HEARING LOSS SIMULATION

by

NOEL WILLIAM THYS

Dissertation submitted in fulfillment of the requirements towards the degree of Master of Technology in Electrical Engineering in the Faculty of Electrical Engineering at the Peninsula Technikon.

Internal supervisor: R. Key
External supervisors: Prof. G. De Jager (UCT)
Prof. J.G. Lourens (US)

Date of submission: 31 August 2000

DECLARATION

I, NOEL WILLIAM THYS, declare that this work is my own original work and has not been submitted before now, in any form whatsoever, by myself or anyone else, to any educational institution for assessment purposes.

The opinions contained herein are my own and not necessarily those of the Technikon.

Signed: .....

Date: 31/8/00

ACKNOWLEDGEMENT

I am grateful to the following people and institutions for their support and contributions to the present study:

My wife Juliet, for providing the idea to research this topic. Thanks for the love, support and patience while I was busy with this study.

Tony Abrahams, my former supervisor and inspiration, for putting me on the road with this study. His guidance laid the foundation of the work done in this research.

Professor G. De Jager for being my external supervisor. It was an honor to have the privilege to share in his knowledge and wisdom.

Robert Key, my internal supervisor, for his assistance, and advice.

Professor J. G. Lourens for helping and guiding me to produce work that meet the required standard. It was a privilege to work under his external supervision.

Nuwe Hoop Centre for the Hearing impaired, which was the motivation behind this investigation.

I would like to express my gratitude to the Peninsula Technikon and the Foundation of Research and Development for their financial support.

SUMMARY

This document gives a report on the research that has been done to simulate hearing loss. People working with the hearing impaired have no idea of what and/or how the hearing-impaired person hears sound. An instrument that enables a normal hearing person to hear what a hearing impaired person hears, is referred to in this document as a Hearing Loss Simulator (HLS).

An investigation of the feasibility and practicability of the abovementioned instrument, has led to the development of the HLS by making use of a distinct type of technology called Digital Signal Processing (DSP) technology.

Before hearing loss can be simulated, the hearing loss first needs to be determined. A study of different procedures and methods for screening hearing has led to the incorporation of an existing instrument called an Audiometer. An audiometer is an instrument that determines the hearing loss by making use of pure tone sine waves. The results are then plotted on a graph called an Audiogram. The results of other methods that determine the hearing loss can also be transferred to the Audiogram. The Audiogram's information which is, in fact, the frequency response of the ear, is stored in a computer and is utilized to realize the HLS function.

Six different DSP based methods were studied to shape any audio information according to a specific frequency response. The optimum method was identified and then implemented. The various methods are the following:
Filter bank method
Inverse Discrete Fourier Transforms (IDFT) method
Inverse Fast Fourier Transforms (IFFT) method
Chirp-z method
Wavelet method
Yule-walker method

The IFFT method was identified as the optimum method and was therefore implemented. The algorithm to realize this method, was carried out by doing the IFFT calculation on computer and subsequently doing the filtering on a DSP processor called an ADSP-2181 processor.

The audio information under investigation is fed into the instrument, it is then filtered according to the audiogram information and then sent out again. Any normal hearing person who wants to investigate the hearing impairment of the hearing impaired person can listen to sound reproduction through either a set of headphones or through a free field. This process is referred to in this dissertation as Hearing Loss Simulation.

OUTLINE OF DOCUMENT

TABLE OF CONTENTS

PART I ***Conceptual idea of the problem and possible solutions***

11. The actual implementation

PART IV Evaluation and conclusions

12. Evaluation

13. Conclusions and recommendations

LIST OF FIGURES

Page

LIST OF TABLES

PROGRAM LISTS

APPENDIXES

Part I

Conceptual idea of the problem and possible solutions

CHAPTER 1

INTRODUCTION

1.1 Introduction

In South African more than 10% of the population has some form of hearing deficiency. According to a report from a subcommittee for hearing impairment of the Department of National Health and Population development (1981:5), the first problem encountered by the hearing impaired person is a language problem. The language problem in turn gives rise to secondary problems such as speech problems, cognitive problems, social and economical problems, emotional problems, educational problems and vocational problems. Even the slightest hearing loss can cause speech disorder and learning problems. Children with hearing impairment often have behaviour problems, secondary to their inability to hear well, follow instructions or interact fully in social communication. In this regard Davis and Silverman (1978) said:

> "Man's need for communication with his fellow man is possibly his greatest need and the fulfilment of his other needs and desires is largely dependent upon, or at the last greatly facilitated by, his ability to satisfy this basic one. The development of language, both spoken and written, as a means of communication is one of mankind's greatest achievements. From birth hearing people, effortlessly, almost unconsciously, have absorbed this magnificent tool simply because they are lucky to hear. This gift is very much taken for granted. The hearing impaired child on the other hand has to learn, bit by bit, word by word, sound by sound, a workable, even though imperfect, language tool for himself".

People in contact with the hearing impaired do not always understand the problems that hearing-impaired persons encounter with respect to their hearing and language development. The opinion of professionals, and often parents, of the pupils' hearing capabilities can be tainted by their inability to experience deafness. There is a need for professionals to have a more realistic understanding of the hearing impaired child based on objective inferences (Schloss and Goldsmith, 1986). A lack of research tools has prevented a clear understanding of the way hearing impaired children perceive sound (Oblowitz, 1988). How the hearing-impaired person hears sound and gives meaning to any audio stimulation has been an unexplored field. The fact that it is literally impossible to get into another person's head and hear what he or she hears has limited research in this area. The question came up amongst teachers of the hearing impaired whether any instrument is available that will enable them to hear what the deaf child hears. As stated by Thys (1999) a teacher at Nuwe Hoop Centre for the hearing impaired (Worcester): " There is a common

need amongst colleagues to know what the hearing impaired has heard during a lesson presentation. Being able to hear what the deaf child hears, would help to understand impaired hearing better. It will also help to know how to approach each student as his or her hearing loss differs individually. When exposed to the sound of the hearing impaired the teacher might adjust the volume and/or the tone of his/her voice or even the articulation and speed of words uttered so as to benefit the child with a specific hearing loss".

As a result of their hearing loss children with impaired hearing and with a normal intellectual potential still show, according to Moores et al. (1978), scholastic and communication insufficiency after years of auditory training and even with a special syllabus offered by dedicated professional staff. Puren (1985) concluded that this clearly shows how far away from their goals educators of children with hearing loss are.

Before a solution is presented to the problem that teachers have in comprehending hearing loss, the term hearing loss needs first to be defined.

The term hearing loss, which is at the centre of many secondary problems, has been defined in many ways. One authoritative definition of hearing loss according to Van Cleve (1987:14) is diminishing hearing sensitivity measured in decibels (dB), which are units indicating the magnitude of sound. The zero or reference decibel measure is calibrated to the threshold level of response of the average, young normal adult to tones of different frequencies used in hearing evaluation. Threshold hearing levels for a hearing impaired person are plotted on a graph called an audiogram. Figure 1.1 shows an audiogram of a person with a hearing loss of 40 dBs at 125 Hz, 50 dBs at 250 Hz, 50 dBs at 500 Hz, 50 dBs at 1kHz, 60 dBs at 2 kHz, 60 dBs at 4 kHz and 40 dBs at 8 kHz. The vertical scale represents Sound Pressure Levels (SPL) measured in dBs, which will be discussed in detail, in chapter 2. The audiogram shows the response of the ear to pure sine wave sound stimulation at the specified frequencies.

No two people have exactly the same hearing loss. Hearing losses differ as much as people's fingerprints differ. A hearing test is performed on each ear so that each ear has its own audiogram. The hearing loss of both ears can also be plotted on the same audiogram where the response of the right ear is marked with circles (O) and that of the left ear is marked with crosses (X) by convention. The line connecting the circles is normally drawn

with a red pen and that connecting the crosses is normally drawn with a blue pen. Figure 1.1 is thus a representation of the hearing of the right ear. From 540 students at the school *Nuwe Hoop Centre for the Hearing Impaired* (in Worcester) no two audiograms were found to be identical. This means that to classify hearing losses would give a general idea of the range of hearing loss, but would be a less than accurate description of a hearing loss.

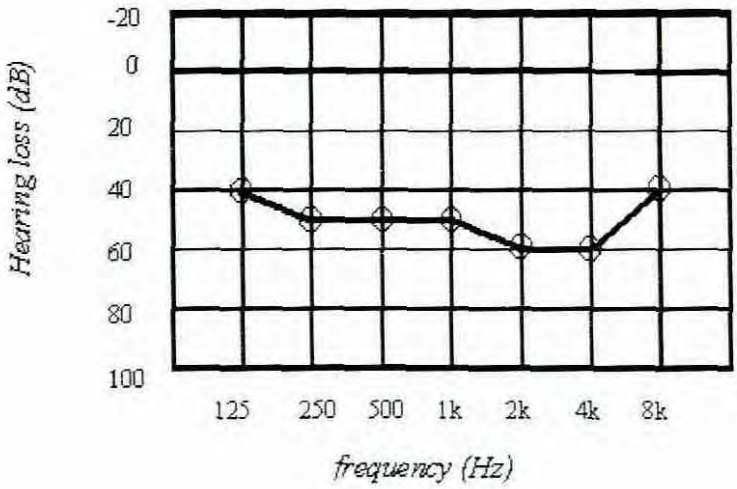


Figure 1.1 Example of an audiogram

Ballantyne (1977:161) stated that there is no satisfactory classification of deafness or hearing loss in children. He went on by saying :

> "We can define the hearing-loss, according to the audiogram, as a 'flat' loss, or a 'high-tone' loss, or a 'low-tone' loss. We can classify it according to degree and we can fix an arbitrary dividing line between a total or sub-total deafness on the one hand and a partial deafness on the other. And we can further subdivide partial deafness into secondary degrees of 'slight', 'moderate', or 'severe'. But here certain difficulties arise. For example, the child who hears normally (on the audiogram) at 500 and 1000 Hz but has an abrupt drop to 80 dB at 2000 and 4000 Hz will have an average hearing loss of 40 dB for the so-called 'speech frequencies'. Yet in most instances, his handicap will almost certainly be less severe than that of a child, who has a 'flat' loss of 40 dB at each of these frequencies, although the 'average' hearing loss is the same in both. Moreover, the steady advance in the development of electronic equipment must surely change our conception of which children we regard as 'deaf' and which as 'partially deaf'."

In response to the last sentence it could be said that an instrument that is able to simulate a hearing loss would be the realisation of the development of electronic equipment to verify our conception of which children are regarded as deaf and which as 'partially deaf'. In

other words an electronic piece of equipment is needed to give a teacher a better understanding of the hearing loss of a specific student. An instrument to solve the problem teachers has in comprehending hearing loss can be called a hearing loss simulator.

The term used in the title of this research a Hearing Loss Simulator (HLS), are defined as follows

Hearing – Perception by ear

Loss – Deprivation

Simulate – To reproduce the condition of a situation

Simulator – An apparatus or instrument which allows a person to sense what real conditions are like.

A hearing loss simulator is thus, an instrument reproducing the condition of deprived perception by ear.

Simulation is, today, an important way to do research in areas where real experimentation is impossible. For example, teachers have no way of hearing the actual sound the hearing impaired child hears. A hearing loss simulator is able to perform this function, since it will generate conditions of hearing loss. Any sound can be used as an input to the hearing loss simulator. The HLS performs the necessary attenuation on the input signal according to the audiogram. The output is fed to a set of headphones over which any interested party can listen to the sound reproduction which is, in fact, the simulation of hearing impairment. By playing back a recorded sound track through the HLS, which performs the necessary attenuation, teachers can hear what the hearing impaired child hears.

Teachers at the aforementioned school state that it is difficult to imagine a specific hearing loss by looking at the audiogram. The Hearing Loss Simulator will help as it will expose teachers, parents or any interesting party to the actual sound heard by the hearing impaired person by filtering out or attenuating sound frequencies according to the audiogram. The teacher can then listen through headphones connected to HLS to the sound track of attenuated sound, which thereby simulates impaired hearing. The presumption is made that if a teacher is able to form an auditory perception identical to that of the hearing impaired child, he or she will understand the student with hearing impairment better. The teacher can literally place him/herself in the shoes of the student with impaired hearing. By listening to a simulation of impaired hearing, thus experiencing impaired hearing, a teacher may

improve his/her teaching methods. For instance, if the teacher knows that the degree of deafness is so large that the student does not benefit from the use of hearing aids, then the teacher might decide to concentrate on sign language in the teacher/student interaction with that specific student.

The teacher might also use the instrument to determine what the probability might be of mastering important auditory skills. Auditory skills that are important and also form part of an auditory training programme are discussed by Norman P. Erber and Ira J.Hirsh in a contributing chapter "Auditory training" in the book "Hearing and Deafness" by David and Silverman (1978:358). A detailed discussion on an auditory training programme would fall outside the scope of this report but aspects that are concentrated on are the following:
Detection: Detection is the basic process of determining whether sound is present or absent.
Discrimination: Discrimination allows the listener to perceive the differences between sounds; failure to discriminate yields perception of similarity.
Identification: Identification, or recognition, responses are simply labels for what the listener has heard. He/she indicates this by saying, pointing to, or writing the word or sentence that he/she has perceived. These all are forms of repetition response, in which the listener must recall the stimulus in some way.
Comprehension: Comprehension is the most difficult auditory skill. Comprehension of speech requires that the listener understand the meaning of acoustic messages, usually on the basis of his knowledge of language. It also requires that he/she be able to acquire new information through his ears and act appropriately on this basis. Teachers of hearing-impaired children often evaluate their pupils' auditory comprehension by giving instructions or asking questions in the classroom under conditions of acoustic-only stimulation.

Apart from the auditory training there is also speech training that the teacher has to consider as part of the education programme. In some cases speech training is given by means of audio stimulation for those students who have mild to severe hearing losses and who wear hearing aids. As for normal hearing children, the child learns words through what he/she hears. By using an HLS the teacher can listen to those words that he/she wants to teach the child. If for instance he/she finds out that the low frequency parts of the words come through well, but not the high frequency parts, then the teacher might consider techniques to teach the high frequency parts. A good reference on techniques for the

development of speech can be found in a book by Davis and Silverman (1978) titled "Hearing and Deafness". A discussion on alternative techniques for speech training falls outside the scope of this report.

1.2 Statement of the problem

The problem is that teachers cannot effectively teach children with hearing loss.

1.3 Delimitation of the problem

This study focuses only on a certain part of the problem. The teaching of children with hearing loss is rather a wide and intense field of research. As such, this study does not intend to solve the entire problem of hearing loss but only to provide an aid for teachers, which would help them in their teaching.

1.4 Objective of the study

The objective of the study is to construct an instrument that can simulate hearing losses as presented on the audiogram. Building a basic filter through which an audio signal is send can accomplish this objective. The research consequently focuses on the following two aspects:

- The choice of filter – what type filter to be used and
- The choice of implementation – the way in which the filter will be implemented

The first stage of this research deals with the choice of filter and is called the "Theoretical solution". In this section various Digital Signal Processing (DSP) methods are investigated, documented and then their feasibility is commented on. It was explicitly instructed that the following methods be studied and documented for the project.

a) Filterbank method using IIR filters
b) Inverse Discrete Fourier Transforms (IDFT)
c) Inverse Fast Fourier Transform (IFFT)
d) Chirp Z transforms
e) Wavelets
f) ARMA filter design by the Yule-Walkers method

One of the above methods is chosen for implementation.

During the second stage, the "Practical Implementation", the chosen method is implemented on specific DSP hardware. The implementation will be discussed in further detail later in the document.

The aim of the project would be to design and develop a hearing loss simulator (as defined in the introduction) that is portable, reliable, fast and accurate. A key objective is to make the instrument affordable and accessible for parents and schools for the hearing impaired or anybody interested in entering the 'world of the deaf'.

1.5 Research methodology

The way to go about solving the problem is as follows.
The research is basically broken up into four phases, which are listed below. A further detailed discussion of each phase is given in the sub sections that follow.

* Conceptual idea of the problem and how it can be solved
* Theoretical solutions
* Practical implementation
* Evaluation

1.5.1 Conceptual idea of the problem and how it can be solved

This includes the awareness of the problem and brainstorming techniques for solving the problem.

The problem is that teachers cannot effectively teach children with hearing loss. This phenomenon has manifested in the development of various philosophies of education regarding the hearing impaired. The different educational philosophies (which will be dealt with in chapter two) are based on diverse views on how a hearing-impaired child hears sound and how he/she gives meaning to it. The idea of this study is to develop an instrument that will illuminate the diverse views and expose teachers to the actual sound that the hearing impaired hears. These philosophies can thus be verified empirically. As stated, the presumption is made that if a teacher hears what the hearing impaired child hears he/she will understand the child better and will come up with an individual plan or strategy

for that specific child. As mentioned under "delimitation of the problem" the aim is to provide an aid for teachers that would help them in their teaching.

The concept of using the audiogram's information was confirmed during a brainstorming session, as the hearing loss information obtained by most methods[1] can be transferred to the audiogram. Obviously the hearing loss first has to be determined before any simulation can take place. The next problem is how the audiogram information can be used.

The idea arose of using an existing instrument, called an audiometer, which measures the hearing loss and automatically plots the audiogram. This instrument is a computerised system and was developed by Thys (1996) as a B.Tech project. The use of a computer enables the audiogram information to be stored and used during the filtering (attenuation process). The existing instrument makes use of Digital Signal Processing (DSP) techniques, which means that the development of the HLS has to be based on the same type of technology.

At this stage the term DSP needs to be explained and its use in this application needs to be motivated. Currently there are two types of competing technologies namely DSP technology versus Analog technology. DSP technology is, and can only be, explained in relation to analog technology.

An analog representation can be defined as a continuous process over a continuous range of values. Most processes and signals in the real world are in analog form. A phenomenon like acceleration presents itself in an analog form, as it is a continuous process over a continuous range of values. It would be impossible to jump directly from 10 km/h to 20 km/h. Digital representation on the other hand is a step by step representation with no ambiguity open to interpretation. The number of students in a class is an example of a digital representation.

Analog technology makes use of discrete components such as transistors, capacitors, inductors and linear integrated circuits to process signals. Analog technology also refers to the traditional way of electronic circuit design. Amplification, for instance, is done by making use of transistor circuits which operate directly on continuous varying signals.

[1] There are various methods to measure hearing loss which are discussed in chapter two under the heading "Determining the frequency response of the ear".

Filters are traditionally designed by making use of discrete components such as capacitors, inductors and resistors.

Digital signal processing, on the other hand, breaks the signal up into samples that have specific numerical values. The signal is then said to be digitised at discrete time intervals. Mathematical calculations are then performed on these samples to produce the required effect. Amplification with DSP technology, for instance, is done by means of a simple multiplication operation on these numerical values. In this case high speed processors optimised for digital signal processing are used to perform the calculation. Circuit functions such as filters can be implemented by the use of difference equations and solved in real time with compact digital hardware of which a DSP processor is the core (Higgins, 1990). The result of any mathematical operation is converted from the digital domain to the analog domain to get it back into continuous form as it is normally represented in the real world.

The key issue to be understood with DSP technology, as apposed to analog technology, is that, "program development replaces circuit development" (Higgins, 1990). The fact that DSP applications are developed by means of program development is the main motivation for doing this project digitally. The advantage of this approach is the prospect that developing extra software or writing additional programs can increase the functionality of an application. In this regard the HLS incorporates an existing program of a computerised, DSP based, two channels, pure tone audiometer. Both functions, the HLS and the audiometer, are performed on the same standard DSP development platform and both programs are self-written in assembler language using an asm 2181compiler. Expanding the application by, for example, adding a spectral analyser would simply mean the development of extra programes. This approach was decided on as this project would later form part of a complete Acoustic analyser. The potential is that it can be implemented as a complete administrator when a database with relative information about patients and/or students is stored. This type of expansion has no analog equivalent.

Other advantages of DSP technology are, as Muller (1990) points out:

1) Flexibility, in that a vast number of functions can be performed by a finite number of elements;
2) High precisions can be obtained by increasing the word lengths of data;
3) Processing time is such that real time digital signal processing can be done.
4) Digital systems are economic to use.

In comparing these two processes Higgins also refers to the predictability of Digital signal processing as an advantage, whereas analog signal-processing components might be a source of unpredictable drift and noise. The advent of Digital Signal Processing (DSP) technology has made the design of a low cost instrument of this nature possible. Because of DSP real time computational power, DSP technology is well suited for the audiological field where measurements and analysis have to be done. The rapid growth in the exploitation of DSP technology is not surprising considering the commercial advantages in terms of fast, flexible, precise and potentially low-cost capabilities offered by these devices.

Any development resulted from this study will be in terms of software development due, to the mathematical nature of DSP. The actual filtering, or the required attenuation, is performed on the DSP side of the instrument, which means that software needs to be written to do the required filtering. When software is developed a specific algorithm is normally followed. The term algorithm is defined by Summer (1987) as a list of instructions, especially to computers, which are carried out in a fixed order to find the answer to a question. The algorithm is normally theoretically based. To bring about the required attenuation (according to the audiogram) various algorithms need to be considered. As the saying goes there is, and should be more, than one solution to a problem. The problem now (which is also the main problem and focus of the study), is to find the best algorithms to realise the required filtering or attenuation with a given hardware. Best in terms of the most economical, the simplest, stable as well as having an execution speed and memory requirements that will suite the proposed hardware. The user interface should be attractive, comfortable and easy for the user to use. The work carried out in finding the optimum technique or, rephrasing it, of finding the best algorithm, is the central theme in part II of the document, which is titled "Theoretical solution". Finding the best theoretical solution and/or finding the best algorithm are used as synonyms in this text.

1.5.2 Theoretical solution

1.5.2.1 Statement of the technique

The very first step in an attempt to simulate hearing loss would be to determine the specific hearing loss. There are various methods to determine the hearing loss or determining the frequency response of the ear. Determining the hearing loss and determining the frequency

response of the ear are, in fact, the same thing and are used as synonyms within this text. The methods for measuring hearing loss are Air conduction tests, Bone conduction tests, ABR and Speech discrimination tests. These methods and also other less complicated methods are documented in chapter two. The results of any of these tests can be transferred to the audiogram. The idea is to use the audiogram information so that the amplitude values at specific frequencies as specified on the audiogram are used to identify the levels of attenuation on the input signal.

There are two techniques that can be used to simulate hearing loss, namely: Time domain analysis and Frequency domain analysis. These techniques are implemented on DSP so that it can have real-time operation.

1.5.2.2 Areas of investigation

The areas of investigation are those listed below, which can be categorised into either frequency or time domain methods:

- *Filter method using a bank of filters - Time domain*
- Inverse Fouriers transforms - Frequency domain
- Inverse Chirp Z transforms - Frequency domain
- Wavelet approach / sub band decomposition - Time domain
- ARMA filter design by Yule-Walkers method - Time domain

A brief overview of each of these methods will be given in chapter 2 and a full investigation will appear in part II, where each method will be discussed in a separate chapter.

1.5.3 Practical implementation

The third phase of the research method forms the heart of the research project. It uses the empirical method to realise the practical implementation. Here practical implementation refers not only to the actual DSP implementation, but also to the simulation of the various algorithms using Matlab. Matlab is a mathematical package with an easy to use graphical interface for PCs. Experiments can be conducted with the built in "Signal Processing Toolbox", by which the proposed algorithms can be visually evaluated. The mathematical parameters can be controlled and the built in mathematical program can be exploited so that

the results can be empirically manipulated and re-evaluated. This results in a reduction in time and complexity and also in an increased accuracy, as it eliminates continuous mathematical derivations. These simulations can be used to confirm the theoretical findings in terms of the optimum method. The practical implementation is also concerned with the design specification. This includes the specification of the Hearing Loss Simulator hardware and software requirements. A brief overview of the hardware and software specifications will be given and a detailed discussion will be done in part III of this document. The hardware requirements will be discussed first, followed by the software requirements.

Hardware requirements: Figure 1.2 displays the block diagram of the HLS. As shown, the instrument consist basically consists of two main components, namely a PC and the DSP hardware. The DSP hardware uses what is called the EZ_KIT Lite. The EZ-KIT lite is a complete development system package, optimal for the development of DSP applications. Onboard is an ADSP_2181 programmable single-chip processor with a base architecture optimised for digital signal processing. All other required DSP hardware such as the CODEC and serial communication facilities is embedded on this evaluation board. Additional analog amplifiers are incorporated to realise the required sound pressure levels. Communication between the PC and the DSP board is by means of serial communication through the serial ports.

The software development of the final product is done in two sections, namely, high level development on the PC, and low level development on the DSP side. The high-level development on the PC is the software or the program written for the PC to establish user interface as well as controlling the digital signal processor. This program is written in Visual Basic language. The low-level development on the DSP is the software or the program written to perform the actual filtering on the DSP side. All functions performed by the hearing loss simulator, use the ADSP 2181 processor and had to be realised by writing software for the asm 2181 compiler in assembler language.

1.5.4 Evaluation

The last phase of the research method is the evaluation. In the evaluation procedures the following specifications of the instrument will be assessed:

- Accuracy
- Speed
- Wave purity
- User interface
- Correlation between Theory and practice

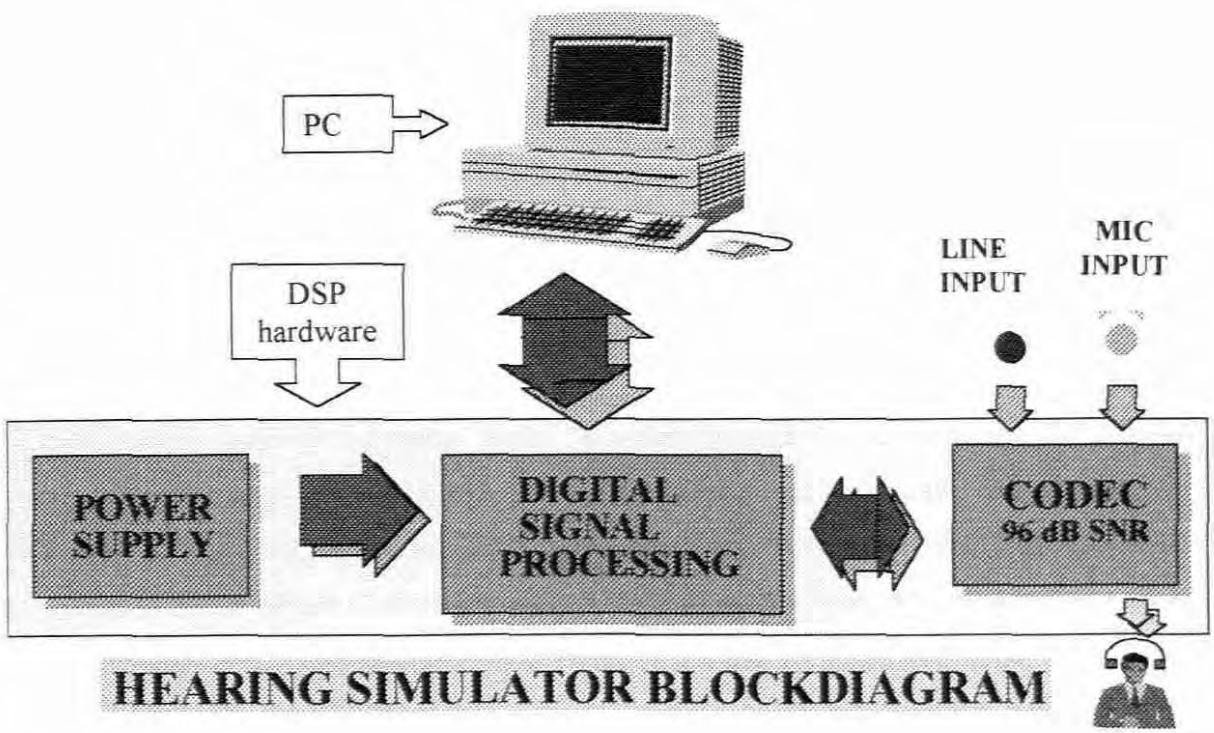


Figure 1.2 HLS platform

Accuracy: Accuracy refers to the frequency accuracy as well as the accuracy in sound pressure level. The possibility exists that certain algorithms might result in frequency drift. The frequency resolution might also affect the accuracy. For instance, if a limited number of bandpass filters are used, then all the frequencies within the band of a specific passband will be attenuated to the same level. This will cause an inaccuracy in amplitude in those signal frequencies further away from the centre of the band. This issue will be discussed in further detail in chapter 3.

Speed and reliability: The HLS has to operate in real time, which requires an algorithm that results in the fastest execution speed and most reliable operation.

Wave purity: Wave purity is concerned with presence of distortion.

User interface: How user-friendly is the instrument?

Correlation between theory and practice: The question here is, what is the degree of correlation between what was presumed to be achievable in theory and that which was actually achieved in practice.

1.6 Limitations of the study

It should be kept in mind that this is an engineering research project and not a research project in the human or social sciences. As such, it will fall within the limits of the engineering field. Thus the aim is not to strategize the use of the instrument, as this is obviously the work of other disciplines. The assumption here is that if the audiogram represents the hearing loss, then an arbitrary shaped filter that matches that frequency response of the ear will produce the effect of that hearing loss. The research is based on that assumption and is limited to the techniques listed in section 1.5.2.2.

This research is limited to a linear system. Most of the pathologies of the auditory system resulting in a hearing loss can be modeled by a linear system such as the one proposed by this project. There are, however, some pathological symptoms that result in non-linear responses. One of these is "Recruitment". Soer (1999) explains: "Recruitment is a large increase in the perceived loudness of a signal produced by relatively small increase in intensity above threshold. This implies that these persons will have a small dynamic range and it is usually associated with damage to the cochlea". Non-linear systems in the field of acoustics have up to now received very little attention. To justify this statement a study of the acoustics characteristics of various hearing aids from various manufactures has been conducted and consequently reveals that there is no hearing aid currently on the market that allows for this phenomenon of non-linearity. By means of a literature survey no model could be found that characterised this non-linear phenomena in contrast to the audiogram that presents and assumes linearity in terms of the loudness scale. It should be acknowledged at this stage that further research in terms of measuring non-linear response of the ear need to be done. The standard audiological test battery in which the audiometer is used does not measure non-linear behavioural characteristics of the ear such as Recruitment. This research does not claim to reproduce all forms of hearing loss but promises better understanding of the audiogram than studying the audiogram visually. The research is thus confined to a linear system, of which the listed techniques are studied and documented. After a comparison of the various techniques only one method/technique is

implemented. Lastly, the material presented in this document assumes a basic knowledge of Digital Signal Processing.

1.7 Report outline

The report is divided into four main parts:

Part I - *Introduction*: Conceptual idea of the problem and a possible solution

Part II – Theoretical solution

Part III – Practical implementation

Part IV – Evaluation and conclusion

Each part has its respective chapters as listed below.

Part I – Introduction: Conceptual idea of the problem and a possible solution.

- Chapter 1. As an introduction, an overview of the research project is given.
- Chapter 2. Background study and literature survey.

Part II - *Theoretical solution*

The algorithms researched and explored with their respective trade-offs (advantages and disadvantages) are discussed in six chapters in the following order.

Chapter 3: Filterbank method

Chapter 4: Inverse Discrete Fourier Transform method

Chapter 5: Fast Fourier transforms

Chapter 6: Wavelets

Chapter 7: Chirp z transforms

Chapter 8: ARMA filter by means of the "Modified Yule-walker" equations

Chapter 9: Comparison between the various methods.

The optimum method for the project, given the constraints, time, cost, analysis speed etc, is discussed with the progression of each chapter in the document.

Part III, which consists of Chapters 10 and 11, is devoted to the practical implementation of the theoretical solution and will cover the following:

Chapter 10 is an overview of the necessary areas of the DSP architecture to explain how the algorithm was implemented.

Chapter 11: Deals with the actual implementation. Section 11.1I introduces both high and low level developments. In section 11.2 the actual implementation on

the host side is explained and section 11.3 explains the low level development on DSP side.

Part IV – Chapter 12.1 to 12.12 discusses the evaluation of the instrument. The results of the evaluation and the interpretation of results are also illustrated. Section 12.13 covers the problems encountered and further areas of studies.

Chapter 13 – Conclusions and Recommendations.

CHAPTER 2

BACKGROUND STUDY
AND
REVIEW ON RELATED LITERATURE

2.1 Introduction

In this chapter the different educational philosophies which highlight the need for a hearing loss simulator are first discussed. Secondly, a further motivation for the study is given by showing the relation to other disciplines. Thirdly, methods to determine the frequency response of the ear are investigated. Investigating the frequency response of the ear is an integral part of the Audiological discipline and will be discussed under the heading audiometry. The chapter ends by giving an overview of related literature, which also acts as introduction to part II.

2.2 Different educational philosophies

2.2.1 Introduction

For the last 200 years the most acrimonious dispute among educators of the hearing impaired and occasionally among hearing impaired persons themselves, has been over communication methods. Moores (1987:10) says that the communication controversy has accounted for more confusion than any other question in the field and it is not surprising that professionals are not clear about the issue. Many different approaches to the education of the hearing impaired have arisen as a result. The appearance of different educational philosophies reflects the diverse views on how a hearing-impaired child hears sound and how he/she gives meaning to it. All professionals who work with these children will have their own bias. A general description of the various approaches is given below. The aim is to show that the outcome of this study (HLS) can be used to either prove or refute the following theories of education.

2.2.2 Auditory Verbal Approach

The auditory verbal approach to the education of hearing handicapped children is based on the premise that most of these children, even those with severe and profound hearing losses, can learn to listen, to hear and to develop speech and language by means of an auditory channel (Davis & Silverman, 1978). Since the goal of this approach is to integrate audition into the total personality of the hearing impaired child, the emphasis throughout

the training is upon listening - not looking. It is not an approach that can be successfully combined with a programme that stresses formal training in lip-reading or the use of any manual system of communication.

The children are fitted with appropriate hearing aids as soon as the diagnosis of hearing impairment has been confirmed. Parents are trained to increase their child's awareness of sound by calling his/her attention to meaningful sounds in his/her environment. They are shown how to help the child develop language comprehension and speech by talking close to him/her so that he/she has the best opportunity to hear what is being said. They are instructed to use the same simple words and phrases over and over in a meaningful way until the child shows that he/she understands. The child is then encouraged to use these words and phrases him/herself, and the parents are shown how to expand his/her language and stimulate its growth. Davis and Silverman (1978:442) stated that the more enthusiastic amongst the "auditory verbal approach" followers say: "There is no such thing as a totally deaf child" and " Every remnant of hearing is usable for developing oral communication". Even if one disagrees with this rather extreme position on the potential usefulness of residual hearing, much success has been reported by the Centre for Demographic Studies at Gallaudet College (1972-1973).

2.2.3 Oral Schools

An oral school means that children are taught primarily through listening, talking and speechreading (lip-reading). Ballantyne (1977:171) argues that lip-reading and auditory training must not be thought of as separate and distinct methods, but that each is rather complementary to the other. He says that anything which will teach a child to communicate is acceptable. Oralism is more than a method, it is rather a philosophy, an expectation that children will learn to talk and understand oral speech and that those with whom they come in contact will talk to them. Written language, charts, graphs, pictures, experiences, and group discussions are all made part of the world of learning. Oralism is, however, distinguished from manual means such as signing and/or fingerspelling.

2.2.4 Total Communication

The total communication approach to the education for the hearing handicapped is based upon the knowledge that many, if not most, prelingually deaf children do not have sufficient usable residual hearing to guarantee language acquisition solely through the auditory channel (Puren, 1985). Total communication, therefore, utilises not only all the techniques of the Oral Method, i.e. auditory training, speech reading, speech therapy, reading and writing, but adds to them a new component, language expressed through signs and fingerspelling.

Because total communication involves all methods of communication, it does not rely solely upon an impaired hearing mechanism for language acquisition. This means that if a child's residual hearing is not of sufficient quality to ensure success through the ear, the child can compensate by using the eye, just as the blind compensate for lack of vision by using their fingers to read.

For some time it was argued that permitting a deaf child to sign would destroy the incentive to master speech. Total communication requires the simultaneous use of spoken language and teachers at Nuwe Hoop Centre for the hearing impaired (Worcester) have found that in some case this has stimulated rather than inhibited speech.

Speech, speechreading, and the use of what hearing is available are other important parts of the total communication method. All of these areas, with signing and reading, are introduced at an early age so that language development will parallel as closely as possible that of the hearing child.

Caccamise (1978: 801) states: " For many years, however, commitment to the development of oral-aural communication skills in hearing-impaired persons led many professionals to reject the use of manual communication (MC) and simultaneous communication (SC). Although this rejection still persists among some people, the trend in services for hearing-impaired persons (educational,, vocational, counselling, etc.) is towards the use of MC and SC(M/SC) as part of a Total Communication Approach".

2.2.5 Schools for the Deaf

In the past, the dominant teaching philosophy in South Africa was to use total communication to educate hearing impaired and deaf children. At this time, however, some schools are encouraging the immersion of hearing impaired and deaf children into the deaf culture. Teachers are being trained in Sign Language and deaf teachers are being hired. Very little spoken English occurs in the classroom. The Deaf community is very proud of its culture and doesn't view deafness as a deficit. It is very rich in traditions, social identity and language. The Deaf community believes in accepting a hearing impaired or deaf child as he/she is and encouraging the use of Sign Language for education, vocation and improved quality of life. Children have the option of wearing hearing aids and other assistive devices.

2.2.6 Mainstreaming

Mainstreaming the hearing impaired child is most desirable and can be done to various degrees (e.g., hearing impaired class, itinerant teacher, full integration) depending on the child's ability. Manning (1987) stresses the fact that the educational and professional support must be there for the child to succeed. He pointed out that this presents a significant challenge to the teacher, especially when the classes are large and the teachers have very little, if any, direct experience with a hearing-impaired student. Flexibility in meeting the hearing impaired child's educational needs is important at each level of development.

2.2.7 Conclusion

There are many philosophies regarding the education of habilitation of hearing impaired children. This proves the diversity in viewpoints. All professionals who work with these children will have their own bias. The important point is that a program should meet the individual child's needs. If the teacher is exposed to sound experienced by these individuals then he/she will understand the specific hearing-impaired better and will be better equipped to strategize an individualised educational plan. It will be reductionistic to ignore other factors (besides the degree of hearing loss) such as the development status, age, IQ, family

support, socio-economical background, outside professional and educational support to name a few, that play a role in the development of the child.

2.3 Relevance to other disciplines

The HLS can also be used in other fields or disciplines. Some of the fields in which self-reliant research can be conducted are audiology, education, psychology or psychoacoustics and will be discussed briefly in the section to follow.

Audiology: Lately audiologists have spent much time on a test which is called 'insertion gain'. The purpose of this test is to determine if the hearing aid produces the required gain or amplification. With this test a probe is inserted between the eardrum and the hearing aid to measure the sound pressure level. A sound source generating pure sinewaves at different sound pressure levels (SPL) and frequencies corresponding to the audiogram, is placed one metre in front of the ear with the hearing aid. The gain is calculated by subtracting the SPL at the eardrum from that at the source. The calculated gain should correspond with the hearing loss specified by the audiogram. If it does not correspond, corrective action should be taken. For instance, the gain of hearing aid can be increased or a stronger hearing aid can be fitted. The distortion can also be measured when the hearing aid operates at the required level. With the hearing loss simulator the process can be reversed. The sound can be first sent through the HLS, which reduces the SPL according the audiogram of a specific hearing-impaired person. The audiologist who performs the test should sit in an anechoic chamber in which he/she is exposed to the reduced sound reproduced by the HLS. The audiologist can then fit the hearing aid of the student or patient to an earpiece of his/her own and listen for him/herself to find out whether the required gain is obtained. Also whether or not any distortion is present, can be determined by listening to the output of the hearing aid. A superior advantage of the HLS would be that with insertion gain only static waves such as pure sine waves are used in the test, whereas for the HLS any varying complex waves such as speech or music can be used. By listening to the output of the hearing aid any distortion that might arise when complex waves are processed can be detected.

Education: The techniques and algorithms being researched in this study could be of benefit to education related to Electrical Engineering (Light current). The study is particularly relevant to students studying Digital Signal Processing. The signal processing algorithms that have been proposed, studied and documented in this report can be perceived as a theoretical model of various methods of mimicing a given audio frequency response. To mimic a given frequency response is, in essence, the same as simulating a hearing loss. An essential element of a basic course in digital signal processing is the design of digital filters to meet a specific frequency response. This report deliberates on those fundamental theories and elaborates on alternative methods for filtering arbitrary shaped frequency responses.

Psychoacoustics: Davis and Silverman (1978:27) define Psychoacoustics as follows:

> " Psychoacoustics is concerned with what we hear. It describes the relations of our auditory sensations to the physical properties of the acoustic stimulus, such as its frequency spectrum, its waveform, its intensity, and its temporal relations. Psychoacoustics deals with attributes of sensation such as pitch and loudness and the apparent location of the source and also with judgements as to how loud a noise is, either relative to another noise or on an absolute scale. It is concerned with the ability of listeners to distinguish differences between stimuli. It is not concerned directly with the physiological mechanisms that underlie the detection or the differentiation of sounds but with the judgements and reports of human listeners. Most tests of hearing used to describe and measure impairments of hearing are actually psychoacoustics tests."

2.4 Audiometry

2.4.1 Introduction

The following part deals with various ways of determining the hearing disability. The hearing ability of a person as defined on the audiogram is analogous to the term frequency response of the ear (as it is used in this text). Obviously one first needs to determine the hearing loss before it can be simulated. The frequency response of the ear is, in fact, the frequency response of a filter that needs to be designed. The aim now is to find an appropriate method for establishing the frequency response of the ear. Studies of the ear's

sensitivity have led to the growth of audiometry, the science of measuring hearing. This part of the document gives only a limited introduction to the hearing and relative issues of the Audiological discipline. This part of the document will thus be divided into five main headings, namely: 1. Threshold of hearing 2. Pure tone audiometry 3. Speech audiometry 4. Auditory brain stem evoked responses 5. Other methods.

First, the threshold of hearing, which leads to the standardised pure tone audiogram, will be discussed.

Secondly, the procedures followed to draw up this audiogram are dealt with under the headings: pure tone audiometry, enviromental factors, air-conduction threshold and bone-conduction threshold

Thirdly, speech audiometry will be discussed. Speech intelligibility threshold (SIT) will then be defined, followed by speech audiometry in practice. The last two paragraphs "free-field audiometry" and "the speech audiogram" will describe the conditions under which the test should be conducted and how the speech audiogram is drawn. The fourth section on Electro-cochleography (ECG) and evoked response audiometry (ERA) will be dealt with only briefly.

The last section deals with other methods of hearing loss measurement. These are:

- The whisper and conversational speech test.
- Rubbing the fingers.
- The watch

Before the conclusion of the chapter, a survey of the various tests is given.

2.4.2 Threshold of hearing

Kinsler, Frey, Coppens and Sanders (1982:257) express the properties of the human ear as phenomenal and show that it is capable of responding over a frequency range of approximately 20 Hz to 20 kHz. The human ear is not equally sensitive to all frequencies. It is only at frequencies near 1000 Hz that sounds are just audible (to the normal ear) when they possess a sound pressure of 2×10^{-5} N/m^2 or 20 μPa. For this reason the sound pressure of 20μPa has been chosen internationally as the 0 dB level (Tuner & Pretlove, 1991). At higher and lower frequencies, the sound will normally need a considerably higher pressure in order to be audible to the normal ear. For example, a tone of 200 Hz is barely

audible at a sound pressure of 200 μPa; it is 20 log $\frac{0.002}{0.0002}$ dB = 20 dB as compared with the internationally laid down 0 dB level. The audible threshold for many frequencies has been determined in this way and the results were plotted on a graph called the "Fletcher-Munson" curve. This graph, which indicates the "normal threshold of hearing", is shown in figure 2.1. When investigating a hearing impairment the first point of interest is how the position and form of the threshold of hearing have altered as a result of this impairment. It is convenient to be able to express the magnitude of this change as a number, or numbers. Ligtenberg (1982) explains that it is natural to try to do this on the basis of a comparison of the threshold of the defective ear (the pathological threshold) with that of a normal ear (the normal or physiological threshold).

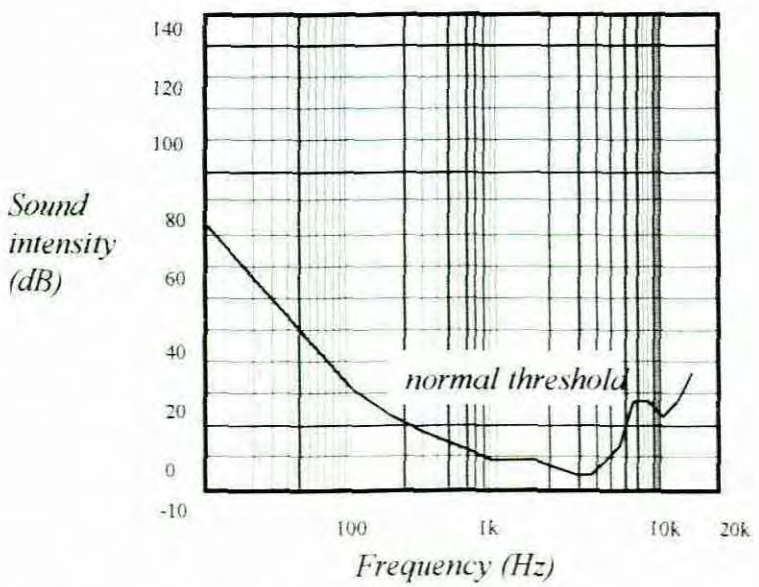


Figure 2.1 Normal hearing threshold

Suppose the pathological threshold has been determined in some suitable way, results could then be plotted as shown in the figure 2.2. The lower line represents the threshold of hearing of the normal ear, and the upper line, that of the deaf ear in question. The area between the two curves is therefore a measure of the loss of hearing.

This method of representation is not particularly easy to use in practice, especially when different measurements have to be compared with one another: the curve form of the threshold makes it an unwieldy standard of comparison. This difficulty has been solved in practice by "straightening" the threshold of hearing of the normal ear and altering the dB

scale correspondingly. The resulting graph is called an audiogram as shown in figure 2.3. The audiogram is thus a particular way of plotting the pathological threshold of hearing against the frequency. It can be seen on figure 2.3 that the obtained curve shows the loss of hearing while the normal curve lies on top of the figure. The worse the hearing, the lower the corresponding audiogram's curve.

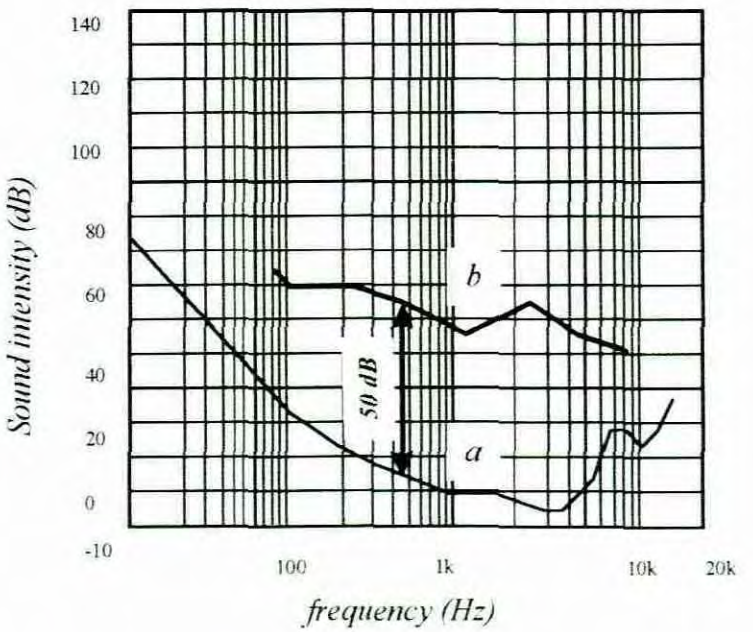


Figure 2.2 The normal threshold of hearing (a) and a pathological threshold (b)

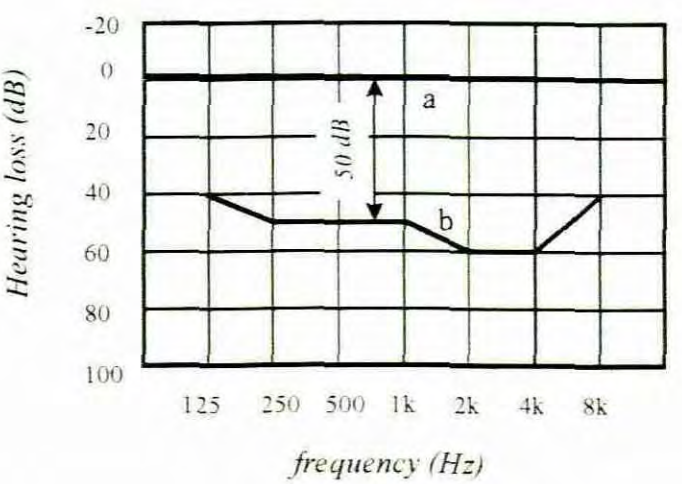


Figure 2.3 Audiogram shows that the same pathological threshold (b) is now related to the "straightened" normal curve (a)

2.4.3 Pure tone audiometry

2.4.3.1 Introduction

Measuring the response of the ear at different frequencies was initially done by the use of tuning forks. A series of tuning forks at standard frequencies 125, 250, 500, 1000, 2000 and 4000 Hz were used. The tuning-fork test has disadvantages such as :

a. The loudness of the sound is not constant, since this depends on how hard the tuning fork is struck.

b. The test tone provided by the tuning fork is subject to damping.

c. The rate of damping of the sound (decay time) is different for tuning forks of different frequencies.

The need for a means of testing hearing acuity which did not suffer from the disadvantages inherent to the tuning fork was felt for a long time. Such a means became available, thanks to the development of electronics.

2.4.3.2 Audiometer

Dale (1962) defines an audiometer as: " An instrument designed to measure human hearing for pure tones. Usually audiometers determine the threshold of audibility of a subject at each octave". Pure tones in this text would refer to pure sine waves. These tones are made audible by high-quality headphones (air conduction). The loudness of the sound produced can be accurately regulated by the person operating the audiometer. A special vibrator (the bone conductor) can also be fitted to the audiometer for measurement of the bone conduction (the vibrations are then transmitted directly to the skull). Some audiometers are fitted with various auxiliary devices for making special tests of hearing, apart from measuring acuity. However, all makes of audiometers are fitted with three basic controls, which allow the determination of the threshold of hearing.

These are:

a. Frequency-selector switch. This is used to select the desired frequency. The frequencies available are usually 250 - 500 - 1000 - 2000 - 4000 and 8000 Hz, by analogy with tuning-fork series. Audiometry done with this frequency series is sometimes called octave audiometry, because each tone is one octave higher than the preceding one.

Sometimes the somewhat differing series of 128 – 256 – 512 – 1024 - 4096 and 8192 is used. Some elaborate types are provided with a continuous variable frequency knob, by which any frequency in the total range may be chosen.

b. Volume control (or attenuator). This regulates the level of sound produced, usually in steps of 5 dB from e.g. 0dB to 100dB.

c. Interruptor. This switch controls the duration of the tone.

2.4.3.3 Enviromental factors

Hearing tests should ideally be conducted in a sound proof room away from street noises or other disturbances, since extraneous noise will mask out audiometer signals, so they cannot be heard or recognised by the subject. The result is an inaccurate or unreliable hearing test.

2.4.3.4 Air-conduction threshold

The test is started by placing the headphones on the patient's/student's head. Next, the person is presented during a short time with a tone of 100 Hz to one ear and then the amplitude is continuously increased until it is clear that he/she has heard it. After the tone is interrupted the sound level is reduced by 10 dB and the tone stimuli repeated again. If this is heard too, obviously the sound level is reduced another 10 dB until he/she shows signs of not being sure whether the tone was audible or not. The sound level should now be reduced in steps of 5 dB, and the duration of the tone extended to a few seconds. The softest tone that is still audible in this way forms a point on the pathological threshold. The airconduction threshold is determined at the following frequencies in the order stated: 1000 - 500 - 250 - 2000 - 4000 and 8000 Hz. The technique used at each frequency is the same as that described above.

Each time the lowest audible sound level at a given frequency is determined, the value should be plotted on the audiogram. The following signs are internationally accepted for this purpose:

Left ear: air conduction X, bone conduction].

Right ear: air conduction O, bone conduction [.

2.4.3.5 Bone-conduction threshold

Measurement of the air-conduction threshold naturally gives no information about the site of the defect in the ear, i.e. whether we are dealing with middle-ear deafness or perception deafness involving the cochlea and/or the auditory nerve. In order to distinguish between conduction deafness and perception deafness, the bone-conduction threshold is always measured too. The headphones are now replaced by the bone conductor, which is placed behind the ear on the mastoid. The audiometer is switched from "air" to "bone". The same procedure is then followed as for the determination of the air-conduction threshold. These measurements give information on the behaviour of the cochlea without the middle ear, since the sound is conveyed directly to the cochlea via the skull. The audiogram is arranged so that the bone-conduction threshold of the normal ear coincides with the air-conduction threshold and there is, therefore, no difficulty in plotting the results. For children, where the air conduction is much less then the bone conduction, even after medical treatment, the use of hearing aids fitted with bone conductors is encouraged.

2.4.4 Speech audiometry

2.4.4.1 Introduction

In speech audiometry the subject under test is presented with a series of words. Ligtenberg (1982) describes speech audiometry in a book "Basics of audiology" and this section on speech audiometry is an extract of her work. She explains that a list of one- and two-syllable words, in which the phonemes (sound elements of a language) occur in the same percentages as is in the normal spoken or written language, are used. These lists of spondees (two-syllable words) and monosyllables are said to be "Phonetically Balanced"(PB lists). Tape recordings of PB lists read by trained speakers are available for test purposes.

2.4.4.2 Speech intelligibility threshold (SIT)

Investigations have been made with the aid of these PB lists into the relationship between the percentage speech intelligibility for normal persons and the sound pressure of the words

presented. The results of these measurements are plotted in figure 2.4. It was found that as 10 dB above the sound pressure of 20 μPa, 50% of the words presented were correctly repeated. This threshold is known as the speech intelligibility threshold (SIT). The speech intelligibility threshold thus lies about 25 dB above the pure-tone threshold. The above should help to clarify the reason for the correction factor of 25 dB.

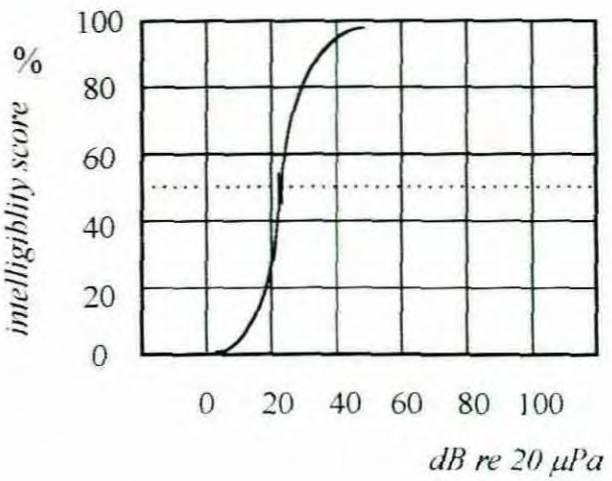


Figure 2.4 Speech intelligibility threshold (SIT) diagram.

People suffering from defective hearing produce speech intelligibility curves which are shifted and/or distorted. A simple way to evaluate these curves is to determine how far the 50% level found for the person in question is shifted with respect to the 50% level for the normal ear. The 50% level for a normal ear is now taken as the zero point.

2.4.4.3 Speech audiometry in practice

An ideal speech audiometry set-up is constructed as follows (see figure 2.5). A number of the PB lists of ten words each are recorded beforehand on a tape recorder. The reader used for this purpose should have a good speaking voice, and should pronounce the words evenly. The recorder is connected to the audiometer; the loudness of the words can then be adjusted by means of the attenuator on the audiometer. The recorder and the audiometer are adjusted so that when the attenuator is set at 0 dB, a hearing person can understand exactly 50% of the words. The investigator and the person on whom the test is performed simultaneously hear the words from the recorder (but the investigator always hears them loudly enough for him/her to understand them).

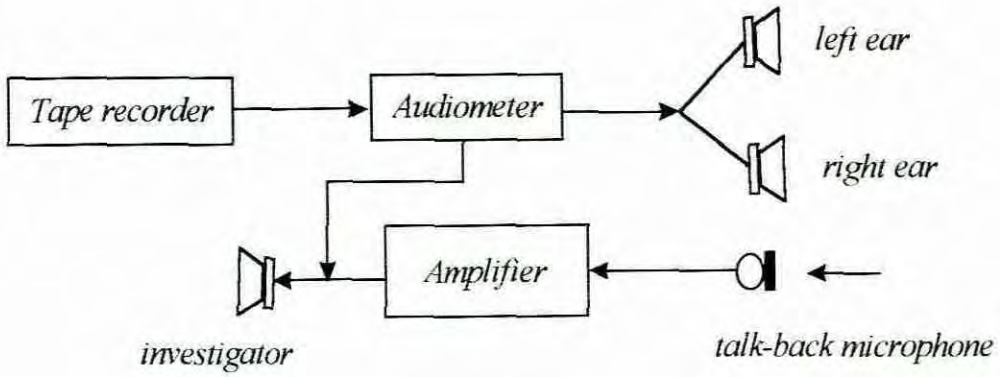


Figure 2.5 Speech audiometry set-up

If the person repeats the word he/she thinks he/she has heard, the investigator also hears this (if necessary, via a microphone and an amplifier). The investigator can thus easily check how many of the words are repeated correctly.

2.4.4.4 Free-field audiometry

Speech-audiometry is often carried out as a free-field method, i.e., instead of the headphones one or more loudspeakers are used. These methods allow for making audiograms with and without a hearing aid, so that the exact improvement by the hearing aid can be determined.

2.4.4.5 The speech audiogram

A word list is first presented at quite a high sound level, so that the subject being tested can hear all the words clearly. Other word lists are then played, each one 10 dB quieter. The number of words repeated correctly from each list is noted and plotted in the speech audiogram. The speech reception threshold (SRT) is compared with the average of the pure tone threshold at 500, 1000, 2000 Hz in order to check the reliability of the responses. Suppose that in a given case the graph shown in figure 2.6 were obtained. Two interesting conclusions may be drawn from this graph. In the first place, we see that the 50% intelligibility level is shifted 60 dB with respect to the speech intelligibility threshold of the normal ear. It is thus highly probable that the pure-tone threshold will also be shifted 60 dB with respect to the physiological threshold.

In the second place, we may see from the graph that, no matter how loudly the words are presented, 100% intelligibility is never reached.

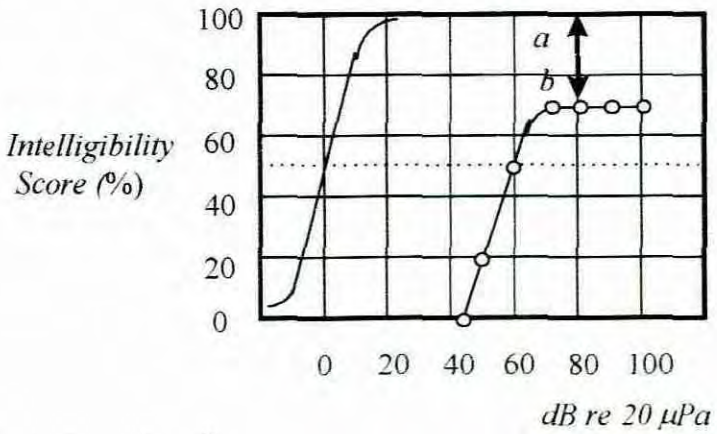


Figure 2.6: Speech audiogram

It would seem that the person is unable to distinguish all the words properly, as the result of a frequency-analysis disturbance. The deafness is therefore not, or not entirely, due to causes in the middle ear, but completely or partly to some fault in the perceptive part of the ear. The distance a-b is a measure of the extent of the frequency-analysis disturbance. In this connection we speak of a loss of discrimination. In the example shown here, this loss is 100 - 70 = 30%. Still more interesting conclusions can be drawn from this method of speech audiometry, but a discussion of them falls outside the scope of this research.

