People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of the Requirements for the Degree of

MASTER

In **Electrical and Electronic Engineering**

Option: **Computer Engineering**

Title:

FPGA-BASED REAL-TIME SPEECH RECOGNITION SYSTEM

Presented by:

- **LALAOUI Dyhia**
- **ABDERRAHMANE Nassim**

Supervisor:

Mr. R.NAMANE

Registration Number:……../2016

Dedications

I would like to dedicate this modest work to all whom I love.

Special thanks to my dearest parents, sisters and brother for their support and encouragement.

A. Nassim

I would like to dedicate this modest work to my dearest family especially my beloved father and mother and my lovely husband for their valuable help and moral support.

To all my friends and colleagues.

L. Dyhia

Acknowledgement

It is a great pleasure to thank our supervisor Mr. NAMANE Rachid for his devotion, orientation, advice and encouragement during the accomplishment of our project.

Then we owe a great many thanks to a great many people who helped during our work.

We owe a debt of gratitude to the teachers of IGEE being a necessary part in our success during our five years of studies at the institute, for their care and support.

ABSTRACT

The main objective of this project is to design and implement a real time speech recognition system using a Field Programmable Gate Array (FPGA). Our system will be based on the Mel Frequency Cepstral Coefficients (MFCC) and a distance-based matching algorithm which fit very well the targeted system. The model will be implemented on the DE2 FPGA board including both SOPC and non-SOPC modules. Spoken words will be entered via a microphone connected to the audio input of the board. The recognized words will be displayed on the LCD module of the board. The obtained system can be used in several applications such as mobile robot control with voice command.

Contents

Contents

Contents

List of Figures

List of Figures

List of Tables

Nomenclature

ASR: Automatic Speech Recognition

CODEC: enCOder/DECoder

CAD: Computer Aided Design

DAC: Digital to Analogue Converter

DCT: Discrete Cosine Transform

DFT: Discrete Fourier Transform

DSP: Digital Signal Processing

EOP: End Of Packet

FFT: Fast Fourier Transform

FPGA: Field Programmable Gate Array

FSM: Finite State Machine

I2C: (IIC) Inter Integrated Circuit

IP: Intellectual Property

LCD: Liquid Crystal Display

LE: Logic Element

LED: Light Emitting Diode

MFCC: Mel Frequency Cepstral Coefficients

MSB: Most Significant Bit

PIO: Parallel Input/Output

PLL: Phase-Locked Loop

RAM: Random Access Memory

ROM: Read Only Memory

SDA: Serial bi-directional data line

SDL: Serial bi-directional clock line

SDRAM: Synchronous Dynamic RAM

SoC: System-on-a Chip

SOPC: System On A Programmable Chip

Nomenclature

SOP: Start Of Packet

SR: Speech Recognition

VHDL: VHSIC Hardware Description Language

VHSIC: Very High Speed Integrated Circuit

A very challenging area of research is making a machine understand natural language, a fact that attracts a lot of researchers in recent years to develop speech recognition tools.

Speech recognition is a software or hardware that is capable to process a speech signal as an input and gives as a result a recognized words or sentences, which can be used in many applications like controlling a robot or simply speech to text applications.

It is of great importance not only in practical application but scientific research. The research on speech recognition technology mainly concentrates on two aspects: One is the software running on computer, the other is embedded systems. The advantages of Embedded Systems are high-performance, convenience, cheap and they have huge potential for development.

The classical front end analysis in speech recognition is a spectral analysis which parameterizes the speech signal into feature vectors. The most popular set of feature vectors used in recognition systems is the Mel Frequency Cepstral Coefficients (MFCC) [25]. They are based on a standard power spectrum estimate which is first subjected to a log-based transform of the frequency axis; it results in a spectral representation on a perceptually frequency scale, based on the response of the human perception system. After, they are decorrelated using a modified discrete cosine transform, which allows an energy compaction in its lower coefficients [25].

In this project, we desire to implement an FPGA-based real-time speech recognition system. The system will be configured on our available Altera DE2 FPGA-based board. The system processes a word spoken from head-styled microphone and outputs it on the inboard LCD accessory. Our implementation includes mainly two parts: hardware and software. The different components of our hardware circuit are designed based on VHDL hardware description language before downloading them into the Cyclone II FPGA device. The software part is developed using C programming language and executed on a soft processor generated by a computer aided design (CAD) software, and which completes the work done by the hardware circuit.

In the hardware part, we implement an audio codec controller that is responsible of the pre-emphasis and sampling, an FFT module for transforming the time domain signal into frequency domain, and finally a NIOS soft processor that will be used to execute the software program which consists mainly of the MFCC and recognition algorithms.

MFCC technique is used to extract feature vectors of the spoken word; this latter is composed of FFT, Mel filter bank, DCT and Mel spectrum. The first component is done in hardware part, the remaining are achieved in software part. The complexity of these components made us decide to develop them in software.

In the software part, a C language program running on the NIOS soft processor used to perform the MFCC (Mel filter bank, DCT and Mel Spectrum) for feature extraction and the matching algorithm for training and recognition.

Our system design is based on the FPGA because of the advantages of short development cycle, low-cost design and low-risk. In recent years, FPGA has become the key components in high-performance digital signal processing systems used in digital communication, network, video and image fields [14].

The report is organized into three chapters:

Chapter I is an introduction to the field of speech recognition, where we have covered the theoretical part of our project, from speech theory to hardware used in our realization.

Chapter II 'Hardware and software implementation' deals with the hardware and software implementation descriptions of our system. The hardware part is divided into three main blocks: audio codec, FFT Module, and NIOS system. This part gives explanation of the different hardware blocks and how they are related to each other to form the overall hardware system. The last part of this chapter deals with the software description of the implemented feature extraction, training and recognition.

Chapter III 'Result and Discussion' represents the obtained results and their analysis.

The report is ended with a general conclusion and some suggestions for future works.

CHAPTER I: THEORETICAL BACKGROUND

Introduction

As a new suitable approach of human-machine interaction, speech recognition is widely applied to many speech products. The crucial objective of speech recognition is to make machine understand natural language. In this chapter we will introduce the theoretical background of our project.

The chapter is divided into two main parts: the first goes through the speech recognition basics and the second part focus on the means used, i.e., hardware and software.

1. Speech Recognition

Speech Recognition can be defined as the process by which a computer (or other type of machine) identifies spoken words (speech signal) using different algorithms implemented in a computer system. The goal of speech recognition is to interact with deferent machines.

Generally speaking, an Automatic Speech Recognition (ASR) system can be organized in two blocks: the feature extraction and the modeling stage (or feature matching)[9]. As described in figure 1.1.

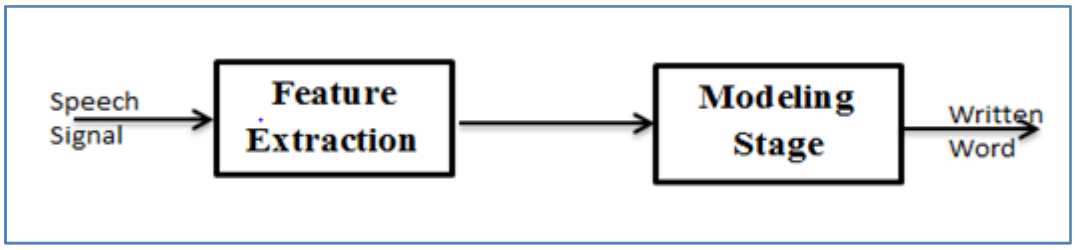

Figure1.1: Block Diagram of Speech Recognition

1.1 Speech Signal

Speech signal is an analog representation of the vocalized form of communication based upon the syntactic combination of lexical and names that are drawn from very large vocabularies.

1.1.1 Type of Speech

Speech recognition system can be classified into different classes by describing what type of utterances they can recognize.

- **Isolated Word**

Isolated word recognizes attain usually require each utterance to have quiet on both side of sample windows. It accepts single words or single utterances at a time .This is having “Listen and Non Listen state”. Isolated utterance might be better name of this class [21].

- **Connected Word**

Connected word systems are similar to isolated words but allow separate utterance to be “run together minimum pause between them [21].

- **Continuous Speech**

Continuous speech recognizers allow user to speak almost naturally, while the computer determines the content. Recognizer with continuous speech capabilities are some of the most difficult to create. They utilize special method to determine utterance boundaries [21].

- **Spontaneous Speech**

At a basic level, it can be thought of as speech that is natural sounding and not rehearsed. An ASR System with spontaneous speech ability should be able to handle a variety of natural speech feature such as complete sentences [21].

1.2 Speech Recognition Basics

- **Utterance**

An utterance is the vocalization (speaking) of a word or words that represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences [26].

- **Speaker Dependence**

Speaker dependent systems are designed around a specific speaker. They generally are more accurate for the correct speaker, but much less accurate for other speakers. They assume the speaker will speak in a consistent voice and tempo. Speaker independent systems are designed for a variety of speakers. Adaptive systems usually start as speaker independent systems and utilize training techniques to adapt to the speaker to increase their recognition accuracy [26].

- **Vocabularies**

Vocabularies (or dictionaries) are lists of words or utterances that can be recognized by the SR system. Generally, smaller vocabularies are easier for a computer to recognize, while larger vocabularies are more difficult [26].

- **Accurate**

The ability of a recognizer can be examined by measuring its accuracy - or how well it recognizes utterances. This includes not only correctly identifying an utterance but also identifying if the spoken utterance is not in its vocabulary. Good ASR systems have an accuracy of 98% or more! The acceptable accuracy of a system really depends on the application [26].

- **Training**

Some speech recognizers have the ability to adapt to a speaker. When the system has this ability, it may allow training to take place. An ASR system is trained by having the speaker repeat standard or common phrases and adjusting its comparison algorithms to match that particular speaker. Training a recognizer usually improves its accuracy [26].
Training can also be used by speakers that have difficulty speaking, or pronouncing certain words. As long as the speaker can consistently repeat an utterance, ASR systems with training should be able to adapt [26].

1.3 Feature Extraction

Feature extraction is a process that extracts valid feature parameters from an input speech signal. Even if the same word is spoken, no two speech instances can produce the same speech waveform. The reason for this phenomenon is that the speech waveform includes not only speech information but also the emotional state and tone of the speaker. Therefore, the goal of speech feature extraction is to extract feature parameters that represent speech information. Further, this process is a part of compressing speech signals and modeling the human vocal tract. The feature parameters are devised to represent the phonemes accurately for speech recognition [22]. MFCC is commonly used for the above mentioned feature extraction.

1.3.1 Mel Frequency Scale and Coefficients

The most popular spectral based parameter used in recognition approach is the Mel Frequency Cepstral Coefficients called MFCC [13]. MFCCs are coefficients, which represent audio, based on perception of human auditory systems.

MFCC is based on human hearing perceptions which cannot perceive frequencies over 1Khz. In other words, MFCC is based on known variation of the human ear's critical bandwidth with frequency. MFCC has two types of filter which are spaced linearly at low frequency below 1000 Hz and logarithmic spacing above 1000Hz. A subjective pitch is present on Mel Frequency Scale to capture important characteristic of phonetic in speech [13].

The overall process of the MFCC is shown in Figure 1.2.

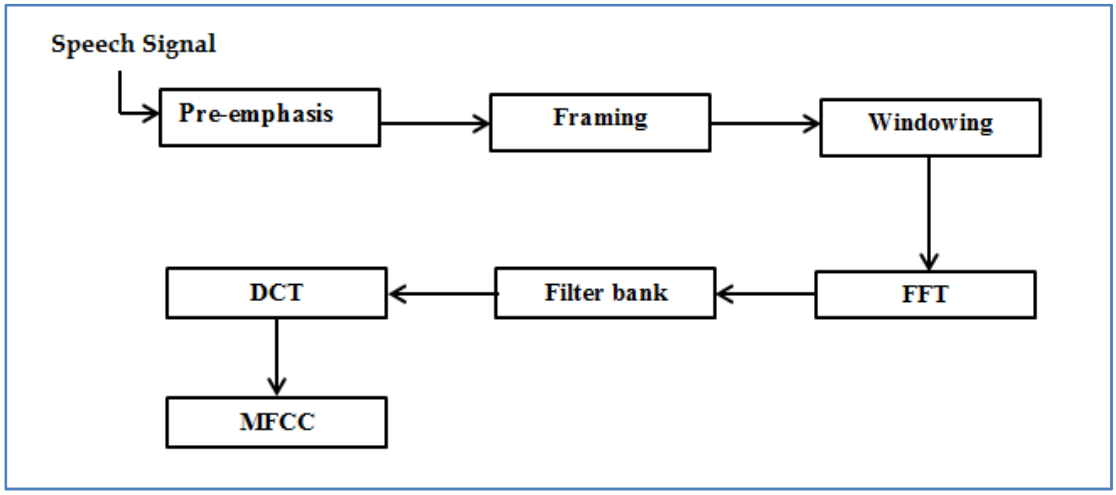

Figure1.2: MFCC Block Diagram

- **Pre-emphasis**

This step processes the input signal through a pre-emphasis filter which has high-pass filter characteristics. This process will increase the energy of signal at higher frequency [13].

- **Framing**

The process of segmenting the speech samples obtained from analog to digital conversion (ADC) into a small frame with the length within the range of 20 to 40 msec. The voice signal is divided into frames of N samples [13].

- **Windowing**

At the edges of each frame, there are discontinuities in the input signal that contain unnecessary information. In order to minimize the discontinuities at the edges of the frames, each frame is multiplied with the window coefficients [13].

Hamming window is used as window shape by considering the next block in feature extraction processing chain and integrates all the closest frequency lines [13].

- **Fast Fourier Transform (FFT)**

In order to extract the feature parameters of the input speech, the FFT algorithm can be applied to convert the time domain into the frequency domain to figure out the frequency characteristics of the input. In the time domain, a speech signal has discrete non-periodic features. Through FFT, which converts the time domain into the frequency domain, a speech signal is transformed into a continuous periodic signal [13].

- **Mel Filter Bank Processing**

The frequencies range in FFT spectrum is very wide and voice signal does not follow the linear scale. The human ear responds non-linearly to a speech signal. When the speech recognition system performs a non-linear process, it improves the recognition performance. By applying a Mel-filter bank, we can obtain a non-linear frequency resolution. The Mel Filter bank method is widely used in the speech recognition process [13].

