete Wavelet Transform
EDF	European Data Format
EEG	Electroencephalography
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HDVL	Hardware Design and Verification Language
IO	Input Output
ISE	Integrated Synthesis Environment
KFD	Kernel Fisher Discriminant
LDA	Linear Discriminant Analysis
LED	Light Emitting diode
LUT	Look Up Table
MIF	Memory Initialization File
PCA	Principal Component Analysis
RAM	Random Access Memory
RTL	Register Transfer Level
SPI	Serial Peripheral Interface
SVM	Support Vector Machines
TNS	Total Negative Slack
WNS	Worst Negative Slack

Chapter 1

INTRODUCTION

1.1 Problem Identification

With the advancement of digital electronics and signal processing, currently there is a growing demand for medical devices which diagnose health conditions of patients. In this context, processing of Electroencephalography(EEG) signals has also become very active research area and most of the interest is currently on implementing them on hardware, because of the advantages it has over software implementations.

When diagnosing a disease, the processing of EEG signals that should be done to a make decision, can be broadly divided in to two stages. That is the relevant feature extraction and classification of EEG signals based on those features. From these two stages, compared to feature extraction, the classification of EEG signals is still mostly done on software which runs offline, even though there are some researches that have been done. However, it is important to implement classification part also in hardware so that full EEG processing system can be implemented on portable medical devices without requiring any operating system. Apart from that there are many advantages also in implementing EEG classification on hardware like being able to process large amount of information with minimum delay because ability of process data parallelly.

This research will be focused on hardware implementation of EEG signal classification system on the FPGA (Field Programmable Gate Array) development board, using Linear Discriminant Analysis (LDA). Here I will focus on developing optimal algorithm and designing the architecture first and implement all in software, which helps to debugging as well. Then after that it will be migrated to hardware platform, that will be coded using SystemVerilog HDL (Hardware description Language). After the hardware implementation, the design has been tested with actual EEG data records available on PhysioNet (https://www.physionet.org), a website which provides research resources for complex physiological signal. Finally, optimized results will be provided compared to existing systems.

1.2 Motivating for the research

There are two researches that have become the background and the motivation for this design of EEG classification system. First research paper presents a generalized platform for a FPGA design architecture that offers preprocessing steps and set of predefined features where user can configure BCI applications [1]. As a future work, this paper suggests to include a set of classification algorithms to further push the hardware interface of BCIs (Brain–Computer Interfaces).

Second paper presents a reasonable and widespread comparison of some frequently used classification methods under the same conditions where the valuation of different classifiers will be more convictive [2]. This paper concludes that the feature extraction and the classification algorithms should be considered together when designing a BCI system.

From the future work part of first paper and the conclusion part of second paper, it can be concluded that it is important to implement a full system in hardware which has feature extraction and set of classification methods, that is integrated together. Therefore, this project will be part of this integrated system where I focus on classification of signals with the help of LDA.

Also, I have implemented this system for two class scenarios, that is to decide whether a person have an abnormal EEG pattern or not. Hence small percentage changes in accuracy should be acceptable for this implementation, since the final decision from integrated system will based on multiple classification systems.

1.3 Existing solutions and technologies

Even though as of now, the researches have not been done to implement EEG classification systems using hardware there are researches that have been done for the software implementations. In the below papers, LDA classifiers have been implemented for EEG signals as a part of the research.

- Research Paper: EEG features extraction using PCA plus LDA approach based on L1-norm for motor imaginary classification [3]
    - PCA plus LDA is used for classifying EEG signals and is implemented on software
- Research Paper: Classification of human emotions from EEG signals using SVM and LDA Classifiers [4]
    - EEG signal classification is done using both SVM and LDA and is also implemented on software

In both of these research papers, LDA is been implemented on software along with other classification systems. And less attention is been given to the ability to implement it on small marketable devices where hardware implementation become important.

1.4 Novel Contribution

When considering the existing LDA classifiers for EEG signals, in current implementations, most of attention is been given for the hardware implementation of EEG feature extraction and most of EEG classification is done on software platforms.

Also, it can be observed that [2], When designing a BCI system, it is important to implement, multiple classification algorithms, in order to make accurate decisions. And Both feature extraction as well as the classification method should be considered for each algorithm.

Based on this analysis, this research project has been done with novelty contribution for development of the algorithm, designing the system and hardware implementation of LDA for EEG classification.

Chapter 2
HARDWARE IMPLEMENTATION

2.1 EEG feature extraction

2.1.1 Obtaining EEG signals

The functionality of this system required three set of EEG signals. That is, first EEG sample from a healthy person who have normal EEG pattern and second EEG sample from a patient who has abnormal EEG pattern. Finally test EEG signal is required, which will be checked for the abnormality. In real world applications, first two samples should be available offline, and third sample will be available in real time. In order to reduce the complexity of implementation, in this research project, all the three samples have been provided offline to the FPGA development board.

For this implementation, EEG signals from The Sleep-EDF Database [5] have been used for both training and testing the system. This is a collection of 197 whole-night PolySomnoGraphic sleep recordings in EDF format (European Data Format), which also contains EEG signals along with other biomedical signals.

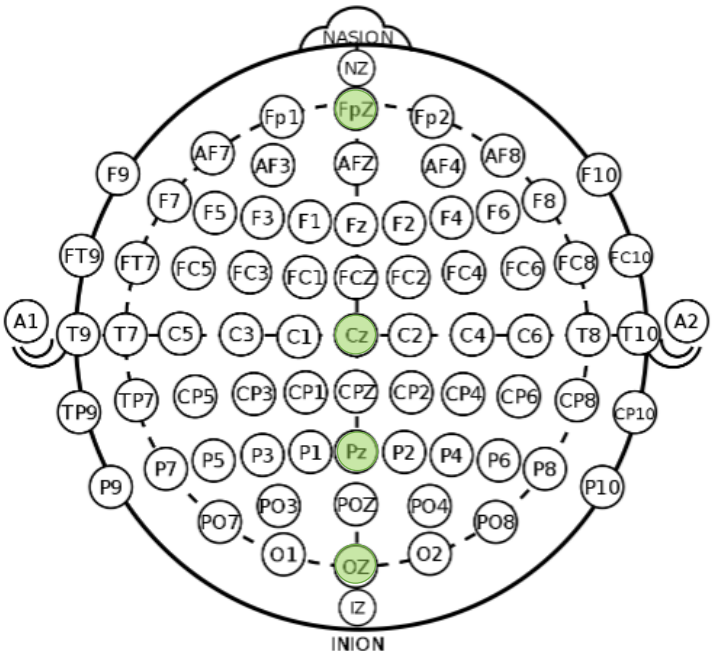


Figure 2-1 EEG Electrode locations

Table 2-1 Abbreviations and Anatomical landmarks for the electrodes

Nasion: the depressed point where the top of the nose meets the ride of the forehead	Inion: Lowest point of the skull at the back of the head, normally felt as a prominent bump
F: Frontal lobe	T: Temporal lobe
C: Central	P: Parietal lobe
O: Occipital lobe	z: zero(midline)
A: Earlobe	Fp: Frontal polar
Even numbers: represent electrodes on the right hemisphere	Odd numbers: represent electrodes on the left hemisphere

These EEG signals are taken from Fpz-Cz and Pz-Oz electrode locations placed on 10-20 international system shown in Figure 2-1. The abbreviations used in this figure is described in Table 2-1. This is an EEG electrode placement system developed from International Federation of Societies for Electroencephalography and Clinical Neurophysiology, in order to ensure standardized reproducibility over time among patients. Here 10 and 20 refer to the actual distance between adjacent electrodes, which is either 10% or 20% of the total front-back or right-left distance of the skull. This electrode placement system has 75 electrodes and 10% division is used. This is a high-resolution version of conventional electrode system which has only 21 electrodes.

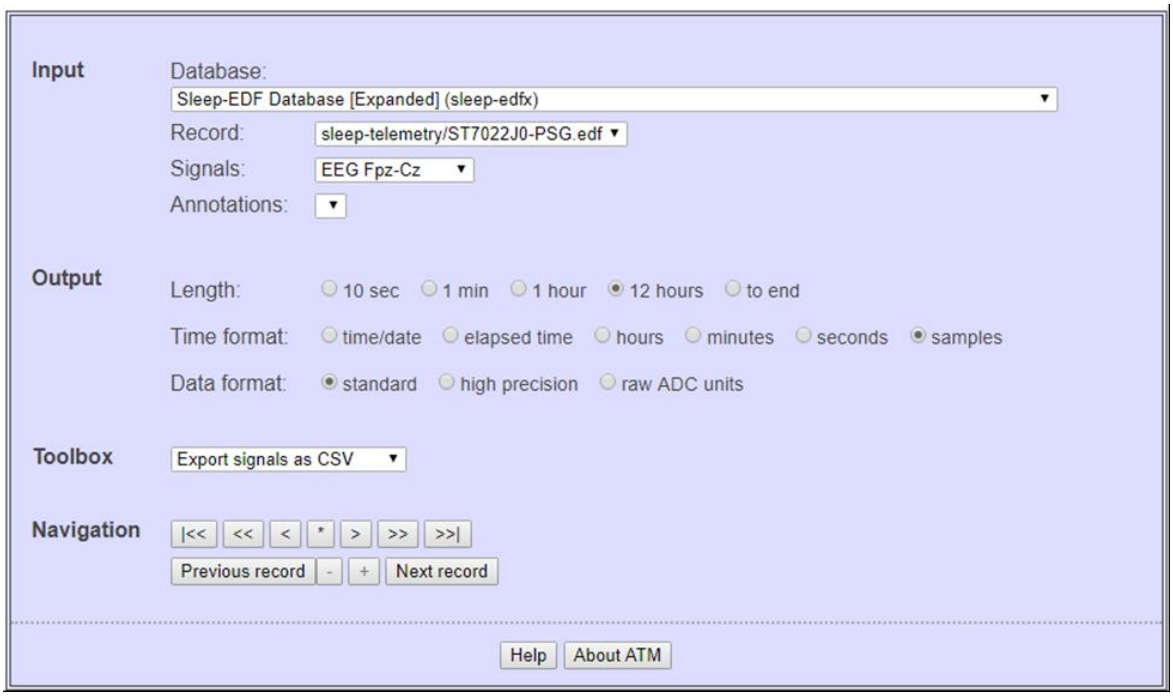


Figure 2-2 First EEG sample from unhealthy patient

For this research project I have used 12 hours long EEG signals. These EEG signals are obtained through the PhysioBank ATM in the PhysioNet (https://www.physionet.org/cgi-bin/atm/ATM). The input configurations for the three EEGs taken from this PhysioBank ATM are shown in Figure 2-2, 2-3 and 2-4. In these records files, SC means Sleep Cassette and these records were obtained through healthy subjects without any sleep related issues. ST means sleep telemetry and it contains records obtained from subjects who had mid difficulty in falling sleep, but were healthy otherwise.

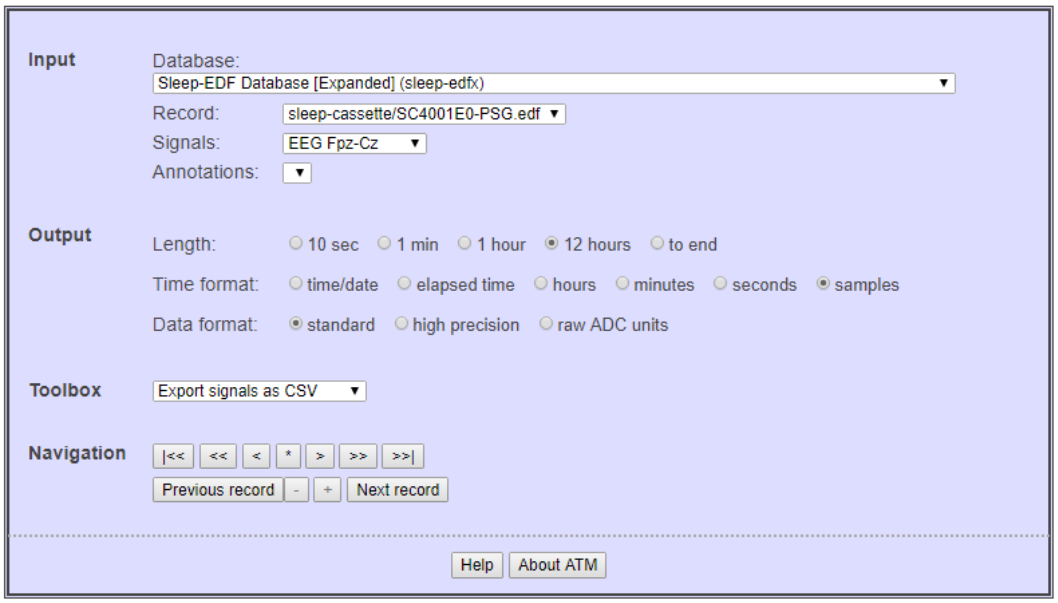


Figure 2-3 Second EEG sample from healthy patient

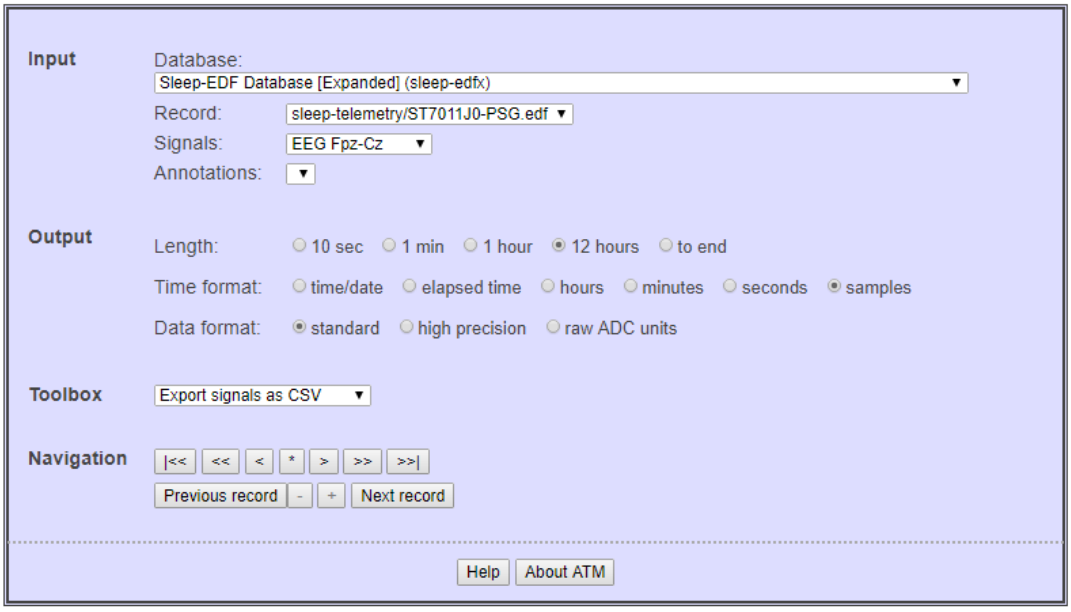


Figure 2-4 Third EEG sample from unhealthy patient

2.1.2 What features to extract?

After obtaining EEG signals, in order to classify them, it is required to extract statistical features which can represent them accurately. Features that can be extracted from EEG signals includes [1], [4].

