Hardware Acceleration of Spoofing Speech Detection

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

Submitted by:

Abhinav Himanshu

Guided by:

Supervisor: Dr. Vivek Kanhangad

Co-supervisor: Dr. Santosh Kumar Vishvakarma

INDIAN INSTITUTE OF TECHNOLOGY INDORE

December 2017

CANDIDATE'S DECLARATION

I hereby declare that the project entitled **"Hardware acceleration of spoofing speech detection"** is submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Electrical Engineering' completed under the supervision of **Dr. Vivek Kanhangad, Associate Professor in Electrical Engineering department and Dr. Santosh Kumar Vishvakarma, Associate Professor in Electrical Engineering department** at IIT Indore, is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Abhinav Himanshu
140002001
Discipline of Electrical Engineering
Indian Institute of Technology Indore

CERTIFICATE by BTP Guides

It is certified that the above statement made by the student is correct to the best of my knowledge.

Co-Supervisor
Dr. Santosh Kumar Vishvakarma
Associate Professor
Discipline of Electrical Engineering

Supervisor
Dr. Vivek Kanhangad
Associate Professor
Discipline of Electrical Engineering

Preface

This report on “Hardware acceleration of spoofing speech detection" is prepared under the guidance of Dr. Vivek Kanhangad and Dr. Santosh Kumar Vishvakarma.

Through this report I have explained a machine learning algorithm for separating spoof speeches from human speeches. And later I have also presented an innovative approach to accelerate the feature extraction process of this machine learning algorithm by proposing a specific hardware architecture for the algorithm. In this report I have discussed other features of speech signals and Gaussian Mixture Models are also discussed in this report. The technological trend which has motivated me to use Field Programmable Gate Arrays (FPGAs) as the computational platform is also explained in details.

I have tried to the best of my abilities and knowledge to explain the content in a lucid manner. I have also added 3-D models and figures to make it more illustrative.

Abhinav Himanshu
B.Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

Acknowledgements

I wish to thank my supervisor Dr. Vivek Kanhangad and co-supervisor Dr. Santosh Kumar Vishvakarma for their kind support and valuable guidance.

It is their help and support, due to which I became able to complete the design and technical report.

Without their support this report would not have been possible.

Abhinav Himanshu
B.Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

Abstract

Speech is a very important biometric identity of an individual. Speech produced by a person carries various features of an individual which are either based on the amplitude or phase of the speech signal. These features perform a very important function in various speaker recognition and speaker verification systems which have become very popular these days. These systems are used in various applications like access control, biometric authentication, and telephone-based applications like phone banking and credit cards.

However, weakness of such biometric systems to various spoofing attacks has recently been exposed. Availability of different modern artificial techniques has facilitated this process of mimicking an individual's voice to fool such speech-driven biometric system. But even researchers have long understood the need to make these systems more robust by developing numerous countermeasure techniques against spoofing attacks. Although there has been a significant research in proposing different countermeasures against spoofing attacks, there is a very limited work done on accelerating these proposed countermeasures to make them more real-time.

This thesis report proceeds in two stages, in the first stage, feature extraction process of Mel Frequency Cepstral Coefficients (MFCCs), is defined and Matlab algorithm is developed for the same. Gaussian Mixture Models (GMM) are trained using these features. The trained GMM model is used for classification of speeches as either human or spoof speech. GMMs are used as they are very popular with speech signals and provide a soft classification very effective in case of speeches. This method is evaluated on ASVspoof 2015 (Automatic spoofing and countermeasures) Challenge dataset. The scepticism surrounding Moore's Law, the breakdown of Dennard Scaling, the dark silicon phenomena and wide scope of making approximation while working with speech processing and machine learning based algorithm together motivates us to choose Field Programmable Gate Arrays (FPGAs) as our hardware platform. During the second stage, we use Xilinx System Generator to develop a hardware architecture for FPGAs to accelerate the MFCC feature extraction process for classification of speeches.

Table of Contents

List of Figures

Chapter 1

Introduction

This chapter highlights the background and motivation for the project. The problem statement has been described for the project and the importance of the results is also clearly portrayed. During the second section of this chapter we will discuss the need to use FPGA as the hardware platform. Towards the end, we will discuss the objectives and expectations to solve the problem statement.

1.1 Background

Speech is one of the most important biometric parameter for identifying individuals. Speech produced by a person carry the features of the source. These features play an essential role in Speaker Recognition Systems and Speaker Verification Systems [1] [2]. Speaker recognition system allows one to identify a person from the properties of a person's voice (voice biometrics). While speaker verification system involves the process of verifying the claimed identity of an individual based on the speech signal from the speaker (voiceprint). Speaker recognition and speaker verification systems have increasingly become popular and are used for various applications like access control, biometric authentication, and telephone-based applications such as in phone banking and credit cards.

However, recently weakness of these systems to spoofing attacks has been exposed. Various new artificial methods are now available which has made this process of mimicking an individual's voice to deceive speech-driven biometric systems even easier. These spoofing attack methods are broadly classified into two categories.

Firstly, Voice Conversion Methods (VC) [3][4] has a goal to modify a speech signal spoken by a source speaker to sound as if it was spoken by a targeted speaker, but still keeping the linguistic contents same. The voice of the source speaker is modified to that of a target speaker by the help of previously collected speech features of targeted speaker. Secondly, Speech Synthesis Techniques (SS) [5], in this method artificial voice of the targeted speaker is directly produced. Voice of the targeted speaker is mimicked from the text.

However, researchers have long noticed the need to ensure the robustness of various Speaker recognition and Speaker verification system and has developed various countermeasures against

these spoofing techniques. All these countermeasures use speech features which incorporates the source speaker characteristics. These features capture two characteristics, 1) Amplitude-based and 2) Phase-based. Amplitude-based features as the name suggest deals with an amplitude of the speech signal. Some common amplitude-based features include Log Magnitude Spectrum and Mel Frequency Cepstral Coefficients which will be discussed in more detail in this report. Phase-based countermeasure includes features such as Cosine Phase, Relative Phase and Modified Group delay [6]. Most of the successful spoofing countermeasures reported in the literature are based on phase.

Although there has been a significant research in proposing different countermeasures against spoofing attacks, there is very limited research on accelerating these proposed countermeasures to make them more real-time. So thesis is done in two stages, In the first stage, MFCC feature is defined and Matlab algorithm is developed for the same. Gaussian Mixture Models (GMM) are used as the classifier, as they are very popular with speech signals and provide soft classification very effective in case of speeches [7]. In the second stage, we propose the Field Programmable Gate Arrays (FPGAs) based hardware architecture for the earlier explained MFCC feature extraction process and its GMM based classification. This method is evaluated on ASVspoof 2015: Automatic spoofing and countermeasures challenge dataset.

1.2 Why FPGAs?

For so long, to improve the computational speed of various digital hardware platforms we were relying on prediction made by Gordon Moore, who said that no. of transistors will double every next two year. However, workloads offered to computers have always been increasing, consider multimedia Industry for an example, back during the time of World War we had only black-white images and videos. During 90's colored images and videos were popular and now, modern computers deal with workloads of processing 3D colored images and videos. Even many Neural Networks based tasks today, has computational requirements which are not fulfilled by traditional multicore-CPU. And relying on a prediction made by studying the trends of industry to keep up with pressing demands of the workloads does not seems like a right option. Moore's law is presented in a very interesting way by the graph shown below.

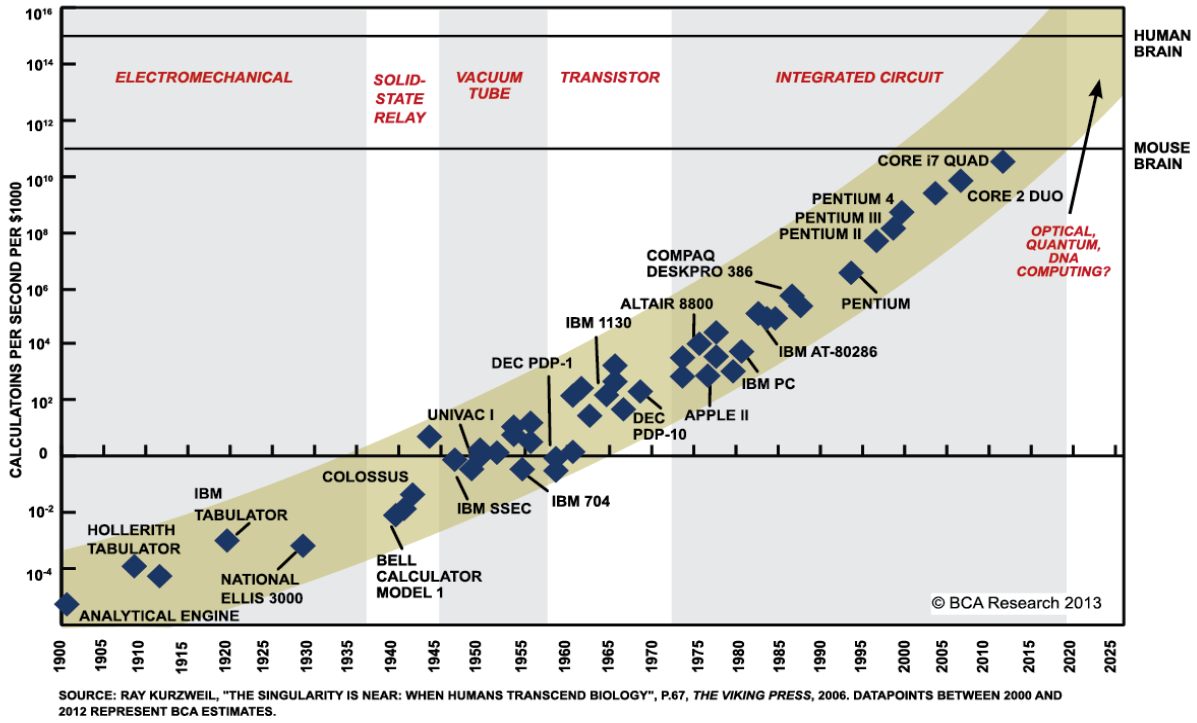


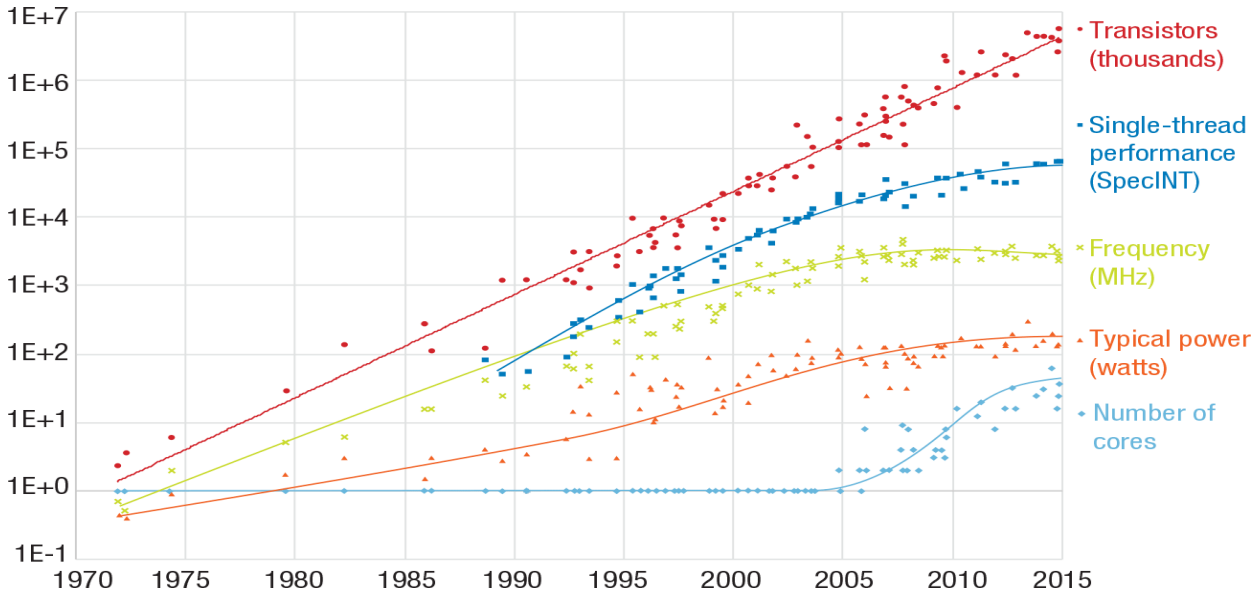
Figure 1.1: A plot of CPU transistor counts against dates of introduction; note the logarithmic vertical scale; the line corresponds to exponential growth with transistor count doubling every two years.

But it's not only the Moore's Law we need to worry about. Until 2005, Dennard Scaling provided a way to shrink the transistor size by scaling the threshold and supply voltages by about the same factor as the feature size, while keeping the power density roughly constant. Consequently, clock speeds of transistors increased and also due to this the computational speed of computers improved. This shrink in size also allowed to put more transistors on a single chip thereby keeping the Moore's Law alive and fit.

However, After 2005 or roughly when Pentium 4 came out, the transistor size was so small already that it was not possible to shrink it anymore, at least by the popular traditional method of Dennard Scaling . Greater challenges faced in preventing current leakages in such small transistors lead to breakdown of Dennard Scaling. And many researchers started to doubt if Moore's Law will even continue to work.

But the need to fulfil the demands of ever increasing workloads motivated researchers to come up with idea of increasing the number of cores in the processors instead of focusing on better clock speeds. This lead to the beginning of the era of a multi-core CPU. Putting more cores on the single chip not only allowed computational speed gains but also the number of transistors on single chip increased thereby still keeping the Moore's Law valid. Everyone thought that as long as we know how to program the multi-core CPU we can always increase the computational speeds by increasing the number of cores on a chip and thus can easily keep up with workloads.