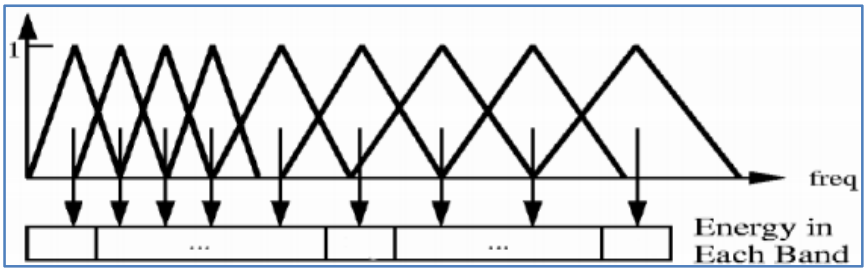

Figure 1.3: Mel Scale Filter Bank [13].

The figure 1.3 shows a set of triangular filters that are used to compute a weighted sum of filter spectral components so that the output of process approximates to a Mel scale. Each filter's magnitude frequency response is triangular in shape and equal to unity at the center

frequency and decreases linearly to zero at the center frequency of two adjacent filters. Then, each filter output is the sum of its filtered spectral components.

- **Discrete Cosine Transform**

The logarithm and discrete cosine transform (DCT) of the Mel-Filter bank energy are computed to extract the required minimum information.

This process is to convert the log Mel spectrum into time domain using Discrete Cosine Transform (DCT). The result of the conversion is called Mel Frequency Cepstrum Coefficient. The set of coefficient is called acoustic vectors. Therefore, each input utterance is transformed into a sequence of acoustic vector [13].

1.4 Matching Technique

The matching technique is used to compare a spoken word to referenced ones in a data base. These words have to be in vector representation form with limited number of entrance, say N. The matching algorithm compares the entries of the spoken word's vector to the referenced one element by element till N, by calculating the difference between them. Then it takes the average of the differences and compares it to a referenced threshold value. Whenever, it finds that average less than the threshold value it senses that it has found a spoken word.

2. Development and Educational Board and Quartus II CAD Software

DE2 is a board conceived for the practical implementation of digital and mixed-signal circuits, even quite complex ones. It includes several I/O devices and interfaces, from simple ones (switches, pushbuttons, LEDs, seven segment displays) to complex devices (LCD matrix display, Ethernet network interface, USB 2.0, SD memory cards, analog audio I/O, analog video input, VGA video output …etc.) [8].

The core of DE2 is an FPGA (Field Programmable Gate Array), composed by more than 30.000 Logic Elements (LE). DE2 may host simple introductory projects, and sophisticated ones that may include one or more microcomputers [8]. The figure 1.4 represents DE2 Board top view.

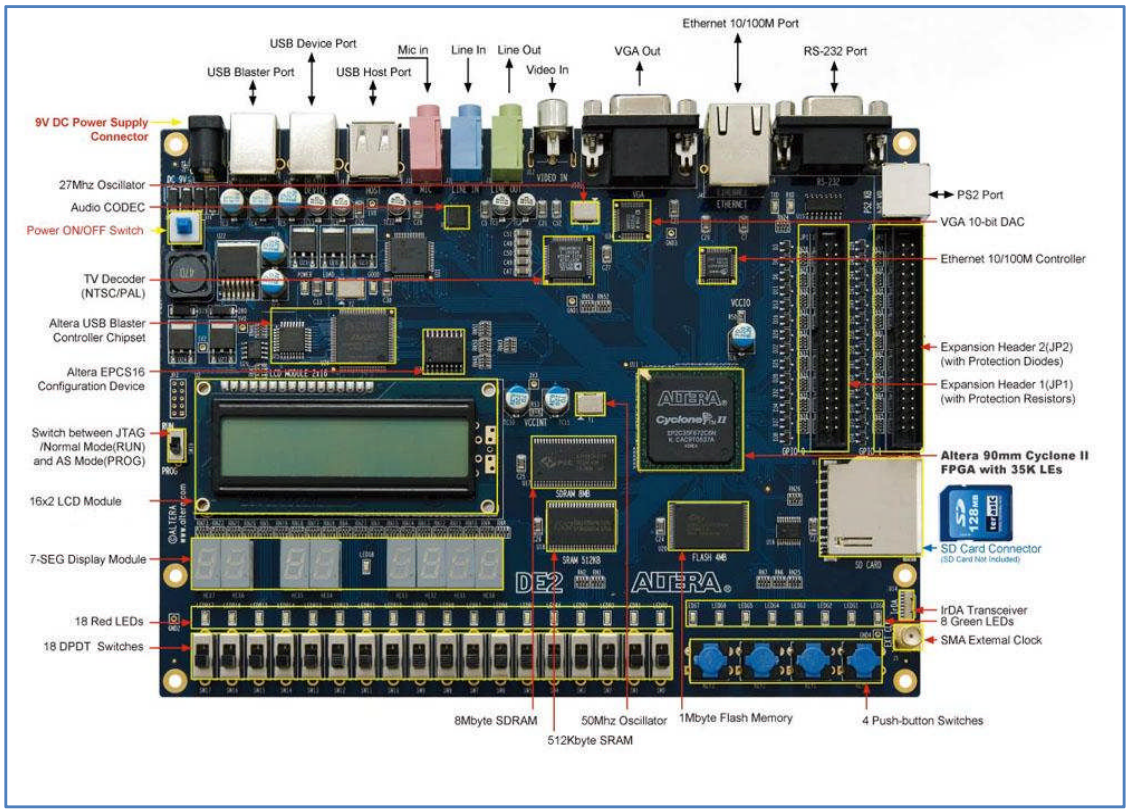

Figure 1.4: DE2 Board Top View [8].

2.1 Audio Codec

The DE2 board provides high-quality 24-bit audio via the Wolfson WM8731 audio CODEC (Encoder/Decoder) [4]. This chip supports microphone-in, line-in, and line-out ports, with a sample rate adjustable from 8 kHz to 96 kHz. The WM8731 is controlled by a serial I^2C bus interface, which is connected to pins on the Cyclone II FPGA [20].

This chip is used to sample and digitalize the audio signal using the Analogue to digital conversion (ADC).

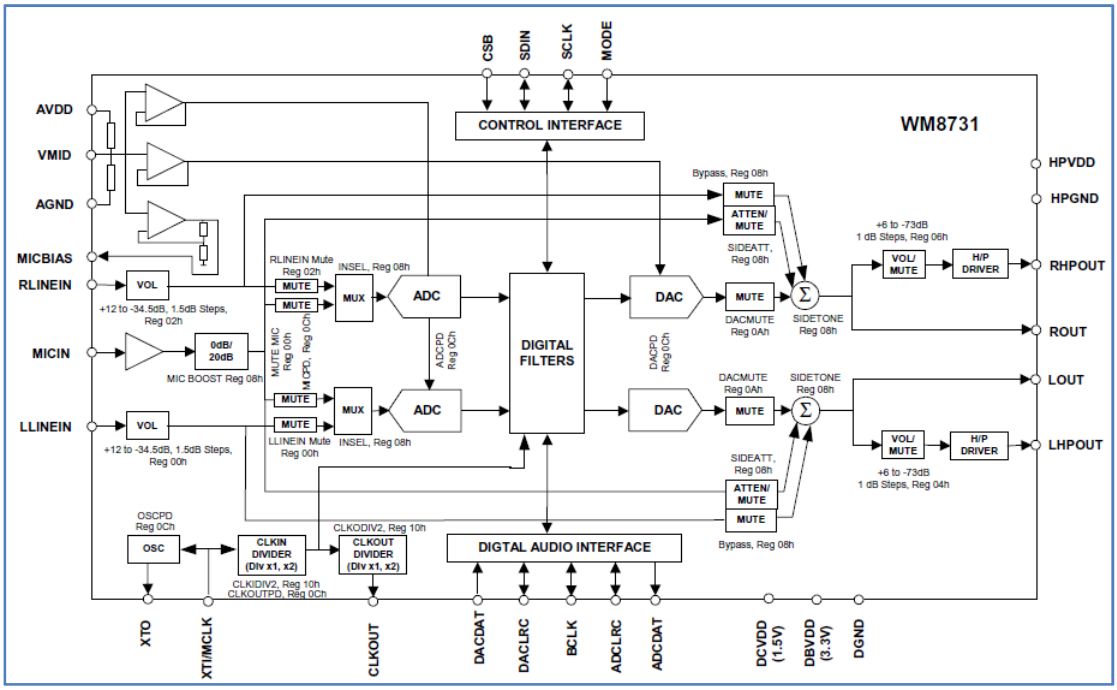

Figure 1.5: Functional Block Diagram of WM8731 Audio CODEC [20].

The digital audio interface takes the data from the internal ADC and places it on the ADCDAT output. ADCDAT is the formatted digital audio data stream output from the ADC digital filters with left and right channels multiplexed together. ADCLRC is an alignment clock that controls whether Left or Right channels data is present on ADCDAT lines. ADCDAT and ADCLRC are syncronised with the BCLK signal with each data bit transition signified by a BCLK high to low transition. BCLK may be an input or an output dependent on whether the device is in master or slave mode [20].

There are four digital audio interface formats accommodated by the WM8731/L. Refer to the electrical characteristics section for timming information [20].

- Left justified mode is where the MSB is available on the first rising edge of BCLK following ADCLRC transition [20].
- I^2S Mode is where the MSB is available on the second rising edge of BCLK following an ADCLRC transition [20].
- Right Justified Mode is where the LSB is available on the rising edge of BCLK preceding an ADCLRC transition, yet MSB is still transmitted first [20].
- DSP Mode is where the left channel MSB is available on either the first or the second rising edge of BCLK following a LRC transition high. Right channel immediately follows left channel data [20].

All these modes are MSB first and operate with data width of 16 to 32 bits, but the right justified mode does not support 32 bits [20].

The Audio Codec has 11 registers with 16 bits per register (7 bit address + 9 bits of data). These can be controlled using either the 2 wire or 3 wire MPU interface. The complete register map is shown in table 1.1.

Table 1.1: Mapping of Program Register [20].

REGISTER	B 15	B 14	B 13	B 12	B 11	B 10	B 9	B8	B7	B6	B5	B4	B3	B2	B1	B0
R0 (00h)	0	0	0	0	0	0	0	LRIN BOTH	LIN MUTE	0	0	LINVOL				
R1 (02h)	0	0	0	0	0	0	1	RLIN BOTH	RIN MUTE	0	0	RINVOL				
R2 (04h)	0	0	0	0	0	1	0	LRHP BOTH	LZCEN	LHPVOL						
R3 (06h)	0	0	0	0	0	1	1	RLHP BOTH	RZCEN	RHPVOL						
R4 (08h)	0	0	0	0	1	0	0	0	SIDEATT		SIDETONE	DAC SEL	BY PASS	INSEL	MUTE MIC	MIC BOOST
R5 (0Ah)	0	0	0	0	1	0	1	0	0	0	0	HPOR	DAC MU	DEEMPH		ADC HPD
R6 (0Ch)	0	0	0	0	1	1	0	0	PWR OFF	CLK OUTPD	OSCPD	OUTPD	DACPD	ADCPD	MICPD	LINEINPD
R7 (0Eh)	0	0	0	0	1	1	1	0	BCLK INV	MS	LR SWAP	LRP	IWL		FORMAT	
R8 (10h)	0	0	0	1	0	0	0	0	CLKO DIV2	CLKI DIV2	SR				BOSR	USB/ NORM
R9 (12h)	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	ACTIVE
R15(1Eh)	0	0	0	1	1	1	1	RESET								
	ADDRESS							DATA								

2.1.1 I²C Bus

The I²C (Inter Integrated Circuit) bus is a simple bi-directional serial bus that supports multiple master and slaves. It consists of only two lines; a serial bi-directional data line (SDA) and a serial bi-directional clock line (SCL) [23].

2.1.2 I²C Protocol

The I²C bus is idle when both SCL and SDL are at logic 1 level. The master initiates a data transfer by issuing a START condition, which is high to low transition on the SDA line while the SCL line is as high as shown in figure 1.6. The bus is considered to be busy after the START condition. After the START condition, a slave address is sent out on the bus by the master. The address is 7 bits long followed by an eighth bit which is a data direction bit (R/W) [23].

The master, who is controlling the SCL line, will send out the bits on the SDA line, one bit per line can be changed only when the SCL line at a low.

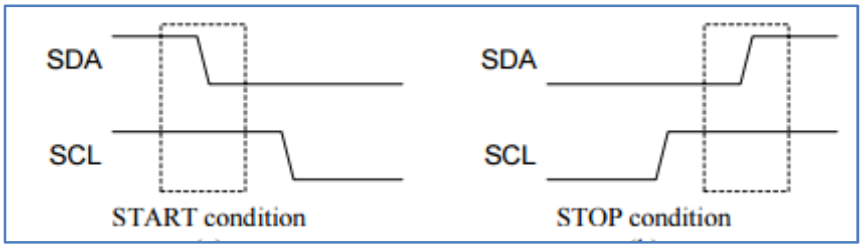

Figure 1.6: Start and Stop Condition of I^2C Bus [23].

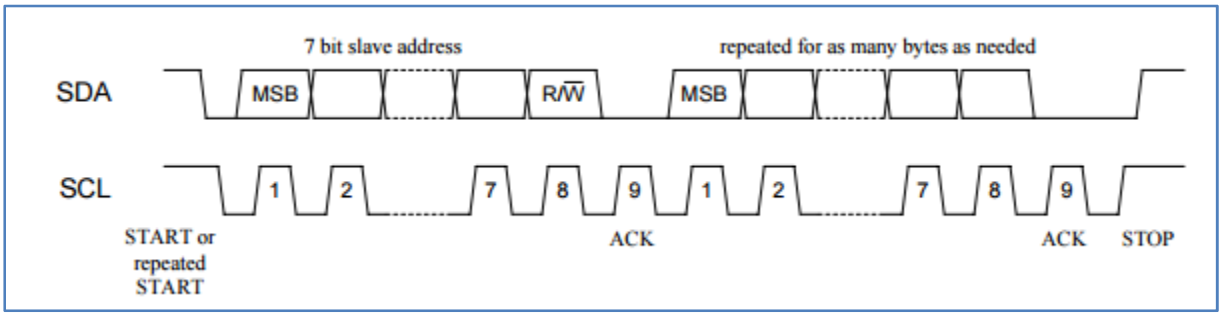

Figure 1.7: Data Transmission in the I^2C Bus [23].

