BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

FPGA IMPLEMENTATION OF REAL TIME ACOUSTIC NOISE SUPPRESSION BY SPECTRAL SUBTRACTION TECHNIQUES

by

Saad Md. Jaglul Haider (0406045)

Upal Mahbub (0406048)

Tauhidur Rahman (0406067)

Supervised by

Dr. A. B. M. Harun-ur Rashid

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF BACHELOR OF SCIENCE
AT
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DHAKA,BANGLADESH
SEPTEMBER 2009

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF

ELECTRICAL AND ELECTRONIC ENGINEERING

The thesis entitled "**FPGA implementation of Real time Acoustic Noise Suppression by Spectral Subtraction Techniques**" by **Saad Md. Jaglul Haider, Upal Mahbub and Tauhidur Rahman** has been accepted in partial fulfillment of the requirement for the degree of **Bachelor of Science** on 30.09.09.

Supervisor: ______________________________

Dr. A. B. M. Harun-ur Rashid

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Date: **September 2009**

It is hereby declared that neither this thesis nor any part thereof has been submitted elsewhere for the award of any degree or diploma.

Author: ____________________________
Saad Md. Jaglul Haider

Author: ____________________________
Upal Mahbub

Author: ____________________________
Tauhidur Rahman

Supervisor: ____________________________
Dr. A. B. M. Harun-ur Rashid

To Our Parents

Table of Contents

Abstract

Speech Enhancement by suppressing ambient acoustic noise has been a challenging topic of research for many years. Among the plentiful approaches to retrieve enhance speech, the spectral subtraction methods are most widely used. The simplicity of these algorithms and comparatively low computational load made them the primary choice for real time applications.

This research paper concentrates on modification and implementation of two different spectral subtraction algorithms. The novel Weighted Moving Average(WMA) Method which is proposed in this thesis is a better variant of the traditional spectral subtraction method. As the simulation result shows, it can improve the Signal to Noise Ratio (SNR) from 2 to 3 decibel for several types of acoustic noise. It is also shown that it can work efficiently in a variable noise ambience.

The second algorithm is one of the most recent advancement in the field of speech enhancement The A Priori SNR Estimation Algorithm. In this research, the algorithm has been simulated, tested and implemented in Cyclone-II FPGA. A SNR improvement of about 2-3 db has been obtained by Average Segmented SNR Method for different Standard noisy signals. Although this improvement seems similar to WMA method, the enhanced speech is considerably more intelligible to human ear.

The implementations of speech enhancement algorithms are mostly done in microprocessors or DSP processors because of their flexibility. But the high cost per logic element

related to these processors makes them improper for large scale use. Thus, the prime concern of this research is an alternative FPGA (Field Programmable Gate Array) based implementation of the two speech enhancement algorithms which results less cost per logic element, while maintaining satisfactory quality. In our research several techniques have been used for minimizing the total resources used. From the experimental results, the WMA method uses 80% of the total 33216 logic elements of the Cyclone II FPGA chip, while A Priori SNR method consumes 70
% of the total logic.

Finally, as the FPGA implementation presented in this research works on streaming speech signals, it can be used in real life environments such as factories, bus terminals, Cellular Phones, or in outdoor conferences where a large number of people have gathered.

Acknowledgements

We would like to thank Professor A. B. M. Harun-ur Rashid, our supervisor, for his many suggestions and constant support during this research. We are also thankful to Professor Md. Kamrul Hasan for his guidance in the theoretical aspects of the research when we were in chaos and confusion.

Taufique Hasan Al Banna, an Alumni of Buet currently researching in University of Texas, Dallas, shared with us his knowledge of speech enhancement and provided many useful references and friendly encouragement.

Of course, we are grateful to our parents for their patience and love. Without them this work would never have come into existence (literally).

Finally, we wish to thank the following: Tarek, Partha, Md. Gaffar, Nasif and all the other friends who gave us love and support.

BUET, Bangladesh The Authors
September 2009

Chapter 1

Introduction

1.1 Background

The problem of enhancing speech degraded by uncorrelated additive noise, when only the noisy speech is available, has been widely studied in the past and it is still an active field of research. Noise, added acoustically to speech, can degrade the performance of digital voice processors used for speech recognition, compression and authentication. In many applications, such as mobile communication, efficient noise reduction techniques are needed, which require a low computational load. Noise suppression and speech enhancement techniques can improve Signal to Noise Ratio(SNR), lower the variance and emphasize the important features of the speech signal.

In 1979 Steven F. Boll proposed the spectral subtraction method for acoustic noise suppression [1]. Thereafter, many approaches have been investigated.

For example, C. Beaugeant and P. Scalart[2] proposed method that tried to enhance noise reduction system based on Wiener filtering in frequency domain by taking into account short time variations of speech.

In 1980, McAulay, Malpass suggested sppech enhancementusing Soft-Decision Noise

Suppression Filter.

In 1984, Ephraim and Malah proposed speech enhancement using Minimum Mean Square Error (or MMSE) estimation [3].

In 1996, Pascal SCALART and Jozue VIERA FILHO [4] confirmed that A Priori SNR estimation leads to the best subjective result to the problem of single microphone frequency domain speech enhancement in noisy environment. Later Improved A Priori SNR method was introduced by Israel Cohen[5], Md. Kamrul Hasan[6] at different times.

In 2004, Yuki DENVA [7]proposed speech recognition with wavelet spectral subtraction in real noisy environment.

1.2 Motivation

With the rapid development of microprocessors and DSP processors, acoustic noise suppression have been widely implemented with microprocessors or digital signal processors. These approaches provide flexibility and are suitable for many applications. However due to their very high cost per logic element microprocessors and digital signal processors are not suitable for widespread use.

In recent years, FPGA-based hardware implementation technology has been used for signal processing field successfully due to advantages of their programmable hard-wired feature, fast time to market and reusable IP (Intellectual Property) cores. Their cost per logic element is reasonable too. Besides, the FPGA based system can get a very high speed level, since it can carry out parallel processing by means of hardware mode.

Thus, one of the motivating factor of this research was the high capabilities of modern FPGA. At the beginning the focus was on the implementation of a relatively easier method, a modified form of Boll's algorithm. Thus, weighted moving average method, a novel but simple method, is introduced here and its implementation in FPGA is shown. Afterwards, the thesis focused on efficient implementation of a more complex A Priori SNR Estimation Method.

1.3 The Research Goal

FPGA implementation of a novel Spectral Subtraction Technique and another more improved Spectral Subtraction Technique is the primary goal of this research. Here, the complete FPGA implementation of Dynamic Moving Average Method and A Priori SNR Estimation Method have been shown. Both the implementation is real time and perfectly suitable for real life use for example mobile communication from a noisy place like factory or a busy road. This research also portrays the evolution of three spectral subtraction algorithms. From Boll's algorithm, which was quite primitive but simple, the thesis work delineates the evolution to a much more improved A Priori SNR algorithm. Although A Priori SNR is a more capable algorithm for expurgating the tainted speech but it is also more complex. One of the prime concern of this research is, thus, to implement a comparatively complex algorithm within a limited resource and achieving the best performance. For the sake of performance evaluation of these spectral subtraction techniques various algorithms has been used including a subjective SNR Estimation and a objective SNR estimation. A Graphical User Interface (GUI) has also been created for measuring the performance of the implementation.

Chapter 2

Spectral Subtraction Techniques

2.1 Basic Spectral Subtraction: Boll's Algorithm

In 1979 Boll introduced the basic idea of spectral subtraction algorithm. In this section, the method is briefly described with all necessary equations.

2.1.1 Subtractive Noise Suppression Analysis

By subtracting an estimate of the noise spectrum from the noisy speech spectrum, the estimator is obtained. Spectral information is required to describe the noise spectrum. From the signal, measured during non-speech activity, the noise spectrum is obtained.

Some assumptions were used in developing the analysis. This includes that the background noise is acoustically or digitally added to the speech and the background noise environment remains locally stationary. It is stationary to the degree that its spectral magnitude expected value just prior to speech activity equals its expected value during speech activity.

Suitably low-pass filtered and digitized speech is analyzed by windowing data from half-overlapped input data buffers. The magnitude spectra of the windowed data are calculated.The spectral noise bias,which is calculated during non-speech activity, is subtracted off. Following the subtraction, resulting negative amplitudes are zeroed out and after that Secondary residual noise suppression is applied. Then, a time waveform is recalculated from the modified magnitude which is then overlapped and added to the previous data to generate the output speech[1].

2.1.2 Additive Noise Model

Let, assume that a windowed noise signal n(k) has been added to a windowed speech signal s(k), with their sum denoted by x(k). Then

x(k)= s(k) + n(k)

The Fast Fourier Transform (FFT) block does the fourier transform on the noisy signal, yielding,

$$X(e^{j\omega}) = S(e^{j\omega}) + N(e^{j\omega})$$

where

$$X(e^{j\omega}) = \sum_{k=0}^{L-1} x(k)e^{-j\omega k}$$

$$x(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega k}d\omega$$

2.1.3 Spectral Subtraction Estimator

The spectral subtraction filter $H(e^{j\omega})$ is calculated by replacing the noise spectrum $N(e^{j\omega})$ with spectra which can be readily measured. The magnitude $\mid N(e^{j\omega}) \mid$ of $N(e^{j\omega})$ is replaced by its average value $\mu(e^{j\omega})$ taken during nonspeech activity, and the phase $\theta_N(e^{j\omega})$ of $N(e^{j\omega})$ is replaced by the phase $\theta_x(e^{j\omega})$ of $X(e^{j\omega})$. These substitution result in the spectral subtraction estimator $\hat{S}(e^{j\omega})$:

$$\hat{S}(e^{j\omega}) = [\mid X(e^{j\omega}) \mid -\mu(e^{j\omega})]e^{j\theta_x(e^{j\omega})}$$

or

$$\hat{S}(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega})$$

with

$$H(e^{j\omega}) = 1 - \frac{\mu(e^{j\omega})}{X(e^{j\omega})}$$

$$\psi(e^{j\omega}) = E[\mid N(e^{j\omega}) \mid]$$

2.1.4 Spectral Error

The spectral error $\epsilon(e^{j\omega}$ resulting from this estimator is given by,

$$\epsilon(e^{j\omega}) = \hat{S}(e^{j\omega}) - S(e^{j\omega}) = N(e^{j\omega}) - \psi(e^{j\omega})e^{j\theta_x}$$

To reduce the auditory effects of this spectral error, a number of simple modification are available which includes: 1) magnitude averaging; 2) half wave rectification; 3) residual noise reduction; and 4) additional signal attenuation during non-speech activity[1].

2.1.5 Magnitude Averaging

Since the spectral error equals the difference between the noise spectrum N and its mean ψ, local averaging of spectral magnitudes can be used to reduce the error. Replacing $\mid X(e^{j\omega} \mid$ with $\overline{\mid X(e^{j\omega} \mid}$ where

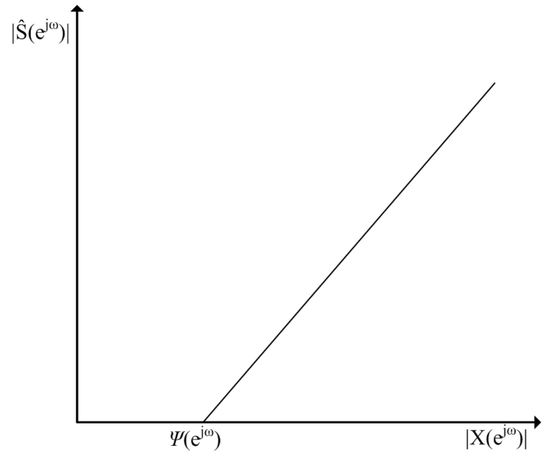

Figure 2.1: Input output relation between $X(e^{j\omega})$ and $\hat{S}(e^{j\omega})$

$$\overline{\mid X(e^{j\omega}) \mid} = \tfrac{1}{M} \sum_{i=0}^{M-1} \mid X_i(e^{j\omega}) \mid$$

$X_i(e^{j\omega})$= ith time windowed transform of x(k)

gives

$$S_A(e^{j\omega}) = [\overline{\mid X(e^{j\omega}) \mid} - \psi(e^{j\omega})]e^{j\theta_x(e^{j\omega})}$$

The rationale behind averaging is that the spectral error becomes approximately

$$\epsilon(e^{j\omega}) = S_A(e^{j\omega}) - S(e^{j\omega}) = \overline{\mid N \mid} - \psi$$

where, $\overline{\mid N(e^{j\omega}) \mid} = \frac{1}{M} \sum_{k=0}^{M-1} \mid N_i(e^{j\omega}) \mid$

Thus, the sample mean of $N_i(e^{j\omega})$ will converge to $\psi(e^{j\omega})$ as a longer average is taken.

2.1.6 Half-Wave Rectification

For each frequency ω where the noisy signal spectrum magnitude $\mid X(e^{j\omega}) \mid$ is less than the average noise spectrum magnitude $\psi(e^{j\omega})$, the output is set to zero. This modification can be simply implemented by half-wave rectifying $H(e^{j\omega})$. The estimator then becomes

$$\hat{S}(e^{j\omega}) = H_R(e^{j\omega})X(e^{j\omega})$$

where

$$H_R(e^{j\omega}) = \frac{H(e^{j\omega})+|H(e^{j\omega})|}{2}$$

The input-output relationship between $X(e^{j\omega})$ and $\hat{S}(e^{j\omega})$ at each frequency ω is shown in fig. 2.1. The input-output relationship of $X(e^{j\omega})$ and $\hat{S}(e^{j\omega})$, as shown in Fig. 2.1, depicts the result of Half-wave rectification at each frequency ω. The advantage of this rectification is that the noise floor is reduced by $\psi(e^{j\omega})$, any low variance coherent noise tones are essentially eliminated[1].