Even though, due to multi core CPU researchers started placing more transistors on a single chip, switching power-per-transistor did not scale proportionately, this lead to increase in power density of chips and thermal hotspots. Researchers soon realized that although they can increase the number of transistors on the single chip, but if they were to make a CPU rather than a barbecue of that chip, then they can only keep some of those transistors running at a time, and thus keeping the other transistors off or dark. This led to rise of phenomena called "Dark silicon phenomena"[8], due to which only a proportion of transistors can be allowed to run on a single chip. And since all transistors were not running at the same time, the rate of increase of computational speeds started to decrease. And currently there is pressing need to look for platforms that can provide better computational gains.



Source: Data collected and plotted by Prof. at Stanford Univ., M. Horowitz, F. Labonte, O. Shacham, K. Olukton,L Hammod and C. Batten

Figure 1.2 Graph showing the trend of no. of transistors, cores and frequency of CPU's

The Fig 1.2 is taken from the research work of Professors at Stanford University which clearly represents this trend

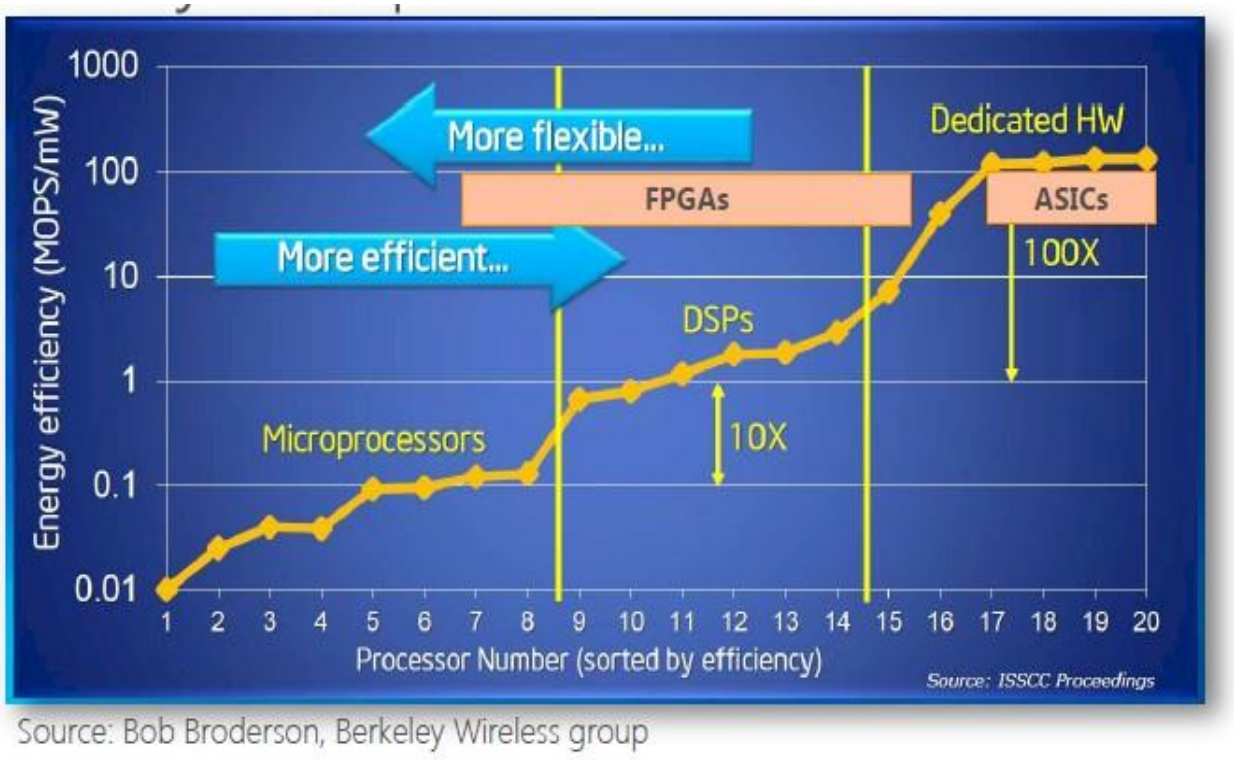


Figure 1.3: Relation between Flexibility and efficiency of different Computing platforms.

The graph shown above provides a relation between flexibility and efficiency of different Hardware Platforms. Microprocessors can compute almost everything comparatively easily, but inefficiently. Digital signal processors are 10X efficient than Microprocessors. While ASICs (Application Specific Integrated Circuits) are even 100X faster than DSP's but they have almost no flexibility. Once a circuit for ASIC is designed to perform a particular task it cannot be reconfigured, if one is even interested in making slightest changes to the earlier design they will have to start with completely new one again, and also amount of time to come up with such designs is way longer than programming normal computers. That's what makes Field Programmable Gate Arrays a platform of our interest. FPGAs provides sufficient flexibility to reconfigure the logic, in some cases even on the fly and at the same time also provides a very high gains in computation speed in comparison to traditional multicore CPU and DSP's. But their popularity as a computational platform has been restricted because of the time, patience, and level knowledge required to work at lower levels of programming, and write codes in hardware description

languages and directly deal with digital circuits. Also lack of availability of widespread platforms, unavailability of easy to use compilers, very time consuming debugging methods makes them very unpopular among programmers, even though the computational gains offered by them are considerably high.

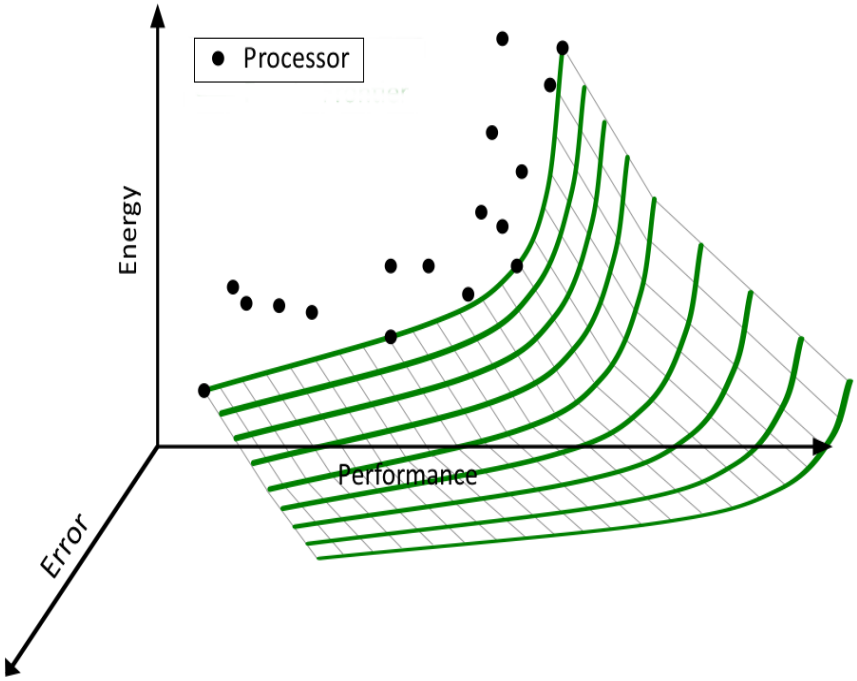


Figure 1.4: Pareto Efficiency frontier that shows trade-off between performances, error, and energy of computing platforms.
Source: By Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, Doug Burger, Communications of the ACM, Vol. 58 No. 1, Pages 105-115

Another reason why we are motivated to use FPGAs is the wide scope of making approximations when working with speech processing and machine learning algorithms. Hardware implementation of such algorithm on FPGA also results in some loss of accuracy, but fortunately, applications dealing with 'noisy' real word signals like speeches always have a scope of approximations. Making these approximations further provides more computational gains in speed. Modern computers are devised to compute accurate results even when it is not required. Approximate Computing [9] can help computers by providing an ability similar to a human brain to scale the degree of accuracy needed for a given task.

1.3 Objectives:

The project is divide into two parts. In first, we develop an algorithm for MFCC feature extraction process and train the Gaussian Mixture Models using the collected features both for spoof and human speeches. Once we have an algorithm for the complete feature extraction process on software, we breakdown the algorithm into parts that can be put on digital circuits. And thus during the second stage, we develop the hardware architecture to accelerate the MFCC feature extraction process. So, we have the following objectives.

1. Write an algorithm for MFCC feature extraction process on Matlab.
2. Train the Gaussian Mixture Models for both human and spoof dataset.
3. Experiment to choose better parameters for the feature extraction process, to improve EER.
4. Propose hardware architecture for feature extraction process.

Chapter 2

Literature Review

In this chapter we discuss the various terminologies and pre-requisite knowledge required to understand the thesis. First section has concise information of how human speech is produced. Then we will discuss different feature of human speech. Later, this chapter covers Gaussian Mixture Models. In the end, we will discuss about the dataset of speeches that we have used for experimentation, and concise information about of Xlinx System Generator, which is a tool that eases the method of implementing Simulink based DSP algorithm on FPGA boards.

2.1 Human speech production

Speech production is the means by which ideas are converted into speech. This involves choosing the words, the framework of different grammatical forms, and then the converting them into sounds using the vocal system. Speech production can be sometime instantaneous for example when someone is reading a particular content from a paper, here the speaker doesn't think or frame his sentences or make choices between different words. Speech production and language production are different. Language are collection of symbols which represent information, like hand signals and actual speech production is not necessary. Usually production of speech happens due to the pulmonary pressure which is generated by the lungs. This pressure produces sounds by phonation by the glottis in the larynx, which is transformed by the vocal tract into different types of vowels and consonants. Speech production can sometimes also, happen without the aid of the lungs and glottis in an alaryngeal speech by utilizing the parts of the vocal tract. Physical framework of the throat, human nose, and vocal chords helps in production of various sounds.

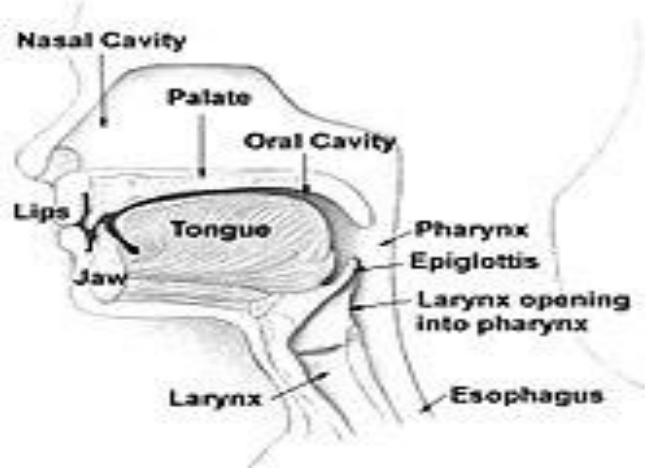


Figure 2.1: Human vocal apparatus used to produce speech
source: Wikipedia

2.2 Different Speech features

The speech produced by an individual carry the features of the source of production with it, these features are broadly classified into two categories i.e. Amplitude based or Phase based features. These features helps in identifying the individual.

2.2.1 Log Magnitude Spectrum

It is an amplitude based feature of speeches. Log magnitude spectrum feature directly takes the logarithm of the magnitude part of the signal's Fourier transform $X(t,\omega)^{ej\theta(t,\omega)}$:

$$L(t,\omega)=\log(|X(t,\omega)|)$$

This feature contains a lot of details about the speech signal. It constitutes of details regarding harmonic structure, pitch and formant. The logarithm here helps in reducing the dynamic expanse of the magnitude spectrum, which makes it fit to use as a feature.

2.2.2 Mel-Frequency Cepstral Coefficients (MFCCs)

It is an amplitude based feature of the speech. It is the most commonly used feature in automatic speech recognition (ASR). MFCC tries to relate to the logarithmic judgment of the pitch and loudness of human auditory system and attempts to remove speaker-dependent characteristics. It tries to correlate the spectrum of speech signal on logarithmic scale. This feature is discussed in further details, later in this report.

2.2.3 Cosine Phase

It is a phase based feature of speech, which is specifically is used for detecting artificial speech detection. It uses very direct relation with phase transformation. After the Short Time Fourier Transform

$$X(\omega,t) = |X(\omega,t)|e^{j\theta(\omega,t)}$$

is calculated for each of the speech frame, the short-time phase spectrum $\theta(\omega,t)$ is then unwrapped to eliminate any discontinuity. After this normalization is done by limiting the resultant in range of [-1, 1]. Later, a DCT is applied to this normalized result to obtain the final result [10].

2.2.4 Relative Phase Shift

Relative Phase Shift [11] features are used to identify artificial speech by the harmonic phase difference between human and artificial speech. To calculate this feature, first calculate the instantaneous phase of fundamental frequency by the following formula.

$$x(t) = \sum_{k=1}^{K} A_k cos(\phi_k(t)), \phi_k(t) = 2\pi F_0 kt + \theta_k$$

Here x(t) is the Fourier transform of instantaneous phase $\phi_k(t)$ at the k^{th} harmonic order of fundamental frequency F_0 and θ_k is the initial phase shift of the k^{th} component. RPS values R_k for harmonic order k can be calculated as follows:

$$R_k = \phi_k(t_0) = \phi_k(t) - k\phi_1(t) = \phi_k - k\phi_1$$

For normalizing the RPS in normalized in interval of [-π, π].

