REAL-TIME IMPLEMENTATION OF LOW BIT-RATE SPEECH CODERS FOR SATELLITE LAND-MOBILE COMMUNICATION SYSTEMS

Thesis submitted to the University of Surrey
for
the degree of Doctor of Philosophy

Hosein G. Asjadi
Department of Electronics and Electrical Engineering
University of Surrey
Guildford
U.K.

July 1990
Summary

The fantastic growth in communication systems points to the fact that human beings need to exchange information in order to achieve better social, cultural and technical developments. Voice communication remains the dominant part of nearly all emerging communication networks. The remarkable progress in both the development of digital signal processing techniques and the associated VLSI technology has made analogue telephony redundant. Low bit-rate digital coding of voice has become of paramount importance in accommodating the tremendous increases in the number of users in the presence of constraints of bandwidth and power in such areas as cellular radio or satellite links.

In this research, we have been concerned with the real-time implementation of speech coders in the range of 13 to 4.8 kbit/s. When this work started, the world's first floating-point processor, AT&T WE-DSP32, had just become commercially available. We may claim that we were amongst the first users of this chip and its enhanced version, DSP32C. During the course of this work, three coders at 13, 9.6 and 4.8 kbit/s were successfully implemented in real-time. The coders were developed for use in three different communication systems covering the fields of maritime (INMARSAT-M), satellite (VSAT), land-mobile (GSM) telephony. It is the aim of this thesis to report on the software and hardware developments of the three coders, together with considerations on their performances.

Although, the major emphasis of the thesis is placed on the software and hardware implementation of the three coders, some of the more general aspects of real-time speech coding are put in perspective. The issue of fixed and floating-point implementation together with a review of a selection of DSP chips in both categories are presented. This study concludes that the more rapid progress of floating-point DSPs and their superior programming flexibilities and precision will probably render them the best choice in the future.
Acknowledgements

I am indebted to my supervisor Professor B. G. Evans for his continuous guidance, friendship and help during the course of this research. Indeed, he has been more than a supervisor to me.

I would like to thank all my colleagues in the Speech Processing Research Group for their friendship and help. Thanks are also due to the Department of Electronics and Electrical Engineering in the University of Surrey for supporting me financially.

My deepest and most sincere thanks are due to my 'Super' parents, who had to make many sacrifices, for me to reach this level of education. They have provided continuous support, encouragement and endless love. As a minute token of appreciation, I would like to dedicate this work to them.
Contents

Summary i

Acknowledgements ii

Chapter 1 Introduction 1
References 6

Chapter 2 Speech Coders For Satellite Land-Mobile Communication Systems 7

2.1 Introduction 7
2.2 Analogue vs Digital Speech Coding 7
2.3 Application Areas of Digital Speech Coding 9
  2.3.1 Public Switched Telephone Network (PSTN) 9
  2.3.2 Digital Satellite Communication 9
  2.3.3 Portable and Mobile Communications 10
  2.3.4 Aeronautical Telephone Services 11
2.4 Requirements of Digital Speech Coders 11
  2.4.1 Speech Quality 12
  2.4.2 Robustness 13
  2.4.3 Delay 13
2.5 Concluding Remarks 13
References 16

Chapter 3 Review of Low Bit Rate Speech Coding Schemes 18

3.1 Introduction 18
3.2 Frequency Domain Coding 20
  3.2.1 Adaptive Transform Coding (ATC) 21
  3.2.2 Sub-Band Coding (SBC) 23
  3.2.3 Harmonic Coding (HC) 25
  3.2.4 Sinusoidal Transform Coding (STC) 27
3.3 Time Domain Coding 29
   3.3.1 Adaptive Predictive Coding (APC) 32
   3.3.2 Residual Excited Linear Predictive Coding (RELP) 34
   3.3.3 Analysis-By-Synthesis Linear Predictive Coding 37
       Multipulse Excited Linear Predictive Coding 37
       Codebook Excitation Linear Predictive Coding 41
       Backward Excitation Recovery LPC Coding 44
3.4 Concluding Remarks 45
References 47

Chapter 4 Real-Time Speech Coding 58

4.1 Introduction 58
4.2 Implementation Strategies 59
4.3 DSP Implementation of Speech Coders 60
   4.3.1 Fixed-Point Implementation 62
   4.3.2 Floating-Point Implementation 65
   4.3.3 Real-Time Software Development 69
4.4 AT&T WE-DSP32 Floating Point Processor 71
   4.4.1 Architecture 71
   4.4.2 Software Development 74
4.5 AT&T WE-DSP32C Floating Point Processor 74
4.6 Concluding Remarks 77
References 79

