DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

FACE AND SPEECH RECOGNITION ON FIELD PROGRAMMABLE GATE ARRAY

by
Gökhan ÇETİN

October, 2010
İZMİR

FACE AND SPEECH RECOGNITION ON FIELD PROGRAMMABLE GATE ARRAY

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical and Electronics Engineering

by
Gökhan ÇETİN

October, 2010
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"FACE AND SPEECH RECOGNITION ON FIELD PROGRAMMABLE GATE ARRAY"** completed by **GÖKHAN ÇETİN** under supervision of **ASST. PROF. DR NALAN ERDAŞ ÖZKURT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis fort he degree of Master of Science.

Asst. Prof. Dr. Nalan Erdaş ÖZKURT

Supervisor

(Jury Member)

(Jury Member)

Prof.Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank my advisor Asst. Prof. Dr. Nalan Erdaş ÖZKURT who always supported and encouraged me in all steps of the thesis. I must express that I always felt her support from initiating to completing the thesis.

I also would like to thank to my dear friend Enes DİLCAN for his absolute guidance, supports and patience. His endless friendship and encouragement always motivated me.

I would like thank to my family, my mother Cemile, my father Mehmet, and my brothers, Okan and Hakan for their never ending support and motivation.

Gökhan ÇETİN

FACE AND SPEECH RECOGNITION ON FIELD PROGRAMMABLE GATE ARRAY

ABSTRACT

Biometric recognition can be defined as automatic recognition of person based on one or more personal traits. Face and speech recognition occupies an important area in biometric recognition techniques.

In this thesis, Principal Component Analysis (PCA) and Fourier Transform Analysis are used as feature extraction methods for face and speech recognition, respectively. Field Programmable Gate Arrays (FPGA) are integrated circuits that can be programmable in the field by the customer after manufacturing. In this study, we aimed to develop a system by using FPGA to recognize the people based on face and speech. For this purpose, face and speech data collected by MATLAB, was sent to FPGA to be processed in feature extraction algorithms. Hence, a database was constructed and loaded to the memory of FPGA. In recognition phase, a face image and recorded speech were processed in FPGA and compared to database to find the possible owner of the feature. All feature extraction algorithms were developed in FPGA. This thesis also presents the development steps of a multibiometric recognition that combines face and speech recognition systems. Multibiometric recognition system makes a fusion process at the decision levels of face and speech recognition systems.

Development of data processing techniques in FPGA is the main subject of this thesis. FPGAs have many advantages that can be used in biometric recognition studies. Therefore, the system can be improved more to use in the areas like security by some enhancements which will be done in the future.

Keywords: Face recognition, speech recognition, multibiometric recognition, FPGA, PCA, Fourier Transform Analysis

SAHADA PROGRAMLANABİLİR KAPI DİZİLERİNDE YÜZ VE KONUŞMA TANIMLAMA

ÖZ

Biyometrik tanıma, bir veya daha fazla kişisel özelliğe bağlı olarak kişinin otomatik tanınmasıdır. Yüz ve konuşma tanıma, biometrik tanıma teknikleri içinde önemli bir alan kaplar. Bu tezde, yüz ve konuşma tanıma için sırayla Temel Bileşen Analizi ve Fourier Dönüşümü, öznitelik çıkarma yöntemi olarak kullanılmıştır.

Programlanabilir Kapı Dizileri, ürettikten sonra, müşteri tarafından sahada programlanabilir tümleşik devreleridir. Bu çalışmada, yüz ve konuşma tanıma yöntemlerine bağlı olarak insanları tanıyabilmek için, Sahada Programlanabilir Kapı Dizilerini kullanarak bir sistem geliştirme amaçlanmıştır. Matlab tarafından toplanan yüz ve konuşma verileri, öznitelik çıkarma algoritmalarında kullanılmak üzere, Sahada Programlanabilir Kapı Dizilerine gönderilmiştir. Böylece, bir veri tabanı oluşturuldu ve Sahada Programlanabilir Kapı Dizilerinin belleğine yüklenmiştir. Tanıma aşamasında, bir yüz resmi ve kaydedilen konuşma işareti, Sahada Programlanabilir Kapı Dizilerinde işlenmiş ve ait olduğu olası kişinin bulunabilmesi için veri tabanıyla karşılaştırılmıştır. Bu çalışma, yüz ve konuşma tanıma sistemlerini birleştiren bir çoklu tanıma sisteminin gelişim aşamalarını tanıtır. Çoklu tanıma sistemi, yüz ve konuşma tanıma sistemlerinin karar verme aşamalarında birleştirme işlemi gerçekleştirir.

Sahada Programlanabilir Kapı Dizilerinde veri işleme teknikleri geliştirmek, bu tezin ana amacıdır. Sahada Programlanabilir Kapı Dizileri, biyometrik tanıma çalışmalarında kullanılabilecek çok sayıda avantaja sahiptir. Bu nedenle, sistem, gelecekte yapılacak geliştirmelerle, güvenlik gibi alanlarda kullanmak üzere daha da iyileştirilebilir.

Anahtar Sözcükler: Yüz tanıma, konuşma tanıma, çoklu biometrik tanıma, Sahada Programlanabilir Kapı Dizileri, Temel Bileşen Analizi, Fourier Dönüşümü

CONTENTS

Page

CHAPTER ONE

INTRODUCTION

1.1 General Overview to Biometric Systems

The need to identify the users and customers is today increasing. It results in a growing number of PIN-codes, cards codes etc. which are hard to remember. A simpler solution can be to construct user recognition systems based on the individual's biometric features, since they are specific to people and can not be forgotten. These systems are called as the biometric verification systems, based on methods for uniquely recognizing humans based on one or more basic physical or behavioral traits.. There are several forms of biometric identification employed in access control: fingerprint, hand geometry, iris and face recognition. The use of biometric technology significantly increases security level of systems. Because it eliminates such problems as lost, stolen or loaned ID cards, and forgotten or guessed PINs. It is also used to identify individuals in groups that are under surveillance. Under the light of these usage areas, banks and security units are potential marketing fields. Biometric characteristics can be divided into two main classes:

- Physiological traits that are related to the shape of the body: fingerprint, face recognition, DNA, hand and palm geometry, iris recognition

- Behavioral traits that are related to the behavior of a person: typing rhythm, gait, and voice.

Biometric face recognition is one of the most popular techniques and widely used on last years. Basically, it works on analyze a subject's facial structure with computer programming methods. Face recognition software takes a number of points and measurements including the distances between key characteristics such as eyes, nose and mouth, angles of key features such as the jaw and forehead, and lengths of various portions of the face or the face image is transformed into another domain to

extract some features. Using all of this information, it creates a unique template including all of the numerical data. This template may then be compared to enormous databases of facial images to identify the subject. In future years, with the improvements of technology, it can be a popular method, especially for security and banking operations. Biometric face recognition can be categorized into:

- *face identification*, which is based on identifying a person in a face database by using facial features.

- *face* ver*ification*, which is based on automatically determining if a person really is the person he/she claims to be.

Figure 1.1 shows the abstraction of an automatic face identification system

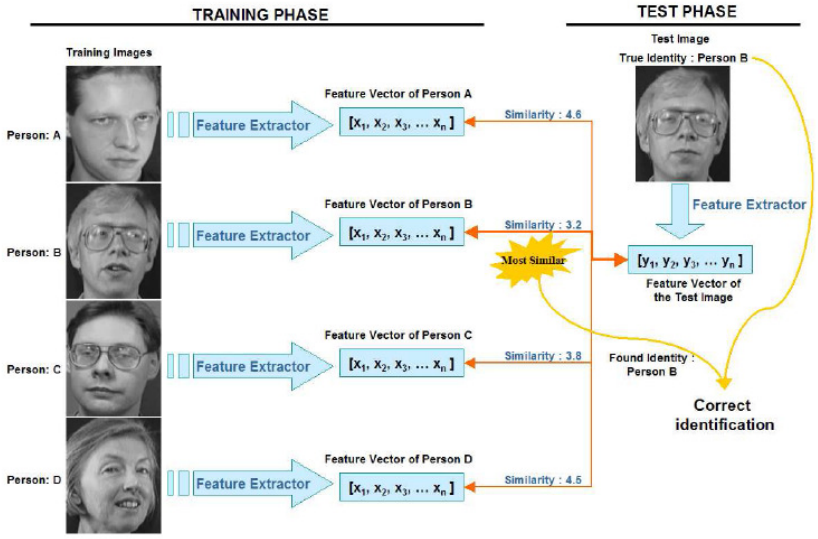


Figure 1.1 Working principle of face recognition system

Speaker recognition is another popular and interesting research field for the last decades, which still yields a number of unsolved problems similar to face recognition. Speaker recognition is basically divided into speaker verification and speaker identification.

- *speaker verification* is the task of automatically searching if a person really is the person he/she claims to be.

- *speaker identification* is the task of searching whom a speech belongs to.

This technology can be used as a biometric feature for verifying the identifying of a person in applications like banking by telephone and voice mail. The speech recognition systems can be categorized into text-dependent and text-independent methods. Text-dependent systems require the speaker to utter a specific phrase. On the other hand, text-independent methods catch the characteristics of the speech irrespective of the text spoken. Hence there are some difficulties hard to resolve for text-independent methods.

Figure 1.2 shows the abstraction of an automatic speaker recognition system. Regardless of the type of the task (identification or verification), system operates in two modes: training and recognition modes. In the training mode, a new speaker (with known identity) is enrolled into the system's database. In the recognition mode, an unknown speaker gives a speech input and the system makes a decision about the speaker's identity. If the type of task is identification, the system searches whom the speech belongs to. If the type of task is verification, the system searches if the speaker is really is the person he/she claims to be.

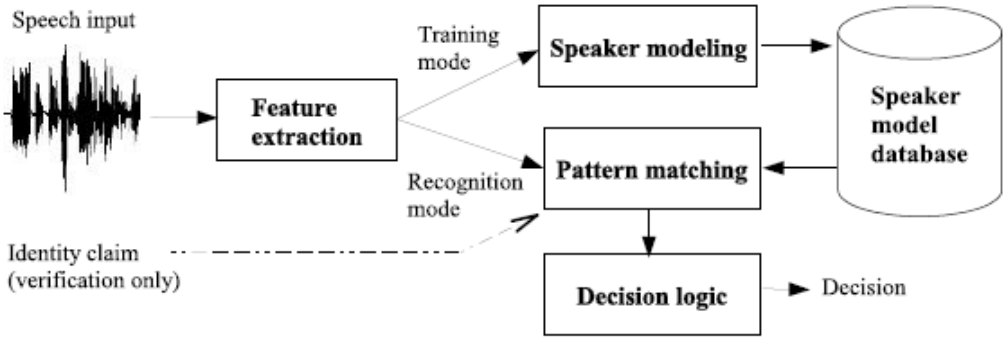


Figure 1.2 Working principle of speech recognition system (Delioğlu, 2007)

1.2 General Overview to Multibiometric Systems

Most biometric systems deployed in real-world applications based on unimodal recognition. They work by processing the data set of a single source of information as single fingerprint, face etc. These systems have to contend with a variety of problems such as:

- *Noise in sensed data* : A fingerprint image with a scar or a voice sample altered by cold are examples of noisy data.

- *Intra-class invariations* : These variations are typically caused by a user who incorrectly interacting with the sensor.

- *Inter-class similarities* : In a biometric system comprising of a large number of users, there may be inter-class similarities in the feature space of multiple users.

- *Non-universality* : The biometric may not be able to acquire meaningful biometric data from a subset of users.

- *Spoof attacks* : This type of attack is especially relevant when behavioral traits such as signature or voice are used (Ross, & Jain, 2004).

Some of these limitations can be resolved by deploying multimodal biometric systems that integrate the evidence presented by multiple sources of information. Such systems, known as multimodal biometric systems, are expected to be more reliable due to presence of multiple, independent pieces of evidence. These systems are able to meet the stringent performance requirements imposed by various applications (Ross, & Jain, 2004).

In a multimodal biometric system, information merging can occur in scenariaos of fusion at the data or feature level, fusion at the match score level or fusion at the decision level (Ross, & Jain, 2004).

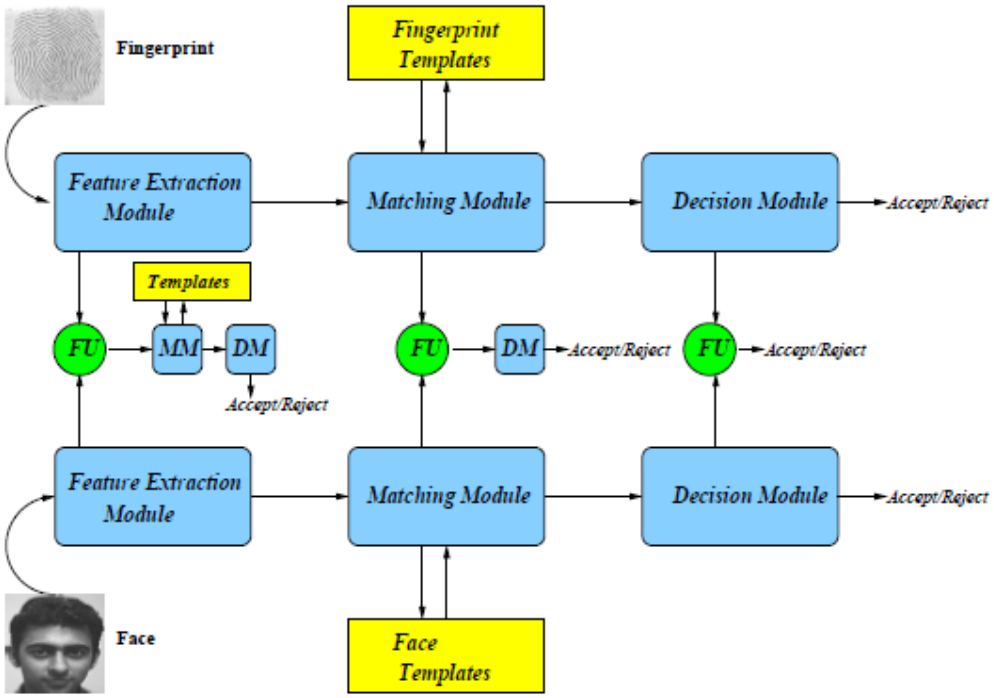


Figure 1.3 Level of fusion in multimodal biometric system where FU:Fusion Module, MM:Matching Module, DM:Decision Module (Ross, & Jain, 2004)

General expectation about multimodal biometric systems is to achieve more effective performance when the information integration is applied at an early stage. Since the feature set contains richer information about the input biometric, fusion at feature level is expected to give better recognition results. However, there are some difficult points of fusion at feature level.

- the feature sets of the various modalities may not be compatible since the feature sets are generated due to distinct feature extraction techniques.

- most commercial biometric systems don't give acces to the feature sets which are used in products.

Fusion at the decision level is hard to deploy since the available information is limited. Thus, fusion at the match score level is usually preferred since it is easier than other scenarios and combines the score presented by the different modalities (Ross, & Jain, 2004).

1.3 History of Face Recognition Systems

During 1964 and 1965, Bledsoe worked on using the computer to recognize human faces with Helen Chan and Charles Bisson (Bledsoe & Chan 1965). He was proud of this work and purposed to documentate them. But, unfortunately, a litle part could be published. Because the funding was provided by an unnamed intelligence agency that did not allow much publicity. He searched the solution of same face recognition question. How could a face image be matched to correct image among a large database of images (in effect, a book of mug shots)? The success of the method could be measured in terms of the ratio of the answer list to the number of records in the database.

This project was labeled as “man-machine”. Because the human extracted the coordinates of a set of features from the photographs, which were then used by the computer for recognition. Using a graphics tablet (GRAFACON or RAND TABLET), the operator would extract the coordinates of features such as the center of pupils, the inside corner of eyes, the outside corner of eyes, point of widows peak, and so on (Ballantyne, Boyer & Hines, 1996). A list of 20 distances, such as width of mouth and width of eyes, pupil to pupil etc. were computed with the help of these coordinates. These operators could process about 40 pictures in an hour. After building the database, the name of the subject in the photograph was associated with the list of computed distances and achieved data was loaded to the computer to store. In the recognition phase, the set of distances was compared with the corresponding distance for each photograph. The closest records are returned.

This brief description is an oversimplification that fails in general. Because, two pictures could match in head rotation, lean, tilt, and scale (distance from the camera). These would be unsufficient to make a succesful recognition. Thus, each set of distances was normalized to represent the face in a frontal orientation. To accomplish this normalization, the program was trying to determine the tilt, the lean, and the rotation. Then, using these angles, the computer undid the effect of these transformations on the computed distances. To compute these angles, the computer

was required to know the three-dimensional geometry of the head. Because the actual heads were unavailable, Bledsoe used a standard head derived from measurements on seven heads in 1964.

After Bledsoe left PRI in 1966, this work was continued at the Stanford Research Institute, by a team in the head of Peter Hart. In experiments, which were performed on a database of over 2000 photographs, the computer consistently outperformed humans when presented with the same recognition tasks. Peter Hart (1996) enthusiastically recalled the project with the exclamation, "It really worked!" (Thorat, Nayak & Dandale, 2010).

By about 1997, the system developed by Christoph von der Malsburg and graduate students of the University of Bochum in Germany and the University of Southern California in the United States outperformed most systems with those of Massachusetts Institute of Technology and the University of Maryland rated next. The Bochum system was developed by the economical support of the United States Army Research Laboratory. The software was sold as ZN-Face and used by customers such as Deutsche Bank and operators of airports and other busy locations. The software was "robust enough to make identifications from less-than-perfect face views. It can also often see through such impediments to identification as mustaches, beards, changed hair styles and glasses—even sunglasses" (Thorat, Nayak & Dandale, 2010).

In about January 2007, the interest of image researches weas the text surrounding a photo. Polar Rose technology could guess from a photograph, in about 1.5 seconds, what any individual may look like in three dimensions. Researchers thought that they "will ask users to input the names of people they recognize in photos online" to help build a large database.

In 2006, the performance of the latest face recognition algorithms were evaluated in the Face Recognition Grand Challenge (FRGC). High-resolution face images, 3-D face scans, and iris images were used in the tests. The results indicated that the new

algorithms are 10 times more accurate than the face recognition algorithms of 2002 and 100 times more accurate than those of 1995. Some of the algorithms were able to outperform human participants in recognizing faces and could uniquely identify identical twins which was really hard to do (Thorat, Nayak & Dandale, 2010).

In 2010, a hardware accelarator for full-search vector quantization has been developed for face recognition by Diem Tran, Thi To, Thuan Huynh, Phuong Nguyen from Faculty of Electronics and Telecommunication in Vietnamme. In the system, the number of elements for each codeword and the number of codewords in the system could be changed easily for different applications. Also, in last years, some multibiometric systems were developed. Gunawan Sugiarta YB., Riyanto Bambang, Hendrawan, and Suhardi from Bandung, Indonesia developed a person identification system which combined multiple biometrics. A feature level fusion of dual tree complex wavelet transform speech and face image features was developed. (Tran, To, Huynh & Nguyen, 2010)

In 2010, a group of researchers developed a secure face recognition system. They represented an accurate face box detection was accomplished by skin color detection followed by NCC and between the two stages light normalization was performed. The system offered correct recognition rates of % 96 (Abdel-Ghaffar, Alam, Mansour & Alsoud, 2010). Future works usually focus on using multiple biometrics for personal authentication.

1.4 History of Speech Recognition Systems

Speech is the primary step of communication between people. For many reasons, research in automatic speech recognition has become an attractive research area over the past five decades.

The first research for speech recognition goes back more than one hundred years. By way of example, in 1881 Alexander Graham Bell, his cousin Chichester Bell and Charles Sumner Tainter invented a recording device that used a rotating cylinder.

The cylinder had a wax coating on which up-and-down grooves could be cut by a stylus which responded to incoming sound pressure. Based on these researches, Bell and Tainter formed the Volta Graphophone Co. in 1888 in order to manufacture machines for the recording and reproduction of sound in office environments. In these years, Thomas Edison invented “Ediphone”. It was a phonograph which was developed on a tinfoil based cylinder. Thus, Thomas Edison could compete Columbia Co. which took over the rights of Volta Graphophone Co. later. The main purpose of these products was to record dictation of letters and notes (Juang & Rabiner, 2004).

Main purpose of these products in these years was to record dictation of notes for secretaries to type them later easily. A similar technology became popular in the 1990’s in the area of “call centers.” By the way of an example, a service was the AT&T Operator line which helped a caller place calls, arrange payment methods, and conduct credit card transactions. By using these products, a call center could provide the capability of handling several thousands of calls while reducing the large operating costs of the center. Automatic speech recognition technologies provided the capability of automating these call handling functions, thereby reducing the large operating cost of a call center. By way of example, the AT&T Voice Recognition Call Processing (VRCP) service, which was introduced into the AT&T Network in the 1992’s, handled about 1.2 billion voice transactions each year using automatic speech recognition technology (Juang & Rabiner, 2004).

Speech recognition technology has been distinguished by a general public since it was used in some blockbuster movies of the 1960’s and 1970’s. The most popular movies used this technology was Stanley Kubrick’s acclaimed movie “2001: A Space Odyssey”. In this movie, an intelligent computer named “HAL” spoke in a natural sounding voice. Also, it was able to understand what the people spoke and respond accordingly. This strange behaviour of HAL made the general public aware of the potential of intelligent machines. The other popular example was the famous Star Wars which the droids like R2D2 and C3PO were able to speak, recognize and understand fluent speech. Also these droids could interact with their environment,

easily. The most recently research was announced in 1988 and in 2011 by Apple. The company created a vision of speech technology and computers, titled "Knowledge Navigator". The company defined the concepts of a Speech User Interface (SUI) and a Multimodal User Interface (MUI) along with the theme of intelligent voice-enabled agents (Juang & Rabiner, 2004). The main focus point of today speech technologies is to enable machines to understand and respond correctly. Although the public is still far from having these machines, many technological advances bring us closer to the "Holy Grail" of machines that recognize and understand fluently spoken speech.

1.5 Biometric Recognition Algorithms

Principal Component Analysis (PCA) is one of the most successful and attractive techniques which have been used in biometric recognition like face and speech recognition. PCA is basically a statistical method based on the broad title of factor analysis. The main purpose of PCA is to reduce the large dimensionality of the data space to the smaller intrinsic dimensionality of feature space (independent variables), which is easier to analysis. In the new feature space, the data is described economically since there is a strong correlation between observed variables.

The only job which PCA can do, isn't dimension reduction. The other jobs done by PCA are prediction, redundancy removal, feature extraction, data compression, etc. The main areas of PCA are linear domain and applications which have linear models such as signal processing, image processing, system and control theory, communications etc.

Independent component analysis (ICA) is also a feature extraction technique which has been basically developed to solve blind signal separation. Hence it is a technique for extracting statistically independent and significant variables from a mixture of data. On the other hand, it may be successfully applied for biometric recognition problem, especially to the face recognition problem. ICA has been used

for many different problems such as MEG and EEG data analysis, finding hidden factors in financial data etc.

The ICA technique tries to obtain a linear transform for the mixture of data as statistically independent as possible. Hence, ICA has similarities with PCA. It is called as the generalization of PCA. ICA returns a representation based on statistically independent variables while PCA aims to create a representation of the inputs based on uncorrelated variables. Hence, in order to resolve the face recognition problem, the basis images obtained with ICA are more local than the images processed with PCA.

Figure 1.4 Some original (left), PCA (center) and ICA (right) basis images (Deniz, Castrillion & Hernandez, 1999)

Linear Discriminant Analysis (LDA) is a technique for classification of a set of data into predefined classes. The purpose of LDA is dimension reduction and data classification. These characteristics of LDA make it attractive for biometric recognition problems. Especially, it is applied to data classification issue in speech recognition problem. LDA easily handles the case where the with-in class frequencies are unequal and their performance has been examined on randomly generated test data. It also gives an idea to user about the distribution of the feature data.

Gabor filter banks are reasonable models of biometric recognition and are one of the most successful techniques for processing images of the human face and speech.

In face recognition issues, Gabor filters are designed to be used for edge detection. Basically, frequency and orientation variations of Gabor filter are evaluated to be similar to those of human visual system.

The recognition rates of Gabor filters are parallel to the success of band pass filter banks. While the optimal filter bank characteristics have been extensively developed in the speech recognition problems, a little work has been done to systematically understand which orientation and frequency bands are optimal for face processing applications. But, the dimensionality of the input pattern increases based on the number of filters in the filter bank. Hence, the dimensionality of the face pattern is reduced by using some techniques like PCA, subspace projection and downsampling techniques.

The Neural Network approach applies biological concepts to machines to overcome recognition problems. A neural network is an information processing system. It consists of many units, called as neurons, which has a high degree of interconnection between each other. The characteristics of neural networks may simulate some functionality of biological brains and neural systems. The main advantages of neural networks are self-organization, adaptive-learning and fault-tolerance capabilities. Due to these characteristics, neural networks are used in many biometric recognition problems. Detailed information about biometric recognition is explained in Section 2.1.3.

1.6 Aim of Thesis

The purpose of thesis is to examine feature extraction and classification methods for face and speech recognition systems, mainly implemented in a Field Programmable Gate Array (FPGA). The project aims to collect the data into the digital environment and to develop a system for the face and speech data to be analyzed succesfully. Hence, data processed in FPGA is obtained with different feature extraction methods and pre-processed to increase the recognition rates. Face and speech recognition systems process features individually to observe the general

performance of each single biometric system. Later, a multiple biometric system implemented to use the advantages of both biometric systems instead of developing a single biometric system. Since FPGA technology is new and good for developing new systems, an effective recognition system in FPGA for obtaining accurate recognition rates have been aimed in this project.

1.7 Overview of Whole Project

This thesis proposes a system to acquire face appereance of any user and process it to understand if he/she is one of people in the training set. The system starts to procedure by taking a photo of tester. This image is read and resized for more effective analysis by MATLAB. Then, it is sent to FPGA-board by serial communication, RS-232 protocol. After that, feature extraction part starts by using PCA which is one of basis feature extraction methods. Extracted feature elements are compared to feature matrix which is extracted and combined from people in the database. In the first step of the recognition, the features, extracted from people in the database by applying PCA are saved to flash memory in the board. The purpose of the system is to find if the test face image belongs to any of the subjects in the database. As soon as the comparison result is evaluated in FPGA, it is presented to the end user.

The second part of the system aims to find the owner of any test speech by analysing the speech data which is already acquired in speech database and test speech data. Similar to the face recognition system, the system starts by collecting speech samples to construct a database. In the run-time of the system, a test speech is collected by MATLAB and is preprocessed before transferring to FPGA. Unsignificant part and noisy part of speech data is eliminated using end-point and noise reduction techniques in MATLAB. Then, significant data block is sent to FPGA over RS-232 port. In FPGA, Fourier Transform is applied to speech data. Transform result of speech samples in database and tester are compared to find the minimum distance. The owner of test speech data is detected and shown to the end user.

A multibiometric recognition system is generated after the implementation of face and speech recognition systems seperately. A matrix is created which includes the integrated form of face and speech database elements in MATLAB. Then, it is transmitted to FPGA by serial port communication. The new generated implementation in FPGA separates the incoming face and speech database elements and processes them seperately. In the recognition phase, test face image and test speech data are acquired and combined. In FPGA, this matrix is decomposed into face and speech data again. Individual recognition systems produce a result for face recognition system and a result for speech recognition system. The results of each recognition are combined due to fusion at decision level scenario. The overall result is true if and only if both recognition systems end up correctly.

1.8 Outline of the Thesis

Chapter 2 includes information about face recognition system and feature extraction method and Principle Component Analysis (PCA). Chapter 3 contains of information about speech recognition system and Fourier Transform Analysis. Chapter 4 defines what Field Programmable Gate Array (FPGA) is and how it works briefly. The devices which are used throughout project are introduced in that chapter. Chapter 5 defines the system and explains briefly Face Recognition System by using Field Programmable Gate Array. The experimental results are presented in Chapter 5. Chapter 6 describes how the multibiometric recognition system is implemented and the general performance of multibiometric recognition system ends up after deploying several tests. The last chapter of the thesis includes conclusions, advantages and disadvantages of the system and future works. The algorithm of whole system is in Appendix part of the thesis.

CHAPTER TWO

FACE RECOGNITION & PRINCIPAL COMPONENT ANALYSIS

2.1 Human Face Recognition

2.1.1 What is Face Recognition?

Face recognition is a task that humans perform routinely in their daily lives. Wide availability of powerful and low-cost desktop and embedded computing systems has created an big interest in automatic processing of digital images and videos. This interest has brought many new solutions in a number of applications, including biometric authentication, surveillance, human-computer interaction, and multimedia management. Research and development in automatic face recognition becomes more popular in last years.

Research in face recognition is motivated not only by the fundamental challenges. Many researchers recognized that there are many practical applications which human identification is needed. Face recognition, as one of the primary biometric technologies, became more important based on the rapid advances in technologies such as digital cameras, the internet and mobile devices and more technological demands on security. Face recognition has several advantages over other biometric technologies. It is natural and easy to use. Also, noise factor can eliminated easily by using digital image processing techniques. It is open to new technologies and a safety method for security applications.

A face recognition system is expected to identify faces presented in images and videos automatically. Face recognition can be operated in either or both of two modes: face verification and face identification. Face identification is based on comparison of a test face image against all the template images in the database to determine the identity of the query face (1:N match type). Face verification is based on comparison of a test face image against a template face image whose identity is

being known already (1:1 match type). It aims to autmatically determine if a person really is the preson he/she calims to be. Although progress in face recognition has been encouraging, the task has also many parameters that makes it difficult to analyse, especially for unconstrained tasks where viewpoint, illumination, expression, occlusion, accessories, and so on vary considerably.

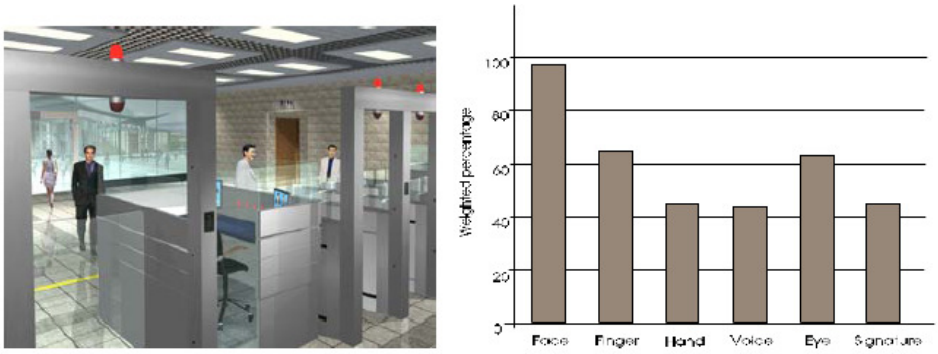


Figure 2.1 A scenario of using biometric systems for passport control and a comparison of various biometric features based on compatibility (Li & Jain, 2004)

2.1.2 Face Recognition Processing

Face recognition is a visual pattern recognition problem. There, a face as a three-dimensional object subject to varying illumination, pose, expression and so on is to be identified based on its two-dimensional image. A face recognition system generally consists of four main parts as shown in Figure 2.2: detection, alignment, feature extraction, and matching, where localization and normalization (face detection and alignment) are processing steps before face recognition (facial feature extraction and matching) is performed.

Face detection segments the face areas from the background. In the case of video, the detected faces may need to be tracked using a *face tracking* component. *Face alignment* is aimed at achieving more accurate localization. Normalizing faces is a part of face alignment. It provides estimates of the location and scale of each detected face to face extraction and face matching modules. Facial components, such as eyes, nose, mouth and facial outline, are located. Based on the location points, the

input face image is normalized with respect to geometrical properties, such as size and pose, using geometrical transforms or morphing.

After a face is normalized geometrically and photometrically, *feature extraction* is performed to provide effective information that will be used for distinguishing between faces of different people. Feature extraction must be stable with respect to the geometrical and photometrical variations to achieve more accurate results. For *face matching*, the extracted feature vector of the input face is matched against those of enrolled faces in the database. It concludes the identity of the face when a match is found with sufficient confidence or indicates an unknown face otherwise (Li & Jain, 2004).