2.2 IP (Intellectual Property) Cores

The Altera IP Library provides many useful IP core functions for a production use without purchasing an additional license. Some Altera MegaCore® IP functions require that you purchase a separate license for production use. However, the OpenCore® feature allows evaluation of any Altera IP core in simulation and compilation in the software [2].

The most important IP core we have used in our project is the FFT IP core.

OpenCore Plus evaluation supports the following two operation modes:

- Untethered—run the design containing the licensed IP for a limited time.
- Tethered—run the design containing the licensed IP for a longer time or indefinitely.

For IP cores, the untethered time-out is 1 hour; the tethered time-out value is indefinite. The design stops working after the hardware evaluation time expires [2].

2.2.1 FFT IP Core

The FFT IP core is a high performance, highly-parameterizable Fast Fourier transform (FFT) processor. The FFT IP core implements a complex FFT or inverse FFT (IFFT) for high-performance applications [2].

The FFT IP core I/O is illustrated in the figure 1.8.

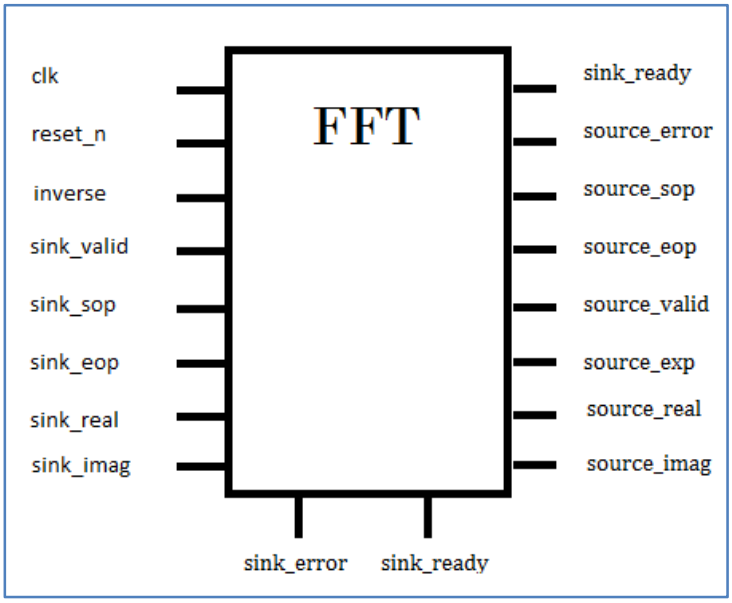

Figure 1.8: FFT IP core I/O

In the I/O Data Flow of an FFT IP Core we distinguish four architectures [2]:

- Streaming FFT that allows continuous processing of input data, and outputs a continuous complex data stream without the need to halt the data flow in or out of the FFT IP core.
- The input order which allows you to select the order in which you feed the samples to the FFT. (Natural order, Bit Reverse order, Digit Reverse Order or –N/2 to N/2 order).
- The buffered burst I/O data flow FFT requires fewer memory resources than the streaming I/O data flow FFT, but the tradeoff is an average block throughput reduction.
- The burst I/O data flow FFT operates similarly to the buffered burst FFT, except that the burst FFT requires even lower memory resources for a given parameterization at the expense of reduced average throughput.

Note

The FFT IP core operates in both tethered and untethered modes [2].

2.3 Altera SOPC

SOPC (System On a Programmable Chip) is a hardware development tool used for integrating various hardware components. The SOPC Builder system development tool simplifies the task of creating high-performance System-On-a-Programmable-Chip (SOPC) designs by accelerating system definition and integration [24].

Using SOPC Builder, system designers can define and implement a complete system, from hardware to software, within one tool and in a fraction of the time of traditional System-On-a-Chip (SoC) design. SOPC Builder is integrated within the Altera Quartus II CAD software to give FPGA designers immediate access to a revolutionary new development tool [24].

SOPC Builder library components supplied by Altera include: Processors, Microcontroller peripherals, Digital signal processing (DSP) cores, Intellectual property (IP) cores, Communications peripherals, Interfaces, Middleware libraries …etc. [24].

2.4 NIOS II Processor

NIOS-II processor is a 32 bits soft processor that can be instantiated on an Altera FPGA device. The NIOS-II processor and its associated memory and peripheral components are easily instantiated by using Altera's SOPC Builder in conjunction with the Quartus II software CAD [24].

2.4.1 NIOS II system

A NIOS-II processor system is equivalent to a microcontroller (or computer on a chip) that includes a processor and a combination of peripherals and memory on a single chip. A NIOS-II system consists of a NIOS-II processor soft core, a set of on-chip peripherals, on-chip memory, and interfaces to off-chip memory, all implemented on a single Altera device. Like a microcontroller family, all NIOS-II processor systems use a consistent instruction set and programming model. The new hardware designs could be tested easily by reconfiguring the FPGA using JTAG interface.

The JTAG interface supports hardware and software development and can be used to perform deferent tasks like: configuring the FPGA, communicating with the FPGA through a UART-like interface and many others [24].

CHAPTER II:

HARDWARE AND SOFTWARE IMPLEMENTATION

Introduction

In this project we implemented a hardware system that recognizes a speech, by the mean of head styled microphone and the Altera DE2-board. We used VHDL language to describe the hardware of the different blocks used in our speech recognition system.

This chapter provides the reader with the different parts and components used in our hardware design followed by a description of their functionalities.

Figure 2.1 shows the general block diagram of our system. As mentioned here, our design is composed of three main components.These components are the audio codec, the FFT module, and the NIOS system.

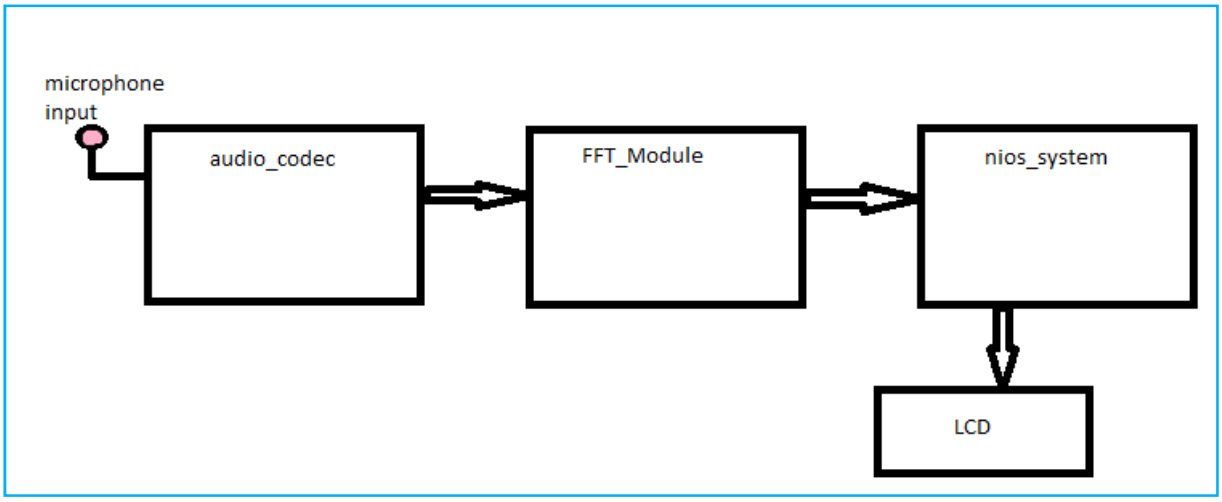

Figure 2.1: General implementation block diagram

1. Audio Codec

This Audio codec is the first block of our system, which is in charge of the reception of the audio signal from the microphone. The analog signal produced by the microphone must be converted to digital before processed by our system. To perform this, an analog to digital converter (ADC) should be used. Fortunately, the DE2-Board is equipped by an audio chip that does this task, which is the Wolfson WM8731 audio codec.

The WM8731 on the DE2 board is the small chip labeled U1 near the row of colored audio connectors; it has a flexible digital interface, that we access using dedicated pins on the Cyclone II FPGA device. It has internal analog-to-digital converters (ADCs) and digital-to-

analog converters (DACs) that allow the device to convert at sampling rates from 8kHz to 96kHz, supporting digital word lengths from 16-32 bits [20].

This chip can be configured by writing to 11 registers which can be accessed using the I2C (Inter integrated circuit) protocol. For audio data path, WM8731 may be operated in either one of the following four offered audio interface modes:

- Right justified
- Left justified
- I2S
- DSP mode

In this part we have developed a VHDL code to get the audio data from the MIC-line in of the DE2 to the Wolfson chip, and save it in a memory block that we created using the Altera Megafunction wizard tool of the Quartus II software. This VHDL code is composed of two main parts, configuration controlling and data flow controlling. To deal with synchronization aspects, we used two phase-locked loops (PLL) for dividing and buffering the 27MHz and 50MHz DE2-board clocks. The block symbol generated using Quartus II to this component is shown in figure 2.2.

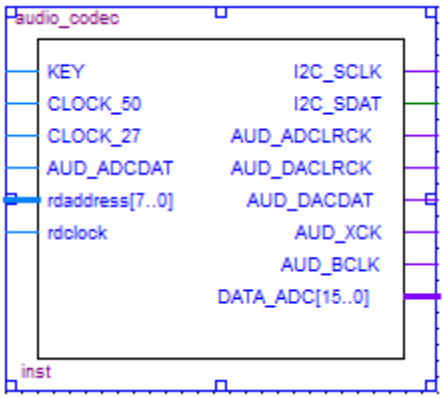

Figure 2.2: audio codec block

This symbol shows the different inputs and outputs of the audio codec. This block will be connected later to other blocks for further processing of the audio signal.

1.1 Configuration Part

Accessing the configuration register needs a solid background on serial communication protocols, specifically the I2C one. Based on the fact that the Wolfson chip supports the I2C protocol, our application will use this chip just to send data (configuration) to it and there is

only one peripheral device that will be interfaced. We designed a specified FSM (finite state machine) that has six states (reset state, start condition, send data, acknowledge, prepare for stop, stop condition).

Following reset, the FSM entered the start condition, in which the I2C clock line was held high while the I2C data line was pulled low. The FSM then entered the send data state, where each of the 24-bits of the current packet was sent. After each 8 bits, the FSM entered the acknowledge state, where the I2C data line was set to high impedance. Following transmission of the full 24-bit packet, the FSM entered the stop condition, where the I2C clock line was held high, while the I2C data line was pulled high.

In this configuration we have set the audio codec chip to be a slave device, and the FPGA be a master device. Figure 2.3 is the FSM diagram we used in this configuration.

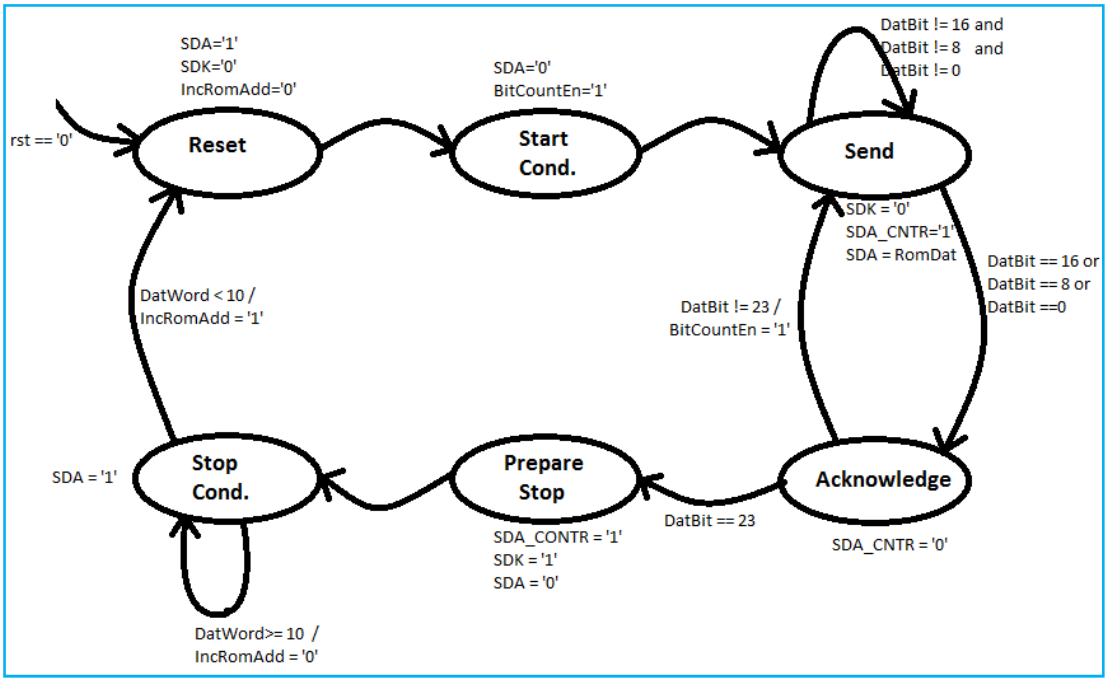

Figure 2.3: I2C FSM diagram

We created a small ROM to save these configuration packets, which will be retrieved by the controller and send via the two-wire interface to the audio device. This process happens only once and must be the first to be done in our system in order to get a correct data in right moment. For this purpose, we created a delay counter module that is used to wait for 42ms until it finishes the configuration, and then enable the data transmission carried by the ADCDAC controller.

This ROM contains our specified configuration which differs from the default one. In fact, we modified some of them and let others as they were by default. Among the modified ones, we can list the most important of them, which are:

- Enable microphone input boost level
- Disable line in input to ADC
- Sampling frequency to 8KHz
- Left justified mode for digital audio

Figure 2.4 represents the symbol generated by Quartus II for configuring the audio codec chip block. The different I/O ports of this block are mentioned in the symbol.

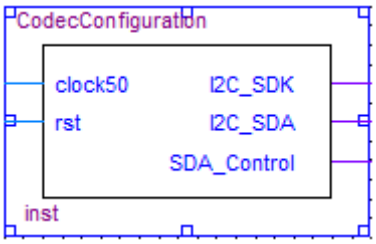

Figure 2.4: Audio codec controller block

As shown in the previous figure, the audio codec controller has two inputs and three outputs. The two inputs are the provided DE2-board 50 MHz clock and a reset which is connected to one of the push buttons of the board. For the outputs, we have the serial clock (I2C_SDK) and the serial data line (I2C_SDAT) that we connect to the specified pins to control the chip serially under the I2C protocol [7].