Chapter 5 Speech Coder For Pan-European Digital Mobile Radio System 83

5.1 Introduction 83
5.2 Overview of the System 85
   5.2.1 The Access Scheme 85
   5.2.2 Speech Coding, Error Protection and Interleaving 87
   5.2.3 Network Organization 88
5.3 Speech Coding Algorithm 90
   5.3.1 Algorithm Description of Original Coder 93
   5.3.2 Algorithm Description of Compromise Coder 96
5.4 Software Implementation of RPE-LTP Encoder 101
   5.4.1 Dual Input and Output Buffering System 103
5.4.2 Pre-Processing of Input Speech 103
5.4.3 Autocorrelation Function 106
5.4.4 Schur Recursion 106
5.4.5 Log.-Area-Ratio (LAR) Transformation 106
5.4.6 Short-Term Inverse Filter 109
5.4.7 LTP Analysis 110
5.4.8 Quantization of RPE Sequence 110
5.4.9 Bit Mapping 115
5.4.10 Computational Complexity and Memory Usage 117
5.5 Software Implementation of RPE-LTP Decoder 118
5.5.1 Bit Decoding 118
5.5.2 Bad Frame Correction 120
5.5.3 Short-Term Synthesis Filter 121
5.5.4 Post Processing 121
5.5.5 Computational Complexity and Memory Usage 121
5.6 Speech Coder Hardware 123
5.7 Voice Activity Detection (VAD) 125
5.7.1 Description of Preliminary VAD Algorithm 126
5.7.2 Software Implementation of VAD 127
5.8 Concluding Remarks 130
References 132

Chapter 6 Speech Coder For V-SAT Business 136
Satellite Communication System

6.1 Introduction 136
6.2 Overview of the System 138
6.2.1 Network Operation 138
6.2.2 Network Size 140
6.2.3 Advantages of MP-SDV System 140
6.3 Speech Coding Algorithm 141
6.3.1 Algorithm Description of Encoder (7 kbit/s) 143
6.3.2 Algorithm Description of Decoder (7 kbit/s) 145
6.4 Software Implementation of Encoder 147
6.4.1 Data Acquisition and Preprocessing 147
6.4.2 LPC Analysis 148
6.4.3 Base-Band Extraction 151
6.4.4 Long-Term Analysis 153
Chapter 7  Speech Coder For INMARSAT-M System

7.1  Introduction 188
7.2  Overview of the System 191
7.3  Speech Coding Algorithm 194
    7.3.1  Algorithm Description of the Encoder (4.8 kbit/s) 194
    7.3.2  Algorithm Description of the Decoder (4.8 kbit/s) 196
7.4  Software Implementation of Encoder 196
    7.4.1  Data Acquisition and Preprocessing 200
    7.4.2  LPC Analysis 201
    7.4.3  LTP Analysis 204
    7.4.4  Sequential Codebook Search 205
    7.4.5  Data Output and Post Processing 207

References 185
7.5 Software Implementation of Decoder 207
7.5.1 Data Acquisition 207
7.5.2 Frame Reconstruction 209
7.5.3 Post Filtering 211
7.5.4 Data Output and Postprocessing 213
7.6 Line Spectrum Frequency Transformation 213
7.6.1 LPC-To-LSF Transformation 214
7.6.2 LSF-To-LPC Transformation 217
7.6.3 LSF Error Detection and Correction 219
7.7 Voice Activity Detection 220
7.8 Computational Complexity and Memory Usage 222
7.9 Speech Coder Hardware 226
  7.9.1 The Processor Board 226
  7.9.2 Audio Input and Output Unit 229
  7.9.3 Digital Input and Output Unit 231
  7.9.4 Power Supply Budget 232
7.10 Coder Performance 232
7.11 Concluding Remarks 236
References 238

Chapter 8 Conclusions 241
References 247

Appendices 248

Appendix A Published Papers 248
Appendix B Sample Source Codes (Assembly Language) 266
Introduction

Speech is one of the basic and most essential capabilities possessed by human beings. It is used to communicate intentions and emotions, and it is speech that distinguishes humans from animals. In fact, most of man's cultural and technical developments are due to his ability to exchange information. Human's need to converse with one another has lead to the formation of an estimated 4000-5000 different languages [1]. The information exchange by human beings has never been restricted to near-distance. Far-distance communications even in early days were made possible by means of flashing mirrors and fire-smoke on hill-tops.

It is a general belief that the real breakthrough in voice communication technology came with the invention of the telephone by Alexander Graham Bell in 1876 [2,3]. This was the beginning of 'analogue telephony', as we know it today. The invention of Pulse Code Modulation (PCM) in 1938, changed the direction of research in the speech communication field and brought about the digital representation of speech signals, 'digital telephony'.