- Power spectral density
- Phase synchronization
- Energy of different bands
- Discrete Wavelet Transform (DWT)
- Zero crossing histogram

From the above available feature extraction methods, DWT has been selected for this project, because it is suitable for analysing spontaneous signals like EEG and also there are many researches related to EEG signals, have already been done using DWT [1].

In DWT, an analog signal is decomposed in to set of coefficients in each sub band as described in the 2.1.3 section below. From these coefficients also one set of features has to be selected that is convenient to be used in the classification method. These include [6], for each sub band, taking maximum or minimum from all the wavelet coefficients or taking mean or standard deviation for all wavelet coefficients. In each sub band, the maximum from all the wavelet coefficients is used for testing this design as described below.

2.1.3 Discrete Wavelet Transform

Wavelet is a wave-like oscillation with zero mean that exist for limited time duration. It decays rapidly and captures both frequency and location information. There are different wavelets like Mexican hat, Morelet, Symlets 4 (sym4), Haar, Daubechies 2 (db2), db4, db6. Based on the calculations done on Section 2.4.2 for this project, I have used the data generated using ‘db6’ wavelet. Also, the smoothing

features Daubechies wavelets are more suitable for the detecting changes in EEG signals [7].

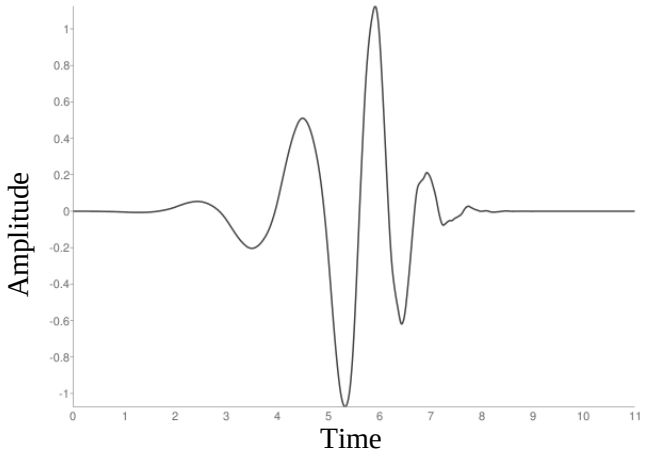


Figure 2-5 Daubechies 6 (db6) wavelet function

The Wavelet Transform is defined by:

$$W_f(s,\tau) = \int f(t)\, \Psi^*_{(s,\tau)}(t)dt \tag{2-1}$$

The wavelets are created from a wavelet function called “mother wavelet” and t is defined as:

$$\Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\Psi\left(\frac{t-\tau}{s}\right) \tag{2-2}$$

Here “s” is the dilation factor and “τ” is the translation factor. The signal f(t) is sampled by dilated and moved mother wavelet versions.

The wavelet transforms can be calculated in two ways. That is discreate manner and continuous manner. Those transforms are called Discrete wavelet transform (DWT) and Continuous Wavelet Transform (CWT). The continuous wavelet transform is calculated by taking summation over all time of the signal which is multiplied by scaled, shifted wavelet. CWT can operate at every scale and it is continuous in terms of shifting. In contrast to that, in DWT the scaling and shifting is

done based on power of two. Hence it is more accurate and efficient. This is done through calculating the approximations and the details for the signal.

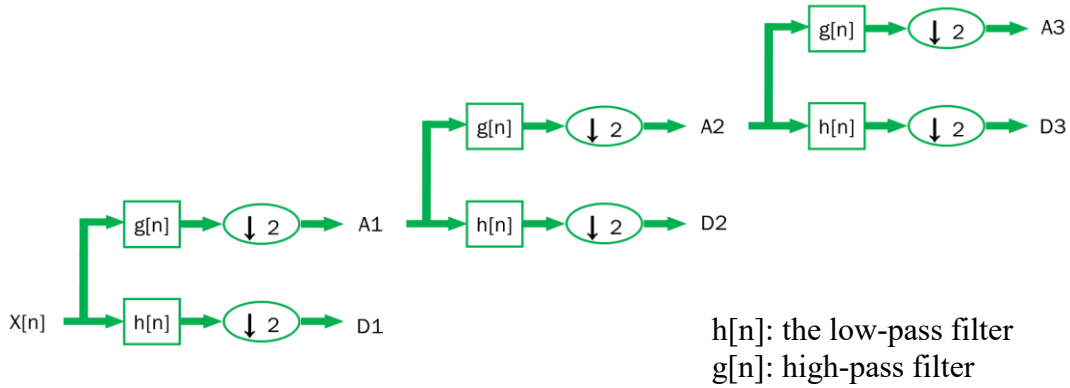


Figure 2-6 Decomposition in discrete wavelet transforms

2.1.4 Calculation of Approximations and Details

The low-frequency, high-scale components of the signal are called the approximation and the high-frequency, low-scale components called the details. High-pass and low-pass filters in each stage are used for the decomposition of approximations and details from the signal as described in the Figure 2-5.

In this project, after calculating Approximations and Details for the EEG signal, following features have been selected to be used in classification.

- Maximum A1 co-efficient for each sample
- Maximum D1 co-efficient for each sample

2.2 Classification of EEG signals

After extracting the features from EEG signals, those can be used to process and classify the EEG signals. Many of the researches have been done based on classification methods such as Support Vector Machine (SVM), principal component

analysis (PCA), Linear Discriminant Analysis, K-nearest neighbors (KNN) and other patter recognition methods.

2.2.1 Selection of classification method

Different classification methods have different efficiency for different application and different testing parameters. According to [4] and [8], Support Vector Machine (SVM) have higher accuracy than LDA. As described in the Figure 2-6 below, the feelings are more accurately classified by SVM [8].

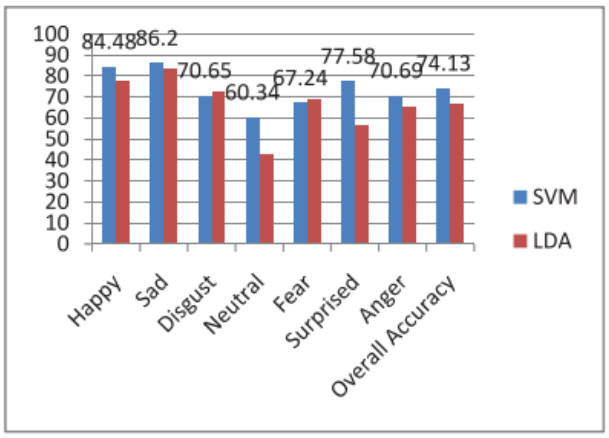


Figure 2-7 Average percentage accuracy of SVM and LDA

Table 2-2 Classification accuracy of different classification methods

Classification Method	Classification Accuracy	
	Dataset I (Motor Imagery)	Dataset II (Finger Movement)
LDA	82.86%	84%
QDA	78.57%	79%
KFD	80.71%	81%
Linear SVM	82.86%	82%
Gaussian SVM	84.29%	84%
MLP	80.71%	81%
LVQ	77.86%	80%
k-NN	84.29%	83%
DT	82.14%	86%

As mentioned in the Table 2-3, according to [2], LDA is not the best classification method for EEG signals, yet it is important to use in classification, since

with actual data there are no one best classification method and every method have different rate of accuracy for different test parameters. Also, it is important to note that the accuracy differences among different classification methods are not significant.

Also, when compared to other classification systems like SVM and PCA, LDA algorithm is less complex, especially for two class classification. And the implementation of less complex algorithm results in less resource utilization in the design which helps to reduce the overall cost. Therefore, for the low-cost classification requirements it is better to select lesser complex algorithms like LDA.

For this implementation, I have selected LDA as the classification method, based on less complexity and also since in actual implementation of a product it is also important to LDA along with other classification methods.

2.2.2 Classification using Linear Discriminant Analysis

LDA is a procedure that uses linear combination of features to differentiate two or more classes of events or objects. LDA create new axis by maximizing the distance between means, while minimizing the variation. The aim of this procedure is to reduce dimensionality, keeping information which helps for class discrimination as much as possible.

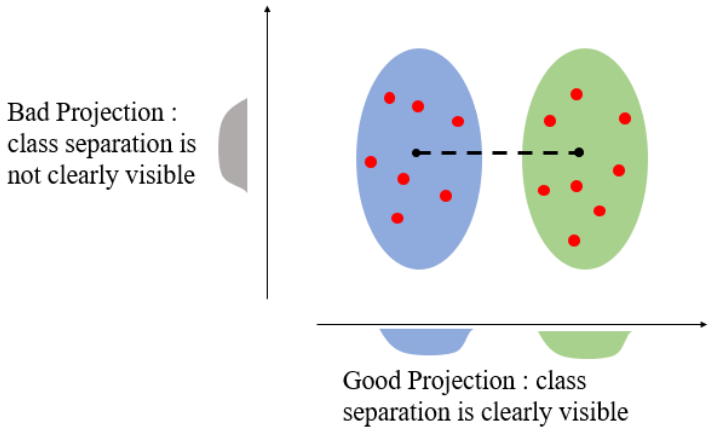


Figure 2-8 Data projection on different axes

As an example, Figure 2-8 shows two classes of objects and two axis which can be used to project the data. The vatical axis is a bad projection, since with that two classes are not distinctive with mean and variance of the classes. The horizontal axis is a good projection, since with that two classes are distinctive with mean and variance of the classes [9]. Likewise, LDA will find a projection axis which will have the maximum separation of two classes. This new axis will be created by LDA as described in Figure below.

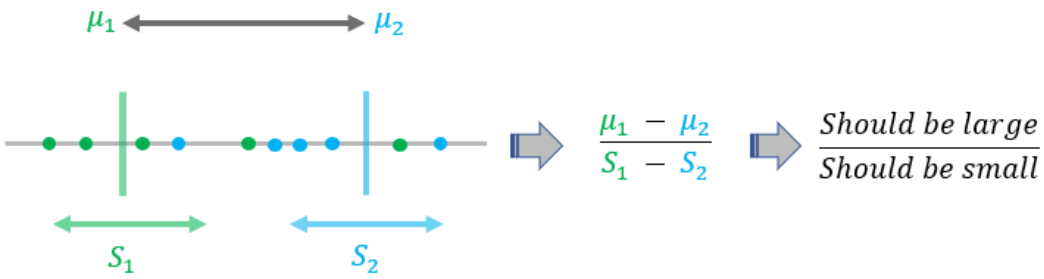


Figure 2-9 New axis creation in LDA

In this project, algorithm will be implemented for two class scenarios. And there are two phases of classification. That is, the Training phase where it calculates projection vector W for the largest eigen value for the system and the Deciding phase where it calculates new (y) values for hyper plane.

- Training phase:

In this phase the system will be trained based on the training sets available. For that, following mathematical operations will be applied on the data sets of each class.

1. Mean calculation for each training class

$$\mu_1 = \frac{1}{N_1}\sum X_1 \tag{2-3}$$

$$\mu_2 = \frac{1}{N_2}\sum X_2 \tag{2-4}$$

Here N_1 and N_2 are the number of samples for 1st and 2nd class respectively.

2. Covariance matrix calculation for each training class

$$S_1 = \sum (x_1 - \mu_1)(x_1 - \mu_1)^T \quad (2\text{-}5)$$

$$S_2 = \sum (x_2 - \mu_2)(x_2 - \mu_2)^T \quad (2\text{-}6)$$

3. Within class scatter matrix. This is calculated from the sum of covariance for each class.

$$S_w = S_1 + S_2 \quad (2\text{-}7)$$

4. Inverse within class scatter matrix

$$S_w^{-1} \quad (2\text{-}8)$$

5. Projection vector W calculation for maximum eigen value

$$w^* = S_w^{-1}(\mu_1 - \mu_2) \quad (2\text{-}9)$$

- Testing phase or Deciding phase

In this phase, the class of the testing sample will be decided based on the distance it has for the projection vector. In order to decide the class, the testing sample date will go through following steps.