2.2 Voice Activity Detection Techniques

Due to the limitations of logic elements in the FPGA that was used (Cyclone II), an easy voice activity detection (VAD) technique was searched. After considering several individual algorithms the following two algorithms proved to be the best choice

2.2.1 Zero Crossing Rate

If successive samples have different algebraic signs, a zero crossing is said to occur in the context of discrete-time signals. The rate at which zero crossings occur is a simple measure of the frequency content of a signal. Zero-crossing rate is a measure of number of times in a given time interval/frame that the amplitude of the speech signals passes through a value of zero (Fig. 2.2, 2.3).

Since speech signals are broadband signals, interpretation of average zero-crossing rate is much less precise. Rough estimates of spectral properties can be obtained using a representation based on the short time average zero-crossing rate [9].

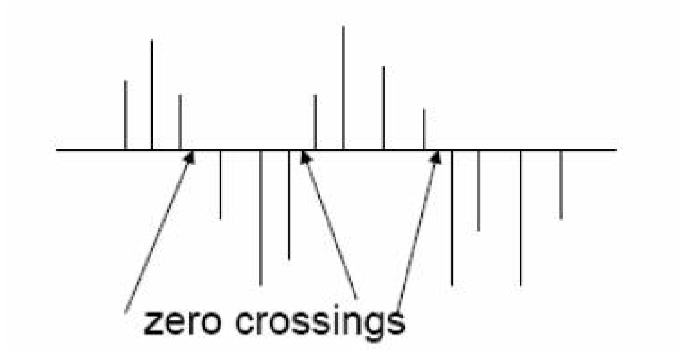

Figure 2.2: Definition of zero-crossings rate

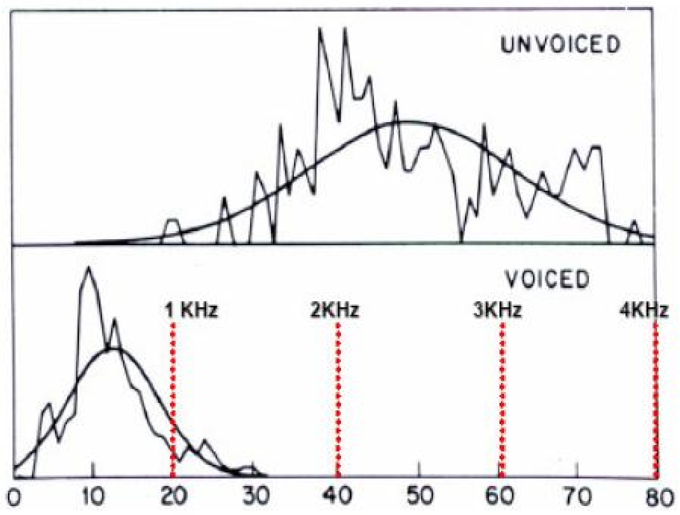

Figure 2.3: Distribution of zero-crossings for unvoiced and voiced speech

A definition for zero-crossings rate is:

$$Z_n = \sum_{-\infty}^{+\infty} |sgn[x(m)] - sgn[x(m-1)]| w(n-m) \text{ where}$$

$$sgn[x(n)] = \begin{cases} 1, & \text{for } x \geq 0 \\ -1, & \text{for } x < 0 \end{cases}$$

and

$$w(n) = \begin{cases} \frac{1}{2N}, & \text{for } 0 \leq n \leq N-1 \\ 0, & \text{otherwise} \end{cases}$$

It was suggested in the model for speech production that the energy of voiced speech is concentrated below about 3 kHz. This is because of the spectrum fall of introduced by the glottal wave. But for unvoiced speech, most of the energy is found at higher frequencies. There is a strong correlation between zero-crossing rate and energy distribution with frequency since high frequencies imply high zero crossing rates, and low frequencies imply low one. It should be a reasonable generalization that if the zero-crossing rate is high, the speech signal is unvoiced and if the rate is low, the speech signal is voiced[9].

2.2.2 Short Time Energy

Generally, the amplitude of the unvoiced speech segments is much lower than that of the voiced one and the amplitude of the speech signal varies with time. These amplitude variations are represented by the energy of the speech signal. Short time energy can be defined as:

$$E_n = \sum_{m=-\infty}^{\infty} [x(m)w(n-m)]^2$$

the nature of the short time energy representation is determined by the choice of the window. We used hamming window in our model which gives much greater attenuation outside the bandpass than the comparable rectangular window.

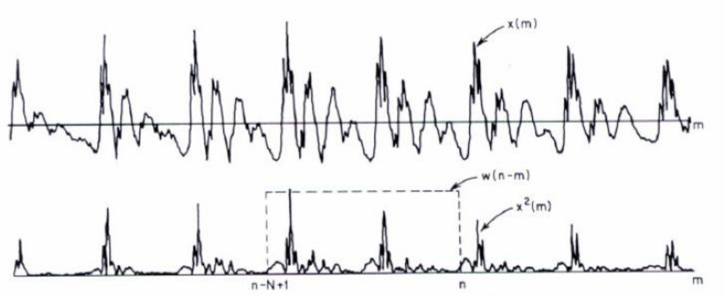

Figure 2.4: Computation of short time energy

$$h(n) = \begin{cases} 0.54 - 0.46cos(\frac{2\pi n}{N-1}), & \text{for } 0 \leq n \leq N-1 \\ 0, & \text{otherwise} \end{cases}$$

The attenuation of this window is independent of the window duration. Increasing the length, N, decreases the bandwidth, fig. 2.4. If N is too small, E_n will fluctuate very rapidly depending on the exact details of the waveform. If N is too large, E_n will change very slowly. Thus it will not adequately reflect the changing properties of the speech signal[9].

2.3 Advanced Approach: A Priori SNR Estimation Technique

2.3.1 Principle of Operation

Figure 2.5 shows the Flow Diagram of the A Priori SNR Technique. Let, an additive uncorrelated random noise n(t) gets added to a clean speech signal s(t). Thus, the total signal, denoted by y(t) is given by,

y(t)= s(t) + n(t)

Now, if at *n*th frame and *k*th frequency bin the noisy speech short-time spectral magnitude is $\mid Y_{n,k} \mid$ and the noise spectral magnitude is $\mid N_{n,k} \mid$, then an estimate of the clean speech short-time spectral magnitude can be obtained as,

$$\mid \widehat{S}_{n,k} \mid = G_{n,k}. \mid Y_{n,k} \mid$$

where the optimal gain function is

$$G_{n,k} = \frac{\Lambda(X_{n,k},q_{n,k})}{1+\Lambda(X_{n,k},q_{n,k})} G^0_{n,k}$$

and $G^0_{n,k}$ is the standard gain

$\Lambda(X_{n,k}, q_{n,k})$ is the generalized likelihood ratio taking into account the uncertainty of speech presence in the noisy observation defined by

$$\Lambda(X_{n,k}, q_{n,k}) = \mu_{n,k} \frac{p(X_{n,k}/H^1_{n,k})}{p(X_{n,k}/H^0_{n,k})}$$

with $\mu_{n,k} = (1 - q_{n,k})/q_{n,k}$, where $q_{n,k}$ is the probability of signal absance in the *k*th spectral componenet, and p(.) denotes a probability density function. $H^0_{n,k}$ and $H^1_{n,k}$ denote the two hypotheses of signal absence and presence respectively, in the *k*th spectral component. Here, however, $q_{n,k} = 0$, $\Lambda/(1 + \Lambda)$ equals unity, and $G_{n,k}$ turns out to be equal to the standard gain function $G^0_{n,k}$ when the speech signal is always present in the noisy observation[4].

The expression for standard gain,

$$G^0_{n,k} = \sqrt{max\{0, (1 - \frac{E\{|N_{n,k}|^2\}}{|Y_{n,k}|^2})\}} \text{ (1)}$$

here E{.} is the expectation operator. Numerous variations of the original subtraction rule has been proposed. Of them the class of power subtraction algorithms decidedly produce the most annoying musical noise[6]. In (1), the gain function takes on a value depending on a posteriori SNR defined by,

$$SNR_{post}(n, k) = \gamma_{n,k} = \frac{|Y_{n,k}|^2}{\sigma^2_d(n,k)}$$

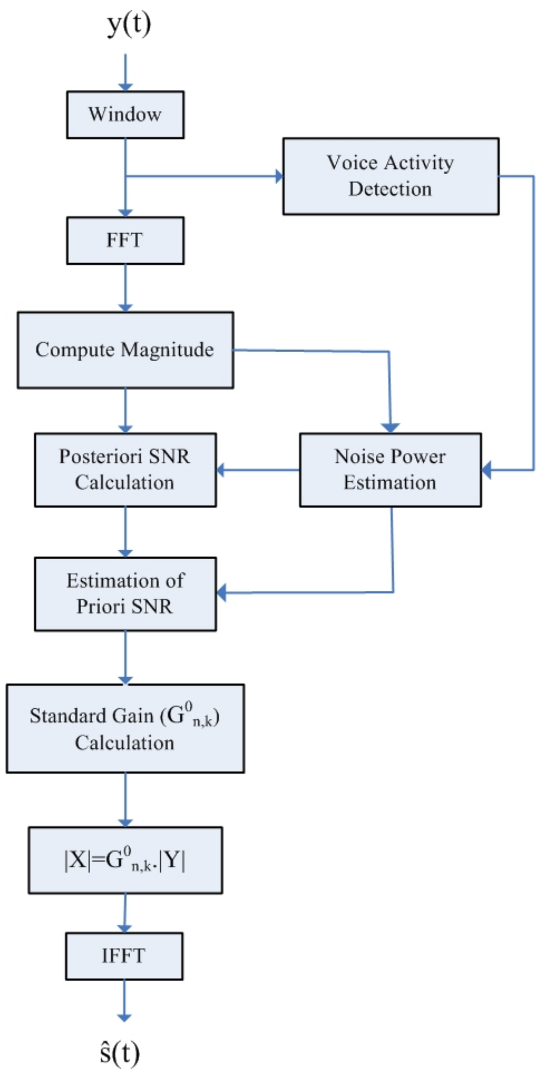

Figure 2.5: Flow diagram of A Priori SNR Technique

where,$\sigma_d^2(n,k) = E\{\mid N_{n,k} \mid^2\}$. On the other hand, a priori SNR is defined as,

$$SNR_{prior}(n,k) = \xi_{n,k} = \frac{E\{|S_{n,k}|^2\}}{\sigma_d^2(n,k)}$$

For a stationary or slowly varying noise, the noise estimate $[\widehat{\sigma}_d(n,k)]^2$ can be obtained from the speech pauses as

$$[\widehat{\sigma}_d(n,k)]^2 = \lambda_D[\widehat{\sigma}_d(n-1,k)]^2 + (1-\lambda_D \mid Y_{n,k} \mid^2$$

where $0.5 \leq \lambda_D \leq 0.9$.

Ephraim and Malah proposed a ”decision-directed” method for estimator $\widehat{\xi_{n,k}}$ of $\xi_{n,k}$ as

$$\widehat{\xi_{n,k}} = \alpha\frac{|\widehat{S}_{n-1,k}|^2}{\widehat{\sigma}_d^2(n-1,k)} + (1-\alpha)F[\gamma_{n,k} - 1]$$

where $\mid \widehat{S}_{n-1,k} \mid^2$ denotes the amplitude estimate of the *k*th speech spectral component, $\widehat{\sigma}_d^2(n-1,k)$ is the estimate of variance of the *k*th noise spectral component in the *(n-1)*th analysis frame, the opreator $F[.]$ denotes half wave rectification, and α denotes an averaging parameter. Considering the maximum likelihood estimate of the a priori SNR, we have $\xi_{n,k} = E\{\gamma_{n,k-1}\}$. Accordingly the gain function of the original subtraction rule can be modified by,

$$G_{n,k}^{PE} = \sqrt{\frac{\widehat{\xi}_{n,k}}{(1+\widehat{\xi}_{n,k})}}$$

and an estimate of the short time clean speech spectral magnitude is obtained as,

$$\mid \widehat{S}_{n,k} \mid = G_{n,k}^{PE}. \mid Y_{n,k} \mid$$

Chapter 3

Introducing Weighted Moving Average Method

3.1 Motivation: Limitations of Boll's Algorithm

Boll's classic spectral subtraction algorithm assumes noise level to be stationary. However, in real life situations, the noise level can change its value with changes in surrounding environment. For example the value of Signal to Noise Ratio (SNR) may decrease abruptly when a nearby car blows its horn, while someone is talking over Mobile Phone. The classic spectral subtraction algorithm doesn't take these situations into account. Thus a modification is needed so that the algorithm can follow the change in noise level and make better estimations. This motivates the introduction of Weighted Moving Average Method.