Digital communication of speech offers many advantages over the analogue systems. The remarkable progress in both the development of signal processing techniques and the VLSI technology has made a significant impact on the digital
encoding of voiceband signals. The digital representation of speech could be viewed as the application of 'Divide and Rule' philosophy to the continuous spectrum of the analogue signal, allowing the bit-by-bit control of the transmitted message. This makes it possible for the introduction of more elaborate techniques such as bit classifications (with respect to channel errors), forward error correction coding and encryption.

Linear predictive coding (LPC), introduced in 1967, is the basis of many successful low rate coding schemes in bringing the initial bit rate of 64 kbit/s (PCM coding) down to rates as low as a few hundred bits per second. Low bit-rate coding of voice is critical for accommodating more users on channels that have inherent limitations of bandwidth or power, such as in cellular radio or satellite links. It can lend flexibility in the design of the evolving integrated services digital network (ISDN), which will reduce communication signals - voice, graphics, video, or computer data - to the common denominator of binary digit sequences [4].

The digital speech coding field has undergone significant advances in the past decade. The evaluation and implementation of speech coders at rates as low as 2.4 kbit/s are no longer restricted to non real-time computer simulations. Small-size, low-power-consumption hardware are produced comprising single fixed- or floating-point programmable processing chips providing low data rate, full-duplex links. There still remains the question of selecting fixed- or floating-point implementations of the algorithms but the more rapid progress of floating-point DSPs and their superior programming flexibilities and precision will probably make them the best alternative in the future.

In this research, we have been concerned with the real-time implementation of speech coders in the range of 13 to 4.8 kbit/s. When this work started, the world's first floating-point processor (AT&T WE-DSP32) had just become commercially available. We could claim that we were amongst the first users of this chip and consequently had to face all the traditional problems associated with a new product. However, during the course of this work, three coders at 13, 9.6 and 4.8 kbit/s were successfully real-time implemented. The coders were developed for the use of three different communication systems covering the fields of maritime, satellite, land-mobile telephony.

In the next seven chapters, we report on our efforts in this respect together with considerations of the general aspects of digital speech coding schemes suitable for satellite, land-mobile communication systems.
Outline of Thesis

The transmission of voice signals can be performed in analogue or digital format and encompass both fixed and mobile systems. Chapter 2 aims to highlight the main differences of analogue and digital telephony and considers the application areas of digital speech coding both in the fixed and mobile communication systems. The proposed and implemented national and international speech coding standards of the public switched telephone network (PSTN), digital satellite communications, portable and mobile systems, and aeronautical telephone services are briefly discussed. In the final part of this chapter, the major parameters in the design of a coding scheme are studied, followed by the main conclusions drawn from this study.

Digital speech coding techniques have been the subject of much research in the past 25 years. Nowadays, PCM coding which was once the only acceptable form of voice coding, acts only as a front-end to many low bit rate speech coders (A/D and D/A conversions). Present day algorithms seek to exploit the intrinsic properties of speech signals, in order to achieve better compression and speech quality characteristics and hence higher efficiency. Chapter 3 reviews some of the modern, and more successful, techniques of digital speech coding. The algorithms are classified into time and frequency domain coding schemes. The brief description of each technique is followed by a report on the latest developments and research work in this area, avoiding any mathematical description of the algorithms. CELP coding which has dominated the medium to low bit rate range is studied in more detail.

Chapter 4 deals with the issue of real-time implementation of digital speech coders. It discusses the recent efforts made in this direction and assesses the currently available VLSI technology for such purposes. It is aimed to show that programmable DSPs, in particular floating-point processors, are the most suitable means of prototyping various speech coders. The main differences of fixed and floating-point DSPs are highlighted, followed by a review of the available DSP chips in both categories. The implementation strategies and the real-time software development cycle are considered. Finally, a detailed description of the architecture of a floating-point DSP, AT&T WE-DSP32 and its enhanced version, AT&T WE-DSP32C, selected for the purpose of implementing our speech coders is given.
The year 1991 will bring significant changes to the operation of mobile telephony in western Europe. The replacement of more than nine incompatible types of analogue cellular telephone systems with a single digital mobile radio system (DMR) will enable people travelling in Europe to make, and receive telephone calls (and data communications) with people anywhere in the 'telephone world'. In chapter 5, an overview of the major components of the new Pan-European DMR system is followed by the details of the theory and implementation of the chosen speech coding algorithms. As part of the British Telecom Research Labs (BTRL) contract, to set-up an independent 'Test-Bed', the speech coder was real-time implemented on the AT&T WE-DSP32 floating-point DSP. The implementation strategy and the problems encountered are reported. The theory and implementation of a preliminary VAD system is also reported.