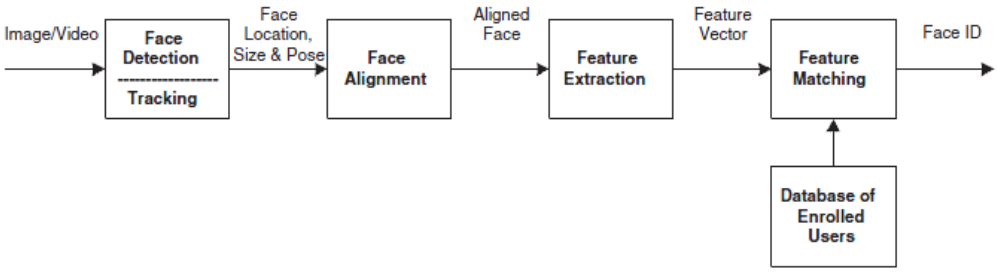


Figure 2.2 Face recongition processing flow (Li & Jain, 2004)

Face recognition depends highly on features that are extracted to represent the face pattern and classification methods, used to distinguish between faces. At that point, face localization and normalization are the basis for extracting effective features to achieve more accurate recognition. These problems may be analyzed from the viewpoint of face subspaces or manifolds.

2.1.3 Face Recognition Algorithms

In this section, popular algorithms used in face recognition algorithms, will be described definitely. Main mission of them is to extract the available features from a facial image for recognize procedure.

2.1.3.1 Principal Component Analysis

Principal Component Analysis (PCA) is a technique developed by Turk and Pentland. It is widely used algorithm on account of high recognition rate. The main requirement of this algorithm is the fact that variations such position and illumination are limited. PCA can handle them, but maximum performance is achieved if these variations are eliminated (Madi, Lahoud & Sawan, 2006).

Face recognition consists of two steps whenever PCA is recognition technique. They are training phase and recognition phase (or test phase). During training phase, feature matrix is constructed that includes each face image as a column vector, with each element of the vector corresponds to an image pixel. Then, average face image is evaluated and all image vectors are normalized due to average face. PCA algorithm calculates the eigenvectors of covariance matrix. The eigenvector matrix is multipled by each face vector to project all face images to a face space. During recognition phase, similar calculations are evaluated to a test face image. It is normalized with respect to the average face. Next, the subject face is projected to face space. The minimum distance is found between the subject face and all known face projections, done in training phase. Mathematical derivation and detailed information about PCA is given in Section 2.2.

2.1.3.2 Independent Component Analysis

Independent Component Analysis (ICA) is other popular technique for face recognition. Basically, the difference of ICA than PCA is to use higher-order relationships between pixels, while PCA works on second-order relationships. However, ICA use higher-order relationships between the pixels and is able to use the phase spectrum (Madi, Lahoud & Sawan, 2006) .

ICA algorithm relies on the infomax algorithm. It receives an n-dimensional random vector as input. PCA algorithm is used to reduce the size of random vector. The higher order relationships aren't affected from dimensional reduction. Then,

ICA algorithm finds the covariance matrix of the result and its factorized form is obtained. Then, some defined methods are performed to obtain the independent components that each face images in face space includes. These methods are whitening, rotation and normalization (Hyvarinen, 1999).

Some definitions should be defıned to work with ICA. Assme that $y_1, y_2, \ldots, y_m$ are some random variables with joınt density $f(y_1, y_2, \ldots, y_m)$ and the variables are zero-mean. The variables y_i are independent, if the density function can be defined as:

$$f(y_1, y_2, \ldots, y_m) = f_1(y_1) f_2(y_2) \ldots f_m(y_m) \tag{2.1}$$

where $f_i(y_i)$ is the margnal denstiy of y_i. This form of indepedence is someimes called as statistical independence. Other basic definiton, related to ICA is uncorrelatedness. It means that:

$$E\{ y_i y_j \} - E\{ y_i \} E\{ y_j \} = 0, \text{ for } i \neq j \tag{2.2}$$

If the y_i is independent, the following equation is satisfied clearly:

$$E\{ g_1(y_i) g_2(y_j) \} - E\{ g_1(y_i) \} E\{ g_2(y_j) \} = 0, \text{ for } I \neq j \tag{2.3}$$

for any functions of $g1$ and $g2$. There is only a a special case, that independence and uncorrelatedness are equivalent. The special case occurs when $y_1, y_2, \ldots, y_m$ have joint Gaussian distributions. Otherwise, ICA isn't possible for Gauussian variables.

The problem of ICA can be defined by different basic definitions. Assume m-dimensional random vector that is denoted by $x = (x_1,\ldots,x_m)^T$ (Hyvarinen, 1999)

Definition 1 : ICA of the random vector x consists of finding a linear transform s = Wx so that the components s_i are as independent as possible, in the sense of maximizing some function $F(s_1, \ldots, s_m)$ that measures independence.

This definition doesn't consists of any assumptions. Hence it is the most general. But, it has a vague such that no definition the measure of independence of s_i. A different approach takes care of theoretical esimations:

Definition 2 : (Noisy ICA Model) ICA of a random vector x consists of estimating the following generative model for the data:

$$x = As + n \tag{2.4}$$

where the latent variables (components) s_i in the vector s = $(s_1,\ldots,s_n)^T$ are assumed independent. The matrix A is a constant m x n 'mixing' matrix, and n is a m – dimensional

.

Definition 3 : (Noise-free ICA model) ICA of a random vector x consists of estimating the following generative model for the data:

$$x = As \tag{2.5}$$

where A and s are as defined in Definition 2. As seen in the definiton, the noise factor is eliminated here (Hyvarinen, 1999).

2.1.3.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is another popular algorithm for data classification and dimension reduction. It works by guaranteening maximum ratio of between-class variance and within-class variance in a data set. LDA works on differences within an individual and those among all individuals. It provides seperability to decide class regions in the best way whereas PCA changes the shape of the original data set by trasforming to a different space (Balakrishnama & Ganapathiraju, 1999).

Mathematical operations of LDA starts with formulation of the data sets and the test sets, which are to be classified in the original space. Data sets can be represented as a matrix in the form given below:

$$set1 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \dots & \dots \\ \dots & \dots \\ a_{m1} & a_{m2} \end{bmatrix} \qquad set2 = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ \dots & \dots \\ \dots & \dots \\ b_{m1} & b_{m2} \end{bmatrix}$$

The mean of each data set are computed . Assume that μ_1 and μ_2 are the mean of set1 and set2 and μ_3 is the mean of the entire data, which is calculated as:

$$\mu_3 = p_1 \, x \, \mu_1 + p_2 \, x \, \mu_2 \tag{2.6}$$

where p_1 and p_2 are the priori probabilities of the classes set1 and set 2 respectively. Next, between-class and within-class scatters are evaluated as:

$$S_w = \Sigma \, p_j \, x \, (cov_j) \tag{2.7}$$

where cov_j is the covariance of each data set $_j$. Covariance matrix is computed using the following equation.

$$cov_{\,j} = (x_j - \mu_j) \, (x_j - \mu_j)^T \tag{2.8}$$

The between-class scatter is computed by the following equation.

$$S_b = \Sigma \, (\mu_j - \mu_3) \, x \, (\mu_j - \mu_3)^T \tag{2.9}$$

As shown in the above equation, S_b is like the covariance of the data set whose members are the mean vectors of each class. The prime mission of LDA is to maximize the ratios of between-class scatter to the within-class scatter. The

maximization factors are changable based on the approach. If LDA is a class-dependent approach which involves maximizing the ratio of between-class variance to within-class variance, the optimizing factors are computed as:

$$criterion_j = inv\ (cov_j)\ x\ S_b \tag{2.10}$$

If LDA is a class-independent approach which involves maximizing the ratio of overall variance to within-class variance, the optimizing factors are computed as:

$$criterion = inv\ (S_w)\ x\ S_b \tag{2.11}$$

By definition, an eigen vector of a transformation represents a 1-D invariant subspace of the vector space in which the transformation is applied. A set of these eigen vectors whose corresponding eigen values are non-zero, are all linearly independent and are invariant under the transformation. Thus, any vector space can be represented in terms of linear combinations of the eigen vectors. A linear dependency between features is indicated by a zero eigen value. To obtain a non-redundant set of features, all eigen vectors corresponding to non-zero eigen values only are considered and the ones corresponding to zero eigen values are neglected.

The data sets are transformed using the sinlg LDA transform or the class specific transforms after calculating the transformation matrices. Assume that A_j is the transformed set of j^{th}, B is the transform of j^{th} set, *data* defines the whole data set and set_j defines the j^{th} set. A decision region is specified in the transformed space. It is a solid line which seperates the transformed data sets for the class-dependent approach:

$$A_j = B_j^T\ x\ set_j \tag{2.12}$$

and for the class-independet approach:

$$A_j = transform\_spec^T\ x\ data^T \tag{2.13}$$

Test vectors are transformed similarly and are classified due to euclidean distance of the test vectors from each class mean. Smallest Euclidean distance is computed using the following equation where μ_{trans} is the mean of the transformed data set, n is the class index and x is the test vector.

$$dist\_n = (transform\_n\_spec)^T \; x \; x - \mu_{trans} \tag{2.14}$$

An euclidean distance is calculated for each test point. Smallest euclidean distance of n distances establishes the class of the test vector (Balakrishnama & Ganapathiraju, 1999).

2.1.3.4 Gabor Filters

Techniques including Gabor filters are extremely effective and one of the most popular methods as well. Gabor filters derives the multi-orientational information at several scales using a face image. Gabor filters consists of a filter bank which is derived at different scales and orientations. The filter bank including several Gabor filters extracts multi-orientational and multi-scale features of any face image. Obviously, the dimensionality of the input pattern increases based on the number of filters in the filter bank (Štruc, Gajšek & Pavešic, 2009). Hence, the dimensionality of the face pattern is reduced by using some techniques like PCA, subspace projection and downsampling techniques. The more effective size of data is sent to classifier finally.

Gabor filters is widely used in face recognition applications. Their use is preferred due to two major and attractive factor; their computational properties and their biological relevance. A 2-D Gabor filter in the spatial domain is defined by the following expression:

$$\psi_{u,v}(x,y) = \frac{f_u^2}{\pi\gamma\eta} \, e^{\left(x'^2 \frac{f^2}{\gamma^2} + y'^2 \frac{f^2}{\eta^2}\right)} \, e^{j2\pi f_u x'} \tag{2.15}$$

where $x' = x \cos \Theta_v + y \sin \Theta_v$, $y' = -x \sin \Theta_v + y \cos \Theta_v$ and the parameters f_u and Θ_v are defined as $f_u = f_{max}/2^{(v/2)}$ and $\Theta_v = v\pi/8$. As seen above, Gabor filters are formulated as Gaussian kerel functions modulated by a complex plane wave whose center frequency and oientation are defined by f_u and Θ_v, respectively. The γ and η parameters determine the ratio between the center frequency and the size of the Gaussian envelope. The other parameter f_{max} defines the maximum frequency of the filters. The general expression of Gabor filters is like the combination of an even (cos-type) and an odd (sin-type) part in shown in figure below.

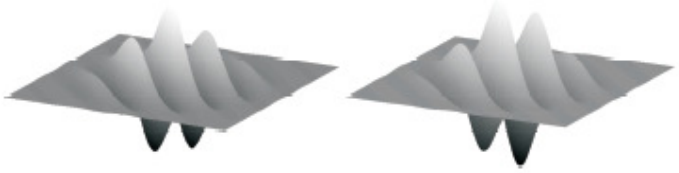


Figure 2.3 An example of a gabor filter: the real part (left), the imaginary part (right) (Štruc, Gajšek & Pavešic, 2009)

Let $I(x,y)$ denote a grey-scale face image and let $\psi_{u,v}(x,y)$ denote a Gabor filter whose parameters are f_u and Θ_v. The Gabor filtering process is defined by the following formulation:

$$G_{u,v}(x,y) = I(x,y) * \psi_{u,v}(x,y) \tag{2.16}$$

where $G_{u,v}(x,y)$ represents a comple convoution result. It can be seperated into real and imaginary part like:

$$E_{u,v}(x,y) = Re\,[G_{u,v}(x,y)] \tag{2.17}$$

$$O_{u,v}(x,y) = Im\,[G_{u,v}(x,y)] \tag{2.18}$$

The phase $\emptyset_{u,v}(x,y)$ and the magnitude $A_{u,v}(x,y)$ filter responses can be computed as:

$$A_{u,v}(x,y) = \sqrt{E_{u,v}^2(x,y) + O_{u,v}^2(x,y)} \tag{2.19}$$

(2.17)

$$\phi_{u,v}(x,y) = \arctan\left(O_{u,v}(x,y) / E_{u,v}(x,y)\right) \quad (2.20)$$

The filter phase response $\emptyset_{u,v}(x,y)$ depends on the spatial lcations and are usually ignored during calculations. The magnitude response $A_{u,v}(x,y)$ is less dependent on the spatial location. So, they are used when deriving Gabor filter based features.

Magnitude responses of the Gabor filters for the constructed filter bank are evaluated to represent a given face image $I(x,y)$ based on Gabor filter based features. The output of each Gabor filters are in the same dimensionality as the input image. Hence, whole size of output data is equal to the total filter number times to initial size of input data. To solve the size problem, a feature selection method or a simple rectangular sampling grid is used to the magnitude responses of Gabor filters. Even the size of any face representation is reduced, it can still have a high dimensionality which is difficult to manage. The common solution to that problem is to apply a feature reduction technique, such as PCA or LDA (Štruc, Gajšek & Pavešic, 2009).

The mathematical properties of Gabor filters obviously has some effects on the characteristics and size of the Gabor face representation. While thee properties are appealing, they can not be the most important part when deriving discriminative and the most of representations of a face pattern. Optimal resolution of filters in the spatial and frequency domain is the best way to derive spatially local features of a confined frequency band. But, it causes a big dimension for any Gabor face representation than the initial size of the input face image. Also, filter characteristics in the filter bank changes the recognition accuracy of the classifier on the Gabor face representation. Different filters from the filter bank are not orthogonal to each other.

2.1.3.5 Neural Network

Neural Network is a biolgical face recognition algorithm. It corresponds to neural system in human body. Like a neuraon receives all electrical inputs, a perceptron sums a weighted form of its inputs. A neural network systems consists of three or more layers. While input layer takes in a weighted sum of image data, output layer

produces a result to decide. Neural Network systems usually contain one or more hidden layer to achieve high recognition rate. But, training time increases exponentially if the system includes more hidden layers than needed. Training phase runs to recognize a person after a neural network is formed. The most widely used training method is back propagation algorithm. It relies on creation weights of connections that will produce high reaction for other face image of trained face image and low reaction for all other face images. During recognition phase, a face image is processed by neural network and network produces a mathematical result that carries information of recognition of input image (Madi, Lahoud & Sawan, 2006).

Although neural network method for face recognition is popular and attractive method, there are some confusing issues. The main problem with neural networks is process time for real-time applications. If an individual is added to training set of face images, whole system has to be trained to recognize new subject. It is a bottleneck because of its consuming time for real-time applications. The other problem is the selection of initial network topologies. This selection is a hard issue since training of neural network systems takes a long time.

2.1.4 Analysis of Face Subspaces

Subspace analysis techniques for face recognition are based on the fact that a class of patterns of interest resides in a subspace of the input image space. For example, a small image of 64 x 64 has 4096 pixels and it can be expressed a large number of pattern classes, such a streets, houses and faces. However, among all possible configurations, only a few correspond to faces. Therefore, the original image representation is highly redundant, and the dimensionality of this representation could be greatly reduced when only the face pattern are of interest.

A small number of eigenfaces may be derived from a set of training face images by using PCA approach. Thus, a face image is efficiently represented as a feature vector (i.e., a vector of weights) of low dimensionality. The features in such subspace

provide more salient and richer information for recognition than the raw image. The use of subspace modeling techniques has significantly advanced face recognition technology. PCA approach will be described briefly in the following sections.

The manifold or distribution of all faces accounts for variation in face appearance whereas the non-face manifold accounts for everything else. If we look into these manifolds in the image space, we find them highly nonlinear and nonconvex. Face detection can be considered as a task of distinguishing between the face and nonface manifolds in the image (subwindow) space and face recognition between those of individuals in the face manifold.

Figure 2.3 further shows the nonlinearity and nonconvexity of face manifolds in a PCA subspace represented by the first three principal components, where the plots are drawn from real face image data.

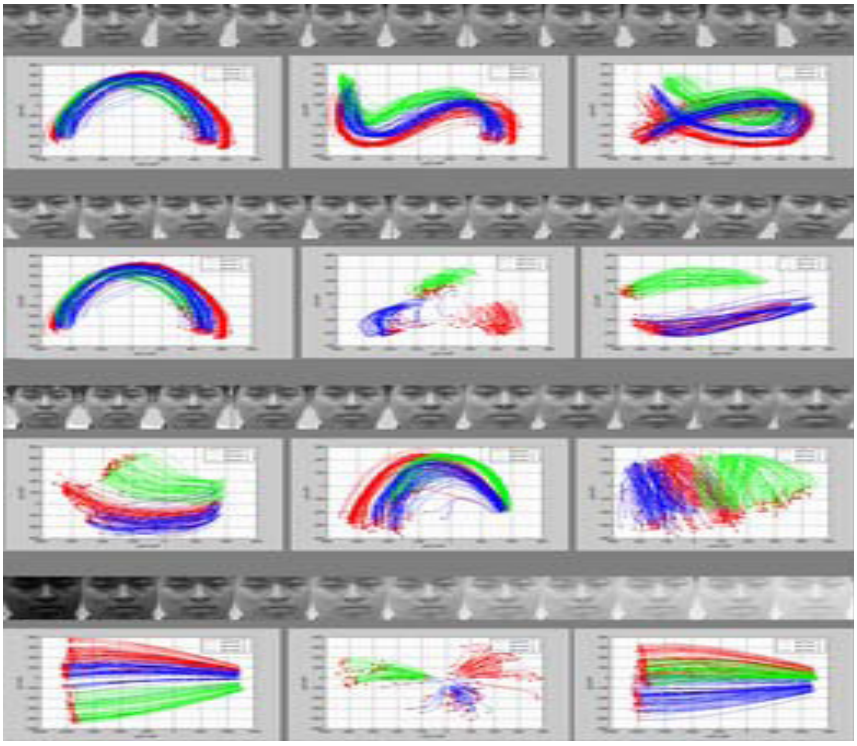


Figure 2.3 Nonlinearity and nonconvexity of face manifolds under translation, rotation, scaling and gamma transformations (Li & Jain 2004)

Each plot depicts the manifolds of three individual components (in three colors) found after process of PCA. There are 64 original frontal face images for each individual. A certain type of transform is performed on an original face image with 11 gradually varying parameters, producing 11 transformed face images; each transformed image is cropped to contain only the face region; the 11 cropped face images form a sequence. A curve in this figure is the image of such a sequence in the PCA space, and so there are 64 curves for each individual. The three-dimensional (3-D) PCA space is projected on three 2-D spaces (planes). The nonlinearity of the trajectories can be seen definitely.

Two notes should follow here: First, while these examples are demonstrated in a PCA space, more nonlinear and more nonconvex curves are expected in the original image space. Second, although these examples demonstrates geometric transformations in the 2-D plane and pointwise lighting and position changes, more significant complexity occurs for geometric transformations in 3-D transformations and lighting direction changes.

2.1.5 Technical Error Sources

The classification problem associated with face detection is highly nonlinear and nonconvex. Many of face recognition research and studies focus on the point that the performance of face recognition methods highly depends on changes in lighting, pose and other factors. Some of these technical challenges are summarized below.

2.1.5.1 Large Variability in Facial Appearance.

The appearance of a face is depend on several other factors, including the facial pose, camera viewpoint, illumination, facial expression, although shape and reflectance are intrinsic properties of a face object. Figure 2.4 shows an example of significant intrasubject variations caused by these factors. In addition to them, various imaging parameters, such as aperture, exposure time, lens aberrations, and sensor spectral response also causes different facial appereances and increase

intersubject variations. Face-based person identification is further complicated by possible small intersubject variations (Figure 2.5). All these factors make image data analysis difficult. So, “the variations between the images of the same face due to illumination and viewing direction are almost always larger than the image variation due to change in face identity”. This variability makes it difficult to extract the intrinsic information of the face objects from their respective images.

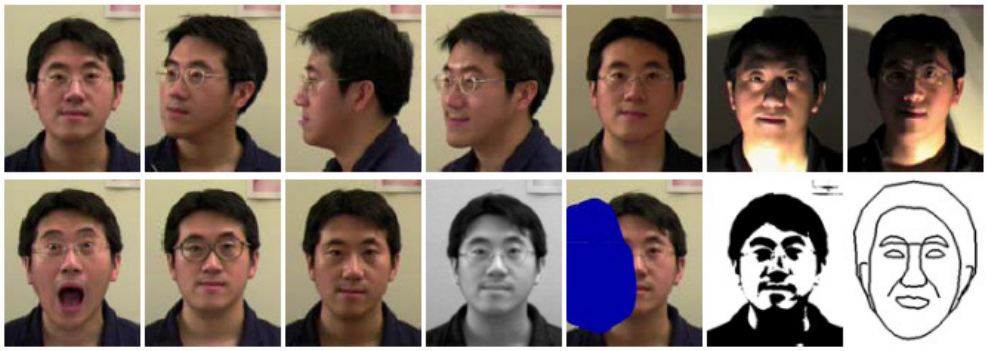


Figure 2.4 Intrasubject variations in pose, illumination, expression, occlusion, accesories (e.g. glasses), color and brightness (Li & Jain 2004)

(a) (b)

Figure 2.5 Similarity of frontal faces between (a) twins and (b) a father and his son (Li&Jain 2004)

2.1.5.2 Highly Complex Nonlinear Manifolds

As described before, the entire face manifold is highly nonconvex. The aim of linear methods such as principal component analysis (PCA), independent component analysis (ICA), and linear discriminant analysis (LDA) are generally same. They

project the data linearly from a high-dimensional space (e.g., the image space) to a low-dimensional subspace. As such, they are unable to preserve the nonconvex variations of face manifolds necessary to differentiate among individuals. In a linear subspace, Euclidean distance and more generally Mahalanobis distance are normally used for template matching. But, they do not perform well for classifying between face and non-face manifolds and between manifolds of individuals in linear subspace as shown in Figure 2.6(a). This critical fact limits the power of the linear methods to achieve more accurate face detection and recognition.

2.1.5.3 High Dimensionality and Small Sample Size

Another fact is the ability to generalize. An example is shown in Figure 2.6(b). A canonical face image of 112 x 92 pixels resides in a 10304-dimensional feature space. Unfortunately, the number of examples per person that is available for learning the manifold is usually much smaller than the dimensionality of the image space. It is generally lower than 10 times and even just one. A system that is trained on so few examples may not characterize unseen instances of the same individual face. It is one of the typical effects which reduces recognition rates dramatically (Li & Jain 2004).

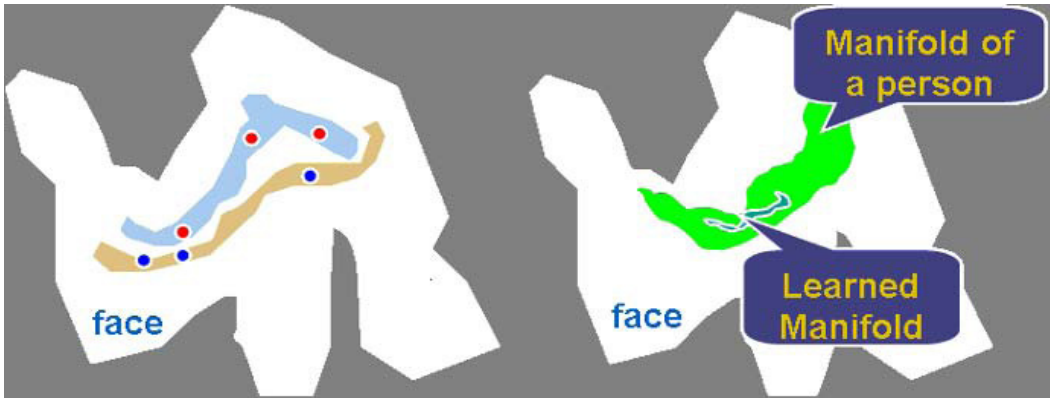


Figure 2.6 Challenges in face recognition from subspace viewpoint. (a) Euclidean distance is unable to differentiate between individuals (b) The learnt manifold or classifier is unable to characterize unseen images of same individual face (Li & Jain 2004)

2.1.6 Technical Solutions

There are two strategies for solving the above difficulties: feature extraction and pattern classification based on the extracted features. One is to construct a good feature space in which the face manifolds become simpler i.e., less nonlinear and nonconvex than those in the other spaces. This includes two levels of processing. Face images are normalized geometrically and photometrically by using techniques such as morphing and histogram equalization. And then, feature extraction methods that are more stable to such variations are preferred. Gabor wavelets are a good example to such methods.

The second strategy is to construct classification engines that are able to solve difficult nonlinear classification and regression problems in the feature space and to generalize better. Although good normalization and feature extraction reduce the nonlinearity and nonconvexity, they do not solve the problems completely. In order to achieve more accurate results and high performance, classification engines able to deal with such difficulties are still needed. A successful algorithm usually combines both strategies.

With the geometric feature-based approach used in the early days, facial features such as eyes, nose, mouth, and chin are detected. Relations between the features are used as descriptors for face recognition. Advantages of this approach include economy and efficiency when achieving data reduction. But this technique is quite insensitive to variations in illumination and viewpoint. Also, facial feature detection and measurement techniques developed up to date are not reliable enough for the geometric feature-based recognition. Because, geometric properties alone are insufficien for face recognition due to rich information loss in the facial appearance is discarded. These are reasons why early techniques are not effective (Li & Jain 2004).

The statistical learning approach learns from training data (appearance images or features extracted from appearance) to extract good features and construct

classification engines. These classification engines may result in acceptable recognition rates. Moreover, they can handle some technical difficulties such variations, suc as facial pose, camera viewpoint, illumination. During the learning, both prior knowledge about faces and variations seen in the training data are taken into account. Many successful algorithms for face detection, alignment and matching nowadays are learning-based.

The appearance-based approach, such as PCA and LDA based methods, has significantly advanced face recognition techniques. This kind of approaches generally operate directly on an image-based representation (i.e., array of pixel intensities). Such an approach extracts features in a subspace derived from training images. Using PCA, a face subspace is created to represent the face object optimally. Using LDA, a discriminant subspace is constructed to distinguish faces of different persons optimally. Several researches on appearance-based approach indicates that LDA-based methods generally yield better results than PCA-based ones.

Although these linear, holistic appearance-based methods avoid instability of the early geometric feature-based methods, they are not still accurate and reliable enough to describe details of original manifolds in the original image space. Because, they have some limitations in handling nonlinearity in face recognition. As an example, protrusions of nonlinear manifolds may be smoothed and concavities may be filled in, causing unfavorable consequences.

Such linear methods can be extended using nonlinear kernel techniques to deal with nonlinearity in face recognition. There, a nonlinear projection (dimension reduction) from the image space to a feature space is performed; the manifolds in the resulting feature space become simple. The advantage of this technique is to preserve details. Although the kernel methods may achieve good performance on the training data, however, it may not be so for unseen data owing to their more flexibility than the linear methods and overfitting thereof (Li & Jain 2004, p:7,8).

2.1.7 Latest Technology Maturity

A face recognition system consists of several components, including face detection, tracking, alignment, feature extraction, and matching. Where are we along the road of making automatic face recognition systems? To answer this question, we have to think of how strong constraints are assumed, including pose, illumination, facial expression, age, occlusion, and facial hair and how succesfully some given constraints can be handled. Even if high succes rates are difficult to obtain for outdoor scenes, new techniques are developed on latest years. On the other hand, real-time face detection and tracking in the normal indoor environment is relatively well solved.

When faces are detected and tracked, alignment can be done, assuming the image resolution is good enough for localizing the facial components. If frontal faces are without strange expressions and under illumination without much shadow, face recognition works well and reach to high recognition rates. Face recognition in an unconstrained daily life environment without the user's cooperation, such as for recognizing someone in an airport, is currently a challenging task and open to development.

2.2 The Principal Component Analysis as Technique of Face Recognition

2.2.1 Mathematics of Principal Component Analysis

A 2-D facial image can be represented as 1-D vector by concatenating each row (or column) into a long thin vector. Let's suppose we have M vectors of size N (= rows of image x columns of image) representing a set of sampled images. p_j's represent the pixel values.

$$x_i = [p_1 \ldots\ldots p_N]^T,\ i = 1, \ldots\ldots, M \tag{2.21}$$

The images are mean centered by subtracting the mean image from each image vector. Let *m* represent the mean image.

$$m = \frac{1}{M}\sum_{i=1}^{M} x_1 \tag{2.22}$$

And let ω_i be defined as mean centered image

$$\omega_i = x_i - m \tag{2.23}$$

Our goal is to find a set of e_i's which have the largest possible projection onto each of the ω_i's. We wish to find a set of M orthonormal vectors e_i for which the quantity

$$\lambda_i = \frac{1}{M}\sum_{i=1}^{M} (e_i^T \omega_n)^2 \tag{2.24}$$

is maximized with the orthonormality constraint

$$e_i^T e_k = \delta_{ik} \tag{2.25}$$

It has been shown that the e_i's and λ_i's are given by the eigenvectors and eigenvalues of the covariance matrix

$$C = W W^T \tag{2.26}$$

where W is a matrix composed of the column vectors ω_i placed side by side. The size of C is N x N which could be enormous. For example, images of size 64 x 64 create the covariance matrix of size 4096 x 4096. It is not practical to solve for the eigenvectors of C directly. A common theorem in linear algebra states that the vectors e_i and scalars λ_i can be obtained by solving for the eigenvectors and

eigenvalues of the M *x* M matrix W^TW. Let d_i and μ_i be the eigenvectors and eigenvalues of W^TW, respectively.

$$W^T W d_i = \mu_i d_i \tag{2.27}$$

By multiplying left to both sides by W

$$W^T W (W d_i) = \mu_i (W d_i) \tag{2.28}$$

which means that the first $M - 1$ eigenvectors e_i and eigenvalues λ_i of WW^T are given by Wd_i and μ_i, respectively. Wd_i needs to be normalized in order to be equal to e_i. Since we only sum up a finite number of image vectors, M, the rank of the covariance matrix cannot exceed $M - 1$. (The -1 come from the subtraction of the mean vector m).

The eigenvectors corresponding to nonzero eigenvalues of the covariance matrix produce an orthonormal basis for the subspace within which most image data can be represented with a small amount of error. The eigenvectors are sorted from high to low according to their corresponding eigenvalues. The eigenvector associated with the largest eigenvalue is one that reflects the greatest variance in the image. That is, the smallest eigenvalue is associated with the eigenvector that finds the least variance. They decrease in exponential fashion, meaning that the roughly 90% of the total variance is contained in the first 5% to 10% of the dimensions.

A facial image can be projected onto M' ($<< M$) dimensions by computing,

$$\Omega = [v_1 v_2 v_3 \ldots . v_M]^T \tag{2.29}$$

where $v_i = e_i^T . \omega_i$. v_i is the i^{th} coordinate of the facial image in the new space, which came to be the principal component. The vectors e_i are also images, so called, eigenimages, or eigenfaces in our case. They can be viewed as images and indeed look like faces. So, Ω describes the contribution of each eigenface in representing the

facial image by treating the eigenfaces as a basis set for facial images. The simplest method for determining which face class provides the best description of an input facial image is to find the face class k that minimizes the Euclidean distance

$$\epsilon_k = \|(\Omega - \Omega_k)\| \quad (2.30)$$

where Ω_k is a vector describing the k^{th} face class. If ϵ_k is less than some predefined threshold Θ_c, a face is classified as belonging to the class k.

2.2.2 How to Use PCA?

In this section, the use of PCA as a feature extractor is explained. Turk and Pentland (1991) describes that PCA is an algorithm that treats face recognition as a two dimensional recognition problem. The correctness of this algorithm relies on the fact that the faces are uniform in posture and illumination. PCA can handle minor variations in these two factors, but performance is maximized if such variations are limited. The algorithm basically involves projecting a face onto a face space, which captures the maximum variation among faces in a mathematical form.