3.2 Basics of Weighted Moving Average(WMA) Method

The Weighted Moving Average Method assumes that, 1) Noise level can be non-stationary. 2) Noise shows up as an additive component in the energy domain. Hence, the additive noise component can be subtracted from the noisy speech energy to estimate the clean

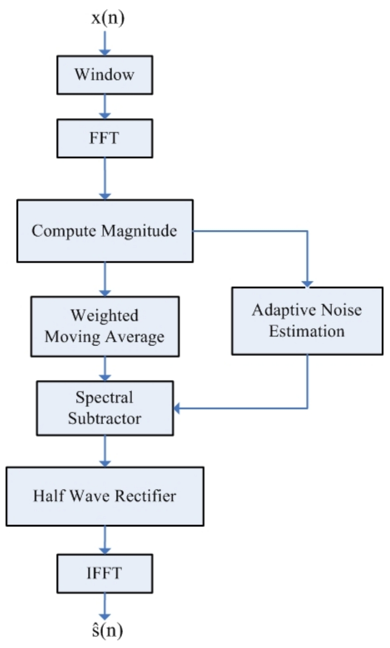

Figure 3.1: Flow diagram for Weighted Moving Average Method

speech energy[10] , 3) The phase distortion can not be perceived by the human ear[11].

The speech, suitably low-pass filtered and digitized, is analyzed by windowing data from the input data buffers. The magnitude spectra of the windowed data are calculated and the spectral noise bias calculated by Weighted Moving Average method is subtracted off. Resulting negative amplitudes are then zeroed out. A time waveform is recalculated from the modified magnitude. Fig. 3.1 shows the Flow Diagram of WMA method.

3.2.1 WMA Operations

To reduce spectral error (arising from the difference of actual noise and estimated noise) local averaging of the spectral magnitudes is done. $\mid X(e^{j\omega}) \mid$ is replaced by its weighted moving average $\overline{\mid X(e^{j\omega}) \mid}$ where

$$\overline{\mid X_i(e^{j\omega}) \mid} = \frac{\sum_{i=0}^{M-1} \alpha_i X_i(e^{j\omega})}{\sum_{i=0}^{M-1} \alpha_i}$$

$X_i(e^{j\omega})$ = *i*-th time-windowed transform of x(k) and α_i = *i*-th time weighting factor, where $\alpha_i \geq \alpha_j$ when $i > j$.

Fig. 3 shows the process of calculating the weighted moving average. Weighted averaging is done in order to give emphasis on the most recent data set, for which the estimation is being done, thus it can be also called Dynamic Moving Average.

3.2.2 Noise Estimation and Spectral Subtraction

The magnitude of the noise for the audio stream is estimated by taking average of a data set during non-speech activity, and it is represented by $\psi(e^{j\omega})$ where

$$\psi(e^{j\omega}) = E\{\mid N(e^{j\omega}) \mid\}$$

The phase $\theta_N(e^{j\omega}$ of $N(e^{j\omega})$ is replaced by the phase $\theta_x(e^{j\omega})$ of $X(e^{j\omega})$. These result in the spectral subtractor estimator

$$\hat{S}(e^{j\omega}) = [\mid X(e^{j\omega}) \mid - \psi(e^{j\omega})] e^{j\theta_x(e^{j\omega})}$$

However, spectral error $\epsilon(e^{j\omega})$ results form this estimator, which is given by,

$$\epsilon(e^{j\omega}) = \hat{S}(e^{j\omega}) - S(e^{j\omega}) = N(e^{j\omega}) - \psi(e^{j\omega}) e^{j\theta_x}$$

To reduce the error magnitude averaging and half wave rectification techniques are implemented [3]. Averaging the magnitude gives,

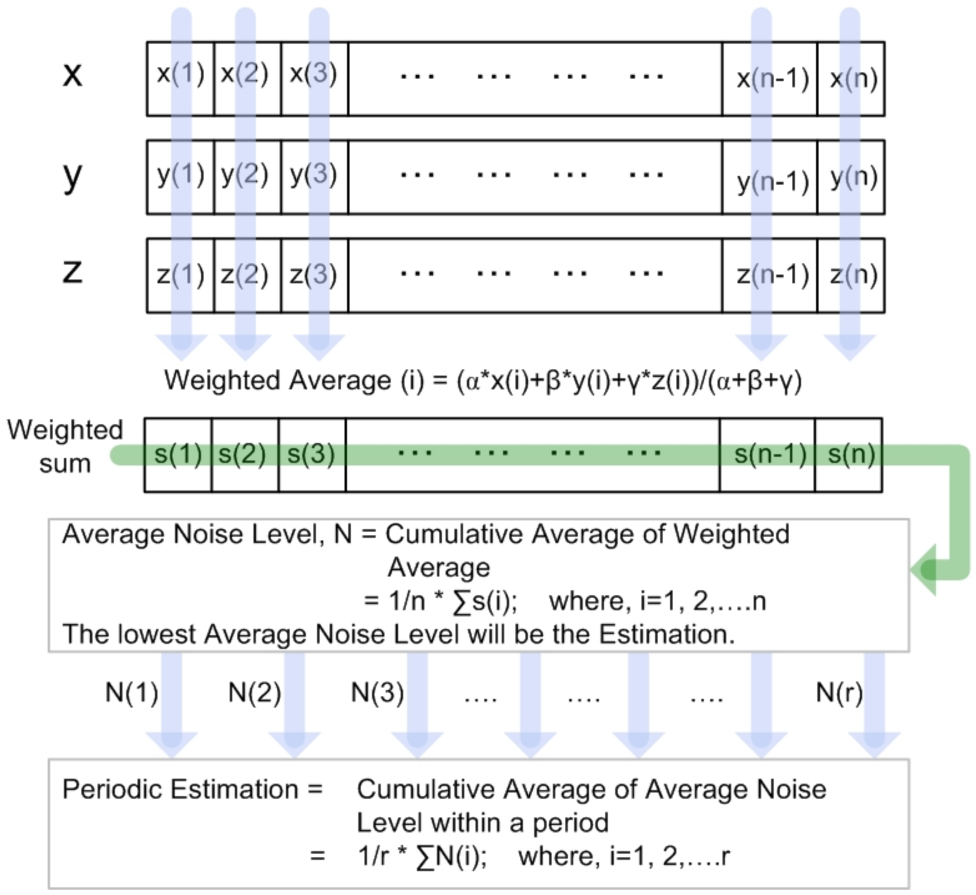
x
y
z
x(1) x(2) x(3) … x(n-1) x(n)
y(1) y(2) y(3) … y(n-1) y(n)
z(1) z(2) z(3) … z(n-1) z(n)
Weighted Average (i) = (α*x(i)+β*y(i)+γ*z(i))/(α+β+γ)
Weighted sum
s(1) s(2) s(3) … s(n-1) s(n)
Average Noise Level, N = Cumulative Average of Weighted Average
= 1/n * ∑s(i); where, i=1, 2,….n
The lowest Average Noise Level will be the Estimation.
N(1) N(2) N(3) …. …. …. N(r)
Periodic Estimation = Cumulative Average of Average Noise Level within a period
= 1/r * ∑N(i); where, i=1, 2,….r

Figure 3.2: Dynamic Moving Average Method

$$S_M(e^{j\omega}) = [\overline{\mid X(e^{j\omega}) \mid} - \psi(e^{j\omega})]e^{j\theta_x(e^{j\omega})}$$

The rationale behind averaging is that the spectral error becomes approximately

$$\epsilon(e^{j\omega}) = S_M(e^{j\omega}) - S(e^{j\omega}) = \overline{\mid N \mid} - \psi$$

where, $\overline{\mid N(e^{j\omega}) \mid} = \frac{1}{n}\sum_{k=0}^{n-1} \mid N(e^{j2\pi fk}) \mid$

Thus, the sample mean of $N_i(e^{j\omega})$ will converge to $\psi(e^{j\omega})$ as a longer average is taken [3].

Fig. 3.2 shows the process of calculating the noise estimation from moving average. It is assumed, that during non-speech activity, only the noise spectra is obtained. The average noise obtained at the very beginning of data acquisition, when the speech has not started, is thus supposed to be the estimated noise. Still, there remains the possibility that the sound will start as well at the very beginning of the process. In that case, average of the first data set will be considered as the noise estimation. For the next data sets, the latest average signal value will be compared with the immediate previous average and if smaller, then the latest average will be the estimation of noise.

To detect the variation in noise level, periodic upgrade of the noise estimation is done. For a given number of data sets (r), all the average noise values are stored and then their mean is calculated. This mean value is set as the new estimation when a period is complete. Thus, periodic estimation,

$$\overline{\mid N_p(e^{j\omega}) \mid} = \frac{1}{r}\sum_{i=0}^{r-1} \overline{\mid N_i(e^{j\omega}) \mid}$$

So, the estimator can respond to any change in noise level within each r data sets and upgrade itself.

Chapter 4

Field Programmable Gate Array (FPGA)

4.1 Basic Idea of FPGA

4.1.1 Architecture

A field-programmable gate array (FPGA) is a semiconductor device that can be configured by the customer or designer after manufacturing. Hardware Description Language (HDL) is used to implement or program a logic circuit diagram or a source code in FPGAs[12].

FPGAs contain arrays of configurable logic blocks (CLBs), I/O pads, routine channels and a hierarchy of reconfigurable interconnections that allow the blocks to be wired together.

Logic blocks are actually programmable logic components which can perform complex or simple combinational functions (like AND or XOR) when programmed. These logic blocks may be simple flip flop or complete blocks of memory. A classic FPGA logic block comprises 4-input lookup tables (LUT), and a flip-flop. 6 LUTs can be used for high performance parts. Generally all routing channels have the same number of wires (width).

Multiple I/O pads may fit into the height of one row or the width of one column in the array.

An application circuit must be mapped into an FPGA with adequate resources. While the number of CLBs and I/Os required is easily determined from the design, the number of routine tracks needed may vary considerably even among designs with the same amount of logic. Since unused routing tracks increase the cost without providing any benefit, FPGA manufacturer try to provide just enough tracks so that most designs that will fir in term of LUTs and I/Os can be routed.

There is only one output that can be either the registered or unregistered LUT output. The logic block has four inputs for the LUT and a clock input. Since clock signals and other high-fanout signals are normally routed via special purpose dedicated routing networks in commercial FPGAs, they and other signals are separately managed.

4.1.2 FPGA Design and Programming

To define the behaviour of the FPGA, the user provides a hardware description language (HDL) or a schematic design. HDL form is easier to work with when handling large structures as it is possible to just specify them numerically rather than having to draw every piece by hand. On the other hand, schematic entry can allow for easier visualization of a design. Using an electronic design automation tool, a technology-mapped netlist is generated which can be then fitted to the actual FPGA architecture using a process called ”place-and-route”. The user have to validate the map, place and route results via analysis, simulation, and other verification methodologies. Once the process is completed, the binary file is generated and is used to reconfigure the FPGA

For the purpose of going from schematic/HDL source files to actual configuration, the source files are fed to a software suite where different steps will produce a file. The file is then transferred to the FPGA via a serial interface (JTAG) or to an external memory device like an EEPROM.

To simplify the the design of complex systems in FPGAs, there exist libraries of predefined complex functions and circuits that have been tested and optimized to speed up the design process. These predefined circuits are called ”IP cores” . In a typical design flow, an FPGA application developer will simulate the design at multiple stages throughout the design process. Initially the RTL description in VHDL or Verilog is simulated by creating test benches to simulate the system and observe results. Then, after the synthesis engine mapped he design to a netlist, it is then translated to a gate level description where simulation is repeated to confirm the synthesis proceeded without errors. Finally the design is laid out in the FPGA at which point propagation delays can be added and the simulation run against with these values back-annotated onto the netlist.

4.1.3 Application of FPGA

Application of FPGA s include digital signal processing, software-defined radio, aerospace and defense system, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation and a growing range of other areas.

With the introduction of dedicated multipliers into FPGA architectures, applications, which had traditionally been the reserve of DSPs, began to incorporate FPGAs instead. FPGA s especially find applications in any area or algorithm that can make use of massive parallelism offered by their architecture.

The adoption of FPGAs in high performance computing is currently limited by the complexity of FPGA design compared to conventional software and the extremely long

Figure 4.1: DE2 Package Contents

turn-around times of current design tools, where 4-8 hours wait is necessary after even minor changes to the source code.