With the remarkable advances in satellite technology and design of earth stations, the traditional role of satellites is now changing. There is tremendous drive towards satellite networks that provide direct user to user services. One of these services is via so called VSAT networks, capable of providing medium speech data channels to individual locations without the need to install dedicated terrestrial circuits. Early VSAT systems were designed exclusively for low data rate traffic. However, developing countries have seen them as a real alternative network for providing rural telephony. Thus, the inclusion of a voice channel has become of paramount importance. Chapter 6 will first present an overview of a VSAT network designed and built by Multipoint Communications, MP-SP2300. This is then followed by discussing the development and implementation of a robust, efficient and self-synchronous digital speech coder operating at 9.6 kbit/s, for integration into this network. It reports on the modifications and refinements that were made to the Base-Band CELP coder, a coding scheme developed in the Speech Research Group of University of Surrey, in order to meet the specifications. The real-time implementation of the coder using two DSP32 chips (one for encoding and the other for decoding) on a double extended Eurocard board is fully documented. Finally, performance of the coder is discussed.

The INMARSAT organization which traditionally provided only maritime communication services is now in the process of expanding into the provision of sea to air and land communication services. The new system, INMARSAT-M standard, is to provide communications quality voice, data and facsimile services via the PSTN network or via closed networks to land-mobile, land transportable and maritime users. Initially, as a proof of concept, INMARSAT required the development and implementation of a digital speech coder operating at 4.8 kbit/s. The coding scheme needed to be robust, produce acceptable speech quality and to be integrated into a compact and power efficient
hardware. Chapter 7 deals with the theory and real-time implementation of this coder using a single AT&T WE-DSP32C on a single Eurocard size board. A brief overview of the INMARSAT-M system is followed by a report on the modifications made to the CELP-BB coding scheme. We then consider the software and hardware implementations of the coder and finally, the performance of the coder is presented.

The main conclusions drawn from this research are listed in Chapter 8. The thoughts that were provoked during the course of this work which may be integrated into the design and implementation of future speech coders are also documented.

The work discussed in this thesis is original in four different respects:

(i) The Pan-European speech coding algorithm was real-time implemented on a floating-point processor. This is the first and only floating-point implementation of the algorithm. It is hoped to show through more extensive studies and future publications that our floating-point implementation outperforms the fixed-point representation of the coding scheme.

(ii) A completely new coding scheme was developed and real-time implemented, operating at 9.6 kbit/s. For the first time a speech coding function was successfully introduced into a V-SAT communication system. During the course of this particular study a new frame synchronization technique was also developed which offers a much faster acquisition time and robustness against channel errors.

(iii) The world's smallest hardware incorporating a single DSP32C (encoder and decoder), responsible for compressing speech signals into 4.8 kbit/s, was produced. We believe this is the first use of the DSP32C for the purpose of digital speech coding. The developed algorithm is unique in its robustness against channel errors and background-noise, making it an ideal candidate for mobile communication systems.

(iv) During the course of this research an extensive software library [5] for the purpose of real-time speech coding on the AT&T WE-DSP32 and DSP32C processors was generated. The software library is first of its kind and all the routines have been hand-optimized to the highest level possible. The software library will offer the future real-time programmers a means of developing numerous low-bit rate speech coders at a much faster time-span.
References


5. G.H. Asjadi, "Real-Time Speech Coding on AT&T WE-DSP32 and WE-DSP32C", To be published.
CHAPTER 2

SPEECH CODERS FOR SATELLITE LAND-MOBILE COMMUNICATION SYSTEMS

2.1 Introduction

The transmission of voice signals from one place to another can either be done in analogue or digital format. In this chapter, we will first investigate the reason behind the digital take-over in the past few years.

A digital telephony system can encompass both fixed and mobile systems. We will consider the application areas of digital speech coding both in the fixed and mobile systems. The requirements of a digital speech coder vary from one application area to another. In the final part of this chapter, the important parameters in designing a coding scheme are considered.

2.2 Analogue vs Digital Speech Coding

Until recently, signal processing has always been done using analogue equipments. The first use of digital signal processing (DSP) techniques were in the speech processing field, as simulations of complex analogue systems.[1] The introduction of the Fast Fourier Transform (FFT) was a large leap towards purely digital processing. The advance in both the theory and digital hardware is forcing more and
more digital processing techniques into telecommunication systems. With regard to transmission of speech signals, digital encoding offers many advantages over analogue [2-5];

- The transmission quality of speech signals is almost independent of distance and network topology. This is due to the fact that digital signals can be regenerated at repeaters placed along the path.

- Different services such as speech, data, video and facsimile data, when encoded to a uniform digital format, can be multiplexed together and sent over the same communication link.

- The adverse conditions in some of the communication channels are better accommodated in digital systems than in analogue. By introducing redundancy into the transmitted string, digital systems can recover the transmitted information even when the noise level exceeds the signal level. Digital coding of speech also offers the possibility of reconstructing lost or corrupted blocks of speech data due to shadowing or burst errors.