1. Mean calculation for each training class

$$\mu_t = \frac{1}{N_t} \sum X_t \quad (2\text{-}10)$$

Here N_t is the number of samples that is used for testing.

2. Calculate the deviation of test sample from each class along the projection vector

$$d_1 = |(\mu_1 - x)w^*| \quad (2\text{-}11)$$

$$d_2 = |(\mu_2 - x)w^*| \quad (2\text{-}12)$$

This will give the absolute distance for two classes along the projection vector from test samples.

3. Classify the test sample based on the absolute deviation

$$if\, d_1 < d_2 : then\ the\ test\ EEG\ signal\ belongs\ to\ 1st\ class$$

$$if\, d_1 > d_2 : then\ the\ test\ EEG\ signal\ belongs\ to\ 2nd\ class$$

2.3 Architecture design

2.3.1 Algorithm for the full system

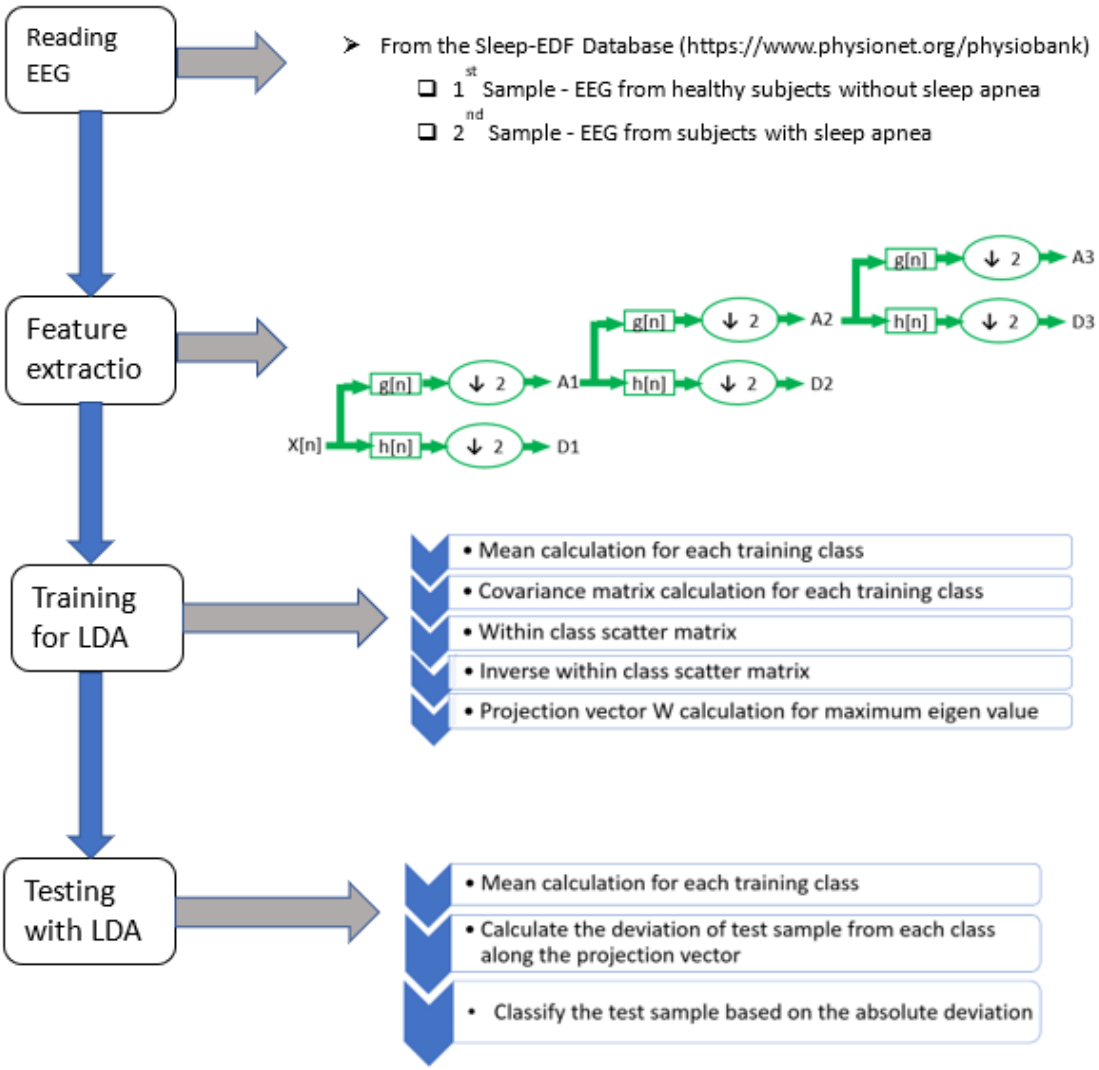


Figure 2-10 Algorithm for the EEG classification system

Figure 2-10 shows the full algorithm for the EEG classification system, which was described in section 2.1 and 2.2 above. Here 'Reading EEG' and 'Feature extraction' is implemented on software platform. And the 'Training for LDA' and 'Testing with LDA' is implemented on hardware.

2.3.2 Architecture of the full design

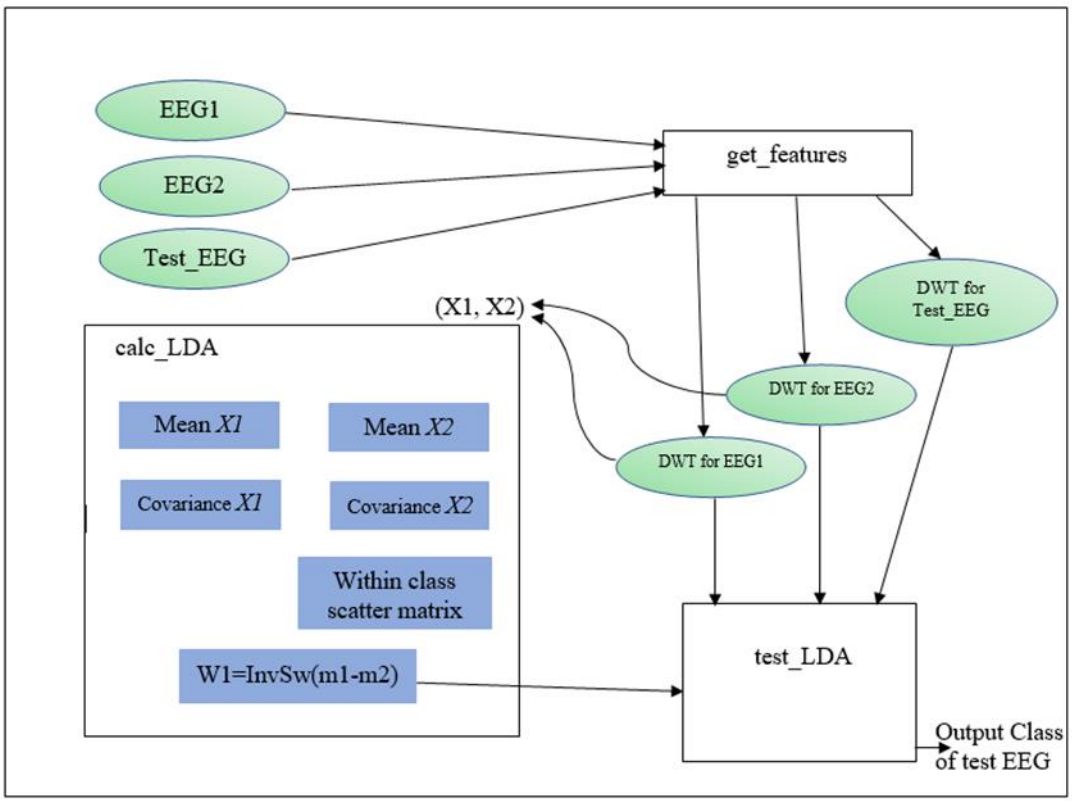


Figure 2-11 Architecture for the EEG classification system

Figure 2-11 above shows the full architecture of EEG classification system. This includes hardware implemented part as well as software implemented part. This full system was first implemented on MATLAB software. Therefore, all the module parts and their functionality are described in 2.4 section below. From these modules, calc_LDA and test_LDA has been used to implement the EEG classification part on hardware.

2.2.3 Requirements to process EEG signals offline

When designing the architecture, it should be decided whether to use real time EEG processing or offline EEG processing for the implementation. For that, advantages and disadvantages of each method should be considered.

To do online EEG processing, this project should be enhanced using Ethernet IP so that it can communicate to a PC in real time, to get the extracted features from EEG data to be processed. But for this implementation, EEG signals are processed offline due to following reasons.

1. Obtaining EEG signals are in the range of milliseconds, even though the processing of EEG data is in the range of micro seconds. (check this after implementation). Therefore, it will not be a bottleneck even though it was implemented separately.
2. This project focuses on increasing the efficiency of EEG classification part
3. In a real product, the part which reads EEG signals can be, implemented without the use of FPGA

2.4 MATLAB implementation

Before implementing the system on RTL, I have implemented the same system in MATLAB. Objectives of implementing the full system in MATLAB is to,

- Generate the DWT features that is used in final hardware implementation
- Select more efficient wavelet transform and fixed-point precision for the implementation
- Debug the algorithm more easily before going to RTL implementation

2.4.1 Classification system

Full design including reading EEG signals, DWT feature extraction and LDA classification has been implemented on MATLAB. Functionality of the MATLAB

files are described in the table below. Also, it describes the functionality of the modules mention in the Figure 2-11 in section 2.3.2 above.

Table 2-3 MATLAB files for EEG classification system

MATLAB file	**Functionality**
eeg_classification_gui.m	This is the main file and it has the code for eeg_classification_gui.fig, which is shown in Figure 2-12.
eegclassify_gui.m	This file is called from the GUI and it corresponds to top most module in Figure 2-11. In the beginning it reads the EEG files and finally it gave the class of the test EEG. This also has all the intermediate data flows.
get_features.m	Corresponds to get_features module. It gives the maximums of each Approximations and Details for EEG signals.
calc_dwt.m	This is called from get_features module and calculated DWT for signals.
calc_lda.m	Corresponds to calc_LDA module. This calculates LDA for two input classes and outputs the highest eigen value
test_lda.m	Corresponds to test_LDA module. This module will decide the class of test signal based on eigen value of projection vector for other two signals
create_mif.m	This creates MIF (Memory Initialization File) files from the features extracted from DWT. These MIF files is used as inputs in RTL implementation

In addition to those files, I have also used following MATLAB internal functions to write and read intermediate data files.

Table 2-4 MATLAB internal function

MATLAB internal functions	**Functionality**
csvwrite	Write data in CVS format
csvread	Read files in CVS format
dlmwrite	Write find in to data with specific delimiter

The graphical user interface (GUI) for the system is shown in the Figure 2-11 below. From this GUI, two training samples and the test sample can be selected which should be given in CVS (Comma Separated Value) format. Then class of the test signal can be checked from "Classify EEG" button.

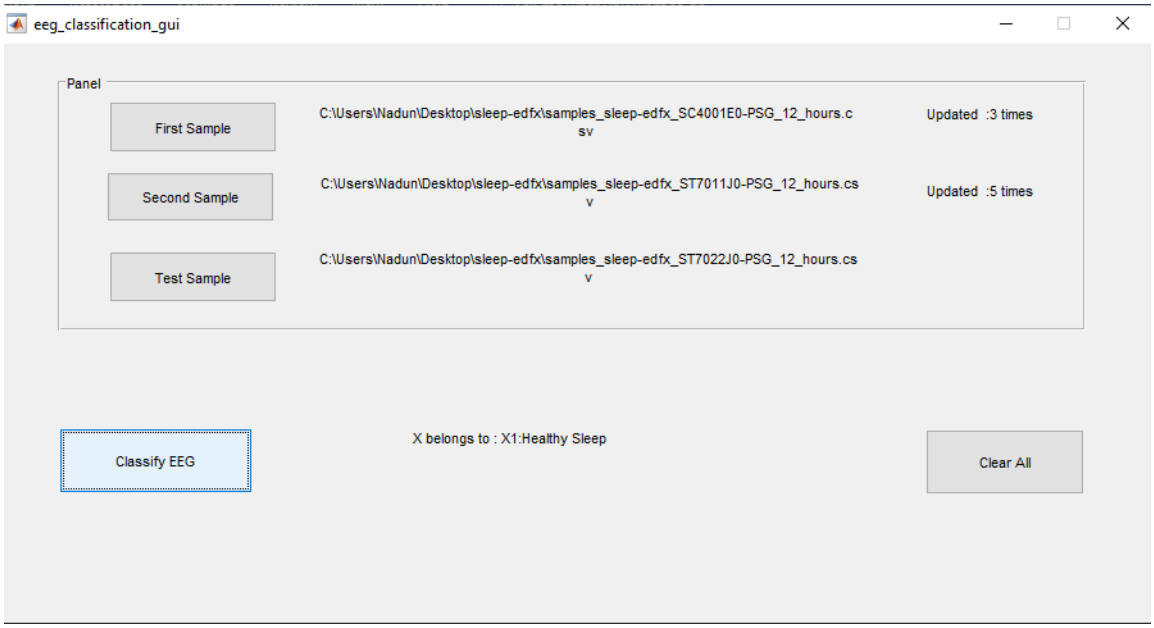


Figure 2-12 GUI for the EEG classifier

2.4.2 Wavelet and fixed points widths selection

Here I have used MATLAB to select most suitable DWT wavelet and the fixed-point width precision using K- fold cross validation [4]. Here I have used 5-fold cross-validation. In that the full data set is separated into 5 equal sets. From these 5 sets, the system is trained using 4 sets and the system is tested using 1 set. Then using that data, I have calculated accuracy, sensitivity and specificity [8] from the equations below.