During the training phase, each face image is represented as a column vector, with each entry corresponding to an image pixel. These image vectors are then normalized with respect to the average face. Next, the algorithm finds the eigenvectors of the covariance matrix of normalized faces by using a speedup technique that reduces the number of multiplications to be performed. This eigenvector matrix is then multiplied by each of the face vectors to obtain their corresponding face space projections. Lastly, the recognition threshold is computed by using the maximum distance between any two face projections

Assume that we have p training images: Ω_i; $i = 1, 2, ..., p$. For each training image, pixel vectors Φ_i should be formed where $\Phi_i \in \acute{R}^k$, $(k = M \times N)$. Our aim is to compute feature vectors ω_i where $\omega_i \in \acute{R}^k$, $(d << k)$. In order to apply PCA to the training set, a training data matrix A should be first formed which data training matrix A contains p rows: at each row Φ_i's are stored. Thus the dimensionality of A is

p x k. First, the covariance matrix of A: C_A is computed. Then the eigenvalues and their corresponding eigenvectors of C_A should be computed. There will be k eigenvalue and eigenvector pairs where each eigenvector e_i is of dimensionality k. The eigenvalues are sorted in decreasing order and the biggest d eigenvalue and eigenvector pairs are selected. The transformation matrix ψ are formed by simply putting the selected eigenvectors as columns of ψ . The transformation matrix Ψ will be used to compute ω_i's from Φ_i's. The computation of ω_i is simply by:

$$\omega_i = \psi^T \Phi_i^T \tag{2.31}$$

where ψ^T and Φ_i^T are the transposes of ψ^T and Φ_i, respectively. Note that each column of ψ corresponds to an eigenvector which is of length k. This is equal to M x N which is the dimensionality (resolution) of input images. Thus, each eigenvector is converted to an image by reversing the concatenation operation. These converted eigenvector images are called eigenfaces since they are similar to human faces.

Once ω_i's are obtained using the largest d eigenvectors, the image of person i can be reconstructed easily. If all k eigenvectors instead of d when forming ψ are used, the reconstructed image will the same as image Ω_i. However, since our aim is dimensionality reduction and $d << k$, reconstructed image Ω_i will be an approximation of the actual Ω_i. The reconstructed image Ω_i can be reconstructed by converting the pixel vector: $\Phi_i = (\psi\ \omega_i)^T$ to an image of resolution M x N.

In the recognition phase, a subject face is normalized with respect to the average face and then projected onto face space using the eigenvector matrix. Next, the Euclidean distance is computed between this projection and all known projections. The minimum value of these comparisons is selected and compared with the threshold calculated during the training phase. Based on this, if the value is greater than the threshold, the face is new. Otherwise, it is a known face.

CHAPTER THREE

SPEECH RECOGNITION & FOURIER ANALYSIS

3.1 Speech Recognition

3.1.1 What is Speech Recognition?

Speech recognition is the process of a machine or program to recognize the phrases spoken by the user and convert it to a format, machine can read.Performance of speech recognition systems may vary due to algorithms developed. More sophisticated algorithms has the ability to recognize the phrases from an unlimited database while some systems has a limited phrase range and may process them if they are spoken clearly.

Speech recognition systems can be categorized based on some parameters. As isolated-word speech recognition needs a paues between phrases while continuous speech recognition doesn't need. Spontaneous speech contains disfluencies and is much more difficult to recognize than speech read from script since it is based on the characteristics of voice of the speaker. Other type of speech recognition is in the form of speaker verification. In that case, speech samples are recorded before using them whereas other systems are said to be speaker-independent.

3.1.2 Elemantary Concepts and Terminology of Speech Recognition

Speech recognition plays an important role for biometric recognition systems in last years, although it still has many problems to be solved. It is basically divided into speech identification and speech verification like other biometric recognition systems. Speech verification aims to determine if a person is really the person who he/she claims to be, while focus of speaker identification is to determine who is the real owner of the test speech.

Speech recognition systems can be divided into text dependency; text-dependent and text-independent methods. Text-dependent systems work on the recognition of a specific phase. User should speak what is written in the phase. On the other hand, text-independent systems focus on the characteristics of the speech independently from the text spoken (Delioğlu, 2007).

3.1.2.1 Identification and Verification Tasks

In the speech identification task, called also 1:N matching, the speech of an unknown speaker is compared to speech samples in a database of N known speakers. The best matching is labeled as the owner of the speech of unknown speaker. The speech verification task, called also 1:1 matching, is a different issue and aims to confirm if the given voice sample belongs to a claimed speaker. Verification task is popular for especially security applications. By the way of example, an identity claim such as PIN code etc. is collected by the system. The voice sample is compared to against the claimed speaker's voice sample. The decision is based on the similarity ratio between the voice samples of the unknown speaker and the claimed speaker. If the result exceeds the defined threshold level, the speaker is confirmed. Otherwise, the speaker is rejected by the system.

When the difficulties of identification and verification tasks are compared, identification task is considered more difficult. Because the number of speakers in database increases, the recognition rate of the system may probably reduce and the system may fails. In theory, verification task isn't so dependent on the number of speakers since the verification is based on comparison between two people.

There is one more classification in speech identification task. It can be divided into open-set and closed-set tasks. If the unknown speaker is assumed to be a claimed member in database, the identification problem is a closed-set problem. If possibly, target speaker isn't one of the registered speakers, the task is called as open-set problem. Open-set problem has some challenging points compared to closed-set problem. In closed-set problem, decision level is compared to a threshold level. The

best matching is forced to one of the registered speakers. But in open-set problem, the similarity level is compared to predefined tolerance level. If it is out of tolerance level, there is no need to search the best matching in database. Because, the result will not indicate the correct claimed speaker (Delioğlu, 2007).

3.1.2.2 Text-Independent and Text-Dependent Tasks

As mentioned before, speaker recognition can be divided into text-dependent and text-independent tasks. In text-dependent task, the password, PIN code etc. is known already and the unknown speaker says it definitely. In text-independent task, there is no information about the text spoken. Hence, all system is based on extracting the properties of the speaker's vocal space.

In general, text-dependent systems determine more accurate results due to double check of both content and voice. For instance, a speech recognizer can be used in recognizing whether the user utters the sentence that the system prompted to the user. This is called as speech verification and for better results, it may supported with speaker verification.

In text-independent recognition, the same phrase is processed for all times. The phrase is changed for a different verification process. Then, the user is confirmed if this phrase is presented to the system. In that case, it is a flexible way to create a pass phrase by concatenating different words. It is called as text-prompted speaker verification. This method has some advantages that a possible intruder cannot detect what the new phrase will be. Also, it may be difficult for the user to playback the pre-recorded phrase. Furthermore, in order to make the system safer, time interval can be limited to utter the pass phrase to prevent the unwanted guests to use it.

3.1.3 Dimension of Difficulty

There are different sources of error in speech recognition. Some of them are related to the speaker itself, and some to the technical conditions.

3.1.3.1 Intra-Individual Variation

It is a well known fact that the physical, psychological states of health and stimulants and drugs may affect the quality of speakers. By the way of example, smokers often have low voice quality than other people or acoustic parameters of depressive subjects are really different from those under relaxed state (Delioğlu, 2007).

The other factor that affects the speaker's voice in long term is aging, weight changes and other physiological changes. According to several experiments, internal variability is the most important change source for the speaker's voice. Some experiences in speech recognition indicate that voice recorded during the same day with the same environmental conditions may not be recognized correctly. Also, if the training material is poorly designed, voice of some speakers can't be modeled and poor recognition rates are obtained. In general, training data should be phonetically balanced. So that, it contains instances of all sounds of the language in different contexts. Otherwise, an arbitrary input data can't be presented to the system.

3.1.3.2 Voice Disguise and Mimicry

Voice disguise means changing the speaker's voice. Hence, it can't match with the other sample from the same speaker. There are some reasons of disguise. For instance, when making a blackmail call, the criminal may keep his nostrils closed, or he might alter his voice in police investigations. Some research have tried to fix that problem. By the way of example, three different parameter sets were studied in order to find out which of these are the most robust against different types of disguise. One of these three parameters was to talk with a pencil between the front teeth and the other one was to talk by whispering. However, they studied only variation within speakers so they did not take into account the inter-speaker variation. However, they studied only variation within speakers so they did not take into account the inter-speaker variation. Even though a certain parameter would give small intra-speaker variation in respect to disguise, it might be a poor parameter otherwise for speaker

recognition. It seems that there is room for studying disguise-resilient parameters. Imitation (impersonation, mimicry) is a special type of voice disguise where the speaker tends to map his voice to sound like another speaker. Disguise and imitation definitely reduces the recognition rate determined from the speech recognition systems. the performance of speaker recognition systems (Delioğlu, 2007).

3.1.4 Technical Error Sources

Several technical error sources may reduce the recognition rate of speech recognition systems. Some of them are shown in Figure 3.1. So that, some assumptions are applied in noise suppression algorithms. The popular assumptions are in the following :

- noise is stable in short term,
- noise has zero mean and
- noise is uncorrelated with the speech signal.

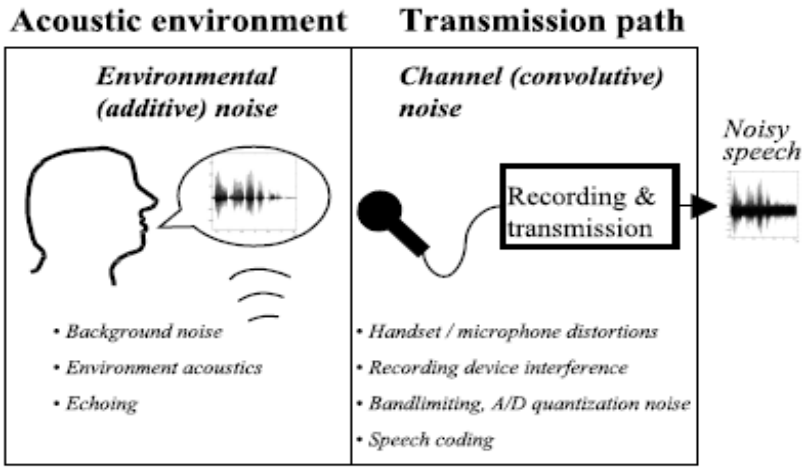


Figure 3.1 Technical error sources in speaker recognition (Delioğlu, 2007)

Since noise is the most important effect which reduces the performance, algorithms are developed to remove the unwanted noise factor. When the speech is recorded with a microphone, environmental noise caused by car engine, computer hum, traffic noise etc. is added to the speech signal. Reverberation adds delayed versions of the original signal to the recorded signal.

Poor-quality microphones affect the true speech signal by adding some nonlinear distortion. By comparing pairs of same speech signals recorded with good-quality and poor-quality microphones, some artifacts such as phantom formants that are created in the form of sums, multiples and differences of the true formants, are observed. Formant bandwidths are also widened and the overall spectral shape is flattened. The noise sources are the A/D converters which add its own distortion, and the recording device which introduces a mobile phone radio-waves. Since the compression techniques add noise into the signal, a telephone network is also noise source. The speech signal processed by the recognition device is never same wave which is transmitted from the speaker's lips. It is sustained to several transformations degrading the quality. So that, the circumstances in the training and testing phases are different from each other. In addition to intra-individual variation, technical mismatches arise from the following factors:

- Environmental acoustics mismatch
- Mismatch in the type and amount of background noise
- Microphone type and recording device mismatch

3.2 Fourier Transform

3.2.1 Background of Fourier Transform

The basic idea behind Fourier Transform is to form any function $f(x)$ as a summation of a series of sine and cosine terms of increasing frequency. In other

words, any space or time varying data can be transformed into a different domain called the frequency space. Joseph Fourier first came up with the idea in the 19th century, and it was proven to be useful in various applications, mainly in signal processing (Yoo, 2001).

Frequency domain offers some attractive advantages for recognition techniques. It makes large filtering operations much faster, and it collects information together in different ways which can sometimes separate signal from noise. Furthermore, the Fourier transform makes it easy to go forwards and backwards from the spatial domain to the frequency space. Assume that an image is converted the frequency space. In that case, any periodic noise in the original image will show up as bright spots. If those points are removed successfully by using a desired filter and inverse Fourier Transform is applied to get the original image, most of the noise is removed and visibility of that image improves (Yoo, 2001).

3.2.2 Theory of Fourier Transform

Any function $f(t)$ which is periodic, can be described by decomposing into sin's and cos's. Also, complex exponentials may be used in place of sin's and cos's. Since they lead to less writing and simpler computations, exponentials can easily be converted into sin's and cos's (Feldman, 2007). If $f(t)$ has period 2ℓ, its (complex) Fourier series expansion is;

$$f(t) = \sum_{k=-\infty}^{\infty} c_k \, e^{ikt\frac{\pi}{l}} \quad with \quad c_k = \frac{1}{2l} \int_{-l}^{l} f(t) e^{ikt\frac{\pi}{l}} \, dt \qquad (3.1)$$

For simplicity, only develop the expansions for functions that are zero for all sufficiently large $|t|$ will be derived. With a little more work, one can show that these conclusions apply to a much broader class of functions. Let $L>0$ be sufficiently large that $f(t) = 0$ for all $|t| \geq L$. We can get a Fourier series expansion for the part of $f(t)$ with $-L < t < L$ by using the periodic extension trick (Feldman, 2007). Define $F_L(t)$ to be the unique function determined by the requirements that;

- $F_L(t) = f(t)$ for $-L < t \leq L$

- $F_L(t)$ is periodic of period $2L$

Then, for $-L < t < L$,

$$f(t) - F_L(t) - \sum_{k=-\infty}^{\infty} c_k\ (L)\ e^{ikt\frac{\pi}{l}} \quad where \quad c_k(L) = \frac{1}{2L}\int_{-L}^{L} f(t)\ e^{ikt\frac{\pi}{l}}\ dt \qquad (3.2)$$

If the limit $L \to \infty$ can be somehow taken, a representation of f that is is valid for all t's, will be created, not just those in some finite interval $-L < t < L$. By the simple expedient of interpreting, the sum as a Riemann sum approximation to a certain integral will be developed. For each integer k, define the k^{th} frequency to be $\omega_k = k\ \pi/L$ and denote by $\Delta\omega = \pi/L$ the spacing , $\omega_{k+1} - \omega_k$, between any two successive frequencies. Also define $f'(\omega) = \int_{\infty} f(t)\ e^{-i\omega t}\ \mathrm{dt}$. Since $f\ (t) = 0$ for all $|t| \geq L$

$$c_k(L) = \frac{1}{2L}\int_{-L}^{L} f(t)\ e^{ikt\frac{\pi}{l}}\ dt = \frac{1}{2L}\int_{-\infty}^{\infty} f(t)\ e^{i(k\frac{\pi}{l})t}\ dt \qquad (3.3)$$

$$= \frac{1}{2L}\int_{-\infty}^{\infty} f(t)\ e^{i\omega_k t}\mathrm{dt} = \frac{1}{2L}\ f'(\omega_k) = \frac{1}{2L}\ f'(\omega_k)\ \Delta\omega \qquad (3.4)$$

In this notation,

$$f(t) = F_L(t) = \frac{1}{2L}\sum_{k=\infty}^{\infty} f'(\omega_k)\ e^{i\omega_k t}\ \Delta\omega \qquad (3.5)$$

for any $-L < t < L$. Due to the assumptation *of* $L \to \infty$, the restriction $-L < t < L$ disappears, and the right hand side converges exactly to the integral $\int_{-\infty}^{\infty} f(\omega)\ e^{-i\omega t} d\omega$. Taking the limit of the last equation as $L \to \infty$ gives

$$f(t) = \frac{1}{2L}\int_{-\infty}^{\infty} f'(\omega)\ e^{i\omega t}\mathrm{d}\omega \quad where \quad f'(\omega) = \int_{-\infty}^{\infty} f'(t)\ e^{i\omega t}\mathrm{dt} \qquad (3.6)$$

The function f' is called the Fourier transform of f. It is to be thought of as the frequency profile of the signal f(t). The Fourier transform of f(t) locates the

frequency components of the signal f(t). The new function itself is called as *frequency domain representation* of the original function (Feldman, 2007).

3.2.3 Selected Properties of Fourier Transform

3.2.3.1 Linearity

If α and β are any constants and we build a new function $h(t) = \alpha\, f(t) + \beta\, g(t)$ as a linear combination of two old functions $f(t)$ and $g(t)$, then the Fourier transform of h is;

$$h'(\omega) = \int_{-\infty}^{\infty} h'(t)\ e^{-i\omega t}\mathrm{d}t = \int_{-\infty}^{\infty} [\alpha f(t) + \beta g(t)]\ e^{-i\omega t}\mathrm{d}t \tag{3.7}$$

$$= \alpha \int_{-\infty}^{\infty} f(t)\ e^{-i\omega t}\mathrm{d}t + \beta \int_{-\infty}^{\infty} g(t)\ e^{-i\omega t}\mathrm{d}t \tag{3.8}$$

$$= \alpha\, f'(\omega) + \beta\, g'(\omega) \tag{3.9}$$

3.2.3.2 Time Shifting

Suppose that we build a new function $h(t) = f(t - t_0)$ by time shifting a function $f(t)$ by t_0. The easy way to check the direction of the shift is to note that if the original signal f(t) has a jump when its argument $t = a$, then the new signal $h(t) = f(t - t_0)$ has a jump when $t - t_0 = a$, which is at $t = a + t_0$, t_0 units to the right of the original jump (Feldman, 2007).

The Fourier transform of h is

$$h'(\omega) = \int_{-\infty}^{\infty} h(t)\ e^{-i\omega t}\mathrm{d}t = \int_{-\infty}^{\infty} f(t - t_0)\ e^{-i\omega t}\mathrm{d}t = \int_{-\infty}^{\infty} f(t')\ e^{-i\omega(t + t_0)}\mathrm{d}t' \tag{3.10}$$

$$= e^{-i\omega t_0} \int_{-\infty}^{\infty} f(t')\ e^{-i\omega t'}\mathrm{d}t = e^{-i\omega t_0}\, f'(\omega) \tag{3.11}$$

3.2.3.3 Scaling

If we build a new function $h(t) = f(t/\alpha)$ by scaling time by a factor of $\alpha > 0$, then the Fourier transform of h is;

$$h'(\omega) = \int_{-\infty}^{\infty} h(t)\, e^{-i\omega t}\mathrm{d}t = \int_{-\infty}^{\infty} f\left(\frac{t}{\alpha}\right) e^{-i\omega t}\mathrm{d}t = \alpha \int_{-\infty}^{\infty} f(t')\, e^{-i\omega\alpha t'}\mathrm{d}t' \tag{3.12}$$

$$= \alpha\, f'(\alpha\omega) \tag{3.13}$$

where $t = \alpha\, t'$, $dt = \alpha\, dt'$

3.2.3.4 Differentiating

If we build a new function $h(t) = f'(t)$ by differentiating an old function $f(t)$, then the Fourier transform of h is;

$$h'(\omega) = \int_{-\infty}^{\infty} h(t)\, e^{-i\omega t}\mathrm{d}t = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\mathrm{d}t \tag{3.14}$$

Now integrate by parts with $u = e^{-i\omega t}$ and and $dv = f'(t)$ dt. So that $du = -i\omega\, e^{-i\omega t}$ and $v = f(t)$. Assuming that $f(\pm\infty) = 0$, this gives

$$h'(\omega) = \int_{-\infty}^{\infty} u\, dv = uv\,|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} v\, du = -\int_{-\infty}^{\infty} f(t)(-i\omega)\, e^{-i\omega t}dt = i\omega f'(\omega) \tag{3.15}$$

3.2.3.5 Duality

The duality property says that if we build a new time–domain function $g(t) = f'(t)$ by exchanging the roles of time and frequency, then the Fourier transform of g is

$$g'(\omega) = 2\pi f(-\omega) \tag{3.16}$$

To verify this, just write down just the definition of *g'(ω)* and the Fourier inversion formula for *f(t)* and, in both integrals, make a change of variables, so that the integration variable is:

$$g'(\omega) = \int_{-\infty}^{\infty} g(t)\ e^{-i\omega t}\mathrm{dt} = \int_{-\infty}^{\infty} f'(t)\ e^{-i\omega t}\mathrm{dt} = \int_{-\infty}^{\infty} f'(s)\ e^{-i\omega s}\mathrm{ds} \qquad (3.17)$$

where t = s

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f'(\omega)\ e^{-i\omega t}\mathrm{d\omega} = \frac{1}{2\pi}\int_{-\infty}^{\infty} f'(s)\ e^{-ist}\mathrm{ds} \qquad (3.18)$$

where w = s

So *g'(ω),* which is given by the last integral of the first line is exactly 2π times the last integral of the secondline with *t* replaced by $-\omega$, which is $2\pi f(-\omega)$ (Feldman, 2007).

CHAPTER FOUR

FIELD PROGRAMMABLE GATE ARRAYS

4.1 Introduction to Field Programmable Devices

The process of design procedure of digital hardware has dramatically changed in last years in parallel to the development of new types of sophisticated field-programmable devices (FPDs). Digital devices consists generally high-density devices on these days. Unlike today's generations of technology, board-level designs included large numbers of SSI chips containing basic gates before the development of FPDs technology. This is in use not only to custom devices like processors and memory, but also for logic circuits such as state machine controllers, counters, registers, and decoders. When such circuits are destined for high-volume systems, they have been integrated into high-density gate arrays. However, gate array costs often are too expensive and gate arrays take too long to manufacture to be ready for prototyping or other low-volume scenarios. For these reasons, most prototypes and also many production designs are now built using FPDs. The biggest advantages of FPDs are instant manufacturing time, low start-up costs, low financial risk, ease of design changes etc. As it is detected from name, FPDs are able to be programmable by the end user in the field. It makes them much popular (Brown & Rose, 1996).

The market for FPDs has grown dramatically over the past decade. Now, there is a wide assortment of devices, which a designer has to research and to choose from. A designer faces a task to analyze the different types of chips, understand which adapts the best for, choose a particular manufacturers' product, learn the intricacies of vendor-specific software and then design the hardware. Today, number of FPDs available is not only confusing points for designers. As new technologies are developed, complexity of the more sophisticated devices grows dramatically. The purpose of this stage is to provide an overview of the architecture of the various types of FPDs. In the next stages of chapter, an overview of field programmable gate arrays and development steps will be provided definitely.

4.2 Evolution of Programmable Logic Devices

The first type of user-programmable chip that could implement logic circuits was the Programmable Read-Only Memory (PROM). Address lines of PROM are dedicated as logic circuit inputs and data lines as outputs. Logic functions, however, rarely require more than a few product terms, and a PROM contains a full decoder for its address inputs. That is why PROMs provide an inefficient architecture for realizing logic circuits, and so are rarely used in practice for that purpose. The first device developed later specifically for implementing logic circuits was the Field-Programmable Logic Array (FPLA), or simply PLA for short. A PLA consists of two levels of logic gates: a programmable wired AND-plane followed by a programmable wired OR-plane. The purpose of a PLA architecture is to AND any of its inputs in the AND'ed plane. Each AND-plane output can correspond to any any product term of the inputs. Similarly, each OR-plane output can be structured to produce the logical sum of any of the AND-plane outputs. With this structure, PLAs are well-suited for implementing logic functions in sum-of-products form. They present quite advantages to designers. Because, AND and OR gates can be configured and many inputs are processed (Brown & Rose, 1996).

When PLAs were introduced in the early 1970s by Philips, their main disadvantages were their high costs to manufacture and low performance. Both disadvantages were caused by design based on the two levels of configurable logic, because programmable logic planes were difficult to manufacture and significant propagation delays occurred in the total of both logic planes. To overcome these weaknesses, Programmable Array Logic (PAL) devices were developed. As shown in Figure 4.1(b), PALs are designed due to a single level of programmability, consisting of a programmable wired AND-plane that feeds fixed OR-gates. To overcome configurability absence of PALs incurred because the OR- plane is fixed, a several variants of PALs are produced, with different numbers of inputs and outputs, and various sizes of OR-gates. PALs usually contain flip-flops connected to the OR-gate outputs, so that sequential circuits can be realized. PAL devices are important in designs of digital circuits. Because they were used effectively on many digital

hardware designs. Also some new, more complicated architecture are designed on the basis of PALs. These new designs will be described shortly. Variants of the basic PAL architecture are featured in several other products known by different acronyms. All small PLDs, including PLAs, PALs, and PAL-like devices are grouped into a single category called Simple PLDs (SPLDs), whose most important characteristics are low-cost and very high pin-to-pin speed-performance due to less propagation delay of PALs .

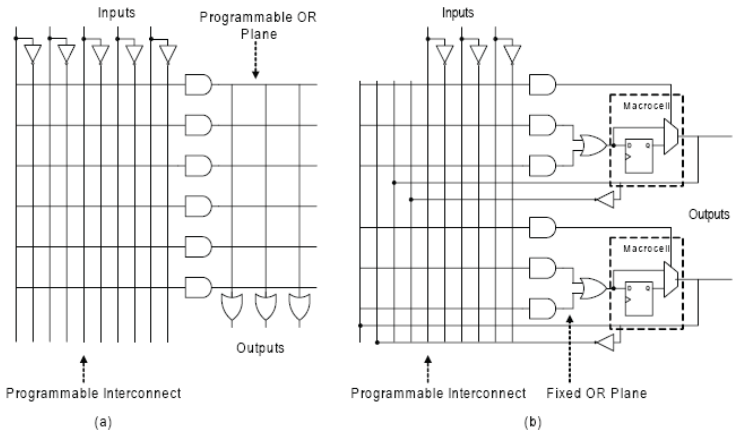


Figure 4.1 Structure of a PLA (a) and PAL(b) (Brown & Rose, 1996)

As more complicated and technological logic arrays has developed, it has been possible to produce devices whose capacity is higher than SPLDs. The difficulty with increasing capacity of a SPLD architecture is their more complex structure. In such a way that, as the number of inputs is increased, the structure of the programmable logic-planes grow dramatically in size. The only feasible way to provide large capacity devices based on SPLD architectures is then to integrate multiple SPLDs onto a single chip and provide interconnect to programmable connect the SPLD blocks together. Hence, although number of inputs grows, useless complexity does not exist. Many commercial FPD products exist on the market today with this basic structure, and are collectively referred to as Complex PLDs (CPLDs). The first CPLD chip family that is called also as Classic EPLDs were developed by Altera. The company presented three more chip family called as MAX 5000, MAX 7000 and MAX 9000 to the market. Growing market of large FPDs became attractive for

all other chip companies, and they developed new devices in the CPLD category. Now, there are now many CPLD devices, that the designers may choose. CPLDs provide logic capacity up to the equivalent of about 50 typical SPLD devices. However, it is quite difficult to extend these architectures to higher densities due to complexity. To build FPDs with very high logic capacity, a different approach is needed (Brown & Rose, 1996).

The highest capacity available today are the traditional gate arrays referred to as Mask-Programmable Gate Arrays (MPGAs). MPGAs consist of an array of pre-fabricated transistors. These transistors are customized into the user's logic circuit by connecting the transistors with wires. Customization is performed during chip fabrication by specifying the metal interconnect. That is why it is quite for a user to employ an MPGA. Because, a large setup cost is involved and manufacturing time is long. MPGAs are clearly not FPDs. However, they are described here because of similar characteristic to the design of the user-programmable equivalent: Field-Programmable Gate Arrays (FPGAs). Like MPGAs, FPGAs consists of an array of uncommitted circuit elements, called logic blocks, and they are interconnected specifically. However, unlike MPGAs, FPGA configuration is performed through programming by the end user. An illustration of a typical FPGA architecture is shown in Figure 4.2. FPGAs are the only type that has a huge number of logic capacity. Hence, FPGAs have been responsible for a major role in digital hardware designs.

4.3 Field Programmable Gate Array (FPGA)

4.3.1 History of FPGA

Field programmable gate arrays (FPGAs) can be used to implement just about any hardware design. One common use of the FPGA is the prototyping of a piece of hardware that will eventually be implemented later into an ASIC. Nevertheless, FPGAs have been increasingly used as the final product platforms (Deschamps,

Bioul & Sutter, 2006). Their use depends, for a given project, on the relative weights of desired performances, development, and production costs.

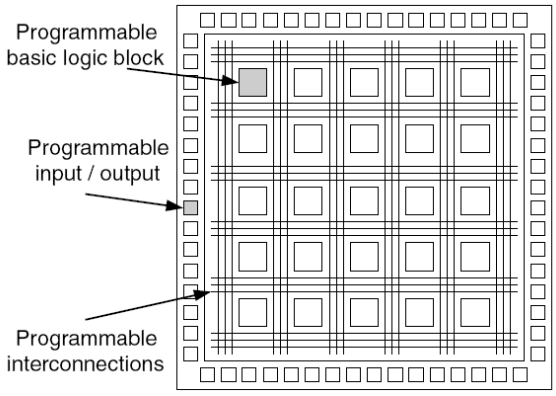


Figure 4.2 Structure of a FPGA(Brown & Rose, 1996)

By the early 1980s, most of the typical logic circuit systems such as microprocessors, bus-I/O controllers, system timers etc. were implemented within a small variety of standard large scale integrated (LSI) circuits. Nevertheless, every system still had the need for random "glue logic" to connect the large ICs, for example, generate global control signals and data formatting like serial to parallel, multiplexing, etc.

Custom ICs were often designed to replace the large amount of glue logic. These systems have provided many advantages by reducing system complexity and manufacturing cost, as well as improving performances. However, custom ICs are expensive to develop, while generating time-to-market (TTM) delays because of the prohibitive design time. Therefore, the custom IC approach was only viable for products with very high volume and not TTM sensitive. Coping with this problem, Xilinx™ (a startup company) introduced, in 1984, the FPGA technology as an alternative to custom ICs for implementing glue logic. With support of computer-aided design (CAD) tools, now FPGA circuits can be implemented in a relatively short amount of time: no physical layout process, no mask making, no IC manufacturing, lower NRE costs, and short TTM (Deschamps, Bioul & Sutter, 2006).

4.3.2 Basic FPGA Concepts

The basic FPGA architecture consists of a two dimensional array of logic blocks and flip-flops with means for the user to configure the function of each logic blocks, the inputs/outputs, and the interconnection between blocks. Families of FPGAs differ from each other by the physical means for implementing user programmability, arrangement of interconnection wires, and basic functionality of the logic blocks (Deschamps, Bioul & Sutter, 2006).

4.3.2.1 Programming Methods

4.3.2.1.1 SRAM Based Technology. FPGA connections are achieved using pass-transistors, transmission gates, or multiplexers that are controlled by SRAM cells. This technology allows fast in-circuit reconfiguration. The major disadvantages are the size of the chip, required by the RAM technology, and the needs of some external source (usually external nonvolatile memory chips) to load the chip configuration. The FPGA can be programmed an unlimited number of times. The figure shown in Figure 4.3 shows (a) SRAM based programming method and (b) antifuse technology, respectively (Deschamps, Bioul & Sutter, 2006).

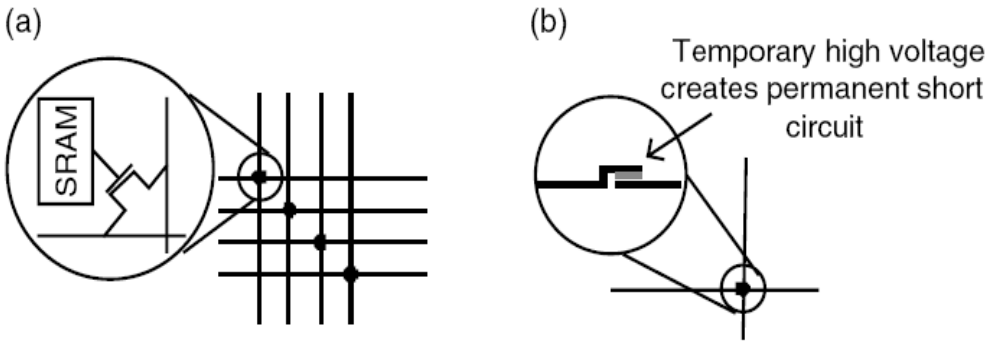


Figure 4.3 Programming methods (a) SRAM connection and (b) antifuse (Deschamps, Bioul & Sutter, 2006)

4.3.2.1.2 Antifuse Technology. An antifuse remains in a high-impedance state, until it is programmed into a low-impedance or fused state. This technology can be

used only once on one-time programmable (OTP) devices. It is less expensive than the RAM technology.

4.3.2.1.3 EPROM/EEPROM Technology. This method is the same as that used in EPROM/EEPROM memories. The configuration is stored within the device, that is, without external memory. Generally, in-circuit reprogramming is not possible.

4.3.2.2 Look-Up Tables

The way logic functions are implemented in a FPGA is another key feature. Logic blocks that carry out logical functions are look-up tables (LUTs), implemented as memory, or multiplexer and memory. Figure 4.4 shows these alternatives, together with an example of memory contents for some basic operations. A 2^n x 1 ROM can implement any n-bit function. Typical sizes for n are 2, 3, 4, or 5.