- Information in a digital format can easily be secured by scrambling and descrambling and hence providing a secure communication link.

- The DSP hardware is usually programmable to allow great flexibility and is not subject to component variability, such as tolerance differences or drift, or to non-ideal behaviour such as leakage, noise, or parasitic. The current rapid growth in Very Large Scale Integrated (VLSI) technology will eventually result in cheaper and very compact equipments.

- There is also the flexibility for complex signal processing operations such as tone generation, echo control, equalization and filtering.

Digital coding of speech has, however, some disadvantages. The finite word length of the data and coefficients limits the accuracy, the dynamic range, and the signal-to-noise ratio (SNR). Digital systems require sampling rates to be twice the highest frequency component of the signal to be sampled. This limits the bandwidth that can be used as input. DSP also requires the integration of theory, software and hardware. This makes good development tools a necessity.
2.3 Application Areas of Digital Speech Coding

Traditionally, speech processing is divided into three related sectors [6];

- Speech Coding: primarily concerns human-to-human voice communication.
- Speech Recognition: mainly human-to-machine communication.

Since we are only concerned with digital speech coding, in this section we reflect the current application areas of this technology in the communication systems.

2.3.1 Public Switched Telephone Network (PSTN)

The first introduction of digital telephony into the Public Switched Telephone Network (PSTN) was in 1972 with CCITT adopting the 64 kb/s Pulse Code Modulation (PCM) scheme as a worldwide standard. Having benefited from facilities that the digital system provided, CCITT then in 1985 adopted a 32 kb/s Adaptive Differential Pulse Code Modulation (ADPCM) coding scheme as the follow-on standard. Following the standardization of 32 kb/s ADPCM, CCITT is now planning to standardize a 16 kb/s speech coder by 1991. The two proposed coders are derivatives of Code-Excited Linear Prediction (CELP) coding scheme. The drive for efficient utilization of the available channel is mainly due to significant increase in the number of PSTN users and the need to integrate many services such as videotex and facsimile into a single network.

2.3.2 Digital Satellite Communication

Digital speech coding is an interesting and attractive topic for digital satellite communication systems, to meet growing demands on telephone services. Efficient encoding techniques can not only effectively expand the channel capacity but also efficiently use the satellite transmitting power, especially in maritime systems. The voice coding technique can also be incorporated into a Digital Speech Interpolation (DSI) system to offer economical digital channels in Time Division Multiple Access (TDMA) satellite communication systems.

Satellite systems have played an important role in improving today's international communications. Although, satellite communication between fixed earth stations is a mature technology after almost 25 years of Intelsat operations, mobile satellite systems are only an emerging technology at this stage [8]. It is anticipated that all these systems will rely on digital techniques as regard to speech communication.

Until very recently analogue schemes such as Amplitude-Companded Single-Sideband (ACSSB) modulation were most favoured but low bit rate digitally encoded speech will probably supersede analogue methods in the medium term. INMARSAT is currently designing a new voice grade satellite communications system, using small mobile terminals suitable for installations in land-mobile vehicles and small ships. It is very likely that they will adopt a low bit-rate (~ 6.4 kb/s) digital scheme for voice communications. Another use of digital speech coding is in the VSAT (Very Small Aperture Terminals) communication systems, where provisions are made for the use of a 9.6 kb/s speech coder.

2.3.3 Portable and Mobile Communications

The frontiers of mobile and portable communication technology and services have expanded rapidly in recent years, resulting in the introduction of new systems and new applications. This expansion has been accompanied by a steady increase in the number of mobile and portable radio users. While most current systems are based on analogue technology, the trend is towards digital techniques. The realization of a second generation cellular radio system will require the availability of high quality speech coding techniques operating at low bit rates [9]. In 1991, a new Pan-European Digital Mobile Cellular Telephone system will be opened for services simultaneously in 16 European countries.

Another interesting area under development is short-range indoor office communication systems using small pocket telephones connected to the office PABX. There is also development towards personal communications. It is predicted that all these new systems will rely on digital techniques for voice transmission [10].
2.3.4 Aeronautical Telephone Services

Until recently, the aircraft remained the only poorly serviced transporter. With INMARSAT and many other international bodies providing communication between ships at sea, car and trucks on the road. With an estimated 1000 wide-bodied jets in operation, several million busy executives traverse the world every year. For these executives, accustomed to mobile telephone and constant contact with their offices, it is unacceptable to be out of touch for the duration of long international flights [11, 12]. Flight crew have also a need for reliable voice communications with Air Traffic Control and their operation centres during transoceanic and intercontinental flights. In February 1989, a trial service was started to provide telephony via satellite to an aircraft. The service is called Skyphone service and uses a 9.6 kb/s speech codec because of the limited satellite bandwidth available. The trial service will soon be replaced by an operational service and by ways of an agreement between British Telecom and the telecommunications administrations of Norway and Singapore will provide a worldwide telephony service between flights over the pacific, Indian and Atlantic oceans [11].