2.3 Gaussian Mixture Models

A Gaussian mixture model is a probabilistic model of a dataset, that is based on the principle that that all points of a data belongs to a finite number of Gaussian distributions. Mixture models don't require an understanding of which subpopulation a data point belongs to, allowing the model to determine the subpopulations by itself. The Gaussian mixture model for speech representation assumes that a M component mixture model with component weights $P(w_m)$ and parameters θ_m can represent the spectral shape. The formula form any mixture model is

$$p(x|\theta) = \sum_{m=1}^{M} P(w_m)\, p(x|w_m, \theta_m)$$

where $P(w_m)$ is prior probability of a components. For GMM the formula is changed to,

$$p(x|w_m, \theta_m) = \left(\frac{1}{(2\pi\sigma_m^2)^{\frac{1}{2}}} \right) * \exp\left[-\frac{(x-u_m)^2}{2\sigma_m^2} \right]$$

Where u_m is the mean and σ_m the standard deviation for component w_m.. In case of speeches, GMM helps by providing a soft classification [12]. Soft classification is very much necessary when dealing with highly random signals like speeches.

2.4 Dataset

The dataset taken from the Automatic Speaker Verification Spoofing and Countermeasures Challenge (ASVSpoof2015) database. Genuine speech is collected from 106 speakers (45 male, 61 female). Collected speeches contains no background noise. Different spoofing algorithms are used for generating artificial voices. The dataset is divided into three groups training, development and evaluation.

The spoof data is generated using following methods:

- S1: The spoof data that belongs to this set uses very fundamental voice conversion technique of frame selection [13].
- S2: It is another VC technique that modifies the first coefficient of MFCC feature while keeping the other details of voice same.
- S3: It is SS method, Hidden Markov Model based speaker adapted speech synthesis is [14] built by using HTS toolkit. System requires 20 utterances of targeted speaker.
- S4: It works very similar to S3 but requires 20 more utterances than S3.
- S5: I is VC technique which uses publicly available tool kit called Festvox system. It uses the principle of maximum likelihood estimation to get better result in VC [15].

Types of Subsets	Number of Speakers		# Number of Utterances	
	# of Male	# of Female	# of Male	# of Female
Training dataset	10	15	3750	12625
Development dataset	15	20	3497	49875
Evaluation dataset	20	26	9404	200000

Table 2.1: Distribution of ASVspoof 2015 challenge

2.5 Xilinx System Generator

System Generator for DSP™ is the best high-level software from Xilinx for developing high performance DSP systems using Xilinx All Programmable devices. System Generator provides a method to develop to complex DSP system with minimal time in comparison to traditional RTL methods.

- It allows to develop DSP algorithms compatible with different FPGA boards.
- It provides system modeling and code generation for Simulink models and Matlab.
- It very efficiently integrates different DSP hardware components, embedded IPs, RTL and Matlab.

That is why we have chosen Xilinx System Generator as the tool for implementing the algorithm on hardware.

Chapter 3

Software Implementation

In this chapter, we start by discussing Mel-Frequency Cepstral Coefficients, and their complete feature extraction process. With each stage, we also discuss various parameter that we have adopted in the system. During the middle of this chapter we have discussed how Gaussian Mixture Models are used as a classifier. In the end, results obtained by the software implementation are discussed.

3.1 Mel Frequency Cepstral Coefficients

MFCC is the most widely used amplitude-based feature for the speaker or speech recognition [16] [17]. MFCC is based on this principle that human ear is sensitive to frequency of audio signal on logarithmic scale. So, if the actual frequency is changed 10X times human ear still senses as if the there is only twice increase in frequency. And this quality is essentially captured in MFCC feature, this helps in differentiating human speech from spoof on the basis of how human ear must have been by correlating spectrum on log scale. Moreover, this feature carries the vocal tract dynamics and pulse train associated with the glottal motor control.

The MFCC is defined as

$$C_i = \sqrt{\frac{2}{N}} * \sum_{j=1}^{N} L_j * \cos\left(\frac{\pi i}{N(j-0.5)}\right)$$

where N is the number of Mel-frequency bins of log spectrum L and i is the number of cepstral coefficients.

3.2 Feature Extraction Process

The flowchart of MFCC feature extraction process is shown in the following figure.

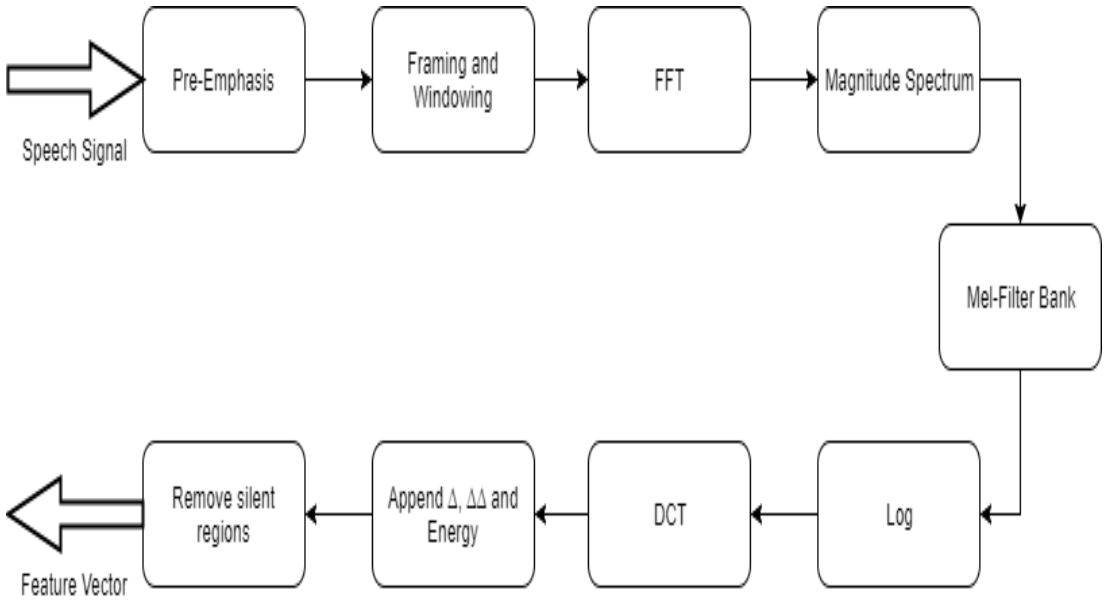


Figure 3.1: Flowchart of MFCC feature extraction process adopted.

3.2.1 Pre-Emphasis

Pre-processing of a signal has a major impact on the MFCC feature extraction process. Pre-processing helps in enhancing the amplitude of high frequency signal in comparison to lower the frequency and thus helps in preserving more information present in high frequency signals. This further helps in smoothening of silent regions.

$$y(n) = x(n) - \alpha * x(n-1)$$

Where α is 0.97 and x(n) is the input speech signal and y(n) is the output of Pre-Emphasis Filter.

3.2.2 Framing and Windowing

Speech signal is divided in windows of 25ms per frame and each frame is shifted from the other by 10ms. Since the sampling frequency of each signal in our dataset 16Khz so each window consists of 400 samples and with a shift of 80 samples. Each frames is multiplied with Hanning

window to preserve the continuity of complete signal while doing STFT. In this stage, we also mark the silent regions of the speeches as they do not contain any speech information and only disturb the pattern of features collected from voiced part, features belonging to these marked frames is discarded in the end. If a frame contains even a single sample greater than a particular threshold than the frames is marked as voice frame otherwise non-voiced or silent.

3.2.3 Short Time Fourier Transform

From past research results and after different trials it has been observed that 512 point FFT is sufficient enough to preserve the required information in Frequency domain. So in order to obtain 512 points 112 zeros are appended in each frame. And in the next stage magnitude of spectrum is calculated.

3.2.4 Mel-Filter Bank Integration

The human ear perceives frequency on non-linear scale across the audio spectrum. So instead of linear spectrum analysis there is a need of non-linear analysis of spectrum and filter-bank analysis provides a solution to this problem. The filters used are triangular and they are equally spaced along the Mel-scale. Mel scale relates perceived frequency or pitch, of pure tone to its actual measured frequency. Conversion from linear frequency to Mel Scale is given by this formula:

$$\mathrm{M(f)} = 1125 * \ln\left(1 + \frac{\mathrm{f}}{700}\right)$$

Conversion from Mel scale to frequency:

$$\mathrm{M^{-1}(m)} = 700\left(\exp\left(\frac{\mathrm{m}}{1125}\right) - 1\right)$$

20 triangular Mel filter banks are created from 30Hz to 8000Hz and energy of audio signal in each such triangular filter bank is calculated. Then we create our filter-banks. The first filter-bank will start at the first point, reach its peak at the second point, then return to zero at the 3rd point. The second filter-bank will start at the 2nd point, reach its max at the 3rd, then be zero at the 4th etc.

A formula for calculating these is as follows:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

Where M is the number of filters we want, and $f()$ is the list of M+2 Mel-spaced frequencies. Further Log of these 20 coefficients is calculated to transform the multiplication into addition, this step is a part of cepstral computation. Only initial 12 coefficients are enough to preserve the characteristic.

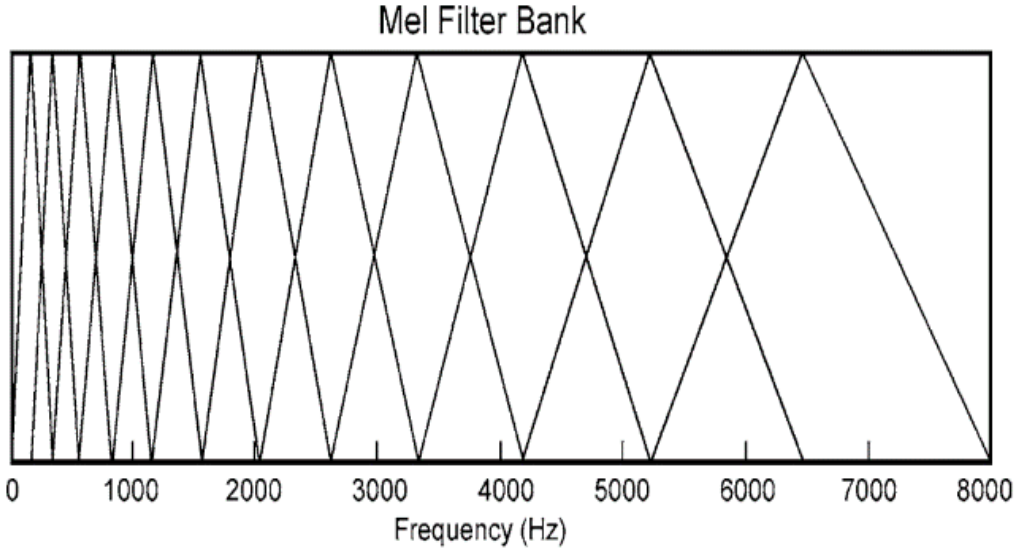


Figure 3.2: Triangular Mel Filter banks

3.2.5 Append Δ, ΔΔ MFCC and Energy

In this stage we append the Δ, ΔΔ and energy coefficients in the previously extracted feature of MFCC. Formula for calculating Δ of function is given by:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

Where d_t is a delta coefficient, from frame t computed in terms of the static coefficients c_{t+N} to

c_{t-N}. A typical value for N is 2. ΔΔ (Acceleration) coefficients are calculated in the same way, only difference is that they are calculated from Δ coefficients and not static coefficients.

These Δ and ΔΔ coefficients retains the temporal characteristics of the speech signal. Energy of each frame is also added to feature vector to give further details about the intensity of the frames. So finally we obtain a feature vector of size 37 i.e. 12 MFCC coefficients, 12 Δ MFCC, 12 ΔΔMFCC coefficients and 1 Energy coefficient. In the next stage we remove all the features which belong to silent regions in the speech signal. It is essential to remove the features belonging to silent regions in the end because otherwise we will not receive the accurate temporal details for few frames by Δ functions.

3.3 Gaussian Mixture Models

We use GMM for classifying the speech signal as either spoof or human. GMM provides a method of soft classification, which is very essential when dealing with random signals like speeches. These feature vector obtained from the training data-set of ASVspoof challenge is used to train the GMM. From experimentation it has been observed that, 10 Gaussian distributions are sufficiently good for both spoof and human dataset. If we increase the number of Gaussian Distributions in GMM than EER usually decreases

GMM for both natural (λ_{human}) and synthetic (λ_{spoof}) speeches are developed. $P(Y/\lambda_{human})$ this gives the likelihood of feature to belong to human and similarly for spoof speech.

$$P\left(\frac{Y}{\lambda}\right) = \left(\frac{1}{N}\right) * \sum_{n=1}^{N} P\left(\frac{Yn}{\lambda}\right)$$

To perform the synthetic speech classification of an utterance, the system tests likelihood values $P(Y/\lambda_{human})$ against $P(Y/\lambda_{spoof})$, where Y is feature vector of the test utterance.

$$\Lambda(Y) = \log(\frac{Y}{\lambda_{human}}) - \log\left(\frac{Y}{\lambda_{spoof}}\right)$$

Then according to the log-likelihood ratio (Λ) the utterance is classified as human or spoof by comparing against a pre-set threshold value θ, this θ is obtained corresponding the EER for

evaluation dataset. If Λ is greater than θ it is human otherwise the speech is spoof. The obtained value for θ in our system is 2.37.