2.4.5 Auditory brain stem evoked responses (ABR)

It would be most convenient if it were possible to produce an adequate measure of a child's hearing loss by some means not dependent upon the child's co-operation and voluntary responses. Newby (1958:185) mentions that in recent years there have been attempts to devise tests that would depend on neurophysiological clues rather than upon voluntary response of the patient. One very popular method is called "Electroencephalography" (EEG).

The electroencephalograph measures the brain wave activities when audio stimuli are applied. The best results are obtained when the patient is in a light sleep. This method is sometimes referred to as an "objective" test as the procedures are said to be scientifically

precise and independent of human judgement. These tests are often used if the normal reactions of the child to acoustical signals are not reliable for some reason or other. The difficulty with this technique as Newby (1958:186) pointed out, is that: " The child must be in a light sleep, which is sometimes difficult to induce. Also, expensive and complicated equipment is required. It is most difficult to interpret variations in brain-wave activities which might be due to a response to sound stimuli or might be due to another kind of stimulus altogether.

2.4.6 Other methods of hearing loss measurements

2.4.6.1. The whisper and conversational speech test.

There are various ways to test hearing loss. The simplest one is undoubtedly to see how well a person hears whispered speech and normal conversational speech. Despite its simplicity, this test must conform to certain rules such as : 1. Quiet room, 2. Uniform loudness of test words, 3. disable lip-reading if it is to give reliable results. If these rules are observed it will not be difficult to estimate the patient's loss of hearing with the aid of the following data:

	at the ear	30 cm	100 cm	300 cm
whisper	75	55	45	35 dB
conversational speech	90	70	60	50 dB

With this method one should bear in mind that the threshold of understanding is about 25 dB's higher (Ligtenberg, 1982). This means that if the person hears a whisper at a distance of 30cm (55dB) and can repeat the words properly, his hearing loss is about 55 - 25 = 30 dB.

2.4.6.2. Rubbing the fingers.

If one suspects that the patient may have a loss for high tones, the hearing acuity can be roughly estimated by rubbing the thumb and the index finger over one another just beside the ear. The level of this sound is about 50 dB, and it contains a large proportion of high

tones. If the person does not hear this, he must therefore have a hearing loss of at least 50 dB for high frequencies.

2.4.6.3. The watch

The watch is also a simple but effective aid. The ticking of a (non-electronic) watch held just in front of the auditory canal (but not touching the head) has a level of about 40dB, and the sound contains a large proportion of high frequencies above 1500Hz. It will be clear that this can be used as a simple though crude test for slight high-tone deafness.

2.4.7 Survey of the various tests

In the previous paragraphs a number of methods were described of investigating people with a hearing defect. This description makes no claim at all to completeness. The test methods described have been found to give a reasonable idea of the loss of hearing when used by an experienced investigator.

Pure-tone audiometry was described first. This method gives an insight into the magnitude and sometimes the nature of the hearing loss. It allows the hearing loss to be measured quite accurately. When this method is used, registration through the ear which is not being tested, can sometimes cause difficulties during bone conduction measurements; in such cases masking should be applied. Next, a test was described which could be used to complement pure-tone audiometry and which gives a better insight into the nature of the hearing defect: namely speech audiometry. It consists of measuring the discrimination loss by determining the percentage intelligibility of phonetically balanced words spoken at a comfortable level over the audiometer. Finally, objective tests were briefly described which permit the condition of the ears to be checked without any cooperation from the patient. An additional test described is the whisper and conversational speech test. This test is a simplified version of the "ideal" method of speech audiometry. The value of this group of tests largely depends on the investigator's experience. Rubbing the fingers and listening to the watch, the simplest, were the last tests discussed for the sake of completeness.

2.4.8 Conclusion

From the various tests used to measure the response of the impaired ear, the best method needs to be identified for practical implementation. The most objective method thus far is the auditory brain stem evoked response (ABR) method. Apart from being accurate in measuring the sound intensities at various frequencies, it is also very reliable provided the relevant information is drawn from measured bio-electrical activities emanating from the brain. This method could only be carried out under clinical conditions. It could be very expensive as a light sleep needs to be induced by means of narcotics. It is also time consuming, which adds to the expense. Other difficulties are the interpretation of the results. Speech audiometry is an excellent method of determining the level of discrimination. It gives a rough estimate of the frequency response at the following frequencies: 500, 1000, and 2000. It should, however, correlate to some extent with the frequency response measured by means of pure tone audiometry.

A pure tone test can be done within 10 minutes with less expensive equipment and procedures than those needed for the ABR. From the methods thus investigated it could be concluded that the best practical method for determining the frequency response for an application to simulate the hearing loss is the use of the pure tone audiometry test. It should be noted that this method is subjective, in the sense that the patient/student and the investigator establish points on the audiogram based on some form of communication between them. The possibility of misunderstanding is always present in human communication. Accuracy and reliability is increased if the test is repeated and the same results are obtained. The older the patient, the more accurate this will be, since he/she will be more likely to understand the instructions and to react reliably to the sound stimuli.

The final conclusion would be that all the various methods could be used. When the pure tone audiogram is used as the developmental instrument, the information from the various tests could be manually transferred from their individual graphs to the audiogram. Having an audiogram with the two axes, namely frequency versus dB SPL, digital signal processing algorithms could then be implemented to do the required filtering, thereby simulating the measured hearing loss. In the next section methods of using the audiogram's information

are used to perform the required attenuation are discussed. At this stage it should be noted that when DSP techniques are used to perform pure tone audiometrical tests, multiplying the sinewave values with the corresponding numerical values in a lookup table produces the actual sound pressure levels. Each frequency axis on the audiogram has its own lookup table. For further detail refer to the report on the design and development of an "Audiometer" by Thys (1996). These corresponding numerical values that are the SPL equivalents (in dB's) at different points on the audiogram, are used as the coefficients of the arbitrary shape filter. This is obviously a representation in the frequency domain. The research focuses on investigating methods of using these coefficients can be utilised to perform the required attenuation. A literature review of techniques studied to do the arbitrary shape filtering is briefly discussed in the next section. These techniques or methods form, in fact, the theoretical solution to the problem and each method is discussed in further detail in part two of this document.

2.5 A review on related literature

2.5.1 Introduction

As this is a new concept, no literature currently exists on exactly the same topic. The simulation of hearing is, however, a basic filtering process. An enormous amount of literature on digital filtering is widely available. The techniques investigated to realise HLS are listed below. An overview of each technique then follows.

i. Filter method using a bank of filters.
ii. Inverse Fourier transform.
iii. Inverse Chirp Z transforms.
iv. Wavelets.
v. ARMA filter design by Yule-Walkers method.

2.5.2 Filter method using a bank of filters

Simulating hearing loss closely relates to Digital Equalisation. The digital equaliser is an "active filter bank with amplitudes adjusted to shape the transfer function over a series of frequency bands" (Higgins, 1990:38). The coefficients for the entire filter bank are stored in

memory and then read out dynamically to reconfigure the processor as the signal is switched through one filter after the other – all with the same signal processor. Either Infinite Impulse Response (IIR) or Finite Impulse Response (FIR) filters can be used. Filter banks making use of FIR or IIR filters can be designed with a package called "Signal Processing Toolbox for Matlab" (Krauss, Shure & Little, 1994:43-48). The IIR filter will require relatively few terms to obtain an acceptable attenuation characteristic outside the pass-band but will have a non-zero phase characteristic, which may be unacceptable in some applications. The FIR filter has excellent phase characteristics but requires a relatively large number of terms to obtain an acceptable attenuation characteristic (Martin, 1991). If sufficient memory is available, then this filter might be easier to implement.

2.5.3 Inverse Fourier transform

The simulation process can be performed within the time or the frequency domain. The input signal and the measured frequency response have to be in the same domain. These two domains are reversible. Inverse Fourier transforms are used to convert from the frequency domain to time domain (Kuc, 1988). With this method the Audiogram's information, which is in fact the frequency response of the ear, can be converted to the time domain. The coefficients thus generated will be used in the convolution process that performs the filtering

2.5.4 Inverse Chirp Z transforms

The "Discrete Fourier transform " equals the z transform of the signal evaluated at equally spaced points around the unit circle (Kuc, 1992: 288-293). The response characteristics of the human ear are not linear. The "Chirp z transform" can be evaluated at certain general contours in the z plane (Krauss, Shure & Little, 1994:1-56). Using the inverse transformations for this application means that frequency points can be taken that are denser at certain points than at others. The effect of this will be that the whole operation can concentrate on frequency areas that are more critical to the ear.

2.5.5 Wavelet approach / sub band decomposition

Subband decomposition decomposes the signal into sub bands so that the filtering can be done by multiplication in the frequency domain. The editorial staff of Analog Devices (1995) describes wavelets as alternatives to the Fourier transforms. They say that instead of transforming a pure "time description" into a pure "frequency description", the new methods find a good compromise -- a time-frequency description. Wavelets are a new and promising tool for realising HLS.

2.5.6 ARMA filter design by Yule-Walkers method

This technique is based on the estimation of the autoregression (AR) and moving average (MA) parameters in two separate steps. First the AR parameters are obtained as the solution of the so-called modified Yule-Walker (MYW) equations, then the MA parameters are subsequently estimated, as will be discussed later. According to Matlab the Yulewalk function performs a least squares fit in the time domain. The time domain behaviour of the ARMA filter can be formalised as a digital filter that consists of a FIR filter with the MA part in series with a IIR filter with the AR part. These filters are respectively responsible for the positioning of the poles and zeroes in the frequency domain. The ARMA filter can be implemented from the general difference equation:

$$y(n) = \sum_{k=1}^{M} a_k y(n-k) + \sum_{k=-N_f}^{N_f} b_k x(n-k)$$

where $x(n)$ is the input signal, $y(n)$ is the output signal and a_k and b_k are the coefficients of the IIR and FIR filters respectively.

2.6 Conclusion

The first two chapters, which form part I of this document are an introduction and provide the background for this study. Part II provides potential theoretical solutions. In part II, which now follows, the above mentioned techniques are discussed in further detail. The best method is then selected and implemented.

Part II

Theoretical solutions

CHAPTER 3

FILTERBANK METHOD

3.1 Introduction

One of the methods for realizing Hearing Loss Simulation (HLS) is to use a filterbank. A filterbank uses a series of Bandpass filters with amplitude adjusted to shape the transfer function over a series of frequency bands. Higgins (1990:48) explains the operations of the filterbank as follows:

" The coefficients of the entire filterbank are stored in memory and then read out dynamically to reconfigure the processor as the signal is "switched" through one filter after the other, all with the same processor. The concept is illustrated in figure 3.1. A portion of the signal is stored in a "circular" buffer, continuously being updated, with older data thrown away. The coefficients are loaded, and the filter operates on the buffer and accumulates the result in a second buffer. The next set of filter coefficients is loaded and the process is repeated."

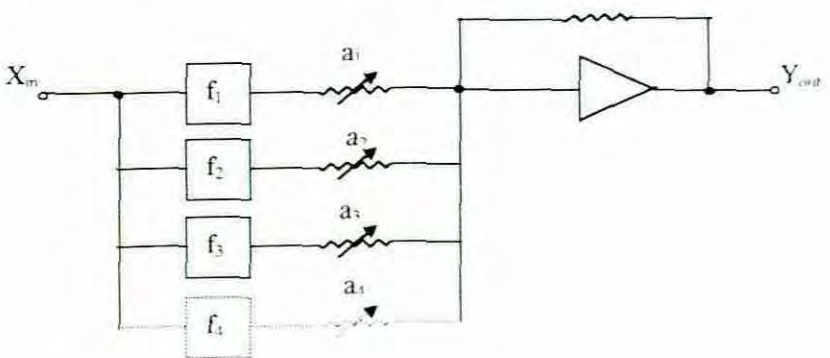


Figure 3.1 (a) Analog block diagram of bandpass filter

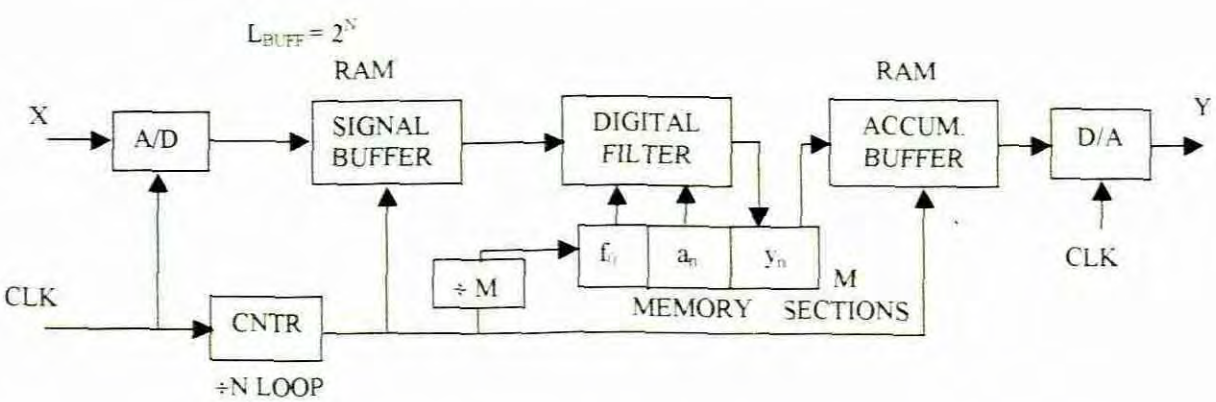


Figure 3.1 (b) Concept of time –multiplexing to replace many identical filters by a single dynamically programmed digital filter. Blocks shown as hardware (e.g., counters) may be replaced by software loops in a program.

The number of filter bands that can be used in this arrangement depends on the speed of the processor and the type of filters. These filters should have centre frequencies around the specified frequencies of the audiogram e.g. 250 Hz, 500 Hz, 1 kHz, 2 kHz, 4 kHz and 8 kHz. The dotted line in figure 3.2 defines these bandpass filters with cut-off frequencies.

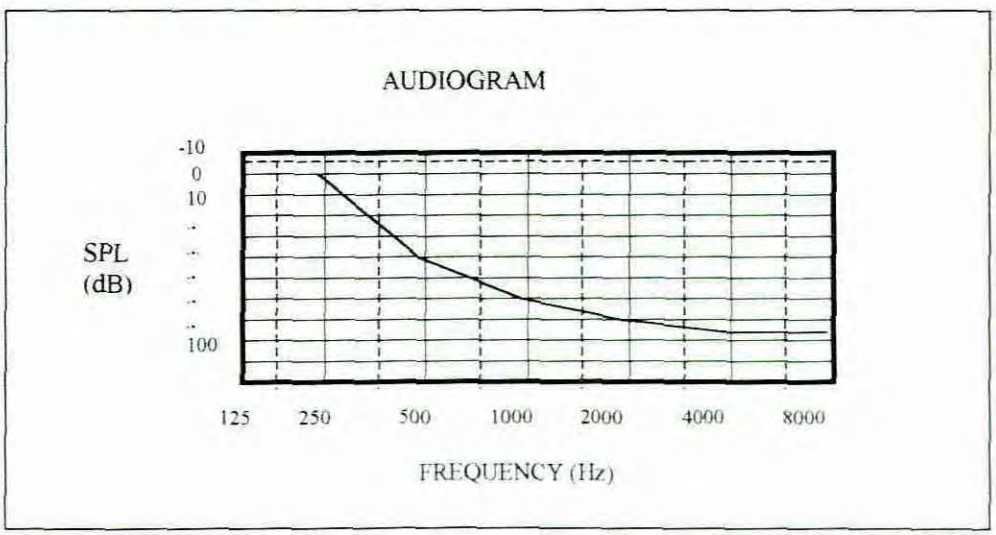
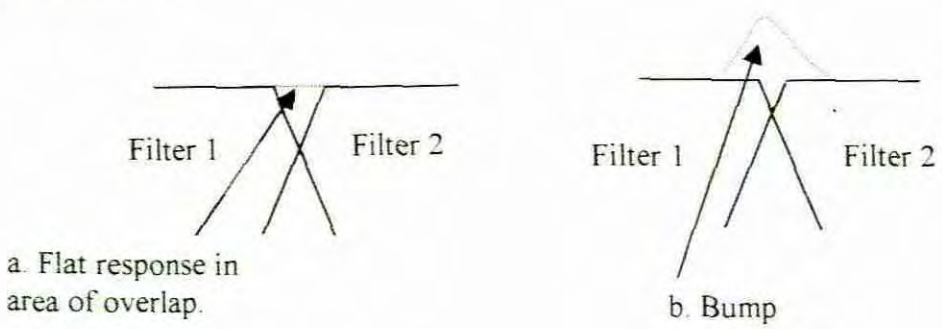


Figure 3.2 Audiogram with cut-off frequencies of bandpass filters

The perfect solution to the problem of designing an HLS with the filterbank method would be the use of ideal brickwall filters. This would ensure that the signal is split into well-defined frequency channels. As is known, an ideal filter is never possible. It is shown in figure 3.3 below that an overlap will occur at adjacent bands. These overlaps would be a source of distortion which would be either difficult or impossible to eliminate. This distortion is the result of the two filters next to one another, which traps the same portion of the signal spectrum.



a. Flat response in area of overlap.

b. Bump

Figure 3.3 Filter overlap

When the signal is reconstructed by addition, these corresponding values are added, which may gives a result bigger or smaller than that which would have resulted if the two filters had not overlap. Figure 3.3 b shows a bump due to the addition of the corresponding values. This is called amplitude distortion.

The main task in using a filterbank to realize HLS is to find filters as close as possible to the brickwall scenario.

Two most common digtal filter topologies that can be used are:

- Finite impulse response (FIR)
- Infinite impulse response (IIR)

Each of these types of filters has its advantages and disadvantages, which will be discussed in detail in chapter 9.

Briefly, the IIR can have a higher filter order with fewer filter coefficients in comparison with the FIR. This is the main advantage, which makes the IIR type of filter a better choice for the design of the required bandpass filters. For this reason, the research is limited to the implementation of the IIR filters when considering the filterbank method. The FIR filters utilize Fourier transforms and are dealt with in the next chapter.

3.2 *Filterbank using IIR filters*

3.2.1 Introduction

The following section deals with the classical IIR filter design through analog prototyping. Infinite-Impulse Response (IIR) filter design procedures are extensions of those originally developed for analog filters and start with the design of the appropriate analog filter in the analog frequency domain. IIR digital filters are commonly used to replace existing analog filters. The complex-valued Laplace variable, $s = \sigma + j\Omega$, plays the same role in the analysis of analog filters as the variable z for discrete-time filters. The s plane singularities of the analog design can be translated to the z plane by means of bilinear transformations. The required bandpass filter can be obtained by first constructing a lowpass filter and then converting it to a bandpass filter by means of frequency transformations. The most commonly used analog filter synthesis techniques for designing a lowpass filter in the s plane are the Butterworth, Chebyshev, Elliptic and Bessel techniques. These are described and their advantages and disadvantages are compared.

A comparison of classical IIR filters can be simulated by means of computer aided designs (CAD), for which purpose Matlab is ideal, as it contains a special DSP filter toolbox.
This toolbox provides five different types of classical IIR filters, each optimal in some way. This section shows the basic analog prototype form for each, and summarizes major characteristics. As this research focuses on finding the best method, the detail of the design procedures is not documented. Rather, the comparison is emphasized. A complete design procedure reference can be found in Ludeman (1994:120).

3.2.2 Butterworth filter

Provides the best approximation to the ideal lowpass filter response at $\Omega = 0$ and $\Omega = \infty$; for any order N, the magnitude squared response has 2N –1 zero derivatives at these locations (maximally flat at $\Omega = 0$ and $\Omega = \infty$). Response is monotonic overall, decreasing smoothly from $\Omega = 0$ to $\Omega = \infty$. Figure 3.4 shows that as n (the order) of the filter increases, $|H_n(j\Omega)|^2$ approaches an ideal low pass frequency response.

$|H(j\Omega)| = \sqrt{\frac{1}{2}}$ at $\Omega = \Omega_c = 1$ rad/sec (for filters that are not scaled)

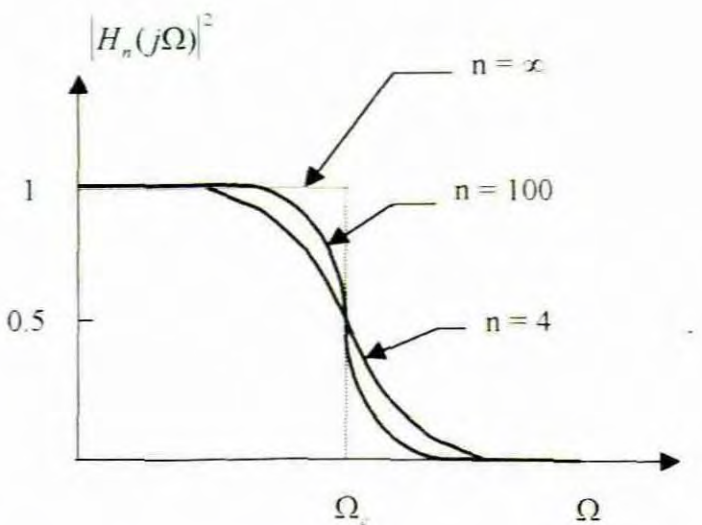


Figure 3.4 The magnitude square frequency response of a Butterworth filter

3.2.3 Chebyshev filters

There are two types of Chebyshev filters, Chebyshev type I and Chebyshev II.

- Chebyshev I

The Chebyshev I contains a ripple in the passband and is maximally flat in the stopband as illustrated in figure 3.5 below. There is less difference between the ideal and the actual frequency response of the Chebyshev than there is in that of the Butterworth. The transition from passband to stopband is also more rapid in the case of the Chebyshev type I than for the Butterworth filter.

- Chebyshev type II

Chebyshev type II, on the other hand, contains a ripple in the stopband and is maximally flat in the passband. It better approximates the ideal filter better over the entire stopband. The stopband does not approach zero as quickly as the Type I filter. Figure 3.6 shows the Chebyshev Type II

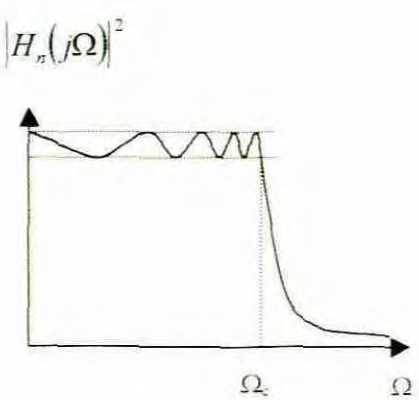


Figure 3. 5 Magnitude square frequency response of the normalized type I Chebyshev filter

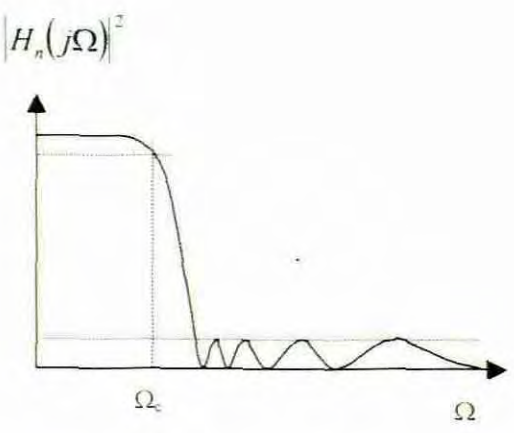


Figure 3. 6 Magnitude square frequency response of the normalized type II Chebyshev filter

3.2.4 Elliptic filter

The Elliptic filter has equal ripple in both passband and stopband. These filters have the minimum transition width. It generally meets filter requirements with the lowest order of any other filter type. Figure 3.7 shows the steep transition.

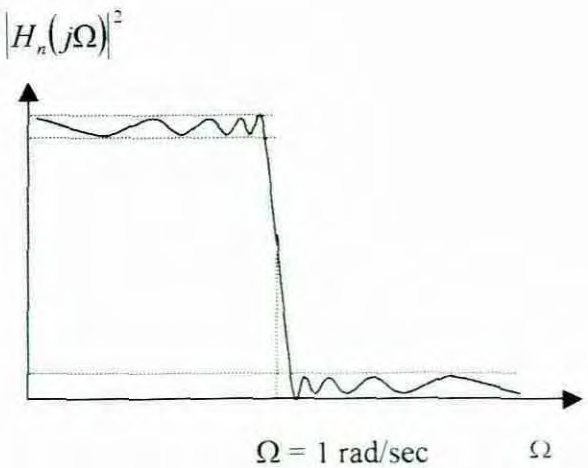


Figure 3.7 Magnitude frequency response of a normalized low-pass elliptic filter.

3.2.5 Bessel filter

The Bessel filter has maximally flat time delay within the passband. This results in filtered signals that maintain their waveshapes in the passband frequency range. The price to pay for this advantage is an amplitude response with steepness less than that of the Butterworth. It therefore requires a higher filter order than other filters.

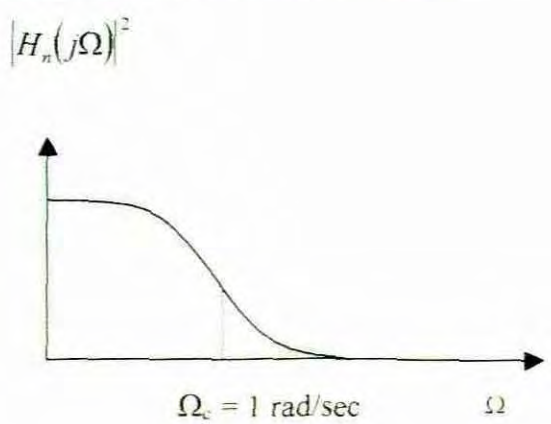


Figure 3.8 Bessel filter frequency response

3.2.6 Filter comparison

The Bessel filter's transient response has vastly superior properties in the time domain, as compared with the Butterworth and Chebyshev. The Chebyshev, with its highly desirable amplitude-versus-frequency characteristics, actually has the poorest time-domain performance of the three. The Butterworth is between the frequency and time-domain properties (Horowitz and Hill, 1995). The elliptic filter has the minimum transition width, which will have the minimum distortion when considering the problem caused by the overlap of adjacent filters.

3.3 Interactive filter design by intuitive Pole/zero placement

A filter structure and its coefficients to meet a desired magnitude response can be determined by placing poles and zeros in the z plane. Although it can also be formally done, poles and zeros are placed in the z plane by intuition, which sets the dips and the peaks of the required magnitude response. The filter design is accomplished interactively by successively adding real and complex conjugate singularities in the z plane and observing whether the magnitude of the resulting filter is approaching the desired magnitude specifications. A benefit of the interactive procedure is that the filter magnitude response resulting from the conventional design approach can often be improved by proper placement of additional poles and zeros.

3.4 Filterbank using Finite Impulse Response (FIR) filters

The central concept in the algorithm of FIR bandpass filter design is the use of inverse Fourier transforms. The coefficients of the FIR bandpass filter are obtained by calculating the inverse Fourier transform on the ideal frequency response and then multipling by an appropriate window. It was found that the same method could be used for filters with arbitrary shaped frequency responses. These will be discussed in detail in the next chapter.

3.5 Drawbacks of filterbank method

This investigation has shown that there are two major drawbacks when a filterbank method is used for the simulation of hearing loss.

I. The first problem is the overlap of the filterbanks, which might cause distortion. Choosing one of the filters previously discussed with the steepest transition could reduce this. The trade-off of a steeper transition is a ripple in the passband. When a filter with less ripple is chosen for better sound reproduction, the order of the filter has to be increased dramatically to reduce the distortion. This will put serious demand on the memory requirements.

II. The next drawback relates to the frequency resolution. A typical audiogram is taken as an example to illustrate this problem. On Figure 3.8 the circles show the points on the audiogram that were established during an audio test. These points exist on specific standard frequencies as discussed earlier. Referring to figure 3.2, it can be deduced that seven filters need to be constructed. The first filter will obviously need to be a lowpass filter and the last (the seventh) a highpass filter. The filters in-between will thus be bandpass filters. The first lowpass filter will have cut-off frequency at 187.5 Hz. The second filter will have a cut-off frequencies at 187.5 and 375 Hz.. The third will have a passband between 375 Hz and 750 Hz. The fourth filter will have cut-off frequencies at 750 Hz and 1.5 kHz. The Fifth bandpass filter will have cut-off frequencies at 1.5 kHz and 3 kHz..

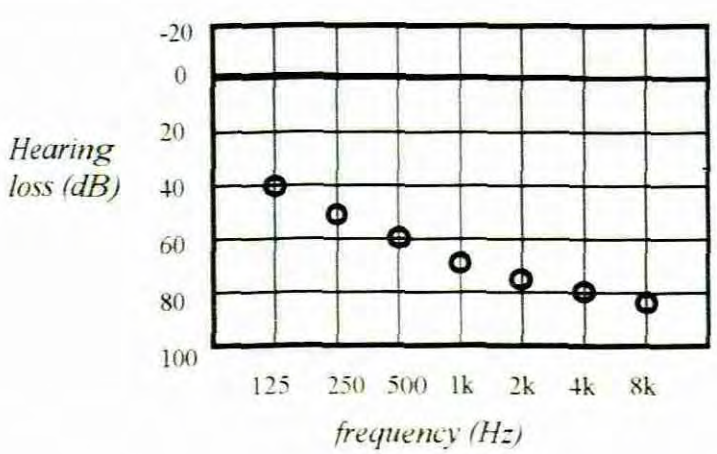


Figure 3.9 Example of test results on audiogram

The sixth bandpass filter will have cut-off frequencies at 3 kHz and 6 kHz. The last filter, a highpass filter, will have a passband starting at 6 kHz.

When applying the bandpass filter method on the given example, frequency responses will result as drawn in figure 3.10. It can be seen that all frequencies in a specific band would be reduced to the same sound pressure level. Obviously this stepped response is not the same as the response of the ear.

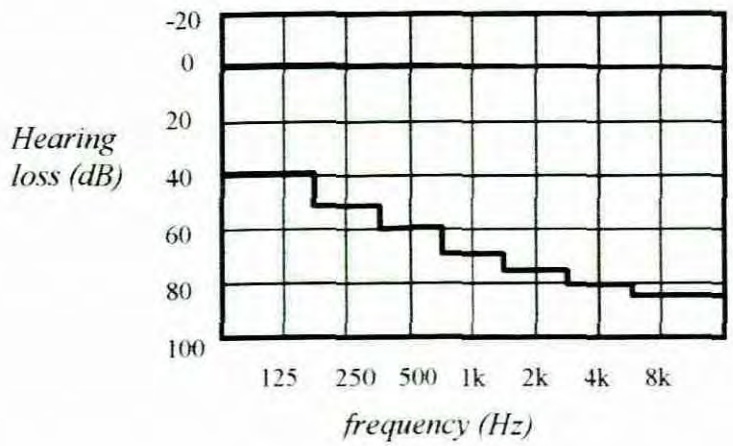


Figure 3.10 Stepped response of filterbank method

The sensitivity to sound pressure levels of the ear would, instead, change gradually over the audio frequency range. The simulation of hearing loss with this method would thus not be completely accurate in that only those frequencies (the centre frequencies of the bandpass filters) at which the test was conducted would be accurately represented. An improvement would be to increase the number of passband filters. Some form of interpolation could be used to calculate the level of attenuation of these additional filters. This, on the other hand, will have an increasing demand on memory, which is not always favourable.

3.6 Summary

This chapter has dealt with the filterbank method to realize hearing loss simulation. The filterbank constitutes a series of bandpass filters. Bandpass filters could be constructed by either Infinite impulse response (IIR) or Finite impulse response (FIR). The advantages and disadvantages of both IIR and FIR have been discussed. Emphasis was placed on the one-radian low-pass filter. Other non-normalized filters could be derived by transformational

methods. An excellent reference on transformation methods can be found in Ludeman (1986). The properties of various IIR filters such as Butterworth, Chebyshev, Elliptic and Bessel filters were briefly discussed with the purpose of pointing out their respective advantages and disadvantages. It was also pointed out that FIR Bandpass filters could be designed by doing inverse Fourier transforms (IFT) on the ideal frequency response. The discovery of the drawbacks of the filterbank method has led to the investigation of the next method, which is the design of filters with arbitrary shape. This method also utilizes IFT and will be discussed in the next chapter.

3.7 Conclusion

The investigation of the filterbank method for simulating hearing loss has revealed two serious disadvantages. The first drawback is the distortion due to overlap of the bandpass filters. By selecting one of the aforementioned filters with the steepest transient, the distortion can be reduced. This, however will result in a ripple in the frequency domain. The second disadvantage is that the filterbank method will introduce a stepped response corresponding to the number of bandpass filters. This response does not resemble the properties of the ear, which perceives rather a gradual change over the audio frequency range. These disadvantages of the filterbank method have urged the research of a method that approximates the properties of the ear more accurately. A method needs to be introduced which works with interpolated points between those at which the tests were done. This newly shaped curve (after interpolation) will more closely approximate the properties of the ear. The next chapter deals with arbitrary shape filters, which provides a better solution to the problem.

CHAPTER 4

INVERSE DISCRETE FOURIER TRANSFORM METHOD

4.1 Introduction

In the previous chapter two problems with the Filterbank method were pointed out. The first problem is the overlap of adjacent bands, which might be a source of distortion. The second problem relates to the accuracy of the response curve (step response) resulting from the limitation on practical, implementable filterbanks. A solution to these problems would be to use the inverse Fourier transform method. A brief overview of the method will be discussed as an introduction, while further detail is explained in the rest of the chapter. The algorithm of the method is as follows: The audiogram information at the seven test points is interpolated over the entire audio frequency spectrum. This gives a smooth frequency response onto a dense, evenly spaced grid as illustrated in figure 4.1

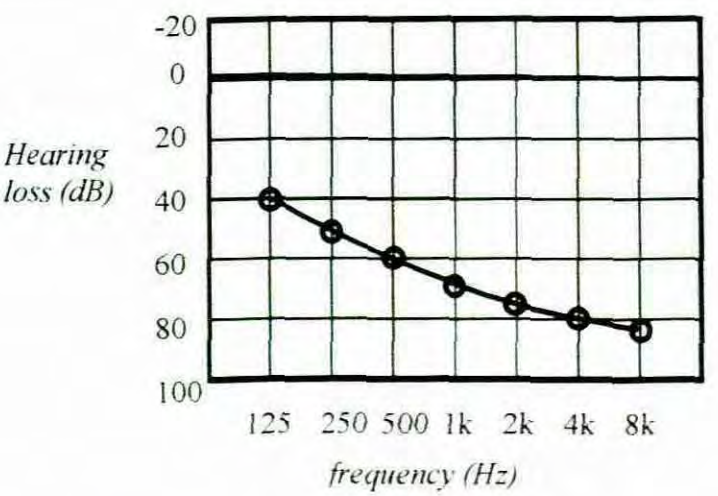


Figure 4.1 Example of test results on audiogram with interpolation.

The filter coefficients are obtained by applying an inverse Fourier transform to the grid. To cover every aspect of this method in detail would be impossible, therefore only relevant issues are discussed in this report. The structure of this chapter is as follows.

a. Definitions
b. Analog signals and Fourier analysis
c. Periodic signals and Fourier series
d. Discrete-time signal and Fourier series
e. Frequency response and the Discrete Fourier Transform (DFT)
f. FIR digital filter design by frequency sampling
g. Digital filtering by multiplication in the frequency domain

h. Conclusion

4.2 Definitions

The definitions in this section are fundamental to the theory of FIR design that will follow later in the chapter.

4.2.1 Linear systems

Martin (1991:120) defines linear systems as follows:
A discrete system is linear if the response of a system or process to the sum of two or more signals is the same as the sum of the responses to the individual signals. Expressed algebraically, consider two input signal x_1 and x_2 respectively. The process is represented by some function F[ ].
So, when the two signals are connected to the input of the process individually,

$y_1 = F[x_1]$ and $y_2 = F[x_2]$

Applying them simultaneously, the system is linear if the following relationship is satisfied:

$a_1 y_1 - a_2 y_2 = F[a_1 x_1 - a_2 x_2]$

4.2.2 Inverse Fourier transforms

Fourier transforms are used to convert back and forward between the time and frequency domains. The Fourier transform is normally used to determine the frequency response when a time domain sequence is given. The inverse Fourier transform on the other hand is used to determine the time domain sequence, given the frequency response. The Fourier series enables one to represent a periodic function as an infinite trigonometrical series in sine and cosine terms.

4.2.3 Convolution

The *convolution operation* is crucial, since it allows us to calculate the output signal, knowing the input signal and the unit-pulse response of the process or system. Ludeman (1986:28) states that the following theorem shows that a discrete-time linear invariant

system can be characterized by its unit sample response $h(n)$. That is $h(n)$ provides all the information needed to determine the response of any input.

Theorem 1. If x(n) is the input to a linear shift invariant discrete-time system then y(n) the output, is given by

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) \qquad (4.1)$$

where $h(n) = T[\delta(n)]$

The proof can be found in Ludeman (1986:28)

4.3 Analogue signals and Fourier analysis

Analogue signals can produce an infinite variety of different waveforms, which can be very complicated and difficult to describe. Fourier analysis is used to relate such waveforms to a range of simpler ones referred to as sinusoids. A sinusoid is a waveform whose voltage x(t) at time t is given by the formula:

$$x(t) = A\cos(\Omega t + \phi) \qquad (4.2)$$

where A is the amplitude (in volts), Ω is the angular frequency in radians per second and ϕ is the phase angle in radians.

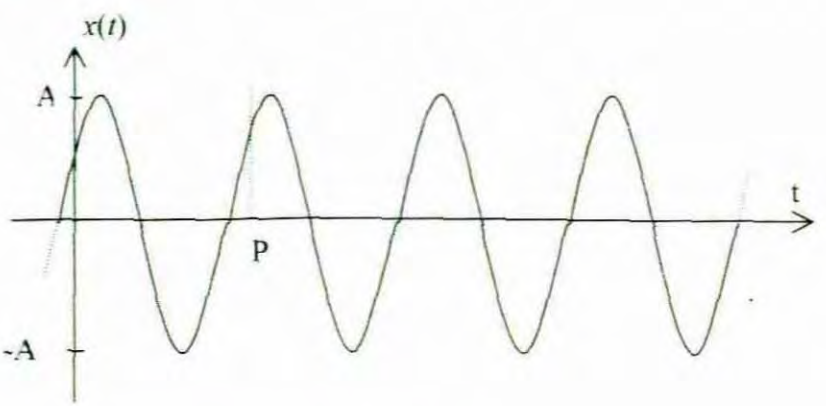


Figure 4.2 Segment of a sinusoidal waveform

Figure 4.2 shows a short segment of the sinusoidal waveform which in theory extends from $t = -\infty$ to $t = +\infty$. The waveform is periodic in that a fundamental cycle from $t = 0$ to $t = P$

is repeated at intervals of P seconds, where $P = 2\pi / \Omega$. The frequency of repetition is $1/P$ cycles per second of Hertz (Hz). The phase angle ϕ determines which point on the waveform occurs at time $t = 0$.

4.4 Periodic signals and Fourier series

Many signals are encountered which are periodic but not sinusoidal. An example is given in Figure 4.3. A periodic signal x(t), with period P seconds, satisfies:

$$x(t+P) = x(t) \text{ for all values of } t \text{ from } -\infty \text{ to } +\infty \qquad (4.3)$$

Under certain conditions normally satisfied by signals of practical interest, the periodic waveform x(t) may be expressed as the sum of a series of sinusoids, i.e.:

$$x(t) = A_0 + \sum_{n=1}^{\infty} A_n \cos(n\Omega_0 t + \phi_n) \qquad \text{(Fourier series)} \qquad (4.4)$$

This is known as a Fourier series with a fundamental frequency of Ω_0 radians/seconds. When $x(t)$ has period P seconds, $\Omega_0 = 2\pi / P$.

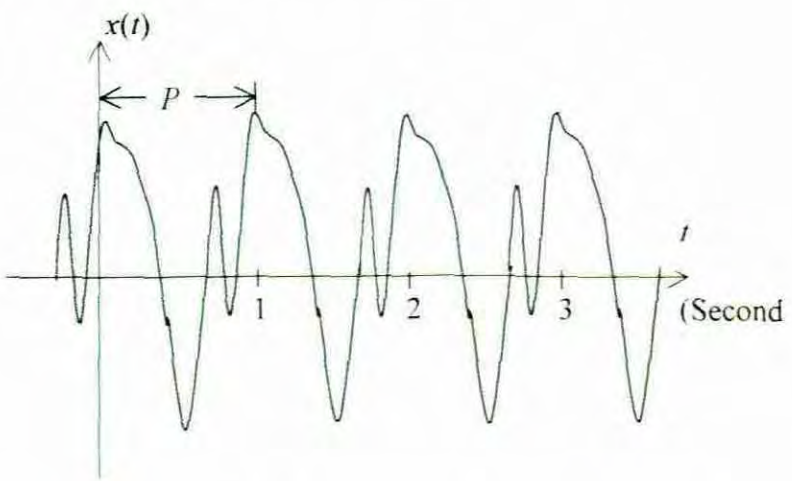


Figure 4.3 Segment of a non-sinusoidal periodic waveform

The Fourier series amplitude coefficients $A_0, A_1, A_2, \ldots$ and phase coefficients $\phi_1, \phi_2, \phi_3 \ldots$ are constants which characterize $x(t)$. Therefore $x(t)$ has been expressed as the sum of a constant A_0 and sinusoids of angular frequency Ω_0, $2\Omega_0$, $3\Omega_0$, and so on. The sinusoid at angular frequency Ω_0, i.e $A_1 \cos(\Omega_0 t + \phi_1)$, is called the fundamental frequency component of $x(t)$. The other sinusoids are called harmonic components at

angular frequency, $2\Omega_0$ being the second harmonic, the component at $3\Omega_0$ being the third harmonic and so on.

It is often convenient to re-express Equation (4.4) in complex form by means of the relation:

$$\cos x = \left(e^{jx} + e^{-jx}\right)/2$$

where $j = \sqrt{-1}$. The complex (or exponential) Fourier series which results is:

$$x(t) = A_0 + \sum_{n=1}^{\infty} \tfrac{1}{2} A_n \left[e^{j(n\Omega_0 t + \phi_n)} + e^{-j(n\Omega_0 t + \phi_n)} \right]$$

$$= \sum_{n=-\infty}^{\infty} C_n e^{jn\Omega_0 t} \qquad \text{(complex Fourier series)} \qquad (4.5)$$

where $C_0 = A_0$, $C_n = \tfrac{1}{2} A_n e^{j\phi_n}$ *and* $C_{-n} = \tfrac{1}{2} A_n e^{-j\phi_n}$

$$\text{for } n = 1,2,3\ldots \qquad (4.6)$$

The Fourier series (4.4), or equivalent (4.5), has an infinite number of terms. Waveforms for which all but a finite number of these terms are zero, are said to have *finite bandwidths*. The example shown in Figure 4.3 has the Fourier series

$$x(t) = 1 + 4\cos(2\pi t) + \cos(4\pi t - \pi/2) + 2\cos(6\pi t + \pi/3) \qquad (4.7)$$

For some idealized waveforms, like the square wave, the Fourier series has an infinite number of non-zero terms. Such idealized waveforms are very useful as mathematical concepts and may be approximated by real analogue signals. The closeness of the approximation will always be limited to some extent by finite bandwidth constraints imposed by practical systems.

4.5 Discrete-time signals and Fourier series

Obtaining the discrete-time sequence from the frequency response is essential to this project, since it would produce the unit pulse response used in the convolution operation needed to perform the filtering at real time. The previous discussion has used continuous-time as an introduction. The same properties could be applied to discrete-time signals. The only difference according to Martin (1991:68) is that the phasors progress in phase jumps of ΩT_s instead of a regular velocity Ω rad/sec. Their complex magnitude can have the same

significance whether it is a continuous-time signal or a discrete-time signal. To take account of the discrete time constraints, t can be replaced by n and Ω by ΩT_S. Eq.4.5 will thus become

$$x(n) = \sum_{k=-\infty}^{\infty} C_k e^{jk\Omega_0 T_S n} \tag{4.8}$$

$$= \sum_{k=-\infty}^{\infty} C_k e^{jk\frac{2\pi}{N}n} \tag{4.9}$$

because $\Omega_0 T_S = \frac{2\pi}{N}$

To prove that $\Omega_0 T_S = \frac{2\pi}{N}$, let T equal the period of the fundamental as illustrated in Figure 4.4.

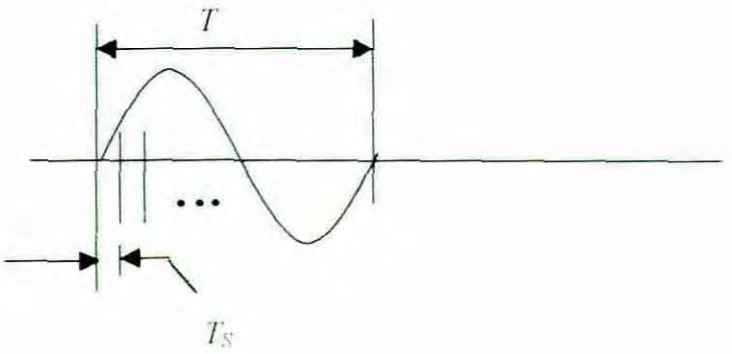


Figure 4.4 One period of the fundamental

$$T_S = \frac{T}{N}$$

where N is the number of samples per period (of the fundamental) and T_S is the period between samples, also called the sampling period.

$$\Omega_0 T_S = \frac{2\pi}{T} \times \frac{T}{N} \qquad \text{for} \quad \Omega_0 = \frac{2\pi}{T}$$

$$= \frac{2\pi}{N}$$

Equation 4.9 will now become $$x(n) = \sum_{k=-\infty}^{\infty} C_k e^{j\omega_k n} \tag{4.10}$$

where $\omega_k = \frac{2\pi k}{N}$

Referring to Equation 4.10, it would be impossible to store an infinite number of complex numbers on a computer. To overcome this problem it means that 'frequency domain

sampling must be used to obtain a representation of C_k as a finite set of complex numbers. A different notation for C_k is $C(k)$ or $C(e^{j\omega})$. For real signals, it would be sufficient to store values of $C(e^{j\omega})$ only in the range $0 \le \omega < \pi$ since $C(e^{-j\omega})$ is the complex conjugate of $C(e^{j\omega})$ and $C(e^{j\omega})$ is repetitive at intervals of 2π. In practice, this range is often extended to $0 \le \omega < 2\pi$ to allow for complex-valued signals generated by mathematical formulae.

Taking M equally spaced frequency domain samples in the range $0 \le \omega < 2\pi$ produces the finite sequence of complex numbers:

$$\left\{C\left(e^{j\omega k}\right)\right\}_{0..M-1} = \left\{C\left(e^{j\omega 0}\right), C\left(e^{j\omega 1}\right),\ldots,C\left(e^{j\omega M-1}\right)\right\} \qquad (4.11)$$

where: $$\omega_k = 2\pi k \ \ for\ k = 0,1\ldots,M-1. \qquad (4.12)$$

This will be discussed in further detail in the next section.

For real signals, $C\left(e^{j(2\pi-\omega_k)}\right)$ will be equal to the complex conjugate of $C\left(e^{j\omega k}\right)$ for all ω_k because the spectrum for $-\pi \le \omega_k \le 0$ is repeated for $\pi \le \omega_k \le 2\pi$.

Representing a time sequence by the Fourier series means that it is represented by 0 to M-1 harmonically-related phasors. Equation 4.10 can thus be rewritten as

$$x(n) = \sum_{k=0}^{M-1} C_k e^{j\omega_k n} \qquad (4.13)$$

4.6 Frequency response and the Discrete Fourier Transform (DFT)

The unit pulse response $h(n)$ can be used to calculate the output signal of a process in response to a general input signal, but it is also know that when the input is a phasor, the output signal can be calculated using the frequency response $\mathbf{H}(\omega)$.

The unit-pulse response may be used for any kind of input signal via the convolution operation, but the frequency response is used only for the restricted case of phasor inputs. These two approaches can be linked. For a sampling interval of T_s, and a single frequency phasor input, the input and output signal are of the form:

$$x(n) = e^{j(\omega T_S n)} \qquad (4.14)$$

$$y(n) = \mathbf{H}(\omega) e^{j(\omega T_S n)} \qquad (4.15)$$

From the convolution Equation 4.1

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} x(n-k)h(k)$$

it can be derived that

$$y(n) = \mathbf{H}(\omega).e^{j(\omega T_S n)} = \sum_{k=-\infty}^{\infty} h(k).e^{j(\omega T_S (n-k))} \qquad (4.16)$$

or $$\mathbf{H}(\omega) = \sum_{k=-\infty}^{\infty} h(k).e^{j(-\omega T_S k))} \qquad (4.17)$$

This is the most important result, relating the phasor or frequency response to the unit-pulse response. It is know as the Discrete Time Fourier Transform (DTFT) relationship. The DTFT essentially transforms an infinite sequence to a continuous function of ω. The problem with the DTFT is that an infinite range of summation would be rather difficult or impossible to implement on special-purpose hardware for digital signal processing. The difficulty is overcome by setting to zero all but a finite set of block of say, N samples of $\{h(n)\}$ to produce the 'windowed' sequence $\{\ldots 0, \ldots, h[0], h[1], h[2], \ldots, h[N\text{-}1]\}$. This infinite sequence may be conveniently represented by the finite sequence:

$$\{h[0], h[1], h[2], \ldots, h[N\text{-}1]\} \qquad (4.18)$$

which can be denoted by $\{h[n]\}_{0,N-1}$.

The imposition of windowing and frequency domain sampling on the DTFT (Equation (4.17)) produces the following equation with x replacing h:

$$X\left(e^{j\omega_k}\right) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_k n} \quad \text{where } \omega_k = 2\pi k/M \qquad (4.19)$$

which is normally evaluated for $k = 0,1,2,\ldots M\text{-}1$. For spectral analysis, the larger M is made, the easier it is to draw a smooth and accurate graph of the modulus and phase of $X(e^{j\omega})$ over the range $0 \le \omega < 2\pi$. However, in many applications it is important to evaluate just sufficient frequency domain samples to obtain a compact and unambiguous spectral representation of a windowed signal as quickly as possible. Multiplying Equation (4.19) by $e^{j\omega_k m}$ and summing over the block of M frequency domain samples:

$$\sum_{k=0}^{M-1} X\left(e^{j\omega_k}\right)e^{j\omega_k m} = \sum_{k=0}^{M-1} \sum_{n=0}^{N-1} x[n]e^{j\omega_k (m-n)}$$

$$= \sum_{n=0}^{N-1} x[n] \sum_{k=0}^{M-1} e^{2j\pi k(m-n)/M}$$

$$= Mx[m] \text{ if } 0 \le m < N \text{ and } N \le M \qquad (4.20)$$

since it may be shown by summing the geometric series that provided $-M < m\text{-}n < M$:

$$\sum_{k=0}^{M-1} e^{2j\pi k(m-n)/M} = \begin{cases} M & : if\ m = n \\ 0 & : if\ m \neq n \end{cases} \qquad (4.21)$$

The restriction $N \leq M$ is needed to ensure that $-M < m - n < M$ for all values of m and n in the range 0 to N. Relaxing this restriction would invalidate Equation (4.20) for some values of m. Therefore, a minimum of N frequency domain samples are needed in the range $0 \leq \omega < 2\pi$ to ensure that all samples of $\{x[n]\}_{0,N-1}$ can be reconstructed exactly, thus guaranteeing that no in formation about $\{x[n]\}_{0,N-1}$ is lost in the frequency domain sampling process. When $M = N$, the complex sequence defined by Equation (4.19) becomes the discrete Fourier transform (DFT) of $\{x[n]\}_{0,N-1}$. Introducing the following notation.

$$X\left(e^{j\omega k}\right) = X[k] \qquad (4.22)$$

the DFT may be defined as the transformation:

$$\{x[n]\}_{1,N-1} \quad \vec{DFT} \quad \{X[k]\}_{1,N-1} \qquad (4.23)$$

with

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jw_k n} \qquad \text{(DFT)} \qquad (4.24)$$

where

$$\omega_k = 2\pi k / N \ \ for\ k = 0,1,2,\ldots,N-1 \qquad (4.25)$$

It is normal to consider $\{x[n]\}_{0,N-1}$ as a complex sequence, although its sample values may be given zero imaginary parts for real signals. The following inverse DFT formula is obtained from Equation (4.20) with $M = N$:

$$\{x[k]\}_{0,N-1} \quad \overleftarrow{IDFT} \quad \{X[n]\}_{0,N-1} \qquad (4.26)$$

with

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{jnw_k} \qquad \text{(IDFT)} \qquad (4.27)$$

and ω_k defined as for the DFT.

Computer programs and hardware can exploit the similarity between Equations (4.24) and (4.27), which are able to perform the DFT or its inverse using essentially the same code. A

program for implementing the DFT and its inverse can be found in Lockhart and Cheetham (1989:29). The implementation of the IDFT in this application has some drawbacks. Practical implementation has proven that because the time it takes to perform the calculation is too slow for practical use. This is the main reason for the development of the Fast Fourier Transform (FFT), which will be discussed later.

4.7 FIR digital filter design by frequency sampling

The Fourier series method depends on the availability of a Fourier series expansion for the target frequency response $H(e^{j\omega})$. According to Lockhart and Cheetman (1989:95) a method called the 'frequency sampling' method can be used to derive an approximation to the required Fourier series through the inverse DFT expression (4.27) and they explain it as follows:

"Consider a finite sequence $\{H[k]\}_{1,2N}$ consisting of $2N+1$ samples of the target frequency response $H(e^{j\omega})$ specified at intervals of $2\pi/(2N+1)$ radians per sample over the relative frequency range $0 \le \omega < 2\pi$. The modulus and phase of $H(e^{j\omega})$ may be chosen arbitrarily for $0 \le \omega < \pi$ except that when $\omega = 0$, $H(e^{j\omega})$ must be real. When $\omega \ge \pi$, $H(e^{j\omega})$ must be equal to the complex conjugate of $H(e^{j(2\pi - \omega)})$. Therefore:

$$H[k] = H\left(e^{j2\pi k/(2N+1)}\right) \qquad \text{for } k = 0,1,\ldots,N \qquad (4.28)$$

with $H[k]$ equal to the complex conjugate of $H[2N+1-k]$ when $N < k \le 2N$. $\{H[k]\}_{0,2N}$ may be regarded as the DFT of a finite sequence of $2N+1$ samples: $\{[h[n]\}_{1,2N}$

By the inverse DFT formula (2.27), the samples of this sequence are:

$$h[n] = \frac{1}{(2N+1)} \sum_{k=0}^{2N} H[k] e^{j2\pi nk/(2N+1)} \quad \text{for n} = 0,1,2,\ldots,2N \qquad (4.29)$$

Since the DFT of $\{h[n]\}_{0,2N}$ is $\{H[k]\}_{0,2N}$, the frequency response of an FIR filter with impulse response:

$$\{h[n] = \{\ldots,0,\ldots,0,h[0],h[1],\ldots,h[2N],0,\ldots,0,\ldots\}$$

will be equal to the target frequency response $H(e^{j\omega})$ at the $2N+1$ frequency sampling points defined above. The FIR filter's frequency response may deviate from $H(e^{j\omega})$ between sampling points, and N must be large enough to ensure that the target frequency response is properly characterized."

Appendix B section D shows the routine for the design of the FIR digital filter based on the frequency sampling method.

4.8 Digital filtering by multiplication in the frequency domain

Signal processing transforms can replace a convolution operation by a multiplication of transforms. This technique is important for real processing, using the numerical form of the DFT. If the signal sequence $\{x(n)\}$ is to be processed by a digital filter having a unit-pulse response $\{g(n)\}$, then the convolution in the time domain of these two sequences can be replaced by the multiplication of their transforms. Expressed in symbolic form, the output signal sequence $\{y(n)\}$ is in principle given by

$$y(n) = \text{inverse DTFT [DTFT}[x(n)] \,.\, \text{DTFT}[g(n)] \text{]}$$

The computational advantage of this strategy is not obvious until it is examined in detail. Martin (1991:284) discusses this method as follows. Let the unit-pulse sequence be FIR, and have a length of L points, while the signal sequence contains N points. A direct convolution of the two sequences requires $L.N$ multiply-add operations. The transformation route requires the following number of complex multiply-add operations:

FFT on $[x(n)]$	$(N/2).\log_2(N)$
FFT on $[g(n)]$	$(N/2).\log_2(N)$
Product of transforms	N
Inverse FFT to give $\{y(n)\}$	$(N/2).\log(N)$

Each complex multiply-add operation is broadly equivalent to 4 real multiply-add operations, so that if the transformation sequence is to be beneficial then

$$6 \,.\, \log_2(N) < (L - 4)$$

For L = 128 for instance, any value of N in the range $128 \leq N \leq 1.66 \times 10^6$ offers an improvement in processing load. In practice, other benefits accrue since the DFT of $\{g(n)\}$ need only be calculated once, and also there are compact FFT routines for real-data which almost halve the computational effort required for the general complex-data version. However, as in all practical enterprises, the trade-offs are not all in one direction. Multiplication of transforms generates a circular convolution, which distorts the result in an unacceptable fashion. The inverse-DFT is a periodic time sequence and the inverse transformation of a finite-DFT $X_N(k)$ gives

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X_N(k).\exp j(2\pi kn / N)$$
$$= x(n + rN)$$

where r is an integer.

The index n is therefore effectively modulo-N, and the function $x(n)$ is periodic in N. Alternatively, since the spectral function is sampled, it follows that the corresponding time sequence is periodic. Convolution of such sequences is written as in Equation 4.30, and since all indices are modulo-N, the resulting sequence is periodic or circular:

$$y(n) = \sum_{M=0}^{N-1} x(m).g(n-m) \qquad (4.30)$$
$$= y(n + rN)$$

The circular convolution causes the function to wrap around add extra products to the convolution expression of Equation 4.30. Clearly this effect must be avoided. There are several strategies to avoid the effects of circular convolution, and a brief description follows of the Overlap-save method in order to illustrate the principle. Consider a filter which has a unit pulse response $\{g(n)\}$ of length L points. A $2L$-point frequency response is first formed, by padding $\{g(n)\}$ out with L zeros:

$$H_{2L}(k) = \sum_{n=0}^{L-1} g(n).W_{2L}^{-kn} \qquad (4.31)$$

where $W_{2L} = \exp j(2\pi / 2L)$.

Using the transform to convolve the expanded unit pulse sequence with a $2L$-point signal sequence, yields a sequence of $2L$ points. Since the second half of the unit-pulse response is forced to zero, the first L points of the result suffers from circular overlap, while the second set of L points is a true aperiodic convolution of the first half of the input signal.

Assuming that the signal to be filtered is a long or infinite sequence, then a procedure outlined in Table 4.1 can be followed. The procedure is graphically illustrated in figure 4.5.

Repeat:

Take a block of 2L points from the input signal, $\{x(n)\}$

Take a $2L$-Point DFT to form the spectrum $X_{2L}(k)$

Multiply transform: $V_{2L}(k) = X_{2L}(k).\ H_{2L}(k)$

Take the inverse transform of $V_{2L}(k)$ to form the $2L$-point set $\{v(n)\}$

Save the second set of L points from $\{v(n)\}$ to form $\{y(n)\}$

$y(n) = v(n+L) \quad 0 \le n \le L-1$

Shift the origin of the input signal by L points, overlapping the original set by L points.