The remaining output line (SDA_Control) is used by the ADCDAC controller that we will introduce in the following paragraphs.

1.2 Data Flow Part

In this part we developed data controller that ensures a send and a receive tasks with the audio codec chip. Dealing with the audio data needs understanding the communication protocol used by the Wolfson chip. To do so, we used its data sheet to take decision about the timing and precision of the data to be used.

The data controller we developed remains idle for 42ms after the reset button was pressed. The controller then generates two clock signals, which were used by the codec. The bit clock signal is used to clock the output bits from the digital to analog conversion by the codec. The

bit clock signal run at 3.07MHz, using the 18.42105 MHz (master clock XCK) signal produced by a PLL. The second clock, the left/right select clock, runs at 192 KHz. The left/right select clock is used to switch channels within the codec [20].

We set the digital audio interface format in the configuration part by accessing the register 7 [20]. We have used the following parameters:

- Left justified mode
- 16 bit data precision
- Right channel ADC/DAC data available when DACLRC/ADCLRC is low
- MSB is available on the first BCLK rising after ADCLRC/DACLRC rising edge
- Right channel ADC/DAC data Right and vice-versa
- Enable slave mode
- Don't invert BCLK

The left justified mode works as follows:

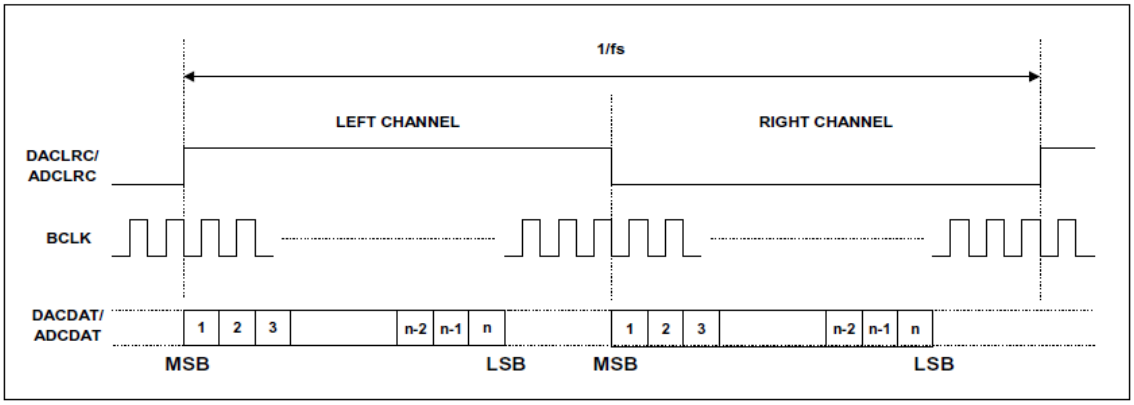

Figure 2.5: Left justified mode [20]

The type of communication we used is the master slave one, where our data controller module acts as a master and the Wolfson chip as a slave; we used five lines to control the data transaction. Three of them are for the different clocks and the remaining ones are for data. The following figure shows how this connection is made.

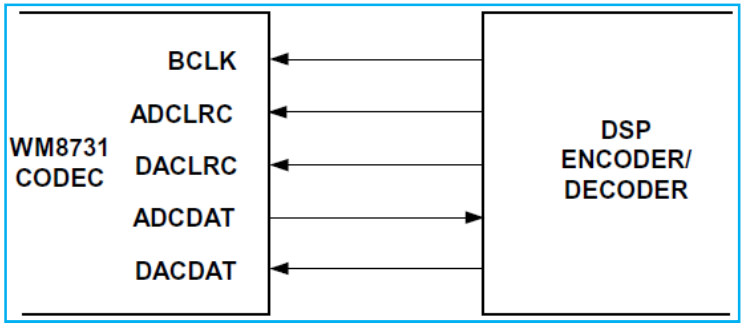

Figure 2.6: Slave Mode Connections [20]

For testing purposes, we created a loop back of the sound from the MIC-line in to the Line-Out line by writing the data gotten from the ADC (ADCDAT) to the DAC using DACDAT line. With this technique we can adjust our configuration in the way to get a right data in the right moment and see the effect of different configurations. The figure 2.7 is the block generated using Quartus II of the data flow controller that we named data_controller, which has three inputs and seven outputs.

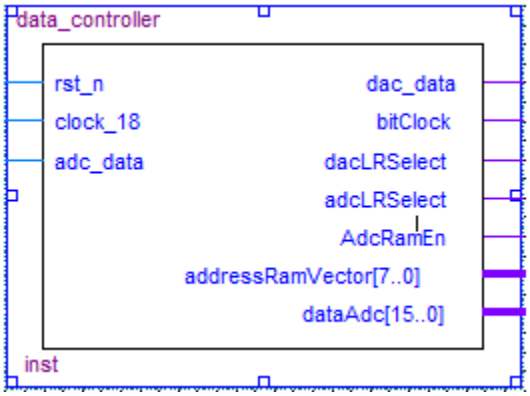

Figure 2.7: data flow controller block

1.3 Delay Counter

Since the configuration part must be held before any data transaction, we estimated a wait time to be 42ms for the data transaction to start after the first one. Thus, we have developed a VHDL code such that the clock signal will be generated to data controller 42ms after the reset button is pressed. The following figure 2.8 shows its block generated using Quartus II.

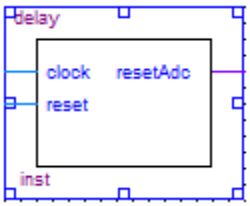

Figure 2.8: delay counter block

1.4 Clock Divider

A clock divider is used to generate a master clock of 18 MHZ. Referred to the Wolfson data sheet, the Wolfson chip is configured under the normal mode to get a sampling frequency of 8 KHz [20].

In order to get this MCLK we use a Megafunction wizard tool of Quartus II software to create a PLL that takes as input the 27MHz DE2-board clock and divides it to get as output clock 18.42105Mhz. Figure 2.9 shows the resulting block symbol.

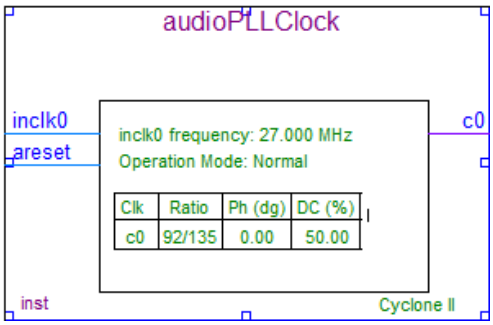

Figure 2.9: clock divider block

1.5 Memory Blocks

Memory blocks are required for the proper operation of the system for either to hold configuration data of to store temporarily audio data. In our design we have tried to use minimal amount of memory in the system. Both memory types are used: ram and rom.

1.5.1 ROM

Read only memory of the size 32x24 bits, which is used to store 10x24 bits of the audio codec configuration. We created this memory using the Megafunction Wizard embedded in Quartus II CAD software. The resulting schematic block of the created ROM is shown in figure 2.10.

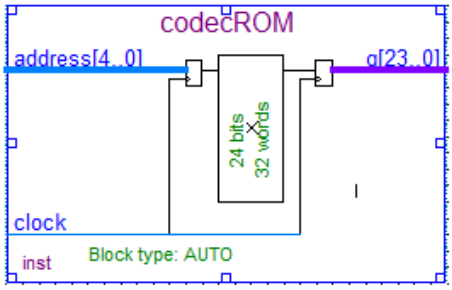

Figure 2.10: ROM Block

1.5.2 RAM

Random access memory is used in our system to store a voice chunk of 32ms duration. Note that the voice is stored before its FFT is performed. Since the sampling frequency used is 8 KHz, a total of 256 samples are taken in a timeframe of 32ms. Therefore the required size of that RAM would be 256x16bits. The resulting schematic block of the created RAM is shown in figure 2.11.

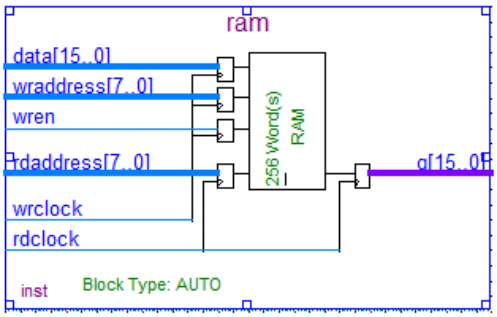

Figure 2.11: 256x16 RAM Block

As noticed, the RAM of the above figure is a 2-port RAM that can be accessed for both read and write actions simultaneously. In fact, we have used this type of memory because we are implementing a real-time speech recognition system which should provide the possibility to write and read data at the same time. This RAM stores the output data from the ADC of the audio codec in the form of a 16-bits word each and which can then be accessed by the FFT module to read that audio data to be transform.

1.6 Audio Codec General Block Diagram

The different circuit diagrams discussed previously are instantiated and connected to construct the audio codec block including both the configuration and the data flow control functions. The resulting audio codec diagram circuit is shown in figure 2.12.

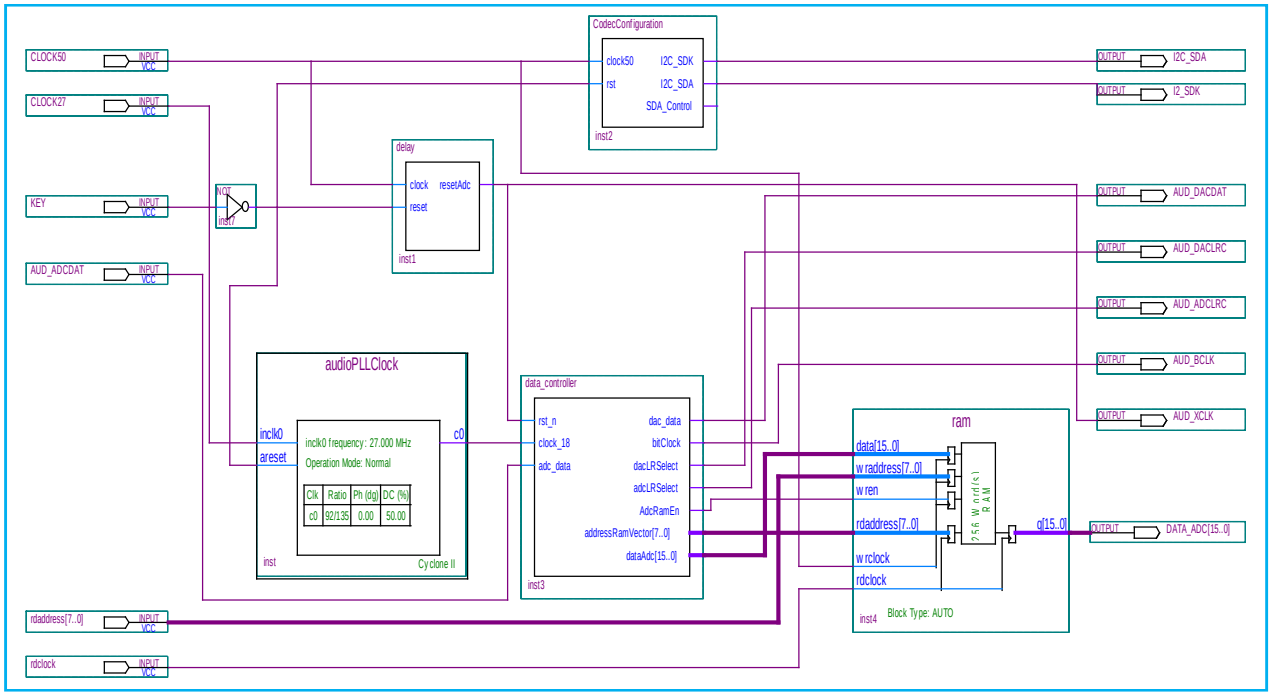

Figure 2.12: Audio Codec General Block Diagram

2. Fast Fourier Transform Module

Fast Fourier transform is a method for fast calculation of discrete Fourier transforms. In this work we are dealing with a speech signal which is digitized using an analog to digital convertor. The output of the ADC is a digital representation of the speech signal in time domain. To extract feature vectors representation of the speech signal, we need to perform some DSP manipulations which cannot be accomplished in time domain. To get rid of this problem, mapping the signal from time to frequency domain should be done, where the calculations are easier than keeping them in time domain. For that reason, calculating the DFT of that signal is needed.

To satisfy the real time criteria in the desired system, a faster algorithm is required, and then the FFT approach fits this requirement perfectly.

The FFT Module contains an FFT and a 256x16 memory block. The following figure is the FFT Module block.

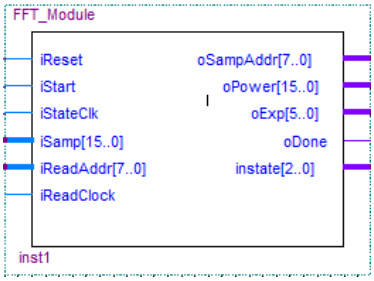

Figure 2.13: FFT Module Block

2.1 FFT IP Core

To calculate the FFT of the speech signal we use the embedded Altera FFT Megafunction core available in Quartus II CAD software. However, this core has many parameters to be taken into account. Understanding the functionality of this mega core has taken a great amount of time for us. Once this was done, a VHDL code to control the core was written, i.e., import data from the RAM of audio codec, treat it, and then save it in another RAM. The second RAM has been implemented in similar fashion as did for the audio codec one. Our code is based on an FSM methodology consisting of four different states.

2.2 FFT FSM

Based on the FFT IP Core and the Verilog code proposed by Cornell University [1] [2], we have developed a VHDL code of an FSM that controls this IP core.

The module is configured so as to produce a 256-point FFT. It takes data input in natural order, and output the transformed data in natural order too. In the Megafuncion parameterization of the FFT module, we tried out various architectures for the FFT, i.e. Burst, Buffered Burst, Streaming and Variable Streaming. It turns out that the Burst architecture was most efficient one [1]. This fact was true for our implementation too.

After studying the Burst architecture and using the FFT Burst Data Flow Simulation Waveform shown in the figure 2.14 and which was taken from [2], we have developed an FSM that has four states.