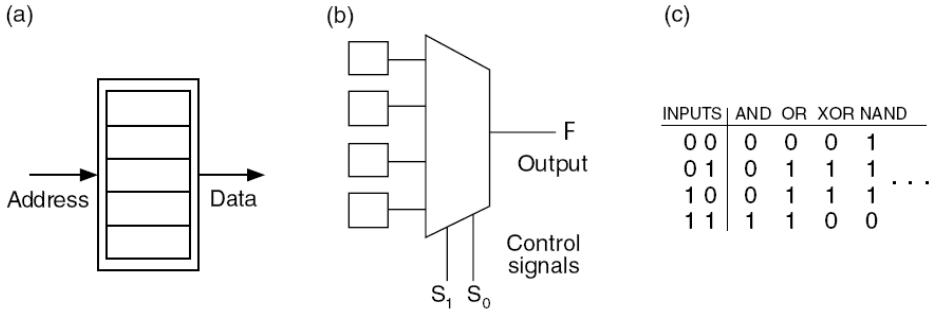


Figure 4.4 Look-up table implemented as (a) memory, (b) multiplexer and memory, (c) memory contents example for different logic functions (Deschamps, Bioul & Sutter, 2006)

In Figure 3.4(a), an n-bit LUT is implemented as a 2^n *x 1* memory; the input address selects one of 2^n memory locations. The memory locations (latches) are normally loaded with values from the user's configuration bit-stream. In Figure 3.4(b), the multiplexer control inputs are the LUT inputs. The result is a general-purpose "logic gate." An n-LUT can implement any n-bit function. An n-LUT is a direct implementation of a function truth table. Each latch location holds the value of

the function corresponding to one input combination (Deschamps, Bioul & Sutter, 2006).

4.3.2.3 FPGA Logic Blocks

A simplified FPGA logic block can be designed with a LUT, typically a 4-input LUT, implementing a combinational logic function, and a register that optionally stores the output of the logic generator, as shown in Figure 3.5.

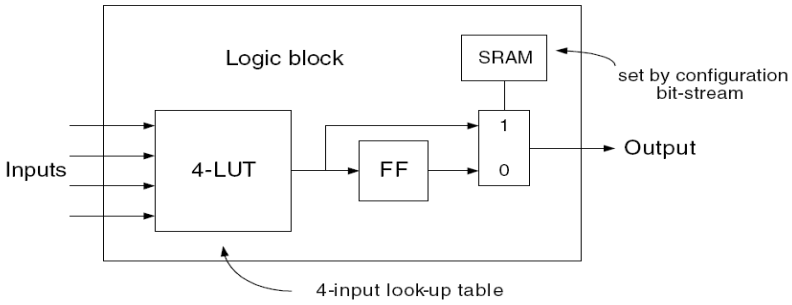


Figure 4.5 A basic FPGA logic block

4.3.3 Other Specifications of FPGA

All Xilinx FPGAs contain the same basic resources. These are Configurable Logic Blocks (CLBs), Input/Output Blocks(IOBs), Programmalbe Interconnections (PIs), RAM. The remaining resources are three-state buffers, global clock buffers, boundary scan logic and so on.

Basic difference of FPGA from CPLDs is to structurized based Configurable Logic Blocks (CLB) instead of a fixed array of gates. This is configurable and allows not only routing on the device, but also each logic block can be configured optimally (Figure 4.6)

The CLB has a Look-Up Table (LUT) that can be configured to give a specific type of logic function when programmed. There is also a clocked D-type flip-flop that allows the CLB to be combinatorial (non-clocked) or synchronous (clocked), and there is also an enable signal.

A typical FPGA will have hundreds or thousands of CLBs, of different types, on a single device allowing very complex devices to be implemented on a single chip and configured easily. Modern FPGAs have enough capacity to hold a number of 32-bit processors on a single device.

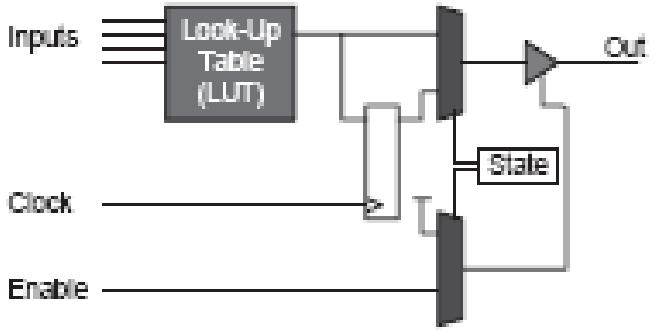


Figure 4.6 A basic FPGA configurable logic block

4.3.4 FPGA Design Flows

The FPGA design flow has several points in common with the semicustom ASIC design flow. Figure 4.7 presents a simplified FPGA design flow. The successive process phases of Figure 4.7 are described as :

- *Design Entry*: creation of design files using schematic editor or hardware description language (Verilog, VHDL, Abel).

- *Design Synthesis*: a process that starts from a high level of logic abstraction (typically Verilog or VHDL) and automatically creates a lower level of logic abstraction using a library of primitives.

- *Partition (or Mapping)*: a process assigning to each logic element a specific physical element that actually implements the logic function in configurable device.

- *Place*: maps logic into specific locations in the target FPGA chip.

- *Route*: connections of the mapped logic.

- *Program Generation*: a bit-stream file is generated to program the device.

- *Device Programming*: downloading the bit-stream to the FPGA.

- *Design Verification*: simulation is used to check functionalities.

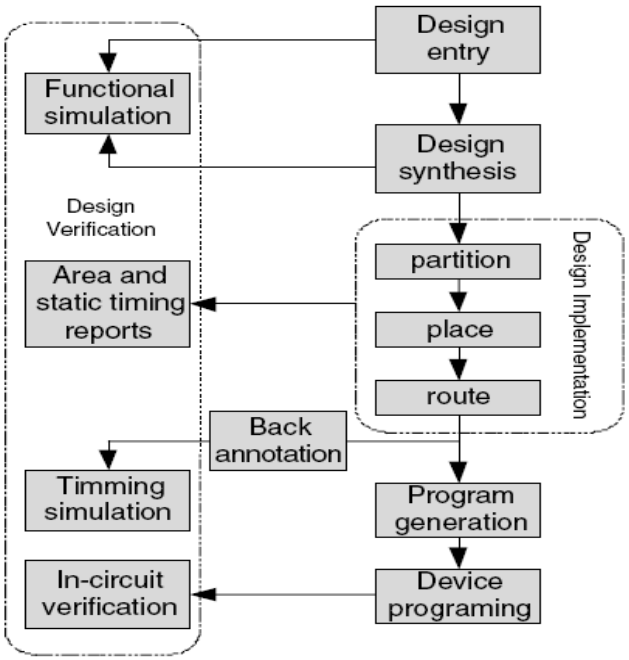


Figure 4.7 FPGA design flow

The simulation can be done at different levels. The functional or behavioral simulation does not take into account component or interconnection delays. The timing simulation uses back-annotated delay information extracted from the circuit.

Other reports are generated to verify other implementation results, such as maximum frequency and delay and resource utilization.

4.4 Hardware Description Languages (HDLs)

A hardware description language (HDL) is a computer language designed for formal description of electronic circuits. It can describe a circuit operation, its structure, and the input stimuli to verify the operation (using simulation). A HDL model is a text-based description of the temporal behavior and/or the structure of an electronic system. In contrast to a software programming language, the HDL syntax and semantics include explicit notations for expressing time and concurrencies, which are the primary attributes of hardware. Languages, whose only characteristics are to express circuit connectivity within a hierarchy of blocks, are properly classified as netlist languages. One of the most popular netlist formats and industry standards is EDIF, acronym for Electronic Data Interchange Format. These standards allow for data to be transferred, stored, and used from a database rather than formatted as an image file. (Deschamps, Bioul & Sutter, 2006)

Traditional programming languages such as C/C++ that are augmented with special constructions or class libraries, are sometimes used for describing electronic circuits. They do not include any capability for expressing time explicitly and, consequently, are not proper hardware description languages. Nevertheless, several products based on C/C++ have recently developed like Handel-C, System-C and other Java-like based such as JHDL or Forge to make development in a more usable way.

Using a proper subset of nearly any hardware description or software programming language, software programs called synthesizers can infer hardware logic operations from the language statements and produce an equivalent netlist of generic hardware primitives to implement the specified behavior. (Deschamps, Bioul & Sutter, 2006)

4.4.1 Development of HDLs

The two main players in this field are VHDL and Verilog. VHDL stands for VHSIC (very high speed integrated circuits) hardware description language. In the 1980s the U.S. Department of Defense and the IEEE sponsored the development of this hardware description language with the goal to develop very high-speed integrated circuits. It has now become one of the industry's standard languages used to describe digital systems. Around the same time another language, later called Verilog, with similarity to the C-language syntax was developed. In 1989, Cadence Company acquired the license and opened Verilog to the public in 1990. Both VHDL and Verilog are powerful languages that allow describing and simulating complex digital systems. Verilog is popular within Silicon Valley companies, while VHDL is used more by governments, in Europe, in Japan, and in most of the universities worldwide. (Deschamps, Bioul & Sutter, 2006) Most major CAD frameworks now support both languages. .

Another recognized HDL is ABEL (advanced Boolean equation language); it has been specifically designed for programmable logic devices. ABEL is less powerful than the formerly mentioned languages and is less used in the industry. Growth in complexity and strict time-to-market requirements for new system designs demand faster and simpler ways to describe system behaviors. The C++ and Java extensions, to support hardware description, seem to have a future because of the possibility to describe both hardware and software. The biggest challenge in this field is to have powerful synthesizers that can recognize and extract hardware and software from a previous non hardware-oriented code. (Deschamps, Bioul & Sutter, 2006)

4.5 FPGAs of the Project

In this section, two FPGA kits which are used during this project has been described based on their technical specifications : UP3 Education Kit and DE2-70 FPGA Board

4.5.1 UP3 Education Kit

4.5.1.1 General Description

The UP3 education kit is a low cost solution to prototype and develops new products. It has powerful characteristics for educational works. The board provides useful means, and widely used for prototyping, emulation and hardware and software development steps. The manufacturer of UP3 Educational Kit is and the board is supported with an Altera Cyclone FPGA(Figure 4.9). It is an Cyclone EP1C6Q240 in a 240-pin in a PQFP. Table 1 list the Cyclone device features.

Table 4.1 Cyclone EP1C6Q240 device features

Logic Elements	5980
RAM Blocks	20
Total Ram Bits	92160
PLLs	2
Maximum User I/Os	185

A designer can develop, prototype and test IP cores or design using HDLs such as VHDL or Verilog. The entire environment helps to implement any processor as well as any real time operating system on the kit. Other characteristic that helps designer is that a designer can simulate and test “C” cod also. The environment on the kit consists of standard interconnections, memory subsystem, multiple clocks, JTAG configuration, expansion headers for more capacity and some other interface features. DSP applications can be supported by the help of interfaces or by implementing DSP functions of the FPGA. Basically, the board contains of two main purposes. It can be used for prototyping and testing VLSI designs.

4.5.1.2 Features of UP3 Education Kit

The following are some of the features of the UP3 Education Kit. This kit includes low level capabilities with respect to other popular boards.

- Features an Altera EP1C6Q240 and EPCS1 configuration devices
- Supports intellectual property based (IP-Based) design both with and without a microprocessor
- USB 1.1 (Full Speed & Low Speed)
- RS 232 Port (Full Modem)
- Parallel port (IEEE1284)
- PS/2 Port
- VGA Port
- IDE (Integrated Drive Electronics)
- 128 Kbytes of SRAM (64K x 16)
- 2Mbytes of FLASH (1M x 16)
- 8MByte SDRAM (4M x 16)
- 2Kbytes of I2C PROM (Expandable)
- Supports multiple clocks like PCI clock, USB clock, IOAPIC clock and CPU clock
- JTAG and Active Serial download capability
- 5V Santa Cruz long Expansion Card Header provides 72 I/O for the development of additional boards providing various functionalities
- One user definable 4-bit switch block
- Four user definable push button switches, and one global reset switch
- Four user definable LEDs
- One 16x2 character display LCD Module
- I2C Real Time Clock

4.5.1.3 UP3 Board Diagram

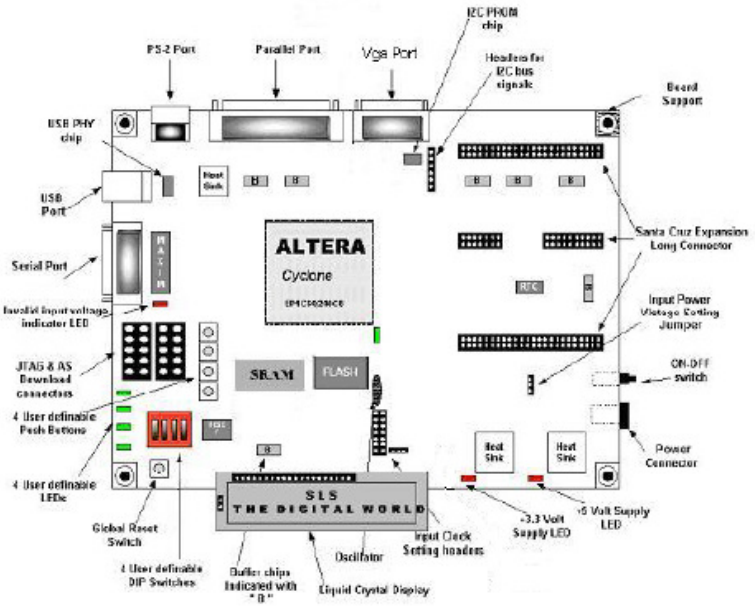


Figure 4.8 UP3 board top view

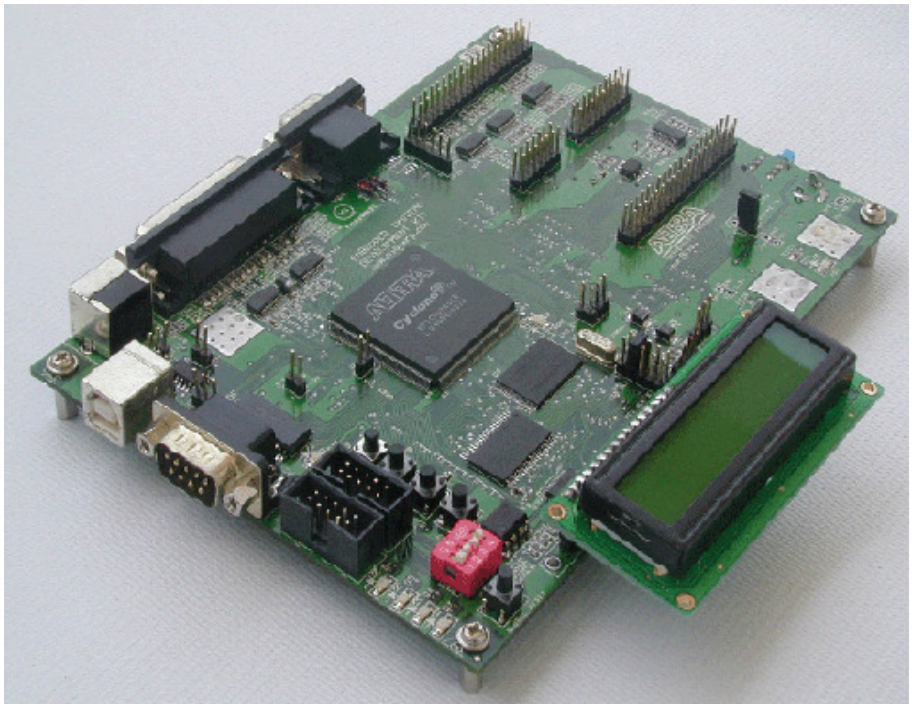


Figure 4.9 UP3 board components

4.5.2 DE2-70 FPGA Board

4.5.2.1 Layout and Components of DE2-70 Board

The DE2-70 board has many features that allow the user to implement a wide range of designed circuits, from simple circuits to various multimedia project. The following hardware is provided on the DE2-70 board:

- Altera Cyclone® II 2C70 FPGA device
- Altera Serial Configuration device - EPCS16
- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 2-Mbyte SSRAM
- Two 32-MByte SDRAM
- 8-Mbyte Flash memory
- SD Card socket
- pushbutton switches
- 18 toggle switches
- 18 red user LEDs
- 9 green user LEDs
- 50-MHz oscillator and 28.63-MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL/SECAM) and TV-in connector
- 10/100 Ethernet Controller with a connector
- USB Host/Slave Controller with USB type A and type B connectors
- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- IrDA transceiver
- 1 SMA connector
- Two 40-pin Expansion Headers with diode protection

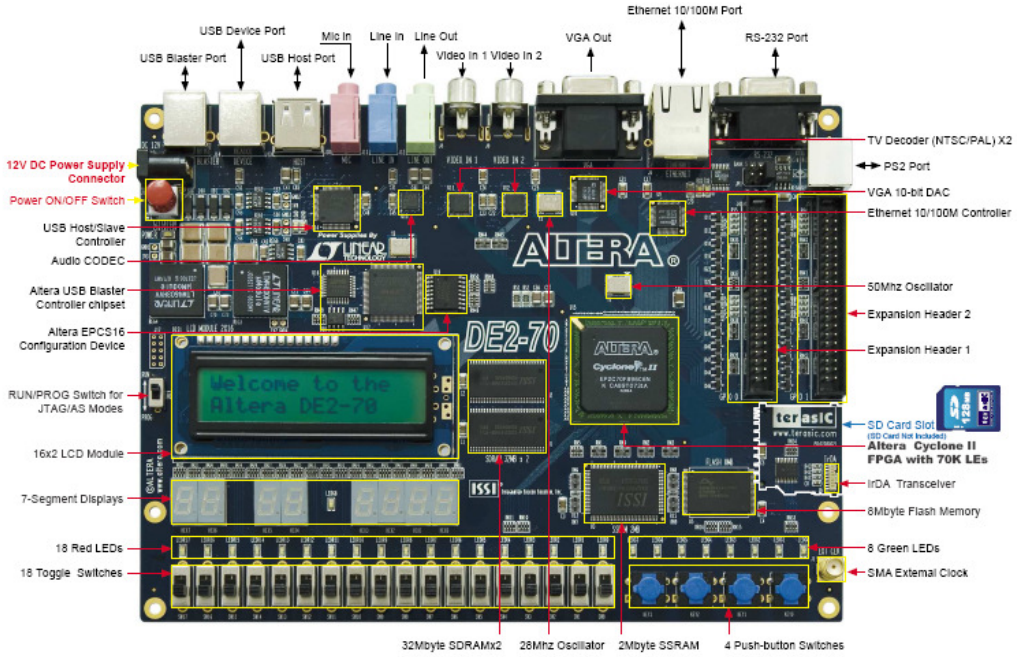


Figure 4.10 DE2-70 board components

4.5.2.2 Block Diagram of DE2-70 Board

Figure 4.11 gives the block diagram of the DE2-70 board. To provide maximum flexibility for the user, all connections are made through the Cyclone II FPGA device. Following items presents more detailed information about the blocks in Figure 4.11:

Cyclone II 2C70 FPGA

- 68,416 LEs
- 250 M4K RAM blocks
- 1,152,000 total RAM bits
- 150 embedded multipliers
- PLLs
- 622 user I/O pins
- FineLine BGA 896-pin package

Serial Configuration device and USB Blaster circuit

- Altera's EPCS16 Serial Configuration device
- On-board USB Blaster for programming and user API control
- JTAG and AS programming modes are supported.

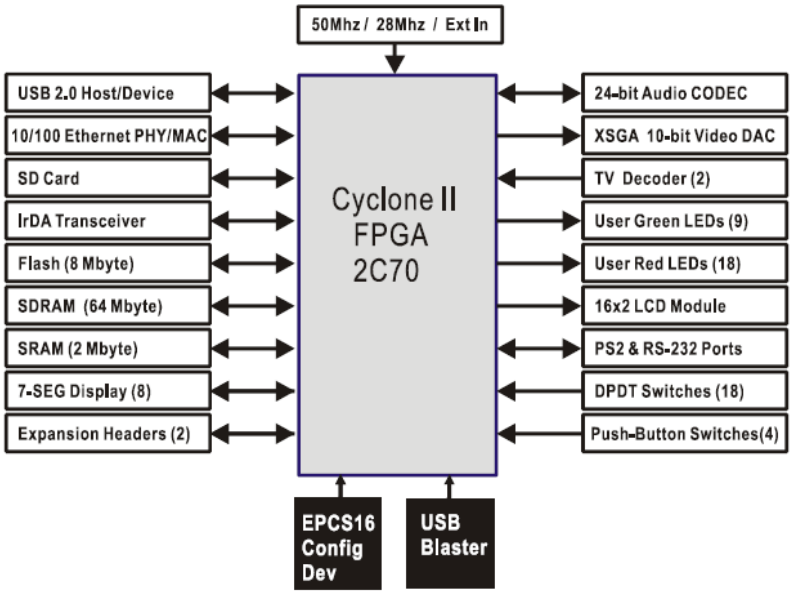


Figure 4.11 Block diagram of DE2-70 board

SSRAM

- 2-Mbyte standard synchronous SRAM
- Organized as 512K x 36 bits
- Accessible as memory for the Nios II processor and by the DE2-70 Control Panel

SDRAM

- Two 32-Mbyte Single Data Rate Synchronous Dynamic RAM memory chips
- Organized as 4M x 16 bits x 4 banks
- Accessible as memory for the Nios II processor and by the DE2-70 Control Panel

Flash memory

- 8-Mbyte NOR Flash memory
- Support both byte and word mode access

- Accessible as memory for the Nios II processor and by the DE2-70 Control Panel

SD card socket

- Provides SPI and 1-bit SD mode for SD Card access
- Accessible as memory for the Nios II processor with the DE2-70 SD Card Driver

Pushbutton switches

- 4 pushbutton switches
- Debounced by a Schmitt trigger circuit
- Normally high; generates one active-low pulse when the switch is pressed

Toggle switches

- 18 toggle switches for user inputs
- A switch causes logic 0 when in the DOWN (closest to the edge of the DE2-70 board) position and logic 1 when in the UP position

Clock inputs

- 50-MHz oscillator
- 28.63-MHz oscillator
- SMA external clock input

Audio CODEC

- Wolfson WM8731 24-bit sigma-delta audio CODEC
- Line-level input, line-level output, and microphone input jacks
- Sampling frequency: 8 to 96 KHz
- Applications for MP3 players and recorders, PDAs, smart phones, voice recorders, etc.

VGA output

- Uses the ADV7123 140-MHz triple 10-bit high-speed video DAC
- With 15-pin high-density D-sub connector
- Supports up to 1600 x 1200 at 100-Hz refresh rate

- Can be used with the Cyclone II FPGA to implement a high-performance TV Encoder

NTSC/PAL/ SECAM TV decoder circuit

- Uses two ADV7180 Multi-format SDTV Video Decoders
- Supports worldwide NTSC/PAL/SECAM color demodulation
- One 10-bit ADC, 4X over-sampling for CVBS
- Supports Composite Video (CVBS) RCA jack input
- Supports digital output formats : 8-bit ITU-R BT.656 YCrCb 4:2:2 output + HS, VS, and FIELD
- Applications: DVD recorders, LCD TV, Set-top boxes, Digital TV, Portable video devices, and TV PIP (picture in picture) display.

10/100 Ethernet controller

- Integrated MAC and PHY with a general processor interface
- Supports 100Base-T and 10Base-T applications
- Supports full-duplex operation at 10 Mb/s and 100 Mb/s, with auto-MDIX
- Fully compliant with the IEEE 802.3u Specification
- Supports IP/TCP/UDP checksum generation and checking
- Supports back-pressure mode for half-duplex mode flow control

USB Host/Slave controller

- Complies fully with Universal Serial Bus Specification Rev. 2.0
- Supports data transfer at full-speed and low-speed
- Supports both USB host and device
- Two USB ports (one type A for a host and one type B for a device)
- Provides a high-speed parallel interface to most available processors; supports Nios II with a Terasic driver
- Supports Programmed I/O (PIO) and Direct Memory Access (DMA)

Serial ports

- One RS-232 port
- One PS/2 port
- DB-9 serial connector for the RS-232 port
- PS/2 connector for connecting a PS2 mouse or keyboard to the DE2-70 board

IrDA transceiver

- Contains a 115.2-kb/s infrared transceiver
- 32 mA LED drive current
- Integrated EMI shield
- IEC825-1 Class 1 eye safe
- Edge detection input

Two 40-pin expansion headers

- 72 Cyclone II I/O pins, as well as 8 power and ground lines, are brought out to two 40-pin expansion connectors
- 40-pin header is designed to accept a standard 40-pin ribbon cable used for IDE hard drives
- Diode and resistor protection is provided.

CHAPTER FIVE

THE REAL-TIME FACE AND SPEECH RECOGNITION SYSTEM DESIGN

5.1 Introduction to Face Recogniton System

This chapter describes how the real-time face recognition system works. The Principal Component Analysis (PCA) is a powerful method that is used for extracting feature matrices. Basically, both database matrices and test vector is analyzed by using PCA and result vectors are compared to find the best match in face recognition system.

At the start of the project, some minor projects such as led blinking, one byte data, a vector of data and a matrix of data transmission to FPGA etc. was developed by using UP3 Educational Kit to familiarize with the FPGA kits. However, this kit couldn't provide sufficient memory and logic elements to handle this project. Thus, new and more complicated FPGA, DE2-70 FPGA board replaced the UP3 Educational Kit. In the parallel, all development steps were renewed due to basic properties of the new FPGA. Detailed information about UP3 Educational Kit and DE2-70 FPGA board are in Chapter 3. In this chapter, the examples with both UP3 and DE2-70 kit are demonstrated to help the FPGA beginners.

In face recognition project, the system starts by analyzing of face images which are used to construct face database. If images were collected in real-time by taking photos of subjects, pre-processing step would be required. Since image files were taken from a database, this step was skipped. These images are resized and rearranged to send to FPGA device. The resultant matrix whose column vectors represents face image, is constructed. While the available FPGA is UP3 Educational Kit, PCA runs in MATLAB and data matrix which is analyzed by PCA, is sent to FPGA by Universal Asynchronous Receiver/Transmitter (UART) to be analyzed. The similar procedure is applied to test face image and the resultant data is sent to FPGA, too. But, some problems described in the subsequent sections blocks the

further development in UP3 Education Kit. This thesis describes the reasons of blocking and switching to other board, DE2-70 FPGA board definitely. On the other hand, while the available FPGA is DE2-70, usage strategy of resources changes. At that time, the whole PCA and comparing algorithm runs in DE2-70 FPGA based on powerful resources of it. When DE2-70 FPGA board is development environment, the duty of MATLAB is to collect the database and test face images and to transmit them to FPGA after resizing. Figure 5.1 shows the block diagram of face recognition system.

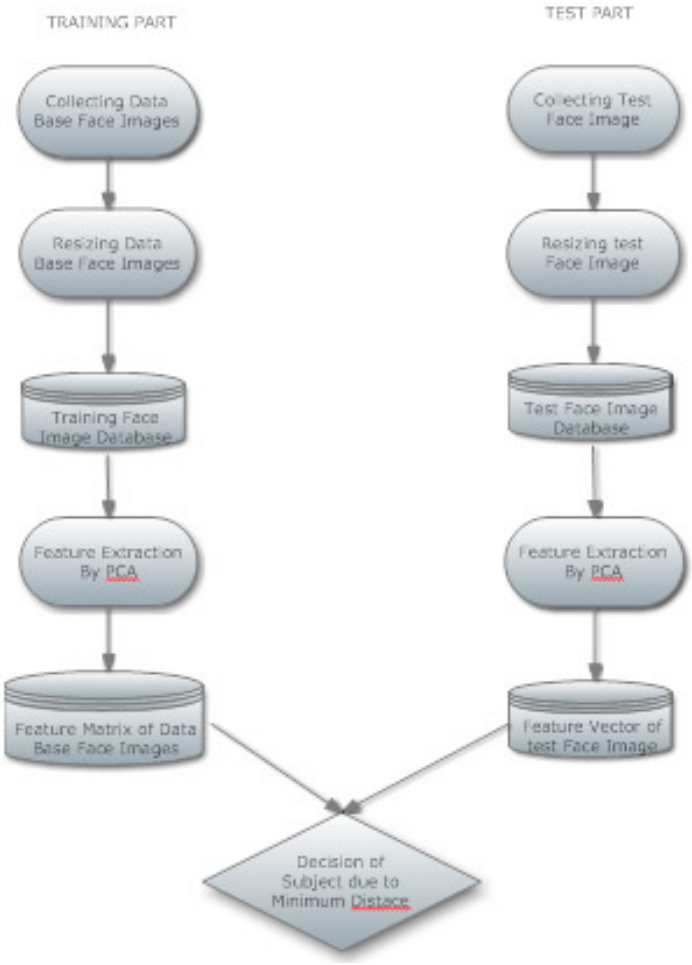


Figure 5.1 Block diagram of face recognition system

5.2 Introduction to Speech Recogniton System

In addition to face recognition system, another powerful recognition technique, speech recognition, has been added to whole system. Speech recognition in the system is developed based on the feature extraction from speech signals using Fourier Transform and comparison of them. This chapter also describes the working of speech recognition system using DE2-70 FPGA.

Speech recognition works in a similar way with face recognition system. But, speech signals are more sensitive to noise factor. So, experimental works with speech signals also requires pre-processing steps to remove noise effects and to increase the quality of the signal.

Speech recognition system starts by data acquisition step for both training and test phases. As mentioned above, pre-processing algorithm runs on speech signals to eliminate noise factor. The other mission of pre-processing is to remove the insignificant part of recorded speech signal in order to reduce the recognition performance. Feature matrix including speech signals of subjects in database are transferred to DE2-70 FPGA via UART. Then, Fourier Transform is applied to whole matrix and transformed data is saved to flash memory. After these applications, DE2-70 FPGA is ready to accept test speech signal to recognize. Data acquisition and pre-processing algorithms runs on the test speech signal, too. Test vector sent from MATLAB via UART is processed by Fourier Transform algorithm. Last result is obtained by comparing transformed form of feature matrix and transformed form of test vector and is shown to end user.

5.3 Source Development Tools : Quartus and Nios

The Altera Quartus design software provides a complete design environment. Thus specific designs can be easily adopted to design requirements. It offers an environment for system-on-a-programmable-chip (SOPC) design. The Quartus II

software includes solutions for all FPGA types. Design Flow for Quartus II is shown in Figure 5.2.

Nios II is a functional processor that employ RISC architecture by using a set of pre-defined instructions. It is used for FPGA target designs developed by Altera family. It includes a 32-bit load/store, Wishbone-compliant processor. It is suitable for a wider range of embedded computing applications, from DSP to system-control. There are 3 types of configurable 32-bit embedded processors for Altera FPGAs. These are fast (/f core) optimized for the highest performance, economy(/e core) optimized for the smallest size and standard(/s core) balanced for performance and size. Nios II includes a rich portfolio of embedded periheral cores like UART, direct memory acces (DMA), ethernet etc. Completed embedded development suite which includes compiler, debugger, Integrated Development Environment (IDE), drivers etc. makes it more suitable for the whole FPGA design.

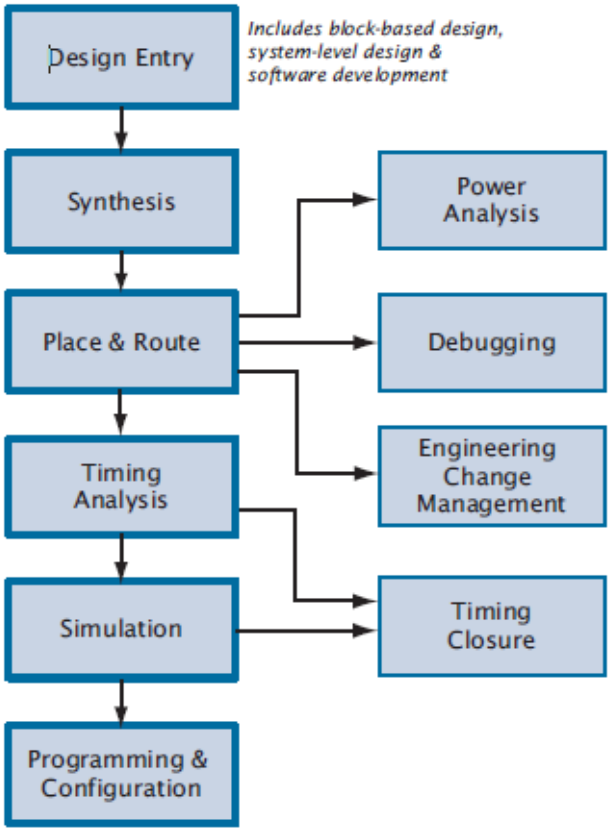


Figure 5.2 Design flow for Quartus II (Altera, 2007)

The Nios II Integrated Development Environment (IDE) provides the tools to accomplish software development tasks such as editing, building, and debugging programs. A PC, an Altera FPGA, and a download cable are enough to write programs for and communicate with, any Nios II processor system. Nios II IDE includes a compiler for C and C++. It makes Nios II more attractive for code development since C and C++ are more generic programming languages for many developers.