4.2 Overall configurations of DE2 Development Board

The following hardware is provided on the DE2 board[13]:

- Altera Cyclone II 2C35 FPGA device
- Altera Serial Configuration device - EPCS16
- USB Blaster (on board) for programming and user API control
- both JTAG and Active Serial (AS) programming modes are supported

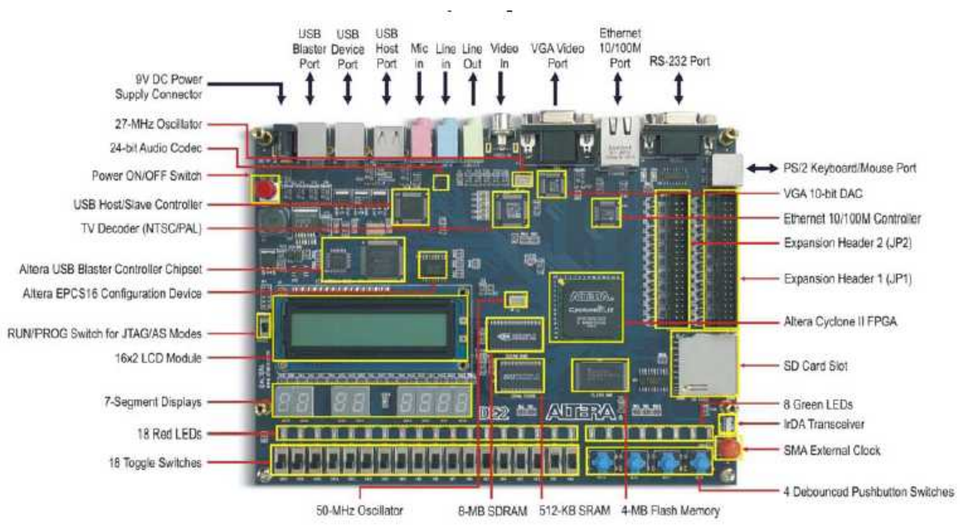

Figure 4.2: Layout of DE2 Board

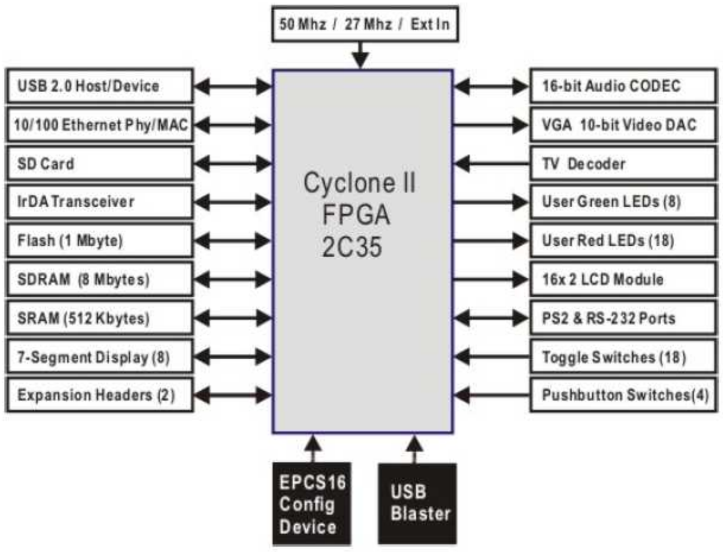

Figure 4.3: Block Diagram of the DE2 Board

- 512-Kbyte SRAM 8-Mbyte SDRAM DE2 User Manual 5
- 4-Mbyte Flash memory (1 Mbyte on some boards)
- SD Card socket
- 4 pushbutton switches
- 18 toggle switches
- 18 red user LEDs
- 9 green user LEDs
- 50-MHz oscillator and 27-MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL) and TV-in connector
- 10/100 Ethernet Controller with a connector
- USB Host/Slave Controller with USB type A and type B connectors
- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- IrDA transceiver
- Two 40-pin Expansion Headers with diode protection

In addition to these hardware features, the DE2 board has software support for standard I/O interfaces and a control panel facility for accessing various components.

4.3 Configuration of Individual Components Used in the Research

4.3.1 Cyclone II 2C35 FPGA

- 33,216 LEs
- 105 M4K RAM blocks
- 483,840 total RAM bits
- 35 embedded multipliers
- 4 PLLs
- 475 user I/O pins
- FineLine BGA 672-pin package

4.3.2 Audio CODEC

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

4.3.3 Clock inputs

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

4.3.4 Pushbutton switches

- 4 pushbutton switches
- Debounced by a Schmitt trigger circuit
- Normally high; generates one active-low pulse

4.3.5 Toggle switches

- 18 toggle switches for user inputs
- A switch causes logic 0 when in the DOWN and logic 1 when in the UP position

4.3.6 Serial Configuration device and USB Blaster circuit

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

4.3.7 SRAM

- 512-Kbyte Static RAM memory chip
- Organized as 256K x 16 bits
- Accessible as memory for the Nios II processor and by the DE2 Control Panel

4.3.8 SDRAM

- 8-Mbyte Single Data Rate Synchronous Dynamic RAM memory chip
- Organized as 1M x 16 bits x 4 banks
- Accessible as memory for the Nios II processor and by the DE2 Control Panel

4.3.9 Flash memory

- 4-Mbyte NOR Flash memory (1 Mbyte on some boards)
- 8-bit data bus
- Accessible as memory for the Nios II processor and by the DE2 Control Panel

Chapter 5

Simulation and Hardware Implementation of Moving Average Method

5.1 Simulation Results

The Speech Enhancement algorithm was at first simulated using Matlab (ver. 2007a). A simple .wav file containing the speech ”Hello, how are you”(Fig. 5.1(a)) was contaminated with additive white Gaussian noise of different levels ranging from 5 db to -5 db (Fig. 5.1(b)). Then their FFT is calculated (Fig. 5.2). In Fig. 5.1(c) the final noise reduced sound signal is shown while Fig. 5.3 shows the average noise magnitude and the estimated noise for different data sets. The data set length was 128, i.e. the FFT/IFFT datalength was 128. The period length was 10 data sets. As we can see from Fig. 5.3 the system tries to detect any change in noise level after each 10 estimations. If the noise level changes then it sets the estimation to the new noise level and thus it detects the change.

The algorithm has been tested for different noisy environment. The improvement in SNR is summarized in Table 5.1. However in some severe cases reduction in intelligibility is observed.

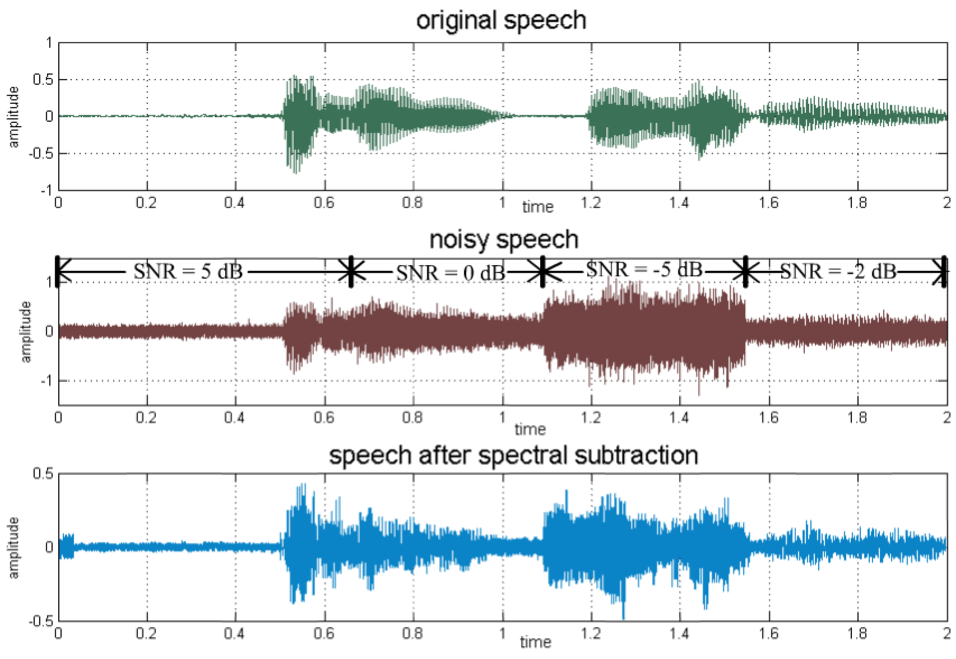

Figure 5.1: Time domain view of (a) Original Speech (b) Noisy Speech (c) Speech after Spectral Subtraction

Table 5.1: Improvement in SNR for different Types of Acoustic Noise

Noise	Input SNR (in dB)	Output SNR (in dB)
WGN	10.4	20.4
Thunder	7.86	10.25
Boeing	4.7	10.86
Applause	1.8	7.1
Engine	0.82	4.32

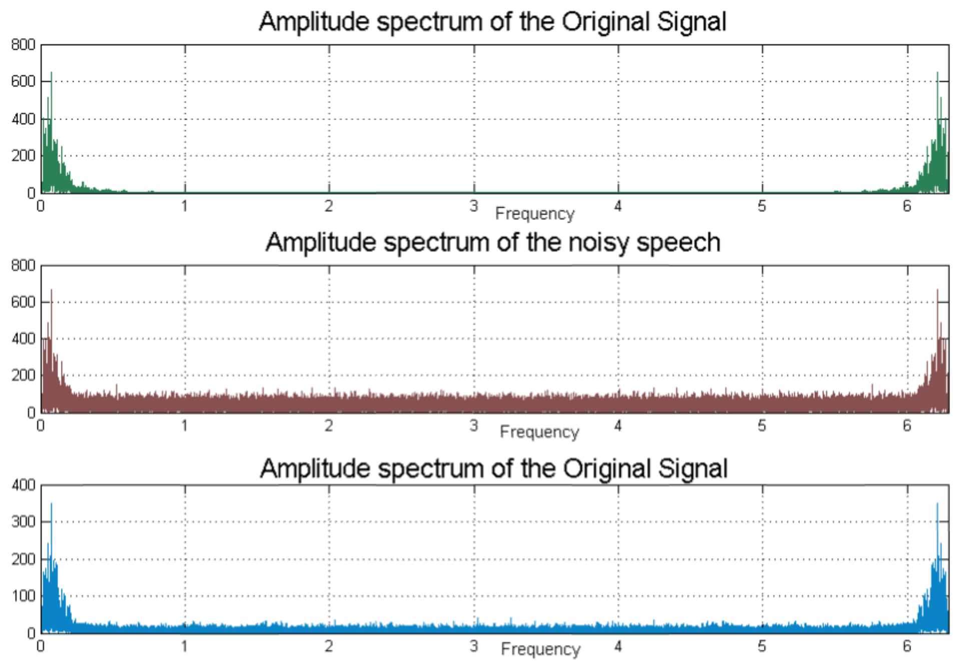

Figure 5.2: Amplitude Spectra of (a) Original Speech (b) Noisy Speech (c) Speech after Spectral Subtraction

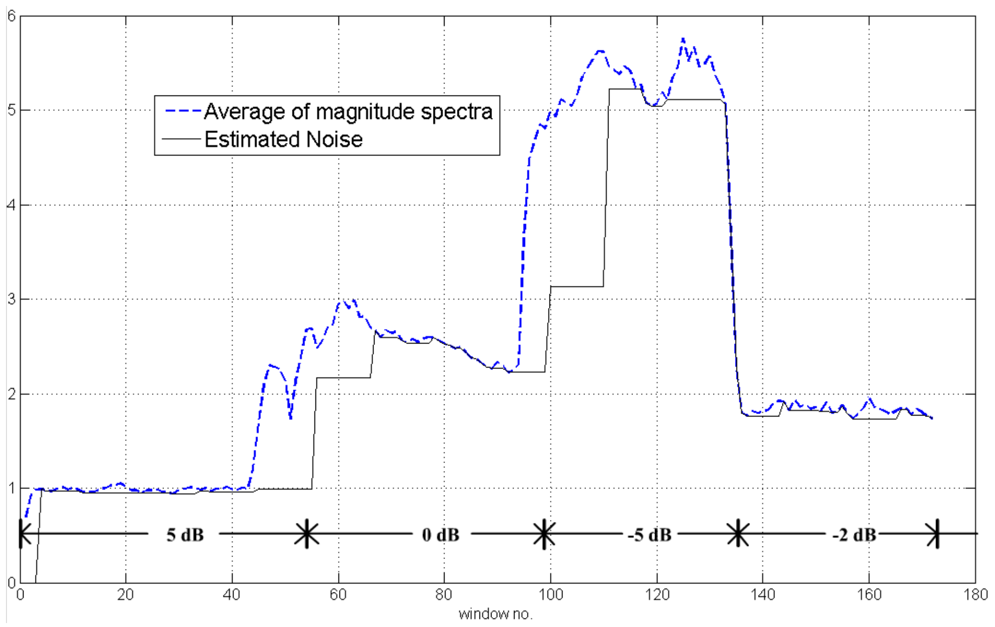

Figure 5.3: Noise Estimations

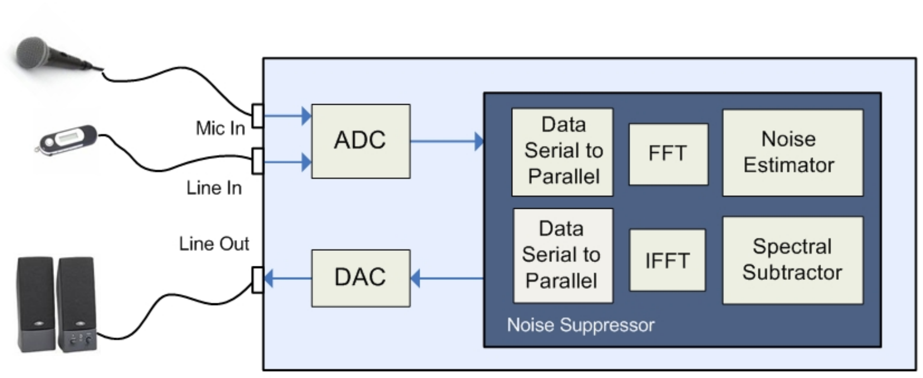

Figure 5.4: Basic Block Diagram

5.2 FPGA Implementation Technique for Dynamic Moving Average Method

5.2.1 Overview

A basic block diagram of the Speech Enhancement system is given in Fig. 5.4. The main modules and block will be described in this section.