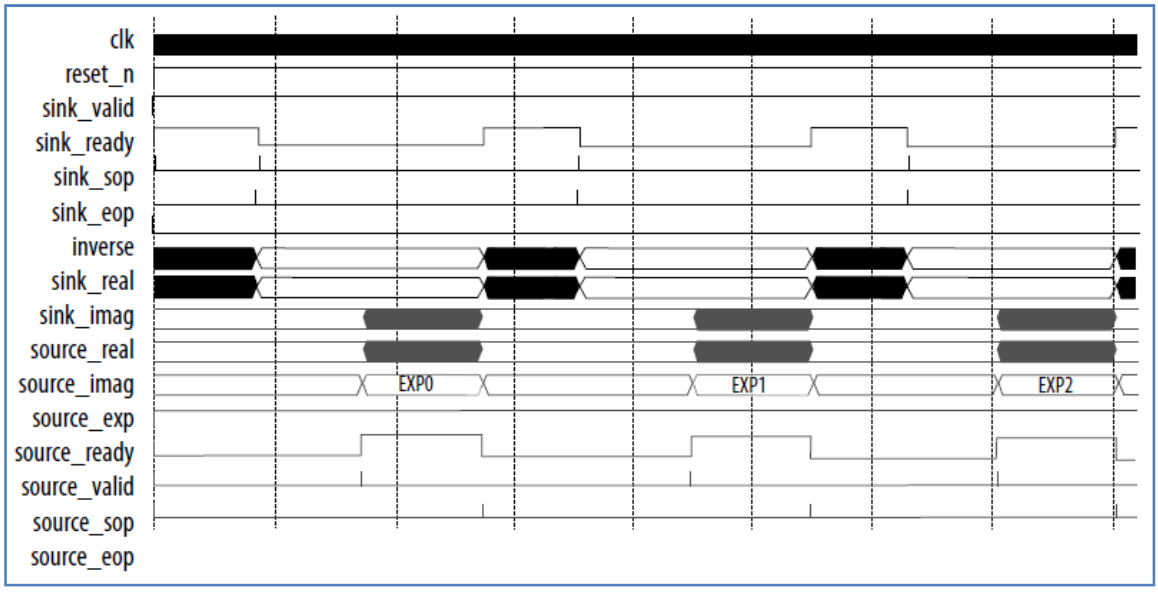

Figure 2.14: FFT Burst Data Flow Simulation Waveform [2]

A best way to handle the inputs of the FFT is by implementing a state machine. This later is implemented to control the functioning of the FFT module using four states. We have set *'reset_n'* to make the program be in ***SOP*** (start of packet) state initially. When it is in the *SOP* state, the *'sink_sop'* is set to indicates the start of incoming FFT frame, begin counting by setting the count to one and assign the input signal to *'sink_real'* that will be processed by the FFT, then it goes to ***Transforming*** state. This state is where it does its job of transforming the time signal to frequency one, it releases *'sink_sop'* and increments 'count' to indicate that it is in that state. Once *'count'* reaches (255-2) it leaves this state and goes to ***EOP*** (end of packet) state where it does the last transformation and signals the end by setting *'sink_eop'*, then enters the **IDLE** state where it will wait for the next incoming FFT frame. The signal *'go'* indicates the presence of data ready for processing, so in this state it stays checking for this signal, when it is set it goes to start state (SOP) and repeat the same process. The following figure represents this FSM.

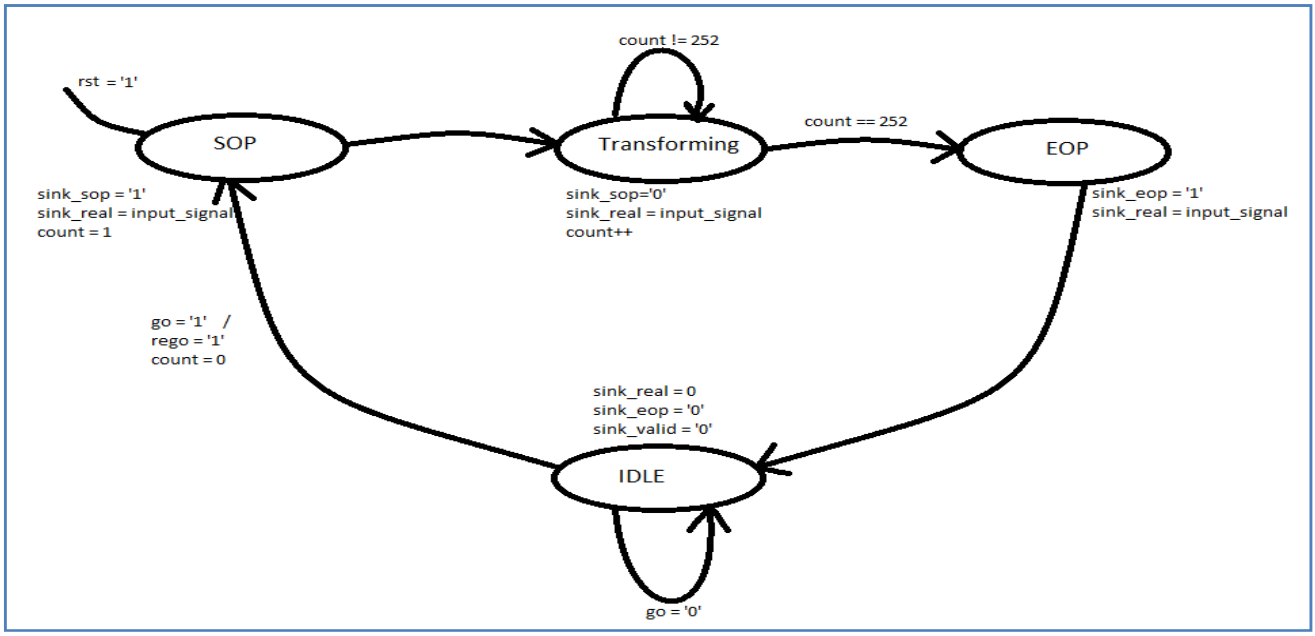

Figure 2.15: FFT Module FSM Diagram

The data processed by the FFT will be saved in a memory block before interfacing it to the NIOS_II.

2.3 FFT Memory Block

We created 256x16 bits of memory in order to store the FFT output. The memory is synchronized with FFT by getting the same clock and is also interfaced to the NIOS-II. The

completion of FFT operation is indicated to the NIOS through a PIO. When the NIOS senses that signal, it checks for the 'source_exp' of the FFT value, to know whether there is a speech. In case of the presence of any speech, the NIOS-II reads the next 31 blocks of that memory.

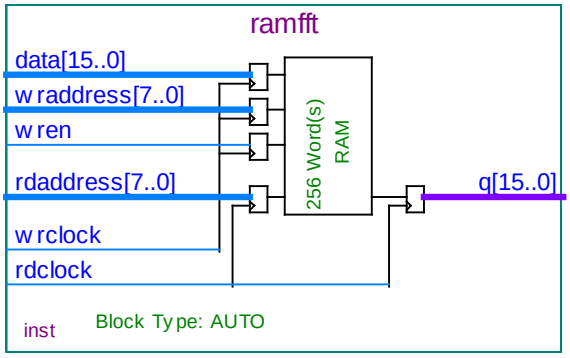

Figure 2.16: FFT RAM Block

3. NIOS II System

The last hardware part of our implementation is creating a NIOS II soft processor in order to treat the data produced by the FFT module, which is very difficult to implement in hardware. This is why we decided to complete the feature extraction in software, i.e., with a C program executed by the soft processor.

Using the SOPC builder tool of the Quartus II, we created a NIOS-II system consisting of:

- NIOS processor
- On-chip memory
- SDRAM
- I/O inputs and outputs for the LEDs and Switches
- I/O inputs and outputs to interface the system to the FFT module
- LCD

After creating this system, we have generated a schematic block for instantiating it and synchronize it to the other parts of the system in the top-level that we called *speech-recognition*. The following figure shows the block diagram representation of this NIOS-System.

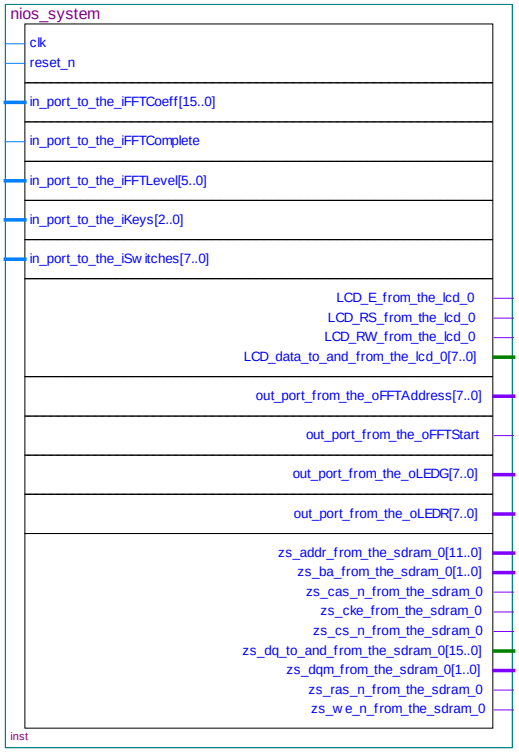

Figure 2.17: NIOS II System Block

4. Top Level

The schematic diagram Top level module connects and synchronizes the audio codec, FFT module and NIOS-II System. In order to properly generate the required clock signals for the NIOS, SDRAM and AUDIO CODEC, it uses 2 PLLs which are inbuilt in the DE2 board. The general block diagram of our hardware implementation is represented in the next page.

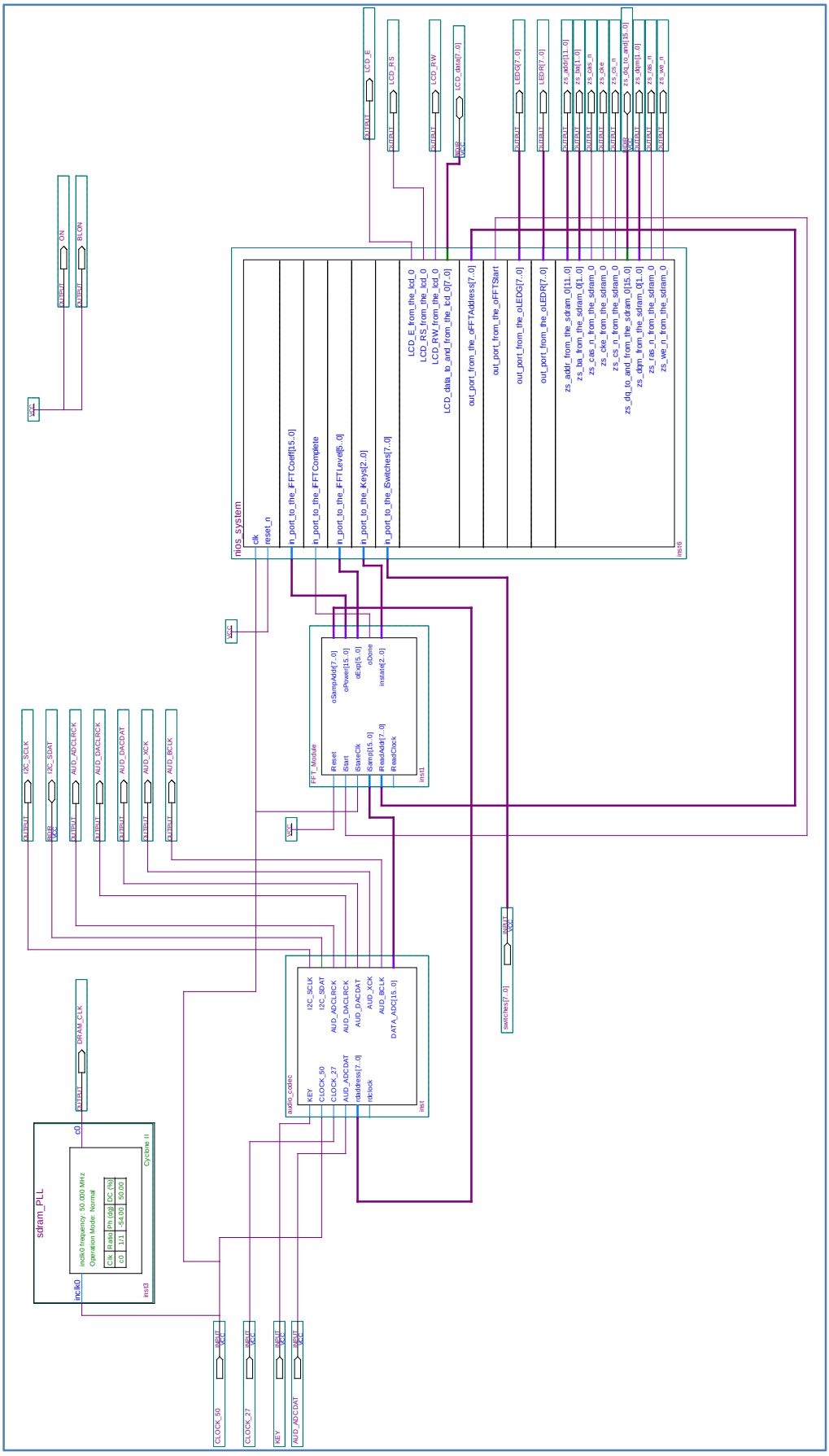

Figure 2.18: Complete Hardware Implementation Block Diagram

5. Software Implementation

As mentioned previously, the first step in any ASR system is to extract features from the audio signal will be used in the recognition stage.

The first different parts of the MFCC algorithm ranging from the pre-processing up to the FFT phase are implemented in the previous section. The remaining parts are implemented using the SOPC builder and the NIOS-II processor. Matching algorithm is also executed within the NIOS processor.

The following section goes through the description of the software part used in our speech recognition system.

5.1 MFCC Algorithm

After processing the audio signal within the FFT module, most of its outputs precede into the NIOS-II system to be process through the remaining parts.

As shown in figure 2.19, first the system has to make sure that it receives a speech signal which is loud enough in order to be processed. The level of the sound is given by the FFT module.

Once the sound is detected, for each sample, FFT coefficients are read from the FFT module that are used as arguments in some required functions.

After getting all the FFT coefficients, the MFC Coefficients are gathered by applying first the filter bank, then computing the Mel transform, and finally applying the DCT transform.