Table 2-5 Definitions of TN, TP, FN and FP

Detection	With Obstructive Sleep Apnea	Without Obstructive Sleep Apnea
NO	FN (False Negative)	TN (True Negative)
YES	TP (True Positive)	FP (False Positive)

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \quad (2\text{-}13)$$

$$Sensitivity = \frac{TP}{(TP + FN)} \quad (2\text{-}14)$$

$$Specificity = \frac{TN}{(FP + TN)} \quad (2\text{-}15)$$

- Comparison of different DWTs

Using the equations mentioned above I have written following MATLAB files in the table to compare the different wavelets. This table does not include the same files mentioned in the Table 2-3.

Table 2-6 MATLAB files to compare the different wavelets

MATLAB file	Functionality
cross_validation.m	K fold cross validation for the EEG classification system. This gives Accuracy, Sensitivity, Specificity for the provided signals
eeg_classify.m	In the beginning it reads the EEG files and finally it gave the class of the test EEG. This also has all the intermediate data flows.

Accuracy, sensitivity and specificity calculated using those MATLAB files for different wavelets are shown in table below. From that table, it can be observed that Daubechies wavelet of order 6 (db6) have the maximum accuracy also higher sensitivity and specificity in general. Therefore, I have selected the db6 wavelet as the feature extraction method for the EEG classification.

Table 2-7 Comparison of different wavelets

Wavelet	Accuracy	Sensitivity	Specificity
haar	0.70	0.88	0.52
db2	0.76	0.72	0.80
db4	0.78	0.72	0.84
db6	**0.80**	**0.76**	**0.84**
bior6.8	0.72	0.60	0.84
sym4	0.60	0.52	0.68
sym2	0.76	0.72	0.80

- Comparison of different fixed points widths

Table 2-8 MATLAB files to compare the different fixed-point calculations

MATLAB file	Functionality
test_lda_precision.m	Same functionality as test_lda.m file mentioned in Table 2-3, but precision of the can be set by the user using *w* variable
calc_lda_precision.m	Same functionality as calc_lda.m file mentioned in Table 2-3, but precision of the can be set by the user using *w* variable

The accuracy, sensitivity and specificity for different fixed-point width has been calculated using the MATLAB files mentioned in the Table 2-7, in addition to the files mentioned in Table 2-6.

For this data set, as mentioned in table 2-8, it can be observed that the accuracy, sensitivity and specificity reduce when the width of fixed-point calculation reduces. Hence for this implementation I have used 32-bit width so that it has the accuracy as required, also the less resource utilization in FPGA implementation.

Table 2-9 Comparison of different fixed-point calculations

Word length of fixed-point calculations	Accuracy	Sensitivity	Specificity
64	0.7000	0.7600	0.6400
32	0.7000	0.7600	0.6400
16	0.6200	0.6000	0.6400

2.5 RTL Implementation

2.5.1 Architecture of RTL design

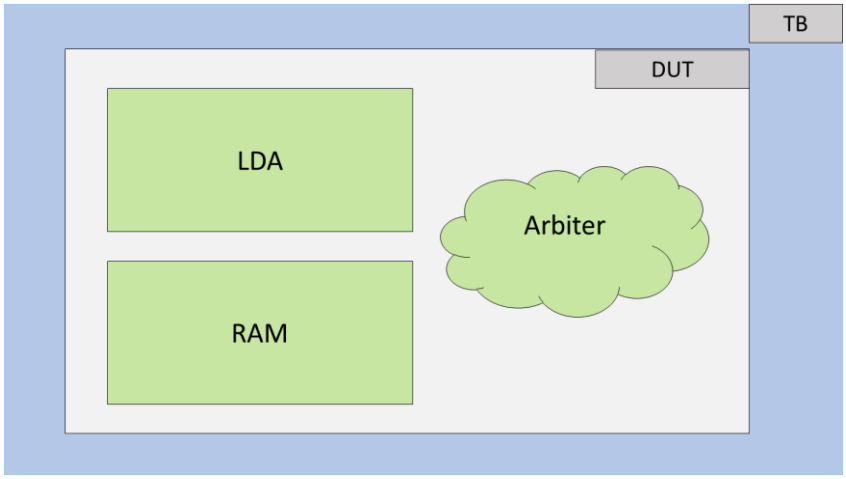


Figure 2-13 Architecture of the RTL design

Architechture of the RTL design is shown in the Figure 2-13 above. The functionality of each module in the design is described in the Table below.

Table 2-10 Functionality of RTL modules

Module	**Functionality**
TB (Testbench)	The wrapper module for the design which has simulation related data.
DUT (Design Under Test)	This is the RTL design which is implemented on FPGA
Arbiter	This module will decide the timing and the sequence of the status that the design operates. This is done through the Finite State Machine (FSM) in this module.
RAM	This stores the DWT features extracted from EEG signals for offline processing
LDA	This is the module where the LDA training and testing algorithms are been implemented

2.5.2 Finite State Machine in the Arbiter

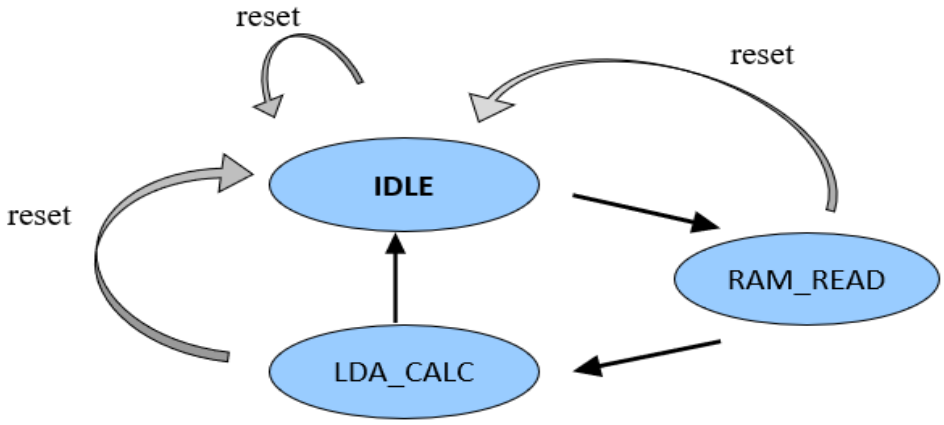


Figure 2-14 Finite State Machine in Arbiter

Above diagram describes the Finite State Machine (FSM) in the Arbiter module, which controls the sequence of states the design should operates. Here, clock pulse count is used to decide the timing to go to next state. And the reset signal can be used to get to the FSM to initial state again, which is IDLE. Each state has three signals

which are used for enabling output of RAM, calculation enable for LDA and output enable or LDA. Following are the Table 2-10 shows the various states of this FSM.

Table 2-11 States in Finite State Machine

State	**Description**
IDLE	This is the initial state of the design. No operation will happen in this state. Also, the design comes back to this state with the reset signal, which can be given by the user from the FPGA development board.
RAM_READ	During this state, the data stored in the RAM will be read and will be given to the LDA module.
LDA_CALC	All the LDA calculations will happen in this stage. This is the final state of the arbiter. Once this state is reached, the arbiter will remain in this status until user send a reset signal.

2.5.3 Fixed point calculations on Verilog

In this RTL implementation, to represent the floating points, Qn.m format has been used [10]. It is a fixed-point number system that can be used for floating points arithmetic. I have referred fixed-pint implementation available in OpenCores [11], for my project, by applying the following arithmetic procedures.

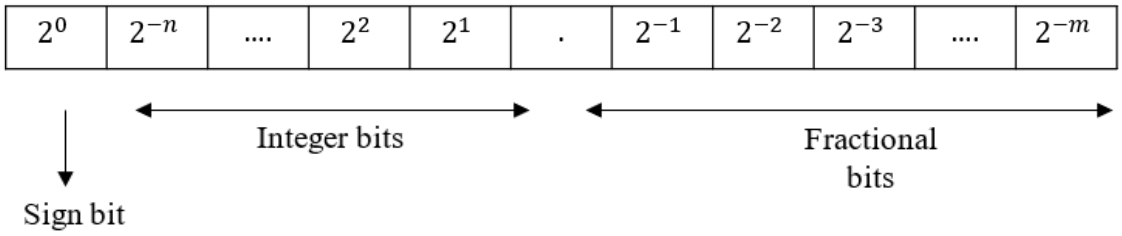


Figure 2-15 Qn.m format for Fixed-point Arithmetic

Figure 2-15 shows how floating points are represented in this format. The sign indicated by the most significant bit. For positive numbers it is 0. Qn.m means, to the left there are n bits and to the right there are m bits, from the binary point.

- Reasons to select fixed point arithmetic

Main advantage in fixed point implementation is, it has simpler hardware compared to floating point hardware, hence consume less power and required less resources. And it does not require normalization after each operation like in floating point arithmetic. Also, it is more convenient since place of decimal point is fixed.

There are some disadvantages also in fixed pint arithmetic. It is less readable and in multiplication operation id doubles the number of bits. However, for this project I have selected fixed point arithmetic, since it has more advantages comparatively for less complex hardware implementations.

Example: 01 1011 0000

In $Q_{2.8}$ format the value is $= 1 + ½ + (½)^3 + (½)^4$

$= 1.6875$

$= 1B0$ in Verilog

- Representation of negative numbers

Negative number in fixed point arithmetic are represented using the '1' in the sign bit of the number

Example: 10 1011 0000

In $Q_{2.8}$ format the value is $= -1 \times (½ + (½)^3 + (½)^4)$

$= -0.6875$

- Addition in Q.n.m format

Assume the two numbers of Qn1.m1 and Qn2.m2 formats are added respectively, and the result is given in Qn.m given format. Then the larger of n1 and n2 is the n and the larger of m1 and m2 is the m. Before the addition operation the points need to be aligned.

Example:

First number a in $Q_{2.8}$ format = 01 1001 0000

Second number b in $Q_{3.4}$ format = 010 0101

Addition (a+b) in $Q_{3.8}$ format = 011 1110 0000

Table 2-12 Addition in fixed point calculations

Number	**Align binary points**												**Value**
a		0	1	.	1	0	0	1	0	0	0	0	1.5625
b	0	1	0	.	0	1	0	1					2.3125
(a+b)	0	1	1	.	1	1	1	0	0	0	0	0	3.8750

- Multiplication in Q.n.m format

Assume the two numbers of Qn1.m1 and Qn2.m2 formats are multiplied respectively, and the result is given in Qn.m given format. Then the larger of n1 and n2 is the n and the larger of m1 and m2 is the m. In Fixed-point multiplication, the position of the binary point should be determined after the multiplication [12].

Example.:

First number a in $Q_{2.8}$ format = 01 1001 0000 == 1.5625

Second number b in $Q_{3.4}$ format = 010 0101 == 2.3125

```
            0110010000
             x   0100101
----------------------------
          01 1001 0000
         000 0000 000
        0110 0100 00
     0 0000 0000 0
    00 0000 0000
  011 0010 000
0000 0000 00
----------------------------
0011 1001 1101 0000        == 0011.1001 1101 0000  == 3. 61328125
```

2.5.4 Simulations

The functionality of a design is validated using the simulation. This is done through test bench in batch mode simulation as well as waveform simulation. Here I have done Post-synthesis, Post-implementation, functional and timing simulations. Functional simulation simply tests the functionality of the design. In timing simulation, in addition to functionality the delays are also considered, hence it is much closer to testing the RTL design in FPGA. It will allow to ensure that the implemented design meets all timing and functional requirements and have expected behavior after downloading in to the FPGA.

Also, it is important to check post-synthesis and post-implementation, since functional changes can be caused after synthesis and implementation due to reasons like, operation of asynchronous paths, differences between synthesis of HDL languages in various simulators, simulation and implementation mismatches caused by synthesis attributes or constraints. Simulation results for this design is presented in 3.2 Section.

2.6 FPGA Implementation

2.6.1 FPGA design flow

Below Figure 2-16 describe the design flow that is followed during the FPGA implementation. All the steps in this flow, except the first step and the last step, are generally done with the help of EDA tools provided by the FPGA vendor. In this project, I have used the Xilinx Vivado® Design Suite, which is compatible with Basys 3 Artix-7 FPGA Trainer Board.

RTL describe the functionality of the design in HDL The RTL should be simulated in order to verify that the design have the required functionality. Then in Logical Synthesis stage, RTL is converted it to a logic circuit, which consist of nets and logical cells. Logic Mapping and Logic Optimization also happens at this stage.

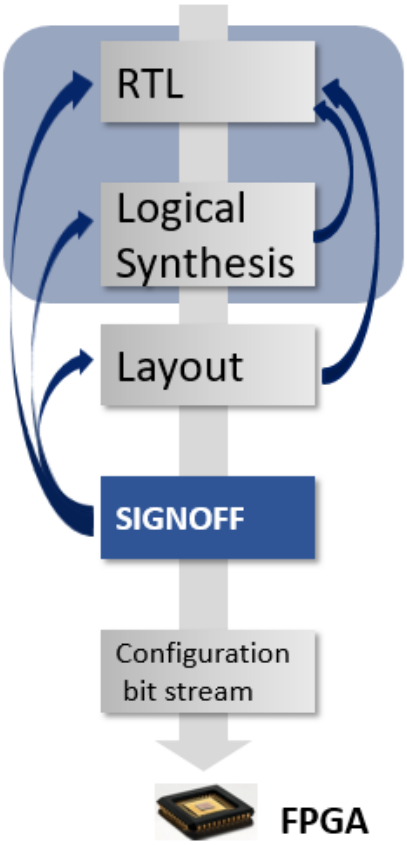


Figure 2-16 FPDA design flow

In Layout stage, the place components and Route the nets is done using the resources available in FPGA device. Then the design tool (Vivado IDE) runs series of must pass the timing and the power requirement. After that it will generate the bit stream according to the configuration provided by the user. Finally, user has to download that generated bit stream in to FPGA device in order to configure it.