The ADC, DAC, Digital Filter and Digital Controller blocks are actually embedded in the single chip audio codec, while the FFT and IFFT, spectral subtraction and noise estimations are done in the single chip FPGA. Memory blocks were also used in the process for storing data.

Of them, the ADC, DAC, Digital Filter and Digital Controller blocks are actually embedded in the single chip audio codec. They were configured to take analog sound input, digitize them (A/D conversion) to 16 bit data and then after filtering, provide the data as serial stream for modification to the FPGA chip. The single chip FPGA holds all the modifying blocks. Of them, the FFT and IFFT blocks convert the data from time to frequency domain and vice versa. The Spectral subtractor block has an embedded moving average block and with the help of the noise estimator it does necessary modifications on the magnitude spectra of the input sound signal. The modified data is then serially supplied to the codec and finally D/A conversion provides the analog sound output which can be heard from the speaker.

In the signal modification portion, first the serial data from the codec is converted to parallel 16 bit data (Data Serial to Parallel block). After windowing a certain time period, Fast Fourier Transform (FFT block) is done on the data to obtain the frequency response. This gives the real and imaginary parts as outputs which are then used to estimate noise as well as data magnitude and the noise is subtracted from the magnitudes of the concerned data set. The noise estimator follows a dynamic algorithm and can upgrade its outputs according to new data set values. After suppressing noise and converting the output to real and imaginary values, the signal is fed to IFFT (Inverse FFT) block and finally the parallel data is converted to serial and sent to audio codec. D/A conversion is done inside the codec and through line out of the board the purified sound can be heard using a headphone or speaker.

Fig. 5.5 shows a more detailed view of the overall system.

5.2.2 Audio Codec

Like most cases, the first step of implementing an FPGA system, for speech enhancement is to modify the audio codec. The single chip audio codec has its internal ADC, DAC, Digital filters, volume controller etc. There are two line in and line out paths(for Left and Right channels) and they are both connected with separate ADCs and DACs. There are also ADC (decimation) filter after ADC which ensures the correct sampling frequency output

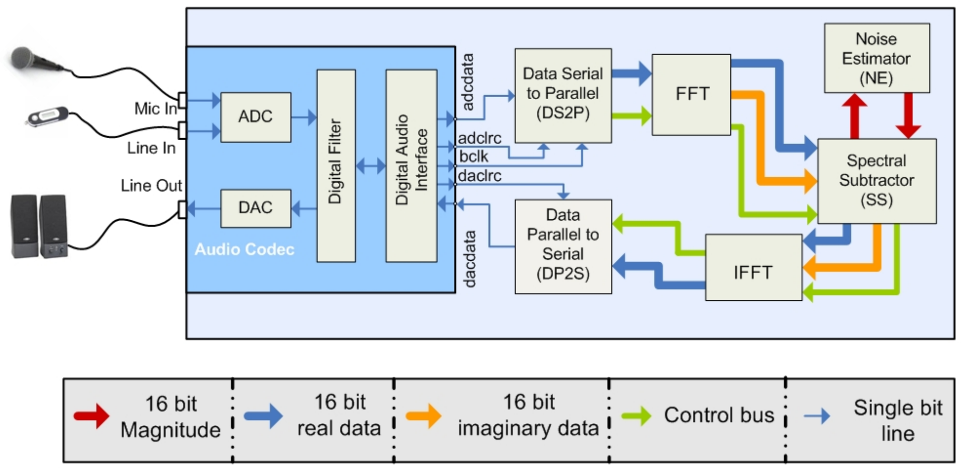

Figure 5.5: System Block Diagram

and DAC filter before DAC that converts the digital audio data at the specified sample rate for processing by the analogue DAC [14].

There are also ADC filter after ADC and DAC filter before DAC. The ADC filter is a decimation filter, which converts the raw over-sampled multi-bit data from the ADC to the correct sampling frequency to be output on the digital interface. On the other hand, the DAC filter converts the digital audio data from the digital interface at the specified sample rate to multi-bit over-sampled data for processing by the analogue DAC [14].

The audio codec outputs a serial datastream and some controlling signals. It also takes serial data stream for DAC.

The output of the ADC filter is converted to serial data streams within the codec and then provided for modifications. The modified data is again given to the audio codec as a serial data stream and then the DAC converts the binary data into sounds (output through line out). The codec also outputs some controlling signals and clocks, which help to modify its outputs.

5.2.3 Data Serial to Parallel Conversion Block

The main operation of this module is to convert the serial data obtained from the codec to parallel data (commonly 16 bits). This block also maintains the synchronization of data acquisition for the two different channels (left and right channel - but only one channel is used) from a single serial data stream. Moreover, it generates some control signals for the next levels.

5.2.4 Fast Fourier Transform Block

The signal from the serial to parallel conversion block is fed to the FFT block to produce the frequency domain response. Here Discrete Fourier Transform (DFT) is done on the data using Fast Fourier Transform (FFT) algorithm. Let $x_0, x_1, ...x_{N-1}$ be complex numbers. The DFT is done by the formula

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x_n(W_N^{Nk}) \text{ k=0,1,2.... N-1}$$

Here, W is the twiddle factor where,

$$W_N = e^{-j\frac{2\pi}{N}} = cos\frac{2\pi}{N} + j(-sin\frac{2\pi}{N})$$

The idea behind FFT is to break up the original N point sample into two (N/2) sequences. A radix-4 decomposition needs only trivial multiplications. To maintain a high signal-to-noise ratio throughout the transform computation, a block-floating-point architecture is used[15].

which is a trade-off point between fixed-point and full-floating point architectures

This results in the highest throughput decomposition, while requiring non-trivial complex multiplications in the post-butterfly twiddle-factor rotations only.

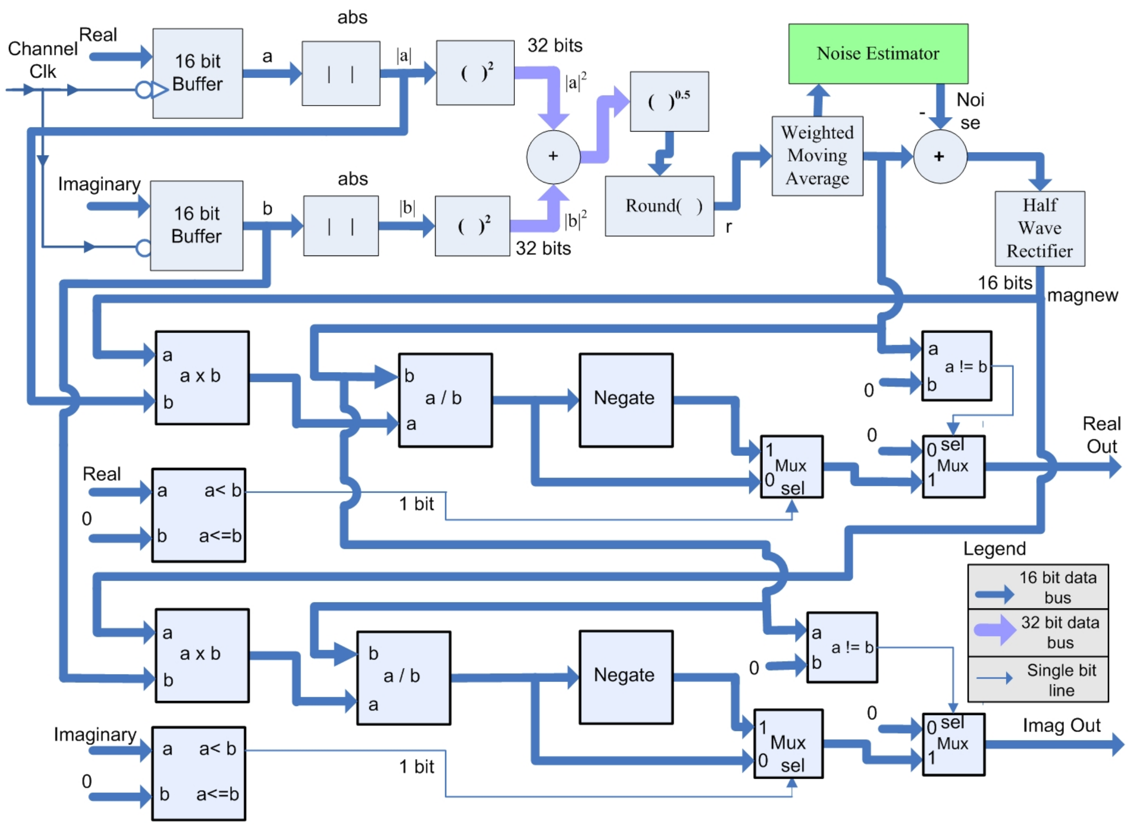

Figure 5.6: Spectral Subtractor block in detail

5.2.5 Spectral Subtractor Block

Fig. 5.6 shows the internal construction of the Spectral Subtractor block. First, it takes the real(a) and imaginary(b) values of FFT output, saves them in a buffer and finds out the magnitude(r) of the complex quantity by the following equation:

$$r = \sqrt{a^2 + b^2} \text{ ; where a+ib is the complex value.}$$

The moving average block then calculates the weighted moving average of the magnitudes. This blocks uses the memory to store the values of three consecutive data sets (including the latest data set), calculates the weighted average and while giving it as a output, it also upgrades its stored values with the latest average. This block follows the equation

$$avg(i) = \frac{\alpha x(i)+\beta y(i)+\gamma z(i)}{\alpha+\beta+\gamma}$$

where $x(1), x(2)....x(n)$, $y(1), y(2).....y(n)$ and $z(1), z(2)....z(n)$ are three consecutive data sets and x(i) is the latest set. After calculating avg(i) for the current set, the block will copy all the elements of y(i) to z(i) and x(i) to y(i), then x(i) will hold the values of the new set of data. The calculation of average will go so on. α, β and γ are weighting factors. The latest data is given the highest weight. Here, $\alpha = 3$, $\beta = 2$ and $\gamma = 1$ was considered.

The noise estimator takes the averages as inputs and gives estimated noise as output. The estimated noise value is then subtracted from the magnitude and the output is half wave rectified, i.e. all the values smaller then zero are made zero. What obtained is the new magnitude *magnew*. The next job is to convert the new magnitude into real and imaginary values again keeping the phase angle same as before. If the previous angle was θ then

$$\cos\theta = \frac{a}{r}; \sin\theta = \frac{b}{r}$$

Thus, the new real and imaginary values will be

$$\mid real - out \mid = magnew * \cos\theta = \frac{magnew*a}{r}$$
$$\mid imag - out \mid = magnew * \sin\theta = \frac{magnew*b}{r}$$

As the initial magnitude (r) is always positive, the signs are dependent on the values of the real input(a) and imaginary input(b). The inputs are compared with zero, and if they are negative, i.e. smaller then zero, then real-out and imag-out are negated respectively.

5.2.6 Noise Estimator Block

This module is responsible for continuous noise estimation from the frequency spectrum of the channel data during non-speech activity. It is directly connected to the RAM in which all the magnitudes of the latest windowed spectrum is stored. After each N frequency response is saved it takes the mean of N magnitudes. If the mean is smaller then the previous estimation (from the previous N windows) then the estimate is updated. The rationale behind this is the spectrum of the newly windowed channel data is a better approximation of non-speech activity or channel noise. Also, periodically the noise level is approximated by averaging the average noise levels of some previous data windows. This has a two fold effect: (1) It detects the change in noise level (if any) (2) the larger the number of noise levels taken for averaging, the closer the estimation to the actual level[1].

5.2.7 Inverse Fast Fourier Transform Block

The IFFT block converts the frequency domain values into time domain signal again. It does the Inverse Discrete Fourier Transform(IDFT), which can be obtained simply by negation of the imaginary part of the twiddle factor W. The other parts of the algorithm are same as FFT.

5.2.8 Data Parallel to Serial Conversion

This block takes the parallel signal data of the two channels as input and outputs their serial combination. This requires fine synchronization which is maintained by using control signals generated from the previous blocks. The output is then fed to the Audio Codec through the digital control interface.

Table 5.2: Resource Usage

Total Logic Element	26,712 / 33,216 (80%)
Combinational Functions	25,146 / 33,216 (76%)
Dedicated Logic Registers	15,768 / 33,216 (47%)
Total Registers	15,768
Total Pins	18 / 475 (4%)
Total Memory Bits	40,108 / 483,840 (8%)
Embedded Multiplier	48 / 70 (69%)

Finally, the signal passes through the DAC filter and DAC and output sound signal is obtained from the Lineout speaker.