2.4 Requirements of Digital Speech Coders

The requirements of digital speech coders are very much application dependent. A coder designed for military type of communications is not suitable for use within a PSTN system. The former requires a very robust speech coder (both in terms of background noise and channel errors) and is able to tolerate a moderate speech quality. These conditions are totally reversed for the latter. The delay of digital speech coders designed for PSTN network should not exceed 5 ms whereas the coders designed for a satellite link may have a back-to-back delay of 90 ms. In general, in a telecommunication environment a speech codec must satisfy the following:

- be able to operate under a wide range of adverse conditions,
- produce good speech quality,
- low transmission delay,
- low bit rate,
- cheap to manufacture,
- and consume little power.
In the following section the important factors that determine the design of a digital speech coder are considered.

2.4.1 Speech Quality

The speech quality is the most important design parameter of a coding scheme. It is usually either measured objectively and presented in terms of signal-to-noise ratio (SNR) or measured subjectively and presented in terms of Mean Opinion Score (MOS). There are other evaluation schemes such as Diagnostic Rhyme Test (DRT) for word intelligibility and Diagnostic Acceptability Measure (DAM). The current evaluation techniques were originally developed for quality measurement of high bit-rate coders and there is a need for new methods of evaluating low bit-rate coders. Speech quality, bit-rate and complexity are nearly inseparable factors in the design of a coder. Figure 2.1 shows the speech quality as a function of bit rate [13]. In order to reduce the bit rate, it is necessary to expect a drop in the speech quality with an increase in the computational complexity. The speech quality requirements vary from one application to the other with the PSTN usually requiring the highest speech quality at a given bit rate.

![Fig. 2.1 Quality, Bit Rate and Complexity of Voiceband Speech Coders [13]](image)
2.4.2 Robustness

The coding scheme needs to be robust both against environmental noise and transmission errors. The effects of environmental noise or background noise needs more attention in mobile systems to prevent noise such as aircraft or truck engine to "break-up" the coder. The effects of transmission errors on the speech quality can be very severe. The transmission errors occur either randomly, in bursts or a combination of both. To minimize the effects of transmission errors, Forward Error Correction (FEC) codes can be added to the raw data to detect and consequently correct the erroneous data. This, however, increases both the transmission capacity and algorithm complexity as well as in some cases introduces an extra delay. An efficient coding scheme is one which relies on built-in error detection and correction.

2.4.3 Delay

The total delay of a communication system is a combination of coding delay and the transmission delay. The coding delay is algorithm dependent. The CCITT specifies the maximum delay for different communication systems with PSTN requiring the smallest and the satellite systems being allowed the largest delay. In systems where there is a large delay, it is vital to incorporate echo cancellation schemes.

It is also important to consider the effects of other parameters such as transcodings, tandeming, speakers dynamic range and voice-band data handling.

2.5 Concluding Remarks

The advantages of digital communication systems over the analogue counterpart are numerous. The most attractive aspect of a digital system is the flexibility of multiplexing together of different services such as speech, data, video and facsimile data over the same communication link, providing a 'Total Communication Link'. In fact, all the future communication systems are being designed with this purpose in mind.
The signal processing techniques as applied to digital voice communications are many and varied, and are largely dependent on the specific application area. In one extreme, are the coders designed for use within a PSTN system where good quality speech and low delay are prime concerns. In the other extreme, are the military type of speech coders where robustness against background noise and channel errors are the important design parameters. Between the two extremes, there is a need for coders with moderate amounts of robustness and speech quality with low delay and small manufacturing costs.

The performance of a speech coder is usually evaluated by both objective and subjective quality measurements. There is a general feeling amongst the researchers in this area that the present performance measurements, both objectively and subjectively, do not suit the low-bit rate speech coders and alternative means need to be explored.