5.4 The First Applications With FPGA

In this section preliminary experiments implemented with UP3 kit has been introduced.

5.4.1 Led Blinking

The project development steps started with implementing some minor projects. It would be useful to learn the working strategy of the device and to experience the hardware development language for FPGA beginners.

VHDL is one of two main hardware development languages. The first project was led blinking. The code part below was developed to blink the leds. But code development in Quartus need some pre-adjustments such as Pin Assignment, compiler and simulator adjustments. As an example, led blinking project was based on assignment of 4 ON/OFF switches on the UP3 Educational Kit and observing hexadecimal number determined by the swithces on the seven-segment display correctly. Hence required switches, seven-segment display pins, clock, reset of UP3 Educational Kit were assigned to the corresponding pins of FPGA carefully. During development of led blinking project, a problem was observed. Clock signal was connected to global clock signal of UP3 Educational Kit, and reset signal was connected to global reset pushbutton. At this stage, although clock signal was same with global clock signal, reset signal wasn’t same with the of global reset pushbutton.

Hence, connection of reset signal was changed with one of DIP switches. It worked after change properly. Source code developed in VHDL, is shown in Figure 5.3.

```
  LIBRARY IEEE;
  USE IEEE.STD_LOGIC_1164.ALL;

  ENTITY seven_segment IS
    PORT (bin: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         leds: OUT STD_LOGIC_VECTOR(1 TO 7));
  END seven_segment;

  ARCHITECTURE Behavior OF seven_segment IS
  BEGIN
    leds <= "0000001" WHEN bin="0000" ELSE
         "1001111" WHEN bin="0001" ELSE
         "0010010" WHEN bin="0010" ELSE
         "0000110" WHEN bin="0011" ELSE
         "1001100" WHEN bin="0100" ELSE
         "0100100" WHEN bin="0101" ELSE
         "0100000" WHEN bin="0110" ELSE
         "0001111" WHEN bin="0111" ELSE
         "0000000" WHEN bin="1000" ELSE
         "0001100" WHEN bin="1001" ELSE
         "0001000" WHEN bin="1010" ELSE
         "1100000" WHEN bin="1011" ELSE
         "0110001" WHEN bin="1100" ELSE
         "1000010" WHEN bin="1101" ELSE
         "0110000" WHEN bin="1110" ELSE
         "0111000" WHEN bin="1111" ELSE
         "1111111";
END Behavior;
```

Figure 5.3 VHDL code body which blinks the leds

5.4.2 UART Implementation

5.4.2.1 Theory of UART

Serial communication is the essential to computers. Since there are two main environments in the project, MATLAB and FPGA, communication was one of the basic parts of the project. Communication between MATLAB and FPGA was over UART which allows the computer to communicate FPGA with low speed.

Asynchronous communication is performed between two devices if operating clock of them are different. Since there is no guarantee that the clocks of the

communicating devices will have the exact frequency. To overcome this problem, additional synchronous bits are added to asynchronous communication to maintain integrity. Asynchronous communication protocol includes a a start bit in the beginning of data. It indicates that a data stream is beginning. Also, the end of data stream is followed by a stop bit which says the receiver that the data stream is about to end. At that time, there is no communication until the next start bit. Figure 5.4 shows the serial data stream of UART.

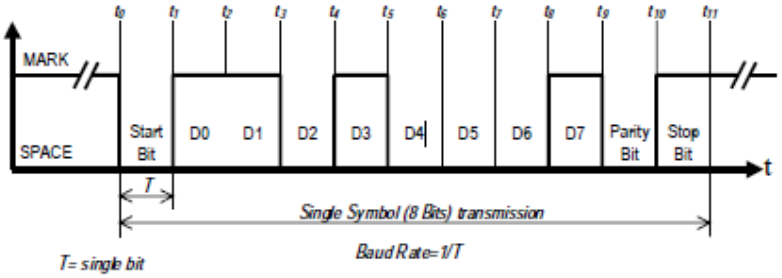


Figure 5.4 Serial data stream in UART

The UART circuit enables a CPU to communicate with other external devices. The interface between the CPU and the UART is usually byte parallel and can be synchronous. The transmission properties of the UART such as parity control, number of symbol bits, number of stop bits etc. can be programmable by changing a control register. Figure 5.5 shows the simplified block diagram of an UART circuit.

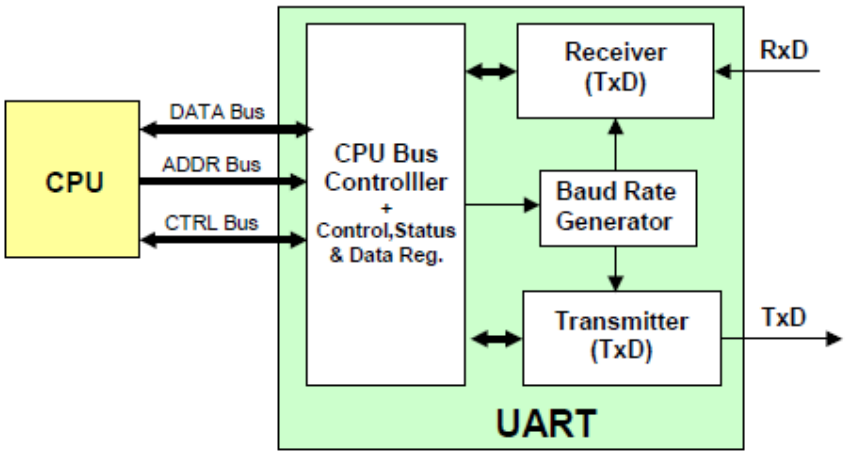


Figure 5.5 Simplified block diagram of an UART circuit

- *CPU Bus Controller:* It provides the parallel data I/O interface to the local processor bus. It create a control signal to enable the CPU to access onto the data, status and control register of the UART circuit.

- *Baud Rate Generator*: It generates transmit and receive bit timing by creating a periodic pulse which observes the baud rate of the UART. It is programmable.

- *Transmitter*: It is responsible for the serial data transmitting which the CPU writes to TxD Hold Register. If the TxD Hold Register is full of data, it loads the data onto the transmit register via local data bus. Next, it sets a signal, "Transmit Data Read", to inform the CPU about the transmitted data.

- *Receiver:* It samples the data bits sequentially whenever the start bit is detected. It receives until it detects the stop bit. Next, received data is written to local data bus and informs the CPU about the data arrival by setting a signal, "Receive Data Read".

5.4.2.2 UART Implementation Coding in VHDL

Development of UART via VHDL started by initializing of variables which was used for porting of transmitter and receiver. Then, component declarations for *uart_transmitter* and *uart_receiver* were done. The two component declarations (for *uart_transmitter* and *uart_receiver*) had to match the corresponding entity declarations exactly with respect to the names, order and types of the ports. It was an important point in the mapping of ports, *uart_transmitter* and *uart_receiver.*

In the body of architectures for *uart_transmitter* and *uart_receiver*, there was a baud generator code segment, named *baud_gen* whose duty was to generate baud. Baud generator works by counting until a constant based on the clock of the system. When it reaches to constant, a variable is set to 1 and counter is reseted. When it reaches to constant again, variable is set to 0 at that time and counter is reseted. This is a periodical sequence and creates a meaningful baud for the system. VHDL coding

of baud generator is shown in Figure 5.6. *dividenum* in the below coding was defined to evaluate a baud rate of 115200 which was assumed as an optimum value.

```
NEWBAUD : process (CLOCK)

begin
if(CLOCK = '1' and CLOCK'event) then
        if (counter = dividenum) then
                        baud_rate <= '1';
                        counter <= (others => '0');
        elsif (tx_ck_enable_baud = '1') then
                        baud_rate <= '0';
                        counter <= counter + 1;
        else
                        baud_rate <= '0';
                        counter <= (others => '0');
        end if;
end if;
```

Figure 5.6 Code segment of baud generator with baud rate of 115200

UART transmitter worked by transferring a bit in each baud basically. There were 10 states. 8 of 10 states were for 8-bit data, 1 of 10 states was for start bit and 1 of 10 states was for stop bit. Current state was shifted one by one while *txd* was loading and transmitting the corresponding bit in each baud. When 8 bit data was transmitted and current state was set to last state, S9, *eot* was set to 1 which informed the CPU about the end of transmitting. VHDL coding of UART transmitter entity and UART transmitter state machine are shown in Figure 5.7 and Figure 5.8 ,respectively.

```
entity uart_transmitter is
   port (clock   : in  std_logic;
         tx_data  : out  std_logic;
         eot     : out  std_logic;
         data     : in  std_logic_vector(7 downto 0);
         write_tr : in std_logic
        );
end uart_transmitter;
```

Figure 5.7 VHDL coding of UART transmitter entity.

```
TX_STATE_MACHINE: process(current_state,tx_enable,datain,data)

        begin
```

Figure 5.8 VHDL coding of UART transmitter state machine

```
case current_state is

        when S0 =>
                datain <= data;
                eot <= '0';
                if (tx_enable = '1') then
                        txd <= '0';
                        next_state <= S1;
                        TRANSMITTING<='1';
                else
                        txd <= '1';
                        TRANSMITTING<='0';
                        next_state <= S0;
                end if;
        when S1 =>
                txd <= datain(0);
                eot <= '0';
                TRANSMITTING<='1';
                next_state <= S2;
        when S2 =>
                txd <= datain(1);
                eot <= '0';
                TRANSMITTING<='1';
                next_state <= S3;
        when S3 =>
                txd <= datain(2);
                eot <= '0';
                TRANSMITTING<='1';
                next_state <= S4;
        when S4 =>
                txd <= datain(3);
                eot <= '0';
                TRANSMITTING<='1';
                next_state <= S5;
        when S5 =>
                txd <= datain(4);
                eot <= '0';
                TRANSMITTING<='1';
                next_state <= S6;
        when S6 =>
                txd <= datain(5);
                eot <= '0';
                TRANSMITTING<='1';
                next_state <= S7;
        when S7 =>
                txd <= datain(6);
                eot <= '0';
                TRANSMITTING<='1';
                next_state <= S8;
        when S8 =>
                txd <= datain(7);
                eot <= '0';
                TRANSMITTING<='1';
                next_state <= S9;
```

Figure 5.8 (continued)

```
                        when S9 =>
                                txd <= '1';
                                eot <= '1';
                                TRANSMITTING<='1';
                                next_state <= S0;
                        when others =>
                                null;

                end case;

  end process TX_STATE_MACHINE;
```

Figure 5.8 (continued)

UART receiver worked similar to UART transmitter. It is as follows that, there were 10 states again. 8 of 10 states were for 8-bit data, 1 of 10 states was for start bit and 1 of 10 states was for stop bit. Current state was shifted from S0 to S9, periodically. When the current state was S9, *eor* was set to 1 which indicated the end of receiver. Also, received data was loaded to *data_out* logic vector and *data_rdy* flag was set to 1. Otherwise, both of *eor* and *data_rdy* flags were set to 0 and the whole data was collected in *data* signal. The ports in a component declaration of *uart_transmitter* and *uart_receiver* are shown below. VHDL coding of UART receiver entity and UART receiver state machine are shown in Figure 5.9 and 5.10.

```
entity uart_receiver is

   port (gl_clock           : in  std_logic;
         rx_data            : in  std_logic;
         rx_enable          : in  std_logic;
         rx_ck_baud_enable  : in  std_logic;
         data_out           : out  std_logic_vector(7 downto 0);
         data_rdy           : out  std_logic;
         eoro               : out std_logic
         );

end uart_receiver;
```

Figure 5.9 VHDL coding of UART receiver entity.

```
RXD_STATE_MACHINE:process(start,current_state,rx_enable,receiving)

       begin
```

Figure 5.10 VHDL coding of UART transmitter state machine

```
        if (rx_enable = '0') then
              if (start = '1' or receiving = '1') then
                    case current_state is
                          when S0 =>
                                eor <='0';
                                if (start = '1') then
                                      next_state <= S1;
                                      receiving <= '1';
                                else
                                      next_state <= S0;
                                      receiving <= '0';
                                end if;
                          when S1 =>
                                receiving <= '1';
                                eor <= '0';
                                next_state <= S2;
                          when S2 =>
                                receiving <= '1';
                                eor <= '0';
                                next_state <= S3;
                          when S3 =>
                                receiving <= '1';
                                eor <= '0';
                                next_state <= S4;
                          when S4 =>
                                receiving <= '1';
                                eor <= '0';
                                next_state <= S5;
                          when S5 =>
                                receiving <= '1';
                                eor <= '0';
                                next_state <= S6;
                          when S6 =>
                                receiving <= '1';
                                eor <= '0';
                                next_state <= S7;
                          when S7 =>
                                receiving <= '1';
                                eor <= '0';
                                next_state <= S8;
                          when S8 =>
                                receiving <= '1';
                                eor <= '0';
                                next_state <= S9;
                          when S9 =>
                                receiving <= '1';
                                eor <= '1';
                                next_state <= S0;
                          when others =>
                                null;
                     end case;
              end if;
        end if;

end process RXD_STATE_MACHINE;
```

Figure 5.10 (continued)

```
RXD_SHIFT : process (rec_baud_clock,rx_enable)
        begin
        if (rx_enable = '0') then
                if (rising_edge(rec_baud_clock)) then
                        if (eor = '0') then
                                data <= rx_data & data(7 downto 1);
                        end if;
                end if;
        end if;
        end process RXD_SHIFT;
```

Figure 5.10 (continued)

Parallel working mechanism of FPGA was new to us, UART implementation by VHDL was also useful to understand this mechanism well in UP3 Educational Kit.

5.4.3 The First Projects for Data Comparison and Data Communication

As mentioned before, there are two main environment in the project, MATLAB and FPGA. UART provides the asynchronous communication between them. Communication in MATLAB was done by the help of MATLAB serial communication toolbox functions. Basically, four functions run sequentially in MATLAB. They were *serial* function to construct a serial port object associated with an existing port, *fopen* to connect the serial port object to FPGA, *fwrite* to write the feature matrix and other data to FPGA which was already connected to defined serial port object and *fclose* to disconnect the serial port object from FPGA. Baud rate was 115200 bits/second which was optimum speed for serial communication. A serial port object was defined in MATLAB to communicate with FPGA. Next, it was opened by *fopen.* If data transmission was required, *fwrite* was used. Also, if data reception was required, *fread* was used over the opened serial port object.

The first data communication project was to transmit a byte data from FPGA. Simulator tool of Quartus II was a good choice to try this communication. So, VHDL code was developed and simulated in Quartus II to see the result. *uart_transmitter* and *uart_receiver* components were still available. Apart form that, signals were defined in the main architecture as shown below. Initialisations of constants and por mapping of transmitter block are shown in Figure 5.11 and Figure 5.12, respectively.

```
architecture behv of main is

--signal datainput: std_logic_vector(7 downto 0):=X"44";
signal datainput: std_logic_vector(7 downto 0):=X"44";
signal transmitselect : std_logic := '1';
signal received_data : std_logic_vector(7 downto 0):=X"00";
signal receive_enable : std_logic := '0';
signal write_main : std_logic := '0';
SIGNAL COUNTDV : STD_LOGIC_VECTOR(23 DOWNTO 0):=X"000000";
signal testwrite : std_logic := '1';
```

Figure 5.11 Initialisation of constants for VHDL coding in UP3 education kit

```
START_ONLY_TRANSMIT: uart_transmitter port map (CLK,TX_OUT,
EOT_LED,datainput,write_main);
```

Figure 5.12 Port mapping of transmitter block

Until simulation step, there were some steps such as compilation and generation of function simulation netlist. As shown in the definition of signals, *datainput* which would be transmitted, was 46H. Pins ported to main were selected to observe. As shown in Figure 5.13, *TX_OUT,* bit which carried the elements of transmitted data respectively, was loaded correctly. 8 states of *TX_OUT* represented the transmitted data in binary format correctly. Figure 5.13 shows the simulation result when datainput is “46h”. The VHDL code of “one byte data transmitter” is in the Appendix with the folder name of “5_4_3_UART_Transmitter”.

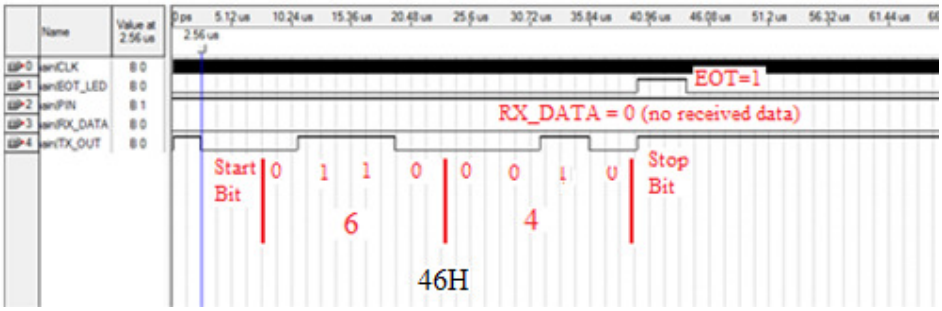


Figure 5.13 Simulation result of the project of “one byte data transmitter”

The second data communication project was an advanced version of the first one. In addition to data transmission, data reception was added to project. Since data reception couldn't be simulated in Quartus II, real environment, UP3 Education Kit and MATLAB, was used. As mentioned before, MATLAB data communication toolbox helped us to transmit a byte of data. A byte of data, sent from MATLAB, was received and transmitted in the next step. Transmitted data was published to hyper terminal screen of computer. Hyper terminal is a terminal program which can be used to set up a connection for data transfer between two computers (such as your desktop computer and a portable computer) using the serial ports. The main additional point of this project was to add port mapping of *uart_receiver*. Also, a few new signals were defined to save the end of receiver, receiver enable state etc. Figure 5.14 shows VHDL code for port mapping of transmitter and receiver blocks.

```
START_ONLY_RECEIVER: uart_receiver port map (CLK,RX_DATA,
receive_enable,reset_receiver,received_data,DATA_RDY,signal_EOR);

datainput <= received_data
EOR <= signal_EOR

START_ONLY_TRANSMIT: uart_transmitter port map (CLK,TX_OUT,
EOT_LED,datainput,write_main);
```

Figure 5.14 Port mapping of transmitter and receiver blocks

Port mapping of *uart_receiver* included clock, whole received data, bit-by-bit received data, data ready signals etc. At the end of this project, data sent from MATLAB was received by FPGA. Next, received data was transmitted immediately. The result was represented on the screen of hyper terminal and it was usually correct. The figure below shows the hyper terminal snapshots when transmitted data block contains "a,h,2,8,4". The VHDL code of "one byte data receiver" is in the Appendix with the folder name of "5_4_3_UART_Receiver". Figure 5.15 shows the test screen in Hyper Terminal and Hyper Terminal communication settings. On the other hand, the same figure shows the specifications of hyper terminal. Since the optimum transmission speed was 115200 bits per second for FPGA device, baud rate was set to 115200. Other parameters were data bits which was set to 8, stop bit which was set

to 1 and parity control which was set to None. The screen of hyper terminal shows each data for two times and next to each other. The first character of each double corresponds to data which FPGA received, and the second character of each double corresponds to data which FPGA transmitted.

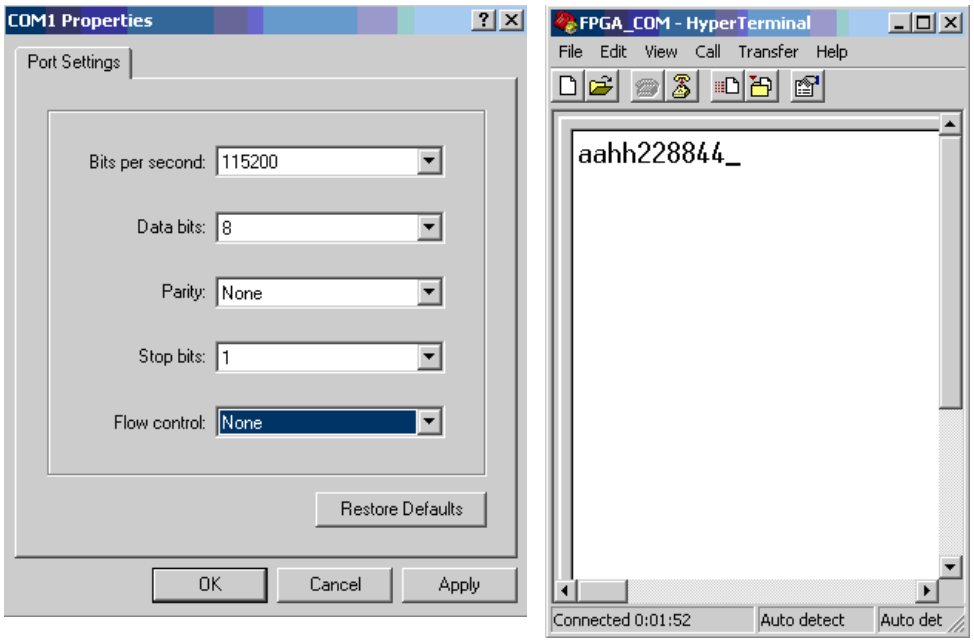


Figure 5.15 Hyper terminal screen which prints transmitted and received data respectively.

The next development about data communication was related to transmission of an array. It was planned to use in the face recognition project. Because, data matrix constructed by PCA would be an array and it would be sent to FPGA via UART. The main idea of array transmission was to transmit an element of the array one by one. Hence an array was constructed with 4 element. In a process, the corresponding element of array was selected and it was sent to FPGA like the project of transmission of a byte data.

The important point of all project was to use the same VHDL files for UART transmitter, UART receiver, baud generator. Only change was at the definition of port mapping of *uart_transmitter*, *uart_receiver*. As shown in Figure 5.17, each range between two sequential high states of *EOT_LED,* the binary format of the element of data array(Figure 5.16) was transmitted correctly. The VHDL code of

"data array Transmitter" is in the Appendix with the folder name of "5_4_3_UART_Array_Transmitter".

```
type data_array is array (3 downto 0) of std_logic_vector (7
     downto 0);

signal data_block : data_array := ( 0 => x"43",  1 => x"44",
                                    2 => x"45",  3 => x"46" );
```

Figure 5.16 Array definition

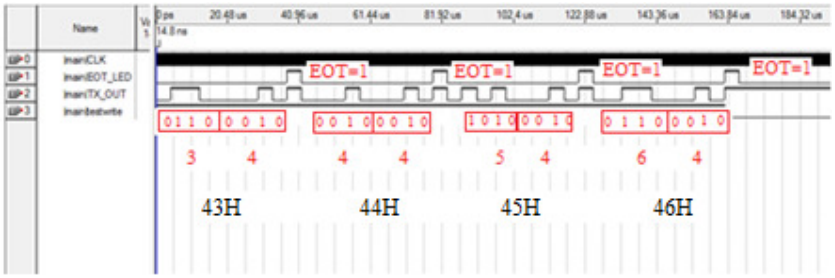


Figure 5.17 Simulation Result of the project of "data array transmitter"

The next project was related to data comparison in FPGA by VHDL. Two arrays in RAM type were created for this purpose. One of them, called as *data_block*, had 16 elements while the other array, called as *test_block*, had 4 elements. *data_block* was divided into 4 parts. Each of 4 parts was compared to *test_block* one-by-one. Total difference of each of 4 parts was saved. The minimum of these differences corresponded to the nearest part of *data_block.* Since only transmitter part of UART was available, simulation of Quartus II would be more practical. Whenever elements of *data_block* and *test_block* were changed, identity of the nearest part of *data_block* was signalized on the simulation tool. The correct identity of 4 parts was on the screen of simulation tool. Figure 5.18 shows how the data and test blocks are defined in VHDL coding format. As shown in Figure 5.19, *signal_c* which loaded the closest group number, is equal to 00000010 since the second group was the closest. The VHDL code of "data comparison" is in the Appendix with the folder name of "5_4_3_Internal_Database_And_Test".

```
type RAM is array (2 ** ADDRESS_WIDTH - 1 DOWNTO 0) ) of
std_logic_vector (DATA_WIDTH - 1 DOWNTO 0);

signal data_block : RAM := (0      => x"04",1      => x"04",
                            2      => x"04",3      => x"04",
                            4      => x"02",5      => x"02",
                            6      => x"02",7      => x"02",
                            8      => x"03",9      => x"03",
                           10      => x"03",11      => x"03",
                           12      => x"05",13      => x"05",
                           14      => x"05",15      => x"05");

type RAMtest is array (ADDRESS_WIDTH - 1 DOWNTO 0) OF of
std_logic_vector (DATA_WIDTH - 1 DOWNTO 0);

signal new_block : RAMtest := (0      => x"01",1      => x"01",
                               2      => x"01",3      => x"01");
```

Figure 5.18 Definition of internal data and test blocks

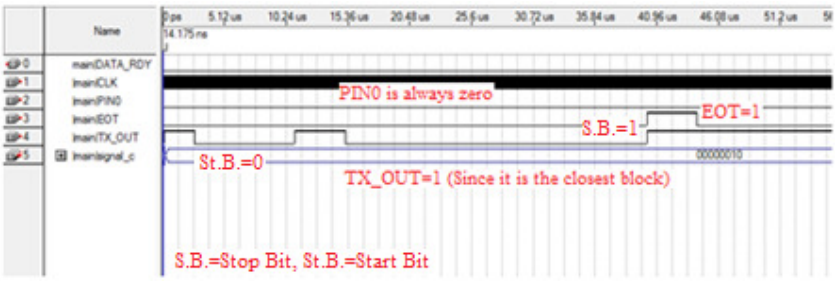


Figure 5.19 Simulation result of the project of "data comparison"

Combination data transmission and data comparison projects mentioned above, was integrated in the next project. Data transferred from MATLAB to UP3 Education Kit was loaded to the logic elements of kit. Then, a smaller size data block was transferred to UP3 Education Kit and it was compared to the each block of the previous data block like in the previous project. The result was transmitted to MATLAB to inform the user. MATLAB snapshot which showed the transferred data blocks and result is represented in Figure 5.20. In this example, the array which is sent initially by *fwrite*, is compared to the array which sent secondarily by *fwrite*. When the first array is divided into 4 parts and each of them is compared to the second array, the closest part is the 3rd part. So, the result which FPGA calculates is correct. Since the calculation time of FPGA is fast enough, the result is displayed on screen at the end of the timeout value of 10 seconds. Also, baudrate is set to 115200

bits/s while a serial object, s2, is defined. The VHDL code of the integrated form of data comparison and transmission systems is in the Appendix with the folder name of "5_4_3_Database_and_Test_sent_from_MATLAB".

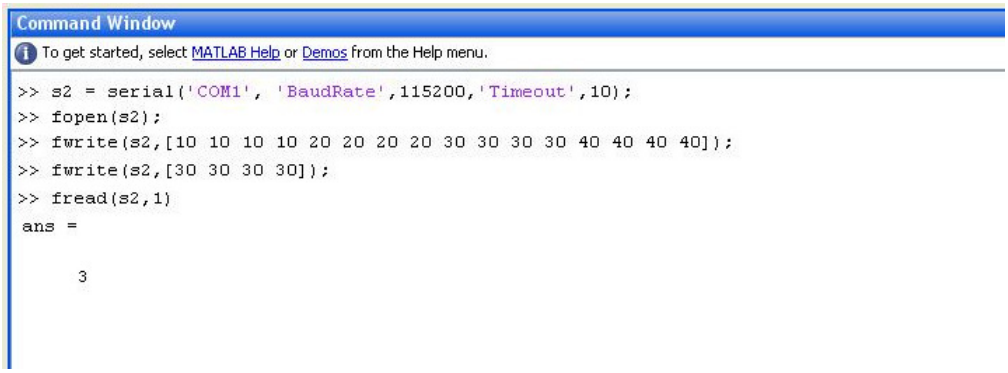


Figure 5.20 Data transmission and results in MATLAB

5.5 Face Recognition System By UP3 Education Kit

System development in UP3 Education Kit started by data matrix creation in MATLAB to send to UP3 Education Kit via UART. Face images in "The ORL Database of Faces" were used. However, they had a quite big size to be processed. Figure 5.21 shows some face images from "The ORL Database of Faces".

Figure 5.21 Face image samples from database

Hence face image files were resized based on the sizes which were defined in the beginning. Then a new matrix was created which had only the training images. Face

images from 6 subjects were used. Each subject had 8 different images. MATLAB source code developed for UP3 Education Kit is in the Appendix with the folder name of "5_5_PCA_MATLAB".

FaceRec.m is the name of the main MATLAB function in this implementation. The real size of the face images was 112 x 92. MATLAB function, *imresize*, was arranged to reduce the size of each image to 48 x 48 in *FaceRec.m.* In order to analyze image files, the matrix which had training image files was rearranged. Whole feature matrix where each column represented a resized image, was created and called as *Xbasis*. The MATLBA function which created *Xbasis* was *Loadpop.m.* The size of feature matrix was 2304 x 48 where 2304 corresponded to transformed form of images with 48 x 48 and 48 corresponede to total image number. *Makebasis.m* function was implemented at that step to create the "face space", i.e. the basis of the space created throught the eigenvectors of the covariance matrix of the *Xbasis*. *Makebasis.m* function started by calculating the covariance matrix of *Xbasis* and continued by evaluating the covariance matrix created before it. At the end, an eigenvector matrix sorted with respect to the eigenvalues, was obtained succesfully. Since the first 3 eigenvectors contains the most of the meaningful information, there was no need to use the remaining part of the eigenvector matrix. So they were eliminated. The main purpose of these calculations was to create a face space basis with available face images. In order to transform of the face images to new space, *Xbasis* which contained the face images in column vectors, were multipled by non-eliminated part of the eigenvector matrix and the result matrix was assigned to *Ytrain*. The similar processes were applied to only one test image to evaluate a test matrix and it was assigned to *Ytest* vector. Since only one image was used to create *Ytest* vector, the size of *Ytest* vector was 2304 x 1.

In the compilation step of VHDL code with above descriptions, an error occurred. Error message told us that the design couldn't fit the UP3 Education Kit. Since size of feature matrix is extremely high, the size of feature matrix was reduced to check whether the complier error would disappear or not. After reducing the size from 2304 x 48 to nearly 5000 elements totally, compliation finished without no error. But

compilation report stated that %98 percent logic elements of FPGA was used. Compliation report created is shown in Figure 5.22.

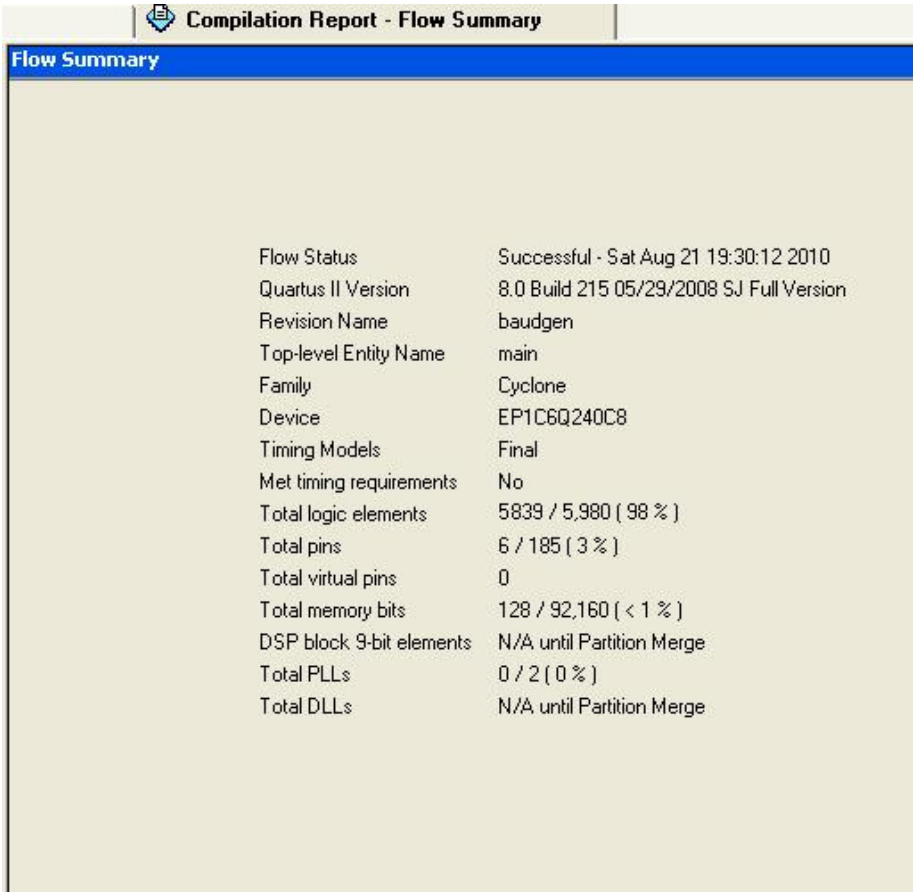


Figure 5.22 Compilation report of UP-3 education kit which was constructed when all database images are used

Memory management of UP3 Education Kit FPGA was based on using FPGA SRAM Cells as memory elements. Since the SRAM Cells of UP3 Education Kit were the memory elements of the our development, unsufficient properties of kit blocked the project. SRAM Cells number of UP3 Education Kit was 5980 and it was very unsufficient to save *Ytrain* matrix whose size was 2304 x 48 or higher. Only some of *Ytrain* matrix could be saved to board. In that step, a solution could handle the problem. 8 MB SDRAM on UP3 was planned to store the matrix elements. However, Nios CPU, a soft-core processor, was required to be handled to reach external memory resources. However, since the USB blaster cable to program Nios II

CPU in UP3 kit is expensive, a new kit with higher memory resources has been purchased.