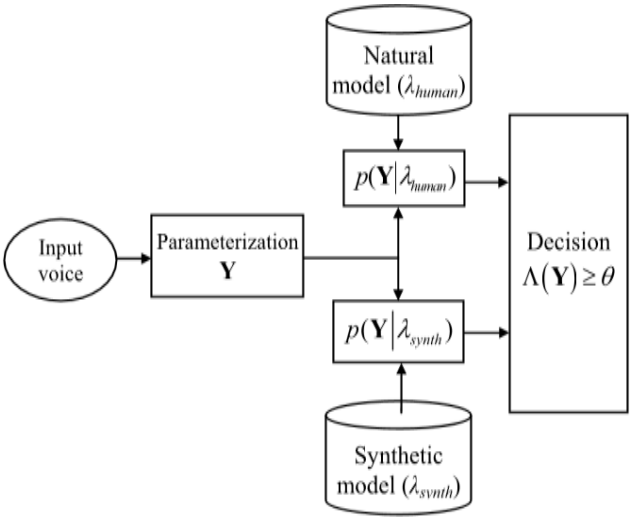


Figure 3.3: Classifier system structure
source: [6]

3.4 Results of Software Implementation

The results obtained from the feature of size 37 (12 MFCC, 12 Δ MFCC, 12 ΔΔ MFCC, 1 Frame-Energy) for test dataset of ASVspoof 2015 challenge is

Equal Error Rate = 6.37%

This result is obtained for 10 Gaussian distributions for both human and spoof GMMs.

Chapter 4

Hardware Implementation

In this chapter we will discuss the hardware implementation of the above discussed algorithm. The extraction process of MFCC has previously once built on FPGA [18] but this architecture doesn't take account of many steps involved in our proposed MFCC feature extraction process. In the beginning this chapter covers the need to make the approximations in the previously discussed algorithm to implement it on the hardware. Then I will make similar approximation in the software and will analyze the modified features. In the end we will propose our hardware architecture and discuss in detail.

4.1 Approximation made for hardware implementation

Approximate computing is the idea that computer systems can let applications trade off accuracy for efficiency. It is a method in which the system deliberately exposes error to the application layer in order to save some resource. Approximate Computing also enhances computational speed. Developing applications using HDL's is always very time-consuming method, so it is always advisable to make suitable approximations when implementing an application on digital circuits.

The Approximations made in our circuit are the following.
In the hardware implementation, there is no sliding of the window by 10ms. Instead, each frame for FFT has a size of 512 points and directly 512 point DFT is taken for each frame. This is done to keep the data flow continuously running with time. In the current architecture, samples are not stored anywhere this allows it to work in any real-time conditions.

Since there is no sliding of each window, each adjacent window is not close enough to carry sufficient temporal details of each frame. Hence, keeping Δ, $\Delta\Delta$ MFCC and energy in features will only disturb the pattern created by 12 MFCC vector. So, Δ, $\Delta\Delta$, and energy vector are removed from the feature vector.

Even though logarithm is an important stage for complete feature extraction process, but the unavailability of accurate digital IP for the same as forced us to remove this stage from feature extraction process. Since most data in each stage is within a fixed range, so fixed point computations are used at all stages of this feature extraction process. Moreover, using fixed-point

computations over floating-point computations provide an advantage in computational speeds. All fixed points data sizes are computed by attempting different trials and making a proper trade-off between speed and accuracy.

Finally, we obtain a feature vector of size 12 which only corresponds to 12 MFCC feature coefficients.

4.2 Back to Software

Since we have made previously mentioned approximations in the hardware implementation of the algorithm it will be wise to make similar approximations in the software implementations of the same.

So, after making the approximations, following are the observations obtained from the new software implementation after each stage.

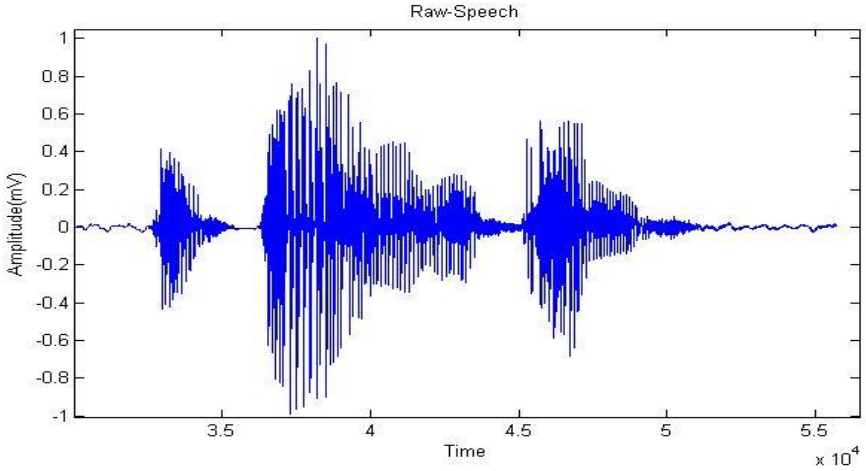


Figure 4.1: Input speech to the modified system

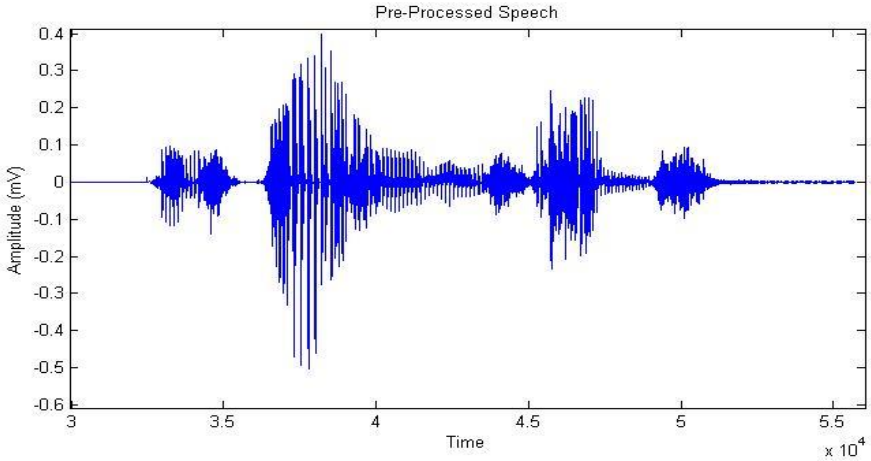


Figure 4.2: Signal after Pre-Emphasis

The result of Short Time Fourier Transform of the speech signal

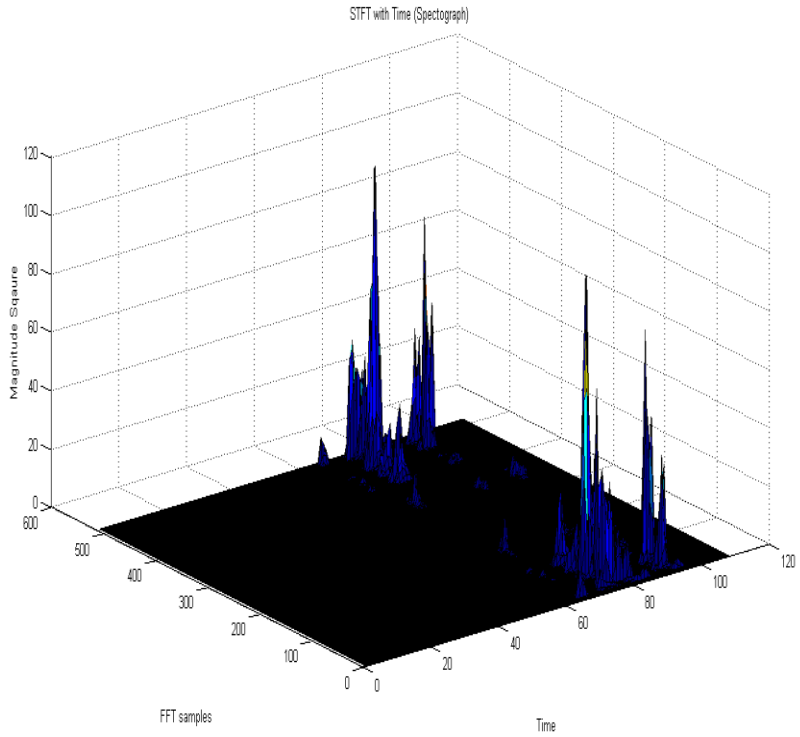


Figure 4.3: Short Time Fourier transform of Speech signal

The result of each Filter Bank Energy values with time

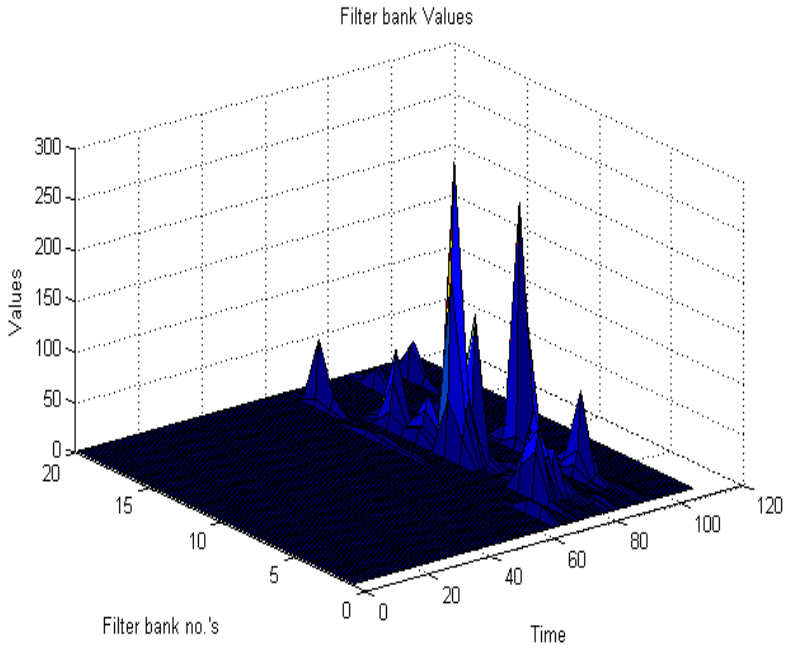


Figure 4.4: Output of Mel-Filter Bank stage

Result of 12 DCT coefficients with time.

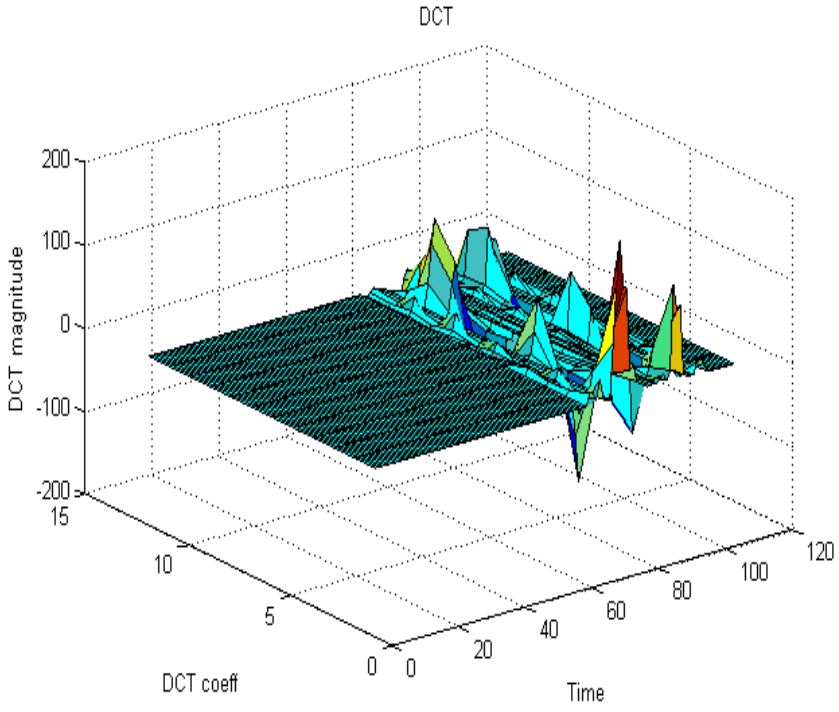


Figure 4.5: Output of DCT stage of feature extraction process

Final, MFCC feature is shown for a speech signal is shown with time.

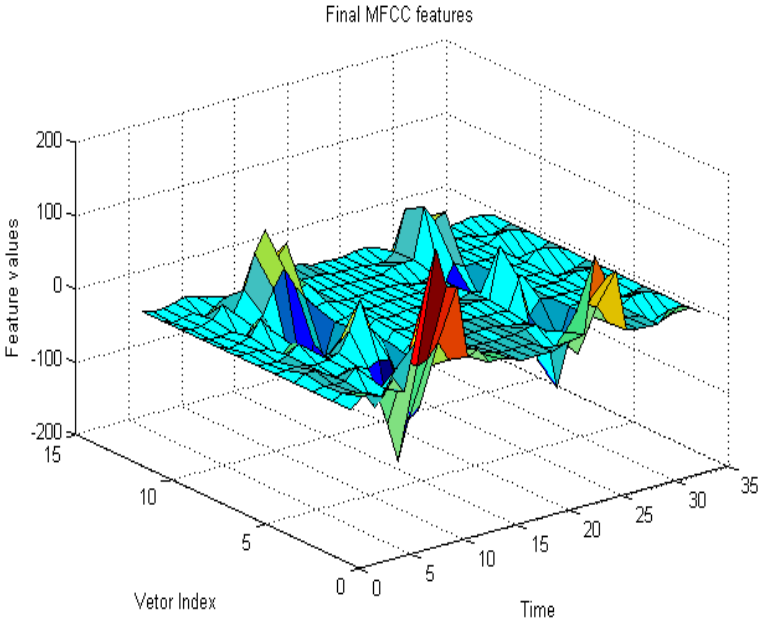


Figure 4.6: Final MFCC feature extracted from software implementation

4.3 Hardware Architecture

The above algorithm is broken down into individual stages, and digital circuit for each stage is designed and converted into a digital module. For the purpose of implementing the algorithm in form of a hardware architecture we have used Xilinx System Generator as a tool. Each block in the circuit uses fundamental digital Xilinx blocks which can be connected together to perform a particular task. These fundamental blocks can be easily realized on the reconfigurable fabric of FPGA. The features of each frame are concatenated in the final block. And at the end of the input speech signal, a particular signal is raised which triggers to send the features collected back to Matlab for further classification. So, Xilinx system generator very easily allows to do Hardware in loop co-simulation of the designed hardware architecture.