Finally, table 2.1 shows a selection of national and international standards that currently exist. It is clear that, presently, Code-Excited Linear Predictive Coding (CELP) is the dominant coding scheme in a number of different application areas.
<table>
<thead>
<tr>
<th>Rate (kbit/s)</th>
<th>Application</th>
<th>Type of Coder</th>
<th>Year of Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>PSTN (1st Generation)</td>
<td>Pulse Code Modulation (PCM)</td>
<td>1972</td>
</tr>
<tr>
<td>16</td>
<td>PSTN (3rd Generation)</td>
<td>Code-Excited Linear Predictive Coding (CELP)</td>
<td>1991</td>
</tr>
<tr>
<td></td>
<td>INMARSAT Standard-B (Maritime)</td>
<td>Adaptive Predictive Coding (APC)</td>
<td>1985</td>
</tr>
<tr>
<td>9.6</td>
<td>Aeronautical Telephone Services</td>
<td>Multipulse Excited - Linear Predictive Coding (MPE-LPC)</td>
<td>1989 (Trial Services)</td>
</tr>
<tr>
<td>6.4</td>
<td>INMARSAT Standard-M (Land-Mobile)</td>
<td>CELP</td>
<td>1990's</td>
</tr>
<tr>
<td>4.8</td>
<td>U.S. Government Federal Standard</td>
<td>CELP</td>
<td>1990's</td>
</tr>
<tr>
<td>2.4</td>
<td>U.S. Government Federal Standard</td>
<td>LPC-10</td>
<td>1977</td>
</tr>
</tbody>
</table>

Table 2.1 Digital Speech Coding Standards
References


CHAPTER 3

Review Of Low Bit Rate Speech Coding Schemes

3.1 Introduction

Digital speech coding techniques have come a long way since the days of direct quantization (over 25 years ago), using Pulse Code Modulation (PCM). Present day algorithms seek to exploit the intrinsic properties of speech signals, in order to achieve better compression and speech quality characteristics and hence higher efficiency. Researchers have been striving to introduce more efficient methods of converting 4 kHz voiceband signals into digital form for transmission or storage reduction in voice messaging systems.

Studies into complex and potentially efficient speech coding algorithms have been encouraged over the last decade in both academy and Industry, as a result of the following two interrelated factors [1]:

(i) the introduction of new applications for the transmission and storage of digital speech (such as satellite/mobile communication services and voice storage and forward systems) where efficient coders are required, and

(ii) the rapid advances in VLSI/DSP technology which have made possible the real-time implementation of relatively sophisticated speech coding algorithms. Already, medium bit rate (16 to 8 kbit/sec) speech coders, capable of producing toll/communication quality speech, can be implemented on a single DSP device.
LOW BIT RATE SPEECH CODING

FREQUENCY DOMAIN
- ADAPTIVE TRANSFORM CODING (ATC)
- SUBBAND CODING (SBC)
- HARMONIC CODING (HC)
- SINUSOIDAL TRANSFORM CODING (STC)

TIME DOMAIN
- ANALYSIS AND SYNTHESIS
- ADAPTIVE PREDICTIVE CODING (APC)
- RESIDUAL EXCITED LPC CODING (RELPC)
- MULTIPULSE EXCITATION LPC CODING (MPE-LPC)
- CODEBOOK EXCITATION LPC CODING (CELPC)
- BACKWARD EXCITATION RECOVERY LPC CODING (BER-LPC)

Fig. 3.1 Modern Low Bit Rate Speech Coding Techniques
The speech coding technology to achieve high voice quality is well developed for bit rates as low as 16 kb/s. Today, the major research activity is focused at bringing the rate down to 4.8 kb/s and lower without degrading speech quality. The redundancies introduced in the speech signal during the human production process, makes it possible to encode speech at low bit rates. Moreover, our hearing system is not equally sensitive to distortions at different frequencies and has a limited dynamic range. Speech coding techniques take advantage of both speech production and perception to reduce the bit rate.

Modern speech coding algorithms are based on models which attempt to successfully describe short-term speech signals. These models can be defined in the "Time" or in the "Frequency" domain and analysis-by-synthesis (ABS) estimation procedures are often used to determine the parameters of the short-term model.

In this chapter, some of the coding techniques for achieving low bit rate, both in the time and frequency domains, are reviewed. It is assumed that the reader is familiar with the fundamental aspects of speech processing. For information on the basic principles of speech production and terminology, the reader should refer to references [2-6].

3.2 Frequency Domain Coding

In frequency domain modelling, the approach is to divide the speech signal into a number of separate frequency components by a filter bank (sub-band coding), or by a suitable transform (transform coding), and then to encode each of these components separately. The frequency domain coding techniques have the advantage that the number of bits used to encode each frequency component can be variable, so that the encoding accuracy is always placed where it is needed. In the lower frequency bands, where pitch and formant structure must be accurately preserved, a large number of bits/sample can be used; whereas in upper frequency bands, where fricative and noise-like sounds occur in speech, fewer bits/sample can be used. Further, quantization noise can be contained within bands to prevent masking of a low-level input in one frequency range by quantization noise in another frequency range. Three basic factors are involved in the design of these coders: (i) the type of the filter bank or transform, (ii) the choice of bit allocation and noise shaping properties involved in bit allocation and (iii) the control of the step-size of the coders [3, 7].
In this section, some of the frequency domain coding techniques that are used to achieve medium to low bit rate coding, are briefly reviewed.