2.6.2 Selection of a FPGA development board

I identified that FPGA (Field Programmable Gate Array) is the best way to start the initial implementation of this design. It provides the way to running the RTL designs inside the Reconfigurable logic. FPGA development boards have readily available rich set of tools and is helpful in initial prototyping an RTL design in early stages of ASIC design process. I targeted the Xilinx FPGA platforms due to wider availability, advanced development tools and wider support.

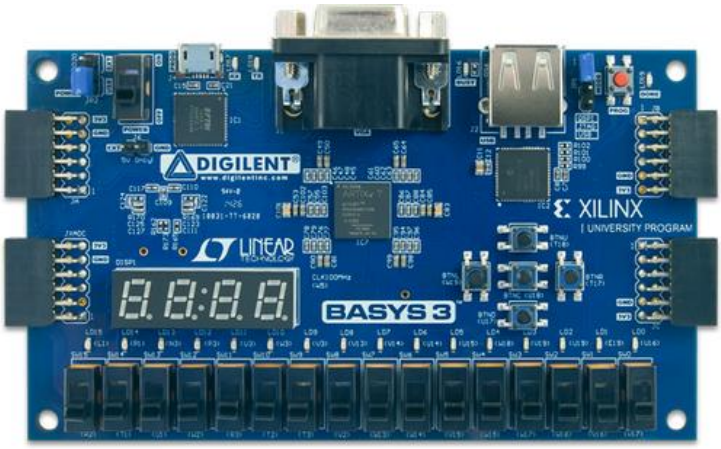


Figure 2-17 Basys 3 Artix-7 FPGA Trainer Board

In order to select best suitable board from the available FPGA development boards, I have compared their features and cost in the below Table 2.2.

Table 2-13 FPGA Boards Comparison

FPGA development kit	**Xilinx Virtex-7 FPGA VC707 Evaluation Kit**	**Atlys Spartan-6 FPGA Trainer Board**	**ZedBoard Zynq-7000 ARM/FPGA SoC Development Board**	**Basys 3 Artix-7 FPGA Trainer Board**	**Genesys 2 Kintex-7 FPGA Development Board**
FPGA	Xilinx Virtex-7 FPGA	Xilinx Spartan-6	Xilinx Zynq-7000 AP SoC	Xilinx Artix-7 FPGA	Xilinx Kintex-7™ FPGA
Memory	1GB DDR3 SODIMM 800MHz / 1600Mbps	2.1Mbits of fast block RAM	512 MB DDR3	1,800 Kbits of fast block RAM	Close to 16 Mbits of fast block RAM
Logic Cells	485,760	6,822	85,000	33,280	50,950
DSP Slices	2,800	58	220	90	840
Cost	$3,495	$490.00	$449.00	$149.00	$999.00
Decision	Enough resources, High cost, Suitable for complex projects	Not supported in Vivado new versions	Enough resources Cost is moderate	Enough resources, Cost is low	Enough resources, Cost is moderate

Based on the comparison in the table above, I have selected Basys 3 Artix-7 FPGA Trainer Board, because of its availability, low cost and it has enough resources to implement this project.

2.6.3 Specifying Constraints using XDC

In order to implement the RTL design in hardware, there are timing and physical configuration that needs to be specified. These are given to FPGA through Xilinx Design Constraints (XDC) file. XDC is an extension of the industry standard Synopsys Design Constraints (SDC). SDC is specifically designed for ASIC designing, therefore it only has Timing constraints. XDC consist of Timing constraints as we as Physical constraints, since in FPGA design flow, physical configuration also needs to be given for the development board. These constraints are order dependent, therefore constraints written in first part of XDC file will be overridden by constraints written in later. XDC file which is in .xdc format is shown in Figure below for this project.

```
##########################################################
##         Timing Constraints
##########################################################

## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -period 80.000 -name clk -add [get_ports clk]

set_input_delay -clock clk -min 1.000 [get_ports reset]
set_input_delay -clock clk -max 2.000 [get_ports reset]
set_output_delay -clock clk -max 4.000 [get_ports {{eeg_class[0]} {eeg_class[1]}}]
set_output_delay -clock clk -min -1.500 [get_ports {{eeg_class[0]} {eeg_class[1]}}]

##########################################################
##         Phisical Constraints
##########################################################

## LEDs
set_property PACKAGE_PIN P1 [get_ports {eeg_class[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {eeg_class[0]}]
set_property PACKAGE_PIN L1 [get_ports {eeg_class[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {eeg_class[1]}]

##Buttons
set_property PACKAGE_PIN U18 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports reset]

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```

Figure 2-18 Design Constraints using XDC

a) Timing Constraints

This includes clock information, input and output timing requirements and exceptions which overrides basic constraints. Performance expectations of the design are communicated to the implementation tool through timing constraints. There are two types of timing constraints. That is can be Global Timing Constraints and Path Specific Timing Constraints.

1) Global Timing Constraints

There are timing constraints which defined for entire design. Following are the main global timing constraints.

Period - (using create_clock) This constraint specifies delay paths between synchronous elements

Offset in – This constraint specifies delay paths from input pins to synchronous elements

Offset out – This constraint specifies delay path from synchronous elements to output pins

2) Path Specific Timing Constraints

These timing constraints are defined only for specific paths in the design.

set_multicycle_path - Defines the multicycle path.

set_false_path – Defines the false path

b) Physical Constraints

This includes all the constraints which are not timing specific like IO (input/output) constraints, floor planning, device configuration. For this design I have used two set of IO constraints. Those are, LEDs to show the final class of EEG test signal and a push button to reset the design, as shown in Figure 2-18.

2.6.4 Timing Closure

After RTL designing and defining design constraints, design tool will run the synthesis and implementation on the design. Then if there are any timing violations, tool will report them in timing report. These timing violations should be resolved before generating bit stream to download in to FPGA device.

Slack is related to difference between required time and the arrival time and negative slack means, the timing requirements are not met. To have timing closure, Worst Negative Slack (WNS) and Total Negative Slack (TNS) should be positive. Also, both WNS and TNS should be improved as much as possible. WNS will limit the maximum frequency that the design can run. This can be achieved by either reducing the clock speed or using different placer directives from Vivado IDE. Several placer directives can be tried to find a better directive for the design. If none of these methods do not improve the negative slack, RTL should be changed in order to correct the issue. This procedure is known as Timing Closure. Figure 2-19 shows an algorithm that I have used to improve negative slack during implementation in Vivado IDE. Results of the timing closure for this design is described in 3.1 section.

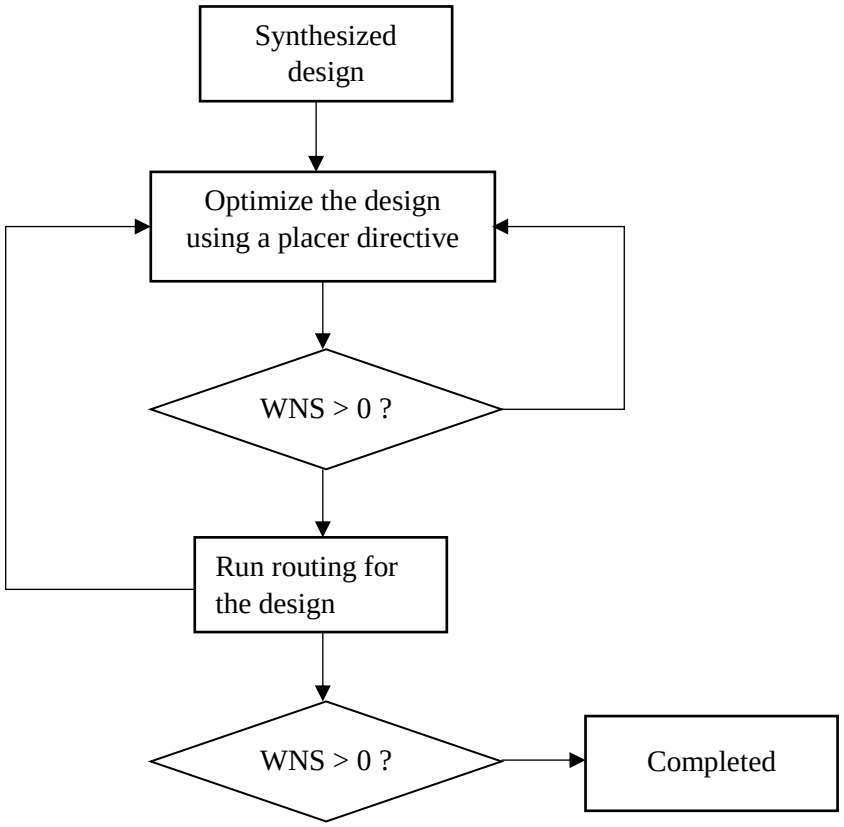


Figure 2-19 Flow to remove negative slack

2.6.5 Power Utilization

It is important to implement any design to consume less power as possible. There are two sources for the power consumption in FPGA devices. That is Static power consumption and Dynamic Power consumption. Static power refers to power dissipation due to leakage current when the device is in standby mode. Dynamic power dissipation happens when the capacitance of logic cells charges and discharges when the device is operating.

Power optimization is done in the synthesis and implementations stages from the design tool. However, it is possible to reduce the power consumption by writing the RTL in optimal manner. I have written the optimal mathematical operations and efficiently used the if else, case statements, so that there will be less power consumption after implementation. The power consumption of this design is described in 3.1 section.

2.7 Feasibility on Arduino implementation

When implementing any system on hardware, most of the time the first choice is to implement it on Arduino board. Therefore, in this section I address the importance of implementing the EEG classification system on FPGA instead of Arduino.

2.7.1 Implementation on Arduino Mega 2650

To implement the EEG classifier, I have selected Arduino Mega 2650. Even though there are many low-cost Arduino development boards available that this one, all of them did not have enough memory to store the source code and the extracted EEG feature data. I have simulated this on “Proteus” software as shown in Figure 27. And also, was able to get the simulation data as shown in Figure 2-20.

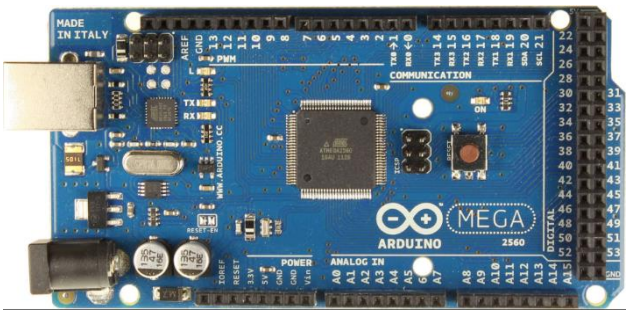


Figure 2-20 Arduino Mega 2650

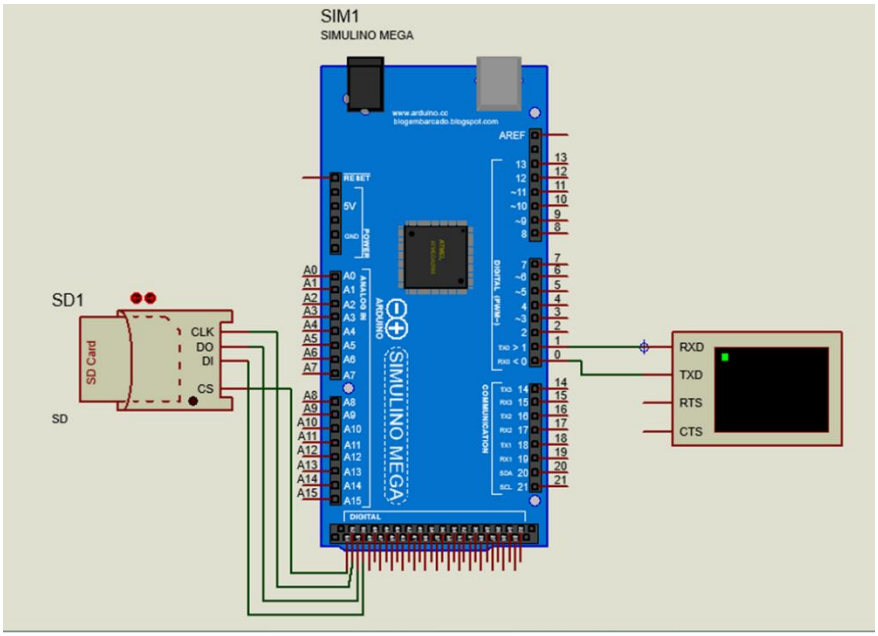


Figure 2-21 Simulation on Proteus with Arduino Mega 2560

```
Sw
1177.40  10285.64  532.77  19954.76  21327.74  12905.64  50037.44  85899.57
10285.64  92958.64  4600.12  182539.21  195428.39  116945.28  456597.93  783784.25
532.77  4600.12  4632.79  10495.63  9152.10  9817.55  32667.34  55545.86
19954.76  182539.21  10495.63  367164.06  390140.21  237096.60  917987.50  1575298.50
21327.74  195428.39  9152.10  390140.21  417858.31  247482.00  971192.18  1667489.62
12905.64  116945.28  9817.55  237096.60  247482.00  175568.79  618311.93  1058448.37
50037.44  456597.93  32667.34  917987.50  971192.18  618311.93  2341050.00  4015052.00
85899.57  783784.25  55545.86  1575298.50  1667489.62  1058448.37  4015052.00  6886597.00