System development with UP3 Education Kit was worthwhile to gain experience on FPGAs. Because, we had no prior information about FPGA in the beginning of the project. VHDL code development and learning the working principles of FPGA were the plus points of using UP3 Education Kit. On the other hand, learning how to use Quartus II tool was another good point of working with UP3 Education Kit. Because it is a generic tool for a big range of FPGA in the market.

5.6 NIOS II Functional Units Implementation of DE2-70 FPGA Board By Altera SOPC Builder

Altera's Nios II is a soft processor, defined in a hardware description language, which can be implemented in Altera's FPGA devices by using the QuartusR II CAD system. To implement a useful system it is necessary to add other funcional units such as memories, input/output interfaces, timers, and communications interfaces. To facilitate the implementation of such systems, it is useful to have computer-aided-design (CAD) software for implementing a system-on-a-programmable-chip (SOPC). Altera's SOPC Builder is the software needed for this task.

The Nios II processor and the interfaces needed to connect to other chips on the DE2 board are implemented in the Cyclone II FPGA chip. These components are interconnected by means of the interconnection network called the Avalon Switch Fabric. The memory blocks in the Cyclone II device can be used to provide an on-chip memory for the Nios II processor. The SRAM, SDRAM and Flash memory chips on the DE2 board are accessed through the appropriate interfaces. Parallel and serial input/output interfaces provide typical I/O ports used in computer systems.

Since all parts of the Nios II system implemented on the FPGA chip are defined by using a hardware description language, a knowledgeable user could write such code to implement any part of the system. This would be a time consuming task. Instead, one can use the SOPC Builder to implement a desired system simply by

choosing the required components and specifying the parameters needed to make each component fit the overall requirements of the system. This section will describe how the system components were implemented by using SPOC Builder.

As mentioned above, SOPC Builder is a tool which allows the user to create a system based on the Nios II processor, by selecting the required functional units and specifying their parameters. To implement the face and speech recognition system, we had to configure the Nios II Processor(referred to as a Central Processing Unit), PLL, JTAG UART, internal timer, two SDRAM units, flash memory, parallel I/O interface and a tri-state bridge. Hence, a project was created and store in a directory. By following the selection of tools, a screen was available to select the system contents of the SOPC Builder which would be used to add components to the system and configure them based on the system requirements. It was possible to specify the clock frequencies of the Nios II processor. For this purpose, a clock named as *clk* was defined and designated as *External* and the frequency level was set to 50.0 MHz as shown in Figure 5.23. All components described above would be selected in the system contents tab window to add to the system. Also, device family was selected as *Cyclone II.* It was an important setting for the development of the components.

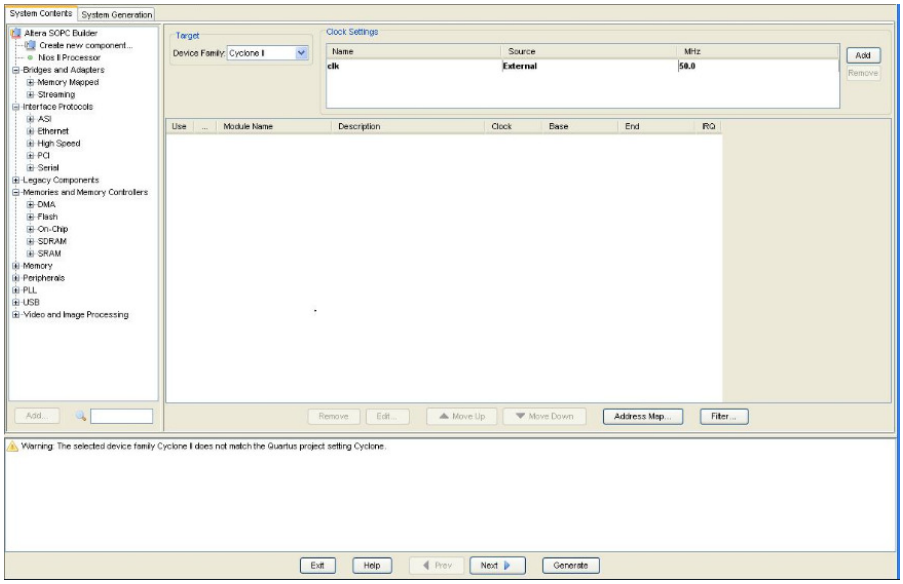


Figure 5.23 System contents tab window

On the left side of the window in Figure 5.23, *Nios II Processor* was selected and added to the system by clicking *Add* which leaded to the window in Figure 5.24.

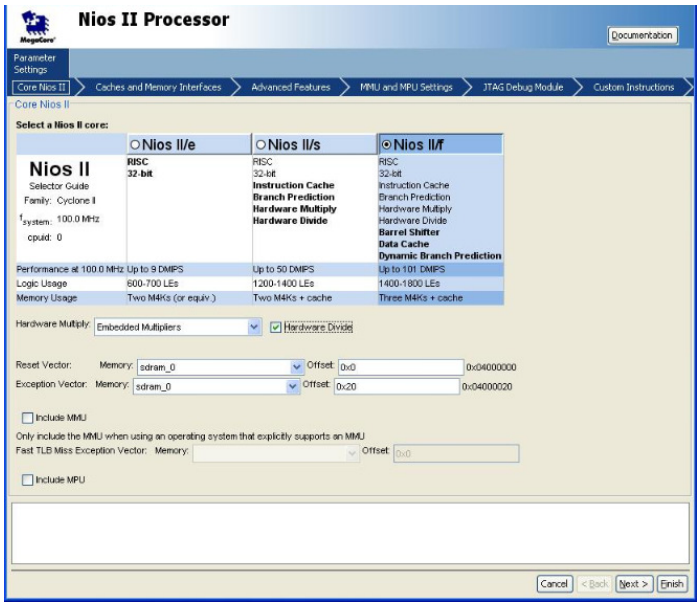


Figure 5.24 Nios II processor selection and setting window

Since a connection between the host computer and the Nios II system would be required, JTAG UART interface was configured. After selecting *JTAG UART* under *Communication* tab and clicking *Add*, the window in Figure 5.25 appeared on screen.

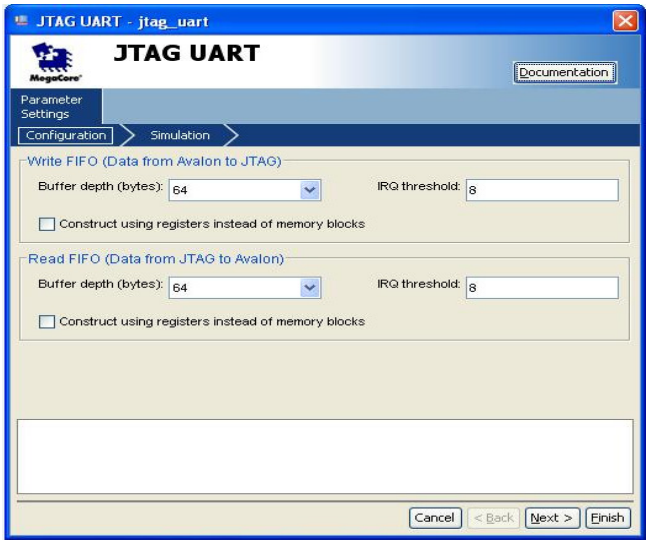


Figure 5.25 JTAG UART interface definition window

Timing operations were one of the basic operations in the system. This requirement leaded to add a timer to the system. Figure 5.26 shows the setting window of an internal timer.

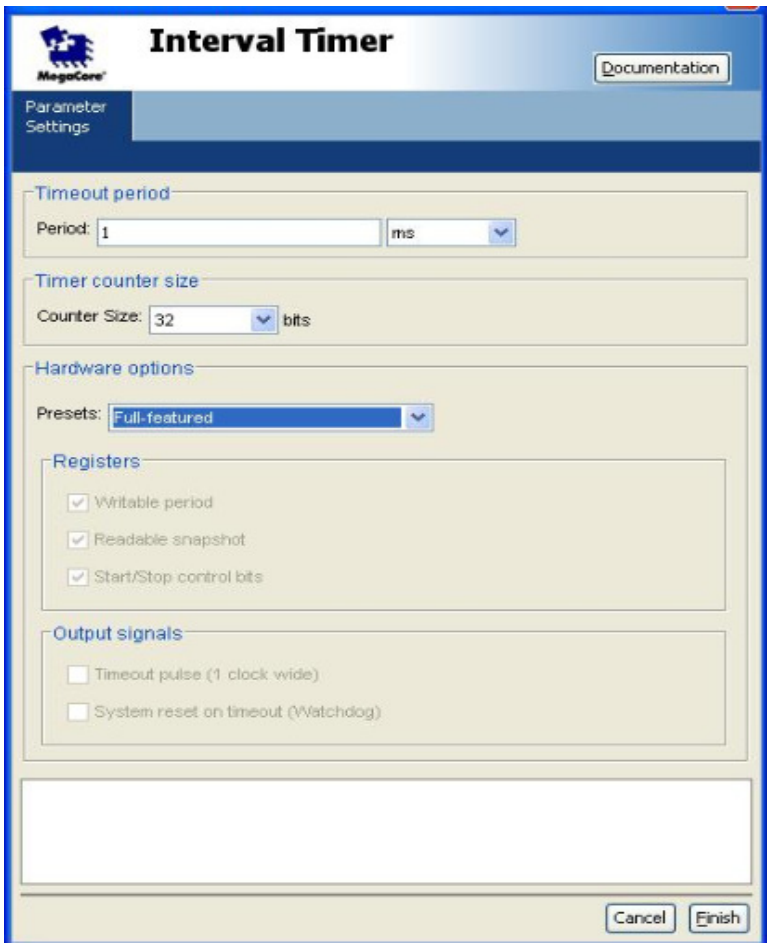


Figure 5.26 Internal timer definition window

The parallel I/O interface was configured by adding the *PIO(Parallel I/O)* and specified the setting in the PIO Configuration Wizard in Figure 5.27. The width of the port was set to 8 bits and the direction of the port was set to *Output ports only.* Also, reset value was set to 0x0 since the system didn't require any reset in its output ports.

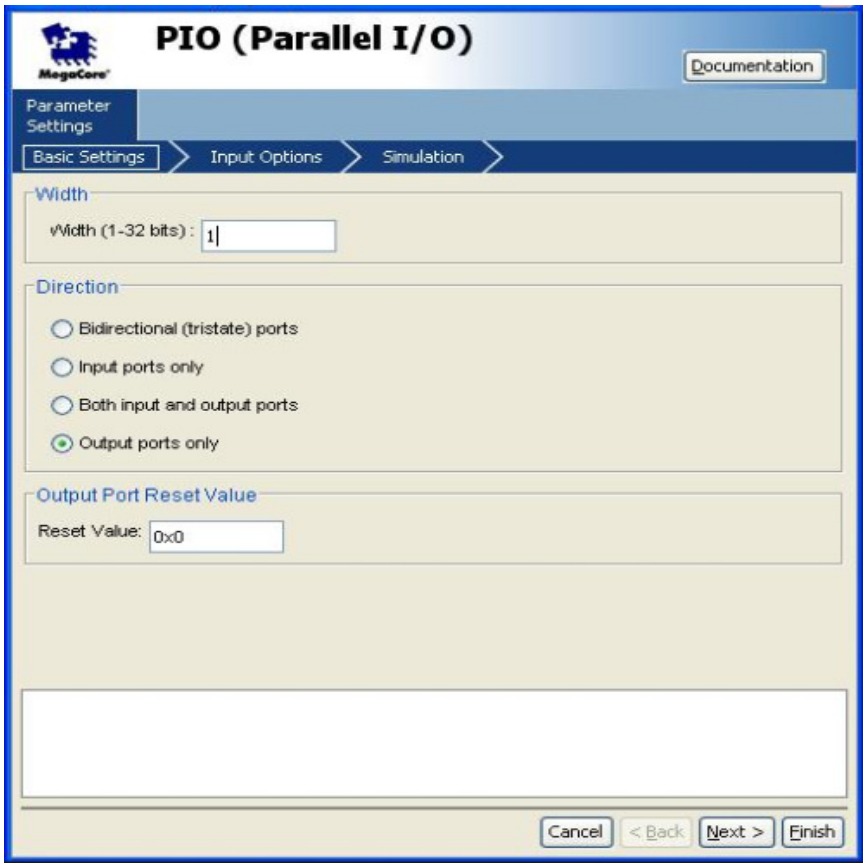


Figure 5.27 Parallel output interface definition window

For proper operation of the SDRAM chip, it is necessary that its clock signal, *DRAM0_CLK*, leads the Nios II system clock, *CLOCK_50*, by 3 nanoseconds. This can be accomplished by using a phase-locked loop (PLL) circuit. There exists a Quartus II Megafunction, called *ALTPLL*, which can be used to generate the desired circuit.So that, a PLL implementation was done by adding it in the system contents window and configuring the PLL by using the the wizard shown in Figure 5.28.

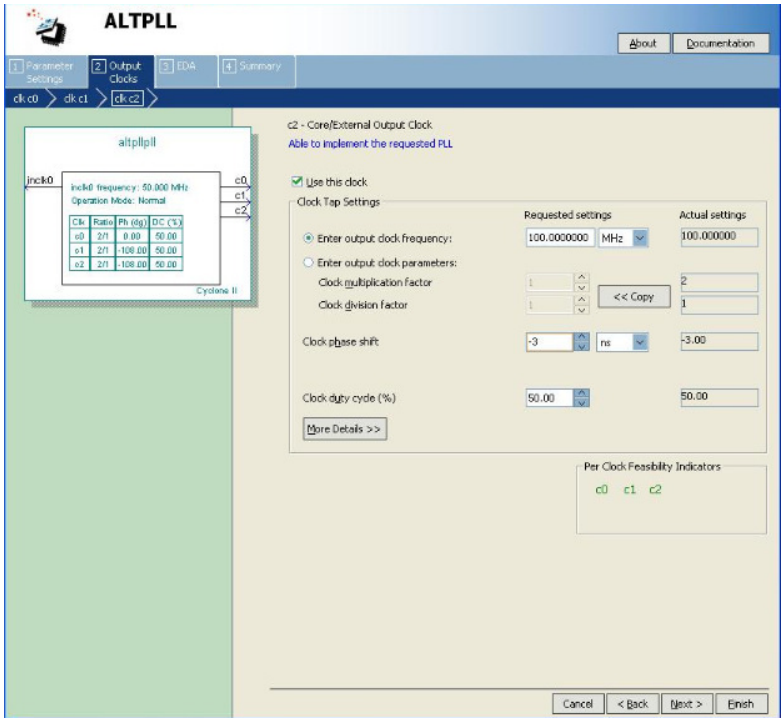


Figure 5.28 PLL interface definition window

DE2-70 board includes two SDRAM block which each have a capacity of 32 Mbytes for each one. All the signals needed to communicate the chip are controlled by a SDRAM controller which can be generated by using the SPOC builder. *SDRAM Controller* was selected under *Memories and Memory Controllers* > *SDRAM* and added by clicking *Add*. A window shown in Figure 5.29, appeared. In the configuration window, the settings were configured as: Data Width parameter to 16 bits, the Row Width to 13 bits, the Column Width to 9 bits, and the default values for the rest. After clicking *Finish,* the SDRAM module was added to the design. In order to produce the address assignments of module, the command under *System* > *Auto-Assign Base Address* was selected. Hence, SOPC Builder assigned the base address 0x2000000 for the SDRAM. This procedure was applied while implementing the second SDRAM Controller to the design.

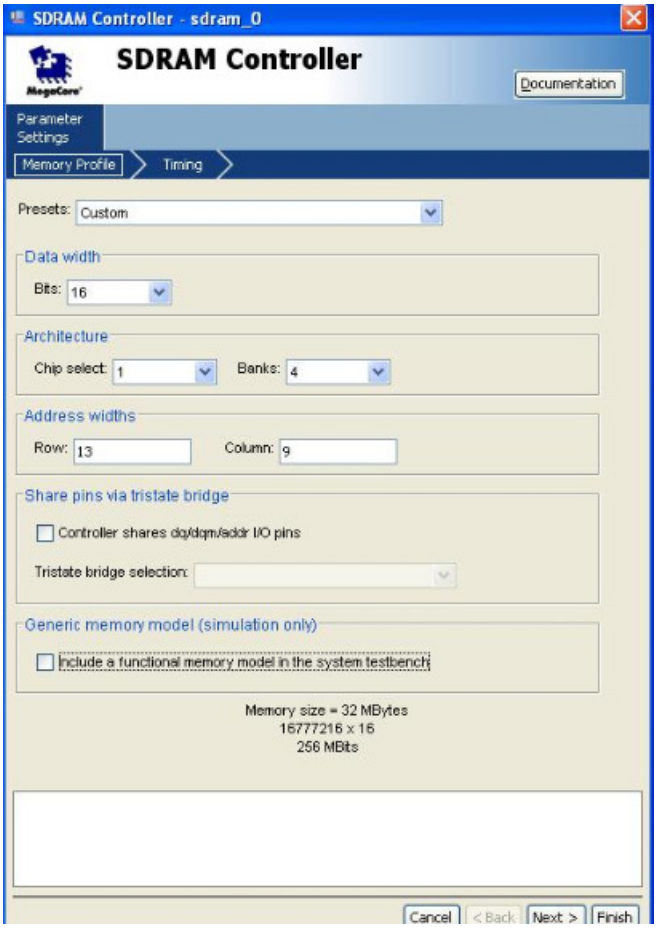


Figure 5.29 SDRAM controller definition window

The design of face and speech recognition required a flash memory since big size of matrix would be available. So that, *Flash Memory (CFI)* was selected under *Memories and Memory Controllers*. By clicking *Add* button, the window in Figure 5.30 appeared on screen to specify the the requirements.

The UART implementation was one of the basic steps, since two main environments, MATLAB and DE2-70 FPGA board, would communicate via UART. For this purpose, *Avalon Components* > *Communication* > *JTAG UART* was selected and *Add* buton was clicked to reach the UART Configuration Wizard in Figure 5.31. Since the default settings met our requirements, any of settings weren't changed.

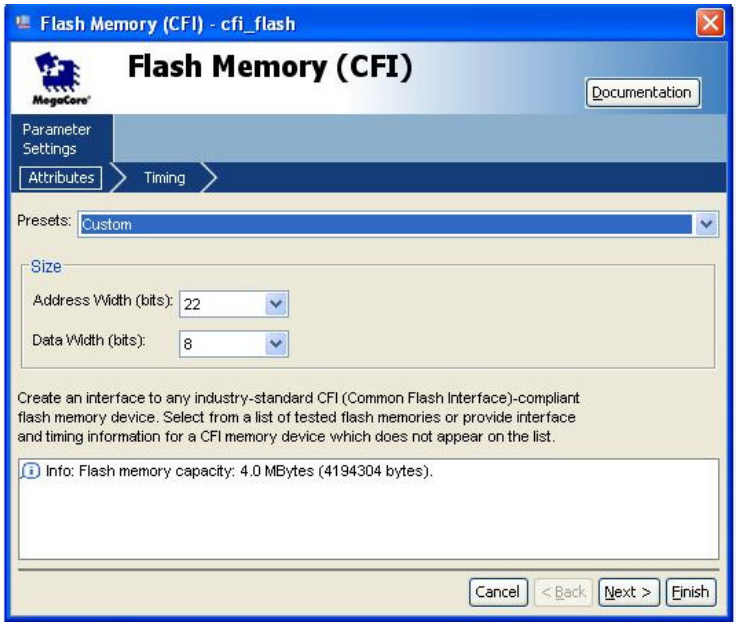


Figure 5.30 CFI (Common Flash Interface) definition window

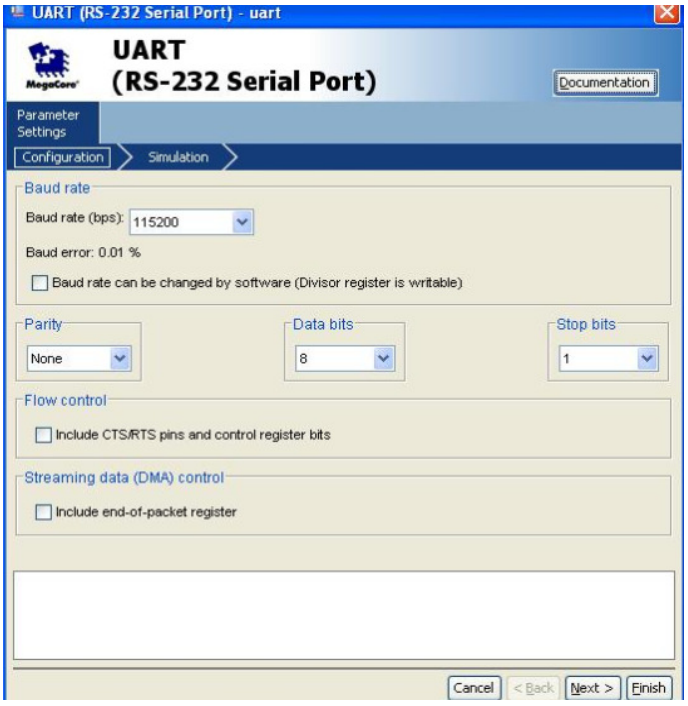


Figure 5.31 UART definition window

The resultant form of the system contents window is shown in Figure 5.32. All components sorted in the beginning of the section were already added and configured

based on the system requirements. Base addresses of the components were assigned as shown in Figure 5.33.

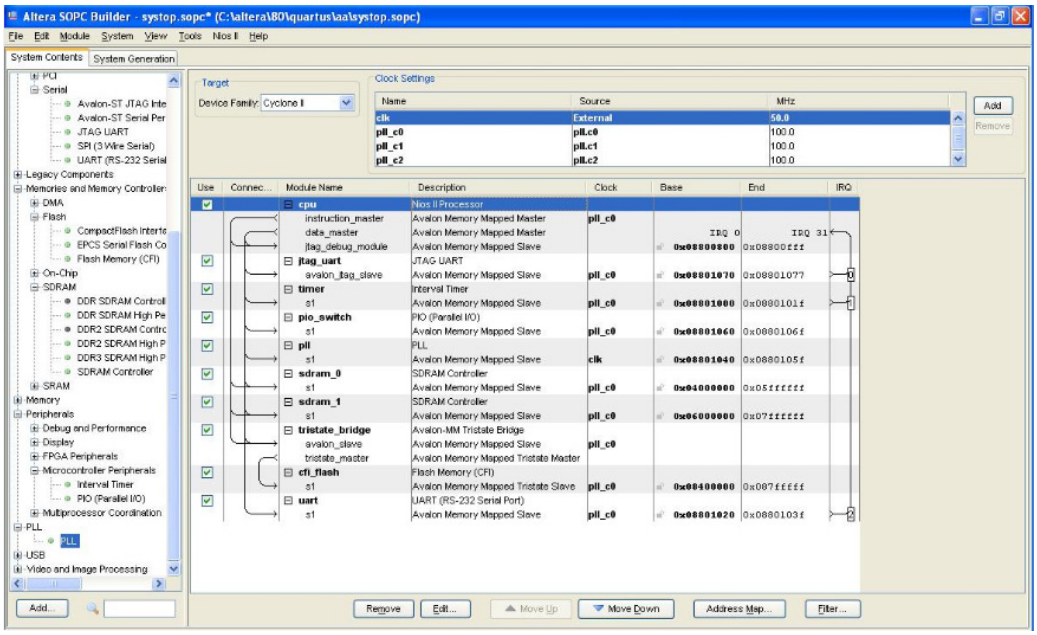


Figure 5.32 Final specifications of the components

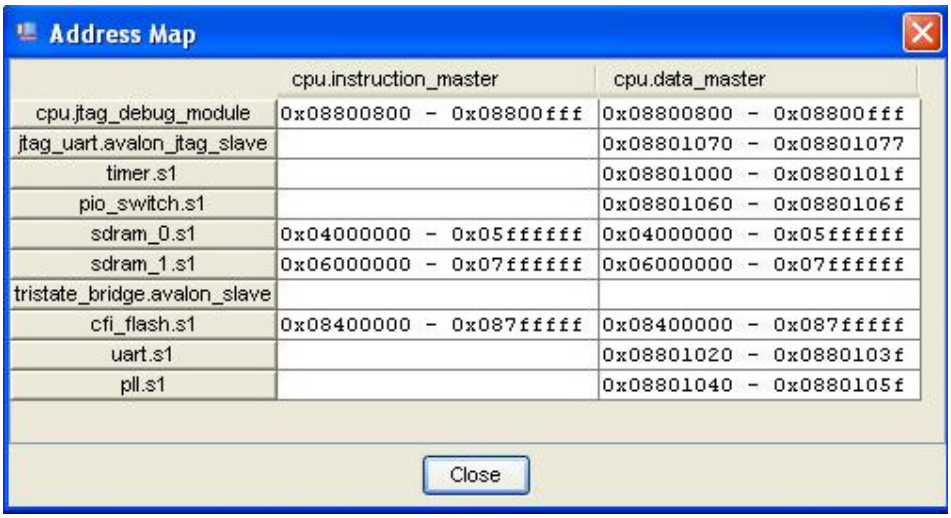


Figure 5.33 Base addresses of the components

Base addresses of the components are also shown in Figure 5.33 definitely. Next, pin assignments of components are done by taking reference the pin tables in documentation of DE2-70 board. By the way of example, the resultant pin

assignments of flash memory and SDRAM are displayed in Figure 5.34 and Figure 5.35, respectively. Also, Figure 5.36 shows the pin assignments of complete system.

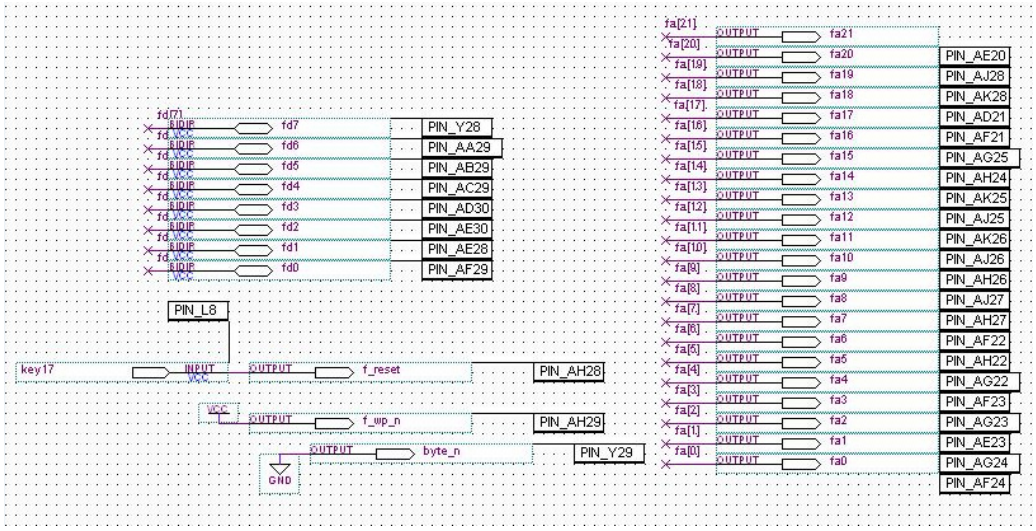


Figure 5.34 Pin assignments of flash memory of DE2-70 FPGA board.

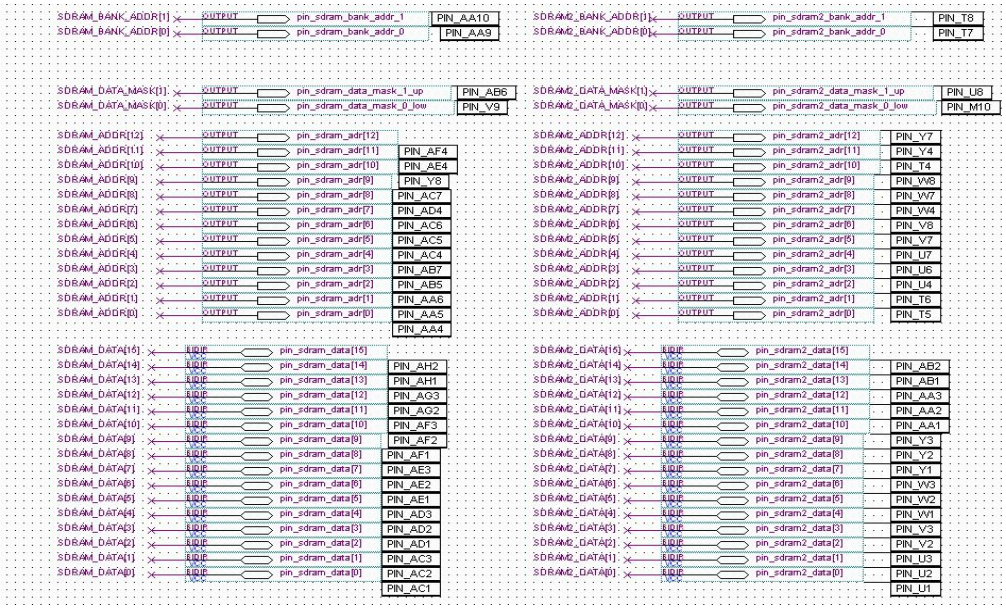


Figure 5.35 Pin assignments of SDRAM of DE2-70 FPGA board.

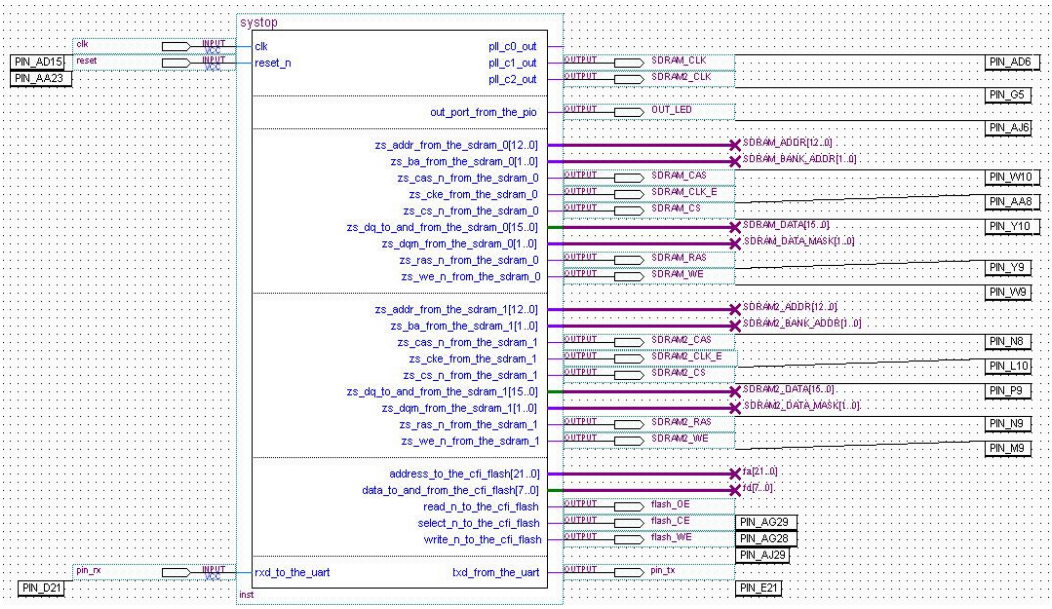


Figure 5.36 Pin assingments of complete system

5.7 Face Recognition System By DE2-70 FPGA

FPGA board used in the project has been changed due to characterictics of UP3 Education Kit. At this step, new development ideas for system generation in DE2-70 FPGA Board occurred in our mind. PCA implementations could be developed in DE2-70 FPGA Board as a new approach in addtion to other steps developed in DE2-70 FPGA Board. Section 5.7.1 describes the all FPGA implementation definitely. Section 5.7.2 describes the all MATLAB implementation steps including preliminary and unstable works and their results. On the other hand, section 5.7.3 describes the final and operative MATLAB implemetation steps and results.