End

Table 4.1 Overlap-*save procedure*

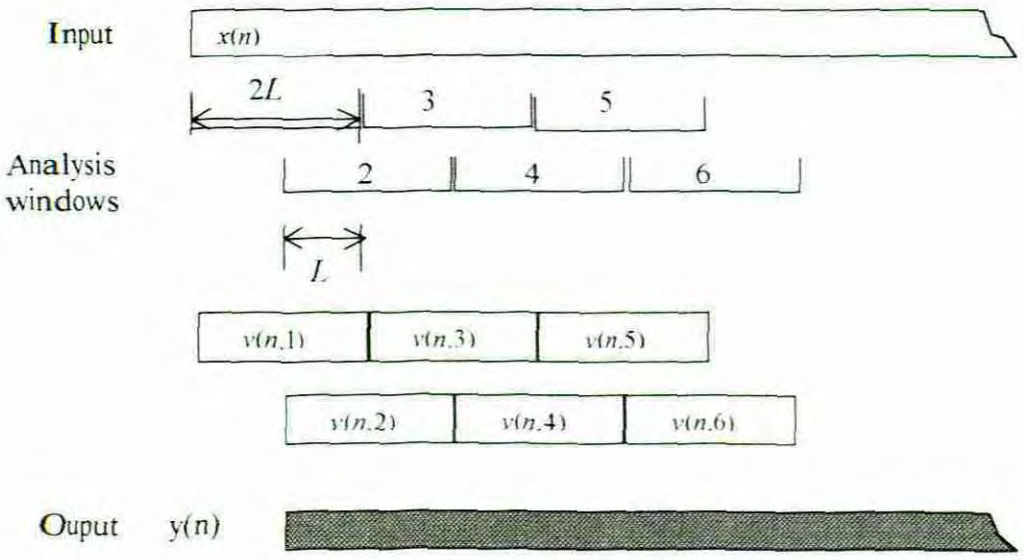


Figure 4.5 Overlap-save convolution

4.9 Conclusion

The filter operation can be done in the frequency domain or the time domain. In the time domain the filtering is done by means of a FIR filter of which the coefficients are determined by the IFFT computation. The FIR filter performs a simple convolution operation in the time domain. Alternatively, the filtering can be done in the frequency domain by means of the Overlap and save method. These methods will be compared in chapter 9. The DFT describe in this chapter take a long time to perform its calculation. The Fast Fourier transform (FFT) is much faster and is discussed in the next chapter.

CHAPTER 5

FAST FOURIER TRANSFORMS

5.1 Introduction

Computing the DFT described in the previous section is extremely time consuming, with the result that it is not feasible to implement it directly on a practical computer based system. This drawback necessitates the introduction of the Fast Fourier transform (FFT) which is much faster. The advancement in speed of FFT's is due to the reduction in the number of multiplications, due to the process of breaking up the sequence into smaller sequences and recombining them to give the total transform. On the history of FFT's Higgins (1990:122) states that "The FFT algorithm was a rediscovery of an idea of Runge (1903) and Danielson and Lanczos (1942) ..." which was later popularized by Cooley and Tukey (Ludeman, 1986:272). Breaking up the transforms is also called the decimation process, of which there are two methods. One is the Decimation in Time (DIT) FFT which is discussed in most literature, and the second, the Decimation in frequency (DIF) FFT. This section deals with DIT FFT and its speed-resolution tradeoffs in detail and gives an overview of the DIF FFT.

5.2 Decimation-in-Time FFT

The Decimation in time Fast Fourier transform breaks an N-point transform into even and odd sequences of length N/2 each, then breaks each N/2 transforms into two N/4 sequences. The method requires that the number of points N be a power of two in order to allow continued division by two. This decimation process continues until only two-point transforms remain. For example, a sequence x(n) = $x(0), x(1), x(2), \ldots, x(N-1)$ can be broken up into an even-index sequence of x(0), x(2)....x(N-2) and an odd-index sequence of x(1), x(3)....x(N-1) to form the first two N/2 sequences. These sequences could be broken up further as illustrated by an N = 8 sequence in Figure 5.1.

The analytical derivation of the Decimation in Time FFT is as follows. Equation 4.24 which is the DFT of a N-point sequence x(n) can be split up into two parts, one for the even and one for the odd-index values.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0,1,\ldots,N-1$$

$$= \sum_{\substack{n=0 \\ n \text{ even}}}^{N-2} x(n) W_N^{kn} + \sum_{\substack{n=1 \\ n \text{ odd}}}^{N-1} x(n) W_N^{nk} \qquad (5.1)$$

where $W_N^{kn} = e^{-j2\pi kn/N}$

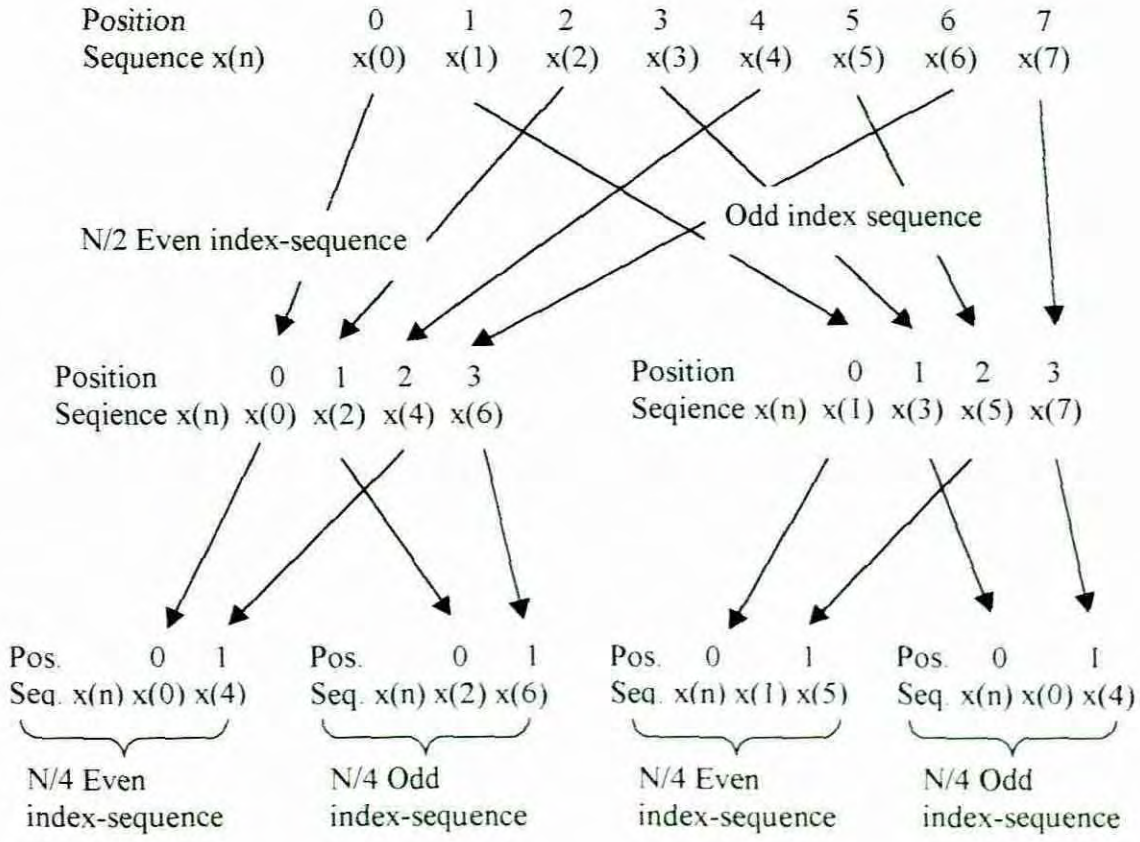


Figure 5.1 Continues splitting of even and odd sequences.

Letting $n = 2r$ in the first term and $n = 2r + 1$ in the second, equation 5.1 can be rewritten as

$$X(k) = \sum_{r=0}^{N/2-1} x(2r) W_N^{2rk} + \sum_{r=0}^{N/2-1} x(2r+1) W_N^{(2r+1)k} \qquad (5.2)$$

$$= \sum_{r=0}^{N/2-1} x(2r) W_N^{2rk} + \sum_{r=0}^{N/2-1} x(2r+1) W_N^{2rk} W_N^{k}$$

$$= \sum_{r=0}^{N/2-1} x(2r) W_N^{2rk} + W_N^{k} \sum_{r=0}^{N/2-1} x(2r+1) W_N^{2rk} \qquad (5.3)$$

If W_N^{2rk} is written as

$$W_N^{2rk} = \left(W_N^2\right)^{rk} = \exp\left(-j\frac{2\pi}{N} 2rk\right) = \exp\left(-j\frac{2\pi}{N/2} rk\right) = W_{N/2}^{rk} \qquad (5.4)$$

then the following (N/2)-point transform is derived.

$$X(k) = \sum_{r=0}^{N/2-1} x(2r) W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x(2r+1) W_{N/2}^{rk} \qquad (5.5)$$

The two sums appear in the form of N/2-point DFTs. To simplify the manipulations let $E(k)$ represent the DFT of the even-index sequence and $O(k)$ represents the DFT of the odd-index sequence . Equation 5.5 can thus be rewritten as

$$X(k) = E(k) + W_N^k O(k) \qquad for \quad 0 \le k \le N/2-1 \qquad (5.6)$$

$$= E(k-N/2) + W_N^k O(k-N/2) \qquad for \quad N/2 \le k \le N-1 \qquad (5.7)$$

Another convenient way to express Equation 5.7 is to replace k by $k+N/2$ in Equation 5.3

$$X\left(k+\frac{N}{2}\right) = \sum_{r=0}^{N/2-1} x(2r) W_N^{2r\left(k+\frac{N}{2}\right)} + W_N^{\left(k+\frac{N}{2}\right)} \sum_{r=0}^{N/2-1} x(2r+1) W_N^{2r\left(k+\frac{N}{2}\right)} \qquad (5.8)$$

$$= \sum_{r=0}^{N/2-1} x(2r) W_N^{2rk} W_N^{2r\left(\frac{N}{2}\right)} + W_N^k W_N^{\left(\frac{N}{2}\right)} \sum_{r=0}^{N/2-1} x(2r+1) W_N^{2rk} W_N^{2r\left(\frac{N}{2}\right)} \qquad (5.9)$$

$$= \sum_{r=0}^{N/2-1} x(2r) W_N^{2rk} + W_N^k W_N^{\left(\frac{N}{2}\right)} \sum_{r=0}^{N/2-1} x(2r+1) W_N^{2rk} \qquad \ldots \quad W_N^{rN} = 1 \qquad (5.10)$$

$$= \sum_{r=0}^{N/2-1} x(2r) W_N^{2rk} - W_N^k \sum_{r=0}^{N/2-1} x(2r+1) W_N^{2rk} \qquad \ldots \quad W_N^{(N/2)} = -1 \qquad (5.11)$$

Following the same steps finally gives

$$X(k+N/2) = E(k) - W_N^k O(k) \qquad for \quad 0 \le k \le N/2-1 \qquad (5.12)$$

From equation 4.4 the DFT calculation requires N number of summations for N frequency points which means that N^2 multiplications are required to perform the DFT computation. The decimation of the first stage will result in $(N/2)^2$ multiplications for each of the two N/2 DFTs plus N/2 number of multiplications for combining $E(k)$ and $O(k)$ to form $X(k)$ for $0 \le k \le N-1$. Because $E(k)$ and $O(k)$ are periodic in k with period N/2, no additional multiplications are required to calculate X(k) for $N2 \le k \le N-1$, only a change in signs as illustrated in equation 5.12. The total number of multiplications will thus be $2\times(N/2)^2 + N/2$ which is almost half of N^2. Equation 5.6 and 5.12 can be represented in a standard flow graph notation as illustrated in Figure 5.2a . This single stage is commonly known as a butterfly. The output of the nodes is equal to the sum of the weighted inputs. The block arrows W_N^k and $-W_N^k$ which are the weights are constant multipliers and are known as the 'twiddle' factor.

The 'twiddle' factor rotates $O(k)$ through an angle of $-2\pi k/N$ radians without affecting its magnitude. A further reduction can be accomplished if the scale factors in Figure 5.2a are removed and placed at the front end of the bottom leg, as shown in Figure 5.2b.

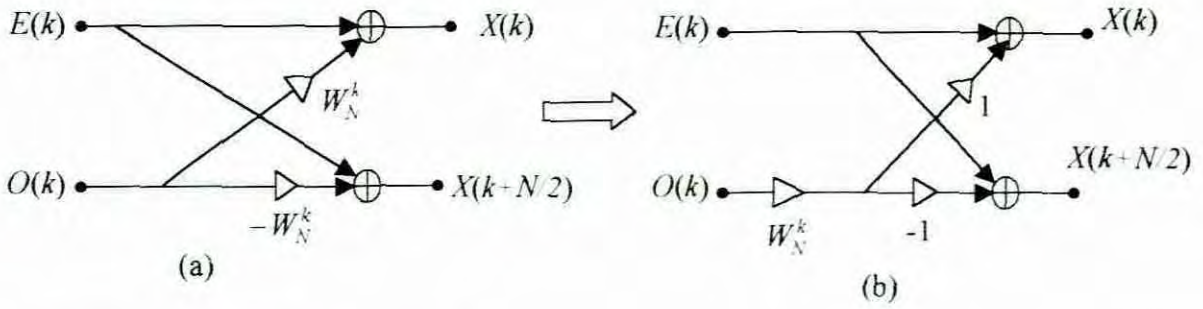


Figure 5.2 The Butterfly patterns

The signs need to be kept in position. This effectively reduces the number of complex multiplications by half. The combination of the two N/2-point DFTs and the derivation of the DFT of the original sequence $x(n)$ is illustrated in Figure 5.3 where N=8.

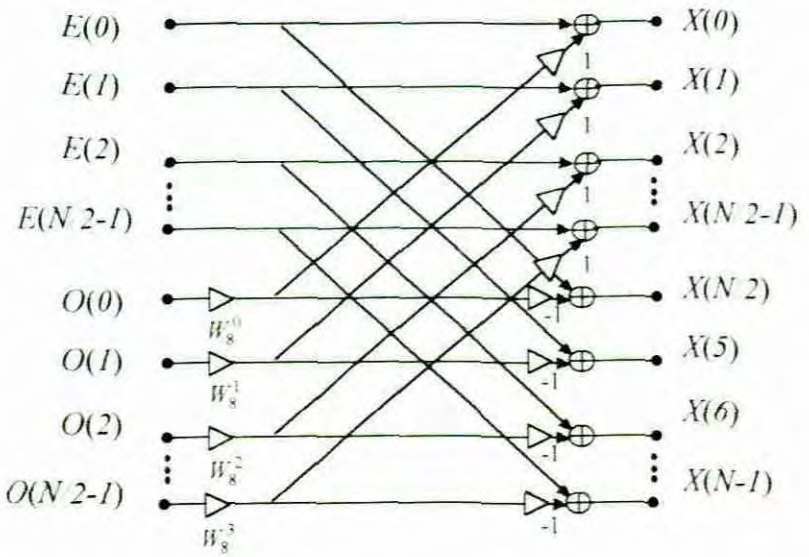


Figure 5.3 First stage of the decomposition of the DIT FFT

It is obvious that if the decimation reduces the sequence, further decimation will reduce it even more. Each of the two N/2-point DFTs can now be seen as new original sequences which can be split further into odd and even sequences to give two N/4-point DFTs on each N/2-point DFT. The DFT of the new even-index sequence can be written as

$$E(k) = E_E(k) + W_{N/2}^{k} O_E(k) \qquad for \quad 0 \le k \le N/4-1$$
$$= E_E(k) + W_N^{2k} O_E(k) \tag{5.13}$$

$$E(k+N/4) = E_E(k) - W_N^{2k} O_E(k) \qquad for \quad 0 \le k \le N/4-1 \tag{5.14}$$

and the DFT of the new odd-index sequence can be written as:

$$O(k) = E_O(k) + W_N^{2k} O_O(k) \qquad for \quad 0 \le k \le N/4-1 \tag{5.15}$$

$$O(k+N/4) = E_O(k) - W_N^{2k} O_O(k) \qquad for \quad 0 \le k \le N/4-1 \tag{5.16}$$

Figure 5.4 shows the combination of the first and second stage of the decomposition of the DIT FFT. In this diagram the weight of the block arrows in the upwards diagonal legs is one and the block arrows in bottom horizontal leg nearest to the nodes is minus one.

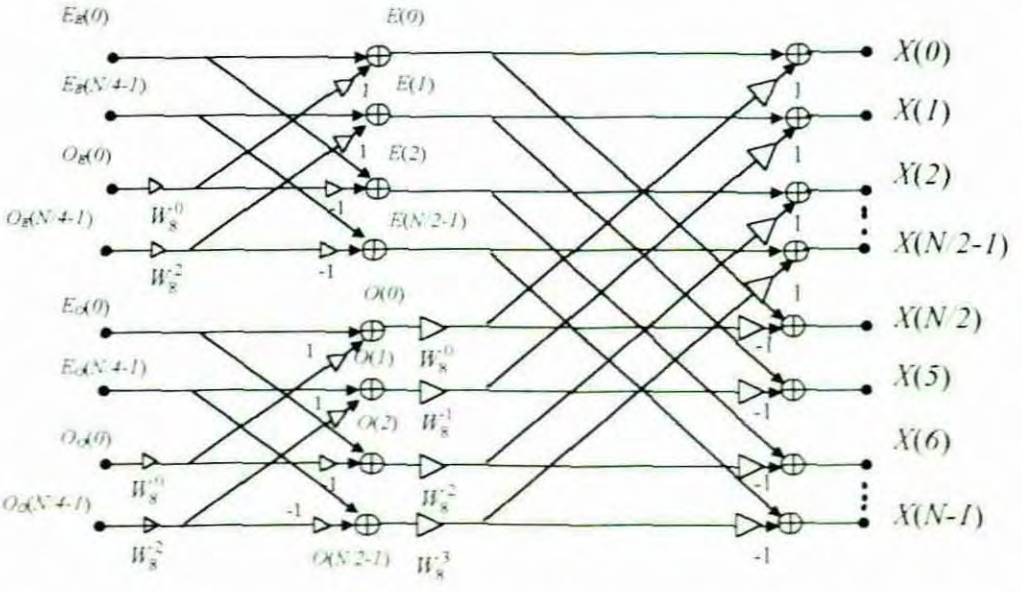


Figure 5.4 Two stages in the decompositioning of the DIT FFT

After the second decimation the number of complex multiplications will be $4\times(N/4)^2 + 2\times N/4 + N/2 = N^2/4 + N$. The first term on the left-hand side is used to calculate the four N/4-point DFTs. The second term is the number of multiplications needed for the two N/2-point combining algebra in the second stage, while the last term is the number of multiplications needed for the combining algebra of the first stage. If further decimation is performed in this example, only single point DFTs remain, as shown in figure 5.5

From the above it can be seen that the number of samples (N), equals $N = 2^m$, where m is the number of stages. The number of stages required for the entire FFT calculation could

thus be calculated as m = $\log_2$N. In Figure 5.5 the stages are numbered from left to right, as a computer program performed the calculation. In the first stage, the first even-index 2-point DFTs are given by

$$E_E(k) = x(0) + x\left(\frac{N}{2}\right)W_{N/4}^{k} = x(0) + x\left(\frac{N}{2}\right)W_N^{4k} \qquad for \quad k = 0,1 \qquad (5.17)$$

$$E_E(0) = x(0) + x\left(\frac{N}{2}\right) \qquad for \quad W_N^{4\times 0} = 1 \qquad (5.18)$$

and

$$E_E(1) = x(0) - x\left(\frac{N}{2}\right) \qquad for \quad W_N^{4\times 1} = -1 \qquad (5.19)$$

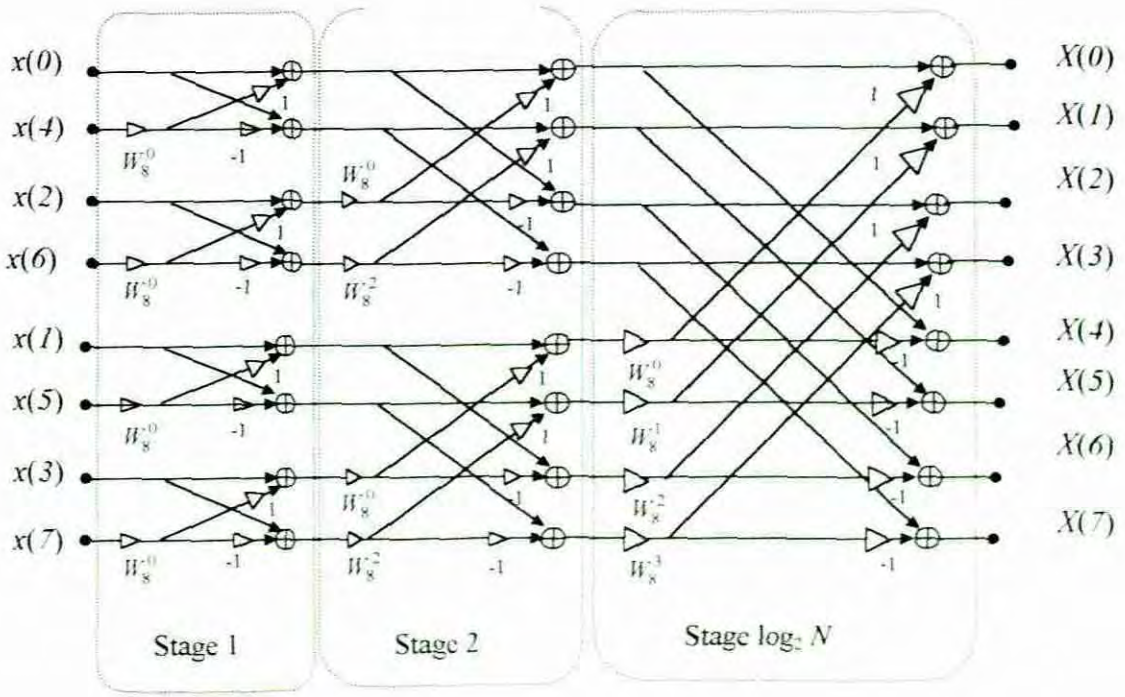


Figure 5.5 Flow graph for an eight point DIT FFT.

Equation 5.18 and 5.19 shows that in the first stage only addition and subtraction operations are performed. It has been a convention, though, to count these addition and subtraction operations as a complex multiplication. The total number of complex multiplications (T_{nm}) is then equal to $\frac{N}{2}\log_2 N$, since the combining algebra at each stage takes N/2 complex multiplications and there are $\log_2$N stages. If a suitable number of points for this project is choosen to be 256, the number of multiplications would be $T_{nm} = (256/2)\log_2 256 = 1024$

For the direct calculation of the DFT, the number of multiplications would be $N^2 = (256)^2 = 65536$.

It could be concluded that the DIT FFT would be $(N^2)/ (\frac{N}{2}\log_2 N) = 65536/1024 = 64$ times faster than direct evaluation of the DFT. Clearly, the higher the number of samples, the higher the reduction in multiplications and consequently the greater saving in speed.

5.3 Bit reversal

The shuffling of the input sequence is illustrated in Table 5.1. By expressing the original sequence as a binary number and then reversing the bits, the pattern in which the input sequence is reordered can be seen. Table 5.1 below reveals this phenomena.

Position	Binary	Bit reversal	Shuffled sequence
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Table 5.1 Shuffling by bit reversal

5.4 Memory requirements

The calculation of the DIT FFT is done from left to right. After the output of the first stage is determined, it is used as new input to the next stage. Input data sequences which are no longer needed shift out of memory, while the new calculated output sequence takes the place of the input sequence. This form of computation is called 'In place' calculation and is the reason why the DIT FFT algorithm is more memory-efficient then the direct DFT computation.

5.5 Coefficient values

The value of the twiddle factor will depend on the position of the butterfly in the structure. It could be calculated in two ways. One way is to calculate the values as the program runs. The other way is to calculate a table of values beforehand, store it in memory and retrieve the correct values from the lookup table at run time. The trade-off between the two methods is that the first method will reduce the overall speed of the computation, as it still needs to calculate the values of the twiddle factor as the program runs. On the other hand, the second method might be faster but will bring about an increase in the complexity of the subroutines since to retrieve the correct value, account should be taken of the fact that the input sequence values of each stage are in reverse-bit order. This will be discussed in further detail in the chapter on implementation consideration.

5.6 Decimation in Frequency Fast Fourier Transform (DIF FFT)

Decimation in Frequency (DIF) is also based on the principle of Decimation as for the DIT but the combining procedure is in the reverse direction. Multiplication of the twiddle factor in the case of the DIF FFT is after the summation step. As stated earlier, only a brief overview of this approach will be given in this section but a complete mathematical derivation can be found in Ludeman (1996:282/5). The following difference is clearly noticeable: The input of the DIF FFT in comparison to the DIT FFT is in normal order while the output is in reverse-bit order as shown in the flow diagram Figure 5.6.

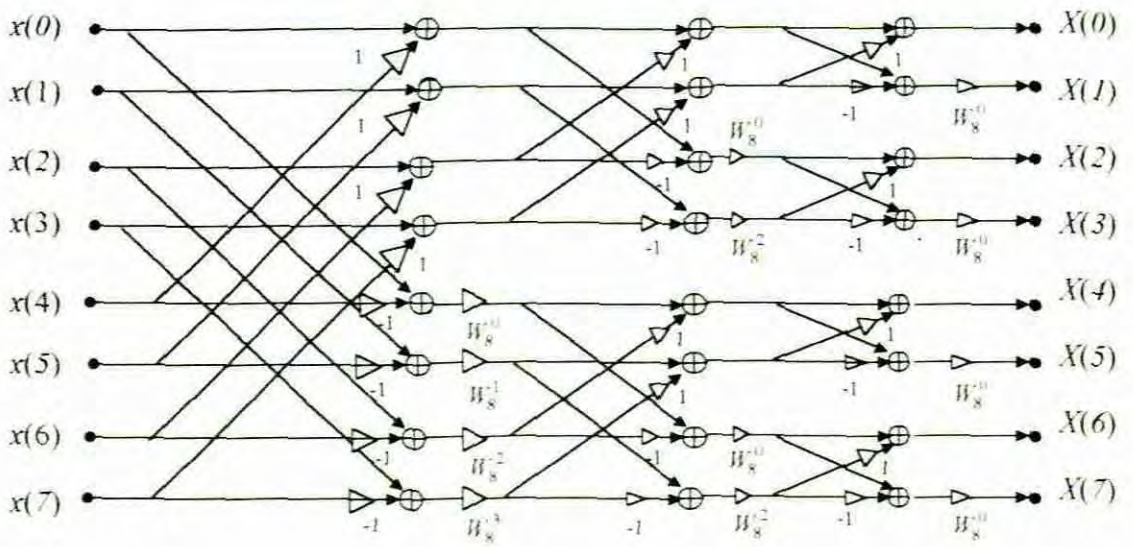


Figure 5.6 Flow graph for an eight point DIF FFT.

5.7 Inverse Fast Fourier Transform (IFFT)

In the previous chapter the equations 4.24 and 4.27, which are the DFT and the IDFT, were derived respectively. If the two are compared it can be seen that computation procedures are the same, except that the twiddle factors are of opposite powers and that the IDFT is scaled by 1/N as shown below.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0,1,\ldots,N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}, \quad k = 0,1,\ldots,N-1$$

where $W_N^{kn} = e^{-j2\pi kn/N}$

The FFT is a way of carrying out the DFT and is inherently the same, therefore the FFT flow diagram can be used to do the IFFT, provided the following steps are taken:

- The positions of X(n)'s and x(n)'s must be exchanged.
- The exponent of W_N must be changed to negative.
- The input data must be scaled by 1/N. If N is a power of two, then the input to each stage can be multiplied by ½ to get an overall scale factor of 1/N.

A wide range of programs for performing various FFT algorithms is available in a variety of high and low-level languages that can easily be obtained from sources like the internet.

5.8 Conclusion

The direct calculation of the DFT is too slow for real-time digital Fourier transforms. Due to the Cooley-Turkey FFT the serious time limitations, which make DFT unrealistic to implement in a project of this nature, can be overcome.

The similarities between the DIT FFT and the DIF FFT in terms of speed, memory and complexity has made them both equally suitable for the implementation as a faster method of computing than the direct DFT. In many cases the developer implements the FFT method as a black box, as there are many existing ready to use programs in different languages. The fact that most of this work is done on the DIT FFT makes this algorithm a natural choice.

CHAPTER 6

WAVELETS

6.1 Introduction

Making use of wavelets is a new and promising tool and technique for realising Hearing Loss simulation. This chapter on wavelets will be structured as follows:

- History of Wavelets
- Theory on the wavelets related to this project
- Implementation
- Conclusion

6.2 A brief history of wavelets

Though its mathematical underpinnings date back to the work of Joseph Fourier in the nineteenth century, wavelet analysis is a new method. Fourier laid the foundations with his theories of frequency analysis, which proved to be enormously important and influential. The attention of researchers gradually turned from frequency-based analysis to scale-based analysis when it started to become clear that an approach measuring average fluctuations at different scales might prove less sensitive to noise. The first recorded mention of the term "wavelet" was in 1909, in a thesis by Alfred Haar.

The concept of wavelets in its present theoretical form was first proposed by Jean Morlet and the team at the Marseille Theoretical Physics Centre working under Alex Grossmann in France. The method of wavelet analysis has been developed mainly by Y. Meyer and his colleagues, who have ensured the dissemination of the method. The main algorithm dates back to the work of Stephane Mallat in 1988. Since then, research on wavelets has become international. Such research is particularly active where it is spearheaded by the work of scientists such as Ingrid Daubechies, Ronald Coiman, and Wickerhauser.

6.3 Theory on the wavelets related to this project

6.3.1 Introduction

The study of wavelets and how this relates to this project will be conducted by means of an empirical investigation. Instead of mining the mathematics that underpin the construction

of wavelets, a practical approach will be taken by which experiments will determine which existing wavelet best suits the application. This approach was decided on since the mathematical depth involved in the theory of wavelets would fall outside the framework of the research project. In this regard a software tool called 'Matlab', with a built in toolbox for wavelets, will be explored and used extensively. A *Guide for the use of the Wavelet tool* in Matlab, written by Misiti, Oppenhein and Poggi (1996), presents the main ideas without mathematical complexities. This study introduces the topic on Wavelets by surveying their work.

Wavelet analysis can be done either in continuous time or discrete time. After an introduction on continuous time methods, the discrete time methods will be focused on, since all DSP based systems operate on discrete signals. Another distinction of the features of wavelet analysis that needs to be stipulated is the one-dimensional data operation versus two and higher dimensional data operation. An audio signal, which is the centre of this investigation, falls into the category of the one-dimensional. Wavelet analysis applied to two-dimensional data (*images*) and higher-dimensional data falls outside the scope of this research.

6.3.2 Fourier transforms versus wavelet transforms

In the previous chapters Fourier transforms were discussed. It would thus be natural to commence a study of wavelets by making use of Fourier transform as a reference.

Misiti, et al (1996) explains that a wavelet (ψ) could be describe as a waveform of effectively limited duration. As such it will start at t = 0 and end at t = N. Wavelets have an average value of zero implying that $\int dt\psi(t) = 0$. Wavelets $\psi_{jk}(t)$ as a basis function is a set of linearly independent functions that can be used to produce all admissible functions $f(t)$:

$$f(t) = \text{combination of basis functions} = \sum_{j,k} b_{jk}\psi_{jk}(t) \qquad (6.1)$$

In Equation 6.1, where j refers to the level of decomposition and k refers to the delay of the wavelet, it will become apparent later in the discussion what the range of summation of j and k are. It could be seen that in Equation 6.1 any signal could be expressed as a combination of wavelets multiplied by coefficients b_{jk} called wavelet coefficients. The

function $\psi(t)$ is called the mother wavelet and $\psi_{jk}(t)$ is called wavelets. All wavelets can be constructed from the mother wavelets.

When wavelets are compared with sine waves, which are the basis of Fourier analysis, it can be seen that sinusoids do not have limited duration - they extend from minus to plus infinity. Sinusoids are smooth and predictable, whereas wavelets, on the other hand, tend to be irregular and asymmetric as Figure 6.1 illustrates.

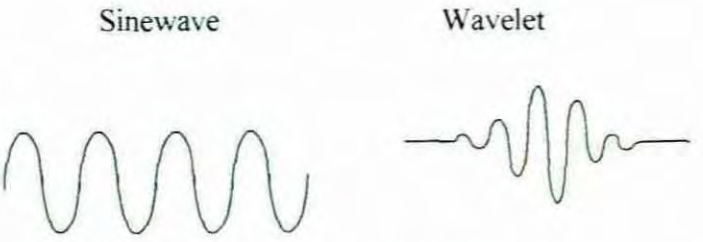


Figure 6.1 Difference between a sinusoid and a wavelet

The similarity between Fourier analysis and wavelet analysis is that in both cases the signal is broken up into different frequency components. The Fourier analysis uses sine waves to break the signal up into different frequencies. Wavelet analysis breaks the signal up into shifted and time scaled versions of the original (or mother) wavelet. Different time scales relate to different frequencies.

Misiti, et al (1996: 1-7) states that: " Mathematically, the process of Fourier analysis is represented by the standard Fourier transform:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

which is the sum over all time of the signal $f(t)$ multiplied by a complex exponential. The results of the transform are the Fourier coefficients $F(\omega)$ which, when multiplied by a sinusoid of appropriate frequency ω, yield the constituent sinusoidal component of the original signal as illustrated in Figure 6.2 ".

Similarly as Misiti, et al ((1996: 1-7) explain, the continuous wavelet transform (CWT) is defined as the sum over all time of the signal multiplied by a scaled, shifted version of the wavelet function ψ :

$$C(scale, position) = \int_{-\infty}^{\infty} f(t)\psi(scale, position)dt \qquad (6.2)$$

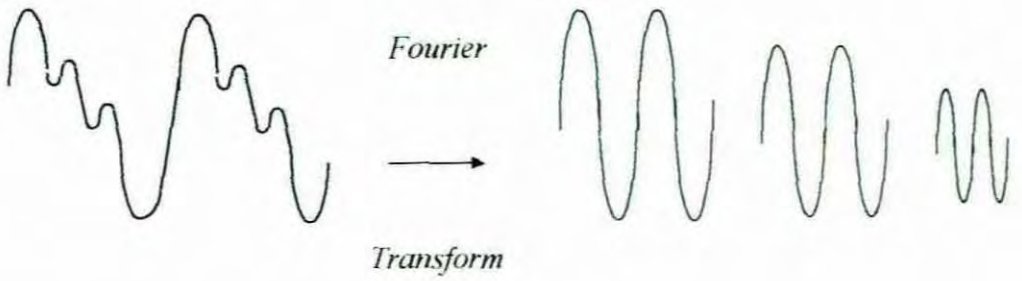


Figure 6.2 Fourier transform

The wavelet coefficients C are a function of scale and position.
Multiplying each coefficient by the appropriate scale and shifted wavelet yields the constituent wavelet of the original signal as shown in Figure 6.3.

Figure 6.3 Wavelet transform

The word scaling mentioned above means stretching (or compressing). The scaling factor is often denoted by the letter a. Scaling wavelets are illustrated by the drawings in Figure 6.4. Just as scaling the sinusoid relates to the frequency, so does scaling in wavelet analysis relate to frequency.

Shifting a wavelet means delaying (or hastening) its onset. Figure 6.5 graphically shows how delaying the wavelet actually shifts the wavelet on the time axis. The wavelet is shifted to position b. Mathematically, delaying a function $f(t)$ by k is represented by $f(t - k)$.

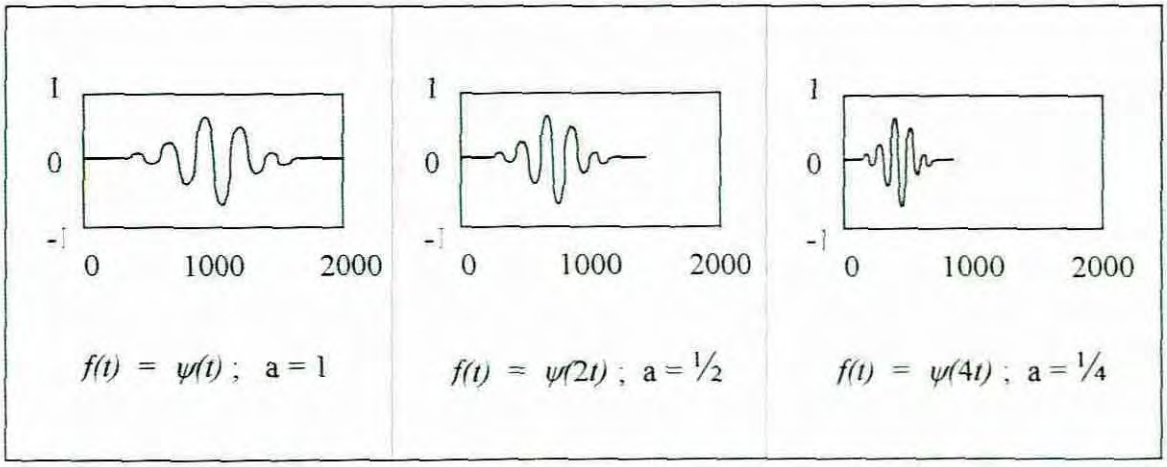


Figure 6.4 Scaling of the wavelet

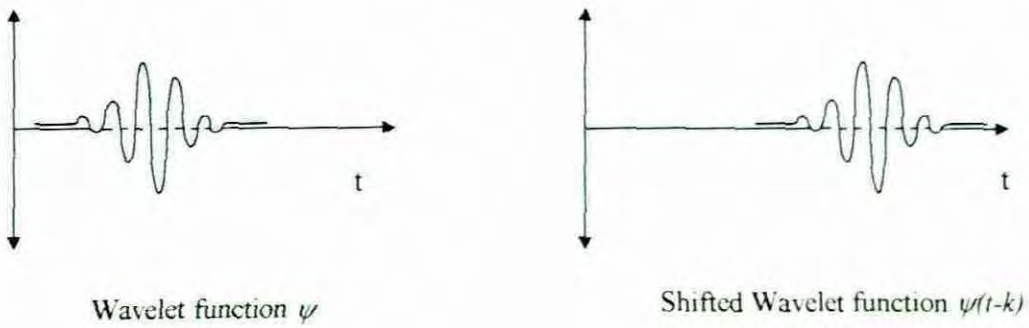
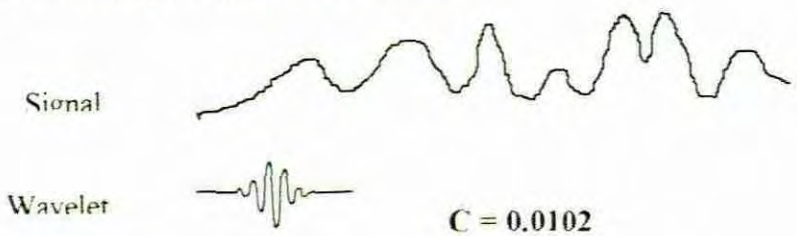


Figure 6.5 Shifting the wavelet

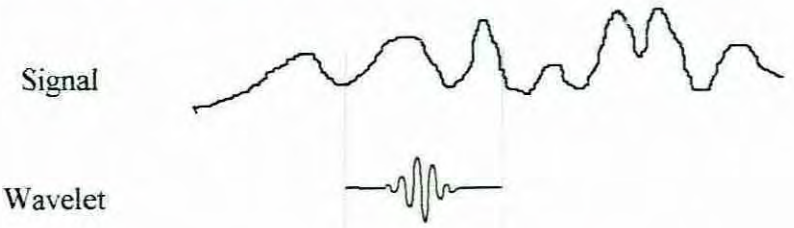
6.3.3 Wavelet transform procedures

There are five easy steps to a continuous wavelet transform and they are done in the following order:

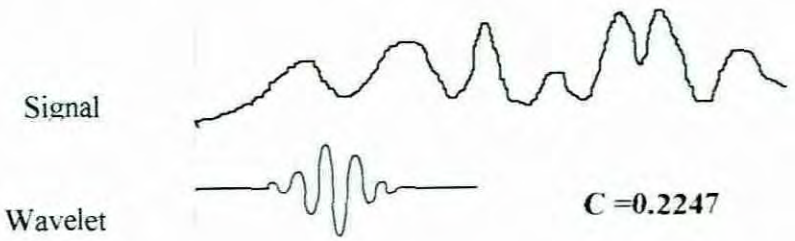
1. A wavelet is chosen and compared to a section at the start of the original signal.
2. A number C is calculated that represents how closely the wavelet correlates with this section of the signal. The higher the C is, the greater the similarity. The results will depend on the shape of the wavelet chosen.



3. The wavelet is shifted to the right or to position b and then steps 1 and 2 are repeated until the whole signal has been covered.



4. Scale (stretch) the wavelet and repeat steps 1 through 3.



5. Steps 1 through 4 are repeated for all scales.

The above steps produce the coefficients at different scales by different sections of the signal. The higher scales correspond to the most "stretched" wavelets. Figure 6.6 shows that the more the wavelet is stretched, the longer the portion of the signal with which it is being compared, thus the coarser the signal features being measured by the wavelet coefficients.

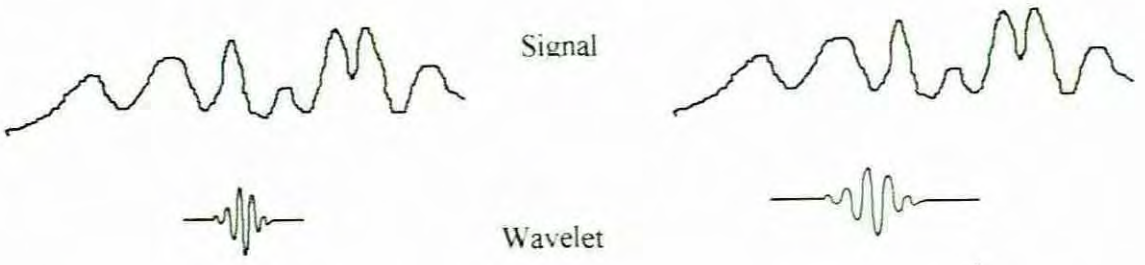


Figure 6.6 Streching the wavelet corresponds to coarser signal measurements

There is a correspondence between wavelet scales and frequency as revealed by wavelet analysis:

- Low scale a $\Rightarrow$ Compressed wavelet $\Rightarrow$ Rapidly changing details $\Rightarrow$ High frequency ω

- High scale a $\Rightarrow$ Stretched wavelet $\Rightarrow$ Slowly changing, coarse features $\Rightarrow$ Low frequency ω

Changing wavelet scale is mathematically represented as $\frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right)$ and shifting or translating as $\psi(x - b)$. The combination is then represented as $\frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right)$.

6.3.4 *Continuous wavelet transforms* (CWT) and Discrete wavelet transforms (DWT).

There are two types of wavelet transforms, namely continuous and discrete.

Both actually operate in discrete time, because any signal processing performed on a computer using real-world data must be performed on a discrete signal which is measured at discrete time intervals. The difference is that continuous wavelet transforms (CWT) can operate at every scale, from that of the original signal up to some maximum scale, whereas the discrete wavelet transform (DWT) uses only a subset of scales and positions at which to make the calculations. Calculating wavelet coefficients at every possible scale, as for the CWT, is a fair amount of work and it generates a lot of data. For the DWT, scales and positions based on powers of two are chosen. The analysis will then be much more efficient and just as accurate. Scales and positions based on powers of two are called the dyadic scales and positions.

Table 6.1 shows the relation between the various analyses summarised by Misiti, et al (1996: 6-13).

Continuous time "Continuous"	Continuous time "Discrete" analysis	Discrete time (Δ=1) "Discrete" analysis
$C(a,b) = \int_R s(t)\frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right)$	$C(a,b) = \int_R s(t)\frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right)$	$C(a,b) = C(j,k) = \sum_{n \in Z} s(n) g_{j,k}(n)$
$a \in R - \{0\}, b \in R$	$a = \Delta 2^j, b = \Delta k 2^j, (j,k) \in Z$	$a = 2^j, b = k2^j, j \in N, k \in Z$

Table 6.1 Three different analyses

In table 6.1 Δ is the sampling period. In discrete wavelet transforms g is the wavelet filter and plays the role of ψ. The scaling filter is represented by h in DWT and plays the role of

ϕ. For $j \in N$, $k \in N$, $n \in Z$ and a scale equal to 2^j the function g is defined as: $g_{j,k}(n) = 2^{-j/2} g(2^{-j} n - k)$.

6.3.5 Wavelets and Hearing loss simulation

Wavelets can be employed in this application as follows. An audio signal is broken up into different frequency bands. After the frequencies are separated, the signal is attenuated according to the audiogram information. A process called decompositioning is responsible for breaking up the signal into different frequencies. The required attenuation can be accomplished in two ways. In the first, corresponding points on the audiogram are multiplied with the new derived coefficients after decompositioning. These attenuated coefficients are then reconstructed. Another way is to do a complete frequency separation and then multiply the separate discrete signal values with the corresponding audiogram values. These attenuated signals at different frequencies are then combined to form the composite signal.

The algorithm investigated for the implementation of a hearing loss simulator is the Mallat algorithm. This scheme is known as a *two-channel subband coder*. The basic *two-channel subband coder* can be iterated to achieve the required frequency resolution. The decomposition and reconstruction will be discussed first, followed by the implementation.

6.3.6 Decomposition

The above-mentioned filtering algorithm yields a *fast wavelet transform*. Signals come into the coder, which output wavelet coefficients. The basic *two-channel subband coder* shown in figure 6.7 is a one stage filter which breaks the signal up into approximations (A) and detail (D). The approximations are the low-frequency components of the signal and come from the low pass filter (lpf). The details are the low-scale, high-frequency components and come from the high pass filter (hpf). When a digital signal is filtered, each filter will contain the same number of samples as the original digitised signal. This will double the amount of data. If each filter takes every second data point, then the total number of data points will be the same as in the original signal. Taking every second data point is called down

sampling. Down sampling introduces aliasing because of the fact that frequency has now effectively doubled. The cancellation of the aliasing will be dealt with later in the document.

To realise hearing loss simulation, the minimum requirement would be to have filter banks relative to the discrete frequency points on the audiogram. To decompose the signal into the relative frequency bands, the basic *two-channel subband coder* can be iterated accordingly. This process is called multilevel Decompositioning or multilevel analysis. The decomposition process is iterated, with successive approximations being decomposed in turn, so that one signal is broken down into many lower-resolution components.

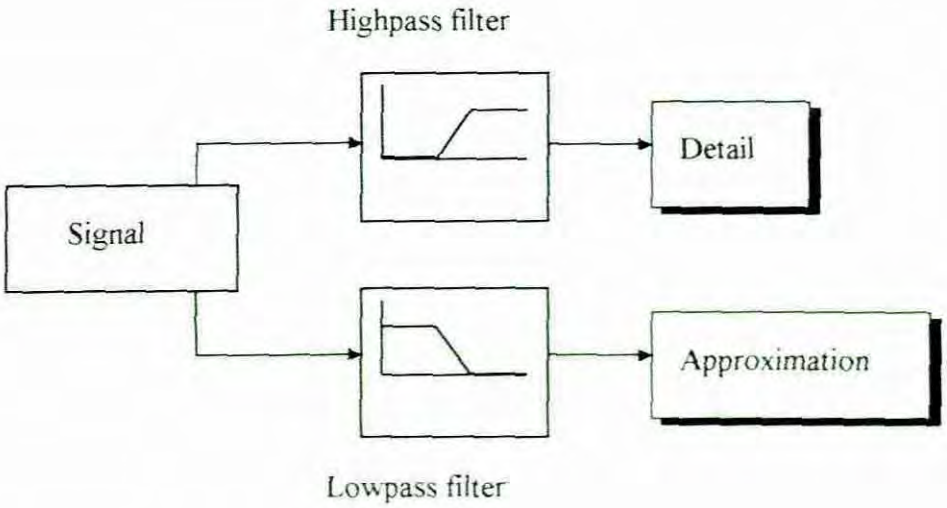


Figure 6.7 Two channel subband coder

Wavelet decomposition produces a family of hierarchically organised decompositions. The selection of a suitable level for the hierarchy will depend on the trade-off between the frequency resolution needed to mimic hearing loss simulation and the lowest resolution that is practically implementable considering the hardware specifications for real time operations.

Figure 6.8 shows the hierarchical decomposition. The symbol ↓ with a circle around it represents down sampling. At each level j, the j-level approximation, A_j is built, or approximation at j and a deviation signal called the j-level detail , D_j, or detail at level j. The original signal is the approximation at 0, denoted by A_0. The word "Approximation" is

justified by the fact that A_1 is an approximation of A_0 taking into account the "low frequencies" of A_0, whereas the detail D_1 correspond to the "high frequency" correction. The organising parameter, the scale a, is related to level j, by $a = 2^j$. The resolution is related to the scale. The resolution therefore decreases as the scale decreases. The lower the resolution, the smaller and finer are the details that can be accessed.

Looking at Figure 6.8 it can be seen that there are several ways to decompose the signal for example:

$$S = A_1 + D_1$$
$$= A_2 + D_2 + D_1$$
$$= A_3 + D_3 + D_2 + D_1$$

In practice this decomposition process can be iterated infinitely. Apart from limits imposed by the hardware, it would not make sense to decompose the signal into finer resolutions than those the ear can detect. In this case the characteristics of the ear will determine in which way and to what level the signal needs to be decomposed.

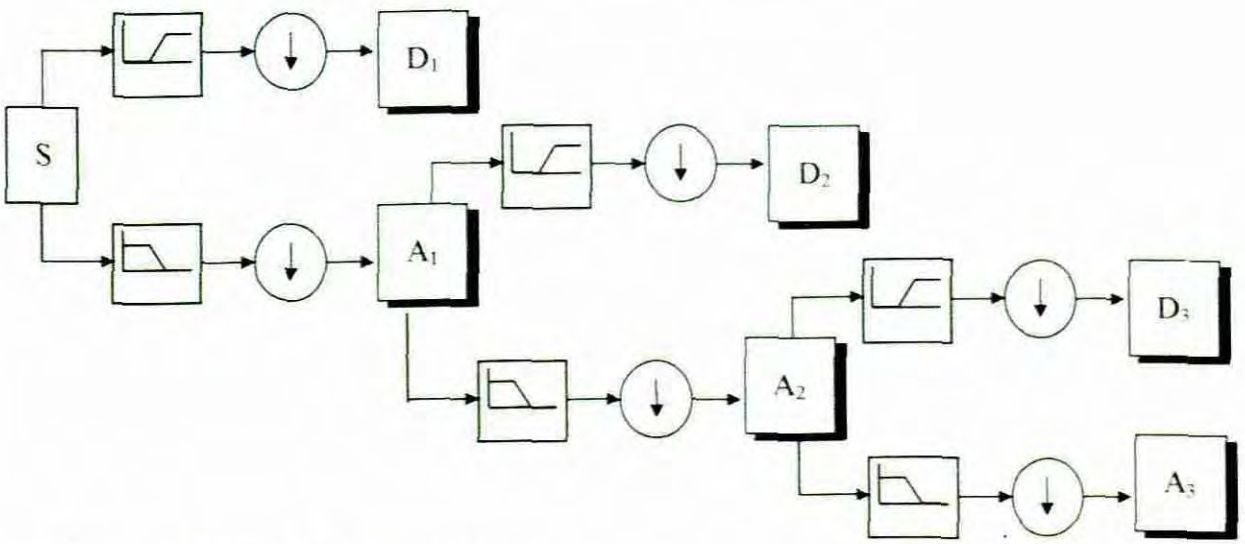


Figure 6.8 Multilevel decomposition

6.3.7 Reconstruction

Reconstruction (also called synthesis) is the process by which the decomposed components are assembled back into the original signal with no loss of information. Figure 6.9

graphically shows the reconstruction process. The symbol ↑ with the circle around it represents upsampling. It can be seen that where analysis involves filtering and down sampling, the wavelet reconstruction involves upsampling and filtering. The wavelet coefficients are used to synthesise a signal. Referring to Equation 6.1 the wavelet coefficients b_{jk} are convoluted with the wavelet to reconstruct the original signal:

$$f(t) = \sum_{j,k} b_{jk}\psi_{jk}(t).$$

Where $\psi_{jk}(t) = \psi(2^j t - k)$

This is called the inverse transform.

Without the attenuation, an important requirement for the reconstruction filter is that perfect reconstruction (PR) must be achieved. Aliasing can be cancelled when conditions of perfect reconstruction are met. Daubechies(1992) has shown that with the right choice of reconstruction and decompositioning filters perfect reconstruction can be achieved. Filters in a synthesis bank must be specially adapted to cancel errors generated in the analysis filter bank. When orthogonal basis is used the analysis and synthesis filters will have symmetry. This is one of the essential conditions for PR.

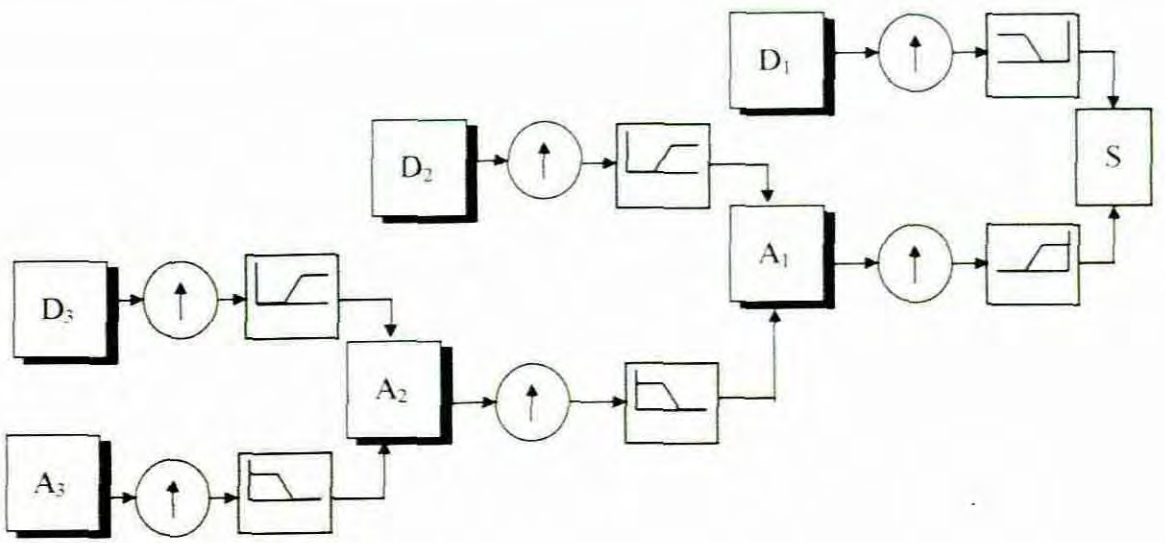


Figure 6.9 Reconstruction filter

A two-channel structure is used to explain how these filters relate. The various filters are derived from a filter arrangement which is called quadrature mirror filters (qmf). The low pass reconstruction filter (Lprf) will be the quadrature mirror of the Hprf. It means that the

coefficient sequence of Lprf will be reversed and every alternative value will be inverted. For example if Lprf has a coefficient sequence of (d, c, b, a), then the Hprf will have a sequence of (-a, b, -c, d) as illustrated in Figure 6.10. The filters in the analysis bank will be the inverse of their synthesis counterpart. This means that the Hprf will be the inverse of the high pass decomposition filter (Hpdf) and Lprf will be the inverse of the low pass decomposition filter (Lpdf).

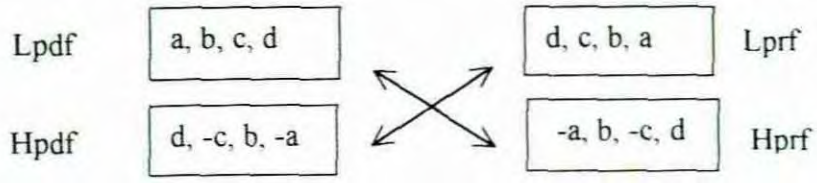


Figure 6.10 Orthogonal filter bank with four coefficients

Perfect reconstruction can also be accomplished if the filterbank is biorthogonal as shown in figure 6.11 where the synthesis bank inverts the analysis bank and the banks are not necessarily the same length.

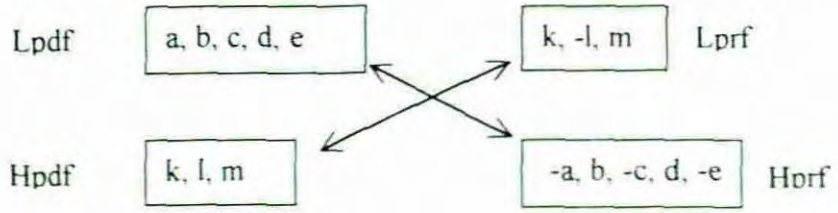


Figure 6.11 Biorthogonal filter bank shows invertability but not symmetry

6.3.8 Details and approximation

It has been said that for DWT the analysis begins with the signal s and results in the coefficients $C(j,k)$, whereas the synthesis starts with coefficients $C(j,k)$ and results in the signal s. Equation 6.1 could be written as $s(t) = \sum_{j \in Z} \sum_{k \in Z} C(j,k) \psi_{j,k}(t)$.

If j is fixed and the summation is done on k, then the Details at a specific level are defined as:

$$D_j = \sum_{k \in Z} C(j,k)\psi_{j,k}(t) \qquad (6.3)$$

Performing the summation now on j, shows that the signal is the sum of all the Details:

$$s = \sum_{j \in Z} D_j \qquad (6.4)$$

If the signal is said to consist only of details, then some of the details must be classified as finer detail and some as coarser details. A reference level J can be taken to make this distinction. Those details for which $j \leq J$ can be defined as the finer details or just the details. In other word those details at scales $a = 2^j \leq 2^J$. Those details for which $j > J$ can be defined as the coarser details or the approximations and can be written as:

$$A_J = \sum_{j>J} D_j \qquad (6.5)$$

The signal s can now be written as $s = A_J + \sum_{j \leq J} D_j \qquad (6.6)$

Equation 6.6 means that the signal s is the sum of its approximation A_J and its fine details. It is obvious that the approximations are related to one another by:

$$A_{J-1} = A_J + D_J \qquad (6.7)$$

6.3.9 Wavelets, scaling functions and Filters

The construction of a wavelet normally starts with the construction of the appropriate filter bank. Normally the low pass filter is designed first. The high pass filter is derived from the low pass filter. Wavelets come from the iteration of filters. In the case of qmf, the high pass reconstruction filter will determine the shape of the wavelet and also produce the coefficients of the wavelet decomposition. The low pass reconstruction filter produces the scaling function ϕ. The scaling function comes from iterating the lowpass filter and with rescaling at each iteration (Strang & Nguyen,1996:3).

If the low pass filter coefficients are h and the high pass filter is g, Vetterli and Kovacevic (1995:238) prove that

$$\phi(t) = 2^{j/2} h^{(j)}[n] \qquad \frac{n}{2^j} \leq t < \frac{n+1}{2^j}$$

$$\psi(t) = 2^{j/2} g^{(j)}[n] \qquad \frac{n}{2^j} \leq t < \frac{n+1}{2^j}$$

The scale and position of the scale function ϕ and wavelet ψ are a function of j and k and the normalised functions can be defined as:

$$(j,k) \in Z^2 \; : \; \psi_{j,k} = 2^{-j/2}\psi(2^{-j}x - k), \quad \phi_{j,k} = 2^{-j/2}\phi(2^{-j}x - k)$$

A fundamental relationship between two successive scales is commonly known as the twin-scale relation. Misiti, et al (1996:6-28) define the twin-scale relation for ϕ and ψ as follows

For ϕ the twin-scale relation is $\phi_{1,0} = \sum_{k \in Z} h_k \phi_{0,k}$ and $\phi_{j+1,0} = \sum_{k \in Z} h_k \phi_{j,k}$.