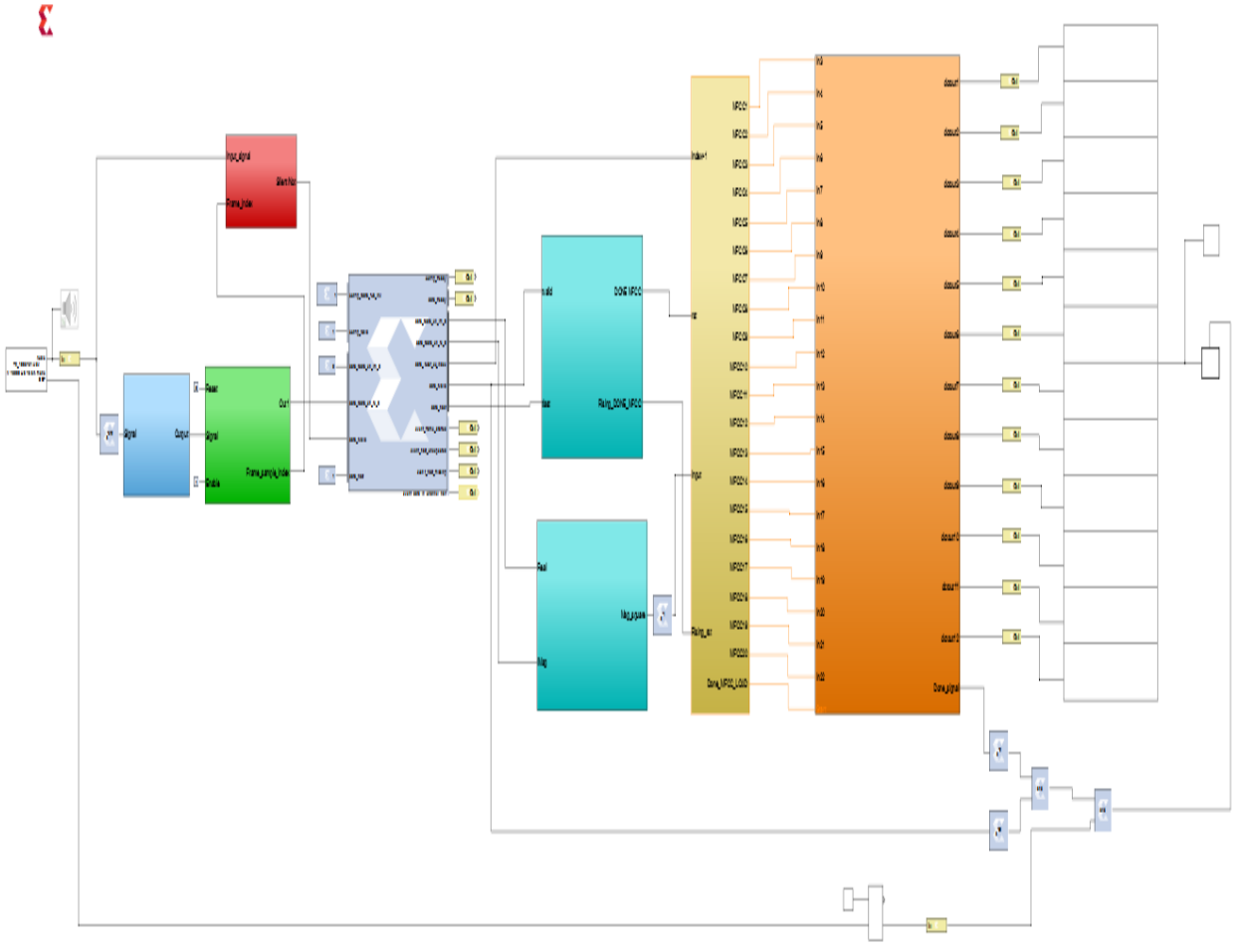


Figure 4.7: Figure of our designed Hardware architecture for MFCC feature extraction process.

4.3.1 Pre-Emphasis Module:

As the name suggests this module is designed for pre-emphasis of raw speech signal. The input pin (1) receives a signed fixed point samples of size 26 bits with 25 binary points at each clock from the source speaker. The digital circuit shown in the figure 4.8 implements the following function.

$$y(n)=x(n)-\alpha*x(n-1)$$

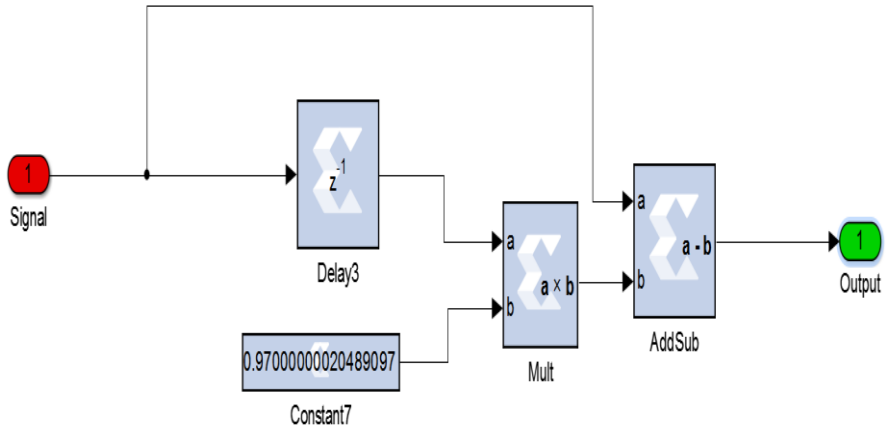


Figure 4.8: Digital circuit of Pre-Emphasis stage

The lagging sample is multiplied by a coefficient constant7 (α), which is stored in a register of size 32 bits with 31 binary point, and resultant is subtracted from the preceding sample. For each input sample, output sample of size 32bits with 31 binary points is received after required pre-emphasis, thus the circuit takes one clock duration for processing each sample.

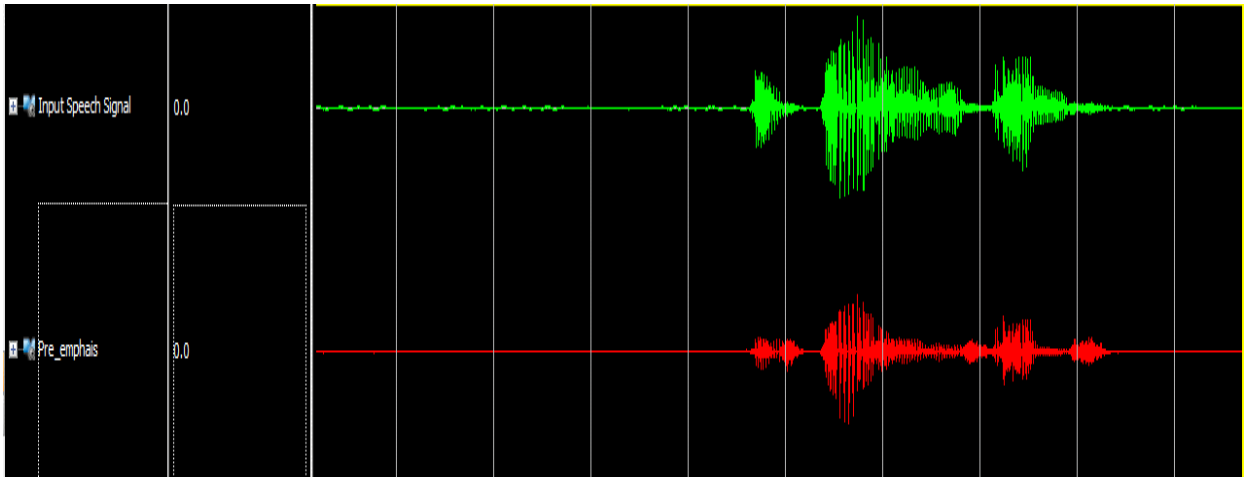


Figure 4.9: Input speech (Green) and Pre-Emphasis (Red) speech signals obtained from Pre-Emphasis module from Xilinx waveforms generator.

4.3.2 Framing and Windowing Module

Function of this module is to frame the speech signal into a window of 512 samples and multiply each singal with a Hanning Window to preserve the continuity of the signal. The ROM used in this module is configured with a depth of 512, and all hanning coeffecients are stored in each of the 512 registers of the ROM. The counter present in module assists in synchronously changing the address of the ROM, thus giving different hanning coeffecients at each clock tick.

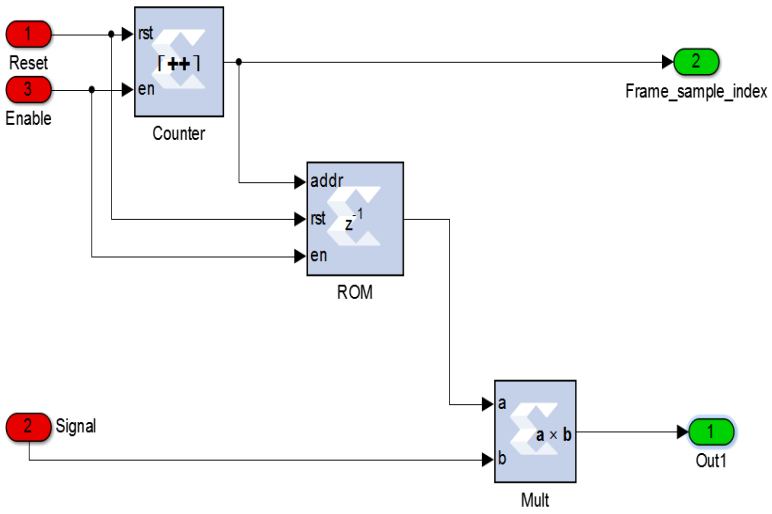


Figure 4.10: Digital circuit of Framing and Windowing stage

The "Out1" port of this module gives the signal samples after multiplication with hanning window. Similar to last module, this also produces a new output at each clock tick. "Frame_sample_index" output of this module gives the information of the index of current sample in a particular frame. Reset and Enable are hold at LOW and HIGH respectively.

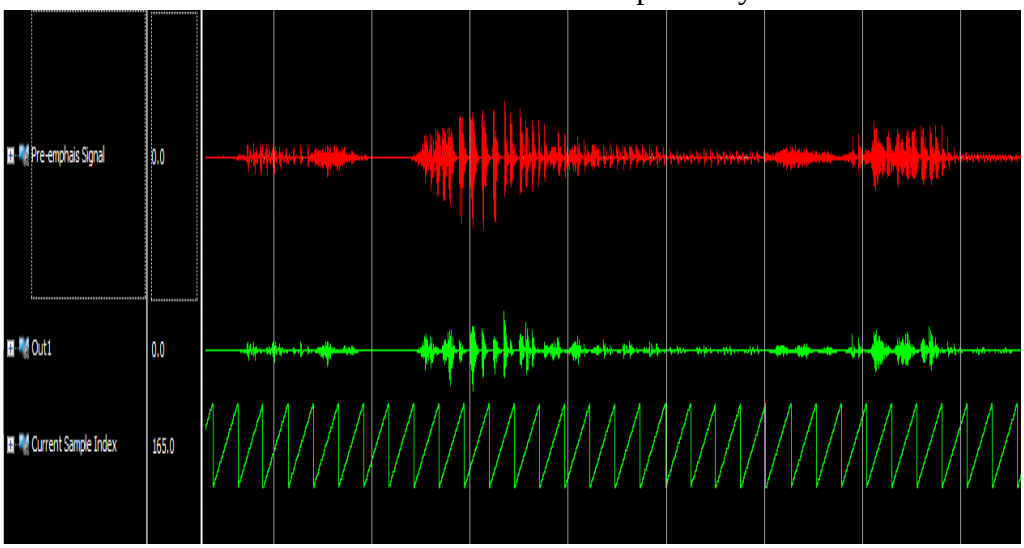


Figure 4.11: Waveform of Framing and Windowing stage.

4.3.3 Silent Removal Module

Function of this module is to mark if a particular frame is silent or not. If a particular frame contains any sample which has a magnitude greater than 0.03 then the output of the module is HIGH during the next window, this allows FFT9 module to receive both the signal sample frame and signal which identifies whether the frame is silent or not at the same time.