InvSw
0.05  -0.01  -0.00  -0.00  0.00  -0.00  0.02  -0.01
-0.01  0.00  0.00  0.00  -0.00  0.00  -0.01  0.00
-0.00  0.00  0.00  -0.00  0.00  0.00  -0.00  0.00
-0.00  0.00  -0.00  0.00  -0.00  0.00  -0.00  0.00
0.00  -0.00  0.00  -0.00  0.00  -0.00  0.00  -0.00
-0.00  0.00  0.00  0.00  -0.00  0.00  -0.00  0.00
0.02  -0.01  -0.00  -0.00  0.00  -0.00  0.03  -0.02
-0.01  0.00  0.00  0.00  -0.00  0.00  -0.02  0.01

m1_m2
4.45
7.14
57.01
24.76
9.33
-6.85
-25.66
-40.97

W1=invSw*(m1-m2)
0.06  -0.01  0.02  0.01  0.00  0.01  -0.07  0.04

D1
2.67

D2
0.38
X belong to class 1 : No sleep apnea
```

Figure 2-22 Simulation results from Proteus

2.7.2 Necessities to implement on FPGA

Following are the necessities to implement on FPGA device instead of Arduino micro controller.

- Memory available on Arduino chips are not enough to process huge amount of data. Even with Arduino Mega 2560 which has 256KB flash memory, there was a memory overload for this design. And also, this Arduino board costs $38.50. Hence it is not cost effective also.
- FPGA Implementation can process data much faster than the Arduino implementation. It can scale the design to be used with more data. Also, it can enhance the design to be used with real time data by using adaptive LDA
- Apart from the implementation, there are also limitations on simulating the design when using Arduino. I have faced below memory allocation issues while during Proteus simulations and was unable to proceed. Main issue here is the Arduino chips as well as the Arduino simulation software are not design to implement more complex system like EEG classifier.

  Invalid opcode 0xFFFF at PC=0xDA10

Hence it is more efficient and convenient to implement this EEG clarifier on FPGA instead of Arduino.

Chapter 3
RESULTS AND DEMONSTRATION

Results of my project "Hardware Implementation of EEG classifier" is presented in this chapter. The main target of my project is to implement a working EEG classifier for two class scenarios in a FPGA with minimal resources. For that it is necessary to show that my design can classify the EEG signal accurately and also it utilizes minimum resources. In addition to that, it is required to show that the design meet all the timing and performance requirements. These results and the demonstration on the FPGA implementation are present in this chapter.

3.1 Timing and Power results for FPGA implementation

Timing summary of the design is shown in the Figure 3-1 below. Here it can be observed that there are no timing violations, no setup time and hold time violations.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.630 ns	Worst Hold Slack (WHS):	0.122 ns	Worst Pulse Width Slack (WPWS):	39.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	1339	Total Number of Endpoints:	1339	Total Number of Endpoints:	1011

All user specified timing constraints are met.

Figure 3-1 Design Timing Summary

Figure 3-2 shows the Timing report for the design. This can be generated using the following TCL command.

report_timing_summary -file timing.rpt

Also, the minimum time requirement in order to run the design and the maximum frequency that the design can operate can be calculated from the 'Max Delay Path' section of that report as shown below. From this report it can be observed that the timing requirement is 80ns and there is a positive slack of 0.630ns. That means clock

period for *clk* signal can be 0.630ns shorter, hence maximum frequency would be 12.58MHz for implementation in this FPGA board, which depends on the RTL.

Minimum time	= 80ns-0.630ns
	= 79.370ns
Maximum Frequency	= 1/(80ns-0.630ns)
	= 1/79.370
	= 12.60MHz
Design clock frequency	= 1/80ns
	= 12.5 MHz

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.630ns  (required time - arrival time)
  Source:                 ARBITER/FSM_onehot_pres_state_reg[2]/C
                            (rising edge-triggered cell FDCE clocked by clk  {rise@0.000ns fall@40.000ns period=80.000ns})
  Destination:            seg[3]
                            (output port clocked by clk  {rise@0.000ns fall@40.000ns period=80.000ns})
  Path Group:             clk
  Path Type:              Max at Slow Process Corner
  Requirement:            80.000ns  (clk rise@80.000ns - clk rise@0.000ns)
  Data Path Delay:        70.260ns  (logic 31.923ns (45.435%)  route 38.338ns (54.565%))
  Logic Levels:           81  (CARRY4=41 LUT2=19 LUT3=6 LUT4=9 LUT5=1 LUT6=4 OBUF=1)
  Output Delay:           4.000ns
  Clock Path Skew:        -5.074ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    0.000ns = ( 80.000 - 80.000 )
    Source Clock Delay      (SCD):    5.074ns
    Clock Pessimism Removal (CPR):    0.000ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns

    Phase Error              (PE):    0.000ns
```

Figure 3-2 Timing Report

As mentioned in 2.6.3 Section, clock cycle of clock signal for this design is set to be 80ns using XDC physical constrains. Which means the provided clock is 12.5MHz. Since this is less that maximum frequency 12.60MHz, design runs without any timing issues.

Power consumption for the design is shown is the Figure 3-3 below. It has low confidence level since the power specifications are not defined in the design. Defining power specifications are not required in this stage of the design. That will be required when this design is going for production. And also, this design has total on-chip power

consumption of 0.138W, which is due to both static and dynamic power consumption (Described in 2.3.5 Section).

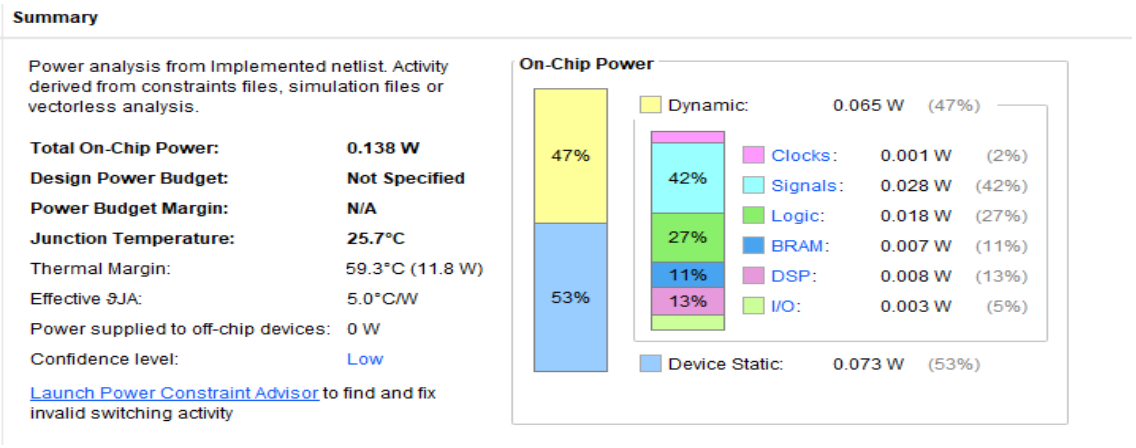


Figure 3-3 Power Consumption

3.1.1 Modular timing analysis

Module timing analysis is important to analyze the timing issues modular wise and also to get to know about what modules should be considered for the parallelization, to speed up a design. For this design, here I have considered LDA, RAM and Arbiter submodules for this analysis, which are shown in figure 2-13. Here LDA is a combinational module while RAM and Arbiter are sequential module. Table 3-1 below shows the worst slack for the setup times and hold times for these modules. From that, it can be observed that, all three modules have same worst slack for setup times and RAM module have the lowest worst slack for hold times. Hence to improve the slack for hold time of overall design, the RAM module should be considered

Table 3-1 Worst slack for the setup times and hold times for sub modules

Sub module	ARBITER	RAM	LDA
Setup time	0.630ns	0.630ns	0.630ns
Hold time	6.604ns	0.122ns	6.180ns

3.2 Simulation results

- Batch mode simulation

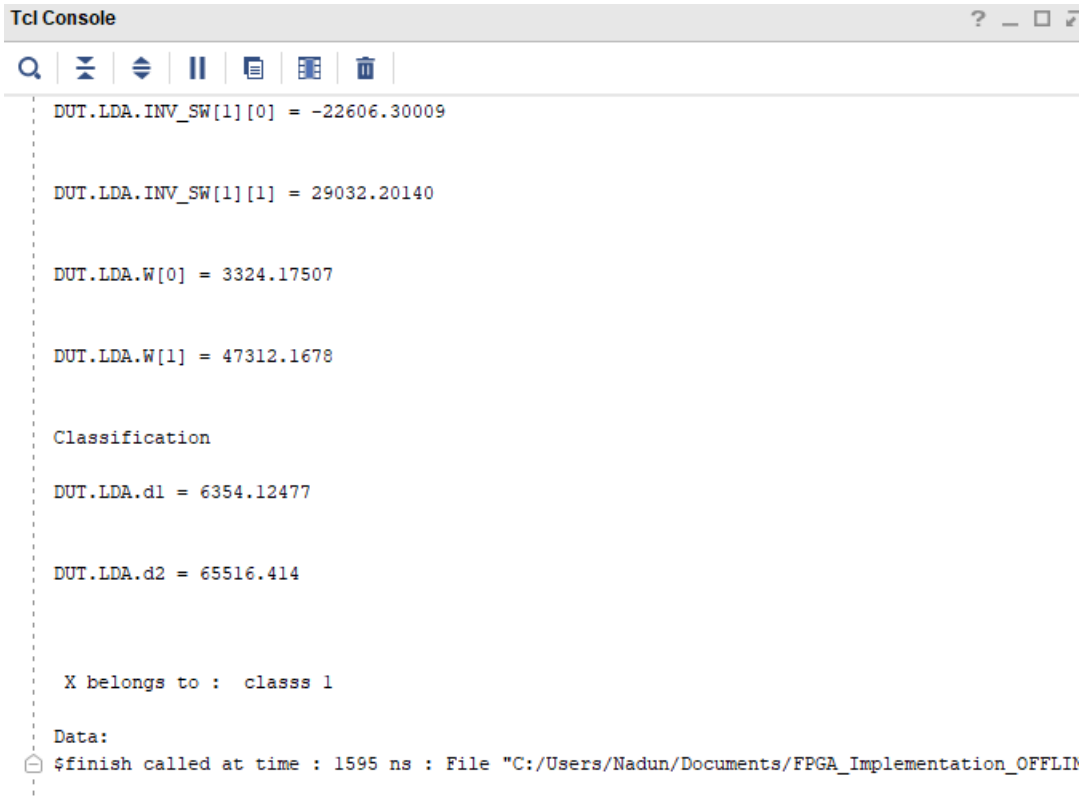


Figure 3-4 Batch mode simulation

Figure shows that the batch mode simulation gives the correct output for this design. Apart from testing the final output, batch mode simulation helps debug the design since from that it is easy to print the required intermediate values. In the Figure 3-4, I have printed one approximation, detail and the distance for each projection vector before taking the final decision, to which class the test EEG belong.

- Waveform mode simulation

In addition to batch mode simulation, to debug the timing it is important to check the behavior through the waveform. From Figures 3-5 to 3-9, it can be observed that behavioral simulation as well as post-synthesis, post-implementation, timing and

functional simulations waveforms show the correct class value for the input EEG signal. In addition to that this also shows the reading of RAM when ‘oe’ signal is high.