For ψ the twin-scale relation is $\psi_{1,0} = \sum_{k \in Z} g_k \phi_{0,k}$ and $\psi_{j+1,0} = \sum_{k \in Z} g_k \phi_{j,k}$.

6.3.10 Wavelet families

The construction of wavelets is a specialised study. The aim of the research is to find the best way for implementing *HLS*. It will be shown in the section on the implementation that using wavelets is a less favourable method for realising hearing loss simulation. Thus documenting the underlining mathematics and theory would fall outside the framework of this research. A summary of existing wavelet families and their associated properties is given in appendix D. The wavelet families included are: Morlet, Mexican Hat (mexh), Meyer, Haar, Daubechies (dbN), Symlets (symN), Coiflets (coif), Splines biorthogonal (BiorNr.Nd) wavelets where the capital N at the end of the abbreviations is the order. Very few wavelets have an explicit analytical expression. Exceptions are the Morlet, the Mexican Hat and wavelets that are piecewise polynomials such as the Haar. Wavelets such as the Daubechies are defined by their functional equations. An in depth study on wavelet theory can be found in *Ten Lectures on Wavelets* by Ingrid Daubechies (1992). The different wavelets can be generated and plotted with the Matlab Wavelet toolbox. A further discussion and an excellent supplementary text with the use of Matlab's wavelet toolbox is the book, *Wavelets and Filter Banks* by Gilbert Strang and Truong Nguyen.

6.4 Implementation

As previously stated a practical approach will be taken. As such, the effect of different wavelets was tested on a given signal, making use of Matlab. The space would not allow the documentation of all simulations, but selected examples will be included for the

purposes of demonstration. The first step in the decompositioning is to break the signal into high and low frequencies. *Matlab has built-in functions such as DWT to do the discrete* wavelet transform. In figures 6.12 and 6.13 wavelets db3 and db8 were used to decompose a signal x(n) = sin(2*pi*250*n/Fs) + sin(2*pi*6000*n/Fs). The program to generate these plots can be founded in Appendix E. In the program a sampling rate of 16 kHz is used. The instruction *dwt* in the program decompose the signal into Approximations which will have a range of 0 – 4 kHz and Details with a range of 4 –8 kHz where the Nyquist frequency is 8 kHz. To test the principle of this method, the coefficients of the Detail are multiplied by zero. In other words an attempt is made to eliminate the high frequency component at 6000Hz by multiplying the Detail by zero. The HLS will have to operate at this stage, between the decompositioning and reconstruction. In other words, the information from the audiogram will be taken and multiplied with coefficients after decompositioning. To continue with the above test the coefficients had to be constructed, this time with Detail equal to zero and Approximations unattenuated. The *idwt* instruction in the program is used to reconstruct the signal. The result can be viewed in figures 6.12 and 6.13 which shows that the 6 kHz part of the signal is removed which proves that the method of subband decoding will work in principle for this application. It can also be seen that the db8 results in a smoother response. In this project it would be standard practice to test different wavelets for the best result beforehand.

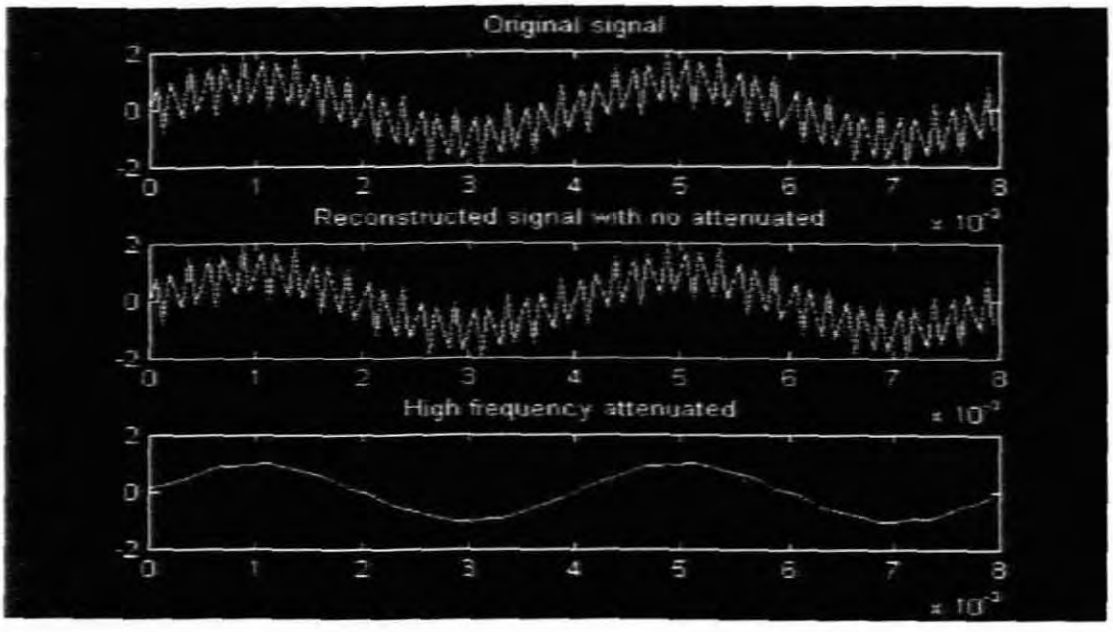


Figure 6.12 Wavelet : db8

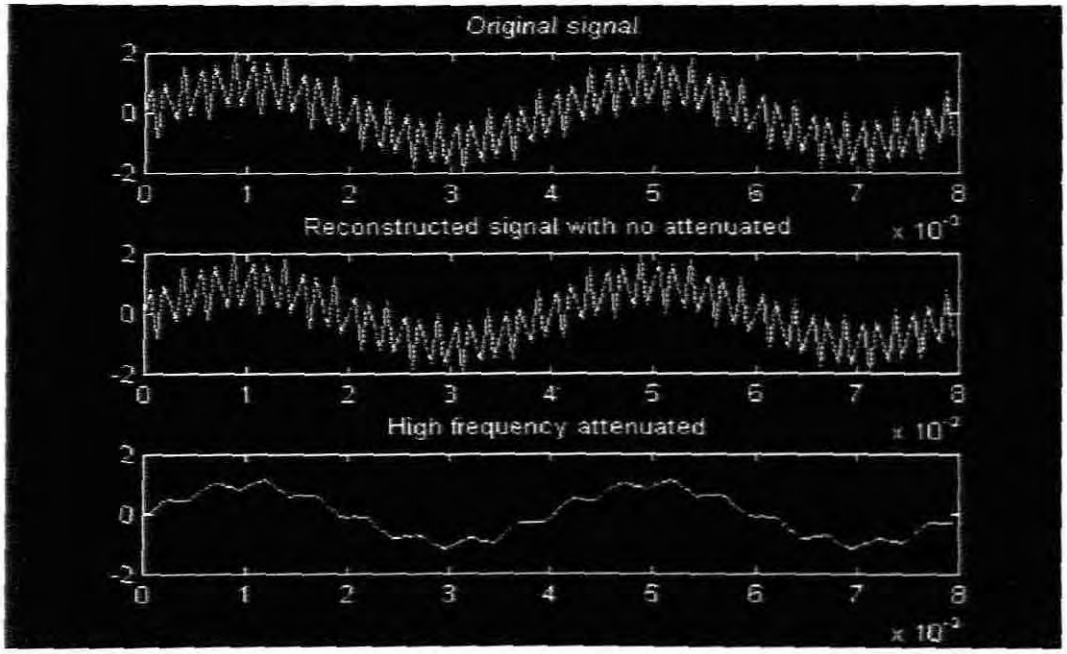


Figure 6.13 Wavelet db3

For HLS an audio spectrum of 16000 Hz will be split into frequencies above and below 8000 Hz which are the details and approximations respectively in the first stage or level. The second stage of decomposition will split the lower 8000 Hz at 4000 Hz. The third stage will be split at 2000 Hz. This process needs to be continued until the same frequency bands as recorded in the audiogram are achieved, as illustrated in chapter 3 figure 3.2. It should be noticed that by the nature of the multiresolution decompositioning using wavelets in the subband decoding process, the signal can be split into octaves, which resembles the characteristics of the ear. Decomposition according to figure 3.2 will go to the sixth level as illustrated in figure 6.14.

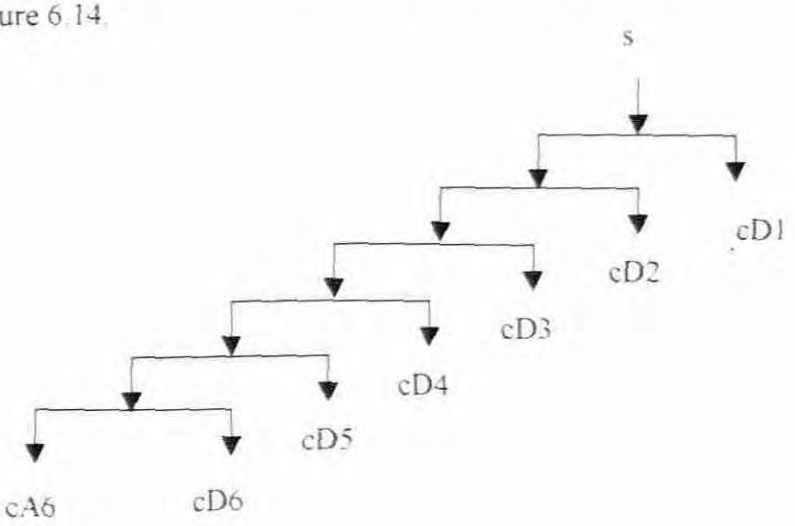


Figure 6.14 Multiresolution decompositioning to level j=6

A program was written in Matlab to demonstrate multiresolution decompositioning to level j = 6 and this can be found in appendix E. In this program a signal consisting of six sinusoids (located in the centre of each band) was constructed. All have the same amplitude of one and frequencies at 125, 375, 750, 1500, 3000 and 6000 Hz respectively. The aim was to find a wavelet that produced the clearest spectral response when the signal was attenuated according to the audiogram. Figure 6.15 shows the original signal and the attenuated signal using the wavelet db2. The drawings on the left show the time domain representation of the original and the attenuated signal. Those on the right show the frequency response generated by Matlab's built in Fast Fourier transform function. From all the wavelets tested in Matlab's toolbox, wavelet db2 has produced the best results. A few comparisons are given in Figure 6.16. In figure 6.16 the first drawing (a) shows the spectrum when db2 is used. (b) shows unwanted frequency components when using db3 and (c) shows that the Haar wavelet produces more distortion. The actual wavelet db2 is shown in (d) with a refinement of 10 iterations. The coefficients of each filter is stored in Matlab and figure 6.16(e) shows the plot of the coefficients of the reconstruction low-pass filter of db2. (f) shows the plot of the coefficients of the reconstruction high-pass filter of db2. As stated earlier, the decomposition high and low-pass filters can be deduced from the quadrature mirror arrangement, which was explained earlier.

It can be seen that this method would result in the same step response as explained in chapter 3. In other words, by applying multilevel-decompositioning in the Wavelet method, the signal can be broken up into different frequency components by a bank of filters in a way similar to that of the Filterbank method discussed in chapter 3. It thus has the same drawbacks as discussed in section 3.6, except that perfect reconstruction will improve on the problem encountered with the overlap of adjacent filter banks. Looking back at figure 3.10, an improvement can be made if the frequency bands, which are the width of each step, are split up into further details and approximations. This process of subcoding a signal into arbitrary shape decompositioning structures is called wavelet packeting. (For further reading on wavelet packeting refer to Strang and Nguyen (1996:72)).

Concentrating on a certain frequency area is called "Zooming" onto that area. This is an excellent feature when only speech needs to be processed and the frequency band 500 Hz – 2000 Hz, where most of the speech frequencies are concentrated, can be zoomed into.

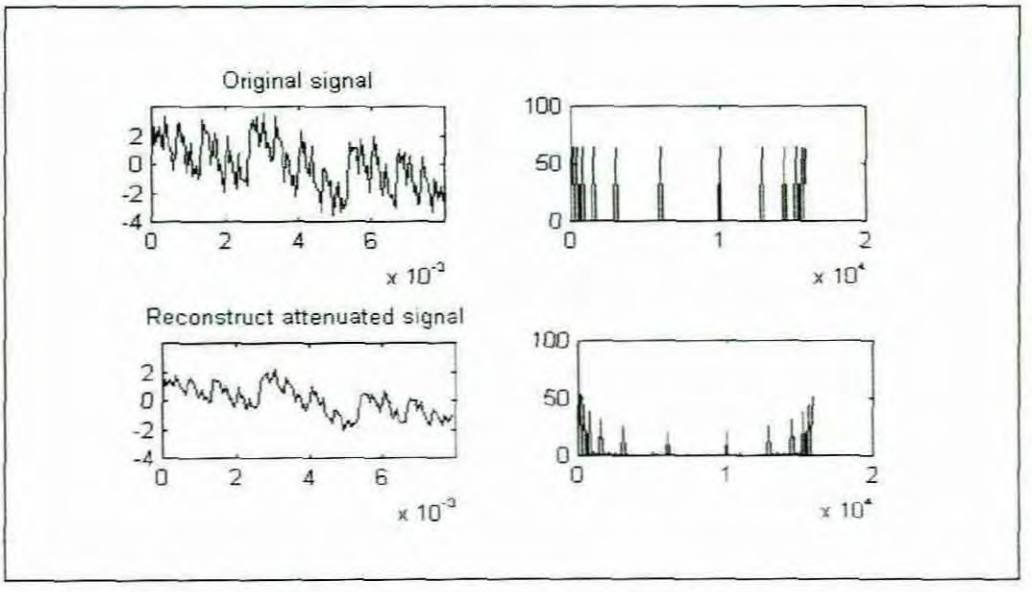


Figure 6.15 Attenuation with signal using wavelet Daubechies :db2

It should be clear that to achieve the same smooth response as for the IFFT a tremendous number of subbands would be required. As said earlier, the IFFT is evaluated at equally spaced points on the frequency axis. The processing of information at frequency ranges which are undetectable to the ear is, in fact, unnecessary. A fair number of subbands can thus be eliminated when the wavelet method is considered. Even if half of the subbands are removed when compared with 256 IFFT, a decompositioning of up to level j = 128 would be required, which is still a heavy computational burden. The number of multiplications between each samples will be equal to the N x j, where N is the length of the filter and j the level of decompositioning. There will be an equal number of summations during each sample period. The number of computations will now add up to 2 x N x j per filter. There is as previously explained a 4 filter arrangement in each stage (level). When the wavelet db2 is used, which has a filter length of 4, the total number of computations is 4(2 x 4 x 128) = 4096 between samples (or between each sample and the next). The proposed hardware uses a DSP processor (ADSP2181) that runs at 33mips and a programmable CODEC configured for a sampling rate of 32 kHz. This means that between samples $(33 \times 10^6)/(32 \times 10^3) = 1031$ instructions can be processed. Clearly it can be seen that it would be impossible to decompose up to this level with the proposed hardware. The highest level to which could be decomposed using the db2 wavelet would be j = 32. If any other wavelet with a higher filter order is used, the level of decompositioning would be less.

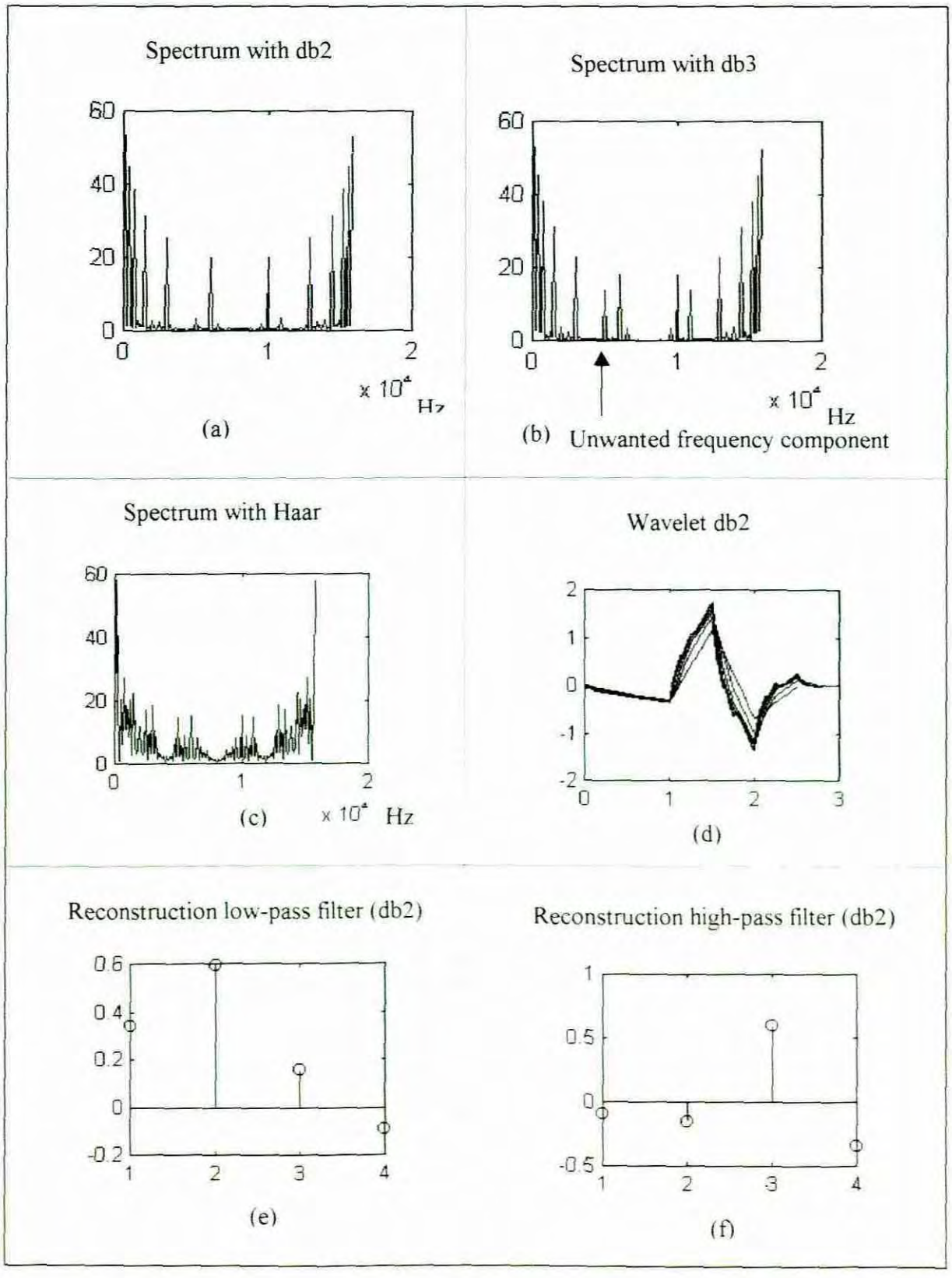


Figure 6.16 (a) Spectrum of db2. (b) unwanted frequency components when using db3. (c) Haar wavelet has more distortion. (d) Refinement of db2 after 10 iterations. (e) Reconstruction low-pass filter of db2. (f) Reconstruction high-pass filter of db2.

6.5 Conclusion

It can be concluded that it would be rather difficult to achieve the same frequency resolution for the DWT as for the IFFT, considering the computational load imposed by the DWT method. The advantage of the IFFT method in an application of this nature is that the inverse transforms do not have to be done in real time. The IFFT computation can be done beforehand on a PC and the derived impulse response can then be used directly in the convolution operation on the DSP side. With the proposed hardware the IFFT method will offer a smaller frequency resolution than the DWT method, which will make the IFFT method preferable for implementation.

To eliminate the unnecessary calculations in the IFFT at points that are undetectable by the ear, a method called the inverse chirp Z transforms can be used, which leads to the next chapter.

CHAPTER 7

CHIRP-Z TRANSFORMS

7.1 Introduction

In chapter 4 the DFT was introduced, followed by the FFT algorithm in chapter 5. The relation between the transforms and their inverse transforms were pointed out. In this chapter the aim is to reveal a method that parallels the DFT but which might have similar characteristics to the human ear. The resemblance between the chirp-z transform (CZT) and the frequency response of the ear necessitates a study of the method. It was shown by Fourier transforms that just changing two variables could derive the inverse transforms. The assumption is that the same transformation can be done upon the chirp-z transforms. The approach would thus be to show the derivation of the chirp-z transform of a finite-duration signal and then apply the transformation maneuver to convert to its inverse chirp-z transform (ICZT). The inverse chirp-z transform is important, since this will produce the impulse response necessary for the convolution in the filtering procedure. Although Matlab simulations will demonstrate the basic principles of chirp z-transforms, this chapter assumes a basic knowledge of z-transforms.

7.2 The chirp-z transform (CZT) algorithm

The discrete Fourier transform (DFT) of an N-point data sequence $x(n)$ is equal to the z-transform of $x(n)$ evaluated at N equally spaced points on the unit circle. It could also been viewed as N equally spaced samples of the Fourier transform of the data sequence. The CZT, on the other hand, evaluates the z-transform of finite duration signal along certain general contours in the z plane. With the CZT the z transform of a signal may be evaluated at points not necessarily equally spaced on the unit circle in the z plane. A similar formalization to the general description given above is given by Krauss, Shure and Little (1994:1.56) in the User's Guide of Matlab's *Signal Processing Toolbox* as:

> " The chirp z-transform, or CZT, computes the z transform along spiral contours in the z-plane for an input signal. Unlike the DFT, the CZT is not constrained to operate along the unit circle, but evaluates the z-transform along contours described by
>
> $$z_l = AW^{-l}, \quad l = 0,\ldots M-1 \qquad (7.1)$$
>
> Where A is the complex starting point, W is a complex scalar describing the complex ratio between points on the contour, and M is the length of the transform."

Equation 7.1 can further be expanded mathematically by defining

$$A = r_0 e^{2\pi\theta} \qquad (7.2)$$

$$W = R_0 e^{2\pi\phi} \qquad (7.3)$$

with r_0 and R_0 being positive real numbers and ϕ and θ arbitrary real numbers

The z-transform of a sequence $x(n) = 0, 1, \ldots N-1$ can be computed at a set of points z_l as :

$$X(z_l) = \sum_{n=0}^{N-1} x(n) z_l^{-n} \qquad l = 0, 1, \ldots M-1 \quad (7.4)$$

where: $$z_l = AW^{-l} = r_0 e^{j2\pi\theta_0} (R_0 e^{j2\pi\phi_0})^l \qquad l = 0, 1 \ldots M-1 \qquad (7.5)$$

If $R_0 > 1$, the contour spirals outward as n increases. If $R_0 < 1$, the contour spiral toward the origin as n increases. If $R_0 = 1$, the contour is a circular arc with r_0. The constants r_0 and θ_0 determine the radius and angle of the starting point. With $R_0 = 1$ and $r_0 = 1$, the contour is a circular arc of the unit circle and would allow the frequency content of a sequence at a dense set of frequencies to be computed. The values of M, ϕ_0 and θ_0 will determine the range that will be covered by the arc. Obviously, if a dense set of frequencies is concentrated on, the whole DFT need not be computed, in which case the DFT of the sequence x(n) is padded with zeros. It could be seen that when $R_0 = 1$, $r_0 = 1$, $\theta_0 = 0$, $\phi_0 = 2\pi/N$ and $M = N$ the CZT reduces to the DFT and the contour is the entire unit circle.

7.3 Demonstrating the evaluation of the CZT in the z-plane

The above is demonstrated graphically in figure 7.1 by a simple program written in Matlab. The program is listed in program list 7.1 and the variables R_0, r_0, ϕ_0, θ_0 and M is changed for each plot. The first plot figure 7.1(a) shows that the DFT is just a special case of the CZT. (b) shows the contour of a concentric circle inside the unit circle. (c) and (d) illustrate that the contour spirals either in or out, depending on the value of R_0

% Program in Matlab to demonstrate CZT in the z-plane

```
close all
  Ro = 1;                          % Ro determines the movement of the arc
  ro = 1;                          % Radius of starting point
  M = 20;
  l = 0:M-1;                       % Number of frequency points
  A = ro*exp(j*pi/6);
  W = Ro*exp(-j*2*pi/M);
  z = A*(W.^(-l));
```

```
subplot(2,2,1);zplane([],z.')          % Plot points in z-plane
```

Program list 1: Program to plot CZT in z-plane

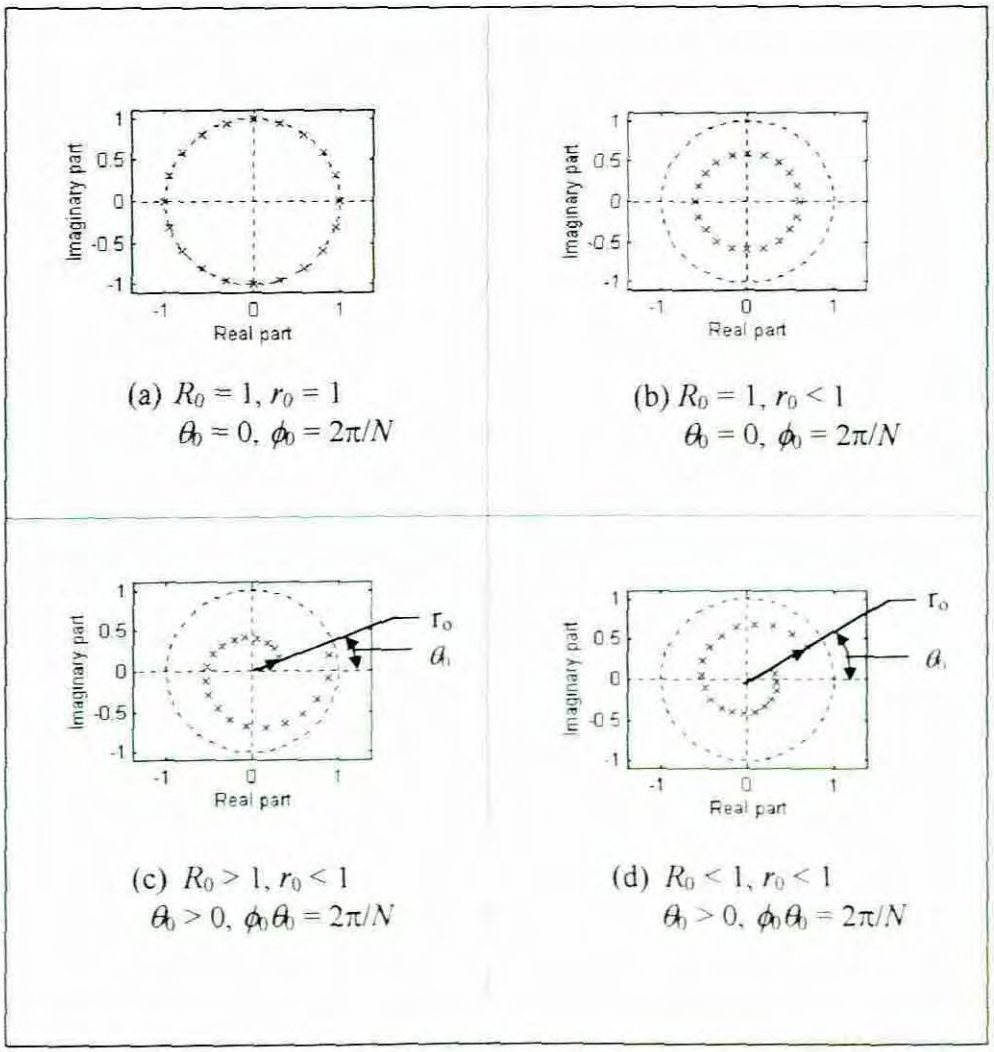


Figure 7.1 Examples of contours, which may evaluate the z-transform

7.4 The Chirp-z transform evaluation

In the previous sections the contours are defined and described. The objective of this section is to derive a formula to evaluate the CZT efficiently along these contours. The most important feature behind the chirp-z transform is the ability to represent a frequency of an exponential signal which increases linearly with time. This feature is known as a chirp in radar systems and that is where its name is derived from. The goal would thus be to prove this characteristic.

Equation 7.5 can be substituted into 7.4 to give:

$$X(z_l) = \sum_{n=0}^{N-1} x(n) z_l^{-n} \qquad l = 0, 1, \ldots M-1$$

$$= \sum_{n=0}^{N-1} x(n) A^{-n} W^{-nl} \qquad (7.6)$$

$$= \sum_{n=0}^{N-1} x(n) (r_0 e^{j2\pi\theta_0})^{-n} W^{-nl} \qquad \text{where} \quad A = r_0 e^{j2\pi\theta} \qquad (7.7)$$

Equations 7.6 and 7.7 can be manipulated in such a way so as to reduce the number of complex multiplications. These equations can be written in a form of convolution by substituting

$$nl = \frac{n^2 + l^2 - (l-n)^2}{2} \qquad (7.8)$$

in Equation 7.7, to give $X(z_l) = \sum_{n=0}^{N-1} x(n) (r_0 e^{j2\pi\theta_0})^{-n} W^{-\frac{n^2}{2}} W^{-\frac{l^2}{2}} W^{\frac{(l-n)^2}{2}}$ (7.9)

$$= W^{-\frac{l^2}{2}} \sum_{n=0}^{N-1} \left[x(n)(r_0 e^{j2\pi\theta_0})^{-n} W^{-\frac{n^2}{2}} \right] W^{\frac{(l-n)^2}{2}} \qquad (7.10)$$

let $g(n) = x(n)(r_0 e^{j2\pi\theta_0})^{-n} W^{-\frac{n^2}{2}}$ (7.11)

and substitute g(n) into Equation 7.10, to give

$$X(z_l) = W^{-\frac{l^2}{2}} \sum_{n=0}^{N-1} g(n) W^{\frac{(l-n)^2}{2}} \qquad (7.12)$$

if $h(n) = W^{\frac{n^2}{2}}$ (7.13)

then $h(l-n) = W^{\frac{(l-n)^2}{2}}$ (7.14)

substituting Equation 7.14 in Equation 7.12 gives

$$X(z_l) = W^{-\frac{l^2}{2}} \sum_{n=0}^{N-1} g(n) h(l-n) \qquad \text{for } l = 0, 1 \ldots M\text{-}1 \qquad (7.15)$$

The summation in Equation 7.15 can be interpreted as the convolution of the sequence $g(n)$ with the impulse response $h(n)$ of a filter. Equation 7.15 is the formula to evaluate the CZT that was the objective to be derived.

Another way of writing (7.15) is $X(z_l) = W^{-\frac{l^2}{2}} y(l)$ for $l = 0, 1 \ldots M\text{-}1$ (7.16)

where $$y(l) = \sum_{n=0}^{N-1} g(n)h(l-n) \qquad (7.17)$$

or $$X(z_l) = \frac{y(l)}{h(l)} \quad \text{for } l = 0, 1 \ldots M\text{-}1 \qquad (7.18)$$

where $$h(l) = W^{\frac{l^2}{2}} \quad \text{and} \quad W^{\frac{-l^2}{2}} = \frac{1}{h(l)} \qquad (7.19)$$

The special characteristic of the CZT, to increase linearly in frequency with time, can be explained as follows.

Substituting $W = R_0 e^{2\pi\phi}$ in Equation 7.13 gives $h(n) = R_0 e^{\frac{(2\pi n\phi)n}{2}}$ (7.20)

If $R_0 = 1$ in the sequence $h(n)$, then it will be noticed that this sequence has the form of a complex exponential with argument $\omega n = (\pi n\phi)n$, which is a linear increasing frequency. This concludes the proof of the special characteristic of the CZT to increase linearly in frequency with time.

7.5 Detail of the CZT operation

The FFT algorithm can be employed to do the linear convolution in Equation 7.15. As such, the convolution summation in Equation 7.15 can be evaluated as a circular convolution of the two sequences $g(n)$ and $h(n)$ through the FFT. Ludeman (1986:267), gives a good reference to circular convolution in *Fundamentals of Digital Signal Processing*. The convolution summation can be performed as follows

$$g(n) \circledast_M h(n) = IDFT_M(DFT_M[g(n)] \times DFT_M[h(n)]) \qquad (7.21)$$

where $\circledast_M$ is the symbol for circular convolution and M the modulus

The chirp-z transform can then be performed by

$$X(z_l) = W^{\frac{-l^2}{2}} \times IDFT_M(DFT_M[g(n)] \times DFT_M[h(n)]) \qquad (7.22)$$

A problem that arises is that the two sequences in the summation are not the same length. The length of $g(n)$ is N and that of $h(l\text{-}n)$ could be infinity. Using the notation $h(n)$ instead of $h(l\text{-}n)$, it is noticed that only a portion of $h(n)$ needs to be used. Let P be the length of $h(n)$ such that $P > N$. If the two sequences are circularly convoluted, the first $N - 1$ point will contain aliasing. The remaining $P - (N - 1)$ contains the values that would have been obtained from a linear convolution. Thus, the size of P needs to be such that it contains

both the valid as well as the aliased values. So if M is equal to the length of the valid points then $P = M + N - 1$. The section of $h(n)$ that is needed for the computation is $-N + 1 < n < M - 1$ which means that the sequence needs to be shifted to start from zero. In this case a new sequence can be defined as

$$h_1(n) = h(n - N + 1) \quad \text{for} \quad n = 0, 1, \ldots P - 1$$

The P point DFT of $h_1(n)$ can be calculated through the FFT to give $H_1(l)$

The sequence $g(n)$ defined by Equation 7.11 is padded with M - 1 zeros and then the DFT (FFT) is performed on the new sequence to give $G(l)$. $G(l)$ will thus also be a P point DFT sequence.

The IDFT of the product $Y_1(l) = H_1(l)\, G(l)$ (7.23)

yields the P point sequence $y_1(n)$, $n = 0, 1, \ldots P - 1$. The first N - 1 points of $y_1(n)$ are infected by aliasing and are rejected. The desired values are $y_1(n)$ for $N - 1 \le n \le P - 1$ which correspond to the range $0 \le n \le M - 1$ so that

$$y(n) = y_1(n + N - 1) \quad \text{for } n = 0, 1, \ldots M - 1.$$

$y(n)$ can now be written as $y(l)$ since $n = l = 0, 1, \ldots M - 1$.

Finally, the value of $X(z_l)$ can be calculated by $X(z_l) = \dfrac{y(l)}{h(l)}$ as defined by (7.18)

where $h(l) = W^{\frac{l^2}{2}}$ as defined by (7.19) and $y(l)$ as defined by (7.17)

In the above operation the number of complex multiplications are, in general, in the order of $P \log_2 P$ where $P = M + N - 1$. Refer to chapter 5 for details on the calculation of the FFTs but in short, there are two DFTs (FFTs) one IDFT (IFFTs), plus P complex multiplications for (7.23) and M complex multiplications for (7.18). If this is compared with the direct convolution, which would be in the order of MN, it can be seen that if the sequence is small, the direct method would be better, but if the sequences are longer then the method described above would be much more efficient.

7.6 The DFT (FFT) versus the CZT

The CZT is more flexible than the DFT, as explained in the previous section.

This advantage of the CZT in comparison with the DFT will be explained by a practical example. The plot in Figure 7.2 was generated in Matlab and shows the difference between CZT and DFT. In this case the *czt* function was used to zoom in on a narrow band section

(100 to 150 Hz) of a filter frequency response. The program to perform this demonstration was taken from Krauss et al (1994:2-68) and is given in program list 7.2.

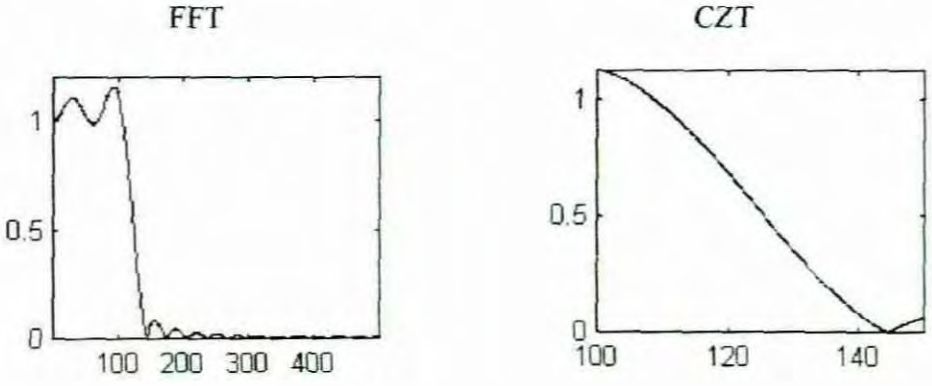


Figure 7.2 Comparing the FFT and the DFT

```
close all
  h =fir1(30,125/500,boxcar(31)); % design filter
  Fs = 1000; f1 = 100; f2 = 150;  % in hertz
  m = 1024;
  w =exp(-j*2*pi*(f2-f1)/(m*Fs));
  a =exp(j*2*pi*f1/Fs);
% Compute both DFT and CZT of the filter
  y = fft(h,1000);
  z = czt(h,m,w,a);
% Create frequency vectors and compare results
  fy = (0:length(y)-1)'*1000/length(y);
  fz = ((0:length(z)-1)'*(f2-f1)/length(z)) + f1;
subplot(2,2,1);plot(fy(1:500),abs(y(1:500)));axis([1 500 0 1.2])
subplot(2,2,2);plot(fz,abs(z)); axis([f1 f2 0 1.12])
```

Program list 7.2 Difference between CZT and DFT

7.7 How the CZT relates to the project

When the Chirp-z transform is used, the audiogram information must be presented in the same order as the CZT. This means that the frequency points at which the audiogram is sampled should not be linear, such as they are when the DFT (FFT) is used. Instead, the frequency sampling points should increase linearly. As for the DFT (FFT) method, the CZT can be implemented in two ways. The first is to perform the computation in real time in the frequency domain. In this way the sampled signal is converted to the frequency

domain by means of the CZT and then multiplied by the corresponding frequency samples of the audiogram. The product yields the attenuated spectrum, which has to be converted back to the time domain. The inverse transform thus has to be performed on the product. The procedures would be basically the same as those explained in chapter 4 section 4.8, but with the inverse CZT used instead. This method obviously performs many calculations in real time, which is a serious constraint in terms of the operating speed of the proposed hardware. The second method has already been discussed. This method is where the inverse transform is performed on the audiogram information and then convoluted with the sampled signal to produce the required filtering effect. The advantage, previously discussed, is that the inverse transforms need not to be calculated in real time. Calculating the inverse transform beforehand make this method extremely fast because only the computed impulse response is convoluted in real time.

7.8 The inverse chirp-z transform

The same relation in the forward and reverse transform are assumed for the CZT as for the DFT. The inverse relation for the DFT as derived in chapter 4 is

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j w_k n} \qquad \text{DFT} \qquad (7.24)$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j n w_k} \qquad \text{IDFT} \qquad (7.25)$$

where $\omega_k = 2\pi k / N \ \ for\ k = 0,1,2,\ldots,N-1$

It can be seen that the difference is that $x[n]$ and $X[k]$ are swapped and the IDFT is divided by N.

The CZT is given by (7.4) and (7.5) as

$$X(z_l) = \sum_{n=0}^{N-1} x(n) z_l^{-n} \qquad l = 0, 1, \ldots M-1 \qquad (7.26)$$

where: $z_l = AW^{-l} = r_0 e^{j 2\pi l / 0} (R_0 e^{j 2\pi \phi_0})^l \qquad l = 0, 1 \ldots M-1$

Equation 7.15 was derived for the CZT as

$$X(z_l) = W^{-\frac{l^2}{2}} \sum_{n=0}^{N-1} g(n) h(l-n) \qquad \text{for } l = 0, 1 \ldots M\text{-}1 \qquad (7.27)$$

where $g(n) = x(n)(r_0 e^{j2\pi\theta_0})^{-n} W^{-\frac{n^2}{2}}$ and $h(l-n) = W^{\frac{(l-n)^2}{2}}$

For this application the inverse transform should be done, where frequency samples are taken over the whole audio spectrum or the whole audiogram. If only a portion of the spectrum is taken then the unit impulse response of the filter will reflect only that portion of the spectrum and the filter will consequently be incorrect.

From inverse relation in (7.24) and (7.25), the inverse CZT is thus written as

$$x(n) = \frac{W^{-\frac{l^2}{2}}}{N} \sum_{l=0}^{M-1} g(n)h(l-n) \quad \text{for } l = 0, 1 \ldots M\text{-}1 \tag{7.30}$$

where $g(n) = X(z_l)(r_0 e^{j2\pi\theta_0})^{-n} W^{-\frac{n^2}{2}}$ and $h(l-n) = W^{\frac{(l-n)^2}{2}}$ (7.31)

With variables $r_0 = 1$ and $\theta_0 = 0$, $g(n)$ reduces to

$$g(n) = X(z_l) W^{-\frac{n^2}{2}} \tag{7.32}$$

7.9 Conclusion

In this chapter it was proven that the CZT could be evaluated on a spiral contour. There will be a high degree of similarity between the CZT and the frequency response of the ear because the CZT does the transform on a linear increasing frequency. The CZT might have an improvement in speed, if points which are not critical to the ear are eliminated.

There are, however, problems with the derivation of the ICZT. The first problem is that no theoretical proof is provided in this chapter to substantiate the correctness of the ICZT. Neither could any literature be found that confirms the legitimacy of the inverse procedure. This is an area that obviously needs further investigation if this method is to be considered for implementation. The first thing in such an investigation would be to test the ICZT practically, if no theoretical proof is given. Practically, a program must be written in Matlab to test the forward and inverse transforms. A suggestion was made to do the CZT on a finite time sequence to place it in the frequency domain and then transform it back by means of an IFFT to see if the same time sequence could be derived. Obviously this would not work because some of the information will be lost in the process. The IFFT takes frequency

samples at equally spaced sampling points, whereas the inverse CZT should take samples which are at linear increasing frequency points. Even if points on the frequency axis are calculated to have equal spacing (after the CZT computation is done), the values at those points will only be estimates because some form of estimation has to be performed to get these values. The correct way would be to transform back to the time domain at exactly the same points that were produced by the CZT, in which case it would be at linear increasing frequency points. To transform back to the time domain from these frequency points which are spaced at linear increasing frequencies, some form of mechanism has to be built into the ICZT that would account for there linear increasing frequency points, as been suggested by Equations 7.30, 7.31 and 7.32.

To verify this method is surely a study on its own and this will obviously hinder its implementation.

Another method that was proposed is the Yule-walker method and this will be discussed in the next chapter.

CHAPTER 8

MODIFIED YULE-WALKER METHOD

8.1 Introduction

In this research project the main aim is to design a filter. The frequency response of the filter should be the same as the frequency response of the ear. Various methods have been presented. Another important method that provides an excellent solution to the problem is the " YULEWALK Recursive filter design using a least-square method". The Yulewalk function is a built in function in MATLAB, which makes it especially attractive to this project. The filter coefficients could easily be calculated in MATLAB and then transferred to the DSP side where the filtering is done. This approach promises a simpler solution to the problem. The chapter will be dealt with in the following order:

- Autocorrelation definitions
- Filter identification
- Basic Yule-walker theory
- Matlab's YULEWALK function
- Implementation
- Conclusions

Most of the theory covered in this chapter is extracted from notes on "Power Spectral Density (PSD) estimates" by Prof M.W. Coetzer.

8.2 Autocorrelation definitions

Autocorrelation is a comparison of a signal with itself and it provides information about the time variation of the signal. Consider two signals where one is a copy of the other. The correlation of the two signals, according to the application engineering staff of Analog Devices (1992,358), is the sum of the scalar products of the signals in which the signal are displaced in time with respect to one another. They describe autocorrelation by equation 8.1 below.

$$R(k) = \sum_{n=0}^{L-k-1} \left(x(n) \times x(n+k) \right) \qquad (8.1)$$

where L is the number of samples and L-k-1 is the number of "overlapping" samples at the displacement k. Various methods for estimating the autocorrelation coefficients is discussed by Coetzer from Andersen (1974) and one of these definitions brings a scaling factor into the equation as follows:

$$R_x(m) \leftarrow C_x(m) = \frac{1}{N-m} \sum_{n=0}^{N-m-1} \left(x(n) \times x(n+m)\right) \qquad (8.2)$$

where A←B indicates that A is estimated by B and N is the number of samples.

In equation 8.2 values for $-m$ are not needed as $R_X(m)$ equals $R_X(-m)$.

8.3 Filter identification

A linear discrete recursive filter has the following general transfer function:

$$H(z) = B(z)/A(z) \qquad (8.3)$$

Where B(z) and A(z) are power series in z.. Sometimes B(z) = 1 or A(z) = 1, in which case they are called all poles or all zero filters respectively. In the case where the order of B(z) and A(z) is higher than zero the filter is called a pole/zero type filter. In statistical terms the filters are referred to as:

(a) The all zero filter, where A(z) = 1 is called a moving average – MA filter.

(b) The all pole filter, where B(z) = 1 is called a autoregressive – AR filter.

(c) The pole/zero filter is called an autoregressive/moving average – ARMA filter.

In this chapter the terms, poles, zeros and AR, MA will be used interchangeably.

The filters designed by the so-called Modified Yule-walker equations are ARMA filters. The ARMA filter can also be though of as an MA filter in series with an AR filter. It can be implemented from the general difference equation 8.4 and will have the general structure as in Figure 8.1.

$$y(n) = \sum_{k=1}^{M} a_k y(n-k) + \sum_{k=-N_f}^{N_f} h_k x(n-k) \qquad (8.4)$$

When designing a filter for this project one needs to start off with what is known, which is the frequency response in this case. The frequency response of the filter can be considered as the response of the filter to white noise. Figure 8.2 shows an example of this process where sampled white noise w_T, is the input to a filter with transfer function H(z) and a sampled output signal x_T is the output.

The autocorrelation of white noise $R_w(m)$ as defined by Coetzer is:

$$R_w(0) = 1 \qquad (8.5)$$

and $R_w(m) = 0$ for $m \neq 0$

thus the PSD for white noise $S_w(z) = 1$ (8.6)

Figure 8.3 illustrates equations 8.5 and 8.6

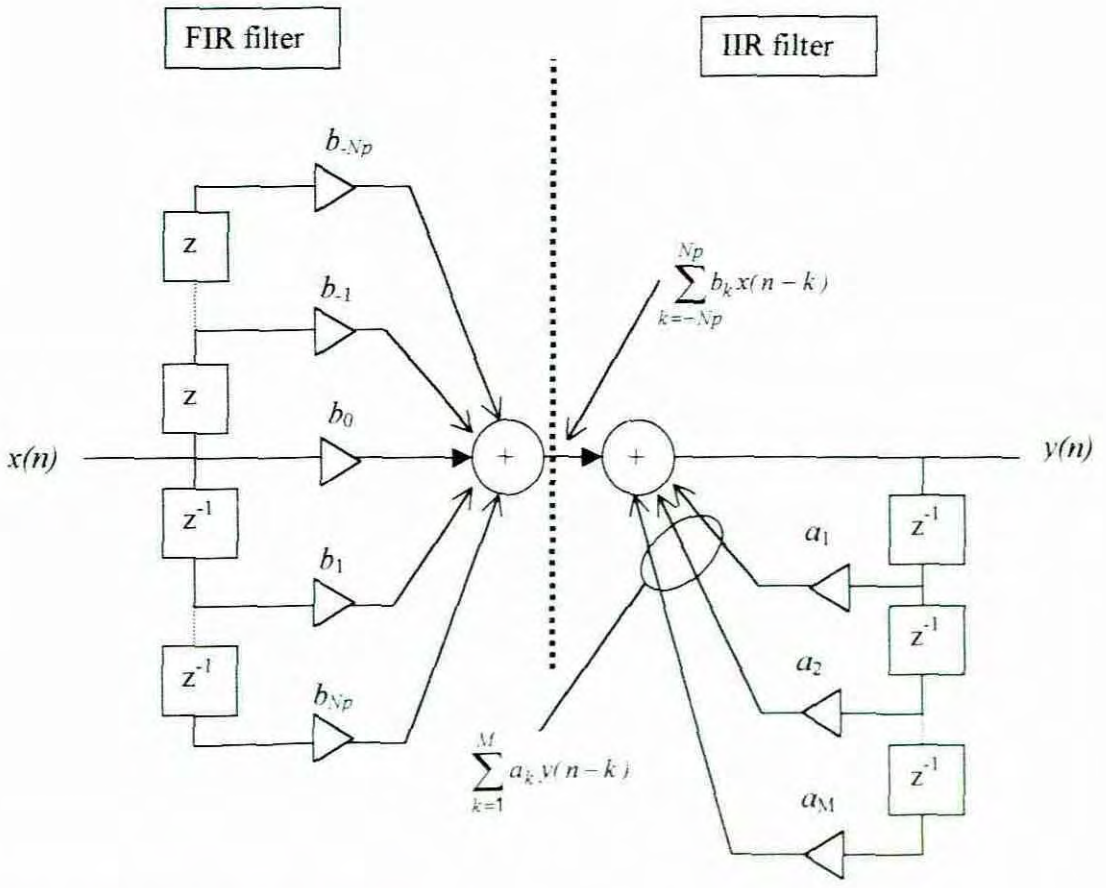
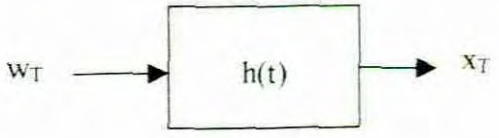


Figure 8.1 General digital filter



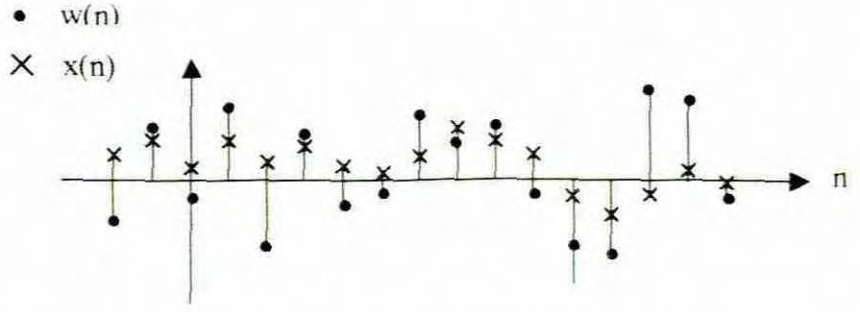


Figure 8.2 Filter with white noise w(n) as input and x(n) as output

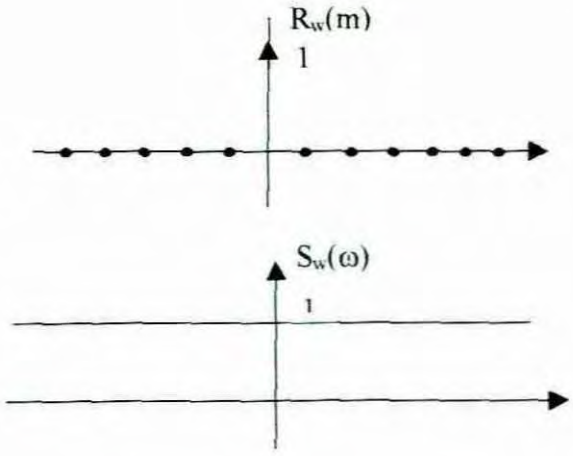


Figure 8.3 autocorrelation and PSD of white noise

If white noise is sent through a filter with transfer function H(z), than the PSD is given by :

$$\begin{aligned} S_X &= H(z).H(z^{-1}).\ S_W(z) \\ &= H(z).H(z^{-1}).1 \\ &= H(z).H(z^{-1}) \end{aligned} \qquad (8.7)$$

This process is illustrated in figure 8.4

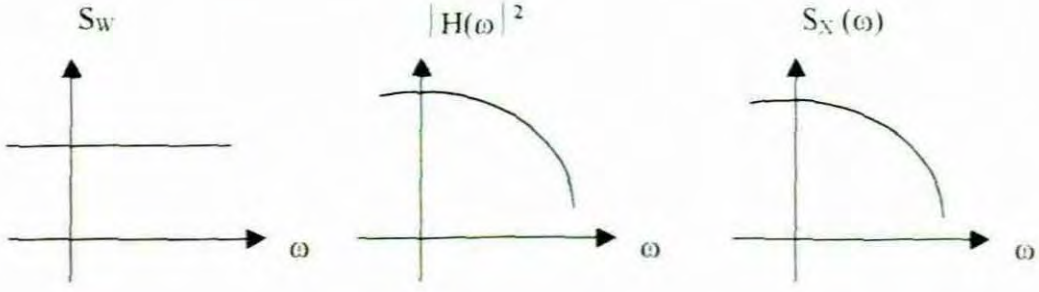


Figure 8.4 Power spectral density estimates by filter identification.

At this point it is worth noticing that performing an IFFT on the given frequency response (which is known and which was determined by the hearing evaluation) would in fact produce the output signal x(n) from the input signal w(n) (white noise) when passed through a filter with transfer function H(z). The output signal x(n) can then be used to determine the parameters of a pole/zero filter model or ARMA model using the Yule-walker method as explained in the next section.

8.4 Basic Yule-walker theory

The approach taken in this section will be to estimate the autoregressive (AR) and the moving average (MA) parameters in two separate steps. The AR parameters are first obtained as the solution to the so-called modified Yule-Walker (MYW) equations at first. Then a method to determine the MA parameters will follow.

In equation 8.3 where H(z) =B(z)/A(z), B(z) and A(z) can be written as:

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots + a_p z^{-p} \qquad (8.8)$$

$$B(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots\ldots + b_q z^{-q} \qquad (8.9)$$

From the general definition of the IIR filter and considering the input signal as white noise the sampled output signal x(n) can be written as:

$$x(n) = -\sum_{k=1}^{p} a_k x(n-k) + \sum_{k=0}^{q} b_k w(n-k) \qquad (8.10)$$

where p and q are the order of A(z) and B(z) respectively.

To obtain the MYW equations let:

$$R_x(m) = R_x(-m)$$

$$= \sum_{n=0}^{N-m-1} x(n) \times x(n+m) \qquad (8.11)$$

where the autocorrelation $R_x(m)$ can be calculated:

1. From a given signal $x(n)$ itself. The signal $x(n)$ is not given in this case.
2. From the IFFT$\{H(z)H(-z)\}$ where $\{H(z)H(-z)\}$ is the Power density spectrum. In this case H(z) is given by the Audiogram so that $R_x(m)$ can easily be calculated.

By doing a IFFT on H(z) the signal $x(n)$ can be derived and then $R_x(m)$ can be calculated by equation 8.11.

To continue the derivation of the MYW, substitute equation 8.10 into equation 8.11

$$= \sum_{n=0}^{N-m-1} \left(-\sum_{k=1}^{p} a_k x(n-k) + \sum_{k=0}^{q} b_k w(n-k) \right) \times x(n+m)$$

$$= \sum_{n=0}^{N-m-1} \left(-\sum_{k=1}^{p} a_k x(n-k) \right) \times x(n+m) + \sum_{n=0}^{N-m-1} \left(\sum_{k=0}^{q} b_k w(n-k) \right) \times x(n+m)$$

$$= -\sum_{k=1}^{p}\left(a_k \cdot \sum_{n=0}^{N-m-1} x(n-k).x(n+m)\right) + \sum_{k=1}^{q} b_k \sum_{n=0}^{N-m-1} w(n-k).x(n+m) \quad (8.12)$$

If $m>q$ then the last term in equation 8.12 equals zero because $x(n+q-1)$ is only dependent on previous input samples of white noise, w_T. Equation 8.12 then becomes

$$R_x(m) = -\sum_{k=1}^{p} a_k R_X(m-k) \quad (8.13)$$

For $m = q+1, q+2 \ldots\ldots$

The set of equations 8.13 can be solved to yield the AR coefficients $\{a_1 \quad a_p\}$. These equations are often called the Modified Yule-Walker equations. According to Friedman and Porat (1983) the name is motivated, by the similarity of these equations to the Yule-Walker equations arising in the autoregressive-modeling problem. The Yule-Walker equations have the form of equation 8.13 but with $q = 0$. For further detail on this topic consult Friedman and Porat (1983).

Equation 8.13 has the following matrix form:

$$\begin{bmatrix} R_X(q) & R_X(q-1) & \ldots\ldots & R_X(q-p+1) \\ R_X(q+1) & R_X(q) & \ldots\ldots & R_X(q-p+2) \\ . & . & \ldots\ldots & . \\ . & . & \ldots\ldots & . \\ . & . & \ldots\ldots & . \\ . & . & \ldots\ldots & . \\ R_X(q+p-1) & R_X(q+p-2) & \ldots\ldots & R_X(q) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ . \\ a_p \end{bmatrix} = \begin{bmatrix} -R_X(q+1) \\ -R_X(q+2) \\ . \\ . \\ . \\ . \\ -R_X(q+p) \end{bmatrix} \quad (8.14)$$

A further equation can be derived for $m = 0$, but would be dependent on both the a_k and the b_k coefficients. As there are easier ways to determine the b_k coefficients this derivation will not be pursued. To determine the b_k coefficients or the MA part of the estimate, the following model can be used.

Assume that the white noise, w_T, goes through two filters $H_1(z)$ and $H_2(z)$ to produce x_T where:

$$H_1(z) = B(z) \quad (8.15)$$

$$H_2(z) = 1/A(z) \quad (8.16)$$

If $H_2(z)$ is estimated by means of (8.13), then x_T can be sent through a filter with a transfer function A(z), to obtain the sampled output signal y(n), so that (with A(z) estimated accurately):

$$S_y(z) = H_1(z).H(z^{-1}) \tag{8.17}$$

Where y(n) is given by:

$$y(n) = \sum_{k=1}^{p} a_k .x(n-k) + x(n) \tag{8.18}$$

8.5 Matlabs YULEWALK function

In this section only an overview of the Matlabs YULEWALK function will be given, as this function is rather complex. The main routine of the YULEWALK function exploits and calls upon numerous other subroutines which, in turn, call on other built in routines. A detailed explanation of how Matlab performs this function would be impossible considering the limits put on the length of this document.

The YULEWALK function in Matlab designs a recursive filter using a least-square method. Krauss, Shure and Little (1994:2-232) explain the YULEWALK function in Matlab as follows:

"The function [B,A] = YULEWALK(N,F,M) finds the N-th order recursive filter coefficients B and A such that the filter:

$$\frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \ldots + b(n)z^{-(n-1)}}{1 + a(1)z^{-1} + \ldots + a(n)z^{-(n-1)}} \tag{8.19}$$

matches the magnitude frequency response given by vector F and M. Vectors F and M specify the frequency and magnitude breakpoints for the filter such that PLOT(F,M) would show a plot of the desired frequency response."

A brief summary of the algorithm, according to a printout of the Matlabs YULEWALK function, is as follows:

The denominator coefficients {a(1), ... a(NA)} are computed by the so called "modified Yule Walker" equations, using NR correlation coefficients computed by inverse Fourier transformation of the specified response H.

The numerator is computed by a four-step procedure. First, a numerator polynomial corresponding to an additive decomposition of the power frequency response is computed. Next the complete frequency response corresponding to the numerator and denominator polynomial is evaluated. Then a spectral factorization technique is used to obtain the impulse response of the filter. Finally, the numerator polynomial is obtained by a least square fit to this impulse response. For a more detailed explanation of the algorithm see Friedlander and Porat (1983)

8.6 Implementation

In the implementation of this technique it should be kept in mind that this project is an idea evolved out of the development of an audiometer. The audiometer, as previously explained, is a DSP based computerized system consisting of a computer to serve as the host interface and a DSP development board with an onboard ADSP2181 DSP processor. When the Yule-Walker technique is used in this project with the existing equipment, it means that three different programs are to be incorporated which are:

- Matlab – Matlab language – Yule-Walker function
- Audiometer/HLS – Visual Basic language – User interface, Audiometrical test and database.
- DSP program – Assembler language (ASM2181) - Filtering

With this arrangement the data that is stored in the database needs to be transferred to Matlab. This data obviously contains the frequency (F) and Amplitude (M) values, which are used in the Yule-Walker function. The frequency (F) and Amplitude (M) values would thus correspond to H(z), which are used to determine the autocorrelation as explained on page 8.5. When Matlab receives these variables F and M, it then calculates the filter coefficients Bs and As and returns them to the HLS program (program custom written with Visual Basic). The HLS program, in turn, sends these coefficients down the serial port to the DSP board, which uses these values in the actual IIR filter.