5.3 Experimental Results

The whole process was implemented in a single chip FPGA (Cyclone II EP2C35F672C6). Also a single chip Audio codec (WM8731) was used for data in/outs. The coding was done in verilog using Quartus II (ver. 7.2 sp3). The codec was set in 16 bit mode and the FFT/IFFT blocks were set for a data length of 128. The output was satisfactory when input volume was low. This occurs, because of the 16 bit data resolution constraint. Figure 5.7 shows the timing diagram of and control signal flow from module to module. As can be seen from the figure, the delay between input and output data stream is about 410μsec. Table 5.2 shows the compilation flow summary of the code. It can be seen from the table that the logic usage is 80% of the total logic elements available. The free logic elements

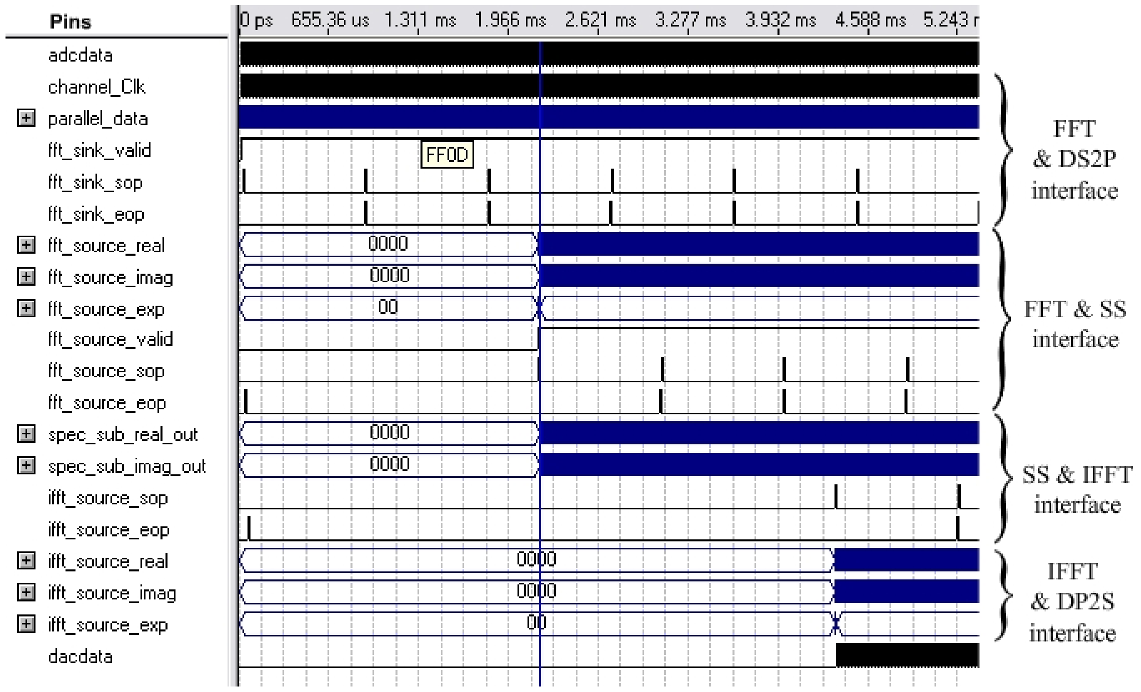

Figure 5.7: Timing diagram of data flow

thus provides scopes for doing some additional calculations (if needed). The combinational functions(76% used), Embedded Multiplier(69% used) and Memory Bits(only 8% used)also provide space for expansion of the system.

Chapter 6

Simulation and Hardware Implementation of an advanced approach: A Priori SNR Method

6.1 Simulation Results

In the simulation phase various noisy speech signals has been tested using Matlab. For testing purpose we have used noise from standard noise database named TIMIT. For a successful evaluation of the algorithm implemented it is incumbent to test it with a motley of acoustic noise. In table 6.1 it can be seen that we have used acoustic noise of car, train, street and white noise. and the table shows how our algorithm improves the segmented SNR of the noisy speech signal.

In fig. 6.1 you can see the original speech signal which was used. Here in this speech signal a male voice has been used as the original speech signal.

For testing whether our implemented algorithm can keep track with any possible change in SNR of noisy speech signal a speech signal with variable SNR has been taken. In fig. 6.2 there we have the same speech signal tainted with consecutively 10, 5, 0 db noise. That is, the first strip of the original speech is polluted by 10 db white noise, the second strip

Table 6.1: Improvement in SNR for different Types of Acoustic Noise[21]

Noise	Input SNR (in dB)	Output SNR (in dB)
car	9.2914	12.1181
train	1.0581	4.8687
train	14.2581	16.6920
street	4.3398	7.4295
street	9.6501	10.3276
white	0.8887	6.5067
white	5.3390	10.4005
white	10.2480	14.1132

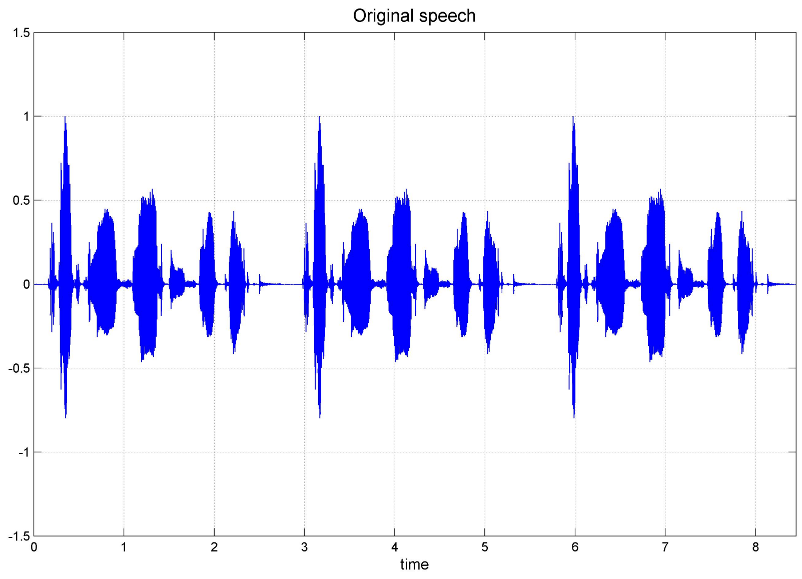

Figure 6.1: Original Speech Signal

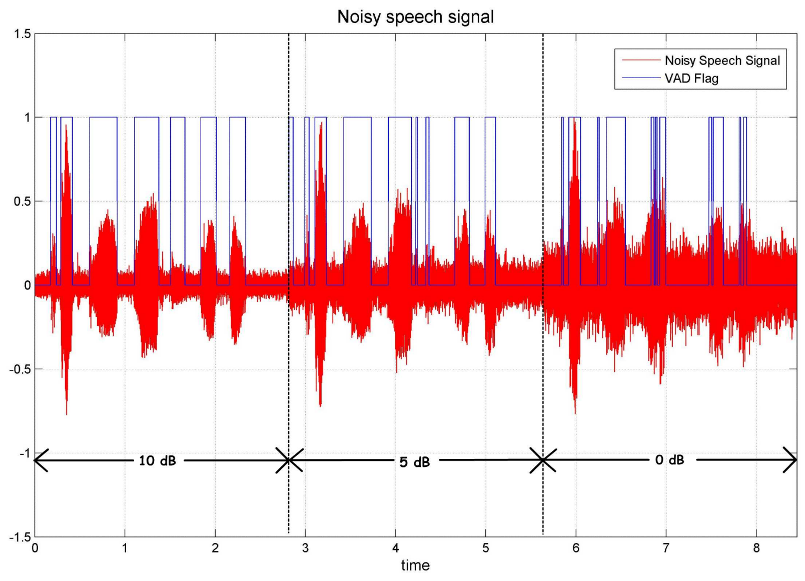

Figure 6.2: Noisy speech signal at different SNR level with voice activity flag

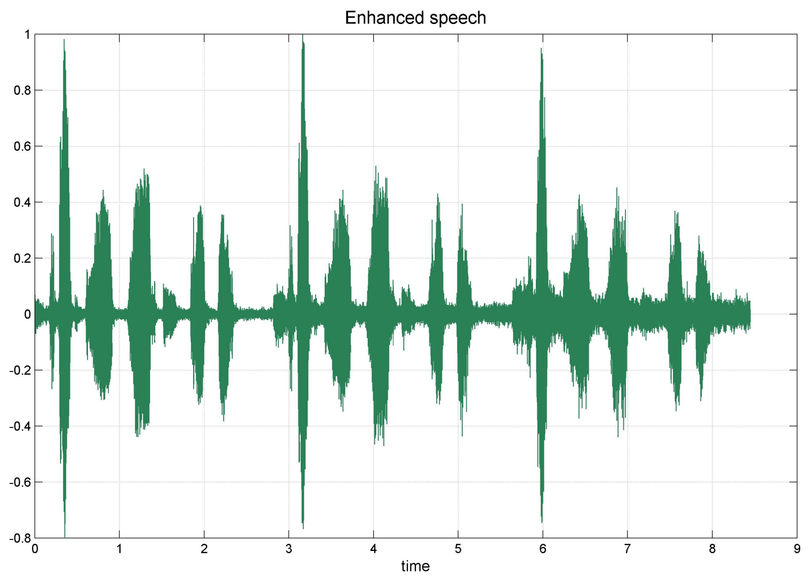

Figure 6.3: Enhanced Speech Signal

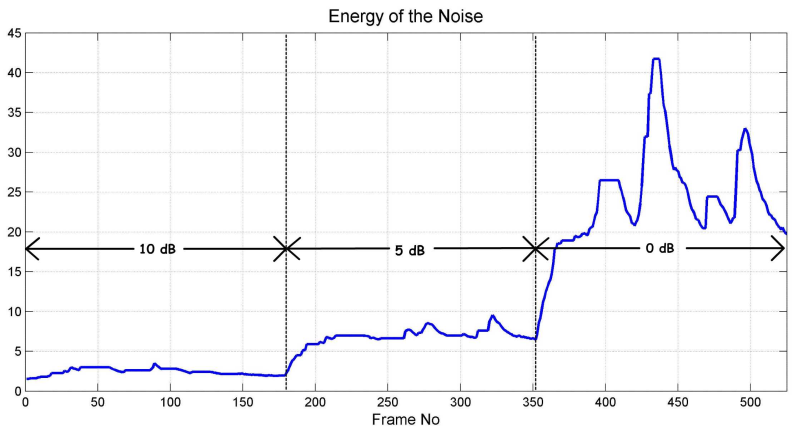

Figure 6.4: Estimated Noise Energy

is polluted by 5 db white noise, and the last part is corrupted by 0 db white noise. In this figure the blue line represents the voice activity flag. When this flag is 1 it means, that the VAD detects the frames as a speech. and when 0 it means that the VAD detects the frames as a Non voice Activity. From the figure it is evident that the VAD can successfully detect Non Voice Activity.

Fig. 6.3 shows the enhanced speech signal, i.e signal obtained by passing the noisy speech through the A Priori Estimation process. It clearly represents the improvement in quality if compared with the noisy signal input (Fig. 6.2).

In fig. 6.4 here is the graph of average estimated noise power in every frame calculated from the noise spectrum (sigma). From the figure it could be readily understood that the algorithm can successfully keep track with any change in noise level in the noisy speech signal.

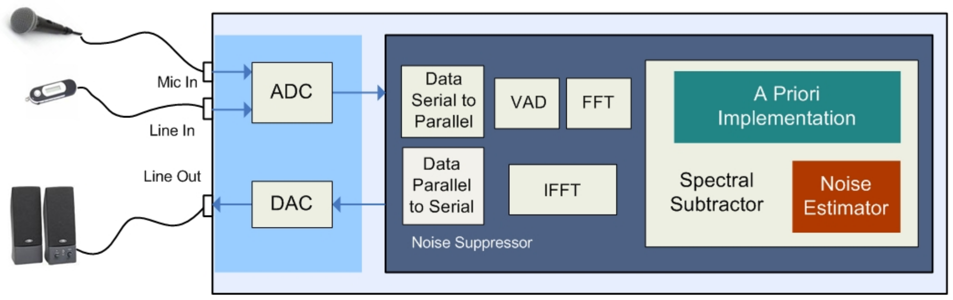

Figure 6.5: Basic Block Diagram

6.2 FPGA Implementation Technique for A Priori SNR Method

6.2.1 Overview

Figure 6.5 shows a simplified block diagram of the whole system of A Priori Estimation Method. The fundamental blocks are almost similar to that of weighted moving average method. However, now there is an additional block before FFT which is VAD (Voice Activity Detection) block. The Noise Estimator block is also embedded in the spectral subtraction process. The function of VAD is to decide whether a frame contains speech signal or not. Depending on this decision the spectral subtractor process implements the A priori estimation algorithm.

The whole process can be described as follows. The speech signal is given as input to the line in, the audio codec handles it and provides it as digitized serial data. The serial data is converted to 16 bit parallel data. The VAD(voice activity detection) block takes this parallelized data as input and calculates the zero crossing rate and short energy (see the VAD algorithms in Chapter 2). The 16 bit data is also fed to the FFT block which takes

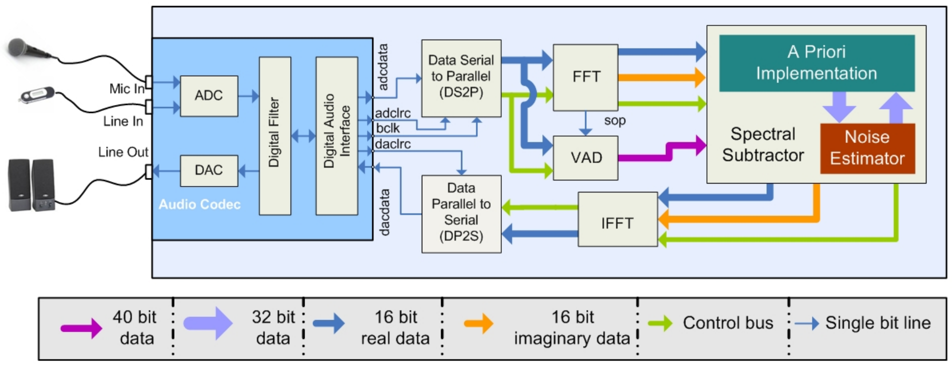

Figure 6.6: System Block Diagram

128 such data (a frame) and gives combined fast fourier transformed output. The magnitude spectra along with the vad output and some controlling signals are connected to spectral subtractor process which includes A Priori Implementation block and a noise estimator block. A Priori SNR estimation and speech enhancement considering voice activity are done by these blocksets. Finally the enhanced speech signal travels through IFFT, data parallel to serial block and audio codec to the outside speaker connected to lineout of the codec. A detailed view of the process is shown in figure 6.6.