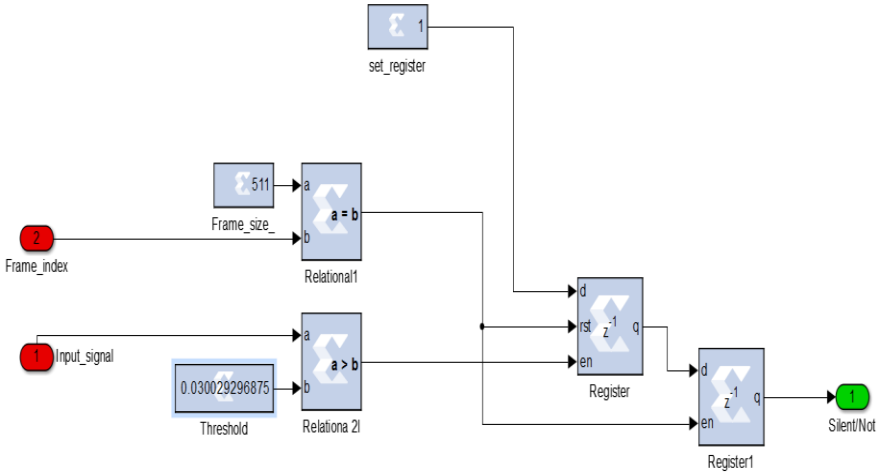


Figure 4.12: Digital circuit of Silent Removal Stage

At input 2 i.e. “Frame_index” signal receives the current index of the sample in a particular frame. If even a single sample frame is more than the threshold then the “en” pin of register is pulled HIGH, and this register allows the module to memorize that this frame is not silent and should give a HIGH output, while at the end of the frame the value of register is transferred to register1.

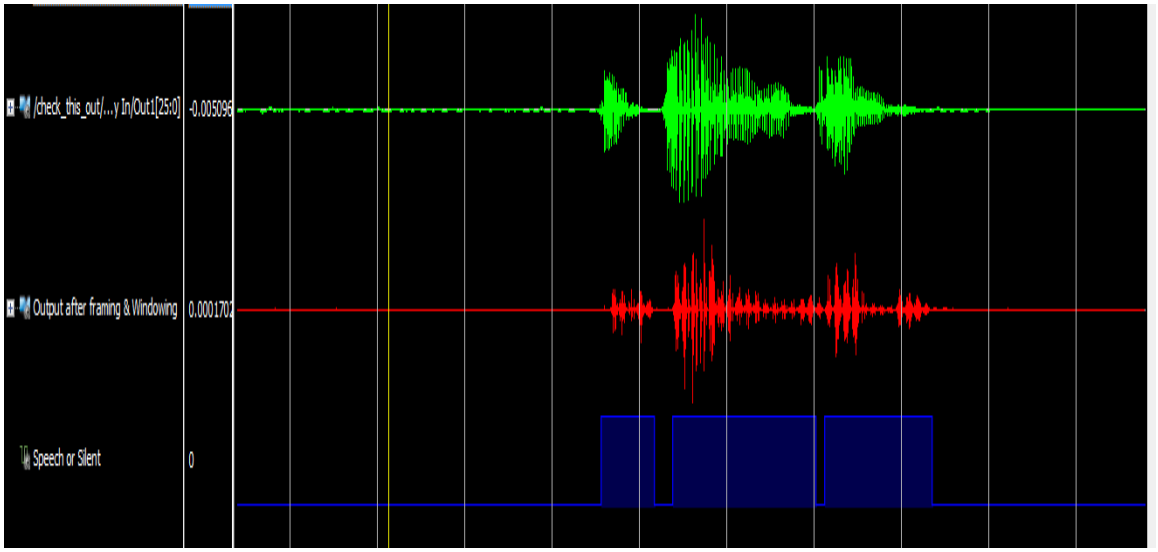


Figure 4.13 Input Speech Signal. Pre-emphasis Signal and Silent or Not output of this module.

4.3.4 Fast Fourier Transform V9.0 module

This IP is already available from Xilinx. Function of this module in the circuit is to take each frame as real and imaginary numbers in input and return FFT of those samples as real and imaginary numbers. This module uses AXI Interface for handshaking or proper communication of data with other modules. This module is configured to use Xtreme DSP slices on the board for butterfly arithmetic, which further accelerates the computation process. Module is configured to calculate the 512 point FFT.

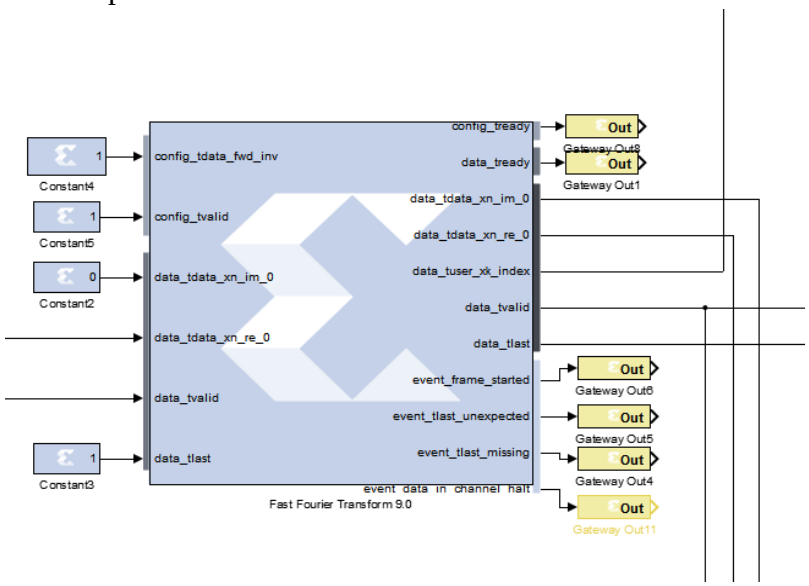


Figure 4.14: Fast Fourier Transform 9 Module

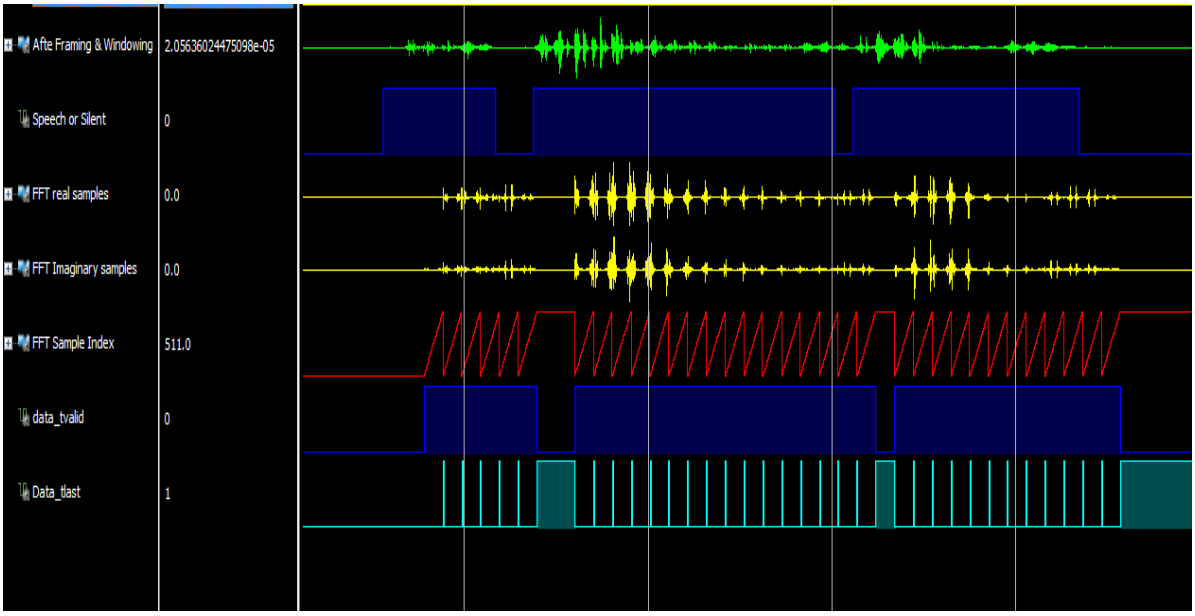


Figure 4.15 Input and output Signal of FFT 9 Module

4.3.5 Magnitude Square Module

As the name suggests this module, takes real and imaginary samples from FFT9 module and calculates the magnitude square of those samples. Output of this module is 32 bit unsigned fixed point number with 25 binary points.

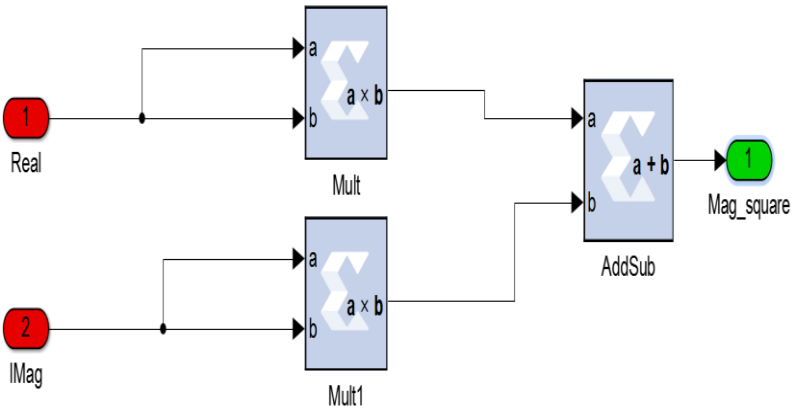


Figure 4.16: Magnitude Square Module

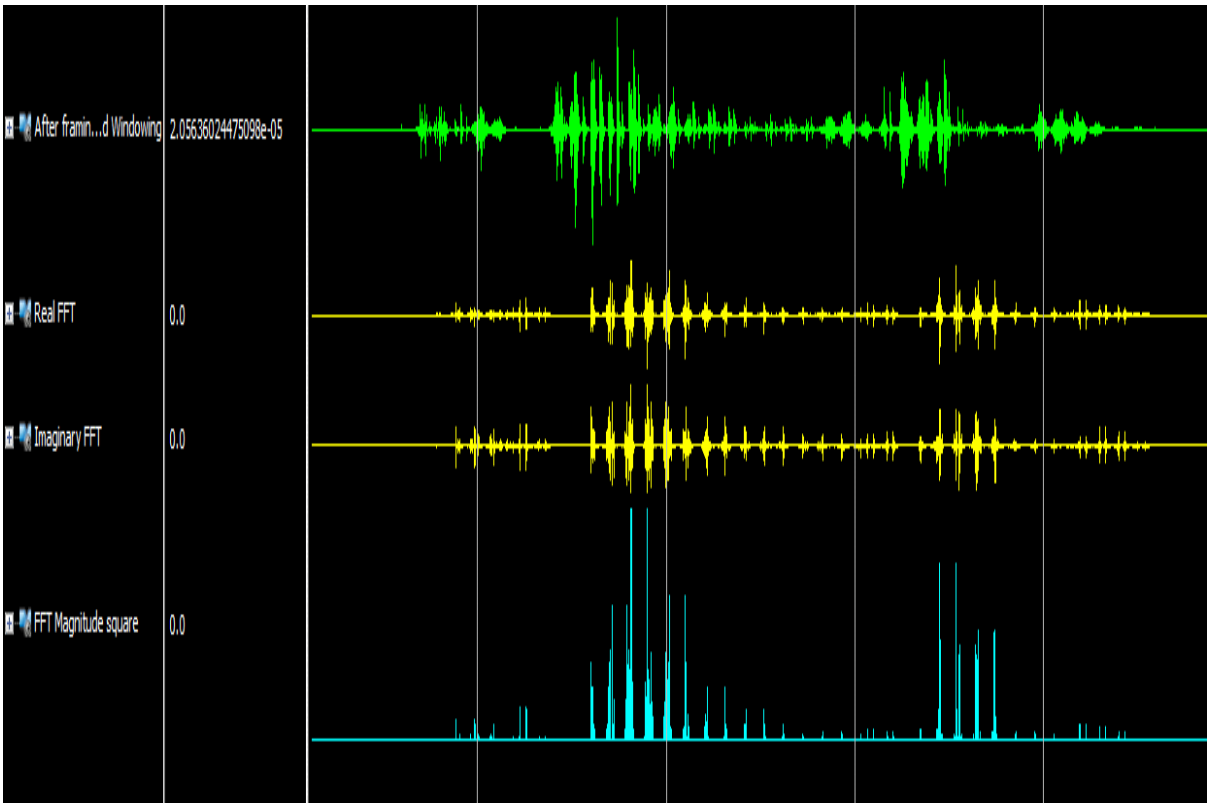


Figure 4.17: Waveforms framing & windowing stage, Real and Imaginary values of FFT, and Magnitude square of signal.

4.3.6 Mel Filter Bank Energy Control Module

Function of this module is to provide control signals to Mel Filter Bank Energy Module. It takes "data_tvalid" and "data_tlast" as signals in "tvalid" and "tlast" ports respectively, according to these signal this module produces two output signals. "Done MFCC" output is high whenever a particular frame is complete and new frame is about to start. "Rising_Done_MFCC" gives pulse at rising edge of "Done MFCC".