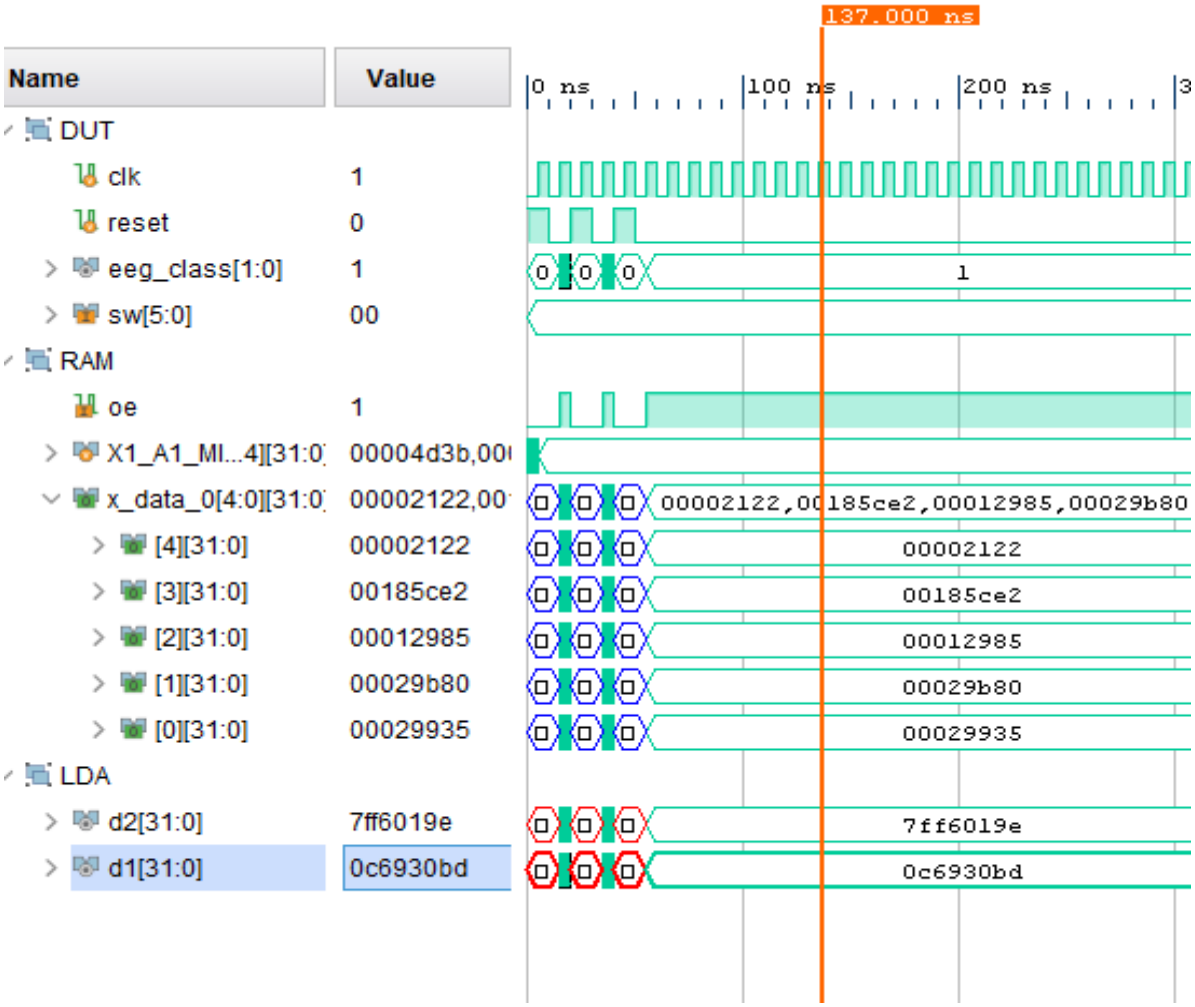


Figure 3-5 Behavioral Simulation

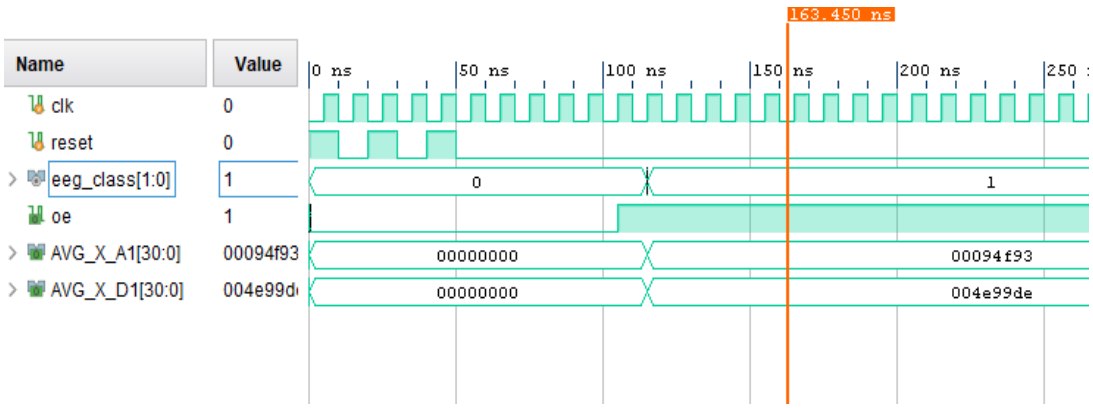


Figure 3-6 Post-Synthesis functional simulation

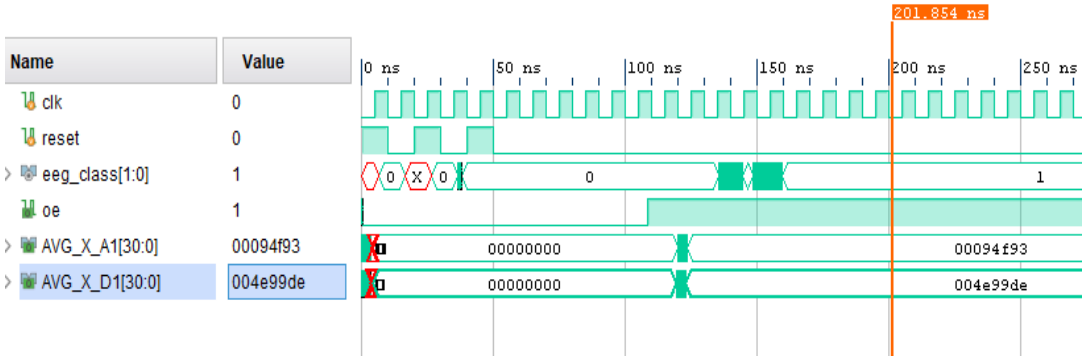
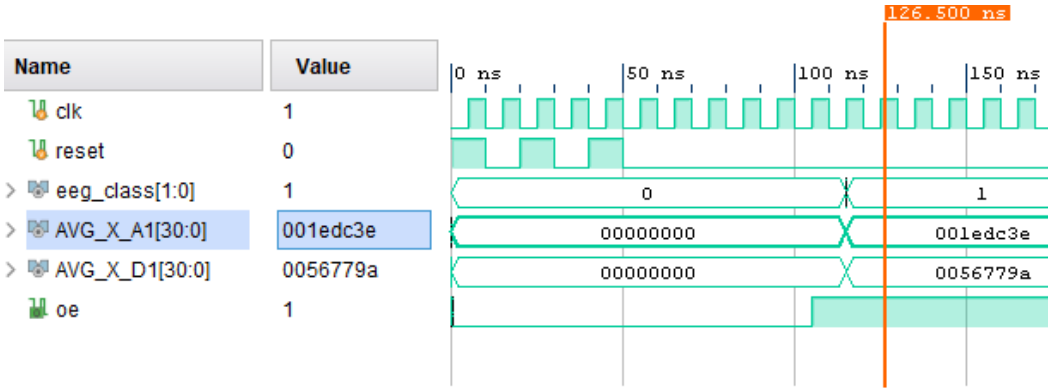


Figure 3-7 Post-Synthesis timing simulation



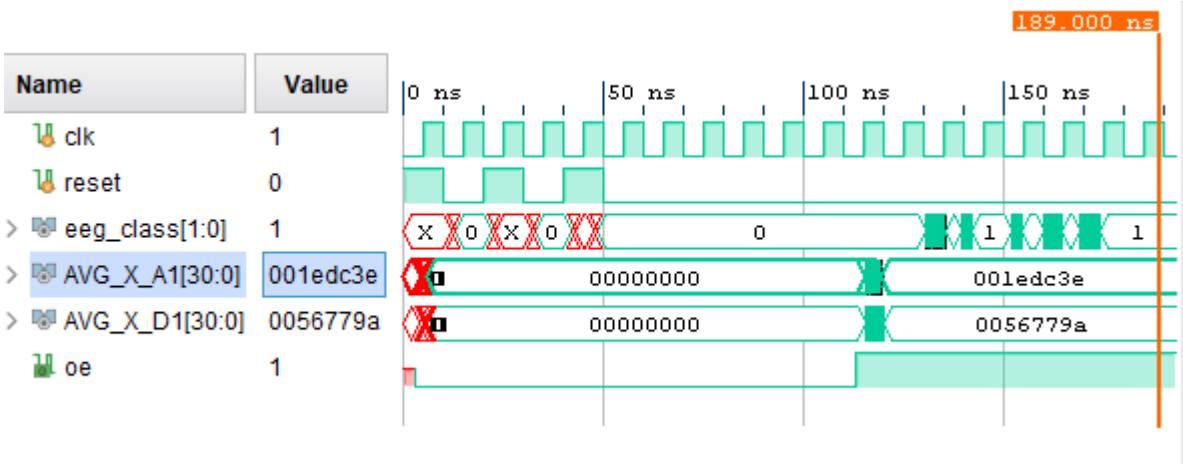


Figure 3-9 Post-Implementation timing simulation

3.2 Demonstration on FPGA development board

Figure shows the implementation of this project in the Basys3 FPGA trainer development board. In this, when the reset button (BINC push button) in push, two LEDs will reset and will tuned off. Otherwise it will show the binary value of the class which the test EEG belong to. In this picture, LD0 LED is tuned off and LD1 LED is tuned on, which means that the class it belongs is 2b'10 or 2nd class.

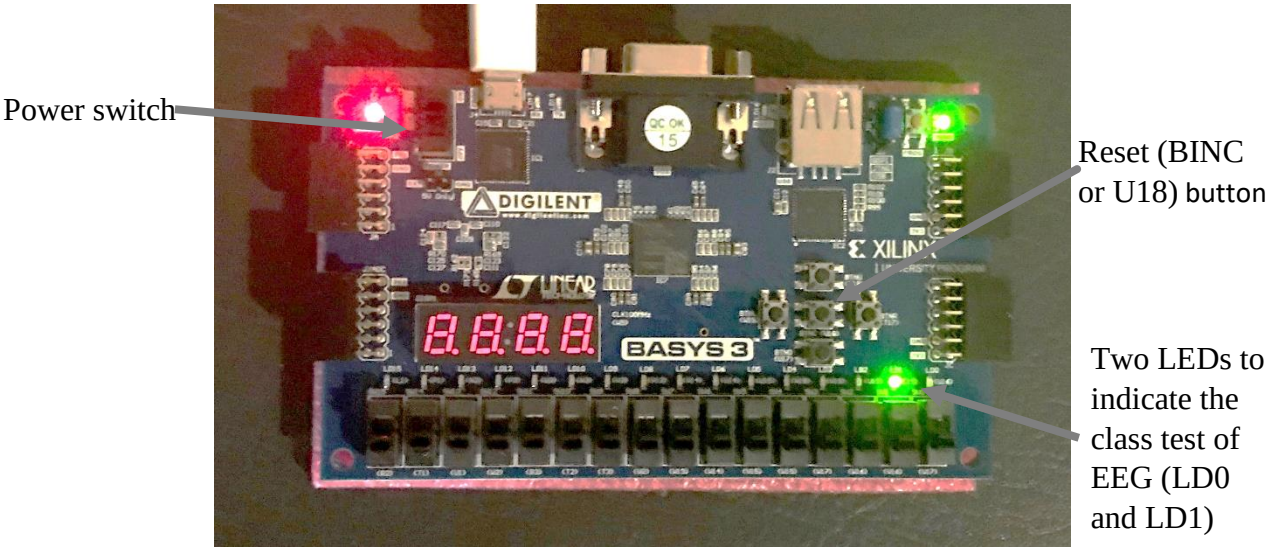


Figure 3-10 Demonstration on Basys3 FPGA board

For the demonstration I will make use of Quads SPI(Serial Peripheral Interface) flash memory (Macronics part number is mx25l3233f) available in Basys 3 FPGA development board. The bit file generated from the program can be stored on this flash memory so that there is no need to reprogram the FPGA device every time when turn on the power.

3.4 Accuracy of hardware implemented design

After implementing the design on Basys3 FPGA development board, I have calculated the accuracy of the output for different the k-fold cross tests using the same formula mentioned in Section 2.4.2. To demonstrate this K fold testing, I have used

the ports and peripherals available in the FPGA development board as shown in the Figure 3-11. Here first 3 switches are used to select the EEG set and it will be shown in 2nd digit in 7 segment display. Similarly, K fold validation and true class can be set using switches and will be shown in other 7 segment digits as shown in Figure 3-11. After that from 1st digit will show the class of test EEG signal. This data can be use to calculate the accuracy of the output.

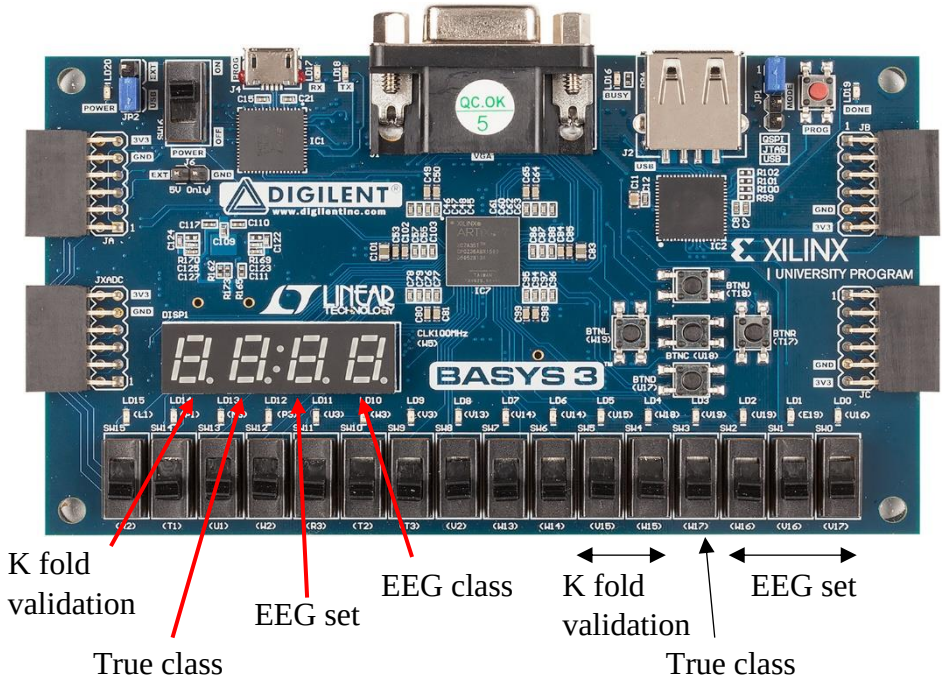


Figure 3-11 Inputs and outputs in K fold testing demonstration

Results from this test is shown in the Table below. Hence it can be concluded that by using 4-fold cross validation testing, this LDA classification design for EEG signals has 80% accuracy.