Data can be transferred between Matlab and the HLS user interface program in two ways:

- Dynamic Data exchange (DDE) or
- Data transfer via temporary files

In the implementation it was found that it is easier to transfer the data (from the HLS to Matlab and back) via temporary files than with DDE links. The programs that perform this function are included in appendix F

8.7 *Conclusion*

Performing the hearing loss simulation by means of the Yule-Walker method in Matlab might seem an easier method than the other methods discussed in this document. With this setup no real calculations need to be done, as Matlab performs all the necessary calculation. The only intricate part is the interface of the various programs.

The disadvantage of this method is that it is a more expensive setup in comparison with the IFFT method. In this case the addition of Matlab adds to the cost. This conflicts with the aim of the project, which is to construct the most economical system. Another hindrance is that when Matlab is started up from within the Visual Basic program, various flash screens associated with Matlabs startup appear, which is a rather unattractive feature.

Another significant drawback is the long time that Matlab takes to startup in comparison to the IFFT method. The IFFT method, in contrast to the Yule-Walker method (with Matlab), is much faster because the IFFT calculation is done within the same Visual Basic program as the host interface. To develop code in Visual Basic that performs the Yule-Walker function equivalent to the Yule-Walker function in Matlab would be extremely difficult and it would probably take years to develop. To write the code in Visual Basic to perform the Yule-Walker function which is equivalent to Matlab's function would be unnecessary work. The first step in Yule-Walker routine is to perform an IFFT calculation. If the results of this IFFT calculation can be used directly in a FIR filter, why still bother with further tedious calculations based on estimates that might affect the accuracy of the simulation.

Lastly, when the IFFT method is compared with Yule-Walker method, it might be argued that the ARMA filter would accommodate steeper slopes in the frequency response with fewer filter coefficients, due the principle of pole manipulation. This would obviously result in faster execution speed of the ARMA filter (in the Yule-Walker method) in

comparison to the FIR filter (in the IFFT method) on the DSP side, where the actual filtering is done by means of the convolution process. To justify the use of the IFFT method for this application the question might be posed as to whether these features are of real concern? Is the provision for extreme steep slope, such as the IIR filters are ideal for, really necessary? The audiograms of over 500 students at Nuwe Hoop Centre for the Hearing Impaired (over the past 10 years) were observed. Not in one case was a slope of more then 100 dBs per octave registered.

A study of the anatomy and the pathologies of the human ear would prove that it is most unlikely to have a complete cutoff (a brick wall scenario) at any one specific point anywhere on the frequency spectrum. Any sensorineural hearing loss due to known pathology would not have hair cells unaffected up to a certain point along the cochlear and completely destroyed along the rest. Therefore it is standard practice that the horizontal scale of the audiogram is never smaller then ½ octave. Most diseases will not create a condition which really necessitates this special feature offered by the ARMA filter. Secondly, using only 256 coefficients in the IFFT is more then sufficient for most audiograms and is still only a fraction of what the ADSP2181 can handle considering a execution speed of 33 million instructions per second. This point will be discussed further in chapter 12.

In conclusion, after comparing the IFFT and the Yule-Walker method, the IFFT method is still preferred for this application.

CHAPTER 9

COMPARISON OF THE VARIOUS METHODS

9.1 Introduction

This research boils down to the design of a digital filter. Filtering can be realized in either the time or the frequency domain. In the time domain the mathematical foundation of filtering is the convolution. A digital filter output *y(n)* is related to its inputs *x(n)* by the convolution with its impulse response *h(n)*:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \tag{9.1}$$

The filtering operation can also be performed in the frequency domain. Fourier transforms make it possible to convert convolution in the time domain to multiplication in the frequency domain. This is demonstrated by Kuc (1988:84) as follows:

"Let y(n) be the output of LTI system. Then, its Fourier transform is given by

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{\infty} y(n)e^{-j\omega n} \tag{9.2}$$

Applying the convolution definition of the output, we have

$$Y\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty}\left[\sum_{k=-\infty}^{\infty} h(k)x(n-k)\right]e^{-j\omega n} \tag{9.3}$$

Multiplying this last equation by unity, in the form $e^{j\omega k}e^{-j\omega k}$, we get

$$Y\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty}\left[\sum_{k=-\infty}^{\infty} h(k)e^{-j\omega k}x(n-k)e^{-j\omega(n-k)}\right] \tag{9.4}$$

Since $h(k)e^{-j\omega n}$ is not a function of n, it can be factored out of the summation with respect to n, giving

$$Y\left(e^{j\omega}\right) = \sum_{k=-\infty}^{\infty}\left[h(k)e^{-j\omega k}\sum_{n=-\infty}^{\infty} x(n-k)e^{-j\omega(n-k)}\right] \tag{9.5}$$

In the sum over n, the term $(n\text{-}k)$ represents a shift of the $\{x(n)\}$ by k samples. Futher, note that the value of the exponent follows the index of $\{x(n)\}$. Since the summation is over the entire sequence, this shift is inconsequential in evaluating the sum. By replacing $(n\text{-}k)$ with the dummy variable m, we note that the second summation becomes the definition of $X(e^{j\omega})$, the Fourier transform of $\{x(n)\}$. Since $X(e^{j\omega})$ is not a function of k, we can factor it out of the summation over k. The remaining terms in the sum then define the Fourier transform of $\{h(n)\}$. Hence, we have arrived at the desired result

$$Y(e^{j\omega}) = H(e^{j\omega}).X(e^{j\omega}) \tag{9.6}$$

This last result indicates that the input spectrum $X(e^{j\omega})$ is changed through a multiplicative operation by the filter transfer function $H(e^{j\omega})$ to produce the output spectrum."

By taking the inverse transforms on both side of Equation. 9.6, the output of the system can be seen as

$$y(n) = \Im^{-1}\,[H(e^{j\omega})X(e^{j\omega})] \qquad (9.7)$$

This idea is illustrated graphically by figure 9.1

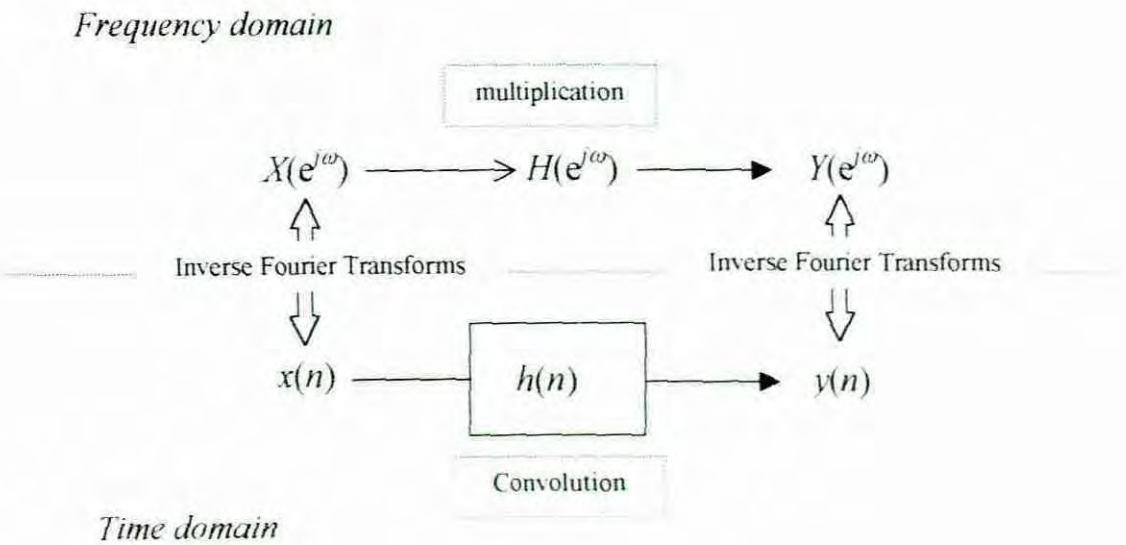


Figure 9.1 Illustration of the relation between Filtering in the Time domain and in the Frequency domain

The time domain behavior of a digital filter can be written in the form of the difference equation 9.9. In general the output of a first order LTI system at time n can be expressed as a linear combination of the inputs and the outputs in the following form

$$y(n) = \sum_{k=1}^{M} a_k y(n-k) + \sum_{k=-N_F}^{N_P} b_k x(n-k) \qquad (9.8)$$

In other word, the current output $y(n)$ is equal to the sum of past outputs, from $y(n-1)$ to $y(n-M)$, which are scaled by the delay-dependent *feedback* coefficients a_k, plus the sum of present and past inputs, which are scaled by delay-dependent *feedforward* coefficients b_k. The output values at time n are determined by the input values from N_F samples in the future $x(n-N_F)$, through the current values $x(n)$, to the sample N_P time units in the past $x(n-N_P)$. For a causal filter $N_F \leq 0$. When $N_F > 0$, then the future values of the input determine the current output value, and the filter is noncausal.

In the frequency domain, the system function can be express by Equation 9.9

$$H(z) = \sum_{k=-N_F}^{N_P} b_k z^{-k} \Big/ 1 - \sum_{k=1}^{M} a_k z^{-k} \qquad (9.9)$$

9.2 Filter categories

Digital filters are often categorized by either the duration of their unit-sample response or by their structure. When a filter produces a unit-sample response that has an infinite duration, it is called an infinite-impulse response (IIR) filter. If the digital filter has a unit-sample response having a finite duration, then it is called a finite-impulse response (FIR) filter. If the output is a function of the past outputs, then a feedback, or recursive, mechanism from the output must exist. Hence, such a filter is referred to as a recursive filter. A recursive filter can be recognized from the defining equations, since at least one of a_k coefficients, for $1 \le k \le M$, in Equations 9.8 and 9.9 is nonzero.

If the filter's output value is a function of only the input sequence values it called a nonrecursive filter. A nonrecursive filter can be easily recognized from the defining equation, for which $a_k = 0$ in Equations 9.8 and 9.9 for all values of k. All poles of a nonrecursive filter are at either $z=0$ or $z=\infty$. They do not contribute to the magnitude response and a nonrecursive filter is also called an all-zero filter in the frequency domain.

To achieve an infinite-duration unit-sample response, some form of recursive structure is necessary. Hence, the terms IIR (infinite impulse response) and recursive are commonly accepted as being interchangeable. Similarly, finite-duration unit-sample responses are typically complemented with nonrecursive filters. Hence, FIR (finite impulse response) and nonrecursive are also usually interchangeable. Several additional terms, used in the time domain, are *autoregressive (AR)*, *moving-average (MA)* and *autoregressive moving-average (ARMA)*. The FIR filter can be viewed as an MA system, while the general IIR filter, having both poles and zeros, is an ARMA system. An AR system is an IIR filter whose system function has all its zeros at $z=0$, that is $b_k=0$ for $k\neq 0$ and $c_k=0$ for all k. An AR filter is also commonly called an all-pole filter in the frequency domain, since all the

zeros are located at $z=0$, and hence are not relevant, in that they do not contribute to the filter magnitude response.

9.3 Comparison of IIR and FIR filters

Advantages of the nonrecursive or FIR Filter

- Stability: FIR filters are stable with respect to oscillation, since there are no poles.
- Accuracy: FIR filters tend not to accumulate errors, because of their finite memory of past events. 12 to 16 bits are generally adequate for FIR
- Design Ease: FIR filters are easy to understand and design, even for the non-specialist. While FIR filters do not correspond to familiar analog filters, they are the digital offspring of an analog predecessor, the delay line filter.
- Easy for Computer Aided Design (CAD): The filter coefficients are computed from a best-fit approximation to the transfer function, specified to desired limits.
- Can achieve linear phase.

Disadvantages of the nonrecursive or FIR Filter

- FIR filters require many coefficients to achieve high performance, resulting in more multiplications and lower bandwidth limits in those cases where an IIR filter could do the same job.

Advantages of IIR filters

- High efficiency: IIR filters have simpler architectures that require fewer coefficients and a smaller number of multiply operations, giving higher throughput while requiring less memory than FIR. They also lend themselves to multiplexing.
- Analog relative: IIR filters retain close contact with analog filter design and its familiar terminology (Butterworth, Chebyshev...).
- CAD design ease: IIR filters can be designed with CAD tools, via programs that optimize the coefficients of standard filter forms to meet stated transfer-function criteria.

Disadvantages of the IIR filter:

- Stability must be carefully considered in the design.

They can suffer from overflow, noise, and quantization errors, because of their long "memory" of past data.

All of the methods researched for this project, except for the wavelet method, fall into one of these categories. The first method on filterbanks, as it is presented in this document falls into the IIR filter category. Krauss, Shure and Little (1994: 2-232) classify the Yule-Walker method as a method to design IIR digital filters. It is in fact a ARMA filter and is referred to as a IIR filter because of the existence of the poles, which causes feedback. Filtering, in which the filter's coefficients are determined by means of the inverse transforms of the DFT, the FFT falls according to Krauss et al (1994:2-109) into the category of FIR filters. The inverse CZT method, as it is presented in this text, could also fall under a method to design FIR filters.

9.4 Comparison of the different methods

A comparison of the various methods presented in this document are dealt with in the following order.

- Execution speed
- Quantization noise
- Frequency resolution
- Stability
- Memory
- Complexity
- User interface

9.4.1 Execution speed

The first method uses a bank of IIR filters. The IIR filter is probably the fastest method because it normally has fewer filter coefficients to process during the computation. The IIR filters make use of feedback loops, which is the reason for the reduction in required operations. The Yule-Walker method (also IIR filter) will be faster than the filterbank

method. FIR filters has no feedback loops and needs more filter coefficients than the IIR filter to achieve the same performance. FIR filters might thus be slower, but have other advantages such as stability and accuracy. In determining the coefficients of the FIR filters by means of the IDFT, it was pointed out that the slow execution speed of the DFT was the main reason for the development of the FFT algorithm. It was shown in chapter 5 that instead of using the IDFT method to derive the FIR filter coefficients, the IFFT method would be 64 times faster than the IDFT method when a 256 point transform is considered.

The filtering operation can also be performed in the frequency domain by the *Overlap-save* method as described in chapter 4, instead of the FIR filter which utilizes convolution in the time domain. Kuc (1988:384) explains that for large values of N_H (the size of the filter duration) the *Overlap-save* method is used to the full advantage, but also the choice for the value of N_Q (the number of samples per block) also becomes important. He shows by means of a curve that $N_Q = 4N_H$ is a reasonable choice that will provide saving in the computation when $N_H > 256$. So when the FIR with coefficients calculated with the IFFT is compared with the *Overlap-save* for $N_H = 256$ no huge saving in terms of execution speed will take place when the latter is used. This should prove that the IFFT method could still be preferred above the *Overlap-save* method, as the later would not provide a significant saving in execution speed for this project where 256 samples from the Audiogram are used. Also, the ADSP-2181 processor running at 33 MIPS is fast enough to handle this FIR filtering.

The execution speed of Wavelets cannot be compared directly with other methods. A comparison should be done based on a specific application. The number of multiplications will be dependent on what wavelet is used. For instance if a higher order Daubechies wavelet is used, there will be more filter coefficients to operate on in the convolution process. Using wavelets for filter banks might be slower than using direct IIR filters. In this case there is a trade-off between accuracy and execution speed. Comparing wavelets and IIR filters in this way, it might be presumed that the wavelet method will take longer as it consists of both decompositioning as well as reconstruction where as the IIR only consists of one filter. On the other hand, quadrature mirror arrangement results in perfect reconstruction which produces exact frequency separation, no distortion and aliassing cancellation. The same argument can be followed when filterbanks are designed by means of FIR filters. The conclusion would be that, whether IIR or FIR filters are used for

filterbanks, the wavelet method would be slower but more accurate. The level of decompositioning also governs the speed of the wavelet method. A reason was provided in section 6.4 why the FFT method is faster then the Wavelet method for this specific application. To get the same resolution in frequency between, for example, a 256 point IFFT and the wavelet method, decompositioning in the wavelet method has to extend to a scale of level $j = 256$. It would be practically impossible when using wavelets to equal this specification.

As the accuracy of the Inverse CZT derivation is in question, as explained in chapter 7, this method is not considered for implementation and any comparison with this method would be unnecessary.

Comparing the MYW method with the IFFT method, the MYW is in theory faster than the IFFT method since the MYW method utilize IIR filters and the IFFT method utilizes FIR filters. A calculation can be performed to determine the time the computer will take to get the coefficients of the FIR filters by means of IFFT computation. It was shown in chapter 5 that the number of multiplications for the FFT algorithm could be calculated as follows.

$$\begin{aligned} \text{FFT steps} &= \frac{N}{2}\log_2 N \\ &= \frac{256}{2}\log_2 256 \\ &= 1024 \end{aligned}$$

If a processor that runs at 100 MHz is used then the time taken to perform this multiplication would be $\frac{1024}{100 \times 10^6} = 10.24\ \mu\text{sec}$ which is acceptable in the context of this application. The MYW method, on the other hand, as was proposed and explained in chapter 8, makes use of Matlab to determine the filter coefficients. In order to perform MYW computation in Matlab, Matlab has to read instructions from a file called a Matlab program. This Matlab program named yul.m (listed in Appendix F.3) is written in Matlab's language. When the computation is to be performed, Matlab has to read these instructions from the program to carry out the calculation and to transfer the data. A program in Visual Basic (listed in Appendix F.1) starts up Matlab and instructs it to load and execute the Matlab program yul.m. It should be clear that Matlab has to be started up every time the computation needs to be performed. Executing these procedures on the same computer with a processor speed of 100 MHz takes a long time. Measuring the time physically results in a

delay of more than 30 seconds to obtain the filter's coefficients. Comparing it with the IFFT method of which the IFFT computation is done within the same Visual Basic program, it could be seen that the IFFT method is very much faster. To write a program in Visual Basic equivalent to how Matlab calculates the MYW function would be an enormous job considering the complexity of how Matlab calls functions upon function to finally derive the MYW function. Writing a program in Visual Basic to perform the MYW function would exceed the time frame of this study.

9.4.2 *Quantization noise*

Quantization is an unavoidable fact in signal processing. Unfortunately, it introduces systematic errors known as "quantization noise". IIR filters might drift due to the quantization error that is continuously added from memory in the feedback loop. FIR filter is not so vulnerable, since it does not have any feedback loops.
In connection with the Wavelet method Meyer (1993:3) says that "The coding algorithm that is used (taking into account the nature of the signal) ought to reduce the effects of quantization noise when decoding takes place. One of the advantages of quadrature mirror filters is that they "trap" this quantization noise inside well-defined frequency channels." This is a major advantage of wavelets above the IIR Filterbank designs.

9.4.3 Frequency resolution

The term Frequency resolution as it is used in this project refers to the width of the passband filter. Many Digital-hearing aids currently on the market use a filterbank with 8 adjacent bandpass filters. In this case, if the audio signal is sampled at 32 kHz and the width of the bandpass filters is the same, then the width of each band could be calculated as 16000 Hz / 8 = 2000 Hz where 16 kHz is the Nyquist frequency. So the term as it is used in this project means that the system has a resolution in frequency of 2000 Hz. If the number of bandpass filters is increased the resolution in frequency decreases. In term of the HLS it would mean that the more adjacent bandpass filters that are used, the more closely the HLS would resemble the ear. The advantage of the filterbank method as well as the Wavelet method is that bandpass filters can be arranged in such a way that width at the lower frequency can be made small and than gradually increased up to the highest frequency. This will coincide with the frequency scale of the audiogram, which is divided into octaves. The

multiresolution property of wavelets will cause frequency splitting into octaves. Looking at the IFFT method it could be said that, for example, a 256 point IFFT would equal a filterbank with 256 adjacent bandpass filters. In this case it would have a resolution of 16000 Hz / 256 = 62.5 Hz. To equal this with the filterbank method means that 256 classical bandpass filters must be designed, which is a tremendous effort. The MYW method, on the other hand, is an IIR filter that would provide an improvement on the filterbank method as it could directly model the whole sloped frequency response. This method would eliminate the problem encountered by trying to make the resolution as small as possible.

9.4.4 Stability

It has been stated earlier that FIR are stable with respect to oscillation, since there are no poles. This means that the IFFT method, which gives the coefficients of the FIR filter, would be preferred over the other methods in terms of stability.

9.4.5 Memory

Previously it was mentioned that IIR filters require less memory than the FIR filter for the same performance. In determining which filter type is preferred, it should be noted that the ADSP-2181 has 16K Data and 16K Program memory (RAM) onboard. If a FIR filter with 256 coefficients is considered for implementation, then ADSP-2181 has more than enough memory available to do the job easily.

9.4.6 Complexity

The IFFT method is by far the simplest method to implement when compared with the other methods presented in this document. The first method, which is the filterbank method, needs the design of several bandpass filters, maybe by means of bilinear transformation. The Wavelet method as explained in chapter 6, is much more intricate than the IIR filterbank or the IFFT method. Although an attempt has been made to derive an inverse Chirp-z transform, this derivation could not be proven theoretically or even practically. That is why the inverse Chirp-z method, as it is originally proposed, is not even considered for implementation. The MYW method, as it was explained in chapter 9, is also more

complicated to implement since it requires an extra software package called Matlab, whereas the IFFT could easily be done within the same Visual Basic program.

9.4.7 User interface

The IFFT method calculates the FIR coefficients within the same Visual Basic program that is responsible for the user interface. The MYW method as it was proposed calculates the IIR coefficients in Matlab, which is a separate software package.
The MYW method resulted in a less attractive user interface in comparison with the IFFT method, as various splash screens appear when Matlab is started up.

9.5 Optimum method

The optimum method would be one that is accurate, economical, has minimum distortion, is easy to implement and fast enough for the system to handle. A method that would require the least computation would contribute to a reliable operation in real time. The aim is thus to perform a basic filter operation in real time. The basic filter is describe by Strang and Nguy (1996:1) as follows:

> " A filter is a linear time-invariant operator. It acts on input vector x. The output vector y is the convolution of x with a fixed vector h. The vector h contains the filter coefficients $h(0)$, $h(1)$,..... The filter is digital, not analog, so the coefficients $h(n)$ come at discrete time $t = nT$. The sampling period T is assumed to be one here.
> The input $x(n)$ and output $y(n)$ come at all time $t = 0, \pm1, \pm2, \ldots$
>
> $$y(n) = \sum h(k)x(n-k) \quad = \quad \textit{convolution } h * x \textit{ in the time domain} \qquad (9.10)$$
>
> One input $x = (\ldots, 0, 1, 0, \ldots)$ has special importance – a unit impulse at time zero. The input has $x(n-k) = 0$ except when $n = k$. The sum in the convolution has only one term, and that term is $h(n)$. This output $y(n) = h(n)$ is the response at time n to the unit impulse $x(0) = 1$. It is the impulse response $h(0), h(1), \ldots, h(N)$. The same filter can be described in the frequency domain. Convolution with the vector h will become multiplication by a function H. The action of the filter in time and frequency is the foundation on which signal processing is built."

Equation 9.10 is the foundation on which this project is built. The strategy is to compute the impulse response beforehand and then use it in the convolution of Equation 9.10 in real

time. The question now is which of the proposed methods should be used to derive the impulse response. From the preceding comparisons and table 9.1 which is a summary of the comparison, it should be clear that the IIR filter banks and the wavelet method could be eliminated. One of the reasons why the IIR filter banks are eliminated, is because of the problem which is encountered in the areas where two adjacent bandpass filters overlap. Also, the resolution in frequency of the filter bank method would be much bigger than that of the IFFT method, since the resolution in frequency is determined by the bandwidth of the bandpass filters. The wavelets method is eliminated because its multiresolution decompositioning is limited compared with the IFFT method. As has been explained in chapter 6 section 6.4, it is the proposed hardware that imposes this limitation. The DFT is eliminated because of its slow execution speed. The inverse CZT is eliminated because of its unproven derivations.

Filter type	Harmonic Content	Algorithn Complexity	Memory Requirement	Execution time	Flexibility	Frequency resolution
IIR filter Bank	HIGH	LOW	LOW	FAST	LOW	POOR
IDFT	LOW with appropriate Window	LOW	MEDIUM	SLOW	LOW	GOOD
IFFT	LOW with appropriate Window	AVERAGE	MEDIUM	FAST	LOW	GOOD
Wavelets	LOW Depending on Wavelet used	MEDIUM	HIGH	SLOW	HIGH	LOW
CZT	LOW with appropriate Window	HIGH	MEDIUM	FAST	HIGH	GOOD
MYW	HIGH	AVERAGE	LOW	FAST	HIGH	HIGH

Table 9.1 Table of comparison of the various algorithms

The only two methods left are the MYW and the IFFT. In chapter 8 it has been shown that the MYW method, as it was proposed, is a more expensive setup because of the addition of Matlab which is contrary to the objective to provide the most economical instrument. Also

the MYW method is more complex than the IFFT method as the calculation of the latter can easily be done within the same Visual Basic program.

9.6 Conclusion on the comparisons

This conclusion is in response to the objective of the study. It relateds to the choice of the filter. In other words, what filter is to be used. With the argument presented above it should be clear that the use of the inverse FFT is well motivated to be the optimum method. The following points are the main reason why the IFFT method is opted for:

- Both the AR and ARMA are based on the placement of pole in the z-plane. The use of poles can results in oscillation. This is not a characteristic of the human ear. The characteristics of the human relates more to the formation of zeros in the frequency domain. It was mentioned earlier that the AR or FIR filter is an *all zero* filter, which is the most important reasons why the FIR filter is chosen and the IFFT method is chosen. The following drawing illustrates this idea

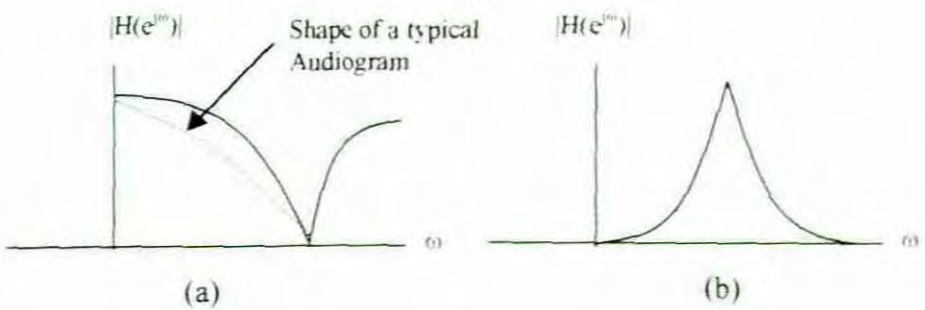


Figure 9.2 Magnitude response of (a) Zero and (b) Pole

By visual inspection of hundreds of audiograms at Nuwe Hoop Centre the *all-zero* filter is most likely to resemble typical pathologies of the ear.

- The second point deals with the issue of implementation, which is also the second point mentioned in the objective of the study. Filtering can be done in either the Time or the Frequency domain. In the Frequency domain, the *Overlap-save* method can be used. This method is a rather complex in comparison to the IFFT method as it needs to implemented for real time operation on the DSP side. The IFFT method is a much simpler arrangement

since it is not necessary to perform the IFFT computation in real time. The IFFT calculation is done beforehand on the PC and only the coefficients of the filter are sent to the DSP, where the filtering is done by means of a simple convolution operation. The DSP hardware employed in this project is, however, fast enough for the FIR filtering.

9.7 Strategy of implementation

The strategy for implementing the inverse fast Fourier transform as a method to realize hearing loss simulation is as follows: The two formulas that form the core of the implementation, are Equations 8.1 and 4.27. Equation 4.27 can be rewritten in terms of $H(k)$ where $H(k)$ is the frequency response of the ear.

from Equation 4.27 $$h[k] = \frac{1}{N}\sum_{n=0}^{N-1} H[n]e^{jkw_n} \qquad \text{k} = 0, 1, \ldots, \text{N-1} \qquad (9.11)$$

from Equation 9.10 $$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \qquad (9.12)$$

In equations 9.11 and 9.12 N is the number of points on the horizontal (frequency) axis of the audiogram. The discretised incoming signal is denoted by $x(n)$ and the output $y(n)$ is produced by the convolution of $x(n)$ with $h(n)$. Equation 9.11, which is an IDFT, will be performed by the IFFT algorithm. As stated previously the inverse FFT (IFFT) is calculated by the same procedures as the FFT except that the twiddle factors are of opposite powers and that the IFFT is scaled by 1/N.

The strategy would thus be to first interpolate the points on the Audiogram at which the hearing tests were performed. This is done by linear piece-wise interpolation. A IFFT calculation is then performed according to Equation 9.11 to determine the coefficients of the FIR filter. These values are then sent to the DSP where the actual filtering is done by means of the convolution according to Equation 9.12.

9.8 Conclusion

In this chapter the use of the IFFT method to perform the simulation was motivated. This method has been shown to be the most economical in the sense that the IFFT computation is done within the same Visual Basic program and requires no extra software packages like Matlab. This is also a fairly simple arrangement as many well-written programs already exist in Basic to perform the IFFT function. Also, on the DSP side, many existing programs

use the FIR filter with its convolution function as a basic function. Considering the fact that the existing hardware of the Audiometer is used, this method is well suited since the ADSP-2181 processor is fast enough and has enough memory to handle this specific application. The next step would be to look at the practical implementation. The practical implementation has to be discussed in terms of the hardware used, as well as the software development. Chapter 10 deals with the necessary areas of the DSP and Chapter 11 discusses the actual implementation of the IFFT method.

Part III

Practical implementation of theoretical solution

CHAPTER 10

AN OVERVIEW OF THE NECESSARY AREAS OF THE DSP ARCHITECTURE

10.1 Introduction

This part of the document overviews the necessary areas of the DSP hardware and how the algorithm is implemented. The implementation of the HLS is done on an ADSP_2181 DSP processor, since this processor was originally utilized in the development of an audiometer. The audiometer forms an integral part of this application as it is the instrument that is used to determine the frequency response (hearing loss) of the ear by means of pure tone audiometry as explained in chapter 2.

An overview of the Audiometer and HLS is presented first, then the development system and the ADSP_2181 DSP processor is discussed in further details.

10.2 Overview of Hardware use in the Audiometer and HLS

The audiometer consists of two sides. The PC (host) side and the DSP side. Figure 10.1 shows this arrangement with additional amplifiers connected to the output of the DSP. The DSP side generates sinewaves by means of a table lookup method. A table on the DSP side called the 'sinewave lookup table' holds the sinewave lookup values. These values were determined by the equation: $X(i) = \sin(i \times 2\pi/N)$. N is the number of values in the sinewave lookup table to complete one complete cycle (360°), in which case N = 256. A sinewave is generated when the program is run through the lookup table and the numerical values are converted from digital to analog form. The 256 sinewave values range between +1v and –1v. If the program steps through each location in the lookup table, a sinewave with a certain frequency is generated depending on the sampling rate to which the DSP side is configured. By increasing the step size to 2, in other words accessing every second memory location in the lookup table, the frequency of the sinewave is effectively doubled. Changing the step-size thus changes the frequency of the sinewaves.

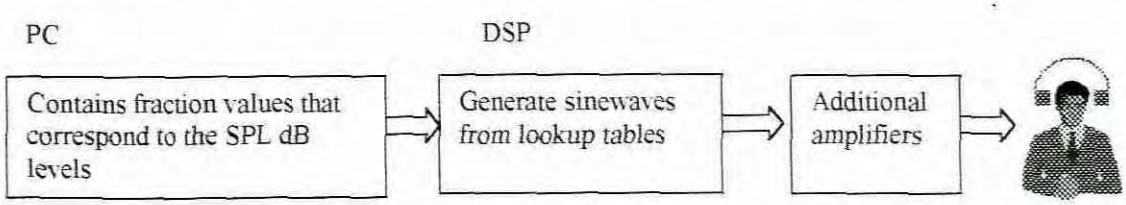


Figure 10.1 Block diagram of the Audiometer

The ADSP_2181 uses the 1.15 fractional binary format so that the sinewave values which range between +1v and –1v, range in the 1.15 format between 0x7FFF and 0x8000 respectively. The ADSP-2100 Family user's manual written by the editorial staff of Analog Devices (1995:2-2) explains that: " The ADSP-2100 family operations presume or support twos-complement arithmetic. It arithmetic is optimized for numerical values in a fractional binary format denoted 1.15. In the 1.15 format, there is one sign bit (the MSB) and fifteen fractional bits representing values from -1 up to one LSB less than +1. Figure 10.2 shows the bit weighting for 1.15 numbers. Below are examples of 1.15 numbers and their decimal equivalents."

1.15 Number	Decimal Equivalent
0x0001	0.000031
0x7FFF	0.999969
0xFFFF	-0.000031
0x8000	-1.000000

2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

Figure 10.2 Bit Weighting For 1.15 Numbers

Table 10.1 shows an example of an 8 valued lookup table used for generating a sinewave. To get a full 100 db sound pressure level from a sinewave with +1v to –1v swing, an additional amplifier for each ear is added as shown in figure 10.1.

Decimal	1.15 Number
0.707	5A82
1	7FFF
0.707	5A82
0	0000
-0.707	A57E
-1	8000
-0.707	A57E
0	0000

Table 10.1 Lookup table with 8 values

The PC, on the other hand, has a lookup table that contains the amplitude values for the different sound pressure levels. When a sinewave with a certain SPL has to be generated, a value that corresponds to that SPL is retrieved from the lookup table and then multiplied with the sinewave values. When, for instance, the value 1 is retrieved from the lookup table and multiplied by the sinewave values, a pure sinewave is produced with a SPL of 100 dB's. When a value of, for example, 0.5 or 0x3FFF is retrieved and multiplied with the sinewave values it might result in a SPL of 50 dB. These values in the amplitude lookup table were calibrated by measuring the actual sound pressure level at the headphones. Each point on the frequency axis, on which the hearing test is performed, has its own lookup table. This point will be discussed further in chapter 11 and 12, but the important thing to notice at this stage is that the audiogram or the frequency response is determined by selecting the appropriate values from these lookup tables held on the host (PC) side.

Conversely, the HLS uses the values determined by the hearing test, which is in fact the response of the ear at different frequencies to pure sinewaves. The theory of Fourier transform is based on sinewaves and is exploited in this application to determine the filter coefficients. The filter coefficients are calculated by performing a IFFT operation on the frequency response of the ear and this is also done on the host (PC) side. The same arrangement as in figure 10.1 is use for the HLS. On the DSP side, the filter coefficients are used in the convolution operation thereby realizing the required digital filtering or attenuation on the incoming signal. Kuc (1988:1) defines a digital filter as a numerical procedure, or algorithm, that transforms a given sequence into a second sequence that has some more desirable properties. Ludeman (1986:5) explains that the filtering operation can be accomplished digitally by using the structure as shown in figure 10.3. The structure is composed of an analog prefilter; an analog-to-digital (A/D) converter; a digital filter represented by a transfer function, H(z); a digital-to-analog (D/A) converter and a reconstruction filter.

The analog prefilter specified by its transfer function $H_{pf}(s)$ is in most cases a low-pass or bandpass filter designed to reduce the effect of out of band interfering signals. Interfering signals could be extraneous noise or higher-frequency signals that, when sampled, produce lower-frequency signals (a phenomenon called "aliasing"). For this reason, the analog

prefilter is sometimes called an "antialiasing" filter in that it combats the aliasing phenomenon.

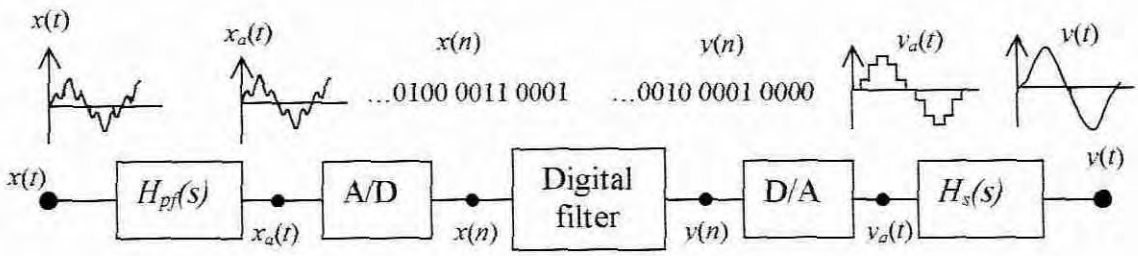


Figure 10.3 Block diagram of a DSP filter system

The A/D converter is a device which will, upon command, give a binary code word corresponding to the quantized level of a continuous-time input signal at that time. The input signal $x(t)$ shown on the left of figure 10.3 is filtered by the prefilter to provide the analog signal $x_a(t)$, which is sampled and coded to give the input sequence $x(n)$. This $x(n)$ is then operated on by the digital filter to produce $y(n)$. The operation produces the output $y(n)$ at a time n as a weighted sum of the past input and presents input values. This is the operation of the FIR filter, which is described by the difference equation:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

This is called the convolution equation with $h(n)$ being the filter's coefficients determined by the IFFT calculation.

The D/A converter shown in figure 10.3 is a device that operates on a sequence of input code words to produce a continuous-output signal, usually of a staircase form noted by the function $y_a(t)$. This staircase form is then smoothed by a reconstruction filter to produce the desired output signal $y(n)$.

10.3 DSP development board (EZ-KIT Lite)

To realize the DSP filter system that has been discussed above, a unit called the EZ_KIT Lite is used. In 1996, when the Audiometer was developed, the EZ-KIT Lite was one of the most cost effective, powerful DSP development systems on the market. The ADSP-2181 used in the EZ-KIT Lite offered the highest integration and performance in the 16 bit

fixed-point DSP processor arena with 32K word of chip RAM, 30 ns instruction cyle time, serial ports, DMA ports and low power modes. Since then, faster DSP processors with more data and program memory storage capacity have been developed. Chapter 11 shows the evaluations of the speed requirements of the application versus the speed of the ADSP_2181. It will be pointed out that the ADSP_2181 is well suited for this application and thus the consideration of any other faster processor, is not really necessary.

Figure 10.4 shows that the EZ-KIT Lite consists of a small ADSP-2181 based development board with full 16-bit stereo audio I/O capabilities. The board's features include:

- The ADSP-2181 33Mips DSP processor that generates the sinewaves and performs the convolution calculation for digital filtering. The ADSP-2181 will be explained in further detail later in the chapter.

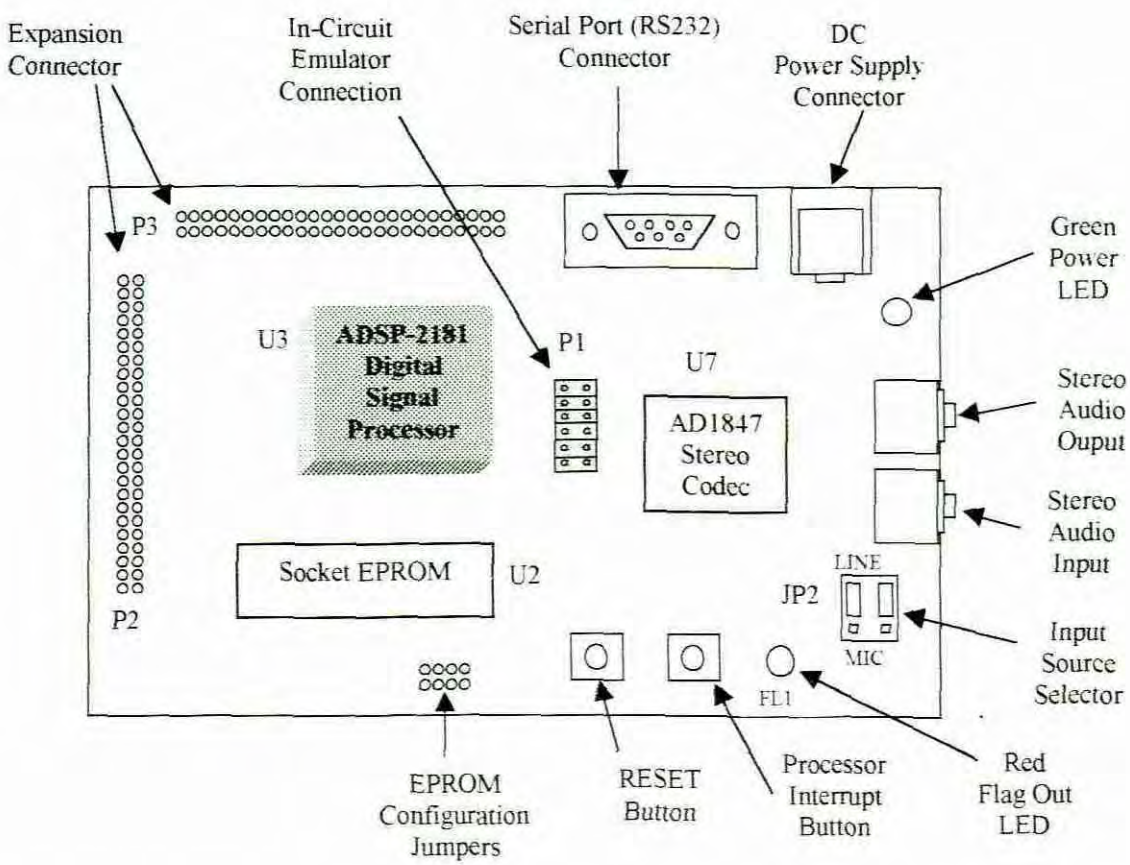


Figure 10.4 EZ-KIT Lite Board Diagram

- The AD1847 Stereo SoundPort is a CODEC (Coder and decoder) that performs the A/D and D/A conversions as explained earlier. It also takes care of aliasing and reconstruction. It would be impossible to explain the CODEC's operation and configurations fully in this document. Further details can be found in the AD1847 CODEC's specifications document by Analog Devices (1995)
- The RS-232 Interface manages the serial communication between the PC and the EZ-KIT Lite.
- The EPROM socket holds the EPROM into which a program can be burned so that it can be operated as a standalone unit. The DSP is configured to boot from EPROM when a reset is deasserted.
- Other features includes: Power supply regulation, Expansion connectors, user's push buttons and user's configurable jumpers.

The Circuit diagrams of the development board can be found in the EZ-Kit Lite users Manual. by Analog Devices (1995).

10.4 The DSP processor (ADSP-2181) used in the HLS

The ADSP_2181 is a programmable single-chip processor with a base architecture optimized for digital signal processing (The Editorial Staff of Analog Devices 1995). Although most of the work in the following two sections is extractions from the "ADSP-2181 Family user's manual (1995)" it does not claim to be a complete description of the ADSP-2181 processor. A complete reference on the ADSP_2181 DSP chip can be found in the "ADSP-2100 Family user's manual" written by The Editorial Staff of Analog Devices (1995). Only areas that are relevant to this project will be elucidated in short. As mentioned, the ADSP_2181 DSP processor was the latest and best DSP processor in the analog devices ADSP_2100 family fixed-point arithmetic range during the development of the Audiometer in 1996 and is the reason why it is used in the HLS. In the rest of the chapter the features of the ADSP_2181 are discussed under the headings "Core Architecture" and "ADSP-2181 integration".

10.5 Core Architecture

Figure 10.5 shows a block diagram of the core architecture of the ADSP-21XX family. The core of the processor contains three independent, full function computation units which are:

- The Arithmetic/logic unit (ALU)
- The Multiplier/accumulator (MAC)
- The shifter

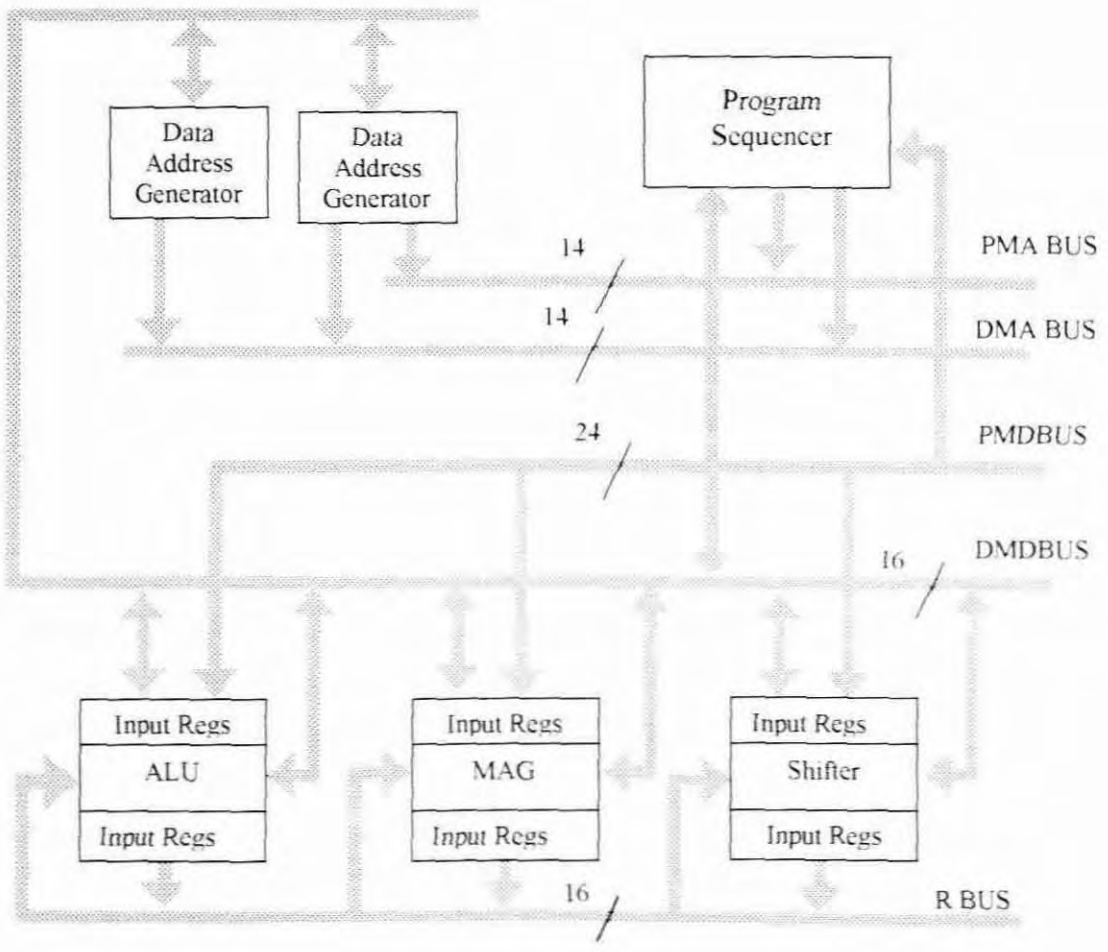


Figure 10.5 Core architecture of the ADS P-2100 family

The computational unit processes 16-bit data directly and has provisions to support multiprecision computations. The ALU performs a standard set of arithmetic and logic functions. The arithmetic functions are add, subtract, negate, increment, decrement and

absolute value. The logic functions are AND, OR, XOR (exclusive OR) and NOT. The ALU is 16 bit wide with two input ports, X and Y, and one output port, R.

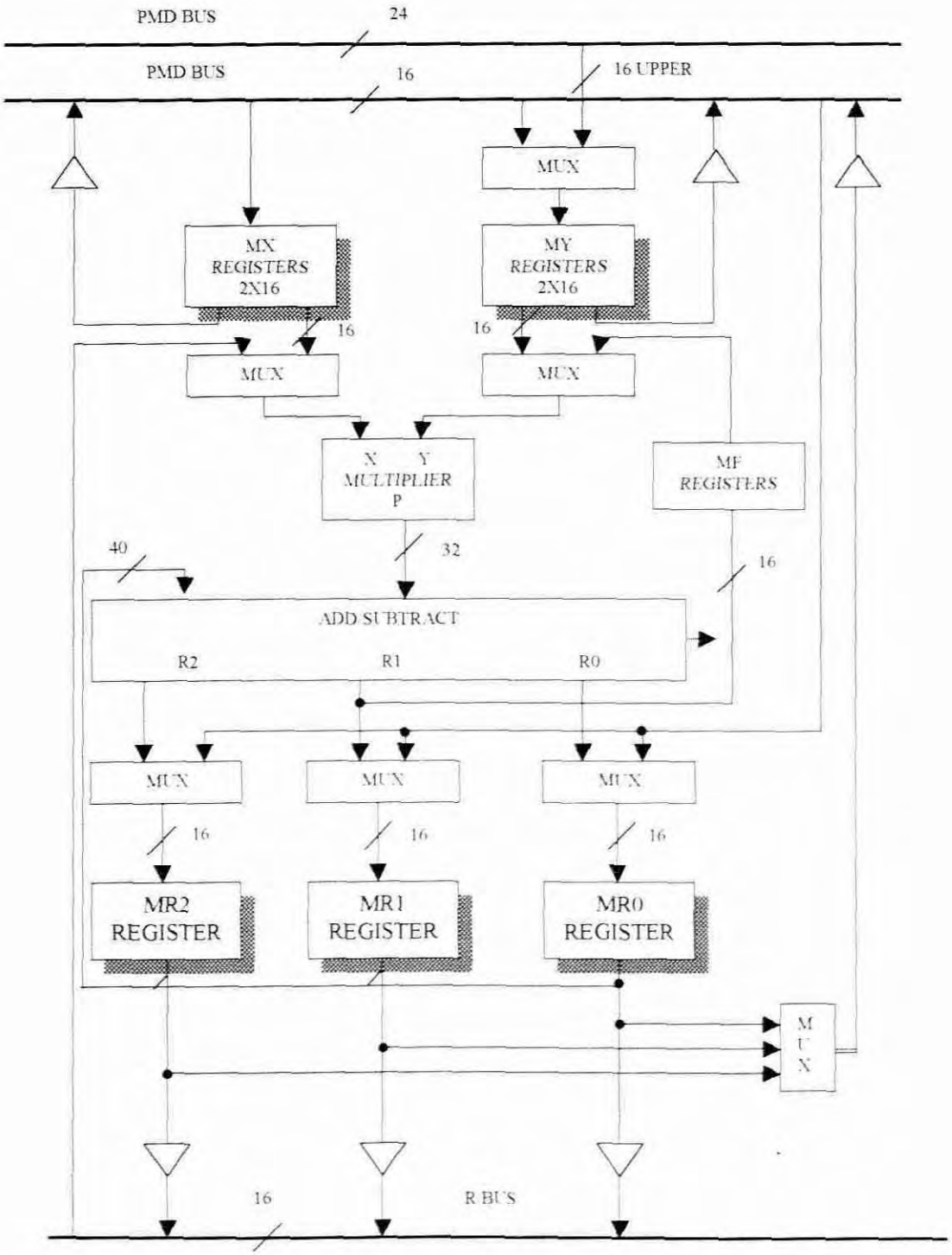


Figure 10.6 MAC Block diagram

The multiplier/accumulator (MAC) provides single-cycle multiplication, multiplication with accumulative addition, multiplication with accumulative subtraction, saturation and clear to zero functions. Figure 10.6 shows a block diagram of the MAC. A feedback function allows part of the accumulator output to be directly used as one of the multiplicands of the next cycle. The multiplier has two 16-bit input ports X and Y, and a 32-bit product output port P. The 32-bit product is passed to a 40-bit adder/subtracter which adds or subtracts the new product from the content of the multiplier result (MR) register, or passes the new product directly to MR. The MR register is 40 bits wide. The MR register consists of three smaller registers: MR0 and MR1 which are 16 bits wide and MR2 which is 8 bits wide. The most common function performed by the MAC that is particularly used in the HLS, is the MR+X*Y multiply/accumulate function which multiplies X and Y operands and adds the result to the MR register in one cycle. This is a basic function for the convolution operation.

The shifter provides a complete set of shifting functions for 16-bit inputs, yielding a 32-bit out. These include arithmetic shifts, logical shift, normalization and derive exponent operations. The shifter array is a 16 x 32-barrel shifter. It accepts a 16-bit input and can place it anywhere in the 32-bit output field, from off-scale right to off-scale left, in a single cycle. This gives 49 possible placements within the 32-bit field.

The program sequencer and two dedicated data address generators as shown in figure 10.5, ensures efficient delivery of operands to these computation units. The sequencer supports conditional jumps, subroutine calls and returns in a single cycle. With internal loop counters and loop stacks, the ADSP-2181 executes looped code with zero overhead; no explicit jump instructions are required to maintain loops. The two data address generators (DAGs) shown in Figure 10.7 provides addresses for simultaneous dual operand fetches (from data memory and program memory). The DAGs provide indirect addressing. Both perform automatic address modification. For circular buffers, the DAGs can perform modulo address modification. In a single DAG there are three register files: the modify (M) register file, the index (I) register file, and the length (L) register file. Each of the register files contain four 14-bit registers, which can be read from and written to via the DMD bus. The I registers (I0-I3 in DAG1 and I4-I7 in DAG2) contain the actual address used to access memory. When data is accessed in indirect mode, the address stored in the selected I register becomes the memory address. The data address generator employs a post-modify scheme. After an indirect data access, the specified M register (M0-M3 in

DAG1 and M4-M7 in DAG2) is added to the specified I register to generate the update I value. The modification values stored in M register are signed numbers so that the next address can be either higher or lower. The address generator supports both linear addressing and circular addressing. For circular buffer addressing, the L register is initialized with the length of the buffer. For linear addressing, the modulus logic is disabled by setting the corresponding L register to zero.

A common requirement for DSP filters is to have a circular buffer. The processors' data address generators (DAGs), using the L (length) registers directly implement this. First the buffer must be declared as circular. `.VAR/DM/CIRC coefficients[256];`
This identifies it to the linker for placement on the proper address boundary.

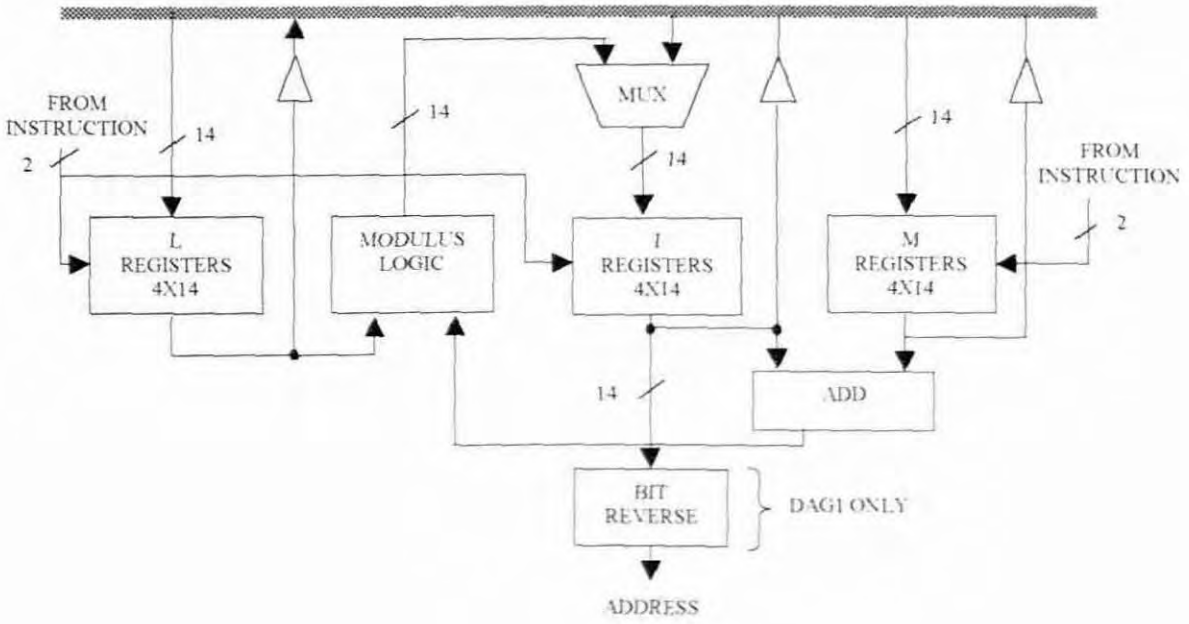


Figure 10.7 Data Address Generator block diagram

Next, the L register must be initialized by using the assembler's % operator (or a constant). The I register is initialized by using the assembler's ^ operator and the M register must have a constant to specify to what number of locations are to be incremented each time. For example:

```
L0 = %coefficients;    {length of circular buffer}
I0 = ^coefficients;    {point to first address of buffer}
M0 = 1;                {increment by 1 location each time}
```

Then a statement like

```
MX0 = DM(I0,M0);     {load MX0 from buffer}
```

placed in a loop, cycles continuously through *coefficients* and wraps around automatically. This function is used in the Audiometer to access the sine values in the sine lookup table. It is also used in the HLS in the main filter routine and its implementation is discussed further in chapter 11.

The transfer of data is achieved with the use of five internal buses namely: the Program Memory Address (PMA) Bus, the Program Memory Data (PMD) Bus, the Data Memory Address (DMA) Bus, Data Memory Data (DMD) Bus and Result (R) Bus. The internal result (R) bus connects the computational units so that the output of any unit may be the input of any unit on the next cycle. The two address buses (PMA and DMA) share a single external address bus, allowing memory to be expanded off-chip and the two data buses (PMD and DMD) share a single external data bus. Byte memory space and I/O memory space also share the external buses. These busses provide the connection between the core and the ADSP-2181 integration.

Before the ADSP-2181 integration is discussed, it is worth mentioning that the ADSP-2181 architecture exhibits a high degree of parallelism, tailored to DSP requirements. In a single cycle, the device can:

- Generate the next program address.
- Fetch the next instruction.
- Perform one or two data moves.
- Update one or two data address pointers.
- Perform a computation.

In that same cycle, processors, which have the relevant functional units, can also:

- Receive and/or transmit data via the serial port(s).
- Receive and/or transmit data via the DMA ports.
- Receive and/or transmit data via the analog interface.

10.6 The ADSP-2181 integration

Figure 10.8 shows the additional ADSP-2181 features that are being integrated into the ADSP-21XX family core. These features are: Program memory, Data memory, Serial ports, timer, interrupts, flags, Byte DMA controller, power down logic, programmable I/O and internal DMA port. As said, only the relevant features will be discussed briefly and for a complete reference, the ADSP-2181 user manual should be consulted.