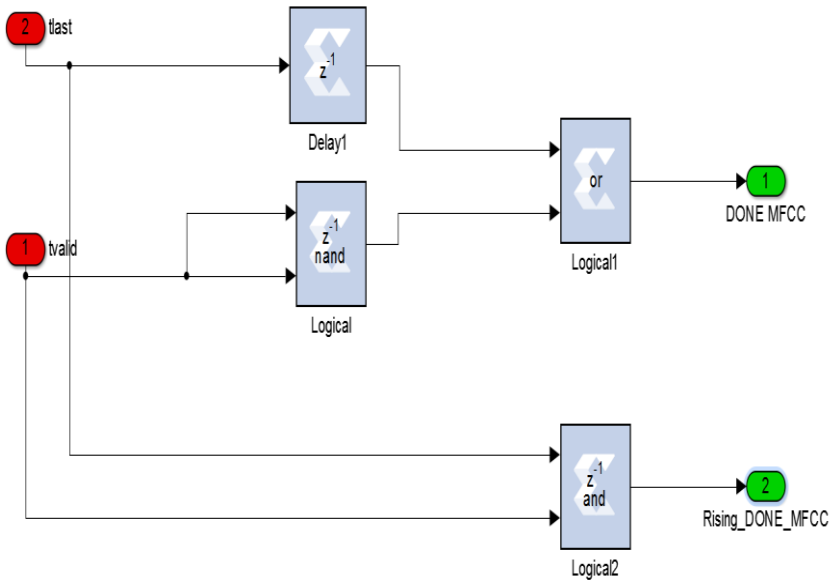


Figure 4.18: Mel Filter Bank Energy Control Module

4.3.7 Mel Filter bank Energy

Function of this module is to take FFT of each frame as an input and give 20 coefficient each one corresponding to energy stored in each Mel filter bank. The ROM is configured with a depth of 512, each register stores coefficients of triangular filter banks. These coefficients are multiplied with frame samples one by one and stored in the accumulator. At the end of each window, the data stored in accumulator is transferred to the register and the accumulator is cleared to store data for next frame. This happens for all 20 coefficients at the same time. This simultaneous computations of all coefficients at same time also adds to better computational speeds for FPGA implementation.

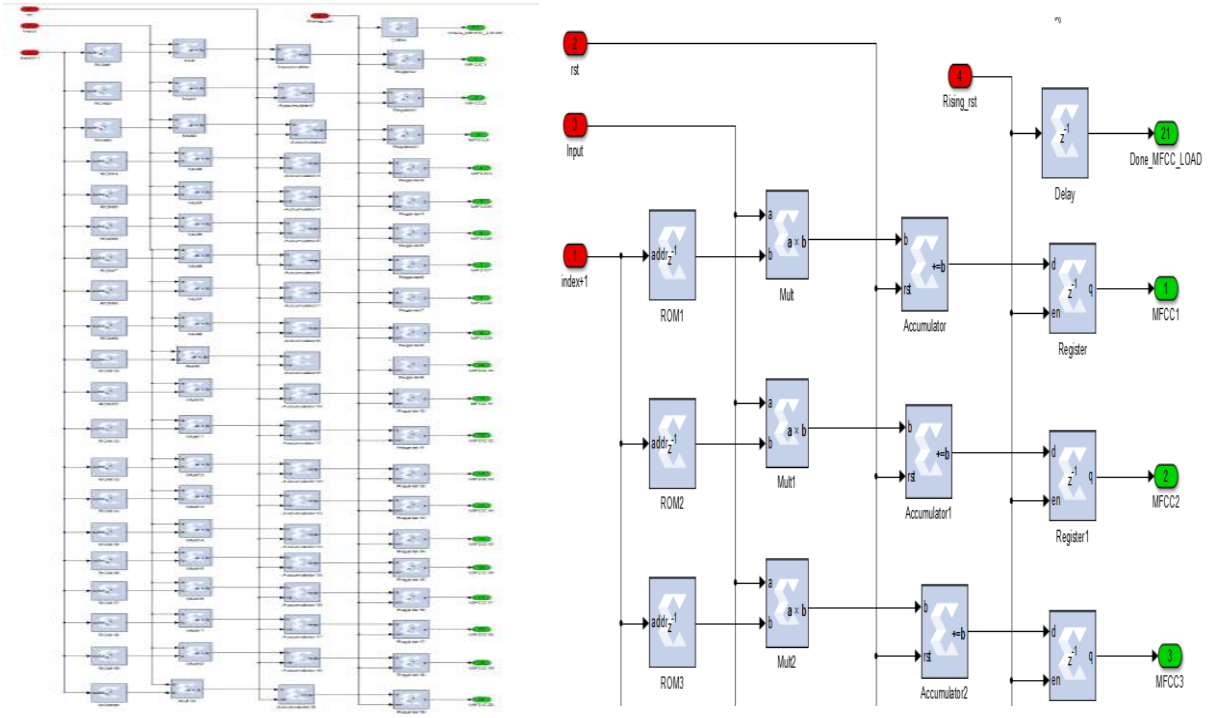


Figure 4.19 Left shows the complete Mel Filter bank Energy Module; on right, zoomed in version of the same module.

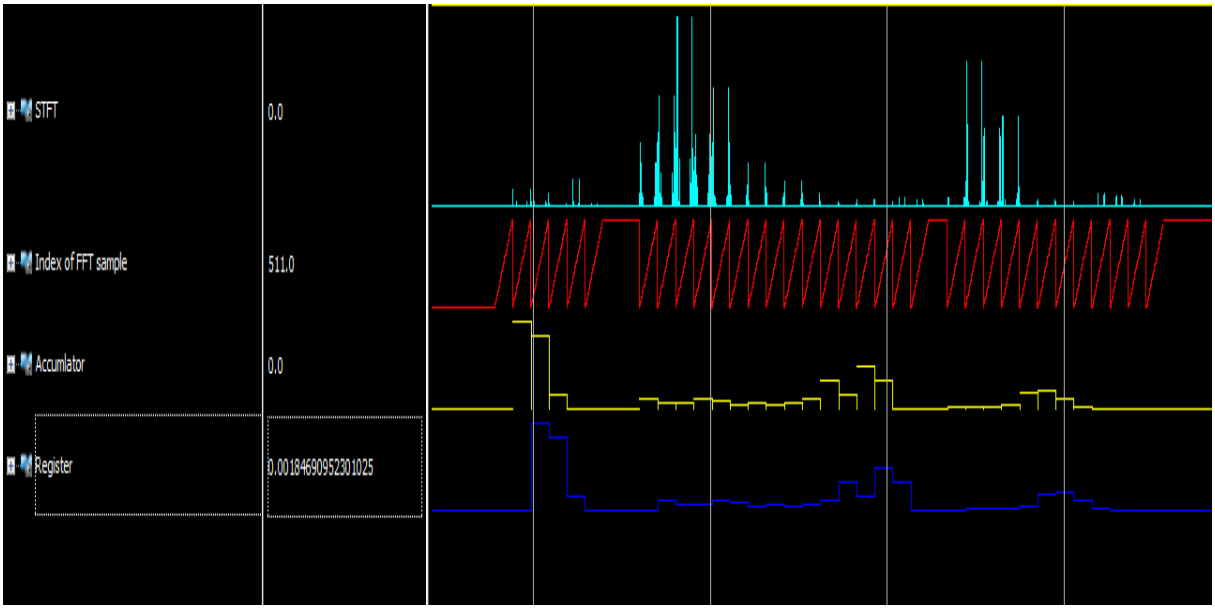


Figure 4.20: Waveforms of STFT, FFT index, Output of Accumulator and Final MFFC2 signal

4.3.7 DCT Module

The function of this module is to take 20 coefficients from the Mel filter bank module and find the DCT of those coefficients, later only initial 12 DCT coefficients are preserved. 20*1 mux helps in converting the parallel data into serial. This serial data is then transferred to the green module in this figure which computes the DCT.

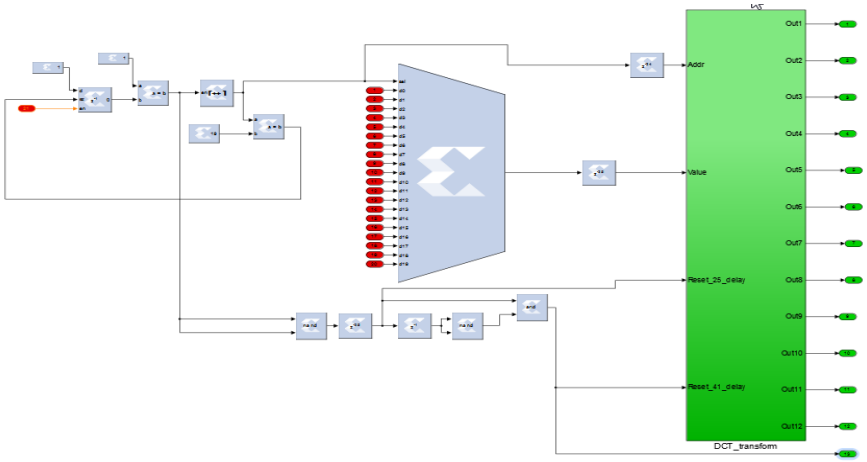


Fig 4.21

DCT Module

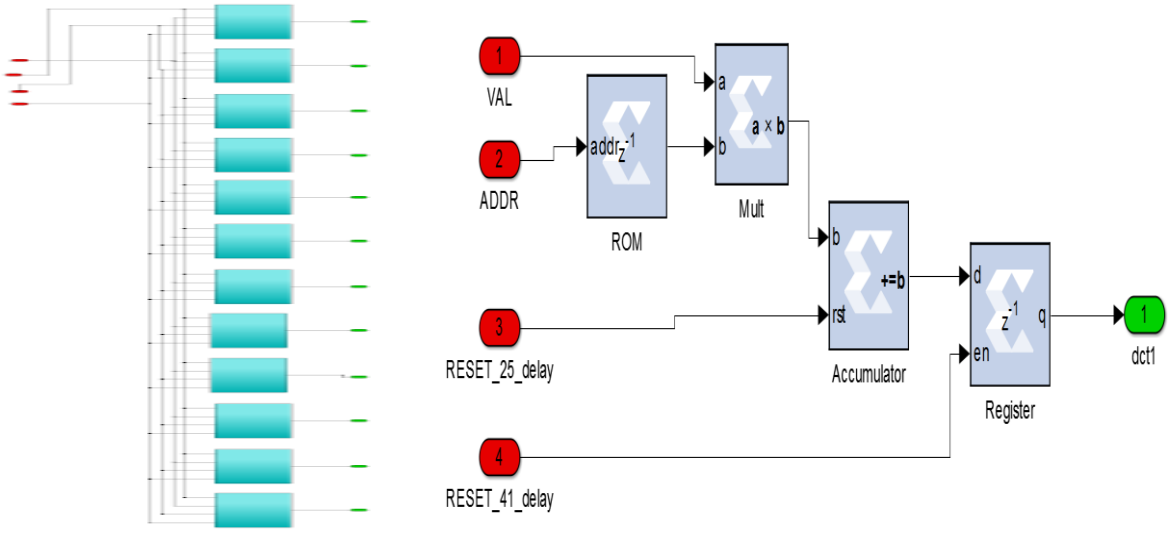


Figure 4.22: on left, internal circuit of DCT transform module shown in Figure 4.21; on right, internal Circuit of each of the blue module in left.

The ROM designed for this circuit has depth of 20, which stores coefficients for computing the DCT for 20 Mel filter bank energy coefficients obtained from the last stage. Similar to last module, all 12 DCT coefficients are computed.

Chapter 5

Experimental Results and Analysis

In this chapter, we discuss and compare the results obtained from both the implementation.

5.1 Observations made by Hardware implementation

The result of pre-processed speech from our designed hardware architecture.

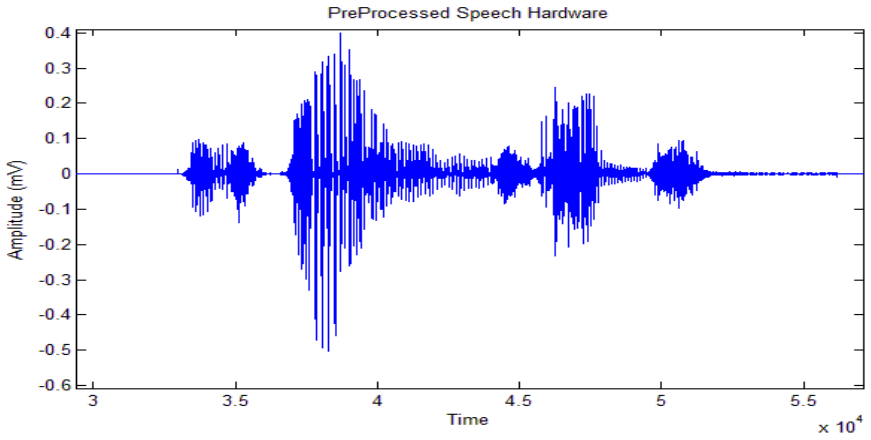


Figure 5.1: Pre-emphasis stage output from hardware

In the hardware implementation the FFT module is connected, so that it doesn't accept data if the frame is silent. Consequently, the STFT generated doesn't have Fourier Transform belonging to silent regions.

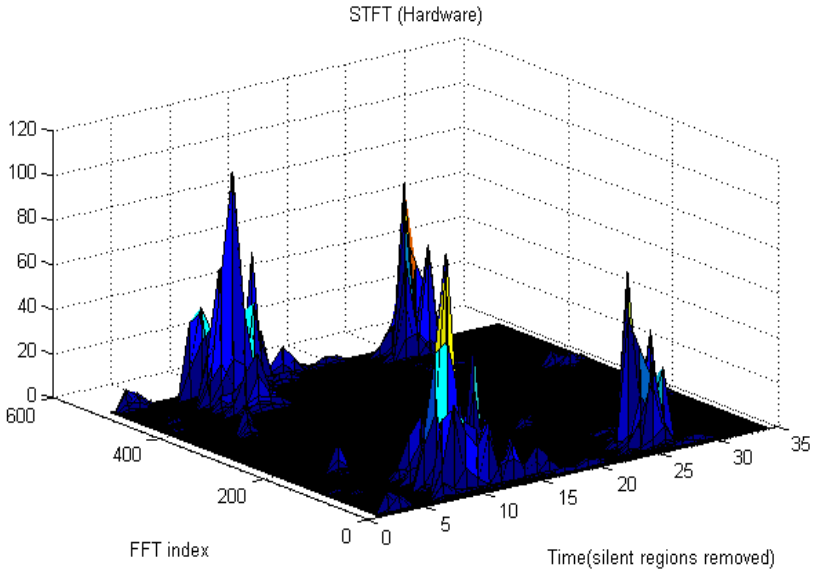


Figure 5.2:

STFT, from hardware

Mel-Filter bank energy values of the speech signal obtained from designed hardware architecture.