Table 3-2 Accuracy for different k-fold validation tests

Cross validation	Accuracy
4-fold	0.8000
5-fold	0.4000
6-fold	0.6000
7-fold	0.6000

3.5 Comparison of hardware and software results

Table 3-3 shows the comparison of accuracies between software implementation and hardware implementation for different cross fold validations. Here the software accuracy is been calculated from the MATLAB code mentioned in the 2.4.2 section. From this it can be concluded that the 4-fold validation has highest accuracy difference for hardware implementation which is calculated from (3-1) equation below.

Table 3-3 Comparison of accuracies for software and hardware implementation

Cross validation	**Accuracy of Software implementation**	**Accuracy difference Percentage of hardware implementation**
4-fold	0.6600	+21.21%
5-fold	0.8000	-50.00%
6-fold	0.7400	-18.92%
7-fold	0.5800	+3.45%

$$\frac{Accuracy\ differance\ for\ software\ and\ hardware\ implementaitons}{Accuracy\ of\ software\ implementation}\ X\ 100\% \qquad (3-1)$$

Chapter 4
DISCUSSION

In this chapter, I hope to discuss about the tools I have used, the problems I have encountered and how I have overcome those during my project. Also, finally I have discussed about the resource utilization of this project.

4.1 Commonly Used Tools

In the design process of the EEG classifier, I have used several software packages for different purposes as mention below.

4.1.1 Vivado Design Suite

Vivado Design Suite is a synthesis and analysis tool developed by Xilinx specifically for FPGA design flow. It also has an in-build RTL simulator. I have used this software as the IDE (Integrated development environment) for RTL coding, simulation, implementation and programming bit file generation. Design implementation includes all other design steps involved in the FPGA design flow like synthesis, translate, map, place & route.

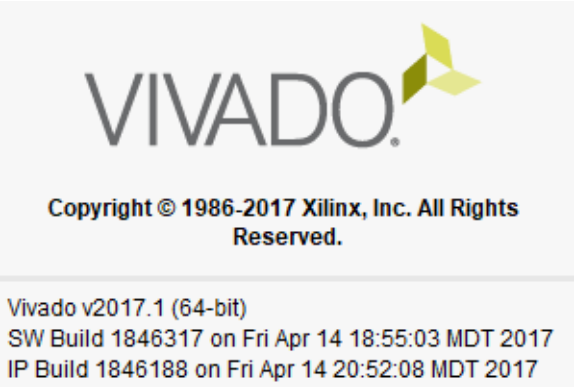


Figure 4-1 Vivado 2017.1

4.1.2 Xilinx ISE

I had to use Xilinx ISE in order to implement the design in Basys 2 board and Atlys Spartan 6 FPGA development boards. This software was developed by Xilinx, before developing Vivado software, and not it is discontinued from further developments.

Figure 4-2 Xilinx ISE 14.7

4.1.3 MATLAB

MATLAB is a numerical computing software developed by MathWorks. I have used this for the software implementation of this project. Also, the DWT features that has been used for hardware implementation was generated using MATLAB.

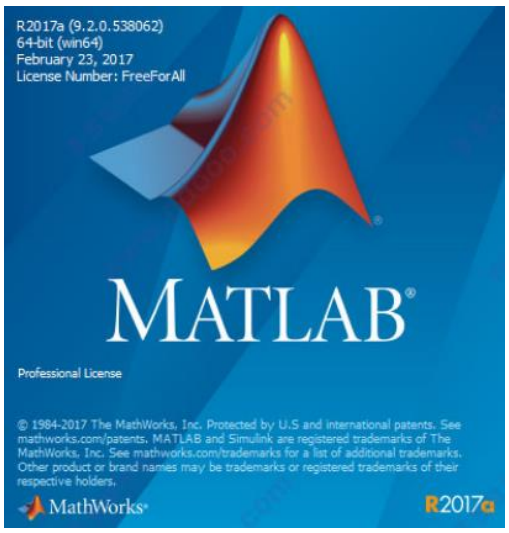


Figure 4-3 MATLAB 2017a

4.1.4 Arduino IDE

The Arduino IDE has been used for programming Arduino board, while doing feasibility study on Arduino implementation of EEG classifier.

Figure 4-4 Arduino 1.8.5

4.2 Problems and solutions

When designing and implementing this project, I had to face many problems. The main problem I have faces is the selecting the FPGA development board. I have created this project, did the designing, debugging and also the implementation for Basys 3 trainer board, because I was in the impression that it is available in the laboratory. However, after finalizing the project, I have got to know that only Basys 2 development board is available in the post graduate laboratory. Hence, I dad to implement the design again for that board. Also, I had to switch to Xilinx ISE software since Vivado do not have the support for Basys 2 development board**.** Compared to Vivado Design suite, ISE do not have support for SystemVerilog, hence I had to convert the design from SystemVerilog to Verilog. However, after implementing it on Basys 2, I have noticed that the resources available in that board is not enough for this project. Then I have implemented the design on Atlys Spartan 6 FPGA development board, however it is not a suitable for this project because of high cost compared to resource utilization of this project. Hence, I have decided to order a Basys3 Trainer board through Digilent web site in order to demonstrate the project.

Another problem that I have faced is representing DWT values in Verilog so that it can be loaded in to RAM. Since DWT values are in floating points, an appropriate representation system should be selected. For this, I have selected fixed point format considering several factors, as described in Section 2.5.3.

4.3 Recourse utilization in FPGA

Resource	Utilization	Available	Utilization...
LUT	19854	20800	95.45
FF	952	41600	2.29
BRAM	14.50	50	29.00
DSP	90	90	100.00
IO	27	106	25.47
BUFG	1	32	3.13

Figure 4-5 Post implementation resource utilization

One major objective of this project is to design a low-cost EEG classifier. For that it should be able to use FPGA with minimum resources. Post implementation resource utilization for this LDA classification is shown in the Figure 4-5 above. From that it can be observed that available Look Up Tables (LUTs), Flip Flops (FFs), DSPs, Input/Outputs (IOs) and BUFGs (Global Buffers) in FPGA is enough for the design. Here FFs indicates the amount of sequential element usage and DSPs indicates the usage of slices dedicated for DSP (Digital Signal Processing) functions. Hence this design can be implemented even with Basys 3 Artix-7 FPGA Trainer board. The resources utilization within the FPGA device is shown in Figure 4-6.

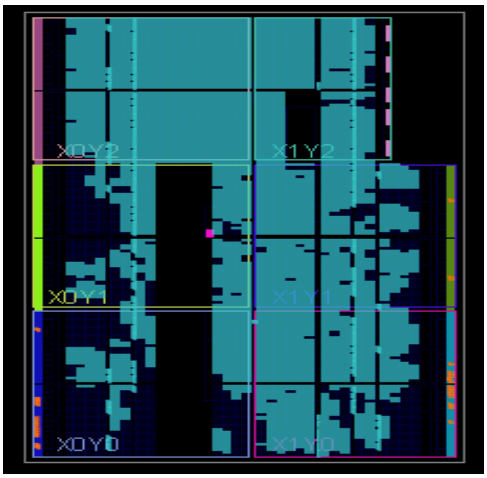


Figure 4-6 Resource utilization with in FPGA device

4.3.1 Issues due to resource utilization

The high resource utilization in FPGA chip might appear to be a practical issue in this implementation. As shown in Figure 4-5, this design utilizes 95.45% of available LUTs and 100% of DSP slices available in the FPGA chip. In an application like sleep detectors for automobile drivers, the FPGA chip will always do calculations continuously. Hence if the FPGA starts to heat due to the higher utilization resources when running for long hours, there might be a necessity to add cooler to the FPGA which will become an additional cost to the final product. However, this will not be an issue like in high processor usage in computers, since the operating frequency is low (12 MHz), compared to computer processor speed which are in Giga Hertz range.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

With the development of medical electronics and popularity of portable devices, there is a huge demand for portable medical devices these days. This research project also can be used for development of those kind of products.

Hardware implementation allows to implement EEG classification systems on portable devices without using software layer, also since this can be implement on FPGA with minimum resources, it is very useful for developing low cost device.

There are many accidents happens due to drivers falling sleep, especially in highways. If it is possible to detect if the is going to sleep, then alarm can be given to avoid the accident. This can be done analyzing the EEG signals. Here it should decide whether driver is going to sleep or not, therefore it is a two-class scenario, similar to this project. Hence sleep detectors for automobile drivers is an ideal use case for this project.

5.2 Future work

There are two areas that can be focused, as future work for this project. That is first integrating the feature extraction method in order to come up with a full system and secondly to enhance the EEG classification as adaptive classifier.

Here I have implemented only the EEG classification part in hardware, since DWT feature extraction is already implemented in hardware in previous projects. However, in order to design a product, it is necessary to integrate DWT feature extraction with classification system. In addition to that apart from LDA classification other classification systems like SVM (Support Vector Machine), KFD (Kernel Fisher Discriminant) can also be included in to the classification part of the design [2]. Then

the final decision that is taken considering several classification methods will be more accurate.

In this project, classification is done based only on the EEG signals provided at the time of testing. This can be further enhanced using adaptive LDA, to use previously provided EEG signal data also to consideration while doing the classification. Also, currently this design decides the class of EEG signal from previous EEG signals recorded for 12 hours. After implementing this in adaptive nature, it will be possible to decide the EEG class in real time, based on the signal receive.

It has been analyzed and proved that supervised adaption is the best option, when implementing adaptive LDA. [13] In order to implement that mean and the common variance used in this design should be replaced by following estimated adaptive mean and the estimated common variance matrix.

Estimated adaptive mean:

$$\mu_i(t) = (1 - UC) \cdot \mu_i(t-1) + UC \cdot x \tag{5-1}$$

with 'i' is the lass of x(t) and UC is the update coefficient

Estimated adaptive common variance matrix:

$$\Sigma(t)^{-1} = \frac{1}{(1-UC)} \cdot \left(\Sigma(t-1)^{-1} - \frac{1}{\frac{1-UC}{UC} + x(t)^T \cdot v(t)} \cdot v(t) \cdot v(t)^T \right) \tag{5-2}$$

BIBLIOGRAPHICAL REFERENCES

[1] L. P. Wijesinghe, D. S. Wickramasuriya, and A. A. Pasqual, "A generalized preprocessing and feature extraction platform for scalp EEG signals on FPGA," *IECBES 2014, Conf. Proc. - 2014 IEEE Conf. Biomed. Eng. Sci. "Miri, Where Eng. Med. Biol. Humanit. Meet,"* no. December, pp. 137–142, 2015.

[2] B. Wang, C. M. Wong, F. Wan, P. U. Mak, P. I. Mak, and M. I. Vai, "Comparison of different classification methods for EEG-based brain computer interfaces: A case study," *2009 IEEE Int. Conf. Inf. Autom. ICIA 2009*, pp. 1416–1421, 2009.

[3] S. Gupta and H. Saini, "EEG features extraction using PCA plus LDA approach based on L1-norm for motor imaginary classification," *2014 IEEE Int. Conf. Comput. Intell. Comput. Res. IEEE ICCIC 2014*, 2015.

[4] A. Bhardwaj, A. Gupta, P. Jain, A. Rani, and J. Yadav, "Classification of human emotions from EEG signals using SVM and LDA Classifiers," *2015 2nd Int. Conf. Signal Process. Integr. Networks*, pp. 180–185, 2015.

[5] A. L. Goldberger, "The Sleep-EDF Database," *physionet.org.* [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edfx/. [Accessed: Dec. 27, 2018].

[6] M. Vatankhah, M. R. Akbarzadeh-T., and A. Moghimi, "An intelligent system for diagnosing sleep stages using wavelet coefficients," *Proc. Int. Jt. Conf. Neural Networks*, pp. 0–4, 2010.

[7] I. Güler and E. D. Übeyli, "Adaptive neuro-fuzzy inference system for classification of EEG signals using wavelet coefficients," *J. Neurosci. Methods*, vol. 148, no. 2, pp. 113–121, 2005.

[8] W. S. Almuhammadi, K. A. I. Aboalayon, and M. Faezipour, "Efficient Obstructive Sleep Apnea Classification Based on EEG Signals," 2015.

[9] K Hong, "Data Compression Via Dimensionality Reduction - Linear Discriminant Analysis (LDA)," 2016. *bogotobogo,com* [Online]. Available: https://www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_Data_Compresssion_via_Dimensionality_Reduction_2_Linear_Discriminant_Analysis.php. [Accessed: Dec. 27, 2018].

[10] Z. Hussain and K. N. Parvin, “Q-Format Data Representation and Its Arithmetic,” vol. 7109, pp. 57–62, 2016.

[11] S. Sam, “Fixed Point Arithmetic Modules,” *opencores.org*, Jan 23, 2018.[Online]. Available: https://opencores.org/projects/fixed_point_arithmetic_parameterized. [Accessed: Dec. 27, 2018].

[12] C. Felton, “A Fixed-Point Introduction by Example,” *dsprelated.com*, April 25, 2011. [Online]. Available: https://www.dsprelated.com/showarticle/139.php. [Accessed: Dec. 27, 2018].

[13] C. Vidaurre, A. Schloegl, B. Blankertz, M. Kawanabe, and K.-R. Müller, “Unsupervised adaptation of the LDA classifier for Brain-Computer Interfaces.,” *Computer (Long. Beach. Calif).*, vol. 2008, no. 2, pp. 1–6, 2008.