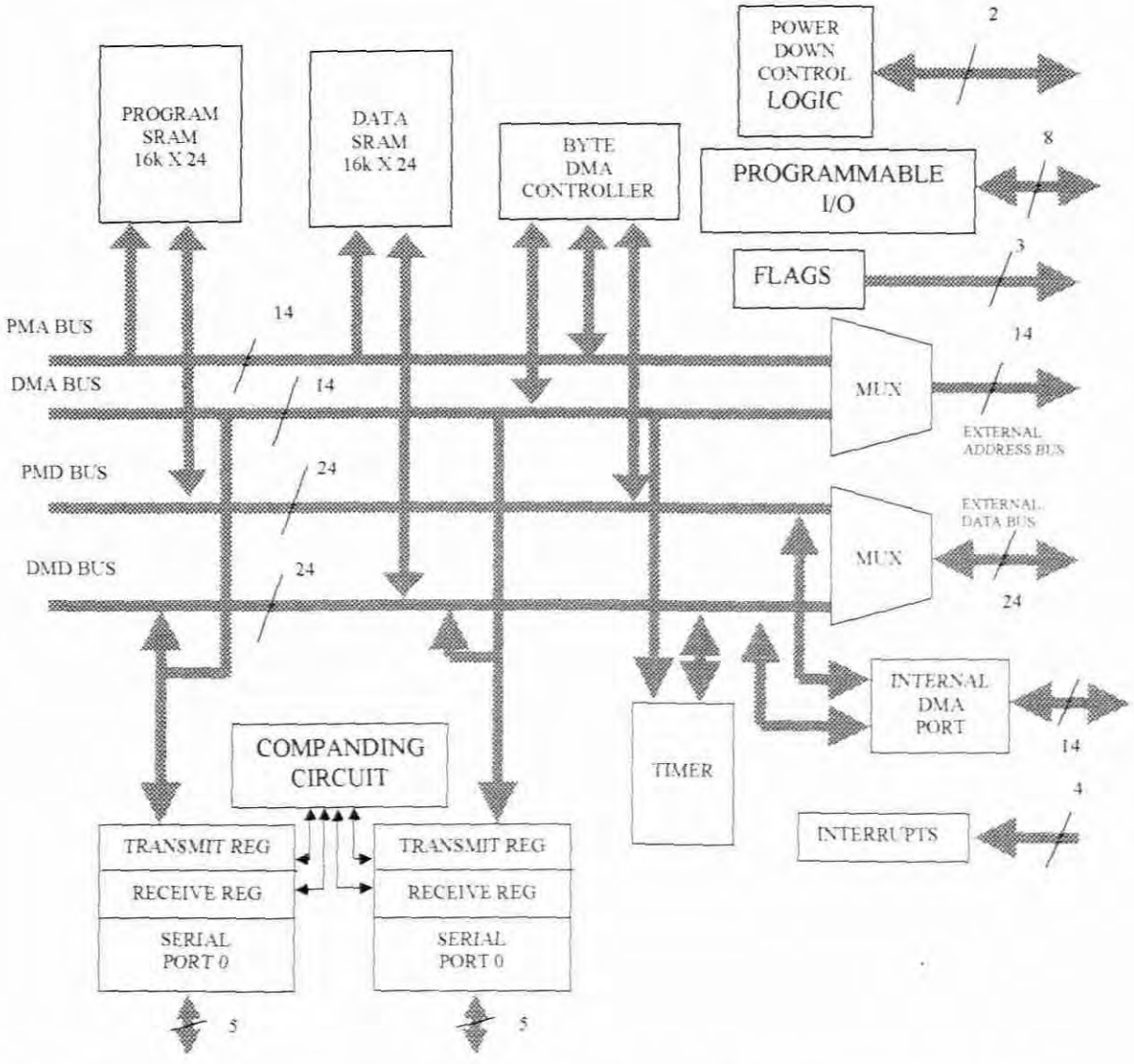


Figure 10.8 ADSP-2181 integration into the ADSP-21XX family Core Architecture

Memory - The ADSP-2181 uses a modified Harvard architecture in which data memory stores data, and program memory stores both instructions and data. It contains onchip RAM that comprises a portion of the program memory space and data memory space. The speed of the on-chip memory allows the processor to fetch two operands (one from data memory and one from program memory) and an instruction (from program memory) in a single cycle.

The ADSP-2181 has 16K Data and 16K Program memory (RAM) onboard. Boot circuitry provides for loading on-chip program memory automatically after reset. This is done through the memory interface from a single low-cost EPROM, through the BDMS port.
Serial Ports - The serial ports (SPORTs) provide a complete serial interface with the host. Each SPORT can generate a programmable internal clock or accept an external clock. SPORT0, which is connected to the CODEC, is used as the input and output for the hearing loss simulator. It is also programmed to accept an external clock, which is necessary for linear buffering for the sinewave generation in the audiometer.

The high-speed synchronous serial port SPORT0 carries all the data, control and status information between the DSP and the CODEC. SPORT1 pins are used to communicate with the host PC via the RS232 interface. Flag In and Flag Out pins carry the receive and transmit data. The receive data also goes to IRQ1 (interrupt number one) so the DSP can detect activity without polling the Flag In pin. Software running on the DSP emulates a UART to provide the proper protocol for asynchronous serial communication at a data rate of 9600 bits per second. This software is supplied with the EZ_KIT Lite. Figure 10.9 shows the block diagram of the serial port. Writing to a SPORT (serial port) TX register readies the SPORT for transmission; the TFS signal imitates the transmission of serial data. Once transmission has begun, each value written to the TX register is transferred to the internal transmit shift register and, subsequently the bits are sent, MSB first. Each bit is shifted out on the rising edge of SCLK. After the first bit (MSB) of award has been transferred, the SPORT generates the transmit interrupt.

The TX register is now available for the next data word, even though the transmission of the first word is ongoing. In the receiving section, bits accumulate as they are received in an internal receive register. When a complete word has been received, it is written to the

Rx register and the receive interrupt for that SPORT is generated. SPORT0, in particularly, is configured to be triggered with external clock pulses from the CODEC.

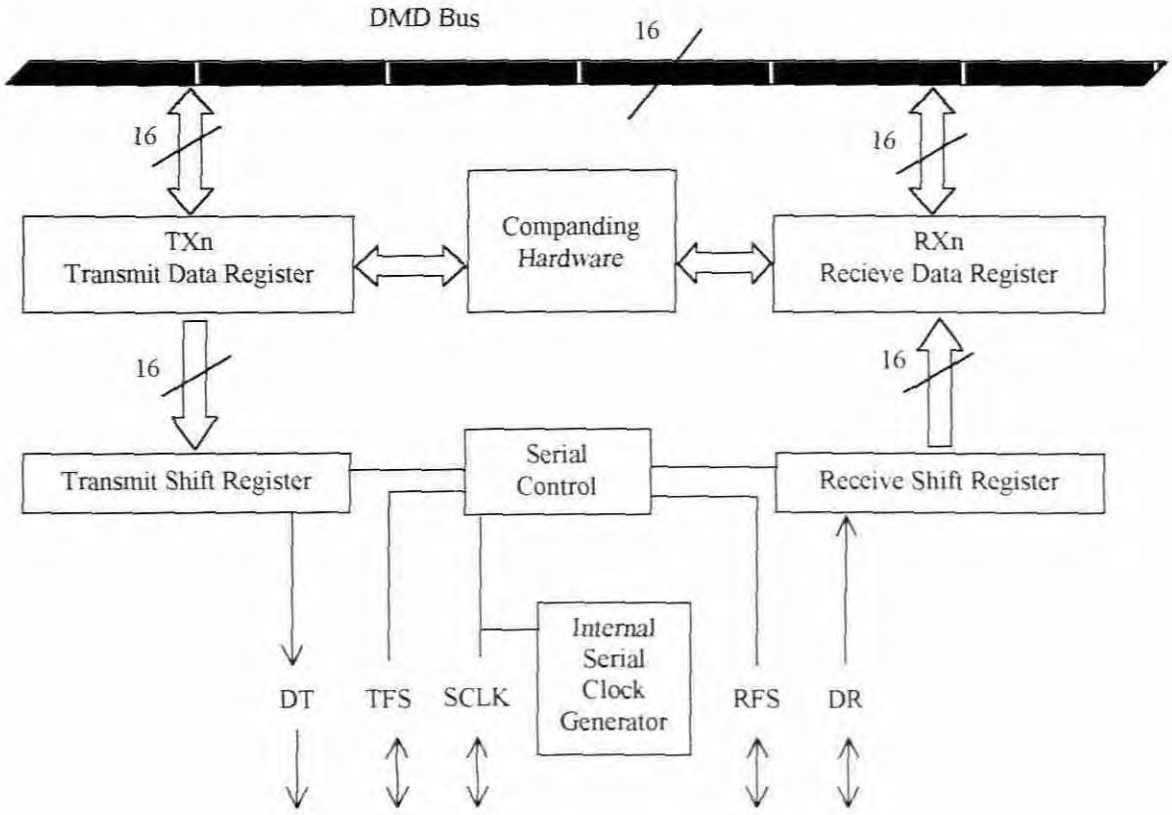


Figure 10.9 Serial port block diagram

The CODEC is configured to sample at 32 kHz. This is twice the highest frequency at16 kHz, which adheres to the Nyquist constrain.

An interrupt controller allows the processor to respond to eleven possible interrupts and reset with minimum overhead. These interrupts are prioritized from high to low in the following order: Power down, $\overline{\text{IRQ2}}$, $\overline{\text{IRQ1}}$, $\overline{\text{IRQ0}}$, SPORT0 Transmit, SPORT0 Receive, $\overline{\text{IRQE}}$, BDMA Interrupt, SPORT1 Transmit or $\overline{\text{IRQ1}}$, SPORT1 Recieve or $\overline{\text{IRQ0}}$ and the Timer interrupt which has the lowest priority. The program sequencer's interrupt controller responds to interrupts by shifting control to the instruction located at appropriate interrupt vector address. The interrupt vector table resides in the first forty locations of the DSP program and is the main determinant for the program flow.

It has repeatedly been mentioned that only the most relevant features of the ADSP-2181 integration would be discussed as an aid to the explanation of the hearing loss simulator. It is therefore recommended that for the other features such as the: Power down logic, Byte DMA controller, internal DMA port, Flags, timer and Programmable I/O, the ADSP-2100 user's manual (that includes the ADSP-2171 and ADSP-2181) by Analog Devices (1995) be consulted.

10.7 Conclusion

This chapter overviews the necessary areas of the DSP architecture to explain how the algorithm was implemented. The Audiometer was discussed primarily to show the relationship with the hearing loss simulator in terms of the hardware. Both functions are controlled from the host which is a PC. The PC stores the ear's frequency response during a pure tone audiometrical test and then uses this information to perform the IFFT to determine filter coefficients. The DSP side, on the other hand, fulfills the sinewave generation for the hearing test and performs the filter function for the hearing loss simulator. To realize the DSP functions the EZ_KIT Lite was used. The EZ_KIT Lite is a complete DSP development system with a ADSP-2181 DSP processor, CODEC, RS-232 serial interface capabilities and EPROM on board. The chapter concluded with the most relevant features of the ADSP-2181 processor, which are necessary for the next chapter which deals with the actual implementation (software) of the HLS.

CHAPTER 11

THE ACTUAL IMPLEMENTATION

11.1 Introduction

In chapter 9 the implementation of the IFFT was motivated. The discussion on the implementation is thus based on this method. The actual implementation is divided into two sections: The actual implementation on the host side and the actual implementation on the DSP side. The two essential components in the development of the HLS are the IFFT computation and the filtering operation. The strategy is to compute the IFFT on the host side and perform the filtering on the DSP side.

11.2 The actual implementation on the host side

11.2.1 Introduction

The software development for the HLS on the host is integrated into existing software. The existing software is that of an Audiometer (As part of an acoustic analyzer) designed by N. Thys for a B.Tech industrial project. The only addition to this program is the implementation of the IFFT algorithm. In this section the main focus will be on the IFFT algorithm and a brief discussion on the serial communication and user's interface will be included. All higher language development was done with Visual Basic, since the existing instrument is written in Visual Basic.

The outline of this section is as follows:

- Visual Basic
- Simulation implementation
- Serial communication
- User's interface

11.2.2 Visual Basic

Visual Basic is a high level language that works on an event driven principle. With Visual Basic the user's interface (screen layout) has to be built in advance, before the program coding is done (Gurewich & Gurewich,1993).

So there are two steps:

- The visual programming step
- The code programming step

In the visual programming step, objects are placed inside a form and their respective properties are set. In the code-programming step, procedures are selected by selecting an object and the event, and then the code is inserted inside the procedure. This code is executed during run-time whenever the event occurs. When the screen layout was designed the icons of the controls were selected from the given provided. Each control has its own set of properties. The complete program list can be found in appendix B. The complete program is not in sequential order but the events are listed in alphabetical order. The code in each event is, however executed in sequential order.

11.2.3 Simulation implementation

Software development for the simulation starts at page E.18 and can be found in appendix B. In most cases the hearing ability of the two ears is not identical. The program thus starts with the left ear and is repeated for the right ear. The section of the program responsible for HLS starts with the declaration of all variables. The program first determines whether the audio-test has been completed before it proceeds, because the simulation would be inaccurate if the test were not completed. The program will be discussed in sections and in the order in which it is executed. A linearpiece wise interpolation scheme is used to get a smooth frequency response before the IFFT is computed.

Section A on page E.20 shows a table with lookup values for sound pressure level (SPL) levels used in the interpolation. These values were adopted from the audiometer development. The standard audiometer has a sound pressure level scale from 0 to 100 dBs in steps of 5 dBs. A program using C++ was written to calculate these values, which are stored in a file called AMP.DAT. The code for this program is shown below in table 11.1. It was calculated according to the formula: dBL = 20 log (Xo/Xref).

```
/*  PROGRAM TO CALCULATE AMPLITUDE VALUES  */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
void main()
{
FILE *dbg0;
float xout,x1;
int     i,amp;
        dbg0=fopen("dbg0.dat","w+");
        for(i=0;i<105;i+=5)                     /* Steps of 5 dBs */
        {                                       /* dBL = 20 log (Xo/Xref).*/
        x1 =((float)(i)*(-1.0))/20.0;           /* -idB/20 = log(Xout/Xref) */
```

```
        xout = 32768.0*pow(10,x1);          /* make Xout subject of formula */
                                            /* Xref = 32768 (highest value for 1)*/
        amp = (int)(xout);                  /* round to nearest integer */
        fprintf(dbg0,"%x\n",amp);           /* convert to hexadecimal */
              }
 fclose(dbg0);
}
```

Table 11.1 Program to calculate amplitude values

The values derived from running the program were verified empirically with a SPL meter. Those numerical values that did not correspond to the measured SPL were readjusted. It should be stated at this point that it was not expected that the calculated dBL values of the program in table 11.1 should exactly match the real sound pressure level. To calculate the power of the actual sound pressure levels in terms of electrical power consumed by the speaker, special mathematical calculations need to be performed. These formulas in the calculations take into account all electrical impedances as well mechanical impedances. It is well known in the field of audio engineering that apart from the problems in finding the correct transducer's characteristics and specifications, the numerous specifications needed in these calculations all have their respective tolerances. When these tolerances enter the applicable equations, the results will have accumulative tolerances. This might lead to inaccurate values, which will require continuous recalculations. To avoid the tedious and time consuming calculations and recalculations it was decided to measure the relative sound pressure level with a SPL meter and then adjust the numerical values in the lookup table. This approach is not only more accurate, but also faster.

In section B, the coordinates of a test point on the audiogram are used to retrieve the corresponding SPL value from the lookup table in section A.

In section C the points on which the tests were performed are interpolated linearly. The horizontal grid is divided into 256 equal spaces. Each point is calculated by the straight line equation $y = mx + c$ where m is the gradient and c the starting point.

Section D shows the program that was run to literally test the speed of the IDFT. Equation 4.27 is directly implemented by noting that $e^{j\theta}$ can be written in the trigonometric form as $\cos(\theta) - j \sin(\theta)$. This part of the program is eliminated though.

Section E shows the various windows that were experimented with.

Section F is where the computation of the inverse fast Fourier transform begins.

The first section explains Fourier time shifting. In the second loop an important general property of the DFT is taken care off. These properties refer to the spectral symmetry which is:

$$|X[k]| = |X[N-k]|$$
$$\angle X[k] = -\angle X[N-k]$$
$$\mathrm{Re}\,X[k] = \mathrm{Re}\,X[N-k]$$
$$\mathrm{Im}\,X[k] = -\mathrm{Im}\,X[N-k]$$

The values $X[N-k]$ which range from π to 2π are known as the complex conjugates and is written as $X^*[k]$. This loop demonstrates symmetry at frequency sample $Ns/2$ where Ns are the number of frequency samples.

In section G, −1 is assigned to IN which means that the inverse Fourier transform is calculated. As previously stated the same routine could be used to perform both forward and reverse transform, so that if IN = 1 the forward transform is calculated.
In section H the twiddle factor is calculated. Referring to figure 5.2 in chapter 5 the position of the twiddle factor $W_N^{kn} = e^{-j2\pi kn/N}$ in the basic butterfly can be noted. In this section the twiddle factor is expressed in trigonometric form. The imaginary part is multiplied by IN = -1, so that the IFFT instead of the FFT is calculated.

Section I is where Bit reversal is performed. The section could be explained by means of a flow diagram as shown in figure 11.1. XR is an array variable for the real parts and XI is an array variable for the imaginary. In the program the same operation is performed on both XR and XI so that only XR can be used in the discussion. The program starts with both IO and JO equal to 0. When the first conditional jump is encountered the flow of the program is directed toward the branch where the content of XR(IO) and XR(JO) is exchanged. Both are zero which means that the content of the first memory location remains in place. If the number of sample Ns is, for example, 32 then the next step will assign K = 16. The next block tests if K >= JO+1 in which case it is "true". K is then added to JO so that JO is now 16. When the loop starts again, IO will be one. In this case IO is smaller then JO which means that content of XR(1) and XR(16) is exchanged. When the program is iterated Ns time the resulting sequence will correspond to the bit reversal sequence explained in chapter 5.

The exchange of the memory content of position 15 and 16 is examined as an example. The decimal number 15 will have a binary value of 01111 and bit reversal value of 11110 that is 30. In this case JO = 30 and K = 16. The test "Is K >=JO+1 ?" will result in "No", which means that in the next step K is subtracted from JO so that JO is 14. The next block where K is divided by 2, results in K = 8. The loop labeled *A* might be executed a few times until K is bigger then JO+1. The test "Is K >=JO+1 ?" will again be executed and result in "No" which mean that the loop is executed a second time so that JO = JO – K = 14 – 8 = 6 and K/2 = 4.∴K is now 4. The test "Is K >=JO+1 ?" is repeated and will result in No. It means that the loop will be repeated a third time so that JO = JO – K = 6 – 4 = 2 and K/2 = 2.∴K is 2. "Is K >=JO+1 ?" is repeated again and will result in No. It means that the loop is once again repeated so that JO = JO – K = 2 – 2 = 0 and K/2 = 1.∴K is 1. This time around when "Is K >=JO+1 ?" is repeated it will result in Yes. The next step will add JO and K so that JO equals 1.

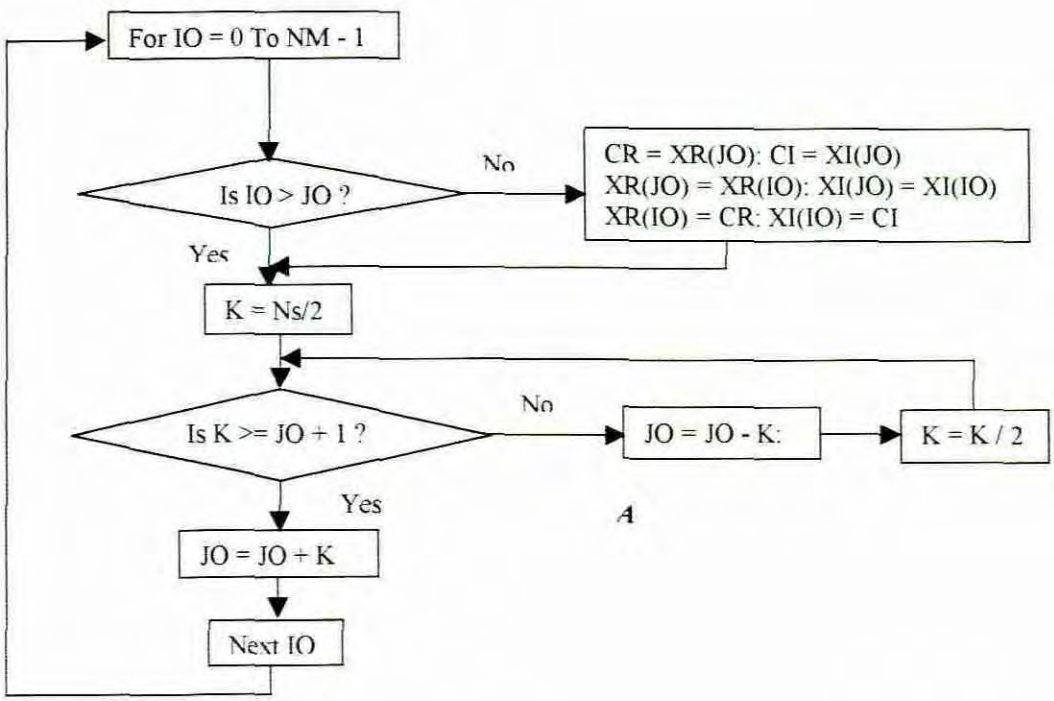


Figure 11.1 Bit reversal flow diagram

With the next iteration IO is 16 and JO is 1. IO is bigger then JO which means that no exchange between XR(IO) and XR(JO) will take place. The exchange need not take place anyway because XR(16) and XR(1) would have been exchanged already during the first iteration. The value 16 has a binary equivalent of 10000 and the bit reversal will thus be 00001 which is decimal 1. This explains and proves the operation of bit reversal part of the

program. Section J is where the FFT computation is done. Referring to figure 5.5 in chapter 5 it could be seen that the computation is started by taking the input sequence as N, 1-sample transforms and proceeding from left to right computing 2, 4, 8,...sample DFTs in successive waves of butterfly calculation until the full N-sample DFT is formed from two N/2 sample DFTs. The first line of section J is where the outer loop starts, which is responsible for accessing each stage in succession up to M stages. The first step in the decimation in time algorithm is to compute the N/2, 2-sample DFTs from the shuffled input. The inner loop does this calculation by accessing each butterfly at a time. The butterfly arrangement figure 5.2 (b) is repeated here for the sake of explanation.

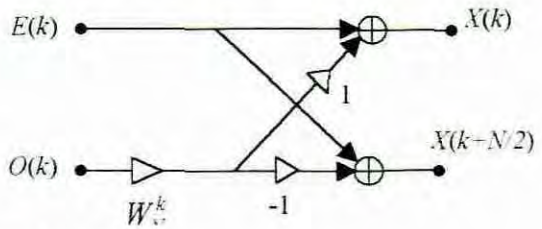


Figure 11.2 Butterfly arrangement

Lines 7 and 8 from the start of section J computes W_N^k in trigonometric form. The *cos* and *sine* values of the twiddle factors which was precomputed and stored in a lookup table are retrieved by lines 11 and 12. The precomputation adds to the increase in speed. It should be noted that when the inner loop is executed it uses the same twiddle factor for each butterfly. That is why the twiddle factor is computed outside the inner loop. For the first stage L = 1, LE = 2 and L1 = 1. XR and XI uses IP = I+1 in line 7 and 8 because IP would then refer to the bottom leg of the butterfly which is the next line downwards in the DIT structure. In the bottom leg W_N^k is multiplied. Line 10 computes X[k] and line 9 computes X[k + N/2]. It can be seen that 'in place' computations are done by which the same memory location is used to store the input sequence, the results of the intermediate DFTs and the final output sequence. This contributes to an efficient use of memory. With LE = 2^1 = 2 in the first stage every second pair of inputs is processed in succession. In the middle loop, J would run from 0 to 0 which means that the middle loop is executed only once. The N/2, 2- sample DFTs are thus computed in the inner loop.

When L = 2 the second stage is executed so that N/4, 4 sample DFTs are computed. In this case LE = 4, L1 = 2, IK = N/4 and J runs from 0 to 1. When J = 0 the inner loop will start from I = 0, with IP = 0+2 = 2. CR and CI are respectively real imaginary values computed for the bottom leg of the butterfly by which the inputs XR(IP) and XI(IP) is multiplied by

the twiddle factor. The bottom leg of the butterfly is now two lines down in the structure of the second stage. The final output values of XR(I) and XR(IP) will thus be two spaces apart. When the inner loop is executed for the second time it will jump LE = 4 positions down the structure and repeat the same procedures using the same twiddle factor. The middle loop will be executed a second time with J = 1. This time the inner loop starts with I = 1, which is the position of the top leg of the second butterfly from the top in the structure of the second stage. The values of the bottom leg are calculated similarly as described above. The same procedures are repeated when the jump of LE = 4 in the loop is encountered. The above procedures produce the N/4, 4-sample DFTs for the second stage. The other stages are computed in the same way using their relative twiddle factors. The process is continued until the full N-sample DFT is formed from two N/2 sample DFTs.

In section K the outputs XR and XI are divided by the number of samples, Ns, so that the IFFT computation can be completed.

11.2.4 Serial communication

The procedure to initiate the serial port for communication is listed in section O page E.13. It shows that in the sub Form_Load procedure serial port COM 2 is initiated. The settings for this port are: 9600 baud rate, no parity, 8 data bits and 1 stop bit. The port is then open, and after Chr(25) is sent out to instruct the DSP to start, the port is closed.

In section M the calculated filter coefficients are sent to the DSP side through the serial port Comm. Port2. By sending Chr(27) down the port, the DSP side will respond by preparing for the reception of the coefficient value. All the values in the lookup table had to conform to the 1.15 format as required by the ADSP_2181 processor. The negative values are calculated by adding the coefficient value to 65535 (0xFFFF) convert it to the 1.15 format as explained in chapter 10. Referring to figure 9.3 it can be seen that in the 1.15 format the negative values ranges from Hex : 0xFFFF = Dec : -0.000031 to Hex : 0x8000 = Dec : -1.000000.

The serial port uses eight bit words in its data transfer. This would consist of two hexadecimal digits. The coefficient that needs to be transferred consists of four

hexadecimal digits. To be able to transfer the four hexadecimal digits, it should obviously be split into two pairs of hexadecimal digits which could be transferred a pair at a time. The next step would be to split the number into higher and lower significant hexadecimal digits. Dividing the calculated output value by 256 and rounding it to nearest lower digit by means of the 'int' function derives the higher significant hexadecimal digits. As an example, a number 24852 with hexadecimal equivalent 0x6114 is divided by 256 which gives

$$24852/256 = 97.078125$$

Round with the int function would give 97 with Hex = 61 which is the first higher significant hexadecimal digit. The LSD is obtained by multiplying the 97 by 256, which equals 24832 and then subtracting it from the original number. For example 24852 – 24832 = 20. When 20 is converted to hexadecimal it gives 14 which is the lower significant digits of the hexadecimal number 0x6114

In section N the higher and lower significant bits are sent out comm.port2

The whole process is then repeated for the other ear.

11.2.5 User's interface

In this document the user's interface refers to the man-machine interface. The screen layout is shown in figure 11.3. As previously said, the simulator is integrated into and exiting from an audiometer design by Thys (1996) and has therefore only a few buttons added to the existing layout. The user's interface of the existing Audiometer will be discussed briefly, followed by the additional features that constitute the HLS.

The horizontal scale of the audiogram indicates the frequency, ranging from 125 Hz to 8 kHz. The vertical scale of the audiogram indicates the SPL, ranging from 0 dB's to 100 dBs in steps of 10 dBs, but the points can be plotted in steps of 5 dBs. When the program is run for the first time, the SPL is automatically set to 0 dB's and Left channel enabled. Frequency tests will start automatically from 250 Hz. Sound stimulation will take place when the Audiomet button is clicked on. The sound signal will only last for duration of 1 second. When the vertical scrollbar is shifted, the amplitude window indicates the increase or decrease in SPL. When the horizontal scrollbar is shifted, the frequency window indicates the increase or decrease in frequency. When the subject indicates that the sound is audible, the Log button is clicked and a circle for Right or a cross for Left is plotted on

the Audiogram. After the spectrum is covered, the draw button can be triggered to draw the lines between the points. If the test was incomplete the text box " Test completed ? " will display the message " incomplete". By clicking on the left or right option the required channel can be selected. If a new test needs to be carried out , the Audiogram can be cleared by clicking the Clear button and the name of the student is entered in the name text box. The Audiogram can be printed when the print button is clicked, and clicking the Exit button ends the program. For further information on the audiometer, refer to Thys (1996).

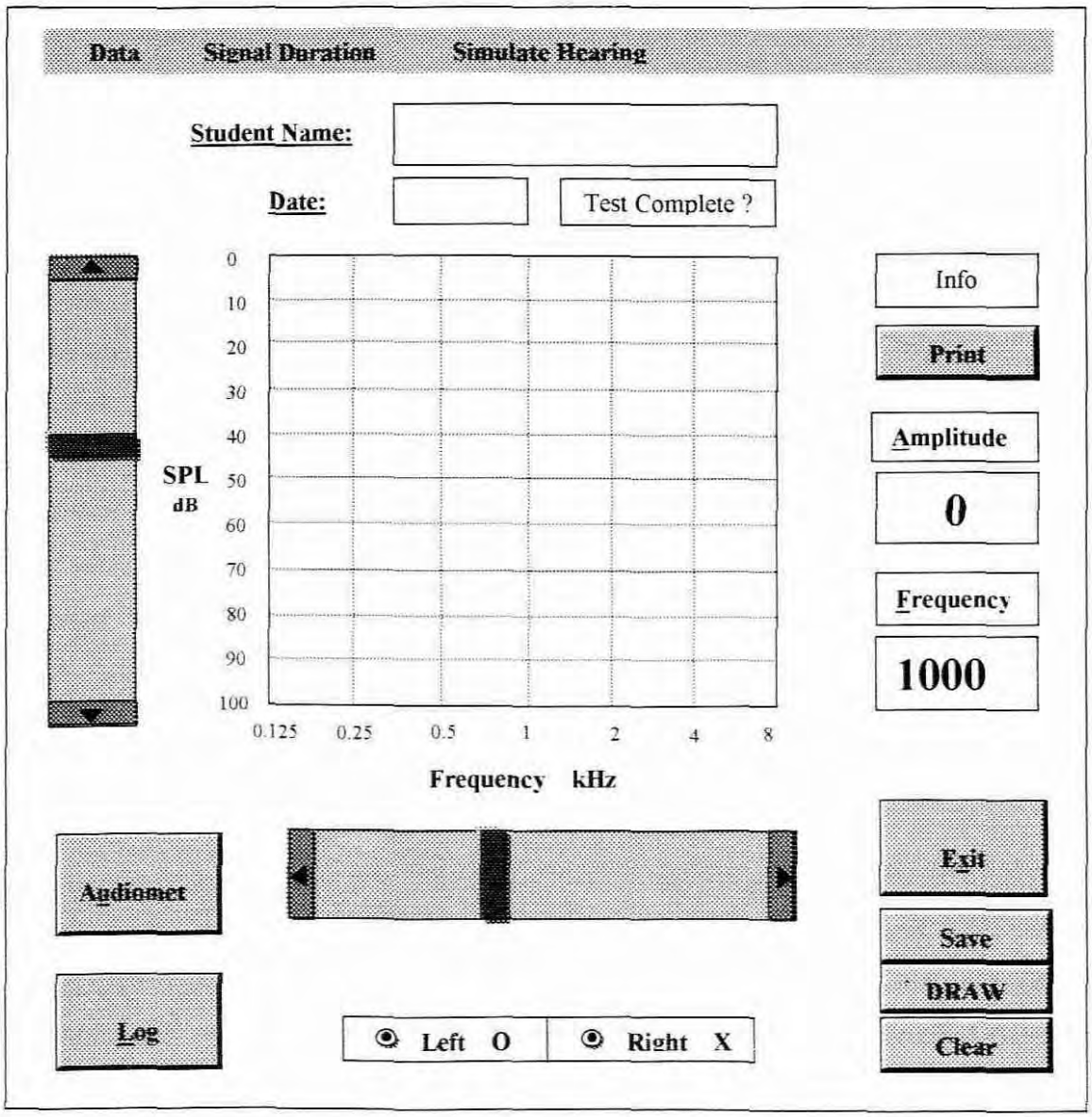


Figure 11.3 Screen layout of Audiometer and HLS

The first addition to the existing instrument is a student/patient database that stores the audiogram information. The information is retrieved from the database when the simulation needs to be performed. The construction of the database will not be discussed in full detail, as it is not the focus of the research. For a complete explanation of the construction of databases with Visual Basic 3, refer to Gurewich and Gurewich (1993).
The database program starts on page E.13 section DA. The data is stored in a file named naud2.dat, at the click of the save button. The code for the click procedure of the object "cmdsave" is shown in section DB page E.11 and is executed when the save command button is clicked. To navigate through the database, the menu was developed in such a way that a list of options is presented by means of a dropdown list. When the Data option is clicked a dropdown list with the options New, Next, Previous and Search is presented. The code for these events can be found in sections DC, DD, DE and DF respectively.

Another addition would obviously be an option to choose between audiometer and simulator. By selecting "Simulate hearing" on the menu-bar and clicking "Simulation" on the dropdown list, the simulation of the hearing of the person currently displayed is performed. Clicking on the "No attenuation" option will enable the audiometer. Shortcut keys can also be used operate the instrument. The letters that are underlined are the ones which are used with the Alt key.

11.3 Low level development on DSP side

The low level development on the DSP side is a program written in assembler language for the asm2181 compiler and the complete program list is shown in Appendix A. The program consists of two parts which are the Audiometer and the HLS. Both functions use the same code to initialize all the EZ-KIT Lite hardware. For a full description on programming the EZ-KIT Lite development board (with onboard ADSP-2181 DSP processor), refer to the ADSP-2100 EZ-KIT Lite Reference manual written by the editorial stuff of Analog Devices (1995). For a full description on the operation, configuration and programming of the ADSP-2181 processor, refer to the ADSP-2100 (which includes the ADSP-2181) user's manual, also written by the editorial staff of Analog Devices (1995).

In the program list of appendix A, section B shows the declarations that are made of all the additional variables, arrays and constants that are made. When the system starts, code will

be executed from location 0 in program memory. The first 48 locations of program memory include the interrupt vector table. Program execution is based on a prioritized interrupt vector table. When an interruption occurs during ADSP operation, the program flow will be redirected to the location shown in the listing (Appendix A). This jump is done automatically by the hardware when an interrupt is detected. Line A of the program shows that the constant declarations are contained in a separate file called system.k. The listing of system.k is shown in Appendix A.

Section B of the program includes the assembler directives defining 6 circular buffers on chip memory : two in program memory (used to store the values of the sine lookup table and the filter's coefficients), four in data memory RAM (used to hold the Codec's control words and initialization commands and also the sampled data). Thirty-two variables in data memory, with their respective labels, are shown in the listing under section B.
Section C shows the initialization of the sine values, frequency values , amplitude values, and codec initializations. The sinewave values are actually loaded from an external file (sine.hex) by the linker. The frequency values (freqTab) are loaded from freqval.dsp and the seven amplitude values(ampTab#) are loaded from their respective ampval#.dsp files. Here # means numerical values from 1 to 7. All Codec interface (software and hardware) has already been done on the EZ_LIT kit and the remarks next to the control word gives an idea of how the codec is configured.

Section D shows interrupt vector table. Since this code module is located at absolute address zero (as indicated by ABS qualifier in the .MODULE directive), the first instruction is placed at the start vector: address 0x0000. The instruction *jump start* (at location 0) is used to jump over the interrupt vector table to where the *start* routine begins. Interrupt vectors that are not used are filled with return from interrupt (rti). This is done for safety. If for some reason program memory gets corrupted or program flow gets "lost" in the interrupt vector table, the rti instruction will bring program flow back into the program.

Section E, start, sets up the index (I) , length (L), and modify (M) registers used to address the three circular buffers. A non-zero value for the length activates the processor modulus logic. Each time the interrupt occurs, the I register pointers advance one position through the buffer.

At F the file i_codcom.dsp is included which restores the monitor timer handler, performs the configuration of SPORT0 for external clock pulses and does the Codec initialization. This file is listed at the end of in Appendix A. Section F also shows that the starting address of the table named *coeftab* is put into index register *i6*. This table holds the 256 coefficients needed for the filtering.

Section G shows that at start up the counter that is used to enable the sound stimulation for one second, is assigned with cnt = 0x7d00. The flag to begin the tone generation is set to zero and the frequency is initially set to 1000 Hz. cntra is used to count the number of bytes and is therefore set to 2.
Section H shows the code for establishing the user's interface by making use of the Monitor Program which resides in the on-board EPROM. It is automatically loaded into the on-chip program and data memories at reset. As soon as the monitor begins execution, it performs a self test of the DSP registers, on-chip memories, and a reset and initialization of AD1847 Codec. It then waits for commands via the RS-232 serial communication port SPORT1. After download of the application program is completed, the monitor software calls the application program as a subroutine. That is why this routine in H ends with a return to subroutine (rts). The first eight lines of the main program in section H reads a command word from the host and loads it into the ax1 register. It should be noted that two types of information are received by the DSP. On the one hand, the filter's coefficients are transferred and on the other hand, the audiometer's frequency and amplitude values are transferred. First the transfer of the Audiometer's information and then the transfer of the filter's coefficients are discussed.

The transfer of the amplitude and frequency values used by the Audiometer is as follows:
Section G shows that the first step in the program is to test ax1 if 0x001b to see if it is received. 0x001b signals the transfer of the filter's coefficients. If ax1 is not 0x001b it is tested for 0x19 and 0x1a, which is used to instruct the audiometer which channel, left or right, to enable. This is done by putting 1 or 0 in the dm(lef) or dm(rig) variables in the lft and rht-sub routines. It is then used to direct the program flow to a routine where either one of the channels is finally cut-off (transmits a zero). The program returns to main without performing any additional function. Any other command word will cause the program to flow through the part where various flags are set and counters are initialized.

The important variables to note now are the cnt that holds the value for a 1 second duration sound stimulation. Also the variable sflag (sound flag) is set to 1. This variable is used to enable the audiometer or shut it down when it is zero. The command word that is in ax1 is then assigned to the variable dm(rxval), which is used to access the content of the frequency and amplitude table. Call lookup is then executed, followed by "jump main".

Section J shows the lookup routine. In this routine the frequency value and amplitude value are retrieved and loaded into dm(freq) and dm(amp). In order to understand this routine, the construction of the control word is explained first.

The control word sent down the serial port is constructed in such a way as to instruct the DSP program to:

1. Change between the frequencies.
2. Change between 21 amplitude values.
3. Select left or right channel.
4. Trigger sound stimuli.

It consists of a start bit, 8 data bits and a stop bit. The diagram below shows the function of each bit in the control data word.

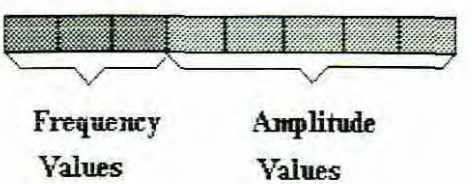


Figure 11.4 Eight bit Control data word

The 5 lower significant bits allows for $2^5 = 32$ variables for amplitude adjustment of which only 21 values are used. The other 3 higher significant bits enable the selection of the 7 needed frequency selections. These values are used to retrieve the actual amplitude and frequency from their respective lookup tables. For example, the data word 10101110 will access the 5 th frequency value which is 2000 Hz in the frequency lookup table and the 18 th amplitude value which is 90 dB in the amplitude lookup table.

The way the DSP program accesses these values is as follows :

To get the frequency values, the received data word is first put into a specific register (AR). The content of this register is then shifted right by five positions so that only the higher 3 bits are left in the shift register(sr). The value in sr register is obviously the

corresponding frequency position that needs to be retrieved. The address pointed to by FreqTab, which is in fact the origin of the frequency lookup table, is assigned to index register i2 , so that i2 points to the table's origin. The content of sr is loaded into modify register m2. Modify(i2,m2) will modify the index register by the content of the m2. Linear buffering is used here, because l2 = 0. (refer to chapter 10, section on data address generators). Index register i2 now points to table offset of the required frequency. ax0 = dm(i2,m2) extracts the frequency value and puts it in ax0. It is then assigned to the frequency variable dm(freq) which is used in the interrupt service routine to generate the required frequency. The amplitude values are extracted from the amplitude lookup table in the same way, but this time they are first ANDed with binary 00011111 to strip the upper three bits. These selected values pointed to by ampTab are smaller than or equal to one (in hexadecimal form) and are multiplied with the sine values to give the actual value that is output onto the CODEC. This is done in the interrupt routine section L. The flow and control diagram of this is shown in appendix C. As the research focuses on the HLS, this chapter will proceed with a discussion on the transfer of the filter's coefficients, while Section K and onwards, which forms the heart of the Audiometer, is explained in appendix C.

Transfer of the filter's coefficients is as follows:

Figure 11.5 shows the program flow and control diagram of the main routine. The last five lines of page D.5 shows that with flag *dlflag* initially zero and on the reception of 0x001b, the program is directed to subroutine *setffl*. In *setffl* the flag *dlflag* is set to 1. In other words 0x001b is used to signal the start of downloading of the filter coefficients. After the program jumps back to main, *dlflag* is tested again and the program will jump to *bran* this time. The function of the subroutines *bran* and *instr0* is to reassemble the 16 bit data word used on the DSP side, as the serial communication transfers only 8 bit characters at a time. This is done as follow: Note that *cntra* was set to 2 initially (page D.5). After 1 is subtracted from the variable *cntra*, the instruction *if eq jump instr0* performs a test to see if *cntra* is zero. If it is not zero then it means that the higher 8 bit has been received. These bits are then shifted up 8 positions and stored in *tempstr1*. When the next character enters this subroutine *cntra* will be reduced to zero and the program will jump to sub routine *instr0*. In *instr0* this 8-bit character (which is in fact the lower 8 bits of the 16-bit data word) is logically ORed with *tempstr1* to give the full 16-bit data word. *fcntr* is used to count if all the 256 coefficients were downloaded. If the transfer of all 256

coefficients has been completed, a call is made to subroutine *stfilf*. In subroutine *stfilf* the download flag *dlflag* is set to zero, the filter flag *fflag* is set to one and the filter coefficient counter *fcntr* is set to zero. This means that the program is ready to receive a new set of coefficients.

When a SPORT0 rx interrupt occurs the program flow will be directed to location *tx_samples* as indicated in the interrupt vector table (page D.4). The first four lines of the subroutine *tx_samples* (page D.9) test the filter flag *fflag* to determine if the sine generation function or the HLS function is to be performed. If *fflag* is one, in which case it is, the instruction *if ne jump startfilter* cause the program flow to jump to subroutine *startfilter*. Figure 11.5 shows that the status of the flag fflag will determine whether the Audiometer or HLS is selected when the SPORT0 rx interrupt occurs.

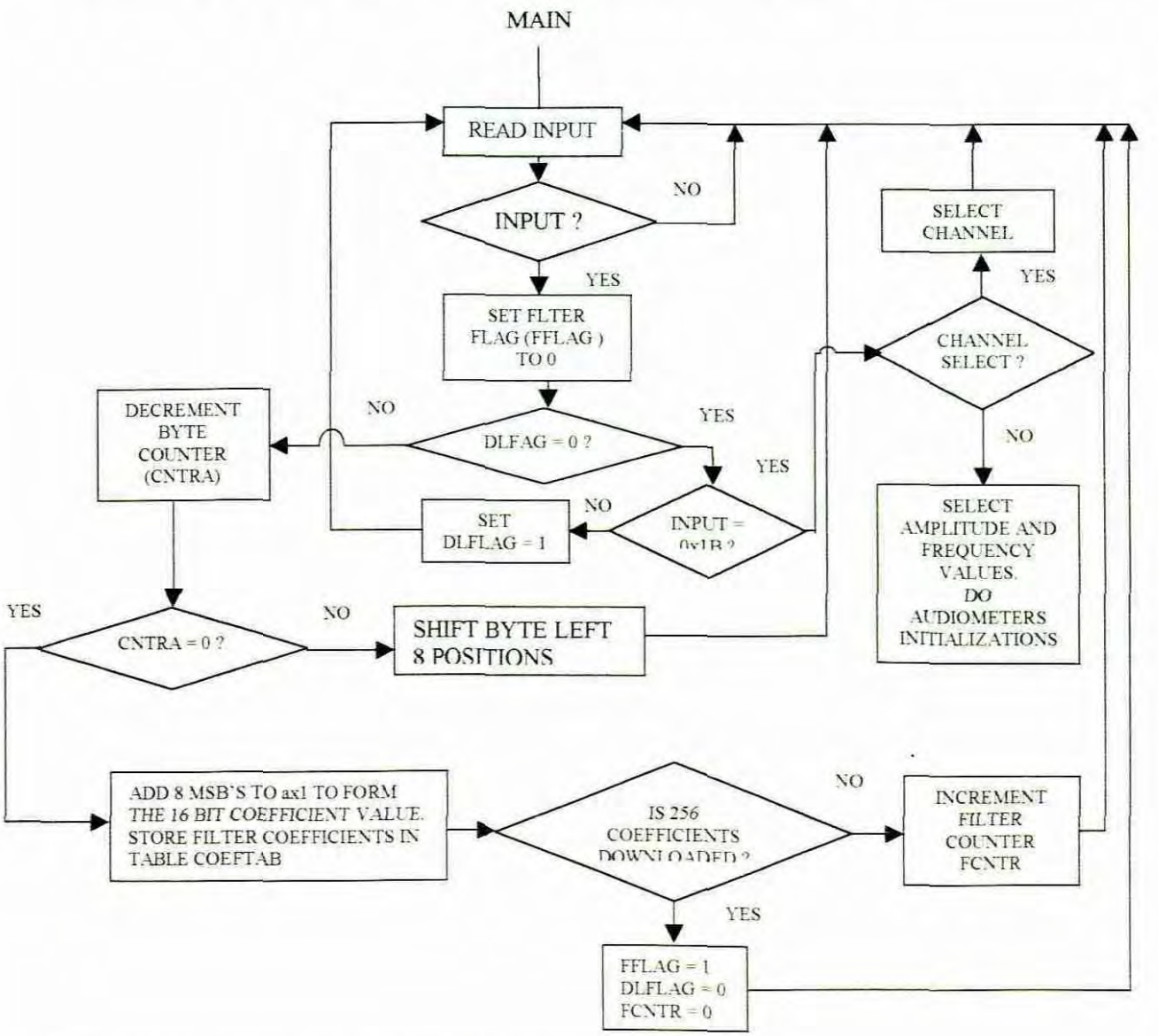


Figure 11.5 Program flow and control diagram

For further details on the program flow and control of the Audiometer refer to appendix C and/or the design documentation of the Audiometer by Thys (1996), as this chapter concludes with the explanation of the filter operation.

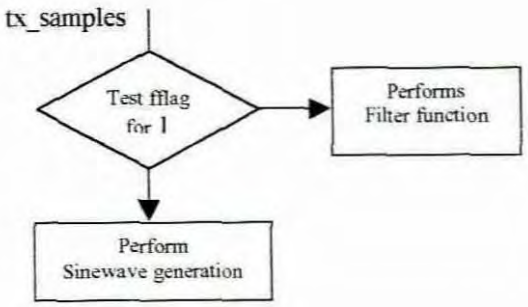


Figure 11.6 SPORT0 rx interrupt Service routine

Section PA of the program in appendix A shows the *startfilter* subroutine where the actual filtering is taking place. A single-precision FIR transversal filter is described in a book 'Digital Signal Processing Applications using the ADSP-2100 Family', by The Application Engineering Staff of Analog Devices and edited by Mar (1995:68) as follows:

" An FIR transversal filter structure can be obtained directly from the equation for discrete-time convolution.

$$y(n) = \sum_{k=0}^{N-1} h_k(n)x(n-k) \qquad 11.1$$

In this equation, x(n) and y(n) represent the input to and output from the filter at time *n*. The output y(n) is formed as a weighted linear combination of the current and past input values of x, x(n-k). The weights $h_k(n)$, are the transversal filter coefficients at time *n*. In the equation, x(n-k) represents the past value of the input signal "contained" in the (k+1)th tap of the transversal filter. For example, x(n), the present value of the input signal would correspond to the first tap, while x(n-42) would correspond to the forty-third filter tap."

The subroutine *startfilter* that realizes the sum-of-product operation used in computing the filter is shown in Listing 11.1 and discussed below.

```
startfilter:
        sr1=dm(rx_buf+2);
        sr1=dm(rx_buf+1);
        dm(i2,m1)=sr1;
        cntr=taps-1;
        mr=0, mx0=dm(i2,m1), my0=pm(i6,m6);
        do fir1loop until ce;
fir1loop:  mr=mr+mx0*my0(ss), mx0=dm(i2,m1), my0=pm(i6,m6);
```

```
    mr=mr+mx0*my0(rnd);
    if mv sat mr;
    sr = ashift mr1 by 1 (lo);
    mr1 = sr0;
    dm (tx_buf + 1) = mr1;
    dm (tx_buf + 2) = mr1;
rti;
```

Program list 11.1 Single-Precision FIR Transversal Filter

The heart of the convolution operation is in the fir1loop. Inside the loop is a multifunction instruction. The first "clause" of the instruction (up to the first comma) is the multiply/accumulate (MAC) needed to perform convolution. In equation 11.1 *x(n-k)* are the unit samples of the input signal. The equivalent of this in the program is the receiver buffer, which contains the unit samples, and is stored in a location in data memory pointed to by the index register *i2*. *I2* holds the address of the beginning of the *data* table (refer to page D.7). *l2* indicate the length of the buffer which was initiated to 256. The content of *i2* is put into register mx0 and the content of *i6* is put into my0. *i6* which contains the filter coefficients, has the equivalent of $h_k(n)$ in equation 11.1 .

The content of the input register mx0 and my0 are multiplied together and added to the contents of the multiplier result register (mr). The second clause, again, loads input register *mx0* from data memory *dm(i2,m1)* and the third clause loads the my0 input from program memory pm(i6,m6). In both cases the index registers are increased by one, as both modify registers, *m2* and *m6*, are one. The Multiply/accumulate operation is repeated 255 times and one more time after the loop to complete 256 iterations. The output *y(n)* is in register *mr1* and put into the output buffer *tx_buf* which is then sent to the codec for digital to analog conversion. The analog signal is sent to the headphones and presented to the person who is investigating the hearing loss.

Part IV

Evaluation and conclusions

CHAPTER 12

EVALUATION

12.1 Introduction

The following instrument characteristics will be evaluated in this chapter:

- Sound pressure level (SPL) and amplitude
- Frequency resolution
- Computational speed limits on the number of taps
- Amplitude values other than the test frequencies
- Phase distortion
- Stability
- Speed
- Quantization error and signal to noise ratio
- User's interface
- Correlation between theory and practice

The chapter ends with 'Problems encountered and further areas of study'.

12.2 Sound pressure level (SPL) and amplitude

The accurate attenuation of the signal is very important. Inaccurate attenuation will result in incorrect simulation of hearing and will consequently lead to false conclusions being made by the teacher. This might have catastrophic educational implications.

The following questions now arise:

* What is the reference level from which the attenuation is taken?
* How accurate are the SPL reductions of the HLS?

The answer to the first question would be that the person who investigates the hearing loss is obviously the reference. His/her hearing is assumed to be normal and is thus accepted as the reference level from which the attenuation is being taken. This point will be picked up again after dealing with the second question.

In order to answer the second question, one first has to understand the audiometer and its SPL levels of which the HLS is an integral part. The output device of the instrument is a pair of headphones, which has its own unique acoustic characteristics. The audiometer generates sinewaves with decimal values ranging from +1 to –1. With no attenuation, or

when the sinewave values are multiplied by one, the output on the headphones should produces a signal with a SPL of 100 dB. Additional amplifiers need to be connected to the CODEC to produce the required sound pressure level, as this is just about the pain threshold. With these amplifiers in place and faced with the problem of reducing the output, one can either reduce the gain of the amplifier or multiply the sinewave values by a fraction of one. With the existing equipment, the multiplication by a fraction would be a better option since it could be done under the control of the computer and *DSP* system, whereas adjusting the gain of the amplifier would have to be done manually. The audiogram SPL scale is in steps of 5 dBs. Each SPL has its own fraction value. These values were obtained by both calculation and empirical methods. The empirical method is the most accurate method, since the output at the headphones is literally measured with a SPL meter and the numerical values are altered until the exact values are obtained. This method results in an audiometer that is more accurate than analog audiometers, which are specified to have a tolerance of ± 3 dB SPL within indication. Each frequency on the audiogram has its own set of values[1] as the frequency responses of the headphone and amplifier are not linear. In other words, each frequency has its own table of values. These are the values that are used by the HLS in its IFFT operation. It could thus be assumed that the HLS is only as accurate as the Audiometer.

In answering the question: "How accurate are the SPL reductions of the HLS?" it should be noted that the person who investigates the hearing loss has to adjust the sound intensity to a level that he/she wants to examine. This level will be the reference level from where the reduction in SPL is taken. As an example, consider the 1000 Hz frequency point. If the reference level is 60 dB and the hearing loss is 40 dB, then it is assumed that the sound intensity experienced by the hearing impaired person will be 60 dB – 40 dB = 20 dB at 1000 Hz. The fraction that corresponds to the 20 dB SPL in the 1000 Hz frequency table is the value that will be taken in the calculation of the IFFT. The other values over the audio spectrum are determined in the same way.

It could be concluded that the accuracy in reduced SPL of the HLS is as accurate as the Audiometer's values in the lookup table. These values are absolutely spot-on for values at frequencies 250 Hz, 500 Hz, 1 kHz, 2 kHz, 4 kHz and 8 kHz, considering the fact that the values in the lookup tables are calibrated with a SPL meter.

[1] The numerical values that are in fractional binary format denoted by 1.15 format, are stored in lookup tables in hexadecimal format

processors specifications t_{mac} = 30 nsec. The maximum number of taps imposed on by the hardware is:

$$N(\text{taps}) = \frac{1}{f_s \times t_{mac}}$$

$$= \frac{1}{32000\ Hz \times 30\ n sec}$$

$$= 1031.25$$

This means that the use of 256 taps in the FIR filter is well within the range for convolution in time domain.

12.5 Amplitude values other than the test frequencies

The coordinates on the audiogram are interpolated to give a total of 256 points on the frequency axis. Establishing the values by a method of curve fitting and performing an IFFT to obtain the filter coefficients for FIR filtering is an enormous improvement on the filterbank method where all the values in specific bandpass filter are forced to the same level.

12.6 Phase distortion

Phase distortion can alter the shape of a sequence considerably, as proven by Kuc (1988:244). Phase distortion is the result of filter designs that have nonlinear phase response. According to Kuc (1988), filters with linear phase response have to be designed to prevent phase distortion. A linear phase response will produce an output sequence identical to the desired part of the input signal, except for a time delay experienced by passing through the filter. To meet specifications with more rigid constraints like linear phase or arbitrary filter shape, Krauss, Shure and Little (1994:1-35) advise the use of FIR design routines, as IIR filters have non-linear phase response which will result in phase distortion. The filters designed by the Modified Yule-walkers equations produce IIR filters which result in phase distortion. This is one of the reasons why the IFFT method is implemented rather than the MYW method.

Phase distortion may also be a problem when the filterbank method with IIR filters is considered. When adjacent bandpass filters are designed in such a way that the addition of

the corresponding samples in the area of overlap results in a flat overall response, then there will be no problem. But if these bandpass filters are IIR filters (which are prone to phase distortion) and have a shifts in phase, then the addition of the corresponding samples (in the area of overlap) may result in what is called either a bump or a ditch. This phenomenon produces what is called amplitude distortion due to the phase distortion. This will produce a type of sound in which the higher and lower pitch components of the sound are not completely in rhythm. Listening to music for instance, through such an instrument may be an irritating experience when the high pitched sound and the low pitched sound are not in synchronous. The outcome of this type of distortion will thus be an inaccurate simulation of the hearing loss. Making use the IFFT method to produce a FIR filter, which has linear phase response, eliminates all problems due to phase distortion.

12.7 Stability

Kuc (1988:247) states that: "FIR filters are always stable, since the poles of the FIR system function are located at $z = 0$ (and at $z = \infty$ for noncausal filters). The design of IIR filters with poles close to the unit circle must be performed with care to ensure that instability does not occur. It is not uncommon to have poles that were intended to be close to the unit circle actually fall outside the unit circle in the implementation, owing to round-off errors in the calculations. In this case, the implemented filter is unstable."

12.8 Speed

It was pointed out in chapter 5 that for a 256 point IFFT, the total number of multiplications (T_{nm}) would be:

$$T_{nm} = \frac{N}{2}\log_2 N$$

$$= (256/2)\log_2 256 \qquad \text{... where } N = 256$$

$$= 1024$$

For the direct calculation of the DFT, the number of multiplications would be $N^2 = (256)^2 = 65536$. It could be concluded that the DIT FFT would be $(N^2)/ (\frac{N}{2}\log_2 N) = 65536/1024 = 64$ times faster than direct evaluation of the DFT. The time for a 256 point IFFT using 16 bit fixed point computation on an 80486 processor running at 100 MHz can be estimated at: $1/f \times T_{nm} = 0.01 \times 10^{-6} \times 1034 = 10.24\ \mu$sec.

A transversal filter subroutine as presented here requires, according to Mar (1992:69), a total of N+6 cycles for a filter of length N. The number of cycles for this application with N = 256 is thus 256+6 = 262 cycles. This means that with a 32kHz sampling rate and a ADSP_2181 running at 33 MHz it will allow 262 cycles for filtering and leave (33 MHz ÷ 32kHz) - 262 ≅ 769 instruction cycles for other operations.

12.9 Quantization error and signal to noise ratio

Digital data are quantized with 2^n levels, corresponding to n bits. Since an operation such as filtering involves a series of computations, care must be taken to minimize the effect of accumulated roundoff error on the accuracy. The following rule-of-thumb has been derived by Higgins (1990:285) for the approximation of best possible (full-scale sinusoid) noise-to-signal (NSR) ratio for quantization noise:

$$\text{NSR (dB)} = -6b + 1.24 \quad \text{[input quantization]}$$

Where b is the number of bits of the registers used. The size of the registers of the ADSP-2181 is 16 bit. Thus the noise-to signal ratio is:

$$\text{NSR (dB)} = (-6 \times 16) + 1.24 = 97.24 \text{ dB}$$

12.10 Users interface

The HLS is incorporated into an existing audiometer and uses, therefore, the same screen layout. The screen layout of the audiometer was developed with the help of audiologists and acousticians who work with these instruments on a daily base. The user's interface is a window-based function. The mouse can be used to activate certain commands as well as short cut keys, which can be used to quickly accomplish a task that is frequently used. A menu bar with dropdown menus was created to prevent the screen from been flooded with irrelevant information. A menu displays a list of commands. The option to uses either the HLS or the audiometer is exercised from the menu bar. The database functions such as search, save, etc. are also done from the menu commands. A unique feature of the design is the use of sliding buttons to adjust the frequency and sound pressure levels. Operating it with the mouse ensures a pleasant interface experience. To simulate the hearing of a specific student, a search is carried out to find the audiogram information of the student followed by the actual Hearing Loss Simulation.

12.11 Correlation between theory and practice

The question comes up "Does theory correlate well with practice?"
In the case of amplitude versus the SPL relation one would not expect any correlation of the theoretical calculated values with the real SPL, but rather an indication of what the amplitude entries of the various sound pressure levels should be. The exact SPL values were determined empirically by means of a SPL meter.

The best way to evaluate the correlation between theory and practice is to perform a freefield audiometrical test as described below.