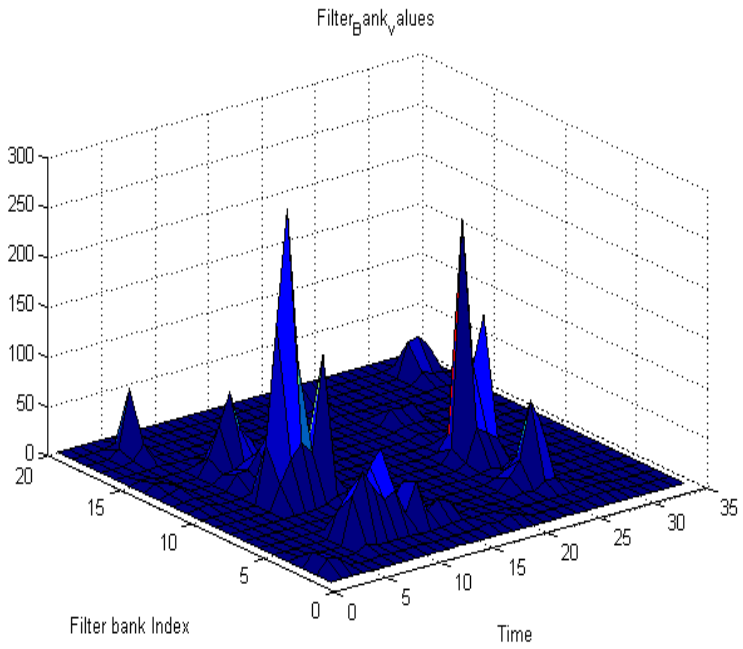


Figure 5.3: Filter bank energy output from Hardware

Final MFCC vector which obtained from our designed Hardware architecture.

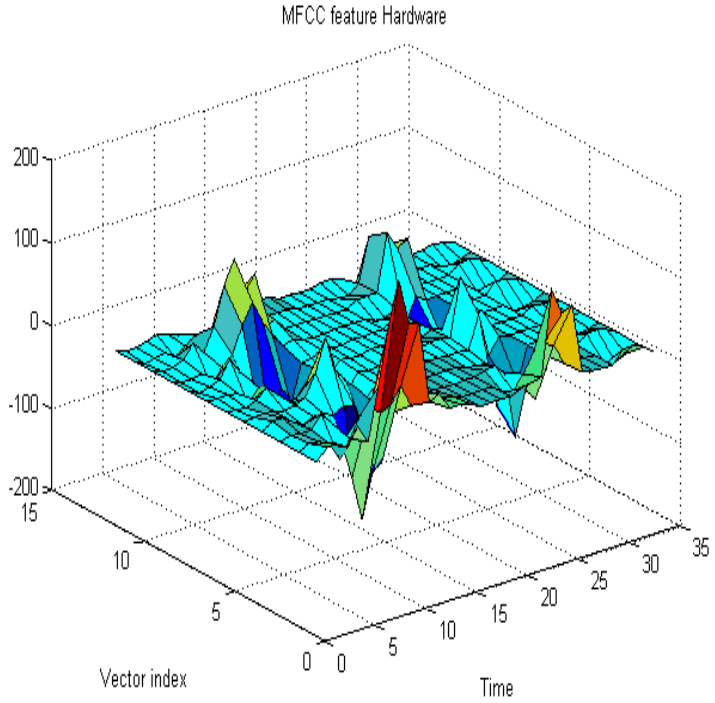


Figure 5.4: Final MFCC feature obtained from Hardware

5.2 Accuracy Comparison

As a consequence of the approximations made for the hardware implementation of the initially discussed algorithm the EER has increased.

EER for this new approximated algorithm is 15.24%.

Features obtained from both spoof and human dataset has 10 Gaussian distributions in their Gaussian mixture model. Each of the software and hardware architecture produces MFCC feature vector of size 12.

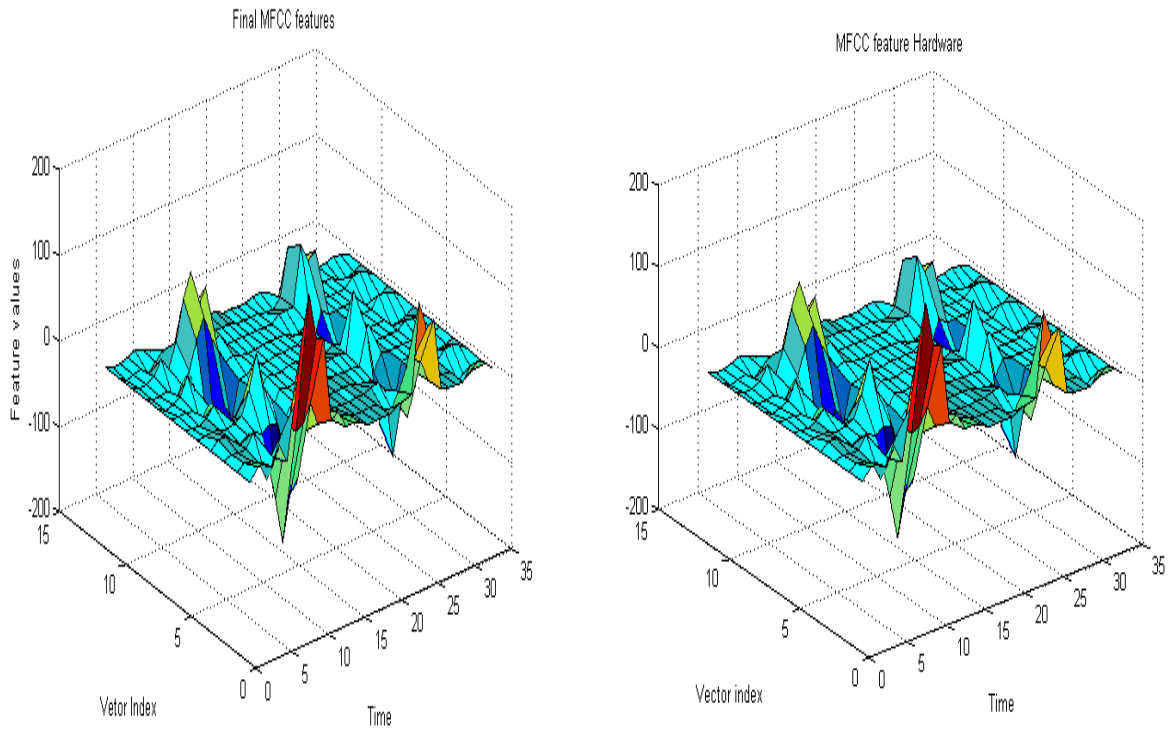


Figure 5.5 Final MFCC feature obtained from both software (Left) and hardware (Right).

Even though each of the results obtained from software and hardware looks very similar, the final MFCC feature obtained from the hardware differs from that of software by less than 7.06%.

5.3 Performance Comparison

Software algorithm was written in Matlab, and was running on a system with 1.7 GHz clock speed and 4 GB RAM. While the specific hardware architecture is designed to run with any clock speeds less than 250 MHz The bottleneck of 250 MHz is created by the FFT9 module in the hardware architecture. There is no direct general matrix to make the time comparison or time complexity of both the methods.

Since all stages of hardware design are pipelined, each stage process one sample in each clock cycle. And each stage is designed to execute the algorithm in parallel. For example consider the FFT9 module which uses Radix 2 burst I/O architecture on FPGA performs the algorithm in parallel, Mel filter Bank Energy Module also computes all 20 coefficients which belongs to each filter bank at the same time, while in software implementation each stage of algorithm runs sequentially. This difference obviously provide major computational gains in hardware implementation

In order to extract features from an audio sample the designed hardware architecture takes 324 more clock ticks than the number of samples in the audio signal. So for example, if an audio sample is 3.44 seconds long or if it has 55120 samples than our proposed hardware architecture will takes 55444 (55120+324) clocks to extract the MFCC feature from the audio sample.

So, if the designed hardware architecture is made to run on clock speed of 34.68Mhz, which is easily available in middle range FPGA boards, and also possible on Nexys DDR 4 board present with us. Then the time it takes to extract MFCC from 1000 such audio files which constitutes to 55120000 on our designed hardware architecture is 1.613128 seconds. While the same amount of processing of MFCC feature extraction process on Matlab implementation takes 21.86 seconds.

So clearly the designed hardware architecture provides around 13.55X speed improvements if the clock speed is chosen to be 34.68 MHz and software algorithm runs on the given system (1.7 Ghz and 4 GB RAM). Clearly, if we increase the clock speeds we can achieve even better computational gains.

Future Work

In order to improve the performance and efficiency of hardware architecture, we will be focusing on the following points as part of our future work:

- Preform more experiments for choosing better fixed point data paths to improve the accuracy of the designed hardware architecture.
- Look for methods to implement Logarithm function accurately on digital circuits.
- Incorporating sliding based STFT approach in hardware implementation of the MFCC feature extraction process.
- Developing hardware architecture for ΔMFCC and ΔΔMFCC in current proposed architecture.
- Making this hardware compatible for direct audio input (real time) via a microphone and pulse delta modulation of received signal on FPGA board.

Bibliography

[1] J. P. Campbell, “Speaker Recognition a tutorial”, *in Proceedings of IEEE*, 1997, vol. 85, pp. 1437-1462.

[2] T. Kinnunen and H. Li. (2010). An overview of text-independent speaker recognition: From features to supervectors. *Journal of Speech Communication,* vol. 52, no. 1, pp. 12-40, 2010.

[3] S. H. Mohammadi and A. Kain. (2017). An overview of voice conversion systems. *Journal of Speech Communication,* vol. 88, pp. 65-82.

[4] T. Kinnunen, Z. Wu, K. A. Lee, F. Sedlak, E. S. Chng and H. Li, “Vulnerability of speaker verification systems against voice conversion spoofing attacks: The case of telephone speech", presented at the Conf. of the IEEE ICASSP, Kyoto, Japan, March 25-30, 2012.

[5] J. Yamagishi, T. Kobayashi, Y. Nakano, K. Ogata andJ. Isogai. (2009). Analysis of speaker adaptation algorithms for HMM-based speech synthesis and a constrained SMAPLR adaptation algorithm. *Journal of IEEE Transactions on Audio, Speech and Language Processing*, vol. 17, no. 1, pp. 66-83.

[6] L. Wang, S. Nakagawa, Z. Zhang, Y. Yoshida, Y. Kawakami. (2017). Spoofing Speech Detection Using Modified Relative Phase Information. *Journal of IEEE Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 660-670.

[7] D. A. Reynolds. (1995). Speaker Identification and verification using Gaussian mixture speaker models. *Journal of Speech Communication*, vol. 17, no. 1-2, pp. 91-108.

[8] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam and D. Burger, “Dark silicon and the end of multicore scaling”, *in Proceedings of Computer Architecture (ISCA)*, 2011, vol. 39, no. 3, pp. 365-376.

[9] C. Li, D. Sengupta, F. S. Snigdha, W. Xu, J. Hu and S. S. Sapatnekar, “Special session: a quantifiable approach to approximate computing”, presented at the Conf. of IEEE CASES, Seoul, South Korea, October 15-20, 2017

[10] Z. Wu, E. S. Chng and H. Li, "Detecting converted speech and natural speech for anti-spoofing attack in speaker recognition", presented at the Conf. of International Speech Communication Association, INTERSPEECH, 2012.

[11] I. Saratxaga, I. Hernaez, D. Erro, E. Navas, and J. Sanchez. (2009). Simple representation of signal phase for harmonic speech models. *Journal of Electronic Letters*, vol. 45, no. 7, pp. 381-383.

[12] L. Wang, N. Kitaoka and S. Nakagawa. (2007). Robust distant speaker recognition based on position-dependent CMN by combining speaker-specific GMM with speaker-adapted HMM. *Journal of Speech Communication*, vol. 49, no. 6, pp. 501-513.

[13] Z. Wu, T. Virtanen, T. Kinnunen, E. S. Chng and H. Li, "Exemplar-based unit selection for voice conversion utilizing temporal information", in Proceedings of the ISCA, INTERSPEECH. Interspeech, 2013, pp. 3057-3061.

[14] J. Yamagishi, T. Kobayashi, Y. Nakano, K. Ogata and J. Isogai. (2009). Analysis of Speaker Adaptation Algorithms for HMM-based Speech Synthesis and a Constrained SMAPLR Adaptation Algorithm. *Journal of IEEE Audio, Speech and Language Processing*, vol.17, no.1, pp. 66-83.

[15] T. Toda, A. W. Black, and K. Tokuda. (2007). Voice conversion based on maximum likelihood estimation of spectral parameter trajectory. *Journal of IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 8, pp. 2222–2235.

[16] S. Furui. (1981). Cepstral analysis technique for automatic speaker verification. *Journal of IEEE Transactions on Audio, Speech and Signal Processing*, vol. 29, no. 2, pp. 254-272 .

[17] J. Deller, J. Hansen and J. Proakis. *Discrete-Time Processing of Speech Signals*, New York, NY, USA: Wiley-IEEE Press, 2000.

[18] M. Bahoura and H. Ezzaidi, “Hardware implementation of MFCC feature extraction for respiratory sounds analysis”, presented at the Conf. of WoSSPA, Algiers, Algeria, May 12-15, 2013.