5.7.1 FPGA Part Implementation

Development in DE2-70 FPGA board started by assigning contans synchronously with the constants in MATLAB. *KISI_SAY* which was used for total number of subjestc and *FEATURE_SAY* which was used for the elements number or each subject in the feature matrix, were defined respectively. Data acception to FPGA was

based on the data located in the flash memory of DE2-70 FPGA board. Memory addresses of 0x10000 and 0x60000 corresponds the first addresses of areas which data was saved. If these addresses were 0xFF (or, 255), it meant that flash memory was already empty and database elements could be received from MATLAB. Otherwise, there was no need to get any more database elements since they were loaded before. At that time, received data was assigned to a test image and the program switched to the test phase immediately.

UART implementation in FPGA worked to receive the data due to interrupts. When an interrupt was created to receive any data, interrupt service routine was called and data receive process started. *status* was read to check continuously, if FPGA UART was ready to receive any data. Hence, it was AND'ed logically by *ALTERA_AVALON_UART_CONTROL_RRDY_MSK* mask. If the result was true, it meant that buffer was ready and data could be recevied succesfully. At that point, situation of existing data became important. Database elements were received over RS-232 sequentially until *RxHeadData* variable that counts the number of received elements, was equal to *DATABASE_SIZE* definiton which was equal to the product of total images and the new size of images. If the program runs for the first time or flash memory of FPGA is erased to renew database, that situation is valid. Whenever that condition was satisfied, there was no need to receive any more database elements and test elements must be collected. *RxHeadTest* variable was used to detect if enough test elements were received or not. Since received data that will be used to test, is never saved to flash memory, a test vector is constructed and sent to FPGA for each test experiment. Source code of UART interrupt service routine is shown in Figure 5.37.

```
void uart_isr(void* context,alt_u32 id)
{
alt_u32 status;

status = IORD_ALTERA_AVALON_UART_STATUS(UART_BASE);

if(status & ALTERA_AVALON_UART_CONTROL_RRDY_MSK)
```

Figure 5.37 Source code of UART interrupt service routine.

```
{

 if(RxHeadData < DATABASE_SIZE)
 {
 RxDataBase[RxHeadData]=IORD_ALTERA_AVALON_UART_RXDATA(UART_BASE);
 IOWR_ALTERA_AVALON_UART_STATUS(UART_BASE,0);

  if((++RxHeadData) > (DATABASE_SIZE - 1))
  {
  //
  }
    }else{
   RxTest[RxHeadTest] = IORD_ALTERA_AVALON_UART_RXDATA(UART_BASE);
   IOWR_ALTERA_AVALON_UART_STATUS(UART_BASE,0);

       if((++RxHeadTest) > TEST_SIZE - 1)
       {
         //RxHeadTest = 0;
       }
    }
 }
}
```

Figure 5.37 (continued)

A preprocessing step was required to prevent unintended weighting effects of features. Hence some functions were developed in the source code. The main purpose of them was to determine the mean of coumn vectors of database matrix (by *mean* function) and to center the column vectors by subtracting the mean value from each elements. Afterwards, mathematical calculations of PCA started to reduce dimensionality. PCA algorithm, used in the project is basically a dimensionality reduction technique which transforms the feature matrix to a new matrix whose dimensionality is much less. Assume that received feature matrix is in size of 48 x 169. Number of 48 means that there are 8 images for each of 6 subject and number of 169 means that each image is represented in the size of 13x13. As mentioned before, each face image is represented as a column vector. That is why column number of database matrix is 169.

PCA requires the creation of variance-covariance matrix. Hence *m x m* covariance matrix is created from given *n x m* feature matrix while *n* is the number of whole image and *m* is the number of pixel data in a image. Since 8 images per each of 6 subjects were available and feature matrix in size of 48 x 169 was used, covariance matrix in size of 169 x 169 was constructed in *covcol* function and it was assigned to

symmat variable. Then, *tred2* function was called for householder reduction to a tridiagonal form. It was a triangular decomposition process which reduced a real, symmetric matrix to a symmetric, tridiagonal matrix. Next, *tqli* function run to evaluate the eigenvalues. *evals* parameters hold the eigenvalues and column of *symmat* matrix hold the associated eigenvectors. Mathematical formulas mentioned in section of PCA, were developed in C programming language sequentially. As a result, projections of column points on the first three principal components were created and stored in *symmat2* variable by overwriting the data which was already stored in it. Size of *symmat2* was 169 x 3 because of taking care of the first three principal components. Trained form of feature matrix was created by multiplication of data matrix which was sent from MATLAB, and *symmat2* matrix which was calculated due to mathematical calculations of PCA. Hence the result of multiplication, called as *NewYtrain*, was a matrix whose size was 48 x 3 (*FEATURE_SAY* x 3). At the end of training phase, the PCA basis matrix (*symmat2*) and projection of database to the face space (*NewYtrain*) were saved to the flash memory. Hence, there would be no need to create PCA basis matrix and the projection of database to the face space in next trials. But, type of elements in these matrices were float which caused errors during reading from and writing to the flash memory. In order to overcome these errors, the elements of these matrices were converted to integer type in *ConvertAllAndWrite2FlashAsChar()* function. *NewYtrain* and *symmat2* was written to flash memory beginning from 0x10000 and 0x60000 addresses, respectively.

As mentioned in the beginning of this section, content of flash memory are checked for all trials to detect if the flash memory is empty or not. The source code which controls the flash memory state is shown in Figure 5.38. The mechanism is based on the check the beginning addresses, 0x10000 and 0x60000. If the content of them isn't equal to 0xFF, it means the the flash memory is already loaded and the code is routed to *TEST_FONKSIYONU*. Otherwise, training phase is processed. *FlashTestYtrain* and *FlashTestPcaBasis* are 2 bytes of data block which read from flash memory.

```
.
..
if ((FlashTestYtrain[0]!=255)||(FlashTestYtrain[1]!=255)
   ||(FlashTestPcaBasis[0]!=255)||(FlashTestPcaBasis[1]!=255))

      {
        printf ("flash does not empty \n");
        goto TEST_FONKSIYONU;

      }else
     {
        printf ("flash empty \n");
       while (1)
      {
        printf ("data gelmiyor...\n");
        ..
        .
       }
```

Figure 5.38 The source code which detect whether the flash memory is empty or not

Recognition of the system would be based on searching that a test face belonged to whom. Hence one test image was experienced by the same calculations, described for the feature matrix. “testiyolla…” message on console indicated that the system required the test image. The basic of recognition phase was to compare the projections of database and test to the face space. So that, the data stored in flash memory were restored in *RestoreAllFromFlash()* function. This function basically restores the content of flash memory, starting from the address passed into the function. Restored data from flash memory were loaded to *FlashYtrain* and *FlashPcaBasis* matrices. Since the type of data stored flash memory was float, type conversion was applied to the elements of *FlashYtrain* and *FlashPcaBasis* and the resultant matrices in int type were obtained. Note that, the resultant matrices which were obtained after type conversion were equivalent to *NewYtrain* and *symmat2,* respectively.

All functions of PCA were applied to the test image, respectively. At the end, a matrix, called as *ytest*, was constructed. The size of *NewYtrain* was 48 x 3 because of 48 images in the database. The size of *ytest* matrix was 3 x 1, because one test face image to be tested.

Decision part of the system worked on finding the block in *NewYtrain* matrix whose difference from *ytest* matrix is the smallest. For this purpose, an algorithm was developed which compared each 3 x 1 part of *NewYtrain* matrix to *ytest* matrix one by one. Summation of the elements of each difference vector was saved to a vector, called as *toplam*. Indice of the minimum element in *toplam* vector corresponded to the face image in the database which was the most look like the test face image. Assume that the minimum element returned as 25. Since there were 6 face images per each of 8 subjects, the indice of 25 meant that test image was the closest to the first face image of 5^{th} subject. The result which indicated the best similarity was driven to the seven segment display to inform the user. In the example, the number of 5 would be displayed in seven segment display. The source code of implementation for DE2-70 FPGA board is in the Appendix with the folder name of "5_7_1_Face_Recognition_Configuration_Sw".

5.7.2 Preliminary Implementations In MATLAB Part

This section describes preliminary implementations in MATLAB which are unsatisfactory in terms of accuracy and stability. This trials were applied by using DE2-70 FPGA board. The main programming changes were to use a different image resize applications and to use windows with different sizes.

In the first applications, total 48 images of 6 subjects with 8 different images in "The ORL Database of Faces" were taken into care. Since each image in database had an original size of 112 x 92, the size of each face image were reduced to 48 x 48. In that case, *KISI_SAY* and *FEATURE_SAY* definitions in source code of FPGA were set to 48 and 2304, respectively. Also, a feature matrix was created in size of 2304 x 48 and a test vector was created in size of 2304 x 1 in MATLAB, since each face image was represented with 2304 (48 x 48) elements in a column vector. The feature matrix, *data*, was used in source code of FPGA as a transpose of the feature matrix in MATLAB. Hence, the size of *data* was 48 x 2304. At the end of PCA calculations, PCA basis matrix, *symmat,* was constructed with a size of 2304 x 2304. After sorting eigenvalues, the most significant 3 elements for each row were taken care and the

remaining elements were eliminated, since they didn't had any prior information. Hence, the a new PCA basis matrix, *symmat2*, was created with a size of 2304 x 3. In order to create the face space projections of images in database, *data* and *symmat2* was multiplied. The resultant matrix, *ytrain,* had a size of 48 x 3 and carried the information of face space projections of images in database. But, this kind of implementation wasn't reliable, since the total calculation time in DE2-70 was too long. The resultant matrix, *ytrain*, was constructed at the end of two and half days. It wasn't an acceptable time duration in real applications.

The first idea to overcome the training time was to reduce the size of database images more. In that case, some experiments were applied by recuding the size of each image to 15 x 15, 20 x 20, 30 x 30, 40 x 40. In this case, training time reduced to acceptable levels. But the recognition rate in these cases couldn't meet the expectations. If new size of images was more than 20 x 20, the recognition rate was approximately %40. Even if new size of images was lower than 20 x 20, the recognition rate was %60 at the most. Table 5.1 shows the recognition rate of the system when new size of images is 15 x 15 and the database consists of 40 images totally (8 images for each of 5 subjects). This system is labelled as System A.

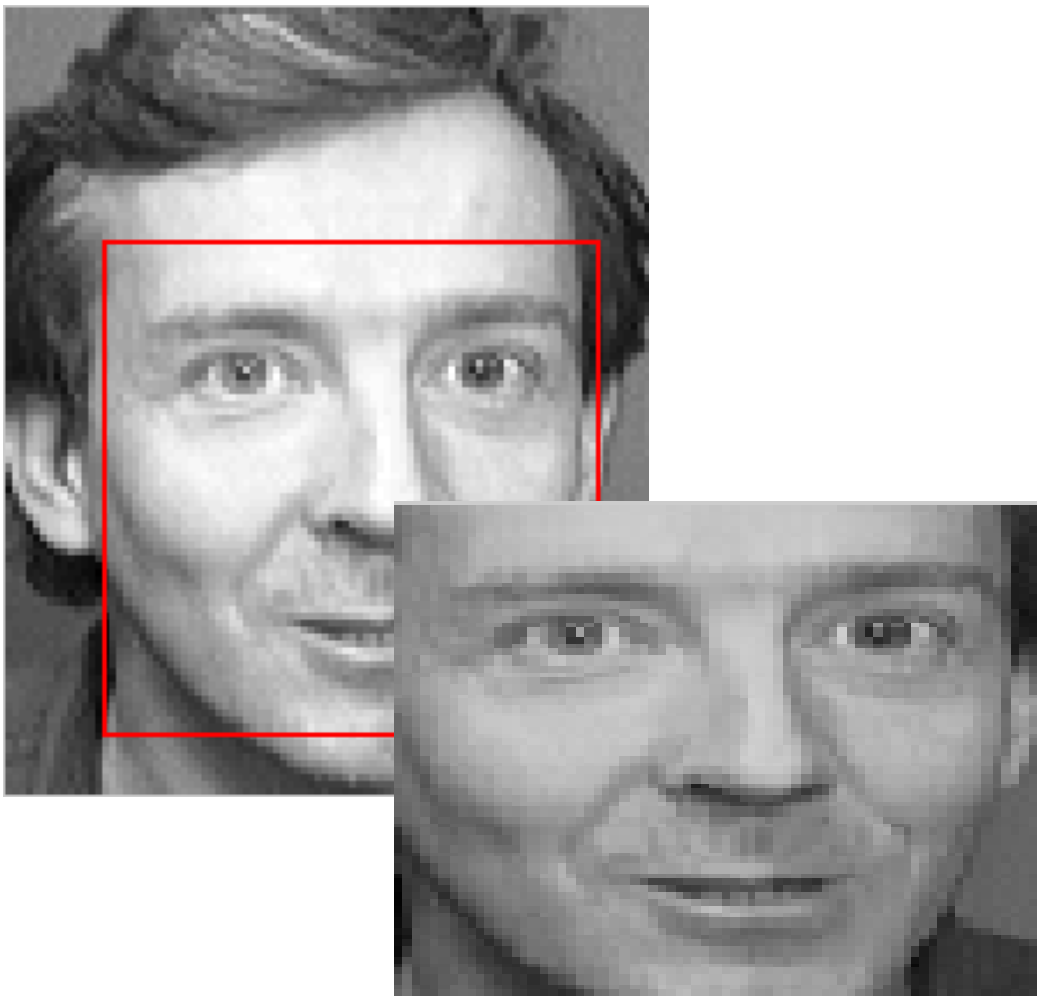
Table 5.1 Recognition table for System.A (40 database images, 15x15 image size)

Recognition Results		Test Image	
		9th	10th
Subject Number	1st	+	+
	2nd	-	+
	3rd	+	-
	4th	+	-
	5th	-	+

The image detection algorithms in MATLAB plays an important role in face recognition systems, since they detect the face in an image. If they don't recognize the face in an image succesfully, wrong feature matrix is constructed and sent to FPGA. Usually, the background in face images is a specific colour. It is defined a pixel value in a small range and it doesn't any information for the face recognition

system to recognize tha face correctly. So that, pixels which defines the face, must be handled as much as possible.

In order to find new approaches about image resizing, some algorithms were searched over internet and an algorithm, called *fdmex*, was found. *fdmex* is a dynamic link library file which used in OpenCV. It can be described as a dll file which detects a face in an image. In the MATLAB implementation of the project, *fdmex* was used before *imresize* to analyze the possible positive or negative effects. Figure 5.39(b) shows a face which is detected by *fdmex* from the image shown in Figure 5.39(a).



(a) (b)

Figure 5.38 (a) A face image from database before *fdmex* is applied and (b)the new form of the face image from database before *fdmex* is applied

After *fdmex* was applied to the images in database, *imresize* is called again to reduce the size of images to desired levels. The face images, *fdmex* was applied, were resized to 30 x 30. This procedure was applied to all images in database. Since the size of feature matrix was extremely high when the new size of images was 30 x 30, the windowing method was implemented to system. A window with size of 15 x 15 was created and each image was divided into four quarters, since size of images after resizing process, was 30 x 30. At that point, we had four sub windows with size of 15 x 15. The best idea was to label each sub window as feautre matrix and to transfer one of these sub windows to DE2-70 FPGA board as feature matrix, since size of sub windows were reasonable to process in DE2-70 FPGA board. Next, the database images were rearranged in order to construct a feature matrix where each column represents a resized image. Since there were 5 subject with 6 images for each subject, column number of feature matrix was 30. On the other hand, new sizes of each sub window was 15 x 15. So that, row number of feature matrix was 225.

The FPGA implementation described in Section 5.7.1, was applied to feature matrix. f*dmex* application was applied to the test image to detect face, too. The resultant image was resized to 30 x 30 and windowing method was applied to test image to get a desired level to transmit to FPGA. Suitable sub window with size of 15 x 15 was labelled as test image and reaaranged to construct a test vector. The size of test vector was 225 x 1 since one of sub windows was used. The test vector was sent to DE2-70 FPGA board after rearranging.

Table 5.2 shows the the recognition results of the system definitely. This system is labelled as System B. In the recognition tables of thesis, the rows represent the people for which the system is trained for. The columns show if the test image recognized for this recognized correctly or not. In this table, since the first 6 images of 5 subjects were used to construct the feature matrix, 7^{th}, 8^{th}, 9^{th}, 10^{th} images of the 5 subjects were labelled as test image. “+” sign defines that the available test image is recognized. On the other hand, “-” sign shows that the available test image isn’t recognized correctly. The source code of *fdmex* implementation is in the Appendix with the folder name of “5_7_2_fdmex_example”.

Table 5.2 Recognition table of System B (the system with *fdmex* algorithm, 30 database images)

Recognition Results		Test Image			
		7th	8th	9th	10th
Subject Number	1st	+	+	-	+
	2nd	+	-	+	+
	3rd	+	+	-	-
	4th	-	+	-	+
	5th	+	-	-	+

As shown in table above, the success rate of the system was approximately %60. In the next step, *fdmex* face detection algorithm was not used in the system. Instead only available image is resized. The windowing method was still available with window size of 15 x 15. While all other MATLAB and FPGA part implemetations were applied in the same way, the recognition results shown in Table 5.3 were obtained. This system is labelled as System C. Although the success rate increased, it wasn't still enough. In Section 5.7.3, the final face recognition results with the resultant technique is described.

Table 5.3 Recognition table of System C (the system without *fdmex* algorithm, 30 database images)

Recognition Results		Test Image			
		7th	8th	9th	10th
Subject Number	1st	+	+	-	+
	2nd	+	+	+	-
	3rd	-	+	+	-
	4th	+	+	-	+
	5th	+	-	+	-

The systems described above was far from meeting the expectations. Test results showed that the best recognition rate was available when the size of images was reduced to 15 x 15. The new approach to achieve better recognition rates was to divide the face image into sub face images after its size was reduced 112 x 92 to 30 x 30. By the way of example, an image with size of 30 x 30 could divided into 4 sub images with size of 15 x 15. Figure 5.40(a) shows the resized form of an image. Figure 5.40(b) shows the face image in the form of seperated into 4 regions.

(a) (b)

Figure 5.40 (a) The form of image after resizing and (b) the form of image after resizing and dividing into 4 regions

In this case, each of regions were thought as an individual image. Hence, all calculations were done by assuming the size of face image as 15 x 15. This procedure were applied to 4 regions, respectively. Table 5.4, table 5.5, table 5.6 and table 5.7 shows the recognition rate of each sub face images, respectively. This system is labelled as System D. The success rate of this system is %80 for region 1, %65 for region 2, %40 for region 3 and %50 for region 4.

Table 5.4 Recognition table for Region 1 of System D (15 x 15 sub face size, 30 database images).

Recognition Results		Test Image			
		7th	8th	9th	10th
Subject Number	1st	+	+	+	+
	2nd	+	+	+	+
	3rd	+	-	-	+
	4th	+	+	+	-
	5th	+	+	-	+

Table 5.5 Recognition table for Region 2 of System D (15 x 15 sub face size, 30 database images).

Recognition Results		Test Image			
		7th	8th	9th	10th
Subject Number	1st	+	+	+	-
	2nd	-	+	+	-
	3rd	-	+	+	+
	4th	+	-	+	-
	5th	-	-	-	+

Table 5.6 Recognition table for Region 3 of System D (15 x 15 sub face size, 30 database images).

Recognition Results		Test Image			
		7th	8th	9th	10th
Subject Number	1st	-	-	-	-
	2nd	-	-	+	+
	3rd	+	+	-	+
	4th	+	-	-	-
	5th	-	-	+	+

Table 5.7 Recognition table for Region 4 of System D (15 x 15 sub face size, 30 database images).

Recognition Results		Test Image			
		7th	8th	9th	10th
Subject Number	1st	+	+	+	+
	2nd	-	-	-	-
	3rd	+	+	-	+
	4th	+	+	-	-
	5th	-	+	-	-

5.7.3 Final Implementation in MATLAB Part

5.7.3.1 Final Implementation Steps in MATLAB Part

MATLAB part of the project had a mission of taking the photos from the world and transfer them to FPGA to be processed after preprocessing. Although MATLAB implementation had two files, one for training and one for testing, both of two files were similar to each other. While MATLAB file for training were preprocessing for database photos, MATLAB file for testing made a similar process for test face image

The system design starts by taking different image files to construct a training set in the training phase of the project. The first job in MATLAB was to assign the number of subject and the number of images per each subject. When the identification of a person was required, an image of that person (test image) was given to the system. This phase was called as the test phase (or recognition phase). There were some ways to integrate these images files into the system. These files could be obtained by taking photo of a human or by using ready image databases.

In development steps of the project, “The ORL Database of Faces” was used since it is a widely used face database. It contains a set of face images. There are 10 different images of each of 40 distinct subjects. For some subjects, the images are taken at different times, varying the lighting, facial details such as open/closed eyes, smiling/ not smiling, glasses/no glasses etc. All the images have a dark homogeneous background with the subjects in an upright, frontal position. The size of each image is 92x112 pixels, with 256 grey levels per pixel. Samples from database are represented in Figure 5.41.

Figure 5.41 Some of image files in the ORL face database

Constants were defined in the beginning of MATLAB training part like number of distinct subject that training set had, number of image files per each subject and size of new figure after face detection since they were required in the construction of database. As mentioned before, the main purpose of the training phase was to extract feature vectors for each image in the training set. Hence the following steps were applied respectively. The database images were copied to a specific folder. This copy process was done from the folder which includes all images to the folder which includes only required images. As an example, if database contained 6 images of each of 5 subject, totally 30 images were copied to the new folder for the further development. Next, size of each image was resized before transferring data matrix of training set images to FPGA The main reason of resizing is computational costs. An optimal size to obtain an acceptable recognition rate has been selected. *imresize*

function is used to resize the training set images in the given height and width values. It read an grayscale intensity image and reduced in different sizes during experiments. Then, all resized images were copied to a new folder to prevent the confusion with non-resized images.

During experiments, several methods were tried to get the best recognition rate. But, these experiments showed that if size of resized image was higher than 16x16, recognition rates were quite low. In order to reduce the computational burden, the windowing method has been applied. After resizing the images, a window whose size was 3x3, was applied to the new images. The mean of each window became an element of the feature matrix. Since the size of resized images was 40x40, 13x13 matrix was created for each image by windowing and mean processes. Windowing was applied to all images in the database. An original and resized face images are shown in Figure 5.42(a)-(b). On the other hand, Figure 5.43 shows the new form of database images after windowing the whole image and taking the mean of the resultant windows.

(a) (b)

Figure 5.42 An original face image (a) and the resized form of the face image (b)

Let Θ_A be a resized training image of a person A which had a pixel resolution of *M x N* (M rows, N columns). In order to send the whole feature matrix, called as *Xbasis* and to extract PCA features of Θ_A, each resized image in training set was converted into a pixel vector $\emptyset_A$ by concatenating each of the M rows into a single vector. It was repeated for each training image Θ_i and whole feature matrix where each column represents a resized image, was constructed. The column number of the feature matrix which would be processed in FPGA, was *k x m* where *k* is number of distinct subject in training set and *m* is the number of image files per each subject. The row number of the feature matrix was *M x N*. Since each image was represented by a 13x13 matrix, row number of the feature matrix was 169.

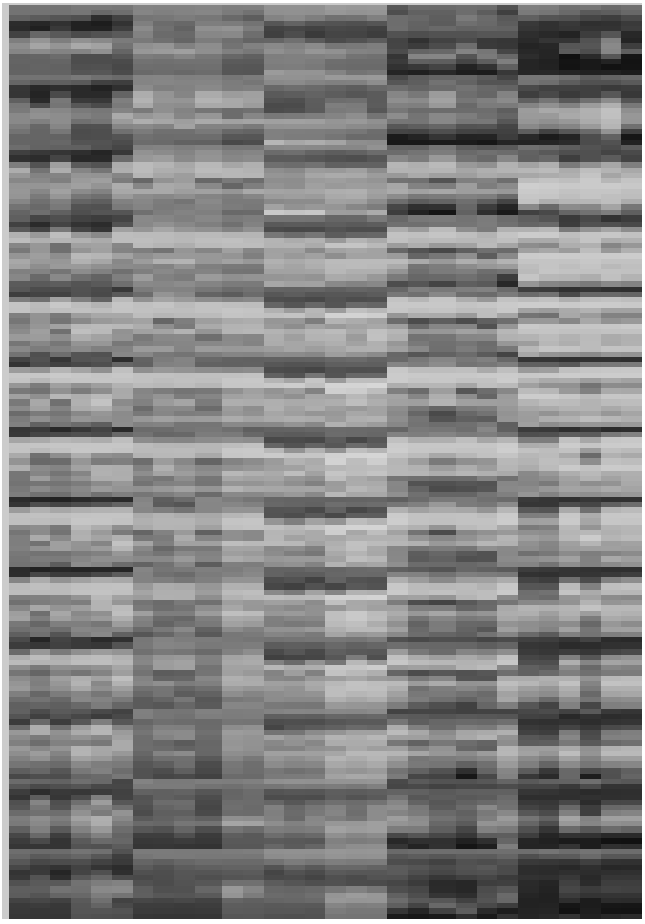


Figure 5.43 A face image after calculating mean of each window

Data communication to FPGA was over UART like the project done in UP3 Education Kit. As mentioned in Section 5.4.3, MATLAB serial communication toolbox was used to transmit the feature matrix to FPGA. Basically, 4 functions run sequentially. They were *serial* function to construct a serial port object associated with an existing port, *fopen* to connect the serial port object to FPGA, *fwrite* to write the feature matrix and other data to FPGA that was already connected to defined serial port object and *fclose* to disconnect the serial port object from FPGA. Baud rate was set to 115200 bits/second which was optimum speed for serial communication.

The part described above was related to obtain the feature matrix depending on the face images in the database. Similarly, test matrix was constructed in MATLAB for the system to be tested. Since only one face image was used to recognize, matrix sizes would be different from the sizes in the training. A face image was resized. Then it was windowed by using the same window used for database images. By taking the mean of all window in the test face, a new matrix was created where each element corresponded to the mean of a window. Next, it was rearranged in such a way that a column vector would represent the image. The matrix whose size was 1 x 169 was transmitted to DE2-70 FPGA to be recognized. UART communication speed to transfer test vector between MATLAB and FPGA device was acceptable. Size of test vector which is sent for each test procedure is much lower than size of feature matrix. Although transfer time for feature matrix is a bit high, it can be acceptable since feature matrix is a process for once. The whole source code of the final implementation is in the Appendix with the folder name of "5_7_3_1_Highest_Recognition_Rate_For_Face_Recognition".

5.7.3.2 Final Implementation Results

The image sizes used in the project was too large for analysis. So that, image sizes were reduced from 112 x 92 pixels to 40 x 40 pixels by *imresize* function in MATLAB. The database consisted of 5 subjects with 7 different images for each subject. Therefore, there were totally 30 images in the database. All images used for

database and test process, were scanned with a 3 x 3 window as explained in Section 5.7.3.1. Face database used in the face recognition system, consisted of 40 subjects with 10 different images for each subject. Hence, in the test process of the system, the remaining 3 images of each subject which not used in database, were identified as test face image. Since the first 7 images of each subject were dedicated as database face image, 8^{th}, 9^{th} and 10^{th} images of each subject were labelled as test image. Table 5.8 shows the recogniton rate of the system. As shown in table below, recogniton rate of the system is quite well. Only two images were recognized wrong. This system is labelled as System E in the table below. The recognition success of this system is %93.3

Table 5.8 Recognition table for System E (40x40 image sizes, 3x3 window and 15 test images)

Recognition Results		Test Image		
		8^{th}	9^{th}	10^{th}
Subject Number	1^{st}	+	+	+
	2^{nd}	+	+	+
	3^{rd}	+	+	+
	4^{th}	+	+	+
	5^{th}	-	+	+

The design described above had a high recognition rate. In order to observe that high recognition success wasn't based on the image number, images of a subject in System E were eliminated completely. While all other specifications were same such as new size of images after resize process, window size, image number in database etc., new tests were applied. Since the first 6 images of each subject in the database folder were dedicated as database face image, 7^{th}, 8^{th}, 9^{th} and 10^{th} images of each subject were labelled as test image. Table 5.9 shows the recogniton rate of the system. As shown in table below, the recognition rate is quite well. Only one image is recognized in a wrong way. This system is labelled as System F in the table below. Note that, the images used in System F belongs the first 6 subjects used im System E. In this case, 3 images used as test of 4^{th} subject failed. Recognition success of this system is %85.

Table 5.9 Recognition table for System F (40x40 image sizes, 3x3 window and other 20 test images)

Recognition Results		Test Image			
		7th	8th	9th	10th
Subject Number	1st	+	+	+	+
	2nd	+	+	+	+
	3rd	+	+	+	+
	4th	+	-	-	-
	5th	+	+	+	+

The next change was to observe the effects the stability of the system. Hence, new 5 subjects were chosen from "The ORL Database of Faces". All images used in system E and System F were replaced by the images of new subjects. New database would have 5 subjects with 7 different images for each subject. The other specifications were same. In other words, new size of images after resize process was 40 x 40 and window size was 3 x 3. Since the first 7 images of each subject in new database folder were dedicated as database face image, 8th, 9th and 10th images of each subject were labelled as test image. As shown in Table 5.10, new form of database didn't reduce the recogniton rate. There was only one image which recognized in a wrong way. This system is labelled as System G in the table below. The recognition succes of the system is %93.3.

Table 5.10 Recognition table for System G (40x40 image sizes, 3x3 window and 15 test images used in System E)

Recognition Results		Test Image		
		8th	9th	10th
Subject Number	1st	+	+	+
	2nd	+	+	+
	3rd	+	+	+
	4th	+	-	+
	5th	+	+	+

The next experiment was to change the subject number in the latest database. Hence, the images of 7th subject were eliminated from database. Since the first 6

images of each subject in new database folder were used as database face image, 7th, 8th, 9th and 10th images of each subject were dedicated as test image. As shown in Table 5.11, new form of database didn't reduce the recogniton rate. Although test image belonged to 5th subject in database, the system returned that it belonged to 1st subject. All other results returned from the system, were correct. This system is labelled as System H in the table below. The recognition succes of the system is %90.

Table 5.11 Recognition table for System H (40x40 image sizes, 3x3 window and 15 test images used in System F)

Recognition Results		Test Image			
		7th	8th	9th	10th
Subject Number	1st	+	+	+	+
	2nd	+	+	+	+
	3rd	+	+	+	+
	4th	+	+	-	+
	5th	+	-	+	+

5.7.4 General Overview to Face Recognition System Performance

General performance of the face recognition system was achieved by creating a confusion matrix depending on the multiple testing. Confusion matrix basically contains information about actual and predicted classifications doene by a classification system. Each row of the confusion matrix includes the instances in a predicted class. On the other hand, each column represents the instances in an actual class. Thereby, it is much more easier to see the confusing degree of two classes.

Previous experiments to see the performance of the face recognition system was based on selection of 10 face images for each of 5 people from ORL Face Database. The 6 face images of each subject were assigned as database images and rest 4 face images were assigned as test face images. In order to generate a confusion matrix, %60 of face images were selected as database face images among 10 face images for

each of 5 people. The remaining %40 face images were used as test face images. The selection of database and test face images was done with the help of cross-validation technique. Cross-validation technique was implemented bu the command of *crossvalind* in Bio-informatic Toolbox of MATLAB. The source code of this system is in the Appendix with the folder name of "5_7_3_3_Cross_Validation". When the cross-validation technique was applied to make selection, a matrix was constructed whose each row represented the face images and each column represented the subject IDs from 1 to 5. The resultant matrix included 0's and 1's. An element of 0 meant the corresponding image would be labeled as a test image while an element of 1 meant the corresponding image would be database face image.

Cross-validation implementation provided 25 different subsystems which different face images of 5 same people were labeled as database and test face images. All face images in subsystems were tested to see if it would be recognized correctly, respectively. Table 5.12 shows the general performance of face recognition system in the format of confusion matrix.