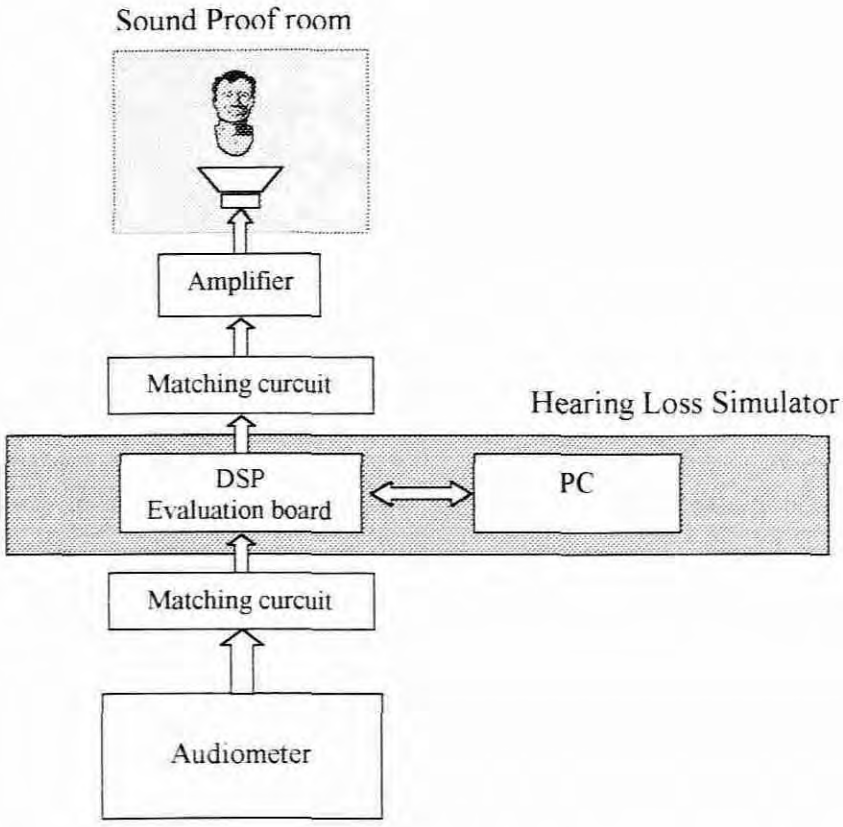


Figure 12.1 Experimental setup to determine correlation between theory and practice

With this setup as shown in figure 12.1 an ordinary hearing test is first performed with the audiometer as described in chapter 2 on a person with a hearing loss. A built in function on the HLS with which one can select between no-attenuation and simulation makes it possible for a signal to be sent from the audiometer through the DSP circuitry without attenuation. Those points plotted on the audiogram are then manually transferred to the

HLS. The HLS simulation function is thereupon selected, whereby the filtering operation is started. The second step is to perform the same test on a person with normal hearing but this time with the HLS operational and functional. It should be clear that if the person with normal hearing produces the same test results as the person with the hearing loss, then the simulator has performed the function of simulating a hearing impairment equivalent to that of the person with the hearing loss. This test described above has been performed in conjunction with Dalvi (An audiologist at Nuwe Hoop Centre for the Hearing Impaired), by which the audiogram information of one of the students, G. Jacobs, with an average hearing loss of 65 dBs where fed into the HLS database. The average hearing loss is determined by the average of the 500 Hz, 1 kHz, and 2 kHz test points.
The same test was performed for Thys (a teacher at Nuwe Hoop Centre, whose hearing is assumed to be normal) but this time with the HLS operating on G. Jacobs's audiogram information. The results obtained from the tests for the two subjects were the same on all frequency points within a $\pm$ 5dB's tolerance level. This outcome proves that there is a high degree of correlation. This test subsequently manifested that what has been set out to be achieved in theory, which is to simulate hearing loss, has in fact been achieved in practice. This small deviation is due to the matching circuits as shown in Figure 1.1 for free field test, which need further refinement. The HLS would obviously produce even more accurate results when it was used with the headphones with which it was calibrated.

The another way to evaluate the correlation between theory and practice is to perform a speech audiometry test. The speech audiometry test, as described in chapter two, should be performed on both the students with the hearing loss as well as a normal hearing person who listens to the same words through the HLS. The degree of correlation between the two subjects will also be a reflection of what the correlation is between theory and practice. Performing this exercise on Jacobs (student at Nuwe Hoop Centre) and Thys (a teacher at Nuwe Hoop Centre) has also proven a high degree of correlation.

12.12 Problems encountered and further areas of studies.

The biggest problem encountered in this research was to find related literature on the subject of simulating hearing loss. Although a recording was made by Oticon (Denmark) to demonstrate the effect of the hearing loss of about five different people, no instrument currently exists that is portable and affordable for the man in street that can simulate the

hearing of any person. The development and construction of an instrument of this nature originated from the very same communication controversy between teachers as was discussed in chapter two. As this is a new idea, no literature exists on the exact topic. Thus only literature on relevant issues such as digital equalization are documented.

All major problems within the scope of the proposal were overcome.
A problem worth mentioning is that it is impossible accurately to simulate the hearing of both ears in a free field. Thus, in a free field only the best ear is represented.

Obviously one is always limited by time and space. The time would not allow an in-depth study of wavelets, which is an interesting field to explore. Also, a thought was given to the idea of developing a modified inverse Chirp Z algorithm to exactly replicate the non-linear characteristics of the ear. The derivation of the inverse chirp Z transformation has been questioned. No literature could be found to substantiate or prove this derivation. Neither was there enough time to practically prove it. These factors prevented this method from being implemented. This is definitely an area for further studies, as this promises to offer an excellent solution to the problem.

A further study can also be conducted in obtaining the ARMA filter's coefficients (by the Modified Yule-walkers equations) within in the same Visual Basic program.

Much of the material studied during this research was excluded from the document. The limited space imposes this restriction. It was stipulated at the beginning of the document that knowledge of the basic theories of DSP is assumed, and these are therefore not covered in the document. Also that material studied which falls beyond the scope of the research is left out and this could also be considered for further studies.

It was mentioned that the development done thus far could form part of a complete acoustic analyzer if other functions such as insertion gain, hearing aid analysis (spectrum analysis and distortion measurement), etc were included. Research on DSP applications in the field of acoustics has endless further possibilities for study.

CHAPTER 13

CONCLUSIONS

13.1 Conclusions

The effective teaching of children with hearing loss was the main problem that motivated this research. The aim of the study was to provide an aid for teachers which would help them in their teaching. Teachers have no idea of how the hearing-impaired child experiences sound. An instrument called a Hearing Loss Simulator, which would expose a teacher to the actual sound of the hearing impaired child, would solve this problem.

The objective of the study was to conduct research in the design, development and construction a Hearing Loss Simulator that would be portable, reliable, fast and accurate. A key objective was to make the instrument affordable and accessible for parents and schools for the hearing impaired or anybody interested in entering the 'world of the deaf'.

The utilization of DSP technology has made the realization of these objectives possible.

The research methodology was as follows. The research is basically broken up into four phases and the entire document is presented in the same logical order. The first phase is the 'Conceptual idea of the problem and how to solve it'. This entails the awareness of the problem and brainstorming techniques for solving the problem. The ideas generated in this phase have provided a basic framework for possible theoretical solutions. The study of the possible theoretical solutions naturally formed the second phase. The areas which were investigated during the second phase were Filter method using a bank of filters, Inverse Fourier transforms, Inverse Chirp Z transforms, Wavelet approach / sub band decomposition and ARMA filter design using the Modified Yule-walker equations. A comparison between the various methods was done in chapter 9. The optimum method for the project given the constraints, time, cost, analysis speed etc, was then identified by the evaluation of the tradeoffs.

The investigation undertaken in the study has identified the use of the IFFT (Inverse fast fourier transform) method as the optimum method for implementation. It was found that the best way to implement the IFFT method is to perform the IFFT on a PC and carry out the filtering on a specialized digital signal processor. This arrangement is inspired by the fact that the IFFT calculation need not be performed in real time when the convolution process for FIR filtering is done in real time.

The practical implementation, which is the third phase of the method of research, was made very much easier with this arrangement. A conclusive summary that overviews the practical implementation of the instrument is given briefly. The HLS uses an existing audiometer to determine the audiogram, as explained in chapter two. The points on the audiogram are interpolated to give the exact frequency response of the filter that needs to be to designed. The use of the IFFT method is, in fact, the design procedure for a FIR filter. By executing the IFFT on the frequency response, the filter coefficients are derived. The filter coefficients are thus the time domain representation of the given frequency response. These filter coefficients are sent to a DSP evaluation board, called the EZ_kit Lite, which uses it in the convolution process. By convoluting the sampled input signal with the filter coefficients, the actual filtering is carried out. The input signal is any sound track (audio signal) such as a recording of a lesson presented by the teacher or even music from a CD or tape recorder. The output of the filter is still in sampled form and is reconstructed by the CODEC to give a continuous signal. The output is sent via an additional amplifier to a pair of headphones through which the teacher listens to the attenuated sound reproduction. The supposition is that the attenuated sound reproduction is the simulation of a hearing loss.

The fourth phase of the research method comprises the evaluation process and is discussed in chapter 12. The quality of the instrument is confirmed by looking into the following evaluation criteria:

Sound pressure level (SPL) and amplitude, frequency resolution, computational speed limits on the number of taps, amplitude values other than the test frequencies, phase distortion, stability, speed, quantization error and signal to noise ratio, users interface, correlation between theory and practice.

The evaluations, which are the fourth phase of the research methodology, conclude the research in terms of the research method.

The document is concluded with chapter 12.2, which covers the problems encountered and further areas of study.

13.2 Recommendations

Apart from the recommended areas for further study in section 12.12, another recommendation would be to integrate as much function as possible into the system. Many functions relating to audiology and acoustics can still be built into the system without adding to hardware costs, as explained in chapter one. Functions such as Hearing aid analyses which constitute distortion measurements, gain measurements, spectrum analyses etc. also insertion gain, which is part of audiology, as explained in chapter one and two, are but a few of the functions that could be recommended for future research and development. When the functionality of the instrument is to be increased, only the appropriate algorithms for implementation that will result in the optimum performance need to be researched.

The biggest recommendation would be to conduct a study in the use of the instrument. In the first and second chapters the development of the hearing loss simulator was motivated. The usefulness of the instrument was illustrated in general but an intense study on implementation strategies in teaching needs to be conducted so as to realize the full potential of this instrument. A research study that investigates this area would obviously be in the interest of other disciplines besides engineering.

REFERENCES

ANDERSEN, N.O. 1974: The calculation of filter coefficients for maximum entropy spectral analysis. Geophysics, Vol. 39, p 69 - 72

BALLANTYNE, J. 1977: Deafness. Edinburgh: Churchill Livingstone.

CACCAMISE, F. 1978: Manual/Simultaneous Communication (M/SC). Instruction at the National Institute for the Deaf (NTID): An Introduction. American Annals for the Deaf, Vol. 123, no. 7, p. 801.

COETZER, M.W. 1994: Power spectral density estimates. Unpublished lecturer's notes: University of Stellenbosch.

DALE, D.M.C. 1962: Applied Audiology for Children. U.S.A.: Charles C Thomas Publisher.

DAUBECHIES, I. 1992: Ten Lectures on Wavelets. Philadelphia, Pennsylvania: Society of Industrial and Applied Mathematics.

DAVIS, H. & SILVERMAN, S. R. 1978: Hearing and deafness (4th ed). New York: Holt, Rhinehart and Winston.

DEPARTMENT OF NATIONAL HEALTH AND POPULATION DEVELOPMENT 1981: Identifisering, klassifisering en plasing van gehoorgestremdes. An unpublished report from a subcommittee for hearing impairment, Pretoria.

FRIEDLANDER, B. & PORAT, B. 1983: The Modified Yule-Walker method of ARMA spectral estimation. IEEE Transactions on aerospace and electronic systems, Vol. AES-20, No. 2, p 158 –173.

GUREWICH, N. & GUREWICH, O. 1993: Teach Yourself Visual Basic 3.0 in 21 Days. Indiana: SAMS Publishing

HIGGINS, R.J. 1990: Digital Signal Processing in VLSI. Englewood Cliffs, NJ: Prentice Hall.

HOROWITZ, P. & HILL, W. 1995: The art of Electronics. Cambridge, UK: Cambridge University Press.

KINSLER, L.E.; FREY, A. R.; COPPENS, A.B. & SANDERS, J.V. 1982: Fundamentals of Acoustics. New York: John Wiley & Sons.

KRAUSS, T.P.; SHURE, L. & LITTLE, J.N. 1994: Signal Processing TOOLBOX For Use with MATLAB, Users Guide. Natick, Mass: The Mathworks, Inc.

KUC, R. 1988: Introduction to Digital Signal Processing. New York: McGraw Hill.

KUN-SHAN. 1986: Algorithms and Implementations. Austin, Texas: Texas Instruments.

LIGTENBERG, C.L. 1982: Basics of audiology. Unpublished lecturer's notes: University of Stellenbosch.

LOCKHART, G.B. & CHEETHAM, B. M. G. 1989: BASIC digital signal processing. Norwich, Norfolk: Page Bros.

LUDEMAN, L.C. 1986: Fundamentals of Digital Signal Processing. New York: Harper and Row.

MANNING, D. 1987: Parents and Mainstreaming. Volta Review, vol 89, no 5, p. 119.

MARTIN, J. D. 1991: *Signal and Processes:* A Foundation Course. London, UK: Pitman Publishers.

MEYER, Y. 1993: Wavelets Algorithms and Applications. Philadelphia: Society of Industrial and Applied Mathematics.

MISITI, M.; MISITI, Y.; OPPENHEIM, G. & POGGI, J. 1996: Wavelet TOOLBOX For Use with MATLAB. Natick, Mass: The Mathworks, Inc.

MOORES, D.F. 1987: Educating the deaf: Psychology, Principles and Practices (3rd ed). Boston, MA: Houghton Mifflin Company.

MOORES, D. F., WEISE, K.L. & GOODWIN, M.W. 1978: Early education programs for hearing-impaired children: major findings. American Annals for the Deaf, vol. 123, no. 8.

MULLER, M. 1990: Application of the programmable filter, the INMOS R100, to Radar Signal Processing. Unpublished Masters thesis: University of Stellenbosch.

NEWBY, H.A. 1958: Audiology. New York: Appleton-Century-Crofts, Inc.

OBLOWITZ, N.G. 1988: The Development of the Self Concept for the Hearing Impaired. Unpublished Master's Thesis: University of Cape Town.

PUREN, H. B. 1985: 'n Totale Kommunikasiebenadering in die onderrig van die Dowe kind met spesifieke verwysing na Taalkommunikasie en Taalverwerwing. Unpublished Master's Thesis, University of Stellenbosch.

SCHLOSS, P. J. & GOLDSMITH, L. 1986: Diagnostic Overshadowing Among Phychologist Working with Hearing Impaired Learner. American Annals for the Deaf, 131(4).

STRANG, G. & NGUYEN, T. 1996: Wavelets and Filterbanks. Wellesly, MA: Wellesly-Cambridge Press.

SUMMER, D. (Editorial director) 1987: Longman Dictionary of Contemporary English. Great Britian: Richard Clay Ltd.

THE APPLICATION ENGINEERING STAFF OF ANALOG DEVICES, DSP Division, Edited by MAR, A. 1992: Digital Signal Processing Applications using the ADSP-2100 Family, Volume 1. Englewood Cliffs, NJ: Prentice Hall.

THE EDITORIAL STAFF OF ANALOG DEVICES. 1995: ADSP-2100 Family EZ-KIT Lite Reference Manual. Norwood: Analog Devices, Inc.

THE EDITORIAL STAFF OF ANALOG DEVICES. 1995: ADSP-2100 Family User's Manual (Includes ADSP-2171, ADSP-2181). (3rd ed). Norwood: Analog Devices, Inc.

THYS, N.W. 1996: AUDIOMETER as part of an acoustic analyzer. Unpublished B.Tech project design documentation: Peninsula Technikon.

TUNER, J.D. & PRETLOVE, A.J. 1991: Acoustics for Engineers. Hong Kong: Macmillan Education Ltd.

VAN CLEEVE, J.V. 1987: Gallaudet Encyclopedia of Deaf People and Deafness. New York: McGraw-Hill Book Company, Inc.

VETTERLI, M. & KOVAČEVIĆ, J. 1995: Wavelets and Subband Coding. Englewood Cliffs, NJ: Prentice Hall.

INTERVIEWS

Briers, J. A. M: ENT Specialist. Worcester.

Thys, J. 1999: A teacher at Nuwe Hoop Centre for the hearing impaired. Worcester.

Theron, L. 1999: Senior lecturer in Audiology at the University of Stellenbosch.

Minnie, A. 1999: Head of psychology: Nuwe Hoop Center for the hearing impaired. Worcester.

Dalvi, R. 2000: Audiologist and speech therapist.

Appendix A DSP Program in ASM2181

```
.module/RAM/ABS=0;
.include <..\system.k>;
/*********************************************************************/
{ Sine Generator variables }
.var/dm     angle;                          /* pointer in the sine table */
.var/dm     delta;                          /* step size in the table */
.var/pm/circ  sintable[256];          /* sine table of 256 points */
/*********************************************************************/
.const              cnt=0x7d00;   /*One sec duration of pulse*/
.const              taps=256;
.var/dm/ram/circ        rx_buf[3];    /* Status + L data + R data */
.var/dm/ram/circ        tx_buf[3];    /* Cmd + L data + R data   */
.var/dm/ram/circ        init_cmds[13];
.var/dm             stat_flag;
.var/dm             testnum;
.var/dm             freq;
.var/dm             amplitude;
.var/dm             AmpTab1[21];
.var/dm             AmpTab2[21];
.var/dm             AmpTab3[21];
.var/dm             AmpTab4[21];
.var/dm             AmpTab5[21];
.var/dm             AmpTab6[21];
.var/dm             AmpTab7[21];
.var/dm             FreqTab[7];
.var/pm/ram/circ        coeftab[256];          /* Filter coefficient table */
.var/dm             dlflag;                    /* Down filter coef. Flag */
.var/dm             cntra;                                  /* count 2 bytes per word */
.var/dm             fcntr;                                 /* count 256 coefficients */
.var/dm             tempstr1;  /* temporally store higher 8 bits of 16 bit word */
.var/dm             rxval;
.var/dm             sflag;                          /* sound flag */
.var/dm             fflag;                      /* flag filtering or sine wave gen. */
.var/dm             count;
.var/dm             lef;
.var/dm             rig;
.var/dm             startemp;
.var/dm             outval;
.var/dm             sjof;
.var/dm             zjof;
.var/dm             accincr;
.var/dm             sincr;
.var/dm             mdcr;
.var/dm             stri2;
.var/dm/circ        data[taps];          * table to hold input signal samples *
.init24 sintable: <sine.hex>;
.init AmpTab1: <ampval1.dsp>;
.init AmpTab2: <ampval2.dsp>;
.init AmpTab3: <ampval3.dsp>;
.init AmpTab4: <ampval4.dsp>;
.init AmpTab5: <ampval5.dsp>;
.init AmpTab6: <ampval6.dsp>;
.init AmpTab7: <ampval7.dsp>;
.init FreqTab: <freqval.dsp>;
.init testnum: 0x7fff;
.init tx_buf:  0xc000, 0x0000, 0x0000; * Initially set MCE      *
.init init_cmds:
        0xc002,   {       Left input control reg
                          b7-6: 0=left line 1
                              1=left aux 1
                              2=left line 2
                              3=left line 1 post-mixed loopback
                          b5-4: res
                          b3-0: left input gain x 1.5 dB   }
        0xc102,   {       Right input control reg
                          b7-6: 0=right line 1
                              1=right aux 1
                              2=right line 2
                              3=right line 1 post-mixed loopback
                          b5-4: res
                          b3-0: right input gain x 1.5 dB   }
        0xc288,   {       left aux 1 control reg
                          b7  : 1=left aux 1 mute
                          b6-5: res
```

```
                        b4-0: gain/atten x 1.5, 08= 0dB, 00= 12dB   }
        0xc388,   {     right aux 1 control reg
                        b7 : 1=right aux 1 mute
                        b6-5: res
                        b4-0: gain/atten x 1.5, 08= 0dB, 00= 12dB   }
        0xc488,   {     left aux 2 control reg
                        b7 : 1=left aux 2 mute
                        b6-5: res
                        b4-0: gain/atten x 1.5, 08= 0dB, 00= 12dB   }
        0xc588,   {     right aux 2 control reg
                        b7 : 1=right aux 2 mute
                        b6-5: res
                        b4-0: gain/atten x 1.5, 08= 0dB, 00= 12dB   }
        0xc680,   {     left DAC control reg
                        b7 : 1=left DAC mute
                        b6 : res
                        b5-0: attenuation x 1.5 dB   }
        0xc780,   {     right DAC control reg
                        b7 : 1=right DAC mute
                        b6 : res
                        b5-0: attenuation x 1.5 dB   }
        0xc856,   {     data format register
                        b7 : res
                        b5-6: 0=8-bit unsigned linear PCM
                            1=8-bit u-law companded
                            2=16-bit signed linear PCM
                            3=8-bit A-law companded
                        b4 : 0=mono, 1=stereo
                        b0-3: 0=  8.
                            1=  5.5125, 2= 16., 3= 11.025,  4= 27.42857,  5= 18.9,  6= 32.,
                            7= 22.05,  8=  0.0 ,  9= 37.8
                            a=  0.0,  b= 44.1,  c= 48.0,  d= 33.075,  e=  9.6,  f=  6.615
                    (b0) : 0=XTAL1 24.576 MHz; 1=XTAL2 16.9344 MHz   }
        0xc909,   {     interface configuration reg
                        b7-4: res
                        b3 : 1=autocalibrate
                        b2-1: res
                        b0 : 1=playback enabled   }
        0xca00,   {     pin control reg
                        b7 : logic state of pin XCTL1
                        b6 : logic state of pin XCTL0
                        b5 : master - 1=tri-state CLKOUT
                            slave  - x=tri-state CLKOUT
                        b4-0: res      }
        0xcc40,   {     miscellaneous information reg
                        b7 : 1=16 slots per frame, 0=32 slots per frame
                        b6 : 1=2-wire system, 0=1-wire system
                        b5-0: res      }
        0xcd00;   {     digital mix control reg
                        b7-2: attenuation x 1.5 dB
                        b1 : res
                        b0 : 1=digital mix enabled   }

{*********************************************************************
 * Interrupt vector table
 *********************************************************************}
D       jump start; rti; rti; rti;    {00: reset }
            rti;    rti; rti; rti;    {04: IRQ2 }
            rti;    rti; rti; rti;    {08: IRQL1 }
            rti;    rti; rti; rti;    {0c: IRQL0 }
            ar = dm(stat_flag);       {10: SPORT0 tx }
            ar = pass ar;
            if eq rti;
            jump next_cmd;
            jump tx_samples;          {14: SPORT0 rx }
            rti; rti; rti;
            rti;    rti; rti; rti;    {18: IRQE }
            rti;    rti; rti; rti;    {1c: BDMA }
            jump irq1isr;   rti; rti; rti;    {20: SPORT1 tx or IRQ1 }
            rti;    rti; rti; rti;    {24: SPORT1 rx or IRQ0 }
            rti;    rti; rti; rti;    {28: timer }
            rti;    rti; rti; rti;    {2c: power down }
E   start:  reset fl1;
            i0 = ^rx_buf;
            l0 = %rx_buf;
            i1 = ^tx_buf;
            l1 = %tx_buf;
```

```
                i3 = ^init_cmds;
                l3 = %init_cmds;
                m1 = 1;
.include <i_codcom.dsp>;
        i6 = ^coeftab;
        l6 = 0;m6=1;

    {*** Main Code Loop ***}
        ax0=cnt;
        dm(count)=ax0;
        ax0=0;
        dm(sflag)=ax0;
        dm(fflag)=ax0;
        dm(fcntr)=ax0;          /* count 256 filter coefficients */
        dm(dlflag)=ax0;         /* initialise download flag to 0 */
        ax0 = 2;
        dm(cntra) = ax0;        /* 16 bit Filter coef.= 2 bytes */
        ax0 = 0;
        dm(startemp) = ax0;
          ax0 = 0x6400;
          dm(delta) = ax0;
          ax0 = 0;
          dm(angle) = ax0;
          reset fl1;
          ax0 = 1000;
          dm(freq) = ax0;
          call set_delta;
    {-----------------------------------------------------------------------
    - command loop.
    -----------------------------------------------------------------------}
H   main:  { any thing from host ?}
                ar = dm (CHAR_WAITING_FLAG);        { kbhit () }
                none = pass ar;
                if ne jump main;
                { has something }
                i4 = dm (PTR_TO_GET_CHAR);      { get int }
                call (i4);
                if lt jump main; { time out }
    { *** Interface Code : input value in ax1 **** }
            {* Clear delay line *}
            ax0=0;
            dm(fflag)=ax0;
            ar = dm(dlflag);
            none = pass ar;
            if ne jump bran;
            ay0 = 0x001b;
            ar =ax1 - ay0;
            if eq jump setffl;
            ay1 = 0x1b;
            ar = ax1 - ay1;
            if eq jump main;
            ay1 = 0x19;
            ar = ax1 - ay1;
            if eq jump lft;
            ay1 = 0x1a;
            ar = ax1 - ay1;
            if eq jump rht;
           ay0 = 0;
           dm(angle) = ay0;
           ay0 = cnt;
           dm(count) = ay0;
           ay0 = 0x7fb7;
           dm(mdcr) = ay0;              * measure decrements in ramp against 1 *
           ay0 = 0;
           dm(accincr) = ay0;           * accumulative increments *
           ay0 =0x0020;
           dm(sincr) = ay0;             * accumulative increments *
           ay0 = 1;
           dm(sjof) = ay0;              * set jump over flag to one *
           ay0 = 1;
           dm(zjof) = ay0;              * set z jump over flag to one (falltime at end)*
I          ay0 = 1;
           dm(sflag) = ay0;             * Set sound flag *
           dm(rxval)= ax1;
           call lookup;
          jump main;
```

```
setffl:  ax0 = 1;
         dm(dlflag) = ax0;
         jump main;
bran:    ax0 = dm(cntra);
         ar = ax0 - 1;
         dm(cntra) = ar;
         if eq jump instr0;
         ar = ax1;                    /* ax1 still holds the input value */
         sr = lshift ar by 8 (hi);    /* shift ar up by 8 bits */
         dm(tempstr1) = sr1;
         jump main;
instr0:  ay1 = dm(tempstr1);
         ar = ax1 or ay1;             /* hi+lo byte = 16 bit word */
         pm(i6,m6) = ar;
         ax0 = 2;
         dm(cntra) = ax0;
         ay0 = dm(fcntr);             /* count up */
         ax0 = 255;                   /* to 256   */
         ar = ax0 - ay0;
         if le call stfilf;
         ar = ay0 + 1;
         dm(fcntr) = ar;
         jump main;
stfilf:  i2=^data;  m1=1;  l2=taps;
         i6 = ^coeftab;
         l6 = taps;
         m6 = 1;
         ax0 = 1;
         dm(fflag) = ax0;             /* set fflag to start filtering */
         ax0 = 0;
         dm(dlflag) = ax0;
         ax0 =0;
         dm(fcntr) = ax0;
         jump main;
iSend:   ay0 = 0x41;
         ar = ar + ay0;
         ax1 = 0x58;                  /* send X */
         i4 = dm (PTR_TO_OUT_CHAR);   { send int }
         call (i4);
         { wait for char to go out }
wt:      ar = dm (CHAR_SEND_DONE_FLAG);
         none = pass ar;
         if eq jump wt;
      jump main;
lft:     ax0=0;
         dm(lef)=ax0;
         ax0=1;
         dm(rig)=ax0;
         jump main;
rht:     ax0=1;
         dm(lef)=ax0;
         ax0=0;
         dm(rig)=ax0;
         jump main;

/****************** Set_Delta and Lookup routines ************************/
K  set_delta:
         ena m_mode;
         mx0 = dm(freq);
         my0 = 2;                     /*2.048 FOR Fs=32000*/
         mr = mx0 * my0 (uu);
         ax0 = mr0;
         dis m_mode;
         my0 = 1573;                  /* 0.048 (fraction*32768)*/
         mr = mx0 * my0 (rnd);
         ay0 = mr1;
         dis ar_sat;
         ar = ax0 + ay0;
         dm(delta) = ar;
         ena ar_sat;
         rts;
J  lookup:   /* input in ar */
         ar= dm(rxval);
         sr = lshift ar by -5 (hi);   * shift ar down by 5 bits */
         i2 = ^FreqTab;
         m2 = sr1;
         l2 = 0;
```

```
            modify(i2,m2);          /* point to table offset */
            ax0 = dm(i2,m2);        /* extract freq value */
            dm(freq)=ax0;
            call set_delta;
        /* now find amplitude */
        ar = dm(rxval);
        ay0 = b#00011111;
        ar = ar and ay0;            /* strip upper 3 bits */
        m2 = ar;
        ax0 = sr1;
        ay0 = 1;
        ar = ax0 - ay0;
        if eq jump at1;
        ay0 = 2;
        ar = ax0 - ay0;
        if eq jump at2;
        ay0 = 3;
        ar = ax0 - ay0;
        if eq jump at3;
        ay0 = 4;
        ar = ax0 - ay0;
        if eq jump at4;
        ay0 = 5;
        ar = ax0 - ay0;
        if eq jump at5;
        ay0 = 6;
        ar = ax0 - ay0;
        if eq jump at6;
        ay0 = 0;
        ar = ax0 - ay0;
        if eq jump at7;
    cnth:  modify(i2,m2);           /* point to table offset*/
           ax0 = dm(i2,m2);
           dm(amplitude) = ax0;
           rts;
    at1:   i2 = ^AmpTab1;
           jump cnth;
    at2:   i2 = ^AmpTab2;
           jump cnth;
    at3:   i2 = ^AmpTab3;
           jump cnth;
    at4:   i2 = ^AmpTab4;
           jump cnth;
    at5:   i2 = ^AmpTab5;
           jump cnth;
    at6:   i2 = ^AmpTab6;
           jump cnth;
    at7:   i2 = ^AmpTab7;
           jump cnth;

    {********* End Main **********}
    {***** Interrupt Routines ********}
    {* Interrupt use for sending  tx samples *}
L   tx_samples:  ena    sec_reg;            /* use secondary bank */
            ar = dm(fflag);             /*  choose between  */
            ar = pass ar;               /*  filtering and   */
            if ne jump startfilter;     /*  sine generation */
                ar =   dm(angle);           /* save cum angle unsigned */
                sr =   lshift ar by -8 (hi);    /* get integer to lookup */
                i4 =   ^sintable;
                m4 =   sr1;
               dm(startemp)=ar;
                l4 =  0;
                modify (i4,m4);             /* point to offset from origin */
                ax1 = pm(i4,m4);
                ax0 = dm(sflag);
           none = pass ax0;
           if eq jump clrf;             /* If sflag is 0 shut down signal*/
           mx0 = dm(amplitude);
           my0 = ax1;
           mr = mx0*my0(rnd);
           ax1 = mr1;
           dm(outval) =ax1;             /* store output val */
           ax0 = dm(sjof);
           none = pass ax0;
           if eq jump jpo;              /* end of ramp if sjof =0 */
          {***** start of ramp *****}
          ax0 = dm(accincr);
```

```
        ay1 = dm(sincr);
        ar = ax0 + ay1;
        dm(accincr) = ar;                /* accumulate increments */
        ax0 = dm(mdcr);
        ay1 = dm(accincr);
        ar = ax0 - ay1;                  /* subtract from one */
        if le call setjof;               /* test for end of ramp */
        mx0 = dm(outval);
        my0 = dm(accincr);
        mr = mx0*my0(rnd);
        ax1 = mr1;
        dm(outval) = ax1;
jpo:    ax1 = dm(outval);                /* output value without ramp*/
        jump out;
clrf:   mx0 = dm(amplitude);
        my0 = ax1;                       /*ax1 comes from pm(i4,m4) */
        mr = mx0*my0(rnd);
        ax1 = mr1;
        dm(outval) = ax1;
        ax0 = dm(zjof);
        none = pass ax0;
        if eq jump clout;
        ax0 = dm(mdcr);
        ay1 = dm(sincr);
        ar = ax0 - ay1;
        if le call clszjof;
        dm(mdcr) = ar;                   /* accumulate increments */
        my0 = dm(mdcr);
        mx0 = dm(outval);
        mr = mx0*my0(rnd);
        ax1 = mr1;
        jump out;
clout:  ax1=0;
        jump out;
setjof: ay0 = 0;
        dm(sjof) = ay0;
        rts;
clszjof: ay0 = 0;
        dm(zjof) = ay0;
        rts;
out:    ay0=dm(lef);
        none=pass ay0;
        if eq call le1;
        if ne call le2;
        ay0=dm(rig);
        none=pass ay0;
        if eq call rt1;
        if ne call rt2;
        ar = dm(startemp);
M       ay1 = dm(delta);                 /* get delta */
        dis ar_sat;                      /* disable saturation */
        ar = ar - ay1;                   /* increment the angle*/
        dm(angle) = ar;                  /* save the cum. angle */
N       ay0 = 1;
        ar = dm(count);
        ar = ar - ay0;
        if eq jump stflag;
        dm(count) = ar;
        nop;nop; rti;
stflag: ax0 = 0;
        dm(sflag) =ax0;
        rti;
O   le1:    dm(tx_buf-1) = ax1;
            rts;
    le2:    ax0=0;
            dm(tx_buf-1) =ax0;
            rts;
    rt1:    dm(tx_buf-2) = ax1;
            rts;
    rt2:    ax0=0;
            dm(tx_buf-2) =ax0;
            rts;
PA  startfilter:    sr1=dm(rx_buf-2);
            sr1=dm(rx_buf-1);
            dm(i2,m1)=sr1;
            cntr=taps-1;
```

```
        mr=0, mx0=dm(i2,m1), my0=pm(i6,m6);
        do fir1loop until ce;
    fir1loop:  mr=mr+mx0*my0(ss), mx0=dm(i2,m1), my0=pm(i6,m6);
        mr=mr+mx0*my0(rnd);
        if mv sat mr;
        sr = ashift mr1 by 1 (lo);
        mr1 = sr0;
        dm (tx_buf + 1) = mr1;
        dm (tx_buf + 2) = mr1;
        rti;
    {* transmit interrupt used for Codec initialization *}
next_cmd:  ena sec_reg;
            ax0 = dm (i3, m1);      { fetch next control word and }
            dm (tx_buf) = ax0;      { place in transmit slot 0   }
            ax0 = i3;
            ay0 = ^init_cmds;
            ar = ax0 - ay0;
            if gt rti;              { rti if more control words still waiting }
            ax0 = 0x8000;           { else set done flag and }
            dm (tx_buf) = ax0;      { remove MCE if done initialization }
            ax0 = 0;
            dm (stat_flag) = ax0;   { reset status flag }
            rti;
    {*********************************************************************
     * A high to low transition on flag_in signifies the start bit; it also
     * triggers IRQ1 ISR which then turn on timer if the timer is off.  This is
     * at to most 1/3 bit period too late but we should still catch the byte.
     *********************************************************************}
    irq1isr:    pop sts;
            ena timer;          { start timer now }
            rts;                { note rts }
    .endmod;



    /* SYSTEM.K */

    .const PTR_TO_GET_CHAR     = 0x3e08;  { function pointer to receive a byte.}
    .const CHAR_WAITING_FLAG   = 0x3e09;  { 0=byte waiting, 1=nothing in     }
    .const PTR_TO_OUT_CHAR     = 0x3e0a;  { function pointer to send a byte.  }
    .const CHAR_SEND_DONE_FLAG = 0x3e0b;  { 0=sending, 1=idling             }
    .const CHAR_RECEIVING_FLAG = 0x3e0c;  { 0=recving, 1=idling             }


    .const L_ADC_IN          = 0x3e01;  { left channel input          }
    .const R_ADC_IN          = 0x3e02;  { right channel input         }
    .const L_DAC_OUT         = 0x3e05;  { left channel output         }
    .const R_DAC_OUT         = 0x3e06;  { right channel output        }


    .const IDMA              = 0x3fe0;
    .const BDMA_BIAD         = 0x3fe1;
    .const BDMA_BEAD         = 0x3fe2;
    .const BDMA_BDMA_Ctrl    = 0x3fe3;
    .const BDMA_BWCOUNT      = 0x3fe4;
    .const PFDATA            = 0x3fe5;
    .const PFTYPE            = 0x3fe6;

    .const SPORT1_Autobuf    = 0x3fef;
    .const SPORT1_RFSDIV     = 0x3ff0;
    .const SPORT1_SCLKDIV    = 0x3ff1;
    .const SPORT1_Control_Reg = 0x3ff2;
    .const SPORT0_Autobuf    = 0x3ff3;
    .const SPORT0_RFSDIV     = 0x3ff4;
    .const SPORT0_SCLKDIV    = 0x3ff5;
    .const SPORT0_Control_Reg = 0x3ff6;
    .const SPORT0_TX_Channels0 = 0x3ff7;
    .const SPORT0_TX_Channels1 = 0x3ff8;
    .const SPORT0_RX_Channels0 = 0x3ff9;
    .const SPORT0_RX_Channels1 = 0x3ffa;
    .const TSCALE            = 0x3ffb;
    .const TCOUNT            = 0x3ffc;
    .const TPERIOD           = 0x3ffd;
    .const DM_Wait_Reg       = 0x3ffe;
    .const System_Control_Reg = 0x3fff;
```

```
/* I_CODCOM.DSP*/
{  shut down sport 0 }
     ax0 = b#0000100000000000;  dm (System_Control_Reg) = ax0;
{  restores monitor timer handler. }
     i7 = 0x3fe8;
     ar = pm (i7, m7);              { px implicit }
     i7 = 0x28;
     pm (i7, m7) = ar;

{=============== S E R I A L   P O R T  #0  S T U F F ===============}
     ax0 = b#0000001010000111;
           dm (SPORT0_Autobuf) = ax0;
     ax0 = 0;   dm (SPORT0_RFSDIV) = ax0;
        {  RFSDIV = SCLK Hz/RFS Hz - 1 }
     ax0 = 0;   dm (SPORT0_SCLKDIV) = ax0;
        {  SCLK = CLKOUT / (2  (SCLKDIV + 1) }
     ax0 = b#1000011000001111;
           dm (SPORT0_Control_Reg) = ax0;
     ax0 = b#0000000000000111;  dm (SPORT0_TX_Channels0) = ax0;
     ax0 = b#0000000000000111;  dm (SPORT0_TX_Channels1) = ax0;
     ax0 = b#0000000000000111;  dm (SPORT0_RX_Channels0) = ax0;
     ax0 = b#0000000000000111;  dm (SPORT0_RX_Channels1) = ax0;

{============ S Y S T E M   A N D   M E M O R Y   S T U F F ============}
           ax0 = 0;
           dm(DM_Wait_Reg) = ax0;                    { no wait states }
{                 5432109876543210}
     ax0 = b#0001100000000000;
           dm (System_Control_Reg) = ax0;
     ifc = b#00000011111111;       { clear pending interrupt }
     nop;
     icntl = b#00010;
     mstat = b#1000000;
     ax0 = 1;
     dm(stat_flag) = ax0;
{      ena ints;}
     imask = b#0001000000;
     ax0 = dm (i1, m1);         { start interrupt }
     tx0 = ax0;
check_init:  ax0 = dm (stat_flag);      { wait for entire init }
     af = pass ax0;             { buffer to be sent to }
     if ne jump check_init;     { the codec          }
     ay0 = 2;
check_aciH:  ax0 = dm (rx_buf);         { once initialized, wait for codec }
     ar = ax0 and ay0;          { to come out of autocalibration }
     if eq jump check_aciH;     { wait for bit set }
check_aciL:  ax0 = dm (rx_buf);         { wait for bit clear }
     ar = ax0 and ay0;
     if ne jump check_aciL;
     idle;
     ay0 = 0xbf3f;              { unmute left DAC }
     ax0 = dm (init_cmds + 6);
     ar = ax0 AND ay0;
     dm (tx_buf) = ar;
     idle;
     ax0 = dm (init_cmds + 7);  { unmute right DAC }
     ar = ax0 AND ay0;
     dm (tx_buf) = ar;
     idle;
           ax0 = 0xc901;                                   { clear autocalibration request}
           dm(tx_buf) = ax0;
           idle;
           ax1 = 0x8000;
           dm(tx_buf) = ax1;
     ifc = b#00000011111111;
     nop;
           imask = b#0000100101;
```

Appendix B Host program in Visual Basic

```
'-------------------------------------------------------------------
'—Program for Hearing Loss Simulation and Audiometer
'-------------------------------------------------------------------
Option Explicit
DefInt A-Z
'-- Student numbers are stored here
Dim StudentNumbers$()
'-- This flag is set when cancel is pressed
Dim CancelFlag
Dim Dummy
Dim InString$
Dim I As Long
Dim values
Dim px, py, px1, py1, pxx0, pxx1, pxx2, pxx3, pxx4, pxx5, pxx6
Dim pyy0, pyy1, pyy2, pyy3, pyy4, pyy5, pyy6, Rpx, Rpy
Dim Rpxx0, Rpxx1, Rpxx2, Rpxx3, Rpxx4, Rpxx5, Rpxx6
Dim Rpyy0, Rpyy1, Rpyy2, Rpyy3, Rpyy4, Rpyy5, Rpyy6
'Declare arrays
Dim Array1(1 To 21) As Integer
Dim Array2(1 To 21) As Integer
Dim Array3(1 To 21) As Integer
Dim Array4(1 To 21) As Integer
Dim Array5(1 To 21) As Integer
Dim Array6(1 To 21) As Integer
Dim Array7(1 To 21) As Integer
Dim xa(0 To 600) As Long
'Declare temporaly store for SPL values
Dim Tstrspl0 As Long
Dim Tstrspl1 As Long
Dim Tstrspl2 As Long
Dim Tstrspl3 As Long
Dim Tstrspl4 As Long
Dim Tstrspl5 As Long
Dim Tstrspl6 As Long
Dim Tstrspl7 As Long
Dim Tstrspl8 As Long

' Declare variables that must be visible in all
' the procedures of the form.
Dim Person As PersonInfo
Dim FileNum As Integer
Dim RecordLen As Long
Dim CurrentRecord As Long
Dim LastRecord As Long
Dim astr As String * 10
Dim num
Dim flnum As Integer
Dim rclen As Long
Dim crec As Long
Dim lrec As Long

Sub cmdaud_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
Dim hval, vval, n, K, v3, v5
Dim hst As String

'Array1 hold values of 125Hz
Array1(1) = 0: Array1(2) = 89: Array1(3) = 309: Array1(4) = 786: Array1(5) = 2725
Array1(6) = 5075: Array1(7) = 9450: Array1(8) = 14943: Array1(9) = 16370 Array1(10) = 18005:
Array1(11) = 19640: Array1(12) = 21275: Array1(13) = 22910
Array1(14) = 24545: Array1(15) = 26180: Array1(16) = 27815: Array1(17) = 29450
Array1(18) = 31085: Array1(19) = 31623: Array1(20) = 32325: Array1(21) = 32767

'Array2 hold values of 250Hz
Array2(1) = 0: Array2(2) = 89: Array2(3) = 309: Array2(4) = 786: Array2(5) = 2725
Array2(6) = 5075: Array2(7) = 9450: Array2(8) = 14943: Array2(9) = 16370
Array2(10) = 18005: Array2(11) = 19640: Array2(12) = 21275: Array2(13) = 22910
Array2(14) = 24545: Array2(15) = 26180: Array2(16) = 27815: Array2(17) = 29450
Array2(18) = 31085: Array2(19) = 31623: Array2(20) = 32325: Array2(21) = 32767

'Array3 hold values of 500Hz
Array3(1) = 0: Array3(2) = 2: Array3(3) = 5: Array3(4) = 14: Array3(5) = 45
Array3(6) = 103: Array3(7) = 255: Array3(8) = 582: Array3(9) = 1036
Array3(10) = 1846: Array3(11) = 3276: Array3(12) = 4455: Array3(13) = 6007
Array3(14) = 7559: Array3(15) = 9111: Array3(16) = 10663: Array3(17) = 12215
Array3(18) = 13767: Array3(19) = 15319: Array3(20) = 16871: Array3(21) = 18426
```

```
'Array4 hold values of 1000Hz
Array4(1) = 0: Array4(2) = 2: Array4(3) = 5: Array4(4) = 14: Array4(5) = 45
Array4(6) = 103: Array4(7) = 255: Array4(8) = 582: Array4(9) = 1036
Array4(10) = 1846: Array4(11) = 3276: Array4(12) = 4455: Array4(13) = 6007
Array4(14) = 7559: Array4(15) = 9111: Array4(16) = 10663: Array4(17) = 12215
Array4(18) = 13767: Array4(19) = 15319: Array4(20) = 16871: Array4(21) = 18426

'Array5 hold values of 2000Hz
Array5(1) = 0: Array5(2) = 10: Array5(3) = 25: Array5(4) = 58: Array5(5) = 143
Array5(6) = 327: Array5(7) = 582: Array5(8) = 1036: Array5(9) = 1842
Array5(10) = 3276: Array5(11) = 5957: Array5(12) = 8638: Array5(13) = 11327
Array5(14) = 14000: Array5(15) = 16681: Array5(16) = 19362: Array5(17) = 22043
Array5(18) = 24724: Array5(19) = 27405: Array5(20) = 30086: Array5(21) = 32767

'Array6 hold values of 4000Hz
Array6(1) = 0: Array6(2) = 10: Array6(3) = 25: Array6(4) = 58: Array6(5) = 143
Array6(6) = 327: Array6(7) = 582: Array6(8) = 1036: Array6(9) = 1842
Array6(10) = 3276: Array6(11) = 5957: Array6(12) = 8638: Array6(13) = 11327
Array6(14) = 14000: Array6(15) = 16681: Array6(16) = 19362: Array6(17) = 22043
Array6(18) = 24724: Array6(19) = 27405: Array6(20) = 30086: Array6(21) = 32767

'Array7 hold values of 8000Hz
Array7(1) = 0: Array7(2) = 89: Array7(3) = 309: Array7(4) = 786: Array7(5) = 2725
Array7(6) = 5075: Array7(7) = 9450: Array7(8) = 14943: Array7(9) = 16370
Array7(10) = 18005: Array7(11) = 19640: Array7(12) = 21275: Array7(13) = 22910
Array7(14) = 24545: Array7(15) = 26180: Array7(16) = 27815: Array7(17) = 29450
Array7(18) = 31085: Array7(19) = 31623: Array7(20) = 32325: Array7(21) = 32767

  values = (hsbfreq.Value * 32) + (20 - vsbamp.Value)
  hval = hsbfreq.Value
  Select Case hval
            Case 0
              vval = Array1(vsbamp.Value + 1)
            Case 1
              vval = Array2(vsbamp.Value + 1)
            Case 2
              vval = Array3(vsbamp.Value + 1)
            Case 3
              vval = Array4(vsbamp.Value + 1)
            Case 4
              vval = Array5(vsbamp.Value + 1)
            Case 5
              vval = Array6(vsbamp.Value + 1)
            Case 6
              vval = Array7(vsbamp.Value + 1)
 End Select
  Comm2.PortOpen = True
  Comm2.Output = Chr(28)
     For n = 1 To 100
     For K = 1 To 35
     Next K
     Next n
    Comm2.Output = Chr(2 ^ (hval))
     For n = 1 To 100
     For K = 1 To 35
     Next K
     Next n
  v3 = Int(vval / 256)
  v5 = vval - (v3 * 256)
 Comm2.Output = Chr(v3)
 For n = 1 To 100
 For K = 1 To 35
 Next K
 Next n
 Comm2.Output = Chr(v5)
 Comm2.PortOpen = False

End Sub

Sub cmdclear_Click ()
  pxx0 = 0:  pxx1 = 0:  pxx2 = 0:  pxx3 = 0:  pxx4 = 0:  pxx5 = 0:  pxx6 = 0
  pyy0 = 0:  pyy1 = 0:  pyy2 = 0:  pyy3 = 0:  pyy4 = 0:  pyy5 = 0:  pyy6 = 0
  Rpxx0 = 0:  Rpxx1 = 0:  Rpxx2 = 0:  Rpxx3 = 0:  Rpxx4 = 0:  Rpxx5 = 0:  Rpxx6 = 0
  Rpyy0 = 0:  Rpyy1 = 0:  Rpyy2 = 0:  Rpyy3 = 0:  Rpyy4 = 0:  Rpyy5 = 0:  Rpyy6 = 0

txtinfo.Text = " Test Incomlete"
```

```
Cls
End Sub

Sub cmddraw_Click ()
If pxx0 = 0 Or pxx1 = 0 Or pxx2 = 0 Or pxx3 = 0 Or pxx4 = 0 Or pxx5 = 0 Or pxx6 = 0 Or
Rpxx0 = 0 Or Rpxx1 = 0 Or Rpxx2 = 0 Or Rpxx3 = 0 Or Rpxx4 = 0 Or Rpxx5 = 0 Or Rpxx6 = 0 Then
 txtinfo.Text = " Test Incomlete"
 Else
Line (pxx0, pyy0)-(pxx1, pyy1)
Line (pxx1, pyy1)-(pxx2, pyy2)
Line (pxx2, pyy2)-(pxx3, pyy3)
Line (pxx3, pyy3)-(pxx4, pyy4)
Line (pxx4, pyy4)-(pxx5, pyy5)
Line (pxx5, pyy5)-(pxx6, pyy6)
Line (Rpxx0, Rpyy0)-(Rpxx1, Rpyy1), QBColor(4)
Line (Rpxx1, Rpyy1)-(Rpxx2, Rpyy2), QBColor(4)
Line (Rpxx2, Rpyy2)-(Rpxx3, Rpyy3), QBColor(4)
Line (Rpxx3, Rpyy3)-(Rpxx4, Rpyy4), QBColor(4)
Line (Rpxx4, Rpyy4)-(Rpxx5, Rpyy5), QBColor(4)
Line (Rpxx5, Rpyy5)-(Rpxx6, Rpyy6), QBColor(4)
txtinfo.Text = " Test Comleted"
  End If
End Sub

Sub cmdexit_Click ()
End
End Sub

Sub cmdlog_Click ()
If hsbfreq.Value = 0 And optleft.Value = True Then
    Circle (pxx0, pyy0), 50, &HFFFFFF
    pxx0 = ((hsbfreq.Value * 600) + 2760)
    pyy0 = ((vsbamp.Value * 120) + 1440)
ElseIf hsbfreq.Value = 1 And optleft.Value = True Then
    Circle (pxx1, pyy1), 50, &HFFFFFF
    pxx1 = ((hsbfreq.Value * 600) + 2760)
    pyy1 = ((vsbamp.Value * 120) + 1440)
ElseIf hsbfreq.Value = 2 And optleft.Value = True Then
    Circle (pxx2, pyy2), 50, &HFFFF&
    pxx2 = ((hsbfreq.Value * 600) + 2760)
    pyy2 = ((vsbamp.Value * 120) + 1440)
ElseIf hsbfreq.Value = 3 And optleft.Value = True Then
    Circle (pxx3, pyy3), 50, &HFFFF&
    pxx3 = ((hsbfreq.Value * 600) + 2760)
    pyy3 = ((vsbamp.Value * 120) + 1440)
ElseIf hsbfreq.Value = 4 And optleft.Value = True Then
    Circle (pxx4, pyy4), 50, &HFFFF&
    pxx4 = ((hsbfreq.Value * 600) + 2760)
    pyy4 = ((vsbamp.Value * 120) + 1440)
ElseIf hsbfreq.Value = 5 And optleft.Value = True Then
    Circle (pxx5, pyy5), 50, &HFFFF&
    pxx5 = ((hsbfreq.Value * 600) + 2760)
    pyy5 = ((vsbamp.Value * 120) + 1440)
ElseIf hsbfreq.Value = 6 And optleft.Value = True Then
    Circle (pxx6, pyy6), 50, &HFFFF&
    pxx6 = ((hsbfreq.Value * 600) + 2760)
    pyy6 = ((vsbamp.Value * 120) + 1440)
 ElseIf hsbfreq.Value = 0 And optright.Value = True Then
    Line ((Rpxx0 - 50), (Rpyy0 - 50))-((Rpxx0 + 60), (Rpyy0 - 60)), &HFFFF&
    Line ((Rpxx0 - 50), (Rpyy0 + 50))-((Rpxx0 + 60), (Rpyy0 - 60)), &HFFFF&
    Rpxx0 = ((hsbfreq.Value * 600) + 2760)
    Rpyy0 = ((vsbamp.Value * 120) + 1440)
ElseIf hsbfreq.Value = 1 And optright.Value = True Then
    Line ((Rpxx1 - 50), (Rpyy1 - 50))-((Rpxx1 + 60), (Rpyy1 + 60)), &HFFFF&
    Line ((Rpxx1 - 50), (Rpyy1 + 50))-((Rpxx1 + 60), (Rpyy1 - 60)), &HFFFF&
    Rpxx1 = ((hsbfreq.Value * 600) + 2760)
    Rpyy1 = ((vsbamp.Value * 120) + 1440)
 ElseIf hsbfreq.Value = 2 And optright.Value = True Then
    Line ((Rpxx2 - 50), (Rpyy2 - 50))-((Rpxx2 + 60), (Rpyy2 + 60)), &HFFFF&
    Line ((Rpxx2 - 50), (Rpyy2 + 50))-((Rpxx2 + 60), (Rpyy2 - 60)), &HFFFF&
  Rpxx2 = ((hsbfreq.Value * 600) + 2760)
  Rpyy2 = ((vsbamp.Value * 120) + 1440)
ElseIf hsbfreq.Value = 3 And optright.Value = True Then
    Line ((Rpxx3 - 50), (Rpyy3 - 50))-((Rpxx3 + 60), (Rpyy3 + 60)), &HFFFF&
    Line ((Rpxx3 - 50), (Rpyy3 + 50))-((Rpxx3 + 60), (Rpyy3 - 60)), &HFFFF&
  Rpxx3 = ((hsbfreq.Value * 600) + 2760)
  Rpyy3 = ((vsbamp.Value * 120) + 1440)
```

```
        ElseIf hsbfreq.Value = 4 And optright.Value = True Then
            Line ((Rpxx4 - 50), (Rpyy4 - 50))-((Rpxx4 + 60), (Rpyy4 + 60)), &HFFFF&
            Line ((Rpxx4 - 50), (Rpyy4 + 50))-((Rpxx4 + 60), (Rpyy4 - 60)), &HFFFF&
          Rpxx4 = ((hsbfreq.Value * 600) + 2760)
          Rpyy4 = ((vsbamp.Value * 120) + 1440)
        ElseIf hsbfreq.Value = 5 And optright.Value = True Then
            Line ((Rpxx5 - 50), (Rpyy5 - 50))-((Rpxx5 + 60), (Rpyy5 + 60)), &HFFFF&
            Line ((Rpxx5 - 50), (Rpyy5 + 50))-((Rpxx5 + 60), (Rpyy5 - 60)), &HFFFF&
          Rpxx5 = ((hsbfreq.Value * 600) + 2760)
          Rpyy5 = ((vsbamp.Value * 120) + 1440)
        ElseIf hsbfreq.Value = 6 And optright.Value = True Then
            Line ((Rpxx6 - 50), (Rpyy6 - 50))-((Rpxx6 + 60), (Rpyy6 + 60)), &HFFFF&
            Line ((Rpxx6 - 50), (Rpyy6 + 50))-((Rpxx6 + 60), (Rpyy6 - 60)), &HFFFF&
          Rpxx6 = ((hsbfreq.Value * 600) + 2760)
          Rpyy6 = ((vsbamp.Value * 120) + 1440)
        End If
        End Sub

        Sub cmdlog_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
        If optleft.Value = True Then
          px = ((hsbfreq.Value * 600) + 2760)
          py = ((vsbamp.Value * 120) + 1440)
          frmaud.Circle (px, py), 40, RGB(0, 0, 0)
        Else
          Rpx = ((hsbfreq.Value * 600) + 2760)
          Rpy = ((vsbamp.Value * 120) + 1440)
            Line ((Rpx - 50), (Rpy - 50))-((Rpx + 60), (Rpy + 60))
            Line ((Rpx - 50), (Rpy + 50))-((Rpx + 60), (Rpy - 60))
        End If
        End Sub

    Sub cmdnew_Click ()
        Cls
        SaveCurentRecord
        ' Add a new blank record.
        LastRecord = LastRecord + 1
        Person.Name = " "
        Person.Date = " "
        Person.pxx0 = 0: Person.pxx1 = 0:Person.pxx2 = 0:Person.pxx3 = 0:Person.pxx4 = 0:Person.pxx5 = 0:Person.pxx6 = 0
        Person.pyy0 = 0: Person.pyy1 = 0:Person.pyy2 = 0:Person.pyy3 = 0:Person.pyy4 = 0:Person.pyy5 = 0:Person.pyy6 = 0
        Person.Rpxx0 = 0: Person.Rpxx1 = 0: Person.Rpxx2 = 0: Person.Rpxx3 = 0: Person.Rpxx4 = 0: Person.Rpxx5 = 0: Person.Rpxx6 = 0:
        Person.Rpyy0 = 0: Person.Rpyy1 = 0: Person.Rpyy2 = 0: Person.Rpyy3 = 0: Person.Rpyy4 = 0
        Person.Rpyy5 = 0: Person.Rpyy6 = 0

        Put #FileNum, LastRecord, Person
        ' Update currentRecord.
        CurrentRecord = LastRecord
        'Display the record that was just created.
        ShowCurrentRecord
        ' Give the focus to the txtName field.
        txtName.SetFocus
        End Sub

D   Sub cmdnext_Click ()
        If txtinfo = " Test Incomlete" Then
        Exit Sub
        End If
        Cls
        ' If the currentRecord is not the last record,
        ' then save the CurrentRecord and skip the next
        ' record.
        If CurrentRecord = LastRecord Then
            Beep
            MsgBox "End of file encountered!", 48
            ShowCurrentRecord
        Else
            SaveCurentRecord
            CurrentRecord = CurrentRecord + 1
            ShowCurrentRecord
        End If

        ' Give focus to the txtName field.
        txtName.SetFocus
        End Sub

DE  Sub cmdprevious_Click ()
        Cls
```

```
    ' If the currentRecord is not the first Record,
    ' then save the current record and and skip one
    ' record backwards.
    If CurrentRecord = 1 Then
      Beep
      MsgBox "Beginning of the file encountered!", 48
            ShowCurrentRecord
    Else
      SaveCurentRecord
      CurrentRecord = CurrentRecord - 1
      ShowCurrentRecord
    End If
   ' Give the focus to the txtName field.
   txtName.SetFocus
End Sub

Sub cmdprint_Click ()
frmaud.Line (pxx0, pyy0)-(pxx1, pyy1)
frmaud.Line (pxx1, pyy1)-(pxx2, pyy2)
frmaud.Line (pxx2, pyy2)-(pxx3, pyy3)
frmaud.Line (pxx3, pyy3)-(pxx4, pyy4)
frmaud.Line (pxx4, pyy4)-(pxx5, pyy5)
frmaud.Line (pxx5, pyy5)-(pxx6, pyy6)
txtampt.Visible = False: txtfreqn.Visible = False: txtamp.Visible = False: txtfreq.Visible = False: txtinfo.Visible = False
vsbamp.Visible = False: hsbfreq.Visible = False: cmdprint.Visible = False: cmddraw.Visible = False
cmdsave.Visible = False: cmdaud.Visible = False: cmdexit.Visible = False: cmdclear.Visible = False
cmdlog.Visible = False
frmaud.PrintForm
printer.EndDoc
txtampt.Visible = True: txtfreqn.Visible = True: txtamp.Visible = True: txtfreq.Visible = True: txtinfo.Visible = True
vsbamp.Visible = True: hsbfreq.Visible = True: cmdlog.Visible = True: cmdexit.Visible = Tru
cmdclear.Visible = True: cmdprint.Visible = True: cmddraw.Visible = True: cmdsave.Visible = True
cmdaud.Visible = True
End Sub

Sub cmdsave_Click ()
    If pxx0 = 0 Or pxx1 = 0 Or pxx2 = 0 Or pxx3 = 0 Or pxx4 = 0 Or pxx5 = 0 Or pxx6 = 0 Or Rpxx0 = 0 Or Rpxx1 = 0 Or Rpxx2 = 0 Or
    Rpxx3 = 0 Or Rpxx4 = 0 Or Rpxx5 = 0 Or Rpxx6 = 0 Then
     txtinfo.Text = " Test Incomlete"
     Else
    SaveCurentRecord
    txtinfo.Text = " Test Comleted"
      End If
    End Sub

    **** SEARCH STUDENT ****
Sub cmdsearch_Click ()
    Dim NameToSearch As String
    Dim Found As Integer
    Dim RecNum As Long
    Dim TmpPerson As PersonInfo
    'Get the name to search from the uesr.
    NameToSearch = InputBox("Enter name to search:", "Search")
    'If the user did not enter a name, then exit
    'from the procedure.
    If NameToSearch = " " Then
      'Give the focus to txtName field.
      txtName.SetFocus
      'Exit this procedure.
      Exit Sub
    End If

    ' Convert the name to be search to uppercase.
    NameToSearch = UCase(NameToSearch)
    ' Initialize the Found flag to false.
    Found = False
    ' Search the name the user entered.
    For RecNum = 1 To LastRecord
       Get #FileNum, RecNum, TmpPerson
        If NameToSearch = UCase(Trim(TmpPerson.Name)) Then
              Found = True
              Exit For
        End If
    Next
    ' If the name was found then display the record
    ' of the found name.
    If Found = True Then
```

```
    SaveCurentRecord
    CurrentRecord = RecNum
    ShowCurrentRecord
  Else
    MsgBox "Name: " + NameToSearch + " not found!"
  End If
  'Give focus to the txtName field.
  txtName.SetFocus
  End Sub

  Sub cmdsend_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    values = (hsbfreq.Value * 32) + (20 - vsbamp.Value)
    Comm2.PortOpen = True
    Comm2.Output = Chr(23)
    Comm2.PortOpen = False
  End Sub

  Sub Command1_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    values = (hsbfreq.Value * 32) + (20 - vsbamp.Value)
    Comm2.PortOpen = True
    Comm2.Output = Chr(24)
    Comm2.PortOpen = False
  End Sub

Sub Form_Load ()
  px1 = 2760
  py1 = 1440
  txtfreq.Text = hsbfreq.Value
    txtfreq.Text = (2 ^ hsbfreq.Value) * 125
    Comm2.CommPort = 2
    ' 9600 baud, no parity, 8 data, and 1 stop bit.
    Comm2.Settings = "9600,N,8,1"
    ' Tell the control to read entire buffer when Input is used.
    Comm2.InputLen = 0
    ' Open the port.
    Comm2.PortOpen = True
    ' Send the attention command to the modem.
    Comm2.Output = Chr(25)
    ' Close the serial port.
    Comm2.PortOpen = False

  '********** DATABASE SECTION **************
  ' Calculate the length of a record.
  RecordLen = Len(Person)
  ' Get the next available file number.
  FileNum = FreeFile
  ' Open the file for ramdom access. If the file
  ' does not exist then it is created.
  Open "naud2.DAT" For Random As FileNum Len = RecordLen
  ' Update CurrentRecord
  CurrentRecord = 1
  ' Find wath is the last record number of
  ' the file.
  LastRecord = FileLen("naud2.DAT") / RecordLen
  ' If the file was just created
  ' (i.e. LastRecord) then update LastRecord to 1
  If LastRecord = 0 Then
    LastRecord = 1
  End If
  ' Display the current record.
  ShowCurrentRecord
  End Sub

  Sub hsbfreq_Change ()
  txtfreq.Text = (2 ^ hsbfreq.Value) * 125
  End Sub

  Sub hsbfreq_KeyDown (keycode As Integer, Shift As Integer)
  'Print keycode
  If keycode = 38 Or keycode = 40 Then
    vsbamp.SetFocus
  ElseIf keycode = 32 Then
    Dim hval, vval, n, K, v3, v5
  Dim hst As String

  'Array1 hold values of 125Hz
  Array1(1) = 0: Array1(2) = 89: Array1(3) = 309: Array1(4) = 786: Array1(5) = 2725
```

```
Array1(6) = 5075: Array1(7) = 9450: Array1(8) = 14943: Array1(9) = 16370 Array1(10) = 18005: Array1(11) = 19640: Array1(12) =
21275: Array1(13) = 22910
Array1(14) = 24545: Array1(15) = 26180: Array1(16) = 27815: Array1(17) = 29450
Array1(18) = 31085: Array1(19) = 31623: Array1(20) = 32325: Array1(21) = 32767

'Array2 hold values of 250Hz
Array2(1) = 0: Array2(2) = 89: Array2(3) = 309: Array2(4) = 786: Array2(5) = 2725
Array2(6) = 5075: Array2(7) = 9450: Array2(8) = 14943: Array2(9) = 16370
Array2(10) = 18005: Array2(11) = 19640: Array2(12) = 21275: Array2(13) = 22910
Array2(14) = 24545: Array2(15) = 26180: Array2(16) = 27815: Array2(17) = 29450
Array2(18) = 31085: Array2(19) = 31623: Array2(20) = 32325: Array2(21) = 32767

'Array3 hold values of 500Hz
Array3(1) = 0: Array3(2) = 2: Array3(3) = 5: Array3(4) = 14: Array3(5) = 45
Array3(6) = 103: Array3(7) = 255: Array3(8) = 582: Array3(9) = 1036
Array3(10) = 1846: Array3(11) = 3276: Array3(12) = 4455: Array3(13) = 6007
Array3(14) = 7559: Array3(15) = 9111: Array3(16) = 10663: Array3(17) = 12215
Array3(18) = 13767: Array3(19) = 15319: Array3(20) = 16871: Array3(21) = 18426

'Array4 hold values of 1000Hz
Array4(1) = 0: Array4(2) = 2: Array4(3) = 5: Array4(4) = 14: Array4(5) = 45
Array4(6) = 103: Array4(7) = 255: Array4(8) = 582: Array4(9) = 1036
Array4(10) = 1846: Array4(11) = 3276: Array4(12) = 4455: Array4(13) = 6007
Array4(14) = 7559: Array4(15) = 9111: Array4(16) = 10663: Array4(17) = 12215
Array4(18) = 13767: Array4(19) = 15319: Array4(20) = 16871: Array4(21) = 18426


'Array5 hold values of 2000Hz
Array5(1) = 0: Array5(2) = 10: Array5(3) = 25: Array5(4) = 58: Array5(5) = 143
Array5(6) = 327: Array5(7) = 582: Array5(8) = 1036: Array5(9) = 1842
Array5(10) = 3276: Array5(11) = 5957: Array5(12) = 8638: Array5(13) = 11327
Array5(14) = 14000: Array5(15) = 16681: Array5(16) = 19362: Array5(17) = 22043
Array5(18) = 24724: Array5(19) = 27405: Array5(20) = 30086: Array5(21) = 32767

'Array6 hold values of 4000Hz
Array6(1) = 0: Array6(2) = 10: Array6(3) = 25: Array6(4) = 58: Array6(5) = 143
Array6(6) = 327: Array6(7) = 582: Array6(8) = 1036: Array6(9) = 1842
Array6(10) = 3276: Array6(11) = 5957: Array6(12) = 8638: Array6(13) = 11327 Array6(14) = 14000: Array6(15) = 16681: Array6(16)
= 19362: Array6(17) = 22043
Array6(18) = 24724: Array6(19) = 27405: Array6(20) = 30086: Array6(21) = 32767

'Array7 hold values of 8000Hz
Array7(1) = 0: Array7(2) = 89: Array7(3) = 309: Array7(4) = 786: Array7(5) = 2725
Array7(6) = 5075: Array7(7) = 9450: Array7(8) = 14943: Array7(9) = 16370
Array7(10) = 18005: Array7(11) = 19640: Array7(12) = 21275: Array7(13) = 22910
Array7(14) = 24545: Array7(15) = 26180: Array7(16) = 27815: Array7(17) = 29450
Array7(18) = 31085: Array7(19) = 31623: Array7(20) = 32325: Array7(21) = 32767

  values = (hsbfreq.Value * 32) + (20 - vsbamp.Value)
  hval = hsbfreq.Value
  Select Case hval
            Case 0
              vval = Array1(vsbamp.Value - 1)
            Case 1
              vval = Array2(vsbamp.Value - 1)
            Case 2
              vval = Array3(vsbamp.Value - 1)
            Case 3
              vval = Array4(vsbamp.Value - 1)
            Case 4
              vval = Array5(vsbamp.Value - 1)
            Case 5
              vval = Array6(vsbamp.Value - 1)
            Case 6
              vval = Array7(vsbamp.Value - 1)
End Select
  Comm2.PortOpen = True
  Comm2.Output = Chr(28)
    For n = 1 To 100
    For K = 1 To 35
    Next K
    Next n
   Comm2.Output = Chr(2 ^ (hval))
    For n = 1 To 100
    For K = 1 To 35
    Next K
    Next n
```

```
 v3 = Int(vval / 256)
 v5 = vval - (v3 * 256)
Comm2.Output = Chr(v3)
For n = 1 To 100
 For K = 1 To 35
 Next K
 Next n
 Comm2.Output = Chr(v5)
 Comm2.PortOpen = False
 End If
End Sub

Sub hsbfreq_Scroll ()
  txtfreq.Text = (2 ^ hsbfreq.Value) * 125
End Sub

Sub LPF_Mousedown (Button As Integer, Shift As Integer, X As Single, Y As Single)
   Comm2.PortOpen = True
   Comm2.Output = Chr(22)
   Comm2.PortOpen = False
End Sub

Sub mnucon_Click ()
  values = (hsbfreq.Value * 32) + (20 - vsbamp.Value)
  Comm2.PortOpen = True
  Comm2.Output = Chr(23)
  Comm2.PortOpen = False
End Sub

Sub mnunatt_Click ()
  Comm2.PortOpen = True
  Comm2.Output = Chr(22)
  Comm2.PortOpen = False
End Sub

Sub mnunew_Click ()
Cls
' Save CurrentRecord.
SaveCurentRecord
' Add a new blank record.
LastRecord = LastRecord + 1
Person.Name = " "
Person.Date = " "
Person.pxx0 = 0:  Person.pxx1 = 0:  Person.pxx2 = 0:  Person.pxx3 = 0:  Person.pxx4 = 0:  Person.pxx5 = 0
Person.pxx6 = 0:  Person.pyy0 = 0:  Person.pyy1 = 0:  Person.pyy2 = 0:  Person.pyy3 = 0:  Person.pyy4 = 0
Person.pyy5 = 0:  Person.pyy6 = 0
Person.Rpxx0 = 0:  Person.Rpxx1 = 0:  Person.Rpxx2 = 0:  Person.Rpxx3 = 0:  Person.Rpxx4 = 0
Person.Rpxx5 = 0:  Person.Rpxx6 = 0:  Person.Rpyy0 = 0:  Person.Rpyy1 = 0:  Person.Rpyy2 = 0
Person.Rpyy3 = 0:  Person.Rpyy4 = 0:  Person.Rpyy5 = 0:  Person.Rpyy6 = 0
Put #FileNum, LastRecord, Person
' Update currentRecord.
CurrentRecord = LastRecord
'Display the record that was just created.
ShowCurrentRecord
' Give the focus to the txtName field.
txtName.SetFocus
End Sub

Sub mnunext_Click ()
If txtinfo = " Test Incomlete" Then
Exit Sub
End If
Cls
 ' If the currentRecord is not the last record,
 ' then save the CurrentRecord and skip the next
 ' record.
 If CurrentRecord = LastRecord Then
    Beep
    MsgBox "End of file encountered!", 48
     ShowCurrentRecord
 Else
    SaveCurentRecord
    CurrentRecord = CurrentRecord + 1
    ShowCurrentRecord
 End If
' Give focus to the txtName field.
txtName.SetFocus
```

```
End Sub

Sub mnuprevious_Click ()
Cls
   ' If the currentRecord is not the first Record,
   ' then save the current record and and skip one
   ' record backwards.
   If CurrentRecord = 1 Then
     Beep
     MsgBox "Beginning of the file encountered!", 48
          ShowCurrentRecord
   Else
     SaveCurentRecord
     CurrentRecord = CurrentRecord - 1
     ShowCurrentRecord
   End If
  ' Give the focus to the txtName field.
  txtName.SetFocus
End Sub

Sub mnusearch_Click ()
Cls
Dim NameToSearch As String
Dim Found As Integer
Dim RecNum As Long
Dim TmpPerson As PersonInfo
'Get the name to search from the uesr.
NameToSearch = InputBox("Enter name to search:", "Search")
'If the user did not enter a name, then exit
'from the procedure.
If NameToSearch = " " Then
  'Give the focus to txtName field.
  txtName.SetFocus
  'Exit this procedure.
  Exit Sub
End If

' Convert the name to be search to uppercase.
NameToSearch = UCase(NameToSearch)
' Initialize the Found flag to false.
Found = False
' Search the name the user entered.
For RecNum = 1 To LastRecord
   Get #FileNum, RecNum, TmpPerson
    If NameToSearch = UCase(Trim(TmpPerson.Name)) Then
          Found = True
          Exit For
    End If
Next
' If the name was found then display the record
' of the found name.
If Found = True Then
  SaveCurentRecord
  CurrentRecord = RecNum
  ShowCurrentRecord
Else
  MsgBox "Name: " + NameToSearch + " not found!"
End If
'Give focus to the txtName field.
txtName.SetFocus


End Sub

Sub mnusec_Click ()
  values = (hsbfreq.Value * 32) + (20 - vsbamp.Value)
  Comm2.PortOpen = True
  Comm2.Output = Chr(24)
  Comm2.PortOpen = False
End Sub

Sub mnusimulation_Click ()

'**** SIMULATION OF HEARING ****
```

```
Dim n, vl11, vl2  As String, vl3 As String, w1 As Double, dt As Double, II As Double, pi As Double, L, q, p, NS1, NSS, M, K As
Long, T As Long, al, J, X, DE, E, F, w, Ns, IN, KK As Double, K1 As Double, N2, NM, JO, IO, CR As Long, CI As Long, LE, L1, UR
As Double, UI As Double, JJ, IK, IP, fg
Dim a1, v1, v2 As Long, v3 As Long, v4 As String, v5 As Long, v6 As String
ReDim hr(600) As Double, he(600) As Double, bb(600) As Double, b(600) As Double,Dim crrec1, crrec2

If pxx0 = 0 Or pxx1 = 0 Or pxx2 = 0 Or pxx3 = 0 Or pxx4 = 0 Or pxx5 = 0 Or pxx6 = 0 Or Rpxx0 = 0 Or Rpxx1 = 0 Or Rpxx2 = 0 Or
Rpxx3 = 0 Or Rpxx4 = 0 Or Rpxx5 = 0 Or Rpxx6 = 0 Then
 txtinfo.Text = " Test Incomlete"
 Else
' **** Table with lookup values of SPL levels ****
```