The result will be 12 coefficients which are the MFCC (coefficients) required for the matching.

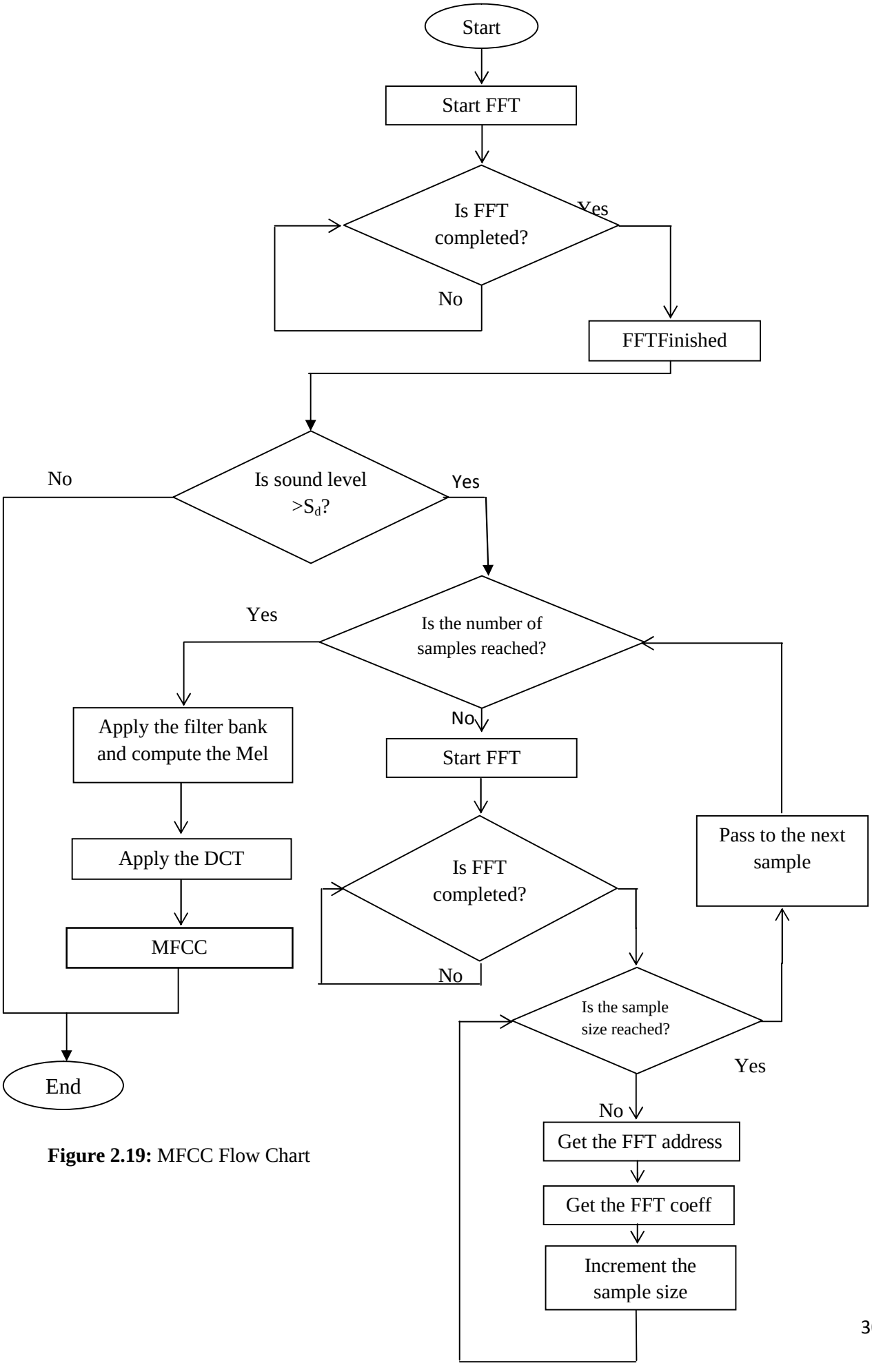

Figure 2.19: MFCC Flow Chart

5.2 Data-Base

In our project we have made our database using the MFCC coefficients resulted from the processing of the speech signal. The data base contains four words each one has 12 coefficients stored in the memory.

In our design we have taken only 12 coefficients of the 26 DCT coefficients into consideration because the higher DCT coefficients represent fast changing in filter bank energies: these fast changes degrade the SR performance.

The figure 2.20 shows how the data stored into the memory.

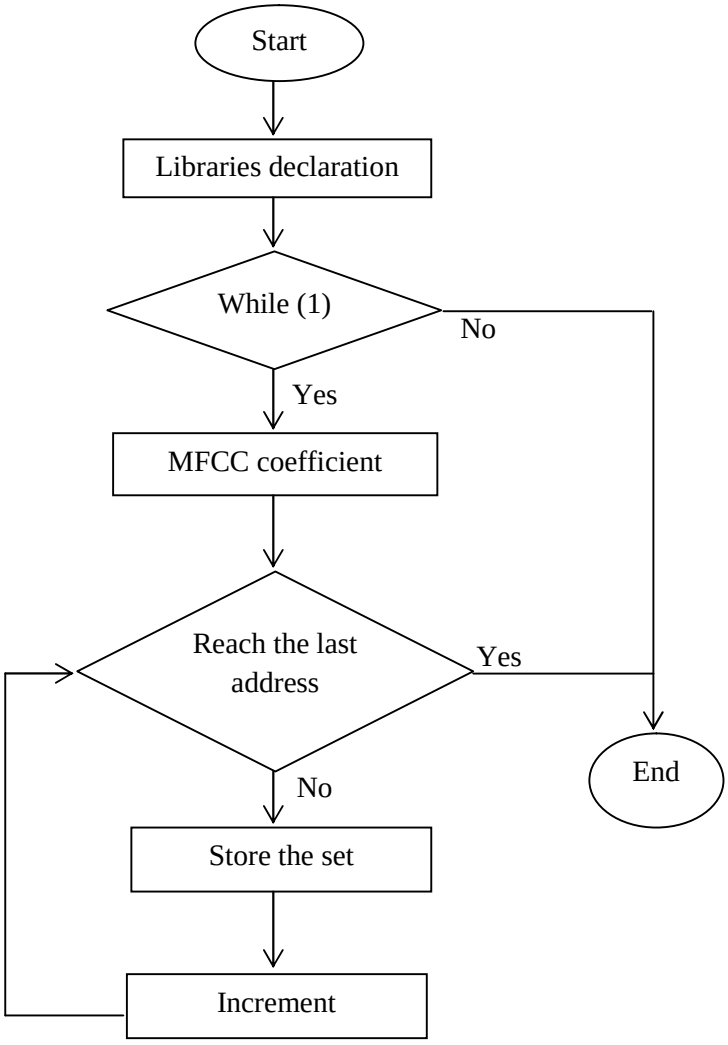

Figure 2.20: Flow Chart of the Data Base

5.3 Recognition Algorithm

The flow chart of figure 2.21 models the recognition part of the system.

After getting the MFCC of the spoken word, the following steps are performed:

1. Retrieve the first predefined word's MFCC from the database
2. Calculate the distance between the spoken word MFCC and the predefined one.
3. Verify the threshold condition ($D_S<|T_h|$):
    - If satisfied, give an order to display the corresponding word on the LCD (word matched)
    - If not satisfied, retrieve the next predefined word from the database and go to step 2.

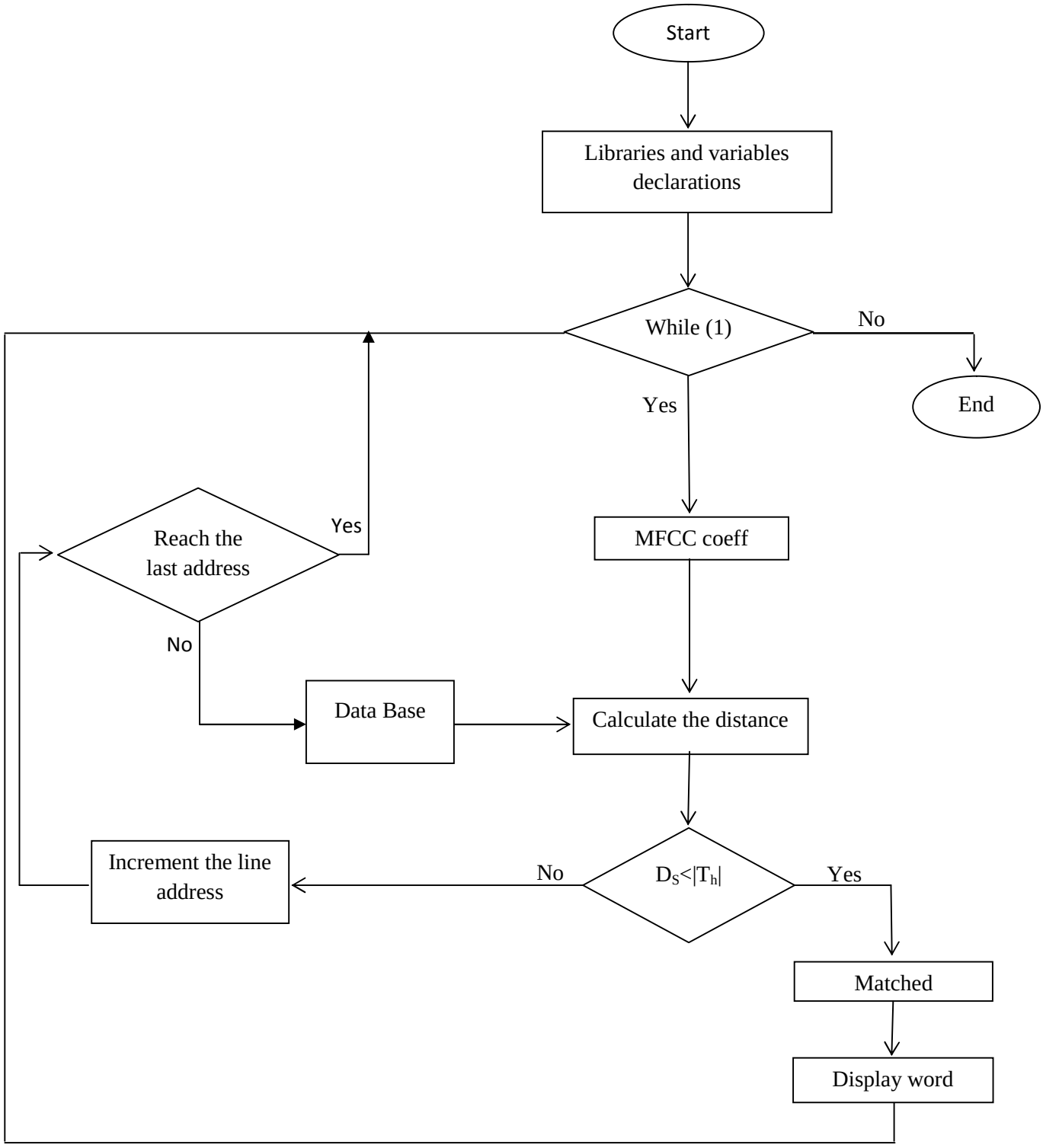

Figure 2.21: Recognition Flow Chart

CHAPTER III: RESULT AND DISCUSSION

Introduction

This chapter deals with the discussion of our simulation results obtained with Quartus II CAD software. Some implementation results will be shown for testing the functionality of our system on DE2 board.

1. Audio Codec Simulation

The simulator tool included in the Quartus II CAD software was used to simulate the behavior of the audio codec block of our system. The time interval chosen for that simulation is 9.5 msec. The obtained results are shown in figure 3.1.

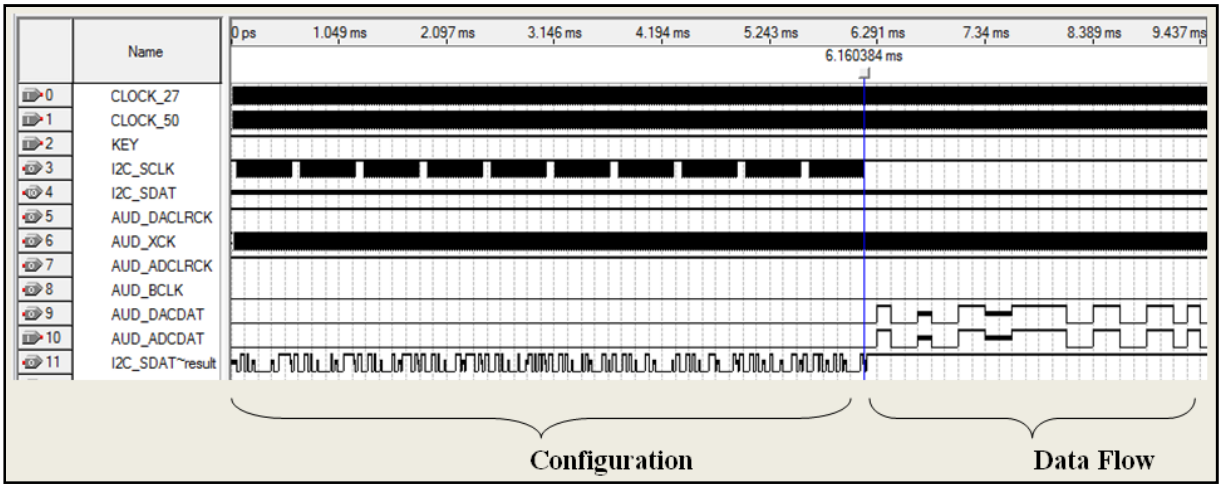

Figure 3.1: Audio Codec Simulation

The above simulation waveforms demonstrate the well functionality of the audio codec circuit. These results deal with the two major phases for the audio codec block, the audio codec chip configuration and the audio data flow. Both phases are discussed below.

1.1 Wolfson Chip Configurations

Figure 3.2 clearly shows the ten activations of the serial clock signal line (I2C_SCLK) for the transmissions of ten frames of data using the serial data line (I2C_SDA). The size of each frame is 24 bits. On each activation, an 8-bits Wolfson's register is configured accordingly.