Table 5.17 Recognition table of the speech recognition system in the format of confusion matrix

	1st	**2nd**	**3rd**	**4th**	**5th**
1st	%95	%0	%2	%3	%0
2nd	%0	%95	%1	%4	%0
3rd	%0	%0	%100	%0	%0
4th	%0	%0	%5	%95	%0
5th	%0	%0	%2	%1	%97

According to Table 5.12, all face images of 3rd subject were recognized correctly. However, there are some incorrect recognition results for the face images of other subjects. As an example, the face images of the 1st subject were recognized as 3rd subject in the rate of %2 and recognized as 4th subject in the rate of %3. Also, the face images of the 4th subject were recognized as 3rd subject in the rate of %5 while all other tests were resulted in correctly. The general performance of the face

recognition system is calculated as %96.4 by dividing all recognition rates to number of subjects.

5.8 Speech Recognition System By DE2-70 FPGA

5.8.1 MATLAB Part Implementation

MATLAB part implementation of speech recognition project mainly includes acquisition of speech signals from outside and transmitting the pre-processed data to FPGA for Fourier Transform. As in the face recognition system, a database was constructed which included speech signals of subjects to be compared in the training phase. A MATLAB m-file, named *sesalma*, was developed to record the speech signals. *sesalma* file was developed mainly using the functions Data Acquisition Toolbox of MATLAB. In *for* loop, an analog input object was created and additionally, hardware channels were added to the onject created. After specfiying the characteristics of the object, the speech data was collected and saved to the specified disk file. There were 5 subjects and 6 speech recordings for each subject. Each subject said a specified word, “bir” which was a Turkish word, for 6 times and as a result, database was built up succesfully . The modified form of m-file was used for data acquisition of test speech data.

Data acquisition part worked on the basis of recording the speech data spoken in 3 seconds. Since the starting and ending point of the word should be located in 3 seconds recording, a pre-processing function called *keser* is used. Also in order to increase the quality of the input data, a low pass filter called *lowpasL* function was developed. Pre-processing started by removing the DC components by subtracting the mean of the speech signal from itself. The next procedure was to remove the unwanted noise and silence. So that, the log energy and zero crossing rates for each speech sample were computed. They were labeled as the upper and lower energy levels.

Once more control mechanism was applied for the lower energy level. If there were frames that had higher energy than levels determined based on the upper energy level, it was replaced with the previous index. If not, results of upper energy level were valid. At the end of *keser* function, the partition which existed between end-points of the speech signals were obtained respectively. Then, low pass filter design, named *lowpasL* function, were implemented to remove noisy part of speech signals. A recorded speech signal is shown in Figure 5.44. On the other hand, the new form of the same speech signal after pre-processing is shown in Figure 5.45.

Since end-point detection algorithms in *keser* function, worked by eliminating the samples whose energy levels are lower than initial and end energy levels, the resultant sample number at the end of *keser* function wasn't constant. Because, the energy levels of each subject is changeable. By the way of example, when the first subject said "bir", the sample number was approximately 1200 after *keser* function was applied. On the other hand, when the second subject said "bir", the sample number was approximately 1500 after *keser* function was applied. In order to generalize the sample number which would be sent to DE2-70 FPGA, a new method was developed in *main_train* and *main_test* functions. According to this, the speech data which end-point detection algorithm was applied in *keser* function, was taken care and the first sample whose magnitude higher than 0.35, was found. The next 1024 sample beginninng from the indice of the available first sample, was assigned to new speech data matrix and all other samples were eliminated. This procedure was applied to all database and test speech data, respectively in order to achieve a fixed data number for each speech data. The samples of speech signals were in the range of -1 and +1. So that, they couldn't be transmitted to FPGA over UART. Because, UART implementation required integer samples to transmit and receive. To achieve a problem-free communication, all data samples were normalized to the range of 0 and 255. Next, mean of the all pre-processed speech samples for each subject were calculated. Since there were 5 subjects in database, a feature matrix was generated with each column represented the mean of the pre-processed speech samples for each subject. Feature matrix was transmitted to FPGA over UART due to applying the

similar procedure done in face recognition project. Figure 5.46 shows the last form of speech signal created by selecting the 1024 samples and eliminating the remaining.

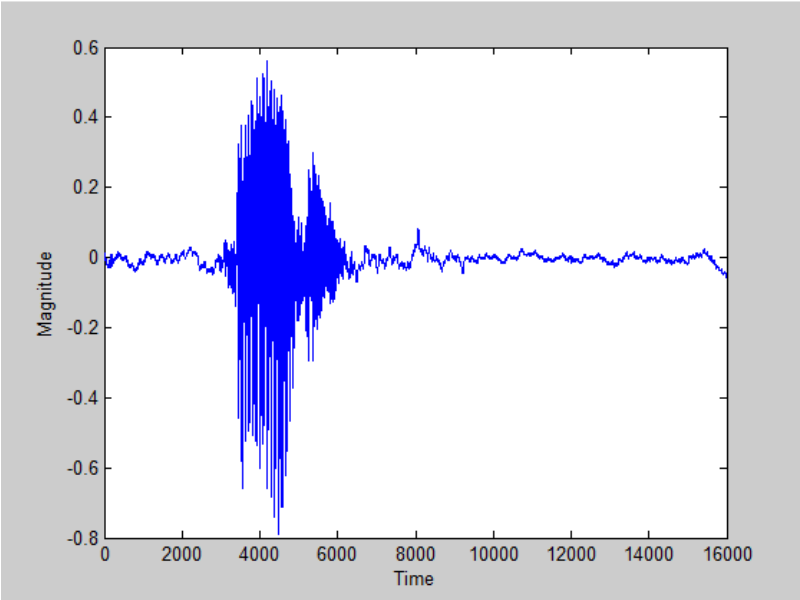


Figure 5.44 An example of recorded speech signal

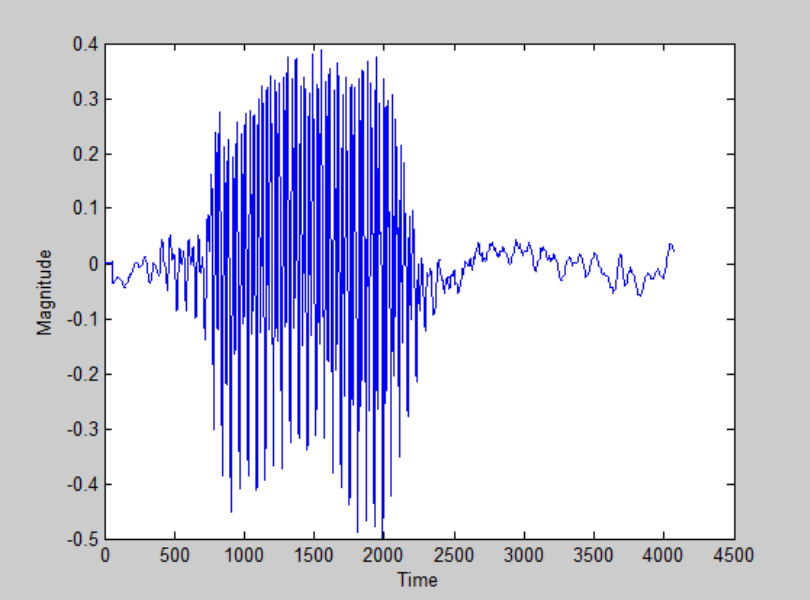


Figure 5.45 An example of pre-processed speech signal

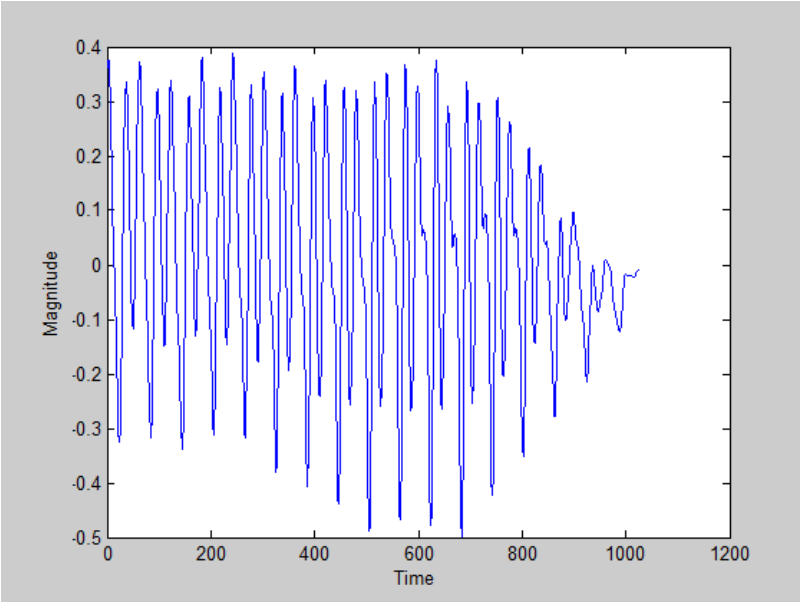


Figure 5.46 An example of speech signal after selecting 1024 samples

MATLAB implementation of the test phase was similar to the training phase. The only difference was to apply all proccedure for a test speech signal instead of applying to all database speech signals in training phase.

5.8.2 FPGA Part Implementation

Initial part of FPGA implemetation of speech recognition project is parallel to initialisation of face recognition project. Constants were assigned synchronously with the constants in MATLAB. Assume that, there were 5 subjects and 6 speech samples for each subject as mentioned before. So that, *FEATURE_SAY* was assigned to 1500 since other samples were eliminated during pre-processing part. UART implemntation of speech recognition of the project is totally same with the implementation developed for face recognition project.

In the first application, database wouldn't be loaded to flash memory definitely. Hence, program counter couldn't jump to the test algorithm directly since data in the available flash addresses were empty and Fourier Transform implementation run for the feature matrix transmitted from MATLAB. For the following experiments,

Fourier Transform implementation would run for only the speech data, transmitted for test procedure unless flash memory was erased totally to update the content of database.

Fourier Transform implementation started by arranging data of feature matrix to be processed. Next, all elements of feature matrix were transformed due to Fast Fourier Transform (FFT) principles, respectively. By following to FFT of all elements based on 1024 points, new transformed matrix was saved to flash memory starting from 0x10000 address as described in Section 5.7.1. Test procedure was similar to the implementation of feature matrix. The only difference was in the size of matrix. Only one speech signal was processed during tests, while speech signals of all subjects in database were processed in training phase. Hence, the FFT code segment used in training phase was changed a little since the sizes were different.

The transformed form of test speech signal was compared to the data block one by one in order to extract the distances. Since FFT was applied based on 1024 points and there were 5 subjects in database, the size of the data block generated in training phase, was 5 x 1024. On the other hand, the size of the data block generated in test phase was 1 x 1024. After comparing each row vector and summing the whole difference vector elements, a vector was constructed which included the summation of differences between the test speech signal and the each speech signal in database. The minimum element in the resultant vector indicated to the owner of the speech signal. The result which indicated the best similarity was driven to the seven segment display to inform the end user.

5.8.3 Results of Speech Recognition System By DE2-70 FPGA

Speech recognition system project started by taking speech recording from the 4 subjects. The word used in speech recognition system was “bir”. Each subject said that word for 10 times. In order to create a speech database, the first 4 speech samples of each subject were used. The procedure followed explained in Section 5.9.3. The significant 1024 samples of each speech recording was used to create a a

feature matrix. Since total speech recording in this case was 20 (4 speech recording for each of 4 subjects), the size of resultant feature matrix was 1024 x 24. The remaining 5th, 6th, 7th, 8th, 9th and 10th speech recordings of each subject were used in test phase of the system. The test results of these speech recordings are shown in Table 5.12. This system is labelled as System I in the table below.

Table 5.13 Recognition table for System I (1024 significant samples for each speech data, 24 speech recording to test from 4 subjects)

Recognition Results		Test Speech Recordings					
		5th	6th	7th	8th	9th	10th
Subject Number	1st	-	+	-	+	+	-
	2nd	+	-	+	+	+	+
	3rd	+	+	+	-	+	+
	4th	-	+	+	+	-	+

The aim of the next combination was to observe the possible effects of change in database size. For this purpose, the only first speech recording of each subject was used to create a database. In this case, all remaining speech recordings could be used to test the speech recognition system. Test was applied by using the speech recordings of 3 subjects. Hence, the feature matrix had a smaller size than the feature matrix in System E. Since 1024 significant samples for each speech recording was used to construct the feature matrix, the size of the resultant feature matrix was 1024 x 3. The test results in this case are shown in Table 5.13. This system is labelled as System K.

Table 5.14 Recognition table for System K (1024 significant samples for each speech data, 27 speech recording to test from 3 subjects)

Recognition Results		Test Speech Recordings								
		2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Subject Number	1st	+	+	+	-	+	-	+	+	-
	2nd	+	+	+	+	-	+	+	+	+
	3rd	+	+	+	+	+	+	+	+	+

As shown in tables above, the results are similar to each other. In order to observe the results when the 4th subject was added to database, the system parameters were updated again. The feature matrix consisted of the first speech recording of each subject. Hence, all remaining speech recordings of each subject could be used to test the speech recognition system. 1024 significant speech samples were taken care to send to DE2-70 FPGA. So that, the size of resultant feature matrix was 1024 x 4. The test results in this case are shown in Table 5.14. This system is labelled as System L.

Table 5.15 Recognition table for System L (1024 significant samples for each speech data, 36 speech recording to test from 4 subjects)

Recognition Results		Test Speech Recordings								
		2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Subject Number	1st	+	+	+	-	+	-	+	+	-
	2nd	+	+	+	+	-	+	+	+	-
	3rd	+	+	+	+	+	+	+	+	+
	4th	+	+	-	-	+	+	+	+	+

In order to compare the results of speech recognition system, calculations in DE2-70 FPGA were moved to MATLAB. *fft* function of MATLAB is a widely used function for Fourier Transform Analysis implementations. The database developed for System L was assigned as input of *fft* function. By using the similar parameters with FPGA, Fourier Transform Analysis was applied to the feature matrix which included the first speech recording of each subject. All other speech samples were used to test the system. The test results in this case are shown in Table 5.15. This system is labelled as System M.

Since Fourier Transform Analysis solutions, developed in MATLAB and FPGA are parallel to each other, similar recognition rates are obtained at the end of tests. These results indicates that the main factor which affects the recognition rate of speech recognition system is the success of preprocessing functions, *keser* and *lowpasL*,

Table 5.16 Recognition table for System M (1024 significant samples for each speech data, 36 speech recording to test from 4 subjects)

Recognition Results		Test Speech Recordings								
		2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Subject Number	1st	+	+	+	+	+	+	+	+	+
	2nd	+	+	+	+	-	+	+	+	+
	3rd	+	+	+	+	+	+	+	-	+
	4th	+	-	+	+	+	+	+	+	+

5.8.4 General Overview to Speech Recognition System Performance

General performance of speech recognition system was observed by developing a test mechanism as described in Section 5.7.3.3. Database and test speech files were selected among 10 speech files for each of 5 subjects with the help of cross-validation technique. %60 of speech files were assigned as database files and %40 of speech files were assigned to apply the test procedure. In this way, 25 subsystems were generated and all of them were testes sequentially. Table 5.17 shows the general performance of speech recognition system in the format of confusion matrix.

Table 5.17 Recognition table of the speech recognition system in the format of confusion matrix

	1st	**2nd**	**3rd**	**4th**	**5th**
1st	%75	%8	%7	%0	%10
2nd	%20	%65	%0	%8	%7
3rd	%0	%0	%100	%0	%0
4th	%5	%0	%9	%82	%4
5th	%9	%0	%27	%0	%64

Table 5.17 shows that the recognition success of speech recognition system is subject specific. As an example, %75 of trials done with the files of 1st subject were recognized correctly. However, this success rate decreases upon to%64 for the 5th subject. The lowest performance of the speech recognition system was observed at the end of trials of the 5th person with an error rate of %36.

The general performance of speech recognition system is %77.2. It is lower that the success rates which was obtained at the end of unimodal face recognition systems. The root cause can be the recognition method used to recognize the speech files.

CHAPTER SIX

FPGA BASED MULTIBIOMETRIC RECOGNITION SYSTEM

6.1 Multibiometric Recognition System Implementation on DE2-70

Multibiometric recognition system implementation is basically to create a system which combines the face and speech recgonition system described in Chapter Five. The main purpose of developing a multibiometric recognition system is to observe the change in the general performance of the system.

It is required some software modifications in both MATLAB and FPGA source code for multibiometric recognition system. As described in Section 5.7.3.1, the face images of ORL Face Database were processed respectively in order to create a feature matrix. Basically, a local window was applied to each of database face images after resizing them. The mean of each window became an element of the feature matrix. Since the size of resized images was 40x40, 13x13 matrix was created for each image by windowing and mean processes. 30 face images were totally used in face database by collecting 5 face images for each of 6 subjects. Hence, a feature matrix whose size was 169x30, was constructed succesfully. In a similar way, speech data files in database were processed to create a feature matrix which would be used in speech recognition system. There were 5 subjects in speech database and one speech data file would be used for each subject. After preprocessing and rearranging each speech data, 1024 samples were generated. As a result, a feature matrix in size of 5x1024 was created. Feature matrices belonging to face images and speech data files were combined in order to create a whole feature matrix of multibiometric recognition system. The resultant feature matrix was sent to FPGA over RS-232.

Software modifications in FPGA were basically to combine the face and speech recognition source codes. Incoming feature matrix was divided into two parts which presented the feature matrix of face and speech recognition systems. Training phase of face and speech recognition systems were processed sequentially. The resultant

TempMatYtrain and *TempMatPcaBasis* matrices for face recognition system and *TempMatFftDatabase* for speech recognition system were saved to the related addresses of the flash memory.

Recognition phase of multibiometric recognition system were based on the principles of the training phase. Test face image and test speech data file were acquired and a complete test matrix was created. It was sent to FPGA to be recognized over RS-232. *TempMatYtrain* and *TempMatPcaBasis* matrices were reloaded for face recognition system and *TempMatFftDatabase* matrix was reloaded for speech recognition system from flash memory. Test procedure was applied seperately for the incoming test face and test speech data.

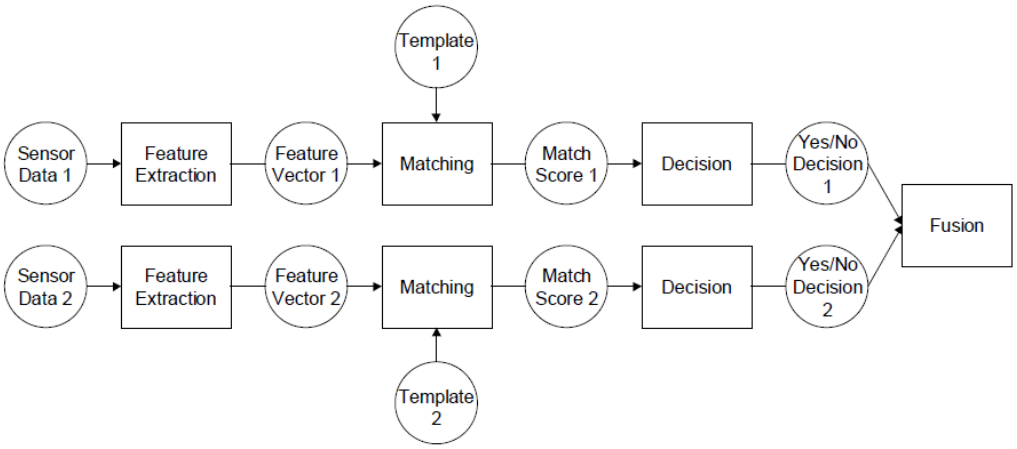


Figure 6.1 Multibiometric recognition system based on "fusion at decision level" scenario

Decision mechanism of multibiometric recognition system was based on the scenario of "fusion at decision level". The results of face and speech recognition systems were compared. If both of results indicated the correct subject, the whole result of multibiometric recognition system was correct and printed to the console as a multibiometric recognition result. Otherwise, the system returned an error message which indicates to the end user that the failure of authentication.

The source code of multibiometric recognition system which includes some differences. MATLAB and FPGA source codes of this system are in the Appendix with the name of "6_1_Implementation_of_ Multibiometric_Recognition_System".

6.2 General Overview to Multibiometric Recognition System Performance

General performance of multibiometric recognition system was achieved in the same way with the method described in Section 5.7.3.3 and Section 5.8.4. 10 face images for each of 5 people from ORL Face Database and 10 speech data files collected from each of 5 people were used to generate a confusion matrix. At that point, cross-validation technique run by the command of *crossvalind* in MATLAB in order to select which files would be database face and speech files. %60 of face images and speech files were selected as database face images and speech files, while the remaining files were used to test the system. Cross-validation implementation provided 25 different subsystems which different face images and speech files for 5 subjects were labeled as database and test face images. Table 6.1 shows the general performance of multibiometric recognition system in the format of confusion matrix.

Table 6.1 Recognition table of the multibiometric recognition system in the format of confusion matrix

	1st	**2nd**	**3rd**	**4th**	**5th**
1st	%72	%8	%7	%3	%10
2nd	%20	%62	%1	%10	%7
3rd	%0	%0	%100	%0	%0
4th	%5	%0	%12	%79	%4
5th	%9	%0	%29	%1	%61

Table 6.1 shows that the combination of face and speech recognition systems decreases the recognition rates of unimodal recognition systems. %72 of files of 1st subject were recognized correctly. In other words, both of face and speech database files of 1st subject were recognized correctly in the rate of %72. However, %8 of trials of 1st subject were recognized as 2nd subject and %7 of trials of 1st subject were recognized as 3rd subject. All trials by using the files of 3rd subject were recognized correctly. The lowest recognition rate belonged to 2nd subject with an error rate of %38. The most of this erroneous recognition was caused by the speech recognition part of multibiometric recognition system.

The general performance of multibiometric recognition system is %74.8. It is lower that the success rates which are observed at the end of unimodal face and speech recognition systems. It can be a result of selected fusion scenario.

CHAPTER SEVEN

CONCLUSIONS

7.1 Overview of the Project

This project basically aims to recognize the people based on two biometric parameters, face and speech. There are some common parts in the recognition system. The first process in face recognition system is to create a database which includes face image samples of some subjects to be recognized. This phase is called as training phase. The face images of subjects are prepared to be proccesed in FPGA. For this purpose, face images are resized and a matrix is created which includes numerical data of images in MATLAB. This matrix is sent to FPGA sequentially. In FPGA, a common feature extraction method, PCA, is applied to incoming feature matrix. The resultant data is saved to memory of FPGA. From now, the system is ready to accept the face images which will be used in comparison. After the test face image is acquired and processed in MATLAB, the resultant vector belonging to test face image is sent to FPGA, too. PCA is applied to test face image. The resultant data is compared to matrix which belongs to database face images. The comparison result return the owner of the test image. This phase is called as recognition phase.

The speech recognition part of the system starts in a similar way with face recognition system. The speech samples belonging to subjects in database are processed in MATLAB. Hence, a matrix is created which is ready to be processed in FPGA. Unlike face recognition system, Fourier Transform Analysis is implmented in FPGA for speech recognition system. The speech samples of subjects in database are analyzed using Fourier Transform and saved to the memory of FPGA. Similarly, speech sample of test subject is accepted by MATLAB and processed to create a vector to send to FPGA. After Fourier Analysis is applied to test speech sample, the resultant vector is compared to matrix which is already saved in FPGA. The result of comparison describes who is the owner of test speech sample.

Multibiometric recognition systems include multiple sources of information to establish the identity. Unimodal face and speech recognition systems are combined after developing them seperately to observe the effects. Multimodal biometric system information reconciliation can occur in some fusion scenarios. “Fusion at decision level” scenario is used to decide if the whole result of the system was accepted or not.

The general performance of the unimodal and multimodal recognition systems is observed by generating a confusing matrix. The face images and speech files are processed by cross-validation technique and 25 subsystems are created. The total accuracy is %96.4 for face recognition system, %77.2 for speech recognition system and %74.8 for multibiometric recognition system. Since the results of face and speech recogntiion systems are AND’ed. Hence, there is an amount of decrement in the success rate of multibiometric recognition system when compared to previous systems.

7.2 Advantages and Disadvantages of the System

FPGAs are future oriented building blocks which offer perfect customization. FPGA components have usable sizes with acceptable prices. During development of the whole system, many major changes occurred in design steps. But application-specific integration of IP cores in FPGA prevented time and cost consuming. If the system had been developed by microcontroller based boards, redesign of board would cause unrequired cost increment. Also, FPGAs are adaptable devices. In other words, this system can be converted to any other application, since their configuration is specified in an abstract hardware description language. However, microcontroller based embedded systems can’t allow to change the interest area, after design is completed.

On the other hand, face and speech recognition system provided us to make an entrance to an environment which would be much popular in the next years. High

recognition rates of the system allow the applications areas which requires high security to own a relatively inexpensive system.

Since two separate environment, MATLAB and FPGA, are used, the system integration to real life can be difficult. FPGA device can be integrated o any system. However, MATLAB integration may cause some problems, since it requires a high-capability computer. Instead of using two different platforms, making all data acqusition steps in FPGA would be more suitable. Also, FPGAs in market are limited-edition. Hence, designers don't have many options to select. The other disadvantage of this design is the working speed, caused by serial running of FPGA. If the system run in parallel mode which is an attractive specifications of FPGAs, the total process time would be shorter. Although FPGAs have cost advantages in general, the total cost of this project is in high levels. Because, all development steps were applied on a prototype board, DE2-70 FPGA Board. But the available prototype was able to develop new systems with its specifications. It was a reasonable prototype board to develop new systems with its specifications for univiersities, research&development centers etc.

7.3 Troubleshooting

Two FPGA boards have been used during the development of the project, UP3 Education Kit and DE2-70 FPGA Board. Some characteristics of the boards created some blocking points during the project. Since UP3 was an education kit, it had some limitations. For example, design in UP3 Education Kit was based on the counting clock signal. However, clock signal of the board was unstable. n^{th} pulse was assigned to make a process in source code, but it couldn't. Because, in the beginning of the generation clock signal, there were no pulse for a time of two or three period time. The first pulse was able to occur in the 4^{th} period of clock signal. The reason of the problem could be understood after analyzing in simulation tool of Quartus. In order to resolve the problem, operations have been adjuested based on the shifted version of the clock signal in UP3 Education Kit. The other problem of UP3 Education Kit was unsufficient number of gate. In early steps of projects, gates had been assigned

as memory elements. However, the kit didn't have sufficient number of gates to save data matrices created during run-time of the system. Hence, the development in UP3 Education Kit was cancelled and a new board, DE2-70 Board, replaced it.

The main disadvantage of DE2-70 Board was stability problems. By the way of example, in any time of run-time, the system returned insignificant results. While previous trial with same parameters returned the correct result, the next trial returned the wrong result even if there wasn't any change in the system. Matrix allocation for all variables in source code resolved that problem. A memory location was allocated for variables before using. After it was used, the memory location was cleaned out by *free_vector* and *free_matrix.* Hence, the stability problem caused by memory was fixed.

7.4 Cost Analysis

The system design in FPGA is a new concept for many designers. Thus, it is difficult to select the suitable board both in terms of cost and capabilities. DE2-70 package was bought for 500 YTL from Çizgi Elektronik, İstanbul. The package included DE2-70 board and other tools like USB cable for FPGA programming and control, CD-ROMs containing Altera's Quartus® II Web Edition and the Nios® II Embedded Design Suit Evaluation Edition software. System development required also MATLAB environment which was already installed in the computers. As a result, system development in FPGA was so economical.

7.5 Future Works

As mentioned in Section 6.1, face and speech samples were acquired and processed by MATLAB before transferring to FPGA. One of the future works is to implement these steps in FPGA like other procedure. On the other hand, feature extraction methods in the system can be replaced with other methods. Independent Component Analysis (ICA), Linear Discriminant Analysis (LDA), Linear Predictive

Coding, Gabor Filters etc. can be implemented instead of PCA and Fourier analysis. Also the fusion scheme can be reorganized to get more successful recognition results.

Since DE2-70 FPGA Board is high capacity board. It includes LCD module, VGA-out connector which can be used to add new features to the system. Also, the comparison method used in the project was based on calculating the absolute distances and comparing them. Instead of this method, other comparison methods can be developed in the recognition phases of systems.

REFERENCES

Abdel-Ghaffar E. A., Alam M. E., Mansour H. A. K. & Abo-Alsoud M. A. (2008). A secure face recognition system. *Computer Engineering & Systems, 2008. ICCES 2008. International Conference.* 95-100. Retrieved August, 2010, from IEEE Xplore database.

Altera (2007). *Introduction to the Quartus II Software Version 10.0*. Altera Corporations

Balakrishnama S. & Ganapathiraju A. (1998). *Linear discriminant Analysis – A Brief Tutorial.* Retrieved March, 2010. from http://www.music.mcgill.ca/~ich/classes/mumt611_07/classifiers/lda_theory.pdf

Brown S. & Rose J.(1996). Architecture of FPGAs and CPLDs : A tutorial. *IEEE Design and Test of Computers, 13,* 42-57

Delioğlu M. Y.(2007). *Feature Extraction For Text-Dependent Spekaer Verification (1st Ed.)*(1-6). Dokuz Eylül University Department of Electrical and Electronics Engineering Undergraduate Thesis

Deschamps J. P., Bioul G. J. A.&Sutter G. D.(2006). *Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems.* Canada : John Wiley&Sons, Inc.

Feldman J.(2007), *The Fourier Transform.* Retrieved June, 2010, from http://www.math.ubc.ca/~ feldman/m267/ft.pdf

Hyvarinen A. (1999). Survey on independent component analysis. *Neural Computing Surveys 2* : (94-128). Helsinki University of Technology, Finland

Juang B. H. & Lawrence R. (2004). *Automatic Speech Recognition – A Brief History of the Technology Development, 50* (2), 637-655. Retrieved May, 2010, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.5614&rep=rep1&type=pdf

Kim K.(2003). *Face Recognition using Principle Component Analysis.* Retrieved May, 2010, from http://www.umiacs.umd.edu/~knkim/KG_VISA/PCA/FaceRecog_PCA_Kim.pdf

Li S. Z & Jain A. K. (Eds.) (2004). *Handbook of Face Recognition.* New York : Springer Science+Business Media, Inc.

Madi R., Lahoud R., & Sawan B. (2006). *Face Recognition on FPGA,* Beirut : American University of Beirut Press.

Ross A., Jain A. K. (2004). Multimodal biometrics : An overview. *12th European Signal Processing* Conference 1221-1224. Retrieved August, 2010, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.6907&rep=rep1&type=pdf

SLS (2004). *UP3 User Manual Version 0.1*, SLS Corporations.

Štruc V., Gajšek R. & Pavešic N. (2009). Principal gabor filters for face recognition. In *Proceedings of the 3rd IEEE international conference on Biometrics: Theory, applications and systems.* (113-118). New York : IEEE Press

Terasic (2009). *DE2-70 User Manual Version 1.08*. Terasic Technologies.

Thorat S.B., Nayak S.K. & Dandale J.P.(2010). Facial recognition technology : An analysis with scope in India. *International Journal of Computer Science and Information Security*,*8* (1), 325-330.

Tran D., To T., Huynh T. & Nguyen P. (2010). Designing a hardware accelerator for face recognition using vector quantization and principal component analysis as a component of SoPC. *Electronic Design, Test and Application, 2010. DELTA '10. Fifth IEEE International Symposium.* 82-86. Retrieved August, 2010, from IEEE Xplore database

Yoo Y. (2001). *Tutorial on Fourier Theory*. Retrieved August, 2010, from http://www.cs.otago.ac.nz/cosc453/student_tutorials/fourier_analysis.pdf

APPENDIX

An *"Appendix CD*" is prepared which contains all MATLAB files, VHDL files and Nios II system designs that are used in this thesis. The folder names are dedicated to section numbers to reach source codes easily. Source code availability is mentioned in each section. As a remember, the content of "*Appendix CD*" is also given in the following with section name and corresponding folder name in the "*Appendix CD*". Appendix files and related sections are described in the following:

Section 5.4.3 The First Projects for Data Compression and Data Transmission
- 5_4_3_UART_Transmitter
- 5_4_3_UART_Receiver
- 5_4_3_UART_Array_Transmitter
- 5_4_3_Internal_Database_And_Test
- 5_4_3_Database_and_Test_sent_from_MATLAB

Section 5.5 Face Recognition System By UP3 Education Kit
- 5_5_PCA_MATLAB

Section 5.7.1 FPGA Part Implementation
- 5_7_1_Face_Recognition_Configuration_Sw

Section 5.7.2 Preliminary Implementations In MATLAB Part
- 5_7_2_fdmex_example

Section 5.7.3.1 Final Preliminary Implementations In MATLAB Part
- 5_7_3_1_Highest_Recognition_Rate_For_Face_Recognition

Section 6.1 Multibiometric Recognition System Implementation on DE2-70
- 6_1_ Implementation_of_ Multibiometric_Recognition_System