A

```
'Array1 hold values of 125Hz
Array1(1) = 0: Array1(2) = 89: Array1(3) = 309: Array1(4) = 786:Array1(5) = 2725: Array1(6) = 5075
Array1(7) = 9450: Array1(8) = 14943: Array1(9) = 16370: Array1(10) = 18005: Array1(11) = 19640
Array1(12) = 21275: Array1(13) = 22910:Array1(14) = 24545: Array1(15) = 26180: Array1(16) = 27815
Array1(17) = 29450: Array1(18) = 31085: Array1(19) = 31623: Array1(20) = 32325: Array1(21) = 32767

'Array2 hold values of 250Hz
Array2(1) = 0: Array2(2) = 89: Array2(3) = 309: Array2(4) = 786: Array2(5) = 2725: Array2(6) = 5075
Array2(7) = 9450: Array2(8) = 14943: Array2(9) = 16370: Array2(10) = 18005: Array2(11) = 19640
Array2(12) = 21275 Array2(13) = 22910: Array2(14) = 24545: Array2(15) = 26180: Array2(16) = 27815
Array2(17) = 29450: Array2(18) = 31085: Array2(19) = 31623  Array2(20) = 32325: Array2(21) = 32767

'Array3 hold values of 500Hz
Array3(1) = 0: Array3(2) = 2: Array3(3) = 5: Array3(4) = 14: Array3(5) = 45:Array3(6) = 103: Array3(7) = 255 Array3(8) = 582:
Array3(9) = 1036:Array3(10) = 1846: Array3(11) = 3276: Array3(12) = 4455: Array3(13) = 6007
Array3(14) = 7559: Array3(15) = 9111: Array3(16) = 10663: Array3(17) = 12215: Array3(18) = 13767
Array3(19) = 15319: Array3(20) = 16871 Array3(21) = 18426

'Array4 hold values of 1000Hz
Array4(1) = 0: Array4(2) = 2: Array4(3) = 5: Array4(4) = 14: Array4(5) = 45:Array4(6) = 103: Array4(7) = 255
Array4(8) = 582: Array4(9) = 1036:Array4(10) = 1846: Array4(11) = 3276: Array4(12) = 4455: Array4(13) = 6007
Array4(14) = 7559: Array4(15) = 9111: Array4(16) = 10663: Array4(17) = 12215: Array4(18) = 13767
Array4(19) = 15319: Array4(20) = 16871 Array4(21) = 18426

'Array5 hold values of 2000Hz
Array5(1) = 0: Array5(2) = 10: Array5(3) = 25: Array5(4) = 58: Array5(5) = 143: Array5(6) = 327: Array5(7) = 582 Array5(8) = 1036:
Array5(9) = 1842: Array5(10) = 3276: Array5(11) = 5957: Array5(12) = 8638: Array5(13) = 11327
Array5(14) = 14000: Array5(15) = 16681: Array5(16) = 19362 Array5(17) = 22043: Array5(18) = 24724
Array5(19) = 27405: Array5(20) = 30086: Array5(21) = 32767

'Array6 hold values of 4000Hz
Array6(1) = 0: Array6(2) = 10: Array6(3) = 25: Array6(4) = 58: Array6(5) = 143: Array6(6) = 327: Array6(7) = 582 Array6(8) = 1036:
Array6(9) = 1842: Array6(10) = 3276: Array6(11) = 5957: Array6(12) = 8638: Array6(13) = 11327 Array6(14) = 14000: Array6(15) =
16681: Array6(16) = 19362 Array6(17) = 22043: Array6(18) = 24724
Array6(19) = 27405: Array6(20) = 30086: Array6(21) = 32767


'Array7 hold values of 8000Hz
Array7(1) = 0: Array7(2) = 89: Array7(3) = 309: Array7(4) = 786: Array7(5) = 2725: Array7(6) = 5075
Array7(7) = 9450: Array7(8) = 14943: Array7(9) = 16370: Array7(10) = 18005: Array7(11) = 19640
Array7(12) = 21275: Array7(13) = 22910: Array7(14) = 24545: Array7(15) = 26180: Array7(16) = 27815
Array7(17) = 29450: Array7(18) = 31085: Array7(19) = 31623 Array7(20) = 32325: Array7(21) = 32767

'left channel's amplitude values
```

B

```
 Tstrspl1 = Array1(21 - ((pyy0 - 1440) / 120))
 Tstrspl2 = Array2(21 - ((pyy1 - 1440) / 120))
 Tstrspl3 = Array3(21 - ((pyy2 - 1440) / 120))
 Tstrspl4 = Array4(21 - ((pyy3 - 1440) / 120))
 Tstrspl5 = Array5(21 - ((pyy4 - 1440) / 120))
 Tstrspl6 = Array6(21 - ((pyy5 - 1440) / 120))
 Tstrspl7 = Array7(21 - ((pyy6 - 1440) / 120))
```

C

```
'**** LINEAR PIECE-WISE INTERPOLATION ****
For I = 1 To 256
If I <= 2 Then
   xa(I) = Tstrspl1
ElseIf I > 2 And I <= 4 Then
   xa(I) = Tstrspl1 - (((Tstrspl1 - Tstrspl2) / 2) * (I - 2))
ElseIf I > 4 And I <= 8 Then
   xa(I) = Tstrspl2 - (((Tstrspl2 - Tstrspl3) / 4) * (I - 4))
ElseIf I > 8 And I <= 16 Then
   xa(I) = Tstrspl3 - (((Tstrspl3 - Tstrspl4) / 8) * (I - 8))
ElseIf I > 16 And I <= 32 Then
   xa(I) = Tstrspl4 - (((Tstrspl4 - Tstrspl5) / 16) * (I - 16))
ElseIf I > 32 And I <= 64 Then
```

```
    xa(I) = Tstrspl5 - (((Tstrspl5 - Tstrspl6) / 32) * (I - 32))
  ElseIf I > 64 And I <= 128 Then
    xa(I) = Tstrspl6 - (((Tstrspl6 - Tstrspl7) / 64) * (I - 64))
  ElseIf I > 128 And I <= 256 Then
    xa(I) = Tstrspl7
  End If
  Next I
  xa(0) = Tstrspl1
  xa(256) = Tstrspl7

D '**** FIR DESIGN BY FREQUENCY SAMPLING ****
  '**** IDFT ****
  'n = 256:  l = n + n + 1:  p = 3.14
  'M = 256                    ' Order of the filter m = 256
  'For I = 1 To 256
  '   HA(I) = (-1 * (p / 2)) - (I * M * p / l)' Quad phase
  'Next I:   HA(0) = 0
  'For I = n + 1 To l - 1
  '   xa(I) = xa(l - I): HA(I) = -HA(l - I)  ' Complex conjucate
  'Next I
  ' * Windowed inverse DFT (Real Pt only)
  'For I = 0 To M
  '   A(I) = 0
  '   For K = 0 To l - 1
  '      T = 2 * p * I * K / l
  '      A(I) = A(I) + xa(K) * Cos(T + HA(K))
  '   Next K
  '   A(I) = A(I) / l
  'Next I

E '**** WINDOWS ****
  ' * Kaiser window
  ' AL = 3                    ' Alpha = 3
  'For J = 0 To M: I = J - M / 2
  '   X = AL * Sqr(1 - I * I * 4 / M / M):  DE = 1:  E = 1
  '   For K = 1 To 25
  '      DE = DE * X / K / 2:   E = E + DE * DE
  '   Next K:  DE = 1: F = 1
  '      For K = 1 To 25
  '         DE = DE * AL / K / 2:  F = F + DE * DE
  '      Next K
  '   W = E / F:  A(J) = A(J) * W
  'Next J
  '*** Hamming window ***
  'For J = 0 To M: I = J - M / 2
  '   W = .5 * (1 + Cos(3.1415 * I / (M / 2 + 1)))
  '   A(J) = A(J) * W
  'Next J:
  '*** Hamming window ***
  'For J = 0 To M: I = J - M / 2
  '   W = .54 + .46 * Cos(3.1415 * I * 2 / M)
  '   A(J) = A(J) * W
  'Next J:
  'printer.Print "H(k)"; "          h(n) "

F '**** INVERSE FAST FOURIER TRANSFORM ****
  ReDim A(600)
  pi = 3.14159265359
  M = 9
  Ns = 2 ^ M                                          ' Number of freq.samples
  NS1 = Ns / 2 + 1
  NSS = Ns / 2
  dt = .5 * (NS1)
  ReDim XR(Ns) As Double, XI(Ns) As Double, SI(Ns) As Double, CO(Ns) As Double
  Rem **** Fourier time shift ****
  For I = 0 To NS1 - 1 Step 1
    hr(I) = xa(I) * (Cos(dt * pi * I / (NS1 - 1)))
    he(I) = -1 * xa(I) * (Sin(dt * pi * I / (NS1 - 1)))
  Next I
  For p = NS1 To (2 * (NS1 - 1) - 1) Step 1
    hr(p) = hr((2 * (NS1 - 1)) - p)
    he(p) = -1 * he((2 * (NS1 - 1)) - p)
  Next p
  '******* Define real and imaginary parts of input ******
  For n = 0 To Ns - 1:  XR(n) = hr(n): Next n
  For n = 0 To Ns - 1: XI(n) = he(n): Next n
  '******* IN = -1 for IFFT ****

G                                                    Appendix B pg11
```

```
IN = -1

'***** FFT ROUTINE *****
'** COMPUTE TWIDDLE FACTOR **
KK = 6.28318 / Ns: Rem 2PI/NS
For n = 0 To Ns - 1
K1 = n * KK
CO(n) = Cos(K1): SI(n) = -Sin(K1) * IN
 Next n

'**** BIT REVERSAL ****
Ns = 2 ^ M: N2 = Ns / 2: NM = Ns - 1: JO = 0
For IO = 0 To NM - 1
   If IO > JO Then GoTo GF
     CR = XR(JO): CI = XI(JO)
     XR(JO) = XR(IO): XI(JO) = XI(IO)
     XR(IO) = CR: XI(IO) = CI
  GF: K = N2
  GBT: If K >= JO + 1 Then GoTo GFT
    JO = JO - K: K = K / 2: GoTo GBT
    GFT: JO = JO + K
Next IO

'*** FFT ALGORITHM ****
For L = 1 To M
 LE = 2 ^ L: L1 = LE / 2: UR = 1: UI = 0: JJ = 0
 IK = Ns / LE
  For J = 0 To 2 ^ (L - 1) - 1
   For I = J To Ns - 1 Step LE
          IP = I + L1
          CR = XR(IP) * UR - XI(IP) * UI
          CI = XI(IP) * UR + XR(IP) * UI
          XR(IP) = XR(I) - CR: XI(IP) = XI(I) - CI
          XR(I) = XR(I) + CR: XI(I) = XI(I) + CI
   Next I
    JJ = JJ + IK
    UR = CO(JJ): UI = SI(JJ)
  Next J
Next L

'*** DO IDFT ***
If IN > 0 Then GoTo LST
   For n = 0 To Ns - 1
     XR(n) = XR(n) / Ns: XI(n) = XI(n) / Ns
   Next n
LST: For n = 0 To NSS Step 1
       b(n) = XR(n)
     Next n

'**** Kaiser window ****
' AL = 3                          ' Alpha = 3
'For J = 0 To M: I = J - M / 2
'   X = AL * Sqr(1 - I * I * 4 / M / M):  DE = 1:  E = 1
' For K = 1 To 25
'    DE = DE * X / K / 2:   E = E + DE * DE
' Next K:  DE = 1: F = 1
'   For K = 1 To 25
'      DE = DE * AL / K / 2:  F = F + DE * DE
'   Next K
' W = E / F:  b(J) = bb(J) * W
'Next J

'**** Hamming window ****
' For J = 0 To NSS Step 1
'   II = J - (NSS) / 2
'   w1 = .54 + .46 * Cos(2 * pi * II / (NSS - 1))
'   b(J) = bb(J) * w1
'Next J

'**** Transfer filter coefficient to DSP via Comm. Port 2 ****
 Comm2.PortOpen = True          ' Open Comm. Port 2 and
 Comm2.Output = Chr(27)         ' instruct DSP to receive  two eigth bit words
 For I = 0 To NSS - 1
    a1 = Int(b(I))
  If a1 < 0 Then                ' Negative values
    v2 = 65535 + a1
    v3 = Int(v2 / 256)
```

```
        v5 = v2 - (v3 * 256)
    Else
        v3 = Int(a1 / 256)              'Positive values
        v5 = a1 - (v3 * 256)
    End If
    Comm2.Output = Chr(v3)          ' Output higher significant bits
    For n = 1 To 100
     For K = 1 To 35
     Next K
     Next n
     Comm2.Output = Chr(v5)          ' Output lower significant bits
        For n = 1 To 100
        For K = 1 To 35
        Next K
        Next n
    Next I
     Comm2.PortOpen = False          ' Close port.

     '**************************************
     '******** Do right channel **********
     '**************************************

     Tstrspl1 = Array1(21 - ((Rpyy0 - 1440) / 120))
     Tstrspl2 = Array2(21 - ((Rpyy1 - 1440) / 120))
     Tstrspl3 = Array3(21 - ((Rpyy2 - 1440) / 120))
     Tstrspl4 = Array4(21 - ((Rpyy3 - 1440) / 120))
     Tstrspl5 = Array5(21 - ((Rpyy4 - 1440) / 120))
     Tstrspl6 = Array6(21 - ((Rpyy5 - 1440) / 120))
     Tstrspl7 = Array7(21 - ((Rpyy6 - 1440) / 120))

    For I = 1 To 256
    If I <= 2 Then
        xa(I) = Tstrspl1
    ElseIf I > 2 And I <= 4 Then
        xa(I) = Tstrspl1 - (((Tstrspl1 - Tstrspl2) / 2) * (I - 2))
    ElseIf I > 4 And I <= 8 Then
        xa(I) = Tstrspl2 - (((Tstrspl2 - Tstrspl3) / 4) * (I - 4))
    ElseIf I > 8 And I <= 16 Then
        xa(I) = Tstrspl3 - (((Tstrspl3 - Tstrspl4) / 8) * (I - 8))
    ElseIf I > 16 And I <= 32 Then
        xa(I) = Tstrspl4 - (((Tstrspl4 - Tstrspl5) / 16) * (I - 16))
    ElseIf I > 32 And I <= 64 Then
        xa(I) = Tstrspl5 - (((Tstrspl5 - Tstrspl6) / 32) * (I - 32))
    ElseIf I > 64 And I <= 128 Then
        xa(I) = Tstrspl6 - (((Tstrspl6 - Tstrspl7) / 64) * (I - 64))
    ElseIf I > 128 And I <= 256 Then
        xa(I) = Tstrspl7
    End If
     Next I
     xa(0) = Tstrspl1
     xa(256) = Tstrspl7

    Rem **** Fourier time shift ****
     For I = 0 To NS1 - 1 Step 1
     hr(I) = xa(I) * (Cos(dt * pi * I / (NS1 - 1)))
     he(I) = -1 * xa(I) * (Sin(dt * pi * I / (NS1 - 1)))
    Next I
     For p = NS1 To (2 * (NS1 - 1) - 1) Step 1
       hr(p) = hr((2 * (NS1 - 1)) - p)
       he(p) = -1 * he((2 * (NS1 - 1)) - p)
     Next p

    '******* Define real and imaginary parts of input ******
     For n = 0 To Ns - 1: XR(n) = hr(n): Next n
     For n = 0 To Ns - 1: XI(n) = he(n): Next n
    '******* IN = -1 for ifft
     IN = -1

    '***** FFT ROUTINE *****
    '** COMPUTE TWIDDLE FACTOR **
    KK = 6.28318 / Ns: Rem 2PI/NS
    For n = 0 To Ns - 1
    K1 = n * KK
    CO(n) = Cos(K1): SI(n) = -Sin(K1) * IN
     Next n
    '** BIT REVERSAL **
    Ns = 2 ^ M: N2 = Ns / 2: NM = Ns - 1: JO = 0
```

```
    Comm2.PortOpen = True
    Comm2.Output = Chr(25)
    Comm2.PortOpen = False
End Sub

Sub Optright_Click ()
    Comm2.PortOpen = True
    Comm2.Output = Chr(26)
    Comm2.PortOpen = False
End Sub

Sub SaveCurentRecord ()
' Fill Person with the currently displayed data.
Person.Name = txtName.Text
Person.Date = txtdate.Text
Person.pxx0 = pxx0:Person.pxx1 = pxx1:Person.pxx2 = pxx2:Person.pxx3 = pxx3
Person.pxx4 = pxx4:Person.pxx5 = pxx5:Person.pxx6 = pxx6
Person.pyy0 = pyy0:Person.pyy1 = pyy1:Person.pyy2 = pyy2:Person.pyy3 = pyy3
Person.pyy4 = pyy4:Person.pyy5 = pyy5:Person.pyy6 = pyy6
Person.Rpxx0 = Rpxx0:Person.Rpxx1 = Rpxx1:Person.Rpxx2 = Rpxx2:Person.Rpxx3 = Rpxx3
Person.Rpxx4 = Rpxx4:Person.Rpxx5 = Rpxx5:Person.Rpxx6 = Rpxx6
Person.Rpyy0 = Rpyy0:Person.Rpyy1 = Rpyy1:Person.Rpyy2 = Rpyy2:Person.Rpyy3 = Rpyy3
Person.Rpyy4 = Rpyy4:Person.Rpyy5 = Rpyy5:Person.Rpyy6 = Rpyy6
' Save person to the current record
Put #FileNum, CurrentRecord, Person
End Sub

Sub ShowCurrentRecord ()
' Fill Person with the data of the current
' record
Get #FileNum, CurrentRecord, Person
' Display Person.
txtName.Text = Trim(Person.Name)
txtdate.Text = Trim(Person.Date)
pxx0 = Trim(Person.pxx0): pxx1 = Trim(Person.pxx1): pxx2 = Trim(Person.pxx2): pxx3 = Trim(Person.pxx3)
pxx4 = Trim(Person.pxx4): pxx5 = Trim(Person.pxx5): pxx6 = Trim(Person.pxx6)
pyy0 = Trim(Person.pyy0): pyy1 = Trim(Person.pyy1): pyy2 = Trim(Person.pyy2): pyy3 = Trim(Person.pyy3)
pyy4 = Trim(Person.pyy4): pyy5 = Trim(Person.pyy5): pyy6 = Trim(Person.pyy6)
Rpxx0 = Trim(Person.Rpxx0): Rpxx1 = Trim(Person.Rpxx1): Rpxx2 = Trim(Person.Rpxx2)
Rpxx3 = Trim(Person.Rpxx3): Rpxx4 = Trim(Person.Rpxx4): Rpxx5 = Trim(Person.Rpxx5)
Rpxx6 = Trim(Person.Rpxx6): Rpyy0 = Trim(Person.Rpyy0): Rpyy1 = Trim(Person.Rpyy1)
Rpyy2 = Trim(Person.Rpyy2): Rpyy3 = Trim(Person.Rpyy3): Rpyy4 = Trim(Person.Rpyy4)
Rpyy5 = Trim(Person.Rpyy5): Rpyy6 = Trim(Person.Rpyy6):

Line (pxx0, pyy0)-(pxx1, pyy1): Line (pxx1, pyy1)-(pxx2, pyy2): Line (pxx2, pyy2)-(pxx3, pyy3)
Line (pxx3, pyy3)-(pxx4, pyy4): Line (pxx4, pyy4)-(pxx5, pyy5): Line (pxx5, pyy5)-(pxx6, pyy6)
Circle (pxx6, pyy6), 50: Circle (pxx5, pyy5), 50: Circle (pxx4, pyy4), 50: Circle (pxx3, pyy3), 50
Circle (pxx2, pyy2), 50: Circle (pxx1, pyy1), 50: Circle (pxx0, pyy0), 50

Line (Rpxx0, Rpyy0)-(Rpxx1, Rpyy1), QBColor(4)
Line (Rpxx1, Rpyy1)-(Rpxx2, Rpyy2), QBColor(4)
Line (Rpxx2, Rpyy2)-(Rpxx3, Rpyy3), QBColor(4)
Line (Rpxx3, Rpyy3)-(Rpxx4, Rpyy4), QBColor(4)
Line (Rpxx4, Rpyy4)-(Rpxx5, Rpyy5), QBColor(4)
Line (Rpxx5, Rpyy5)-(Rpxx6, Rpyy6), QBColor(4)
    DrawWidth = 1
    Line ((Rpxx0 - 50), (Rpyy0 - 50))-((Rpxx0 + 60), (Rpyy0 + 60)), QBColor(4)
    Line ((Rpxx0 - 50), (Rpyy0 + 50))-((Rpxx0 + 60), (Rpyy0 - 60)), QBColor(4)
    Line ((Rpxx1 - 50), (Rpyy1 - 50))-((Rpxx1 + 60), (Rpyy1 + 60)), QBColor(4)
    Line ((Rpxx1 - 50), (Rpyy1 + 50))-((Rpxx1 + 60), (Rpyy1 - 60)), QBColor(4)
    Line ((Rpxx2 - 50), (Rpyy2 - 50))-((Rpxx2 + 60), (Rpyy2 + 60)), QBColor(4)
    Line ((Rpxx2 - 50), (Rpyy2 + 50))-((Rpxx2 + 60), (Rpyy2 - 60)), QBColor(4)
    Line ((Rpxx3 - 50), (Rpyy3 - 50))-((Rpxx3 + 60), (Rpyy3 + 60)), QBColor(4)
    Line ((Rpxx3 - 50), (Rpyy3 + 50))-((Rpxx3 + 60), (Rpyy3 - 60)), QBColor(4)
    Line ((Rpxx4 - 50), (Rpyy4 - 50))-((Rpxx4 + 60), (Rpyy4 + 60)), QBColor(4)
    Line ((Rpxx4 - 50), (Rpyy4 + 50))-((Rpxx4 + 60), (Rpyy4 - 60)), QBColor(4)
    Line ((Rpxx5 - 50), (Rpyy5 - 50))-((Rpxx5 + 60), (Rpyy5 + 60)), QBColor(4)
    Line ((Rpxx5 - 50), (Rpyy5 + 50))-((Rpxx5 + 60), (Rpyy5 - 60)), QBColor(4)
    Line ((Rpxx6 - 50), (Rpyy6 - 50))-((Rpxx6 + 60), (Rpyy6 + 60)), QBColor(4)
    Line ((Rpxx6 - 50), (Rpyy6 + 50))-((Rpxx6 + 60), (Rpyy6 - 60)), QBColor(4)

' Display the current record number in the
' caption of the form.
frmaud.Caption = "Record " + Str(CurrentRecord) + " " + Str(LastRecord)
End Sub

Sub vsbamp_Change ()
```

```
txtamp.Text = vsbamp.Value * 5
End Sub

Sub vsbamp_KeyDown (keycode As Integer, Shift As Integer)
If keycode = 37 Or keycode = 39 Then
   hsbfreq.SetFocus
Elself keycode = 32 Then
Dim hval, vval, n, K, v3, v5
Dim hst As String

'Array1 hold values of 125Hz
Array1(1) = 0: Array1(2) = 89: Array1(3) = 309: Array1(4) = 786: Array1(5) = 2725: Array1(6) = 5075: Array1(7) = 9450: Array1(8) =
14943: Array1(9) = 16370 Array1(10) = 18005: Array1(11) = 19640: Array1(12) = 21275 Array1(13) = 22910: Array1(14) = 24545:
Array1(15) = 26180: Array1(16) = 27815: Array1(17) = 29450: Array1(18) = 31085: Array1(19) = 31623 Array1(20) = 32325:
Array1(21) = 32767

'Array2 hold values of 250Hz
Array2(1) = 0: Array2(2) = 89: Array2(3) = 309: Array2(4) = 786: Array2(5) = 2725: Array2(6) = 5075: Array2(7) = 9450: Array2(8) =
14943: Array2(9) = 16370: Array2(10) = 18005: Array2(11) = 19640: Array2(12) = 21275 Array2(13) = 22910: Array2(14) = 24545:
Array2(15) = 26180: Array2(16) = 27815: Array2(17) = 29450: Array2(18) = 31085: Array2(19) = 31623 Array2(20) = 32325:
Array2(21) = 32767

'Array3 hold values of 500Hz
Array3(1) = 0: Array3(2) = 2: Array3(3) = 5: Array3(4) = 14: Array3(5) = 45
Array3(6) = 103: Array3(7) = 255: Array3(8) = 582: Array3(9) = 1036
Array3(10) = 1846: Array3(11) = 3276: Array3(12) = 4455: Array3(13) = 6007
Array3(14) = 7559: Array3(15) = 9111: Array3(16) = 10663: Array3(17) = 12215
Array3(18) = 13767: Array3(19) = 15319: Array3(20) = 16871 Array3(21) = 18426

'Array4 hold values of 1000Hz
Array4(1) = 0: Array4(2) = 2: Array4(3) = 5: Array4(4) = 14: Array4(5) = 45
Array4(6) = 103: Array4(7) = 255: Array4(8) = 582: Array4(9) = 1036
Array4(10) = 1846: Array4(11) = 3276: Array4(12) = 4455: Array4(13) = 6007
Array4(14) = 7559: Array4(15) = 9111: Array4(16) = 10663: Array4(17) = 12215
Array4(18) = 13767: Array4(19) = 15319: Array4(20) = 16871 Array4(21) = 18426

'Array5 hold values of 2000Hz
Array5(1) = 0: Array5(2) = 10: Array5(3) = 25: Array5(4) = 58: Array5(5) = 143: Array5(6) = 327
Array5(7) = 582: Array5(8) = 1036: Array5(9) = 1842: Array5(10) = 3276: Array5(11) = 5957
Array5(12) = 8638: Array5(13) = 11327: Array5(14) = 14000: Array5(15) = 16681: Array5(16) = 19362
Array5(17) = 22043: Array5(18) = 24724: Array5(19) = 27405: Array5(20) = 30086: Array5(21) = 32767

'Array6 hold values of 4000Hz
Array6(1) = 0: Array6(2) = 10: Array6(3) = 25: Array6(4) = 58: Array6(5) = 143: Array6(6) = 327
Array6(7) = 582: Array6(8) = 1036: Array6(9) = 1842: Array6(10) = 3276: Array6(11) = 5957
Array6(12) = 8638: Array6(13) = 11327 Array6(14) = 14000: Array6(15) = 16681: Array6(16) = 19362
Array6(17) = 22043: Array6(18) = 24724: Array6(19) = 27405: Array6(20) = 30086: Array6(21) = 32767


'Array7 hold values of 8000Hz
Array7(1) = 0: Array7(2) = 89: Array7(3) = 309: Array7(4) = 786: Array7(5) = 2725: Array7(6) = 5075
Array7(7) = 9450: Array7(8) = 14943: Array7(9) = 16370: Array7(10) = 18005: Array7(11) = 19640
Array7(12) = 21275 Array7(13) = 22910: Array7(14) = 24545: Array7(15) = 26180: Array7(16) = 27815
Array7(17) = 29450: Array7(18) = 31085: Array7(19) = 31623 Array7(20) = 32325: Array7(21) = 32767

  values = (hsbfreq.Value * 32) + (20 - vsbamp.Value)
  hval = hsbfreq.Value
  Select Case hval
          Case 0
            vval = Array1(vsbamp.Value + 1)
          Case 1
            vval = Array2(vsbamp.Value + 1)
          Case 2
            vval = Array3(vsbamp.Value + 1)
          Case 3
            vval = Array4(vsbamp.Value + 1)
          Case 4
            vval = Array5(vsbamp.Value + 1)
          Case 5
            vval = Array6(vsbamp.Value + 1)
          Case 6
            vval = Array7(vsbamp.Value + 1)
End Select
   Comm2.PortOpen = True
   Comm2.Output = Chr(28)
   For n = 1 To 100
   For K = 1 To 35
```

```
    Next K
    Next n
   Comm2.Output = Chr(2 ^ (hval))
    For n = 1 To 100
    For K = 1 To 35
    Next K
    Next n
  v3 = Int(vval / 256)
  v5 = vval - (v3 * 256)
Comm2.Output = Chr(v3)
 For n = 1 To 100
 For K = 1 To 35
 Next K
 Next n
 Comm2.Output = Chr(v5)
 Comm2.PortOpen = False
 End If
End Sub

Sub vsbamp_Scroll ()
txtamp.Text = vsbamp.Value * 5
End Sub
```

Appendix C Audiometer – DSP side

Section K and onwards forms the heart of the Audiometer. The set_delta subroutine under the user routine takes the frequency value in the dm(freq) and multiply it with 2.048. Why it is multiplied with 2.048 will be explained shortly. Take for example a frequency of 1000Hz will now be 2048. i.e. (1000 x 2.048) This value is stored in data memory dm(delta). The ena ar_sat will ensure that this calculated value would not cause a AR (accumulator result register) overflow, which mean that it will not exceed 65536 (16 bit register). In the interrupt routine section M , the angle which is first set to 0 is shifted to the right by 8 positions ,which means that it will be dived by 256. Though it would not have any effect at this stage it will assign the SR(shift register) with 0. The i4 = ^sintable statement will make the index register (i4) point to the starting address of the sine lookup table. In the first iteration m4 which is the modifier register will be 0 (m4 = sr1). Assigning the length register to 0 (l4 = 0) will insure linear buffering. Linear buffering will cause wrapping around at the end of the sine table without causing an interrupt like for autobuffering. Modify (i4,m4) will modify the address at i4 by m4 , which is at first 0. So for the first iteration the content of that address in program memory pointed to by i4 (which is in effect the starting address of the sine lookup table) , is send to the ax1 register that will finally send it out the Transmitter buffer. This is for the first value in the lookup table. The first 4 lines by label M will produce the next address in the lookup table. It is done by first putting delta (step size) which is calculated to be 2048 into the ay1 register. Clearly it could be seen that this value is not the real step size , but rather a magnified value of the real step size.

The real step size should be $\Delta = \frac{Nf}{fs}$ which is $\frac{256 \times 1000}{32000} = 8$

One can see that the step size is magnified or multiplied by 256 ..... $\left(\frac{2048}{8}\right)$

The modified formula for the step size is now $\frac{256 \times 256 \times 1000}{32000} = 2048$

In the next step dis ar_sat insure that AR will not be saturated. The value of the modified step size is now added to AR register which (in executing the interrupt routine the first time) is 0 . When the statement dm(angle) = ar is executed , the variable "angle" in data memory will then have the value of 2048. On executing the interrupt routine for the second time the whole procedure will be repeated but this time with the angle value of 2048. When it is divide by 256 this time around SR will now contain 8 and this is the integer part of the real step size. Following through the same procedure the index register will now point the 8th position from the offset of the lookup table, because when SR is assigned to m4 the i4 is modified by 8. So this value is send out this time.

Further AR still contain the value of 2048 and will be added to delta (2048) so that the angle will become 4096. (In section M) With the next iteration m4 will be 4096/256 = 16. So the content of the 16th address location from the origin of the sine table will be send out this time. Clearly it could be seen that the step size is really 8 in this example. This process will repeat itself up to and exceeding the point were AR riches it maximum value which is 65536. In this example the accumulative angle after the whole sine lookup table is covered falls fortunately exactly on 65536. In all cases of the selected frequencies for this project this happen so that the sine lookup table will always be accessed from its origin. It might happen for other frequencies that a modulus overflow will occur that will bring one at an offset which is not the origin. In such a case AR register is used for the point of wrap around.The accumulative angle plus the delta value will determine the point of offset after wrapping around the maximum value of the AR register which is 65536. From the above it could be seen why it is important to force the real step size into a format that can be measured against the AR register. In other words why one has to multiply or magnify the step size with 256. Getting back to the 2.048 factor it should be clear that from the last equation a formula for this part of the application can be derived

for calculating the modified step size $\Delta = \frac{N \times \frac{\text{mod(registersize)}}{N} \times f}{fs}$

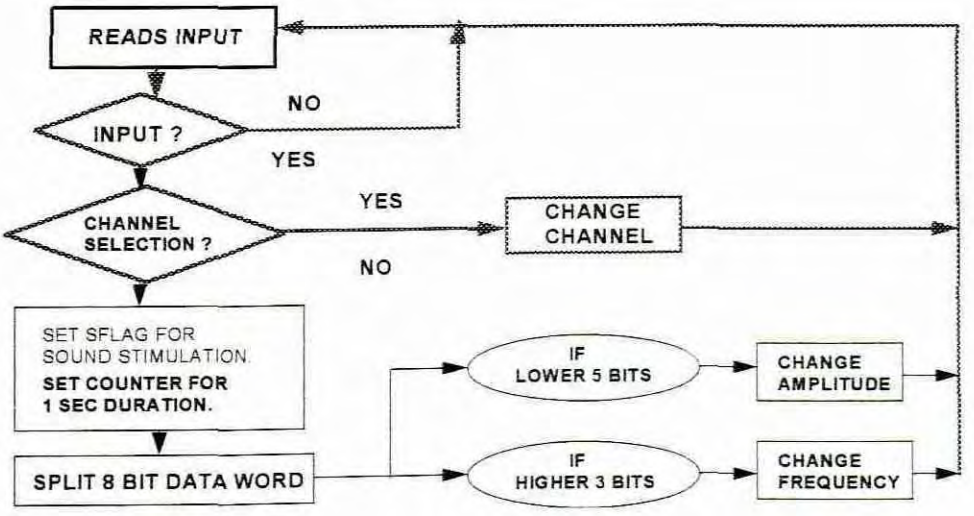
where f is the frequency.
N the number of samples in the sine lookup table.
fs the sampling frequency .
and mod(register_size) the maximum value of register used as modulus logic. (In this case $2^{16} = 65536$)

$$\therefore \Delta = \frac{256 \times \frac{65536}{256} \times f}{32000} \quad \ldots\ldots \quad (N = 256,\ fs = 32000)$$

$$= 2.048 \times f$$

Section N decrement the counter to check if the one second of sound stimulation is finish. If the counter rich zero the jump stflag is executed , were the sflag is set to 0. If not, the current count is stored and the rti is executed. Section O shows that SPORT0 will automatically transmit values of ax1 which is pointed to by the I register from the circular buffers named (tx_buf + 1) for right and (tx_buf + 2) for left. SPORT0 will receive values as they are sent to the SPORT and automatically transfer the data into the receiver buffer. A receive interrupt will be generated once the receiver has been completely filled and a clock pulse is received at the SCLK pin 50 as this port is configured for external interrupt. (The Codec supply the clock pulses.) Every time an interrupt is generated on the SPORT0 the jump tx_samples in the interrupt table is executed which will send the next sample out the port. The last part in section P is an interrupt routine that actually send out the Codecs control information during Codec initialisation. Using the EZ_kit Lite unit the interface is already been taken care of. Both hardware and software supplied is pretty much complete and needed minimal additional configurations. For this project only the clock speed needed to be configured which is used as an external trigger pulse to start the interrupt routine that reads the sinewave values from memory. This is done by changing the 9th index register of the codec to 0xc856 which means that clock speed is set to 32000 Hz.

FLOW AND CONTROL DIAGRAM



Appendix D Wavelet comparison table

By: MISITI, M.; MISITI, Y; OPPENHEIM, G. & POGGI, J. (1996:6-69

	Morlet	mexh	meyer	Haar	db N	SymN	coif	biorNr.Nd
Crude	*	*						
Infinitely regular	*	*	*					
Compactly supported orthogonal				*	*	*	*	
Compactly supported biorthogonal								*
Symmetry	*	*	*	*				
Assymetry					*			
Near symmetry						*	*	
Arbitrary number of vanishing moments					*	*	*	*
Vanishing moments for ϕ							*	
Arbitrary regularity					*	*	*	*
Existence of ϕ			*	*	*	*	*	*
Orthogonal analysis			*	*	*	*	*	
Biorthogonal analysis			*	*	*	*	*	*
Exact reconstruction			*	*	*	*	*	*
FIR filters				*	*	*	*	*
Continues transform	*	*	*	*	*	*	*	*
Discrete transform			*	*	*	*	*	*
Fast algorithm				*	*	*	*	*
Explicit expression	*	*		*				for splines

Appendix E Program to demonstrate two channel and multiresolution subband decoding

```
% Program to demonstrate two channel subband decoding
% with attenuation on high frequencies
close all
r=128;
t=(0:.0000625:1)';              % sampling rate = 16 kHz
x1= sin(2*pi*250*t)+sin(2*pi*500*t)+sin(2*pi*1000*t)+sin(2*pi*2000*t)+sin(2*pi*4000*t);
figure
subplot(3,1,1);plot(t(1:r),x1(1:r));axis([0 .008 -3 3]);title('Original signal');
 [ca1,cd1]=dwt(x1,'db8');
 x2=idwt(ca1,cd1,'db8');
subplot(3,1,2);plot(t(1:r),x2(1:r));axis([0 .008 -3 3]);title('Reconstructed signal with no
attenuated ');
 cd1=cd1*0;
 x3=idwt(ca1,cd1,'db3');
subplot(3,1,3);plot(t(1:r),x3(1:r));axis([0 .008 -3 3]);title('High frequency attenuated ');


% Program to demonstrate multiresolution subband decoding
% with attenuation
close all
w='db2';
r=128;
Fs=16000;
t=(0:1/Fs:1)';              % sampling rate = 16 kHz
 x1=
sin(2*pi*125*t)+sin(2*pi*375*t)+sin(2*pi*750*t)+sin(2*pi*1500*t)+sin(2*pi*3000*t)+sin(2
*pi*6000*t);
figure
k=fft(x1,128);
N=length(k);
f=Fs*(0:N-1)/N;
subplot(3,2,2);plot(f,abs(k));
subplot(3,2,1);plot(t(1:r),x1(1:r));axis([0 .008 -4 4]);title('Original signal');
[C,L]=wavedec(x1,6,w);

A6=wrcoef('a',C,L,w,6);
D6=wrcoef('d',C,L,w,6);
D5=wrcoef('d',C,L,w,5);
D4=wrcoef('d',C,L,w,4);
D3=wrcoef('d',C,L,w,3);
D2=wrcoef('d',C,L,w,2);
D1=wrcoef('d',C,L,w,1);
D1=D1*.3;D2=D2*.4;D3=D3*.5;D4=D4*.6;D5=D5*.7;D6=D6*.8;A6=A6*.9;
x2=D1+D2+D3+D4+D5+D6+A6;

k=fft(x2,128);
N=length(k);
f=Fs*(0:N-1)/N;
subplot(3,2,6);plot(f,abs(k));
subplot(3,2,5);plot(t(1:r),x2(1:r));axis([0 .008 -4 4]);title('Reconstruct attenuated signal ');
```

```
/-------------------------------------------------------
/ Changes to main routine section of DSP program
/-------------------------------------------------------

{*** Main Code Loop ***}

      ax0=0;
      dm(sflag)=ax0;
      dm(fflag)=ax0;
      dm(fcntr)=ax0;                    /* count 18 filter coefficients */
      dm(dlflag)=ax0;                   /* initialise download flag to 0 */
      ax0 = 2;
      dm(cntra) = ax0;                  /* 16 bit Filter coef.= 2 bytes */
      ax0 = 0;
          dm(freq) = ax0;
      ax0 = 0;
      dm(startemp) = ax0;

/-------------------------------------------------------
/  Changes to command loop section of DSP program
/-------------------------------------------------------

stfilf:
          i2=^data;  m1=1;  l2=taps;
          i6 = ^coeftab; l6 = 0; m6 = 1;
          i5 = ^fbcoef; l5 = 0; m5 = 1;
          i4 = ^ffcoef; l4 = 0; m4 = 1;
          ax0 = 9;
          dm(cntrcoef) = ax0;

sepcoefs:    mx0 = pm(i6,m6);
          pm(i5,m5) = mx0;              /* Put first 9 values in feedback buffer*/
          ax0 = dm(cntrcoef);
          ar = ax0 - 1;
          dm(cntrcoef) = ar;
          if ne jump sepcoefs;
          ax0 = 9;
          dm(cntrcoef) = ax0;

sepcoefs1:    mx0 = pm(i6,m6);
          pm(i4,m5) = mx0;              /* Put last 9 values in feedforward buffer*/
          ax0 = dm(cntrcoef);
          ar = ax0 - 1;
          dm(cntrcoef) = ar;
           if ne jump sepcoefs1;
          ax0 = 1;
          dm(fflag) = ax0;              /* set fflag to start filtering */
          ax0 = 0;
          dm(dlflag) = ax0;
          ax0 =0;
          dm(fcntr) = ax0;
          jump main;

/-------------------------------------------------------
/ Changes to Interrupt Routines section of DSP program
/-------------------------------------------------------

{* Interrupt use for sending  tx samples *}
tx_samples:
startfilter:
```

```
        i2=^data; m1=1; l2=taps;
        i6 = ^coeftab; l6 = 0; m6 = 1;
        i5 = ^fbcoef; l5 = taps; m5 = 1;
        i4 = ^ffcoef; l4 = taps; m4 = 1;
        m2 = 2; m0=0;
        mr0 = 0;
        cntr = taps - 2;
        ax0=taps-1;
    sr1=dm(rx_buf+2);
    sr1=dm(rx_buf+1);
    dm(i2,m1)=sr1;
    my0=pm(i5,m4);                          /*ignor 1st value in the denom polinomial*/

     mx0=dm(i2,m1), my0=pm(i5,m4);
     DO poleloop UNTIL CE;
poleloop:          mr=mr+mx0*my0(SS), mx0=dm(i2,m1), my0=pm(i5,m4);
    mr=mr+mx0*my0(RND);
    cntr=ax0;
    dm(i2,m0)=mr1;
    mr=0, mx0=dm(i2,m1), my0=pm(i4,m4);
    DO zeroloop UNTIL CE;
zeroloop:          mr=mr+mx0*my0(SS), mx0=dm(i2,m1), my0=pm(i4,m4);
    mr=mr+mx0*my0(RND);
        MODIFY (i2,m2);
    if mv sat mr;
    sr = ashift mr1 by 1 (lo);
    mr1 = sr0;
    dm (tx_buf + 1) = mr1;
    dm (tx_buf + 2) = mr1;
    rti;
```

```
%-----------------------------------------------------------------------
% Program Yul.m in Matlab to perform Yul-walker function
%-----------------------------------------------------------------------

flid = fopen('c:\VB\Tfile.dat','r+');
 A = fscanf(flid,'%5f');
H =A';
fclose(flid);
f = [0  0.0078  0.0156  0.0312  0.0625  0.125  0.25 0.5 1];
 N = 8;
 [Bh,Ah] = yulewalk(N,f,H);
t = mat2str([Bh,Ah],4);
fid = fopen('yul2.bin','wt+');
fwrite(fid,t,'uchar');
fclose(fid);
End
```

Appendix F Additional software for Yul-walker function

```
'-------------------------------------------------------------------------------------------------
' Part of host program written in Visual Basic that interface with Matlab
'-------------------------------------------------------------------------------------------------

' OPEN A FILE INTO WHICH AMPLITUDE VALUES ARE WRITTEN
Open "TFILE.dat" For Output As #2    ' Open to write file.
For I = 1 To 9
   Print #2, Format(x(I), "#.####");                    ' x(I) = amplitude values
                                                        ' same format for all values
   Print #2, " ";                                       ' space to mark end of file
Next I
   Close #2
L = Shell("c:\matlab\bin\matlab.exe", 1)                ' instruct Matlab to startup
SendKeys "yul{Enter}", True                             ' retrieve the program to instruct Matlab -
                                                        ' to perform Yule-Walkers function

' READ NUMERATOR & DENUMERATOR COEFFICIENTS
temp = ""
acount = 1
filenum = FreeFile
Open "c:\VB\Yul2.bin" For Input As #filenum ' Open file for input.
Do While Not EOF(1) ' Check for end of file.
      char = Input(1, #filenum)
   If char = "[" Then
      GoTo slot
     ElseIf char = "]" Then
       Exit Do
     ElseIf char = " " Then
       GoSub con
     Else
       temp = temp + char
   End If
slot:   Loop
Close #filenum
    arr(acount) = temp
    arr1(acount) = Val(arr(acount))
GoTo jend:
con:  arr(acount) = temp
arr1(acount) = Val(arr(acount))
acount = acount + 1
     temp = "": Return
jend:
 End If

' SCALE COEFFICIENTS TO FRACTIONAL FORMAT
t1 = 0
For I = 1 To acount                                     'determine the highest integer
   If Abs(arr1(I)) > t1 Then
      t1 = Abs(arr1(I))
   End If
Next I
t2 = Abs(Int(t1))

If t2 > 0 Then                                          'normalize but avoid dividing by zero
  For I = 1 To acount
    arr1(I) = arr1(I) / (t2 + 1)
    arr1(I) = arr1(I) * 32767
  Next I
Else
```

```
  For I = 1 To acount
    arr1(I) = arr1(I) * 32767
Next I
End If

'**** Serial ouput to commport2
 Comm2.PortOpen = True
 Comm2.Output = Chr(27)
 For I = 1 To acount
    a1 = Int(arr1(I))
If a1 < 0 Then
 v2 = 65535 + a1                    'conversion of negative values to 1.15 format
 v3 = Int(v2 / 256)                 'calc LSB's
 v5 = v2 - (v3 * 256)               'calc MSB's
Else
 v3 = Int(a1 / 256)
 v5 = a1 - (v3 * 256)
End If
Comm2.Output = Chr(v3)              ' output LSB`s
For n = 1 To 100
 For K = 1 To 35
 Next K
 Next n

 Comm2.Output = Chr(v5)             ' output MSB`s
 For n = 1 To 100
 For K = 1 To 35
 Next K
 Next n
  Next I
  Comm2.PortOpen = False            ' Close Port
```

/----------

/ Addition to declaration section of DSP program

/----------

```
.const          taps=9;
.const          coeftaps=18;        /* total number of coefficients to bo download*/
.var/dm          freq;
.var/dm          amplitude;
.var/dm          FreqTab[7];
.var/pm/ram/circ     coeftab[18];   /*Filter coefficient table*/
.var/pm/ram/circ     ffcoef[9];     /*Feedforward Filter coefficient table*/
.var/pm/ram/circ     fbcoef[9];     /*Feedback Filter coefficient table*/
.var/dm          dlflag;            /*Down filter coef. flag*/
.var/dm          cntra;
.var/dm          cntrcoef;
.var/dm          fcntr;
.var/dm          tempstr1;
.var/dm          sflag;
.var/dm          fflag;             /* flag filtering */
.var/dm          startemp;
.var/dm/circ      data[taps];
i6 = ^coeftab;
l6 = 0;m6=1;
```