6.2.2 Voice Activity Detector Block

The basic operational procedure of Voice Activity Detector(VAD) block is shown in Fig. 6.7. It takes the output of the data serial to parallel block as input, i.e. it works on the digitized input signal. The decision of presence of speech signal in a frame is affected by two computational results. One is the calculation of zero crossing rate in each frame. This is done by comparing the sign of the n-th data with the (n-1)-th data. If the signs are different then a variable is incremented, else no change is done. The rate of zero crossing

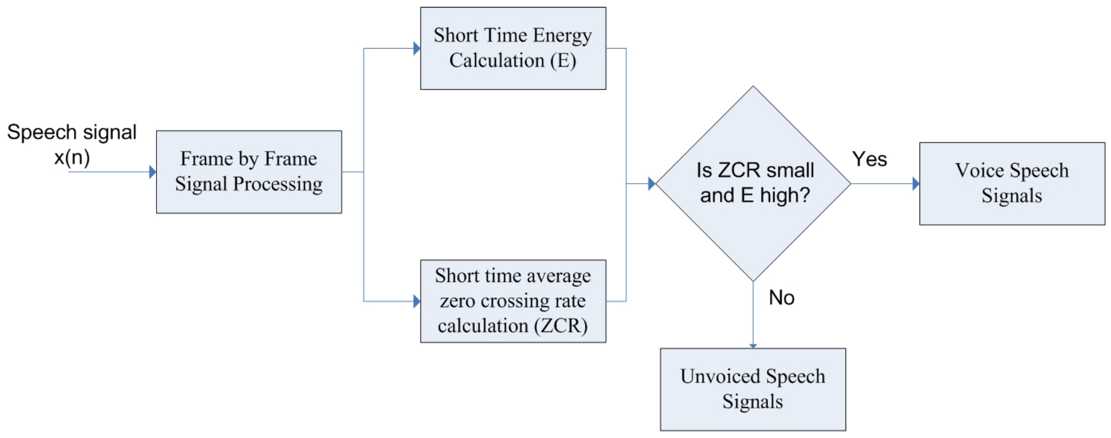
Speech signal x(n)
Frame by Frame Signal Processing
Short Time Energy Calculation (E)
Short time average zero crossing rate calculation (ZCR)
Is ZCR small and E high?
Yes
No
Voice Speech Signals
Unvoiced Speech Signals

Figure 6.7: System Block Diagram

is normally too high for noisy frames. In this research, there were 128 data points in a single frame and a zero crossing rate above 80 was considered to be the condition of a noisy frame.

However this condition was not enough, because in many cases, using only this condition may include speech signal containing frames as noisy frame. Specially for high pitched speech. Thus another condition for ensuring that no speech frame is misinterpreted as noise is needed to be considered. Calculating the short term energy is a suitable choice for that. It is very easy to calculate and so it does not need much logic elements. To get short term energy all the values of a frame are just squared and summed and then divided by the frame length. If the value of energy is large enough, the frame can be considered as speech containing frame. A condition is thus set for noisy frames, which is, if the value of short term energy is below 3700000 then the frame is to be considered noisy. This constant value is chosen by inspection and it simplifies the implementation process. However, it is suggested to use a conditional value for better result (if enough logic elements are available) as can be obtained from simulation.

Thus, if the zero crossing rate of a particular frame is above 80, while its energy level is below 3700000, then the frame is considered noisy in this implementation. Otherwise, the frame is to be considered a speech containing frame. The VAD block makes this decision.

6.2.3 Spectral Subtractor Block with A Priori SNR Estimation

The block diagram of Spectral Subtractor module (fig. 6.8), at a first glance, may seem almost same as that of moving average method. There are obviously some similarities like the intake real and imaginary values and calculation of the magnitude spectra and also the way of giving real and imaginary output values. However, the operation of A priori SNR speech enhancement module with the help of VAD and noise estimator is quite different.

The VAD block only provides the information, whether a frame contains speech or noise. Depending on this information the noise estimator block updates its estimation. If the current frame contains only noise then the noise estimator will use the following equation to obtain new estimation:

$$[\widehat{\sigma}_d(n,k)]^2 = \lambda_D[\widehat{\sigma}_d(n-1,k)]^2 + (1-\lambda_D \mid Y_{n,k} \mid^2$$

here $[\widehat{\sigma}_d(n,k)]^2$ is the new estimation of noise. The value of λ_D was chosen to be $\frac{116}{128} = 0.90625$ because it is computationally easier to multiply or divide by a number which can be represented as a power of 2. 128 is 2^7, so division is done internally only by shifting. In case of multiplication we write 116x as (128x-8x-4x) and thus make the process easier for hardware.

The latest and the previous noise estimations are both input to the A Priori Estimator block. The operation of the estimator is shown in fig 6.9. σ_{in} and σ_{out} are input from

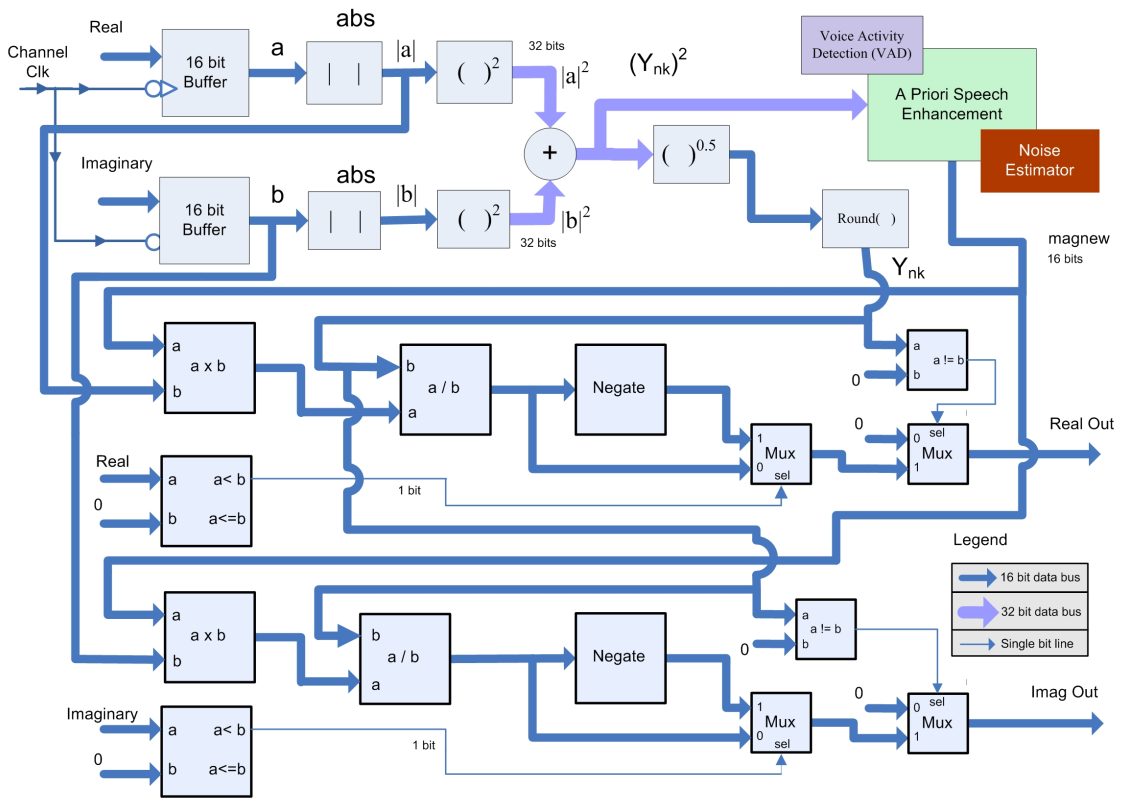

Figure 6.8: Spectral Subtractor Module in detail

the noise estimator. Another input, $Y_{n,k}^2$, is the squared value of the magnitude spectra, while X_{2nkin} is the Using these three values posteriori and priori SNR values are calculated. Finally the enhanced signal values are calculated using the equations described in the theoretical portion. The RAM block shown in the figure represents the memory block of the hardware. It stores, as shown in the figure, several arrays of data inputs and outputs.

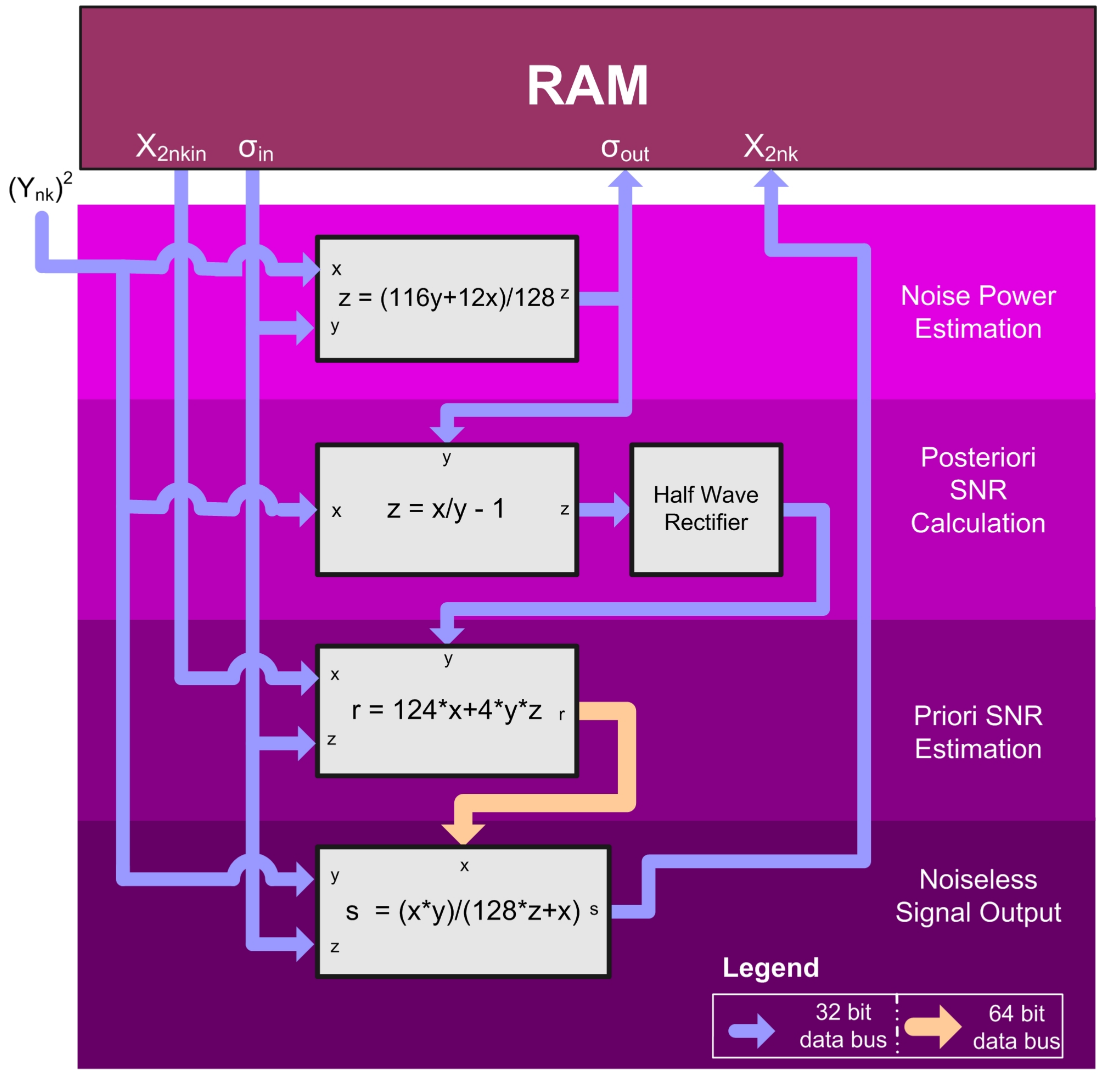

Figure 6.9: A Priori SNR Implementation Block

6.3 Experimental Results

Verilog HDL has been used as the Hardware Definition Language for this research. The complete implementation has been experimented in seven modules and a Audio Codec. The Resource usage in the FPGA board is shown below in Table 6.2. As can be seen, the efficient reduction of logic elements has lead to a total logic usage of only 70% for this complex algorithm, while it was 80% for WMA. The following techniques were applied for logic usage reduction:

- The multiplications and divisions were done by shifting, i.e. they were all represented by a power of 2.
- In many cases additions and subtractions were done instead of multiplication or division.
- Conditional statements for flooring or ceiling were ignored whenever acceptable (simulation was done to check acceptability).
- Recursive equations were completely ignored, temporary variables were declared instead, which lead to reduction of complexity but increment of memory usage.
- For Real Time Implementation of the algorithm, the operations were done in consecutive edges of clocks, so that no frame gets lost. This also made possible reuse of same logic elements at different time, which lead to huge logic element reduction.