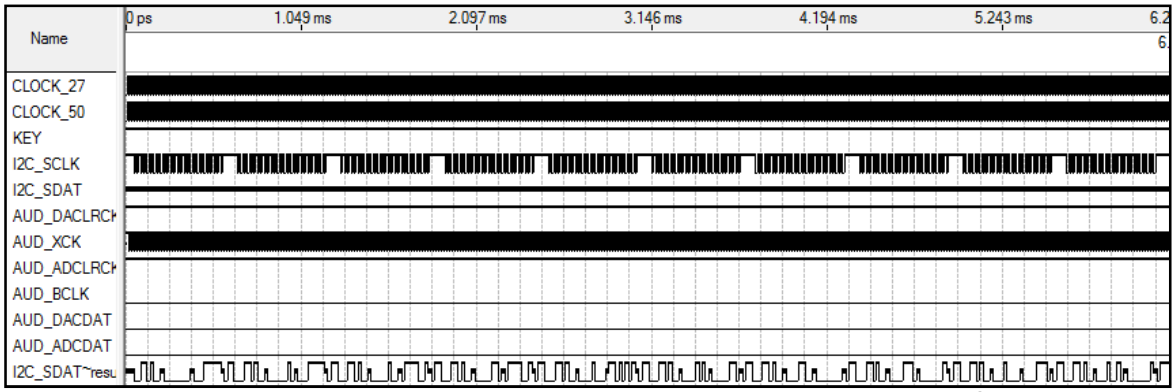

Figure 3.2: Transmission of Ten Frames of Data to the Wolfson Chip Using I^2C Protocol

The two first corresponding Wolfson chip register configurations are hold by the two first frames. The Wolfson chip can be accessed using two addresses which are selected using the control interface address selection as shown in the table below.

Table 3.1: Control Interface Address Selection [20]

CSB STATE	ADDRESS
0	0011010
1	0011011

Since the CSB is tied to ground, and referring to the schematic in the DE2 user Manuel. The audio codec address is '0011010'. The seventh bit is zero because the Wolfson chip is configured in a slave mode (just writing to it).

The first 8-bits of each frame correspond to the address of the audio codec chip (bits [7:1]) and a writing bit (bit 0). The corresponding value is '00110100'.

The next 8-bits is the configuration register's address (bits [7:1]), followed by the writing bit (bit 0). The last 8-bits are for the configuration. As shown in figure III.3

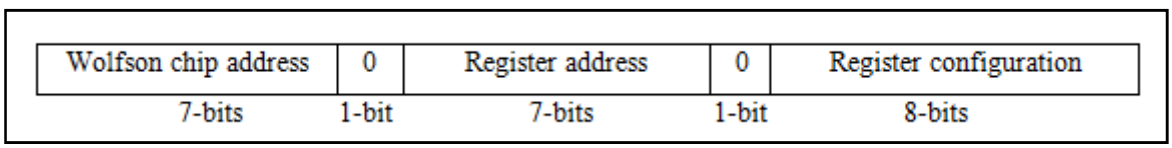

Figure 3.3: 24-Bits Configuration Frame

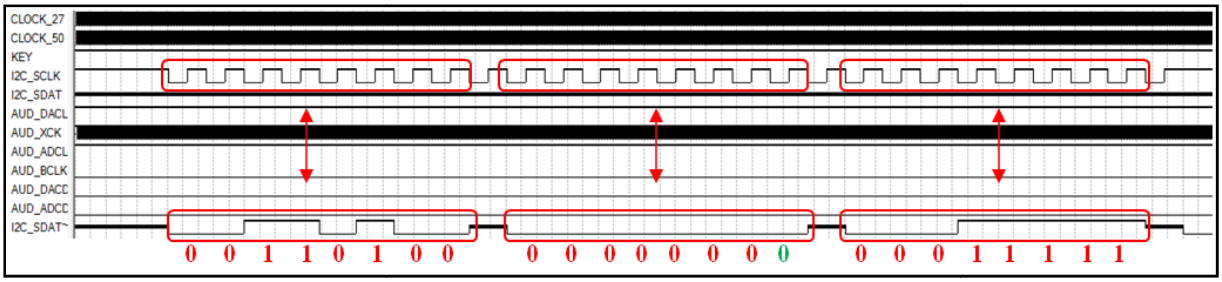

Figure 3.4: Transmission of Register 1 Data Configuration

In figure 3.4, bits [4:0] are set to '11111' for stepping up the Left channel input volume, then disable the line input mute to ADC by setting bit 7, and left the other configurations as default.

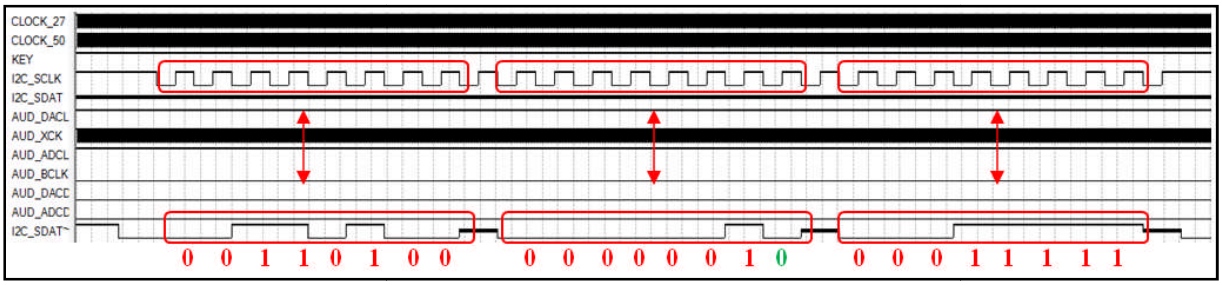

Figure 3.5: Transmission of Register 2 Data Configuration

In figure 3.5, bits [4:0] are set to '11111' for stepping up the right channel input volume , then disable the line input mute to ADC by setting bit 7, and left the other configurations as default. These configurations are achieved referring to the table 3.2.

Table 3.2: Line Input Software Control [20]

REGISTER ADDRESS	BIT	LABEL	DEFAULT	DESCRIPTION
0000000 Left Line In	4:0	LINVOL[4:0]	10111 (0dB)	Left Channel Line Input Volume Control 11111 = +12dB . . 1.5dB steps down to 00000 = -34.5dB
	7	LINMUTE	1	Left Channel Line Input Mute to ADC 1 = Enable Mute 0 = Disable Mute
	8	LRINBOTH	0	Left to Right Channel Line Input Volume and Mute Data Load Control 1 = Enable Simultaneous Load of LINVOL[4:0] and LINMUTE to RINVOL[4:0] and RINMUTE 0 = Disable Simultaneous Load
0000001 Right Line In	4:0	RINVOL[4:0]	10111 (0dB)	Right Channel Line Input Volume Control 11111 = +12dB . .1.5dB steps down to 00000 = -34.5dB
	7	RINMUTE	1	Right Channel Line Input Mute to ADC 1 = Enable Mute 0 = Disable Mute
	8	RLINBOTH	0	Right to Left Channel Line Input Volume and Mute Data Load Control 1 = Enable Simultaneous Load of RINVOL[4:0] and RINMUTE to LINVOL[4:0] and LINMUTE 0 = Disable Simultaneous Load

The most important register, by which our work is concerned more, is the analog path control one. The zoomed-in view of the simulation results for this phase is illustrated in figure 3.5.

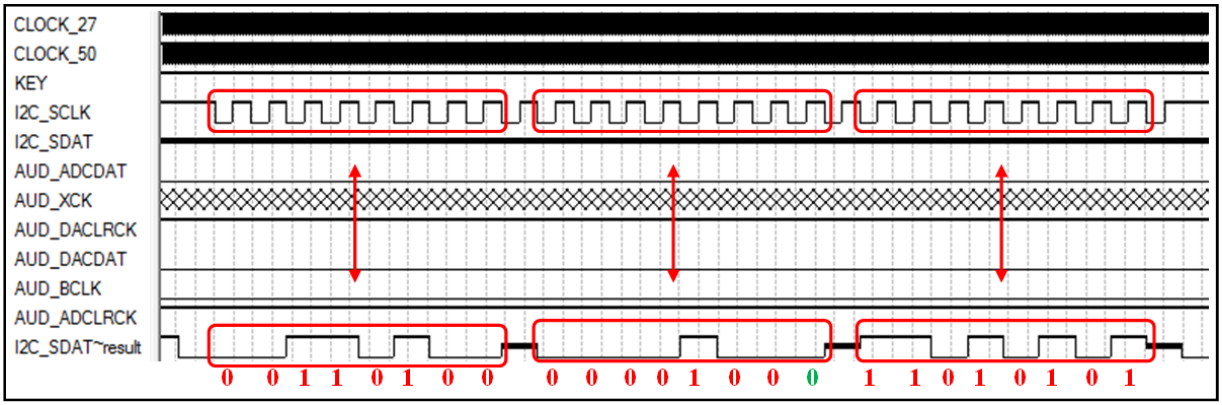

Figure 3.6: Transmission of Register 4 Data Configuration

Notes

- Enable microphone input level boost (bit0 = '1')
- Disable MIC input mute to ADC (bit1 = '0')
- Select microphone input to ADC (bit2 = '1')
- Disable bypass (bit3 = '0')
- Select DAC for testing purpose (bit4 = '1')
- Enable side tone (bit5 = '1')
- Set side tone attenuation to -15dB (bit[7:6] = '11')

The following table is used in this configuration.

Table 3.3: Analog Audio Path Control [20].

0000100 Analogue Audio Path Control	0	MICBOOST	0	Microphone Input Level Boost 1 = Enable Boost 0 = Disable Boost
	1	MUTEMIC	1	Mic Input Mute to ADC 1 = Enable Mute 0 = Disable Mute
	2	INSEL	0	Microphone/Line Input Select to ADC 1 = Microphone Input Select to ADC 0 = Line Input Select to ADC
	3	BYPASS	1	Bypass Switch 1 = Enable Bypass 0 = Disable Bypass
	4	DACSEL	0	DAC Select 1 =Select DAC 0 = Don't select DAC
	5	SIDETONE	0	Side Tone Switch 1 = Enable Side Tone 0 = Disable Side Tone
	7:6	SIDEATT[1:0]	00	Side Tone Attenuation 11 = -15dB 10 = -12dB 01 = -9dB 00 = -6dB

1.2 Data Flow

In our work and for testing the configuration, a loop back of audio data is programmed. Thus, the ADCDAT (ADC Data) is tied to DACDAT (DAC Data) in order to make the output of the inbuilt ADC of the Wolfson chip be the input of the inbuilt DAC of the codec chip. The following figure (figure 3.6) demonstrates that the audio data follows correctly the path from the MIC line to the FPGA, and back to the line out.

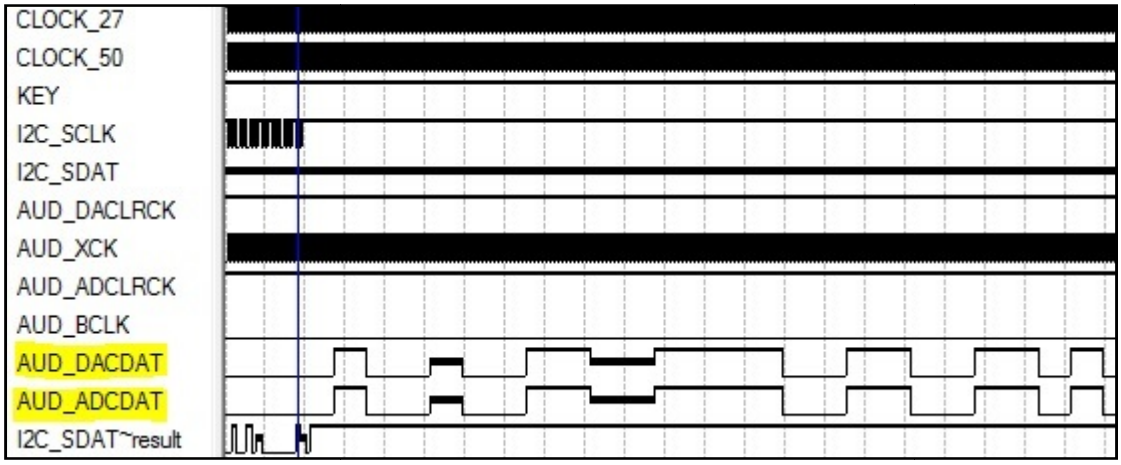

Figure 3.7: Audio Data Flow Simulation.

1.3 Master Clock (XCLK) generation

For sampling issue which is very important in time and memory considerations, we have generated a master clock of 18MHz.

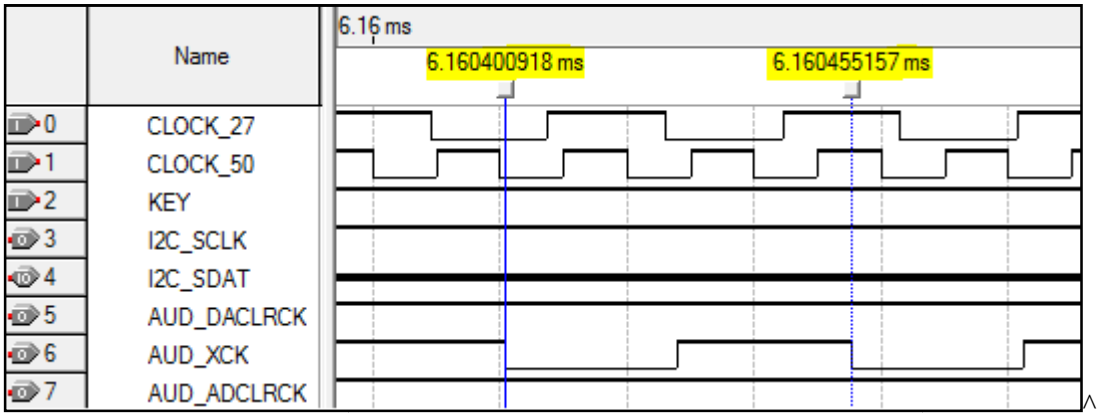

Figure 3.8: Master Clock (XCK) Generation Simulation.

The figure above shows the simulation result of the master clock. The difference between the two highlighted values gives right the XCK period of 0.00005423 ms which is equivalent to 18.4346918 MHz.

2. Hardware Result

2.1 Compilation Report

Using the Quartus CAD software we have compiled the project named *speech_recognition* and we have gotten the report result shown in figure 3.9.