Similar to the WMA method, the whole process was implemented in a single chip FPGA (Cyclone II EP2C35F672C6). Also a single chip Audio codec (WM8731) was used for data in/outs. The Verilog coding was done in Quartus II (ver. 7.2 sp3). The codec was set in 16 bit mode and the FFT/IFFT blocks were set for a data length of 128. The

Table 6.2: Resource Usage

Total Logic Element	23,154 / 33,216 (70%)
Combinational Functions	22,185 / 33,216 (67%)
Dedicated Logic Registers	7,168 / 33,216 (22%)
Total Registers	7168
Total Pins	14 / 475 (3%)
Total Memory Bits	46,092 / 483,840 (10%)
Embedded Multiplier	70 / 70 (100%)

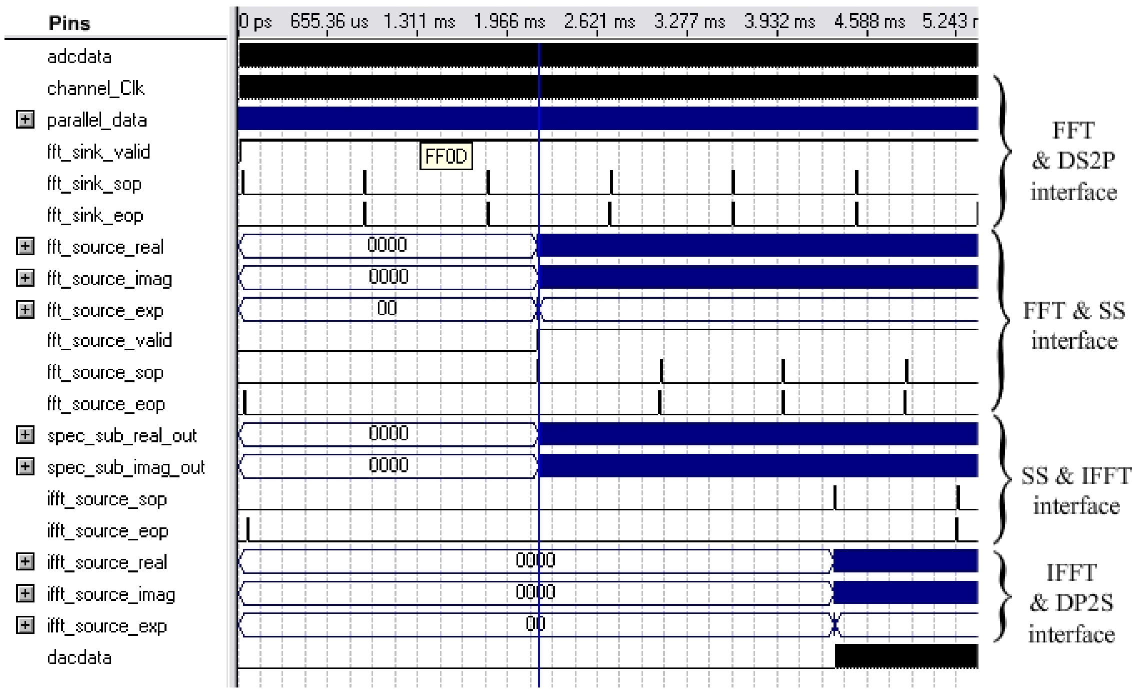

Figure 6.10: Timing diagram of overall data flow

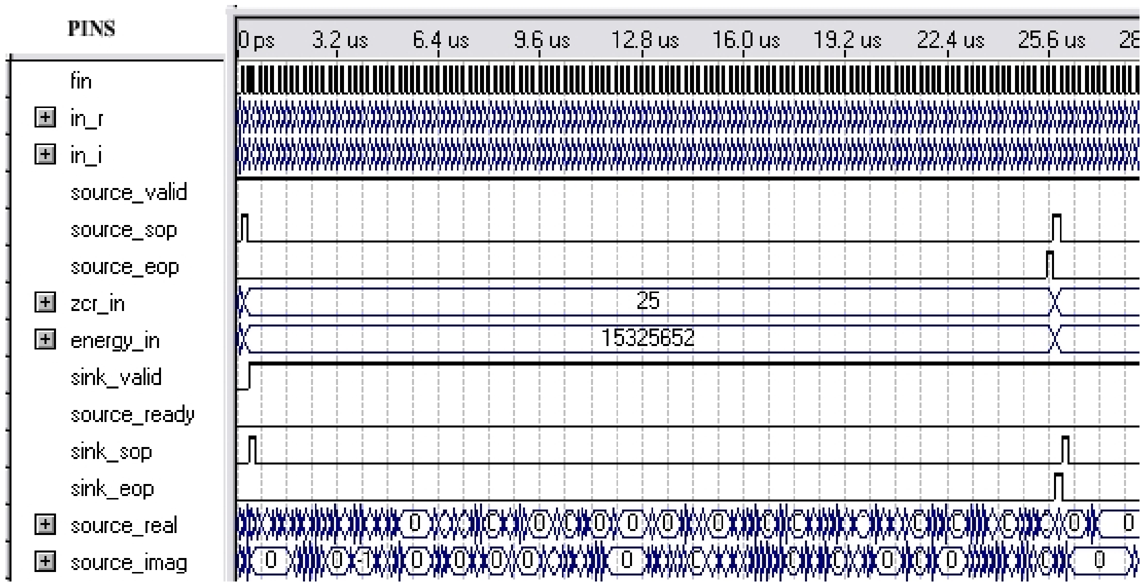

Figure 6.11: Timing diagram of the spectral subtractor block

output was satisfactory when input volume was low. This occurs, because of the 16 bit data resolution constraint.

Figure 6.10 shows the timing diagram of and control signal flow from module to module. As can be seen from the figure, the delay between input and output data stream is about 410μsec.

On the other hand, figure 6.11 shows the timing diagram of the spectral subtractor block for a given test data input. In the figure, fin represents the channel clock signal. in_r and in_i are real and imaginary data inputs respectively, source_valid, source_sop and source_eop are control signals from the previous FFT block. source_sop represents the start of a frame while source_eop indicates the end. Thus, for a frame length of 128, consecutive sop and eop signal contains 128 data points within themselves. Zcr is the estimate of zero crossing rate of the signals in a frame and energy is the estimation of signal energy level. Combining these two information the voice activity flag can be determined (very high zcr or very low energy means noise, so flag is zero, else one). Depending on this flag, the noise energy estimation is updated. The sink_sop and sink_eop signals are just shifted versions of source_sop and source_eop, because the spectral subtractor block starts giving output after just one cycle of fin. This figure implies that the implemented version is fast enough to give real time output.

Chapter 7

Conclusion

7.1 Summary

In this research, so far, we have successfully designed, simulated and implemented two speech enhancement techniques based on spectral subtraction. The simulation and experimental results manifest the improvement in overall quality.

The WMA method, though primitive in comparison to A Priori SNR method, but it still shows an improvement in following the variable SNR level. The increment in SNR ranges from 2 to around 8 dB, however in some cases intelligibility was lost.

On the other hand, the A Priori method improves the SNR from 2 to 4 dB while maintaining intelligibility of the voice signal.

This thesis thus contains substantial comparative analysis of these two algorithms. Simulation results for different noise types at different noise level are included. Standard noise signals were used for this methods. The calculation of SNR were also done following well-known methods like Averaging Segmented SNR and I.S. Method.

The final outcome of this research is the successful implementations of efficient noise suppression techniques in Cyclone II FPGA, while keeping the cost within limit.

7.2 Future Design Modifications

The implementation of a more complex but efficient algorithm is dependent on the total logic elements and memory elements of the FPGA chip. Cyclone II chip, used in this research, is a quite ordinary FPGA with dsp processing capability, specially for speech signal processing. In this research the prime concern was, thus, to implement speech enhancement algorithms in minimum resources. But this lead to precision loss in many cases (for example, floating point operation could not be done due to shortage in logic element), thus degraded quality of output. So, the first step of future improvement is to use a more powerful chip, for example Cyclone III or Stratix III.

Future works may also include upgrading the algorithms by considering the phase information and residual error. The modified A Priori SNR estimation with a variable value of the averaging parameter(α) may improve the performance by reducing musical noise and transient distortion[6]. The system could be developed for suppressing Non-white and Non-Stationary noise[16]. Several improved techniques like non-linear spectral subtraction[11] , variance measuring method[10] and wavelet spectral subtractor[7] may be consulted for further development.

Appendix A

Matlab Simulation Codes

A.1 For Weighted Moving Average Method

garbage

A.2 For A Priori SNR Estimation Method

garbage

garbage

Appendix B

Verilog codes

B.1 Common Basic Block Codes

B.1.1 Data Serial to Parallel Block

garbage

B.1.2 FFT/IFFT Block Using Megacore Functions

garbage

garbage

garbage

garbage

B.1.3 Square Root Megafunction

garbage

B.1.4 Data Parallel to Serial Block

garbage

B.2 Algorithm Implementation Blocks

B.2.1 Weighted Moving Average

Spectral Subtractor Block

garbage

garbage

Moving Average Block

garbage

Noise Estimator Block

garbage

B.2.2 A Priori SNR Estimation

Spectral Subtractor Block

garbage

garbage

garbage

Memory Block

garbage

Bibliography

[1] Steven F. Boll. ”Suppression of Acoustic Noise in Speech Using Spectral Subtraction”. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-27, No. 2, April 1979.

[2] C. Beaugeant and P. Scalart. ”Noise reduction Using Perceptual Spectral Change”. *EUROSPEECH’99*, pages 2543–2546, 1999.

[3] Ephraim and Malah. ”Speech Enhancement Using a Minimum Mean Square Error Short Time Spectral Amplitude Estimator”. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-32:1109–1121, 1984.

[4] Pascal Scalart and Jozue VIEIRA FILHO. ”Speech Enhancement Based on A PRIORI Signal to Noise Estimation”. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ICASSP, Vol. 2:629–632, 1996.

[5] Israel Cohen. ”Speech Enhancement Using a Noncausal A Priori SNR Estimator”. *IEEE Signal Proecssing Letters*, Vol. 11, No. 9, September 2004.

[6] Md. Kamrul Hasan, Sayeef Salahuddin, and M. Rezwan Khan. ”A Modified A Priori SNR for Speech Enhancement Using Spectral Subtraction Rules”. *IEEE Signal Proecssing Letters*, Vol. 11, No. 4:450–453, April 2004.

[7] Yuki Denda, Takanobu Nishiura, Hideki Kawahara, and Toshio Irino. ”Speech Recognition with wavelet spectral subtraction in real noisy environment”. *ICSP’04 Proceedings*.

[8] Bachu R.G., Kopparthi S., Adapa B., and Barkana B.D. ”Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal”. *ASEE*, 2008.

[9] Rabiner, L. R., and R. W. Schafer. ”Digital Processing of Speech Signals, Englewood Cliffs”. *512-ISBN-13:9780132136037*, 1978.

[10] Ashish Panda, Neha Tripathi, and Thambipillai Srikanthan. ”Improved Spectral Subtraction Technique for Text-Independent Speaker Varification”. *15th Intl. Conf. on Digital Signal Processing (DSP)*, 2007.

[11] Jiri Poruba. ”Speech Enhancement Based on Nonlinear Spectral Subtraction”. *Fourth IEEE International Conference on Devices, Circuits and Systems, Aruba*, April 17-19, 2002.

[12] Wikipedia. www.wikipedia.com/FPGA.

[13] *User Manual of DE2 Development and Education Board*. Altera Corporation.

[14] Audio Codec *WM8731/8731L* datasheet. Altera Corporation.

[15] *FFT Megacore Function, User Guide ALTERA MegaCore Version 7.2*. Altera Corporation.

[16] Scott M. McOlash, Russell J. Niederjohn, and James A. Heinen. ”A Spectral Subtraction Method for the Enhancement of Speech Corrupted by Non-White, Non-Stationary Noise”. *IEEE, IECON*, 1995.

[17] B. Gold. ”Robust speech processing”. Jan. 27, 1976. 1976-6, DDC AD-A01Z P99/0.

[18] Nicholas W.D. Evans, John S. D. Mason, Wei M. Liu, and Benoît Fauve. ”An Assessment of the Fundamental Limitaitons of Spectral Subtraciton”. *ICASSP*, 2006.

[19] Altera Corporation. *Quartus Hand Book*. www.altera.com.

[20] John G. Proakis and Dimitris G. Manolakis. ”Digital Signal Processing, 3rd Edition”.

[21] ”http://www.utdallas.edu/ loizu/speech/noizeus/”.

[22] Michael D. Ciletti. ”Advanced Digital Design with the Verilog HDL”. 2003 Edition.

[23] Scott M. McOlash, Russell J. Niederjohn and James A. Heinen. ”A Spectral Subtraction Method for the Enhancement of Speech Corrupted by Non-White, Non-Stationary Noise”. copyright IEEE 1995.

[24] I.Y.Soon and S.N.Koh. ”Low Distortion Speech Enhancement”. *IEEE Proc-Vis. Image Signal Process*, Volume-147, No-3, June 2000.

[25] Wee-Soon Ching and Peng Seng Toh. ”Enhancement of Speech Signal Corrupted by High Acoustic Noise”. *IEEE TENCON*, 1993.