Flow Summary

Flow Status	Successful - Tue May 17 14:16:24 2016
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	Speech_recognition
Top-level Entity Name	Speech_recognition
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	No
Total logic elements	8,463 / 33,216 (25 %)
Total combinational functions	6,919 / 33,216 (21 %)
Dedicated logic registers	5,878 / 33,216 (18 %)
Total registers	5930
Total pins	102 / 475 (21 %)
Total virtual pins	0
Total memory bits	120,832 / 483,840 (25 %)
Embedded Multiplier 9-bit elements	28 / 70 (40 %)
Total PLLs	2 / 4 (50 %)

Figure 3.9: Compilation Report

As shown in the compilation report we have used about 25% of total logic element and about 21% of total pins of our FPGA device.

2.2 Download the Project to the DE2 Board

Using the programmer tool in the Quartus software, we have downloaded the project to the FPGA. Then received two messages: the first, indicating that the Megafunctions used will work for a limited time(1 hour as mentioned in the Altera user manual) as illustrated in figure 3.10, the second, indicating that the system must be connected to the computer for the proper function, as shown is figure 3.11. To avoid those problems our Quartus software should be licensed.

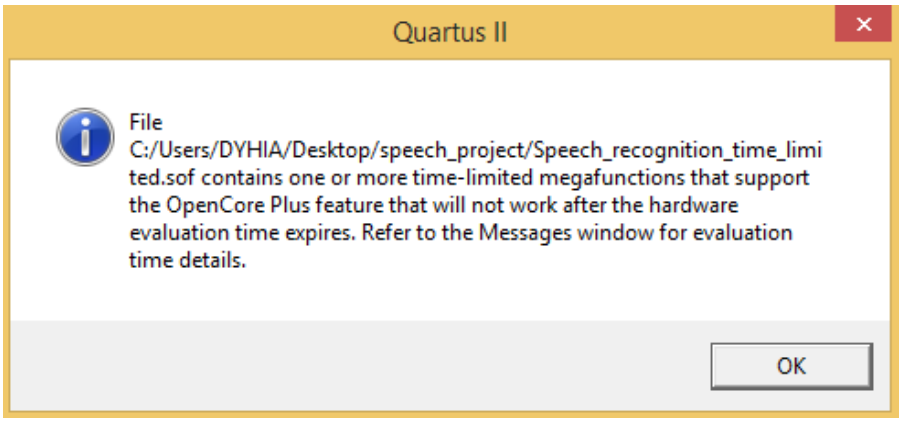

Figure 3.10: Hardware Limited Time Message

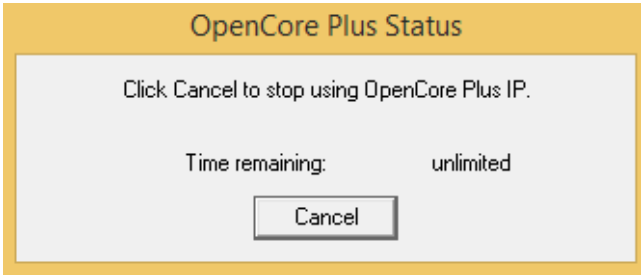

Figure 3.11: Connecting to Computer Message

2.3 Results

Using NIOS II 9.1 IDE we run the C program in the NIOS soft core. We plugged the head styled microphone to the DE2 board MIC line.

Our data base contains four words:

- Right
- Left
- Yes
- No

The result of saying the word ‘Right’ is shown in figure 3.12.

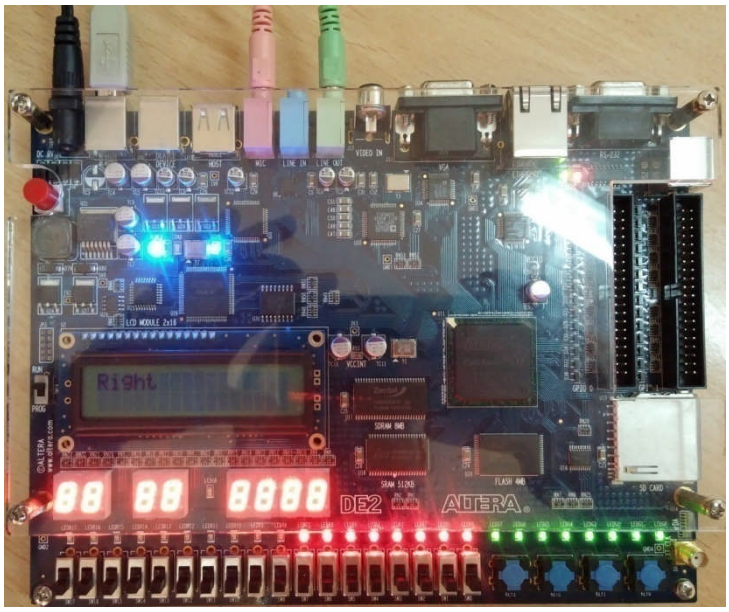

Figure 3.12: Word Recognized ‘Right’

The result of saying the word ‘Left’ is shown in figure 3.13.

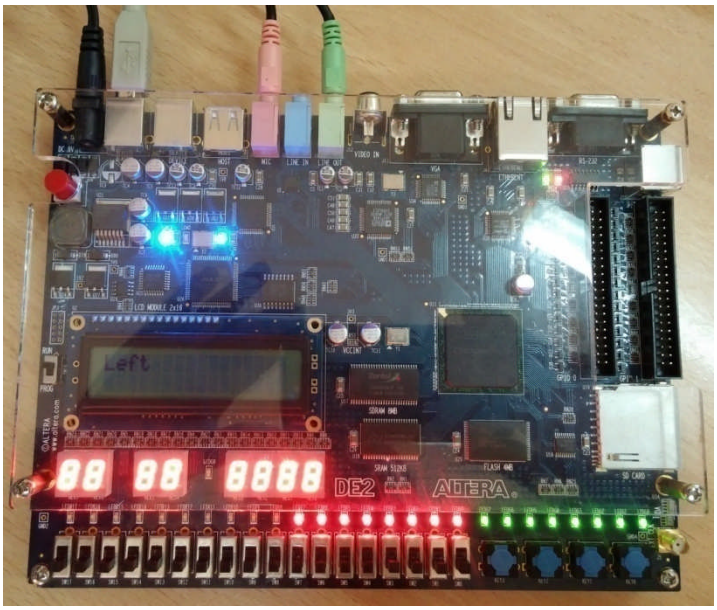

Figure 3.13: Word Recognized ‘Left’

Discussion

We notice that the red LEDs are **ON**, which means a presence of a speech sound. The green LEDs are also **ON** which indicates that a word in a dictionary has been recognized. The word recognized is displayed on the LCD. ‘*Right*’ in figure 3.12 and ‘*Left*’ in figure 3.13.

The result of saying the word ‘Hello’ is shown in figure 3.14.

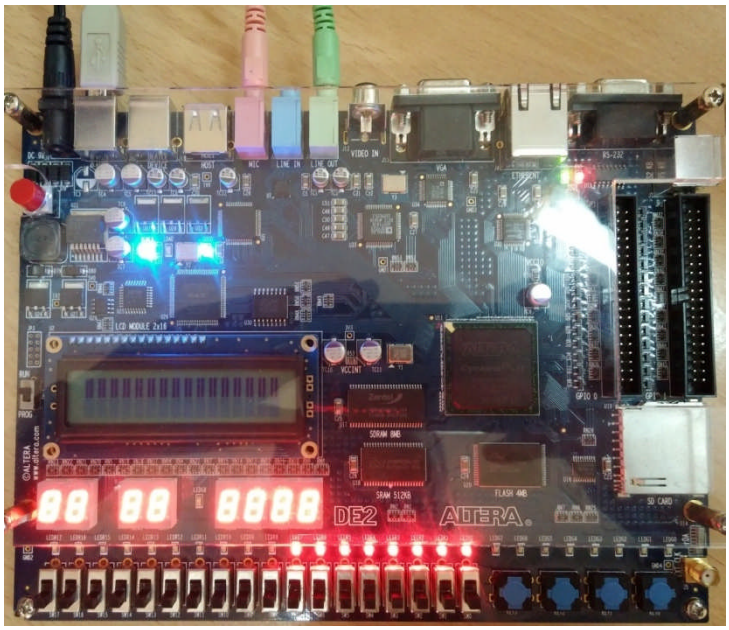

Figure 3.14: Word Not Recognized ‘Hello’

Discussion

In figure 3.14 the red LEDs are turned **ON** which signifies a presence of speech sound. However, the Green LEDs are **OFF** meaning that the spoken word is not in the data base.

When there is no speech, i.e., silence, the Red LEDs as well as the LCD are **OFF**, as illustrated in the figure 3.15.

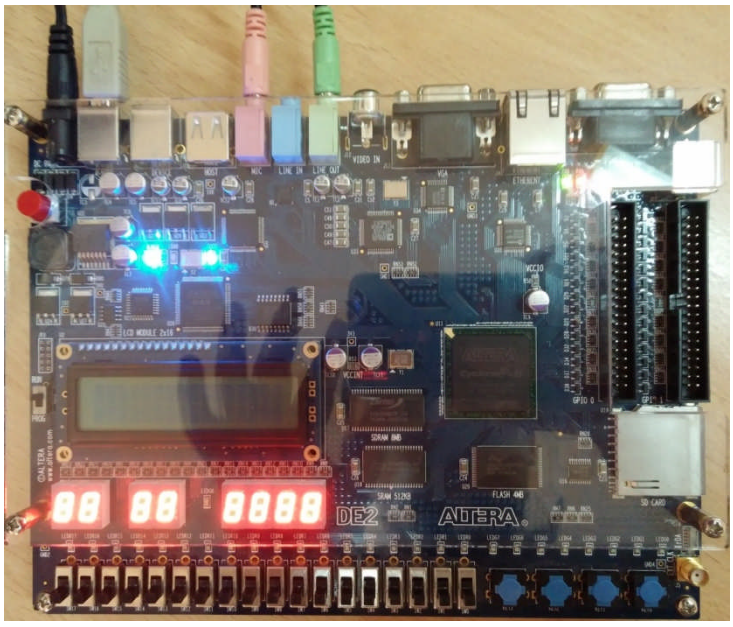

Figure 3.15: Speech Not Present

Developing a human machine interaction mean that fits real-time criteria and gives a facility for both the user and a machine is what motivated our project. Speech recognition systems are the perfect solution for this interaction. Many software applications perform the recognition. However, performing the same in hardware will result in better performances: fitting the real-time criteria and accuracy in speech recognition.

In this report, the design and implementation of an FPGA-based real time speech recognition is presented. This system is developed using the two aspects of hardware and software implementations. In the beginning we desired to realize the whole system in hardware, but implementing the MFCC algorithm in hardware is much difficult and will take a lot of our reduced time. This is why we used the NIOS-II soft processor to execute a C program performing this MFCC algorithm. The soft processor is also used in performing the recognition using the matching approach.

We have successfully designed a real time speech to text engine using FPGA technology, taking advantage of parallel processing and very high speed resources proposed by the Altera DE2 board.

A real time isolated word is implemented in this project using a matching approach. As improvement of this work we suggest to advance this system into a continuous speech recognition engine, changing the matching technique by the HMM (Hidden Markov Model) algorithm.

References

[1] Aakash Jain, Krishna Teja Panchagnula, Nitish Paliwal, (2010). *Real Time Speech Recognition Engine*, Cornell University.

[2] Altera Corporation, (2015*). FFT IP Core User Guide*.

[3] Altera Corporation, (2014). *Embedded Memory (RAM: 1-PORT, RAM: 2-PORT, ROM: 1-PORT, and ROM: 2-PORT) User Guide.*

[4] Altera Corporation, (October 2006). *Audio / Video Configuration Core for DE2/DE1 Boards*.

[5] Altera Corporation, (May 2011). *NIOS II Hardware Development, Tutorial.*

[6] Altera Corporation, (2008). *Introduction to the Altera SOPC Builder Using VHDL Design*.

[7] Altera Corporation, *Altera DE2 Board Pin Table*.

[8] AnkitaGoel , Hamid Mahmoodi, (spring 2012). *Embedded Systems Design Flow Using Altera's FPGA Development Board (DE2-115 T-Pad).*

[9] Carlos Asmat, David Lopez Sanzo, Kanwen Wu, (June 2007). *Speech Recognition Using FPGA Technology*.

[10] José Ignacio, Mateos Albiach, (2006). *Interfacing a Processor Core in FPGA to an Audio System, Master Thesis Performed in Electronics Systems*.

[11] HuanFang , Xin Liu. *SOPC-based Voice Print Identification System*.

[12] Lattice Semiconductor Corp., (2015). *I^2C Master Controller*.

[13] Lindasalwa Muda, Mumtaj Begam, I. Elamvazuthi, (March 2010). *Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques.*

[14] Haitao Zhou, Xiaojun Han, (August 2009). *Design and Implementation of Speech Recognition System Based on Field Programmable Array, Tianjin Polytechnic University,China.*

[15] M.T. Bala Murugan and M. Balaji, Instructor: Dr. B. Venkataramani, (2006). *SOPC-Based Speech-to-Text Conversion.*

[16] Pong P. Chu, Cleveland State University (2008). *FPGA Prototyping By VHDL Examples*.

[17] Ricardo da Silva, Professor Steve Gunn, (April, 2013). *Speech Recognition on Embedded Hardware.*

[**18**] Shivanker Dev Dhingra, Geeta Nijhawan, Poonam Pandit, (August 2013). *ISOLATED SPEECH RECOGNITION USING MFCC AND DTW.*

[19] Vonei A. Pedroni, MIT Press, (2004). *Circuit Design with VHDL.*

[20] Wolfson Microelectronics, (April 2004). *Data sheet, WM8731/WM8731L, Portable Internet Audio Codec with Headphone Driver and Programmable Sample Rates.*

[21] SantoshK.Gaikwad, BhartiW.Gawali, PravinYannawar, (November 2010). *A Review on Speech Recognition Technique.*

[22] Young-Uk Chang1, Chang Choo, Il-Young Moon, (September 2015). *FPGA-Based Hardware Accelerator for Feature Extraction in Automatic Speech Recognition.*

[23] Enoch Hwang, (April 2008). *Implementing an I^2C Master Bus Controller in a FPGA.*

[24] Altera Corporation, (June 2003*). SOPC User Guide.*

[25] Noelia Alcaraz Meseguer,(July 2009). *Speech Analysis for Automatic Speech Recognition. Norwegian University of Science and Technology.*