### 3.2.1 Adaptive Transform Coding (ATC)

Adaptive Transform coders derive a spectral representation of the short-term signal and quantize the spectral coefficients using dynamic bit allocation. At the decoder an inverse transformation procedure takes the received coefficients and generates the recovered speech signal. The dynamic bit allocation is usually based on an estimate of the spectral envelope of the signal and ensures that the signal to noise ratio is maximized. The bit allocation can also be adjusted to provide a "perceptually" optimized spectral distribution for the coding distortion present in the recovered signal and thus an improved coding performance [8, 9].

The simplest form of ATC is that proposed by Zelinsky and Noll. Figure 3.2 shows the block diagram of this coder. A block of N speech samples is first normalized by its estimated standard deviation and then transformed into a set of frequency domain coefficients via an N-point Discrete Cosine Transform (DCT). A coarse description of the cosine basis spectrum is extracted and transmitted to receiver as side information. This coarse spectral estimate (quantized) is used at both transmitter and receiver to calculate the optimum assignment of bits and the quantizer step sizes for coding the coefficients. The spectral estimate consists of a small number of samples computed by averaging the DCT spectral magnitudes. These samples are then geometrically interpolated to yield the expected spectral levels at all frequencies used for determining the quantizer parameters.

ATC offers near toll quality at 16 kb/s but its performance deteriorates rapidly below 10 kb/s. This is due to the fact that as the bit rate reduces the number of available bits for spectrum representation reduces. This results into zero bit allocation at higher frequencies, known as the "low pass effect" that significantly degrades the speech quality of ATC. The "Vocoder driven" ATC [10] overcomes this problem and operates satisfactorily in the range 8 to 16 kb/s. The algorithm is based on an all-pole model of speech for the envelope reproduction and a pitch model to represent the fine structure, that is used to "steer" the ATC to its adaptive bit allocation process. Due to the gaps between the pitch peaks in the frequency domain more zero bits are allocated to the lower frequencies releasing bits to be allocated at the higher frequencies [11].
Fig. 3.2 Block Diagram of an Adaptive Transform Coder
In order to reduce the bit-rate efficiently, hybrid transform coders were developed [12-14]. These schemes combine ATC, RELP and vector quantization techniques to improve the quality of low bit-rate transform coders. The short and long-term correlations are removed from the input block of speech and then the remaining residual signal is frequency transformed. Vector quantization may also be used to quantize the LPC coefficients and the residual frequency coefficients.

3.2.2 Sub-Band Coding (SBC)

Subband coding (SBC) of speech is a relatively mature form of waveform coding of speech. The speech is first sub-divided into a number of subbands (2 to 16) by a bank of bandpass filters, which are then individually encoded. The underlying principle for the coder is that the bit allocation can be weighted so that those subbands with the most important information get the most bits. The advantage of subband coding may be viewed from several different angles. The most common explanation focuses on the perceptual merits of this technique. Since the human auditory mechanism responds differently to coding noise in different spectral regions, it is clearly advantageous to be able to control the spectral shape of the noise. This is achievable by coding the speech in subbands. Also, confining the coding noise that is generated in a certain region of the spectrum to that region activates the auditory masking effect which masks the noise and makes it less noticeable. Another aspect of this scheme is more fundamental from a data compression point of view. As in transform coding, this method transforms the speech signal into a new domain in which the structure of the signal is maintained by a generally unequal energy pattern of the different bands. This energy pattern is used for efficiently controlling the allocation of the available bit resources [15].

Figure 3.3 illustrates a basic block diagram of the subband coder. The most complex part of the coder is the filter bank implementation. The two most popular scheme of splitting the bands are (i) integer band sampling [16] and (ii) Quadrature Mirror Filter (QMF) [17]. The coder consists of M bandpass filters, followed by subband encoders which typically comprise different versions of PCM coding and a multiplexer. The receiver has the inverse stages of demultiplexing, decoding and bandpass filtering prior to subband addition. The allocation of bits for coding each subband may be fixed or adaptive. The non-flat spectral density of speech signals is exploited to apply unequal quantization to the frequency bands. In two early fixed bit allocation designs [17, 18], one uses the backward adaptive Jayant quantizer and the other block quantization with forward
Fig. 3.3 Block Diagram of a Subband Coder
transmission of step sizes. Better coding efficiency is achieved by allowing the number of bits assigned to each frequency band to vary according to local signal statistics. Adaptive or dynamic techniques of bit allocation [19, 20] attempt to distribute available bits more efficiently by assigning bits to the subbands according to their energy composition over a short segment of speech (10-30 ms). This, however, means the transmission of side information periodically so that the receiver is kept informed of the update in the allocation patterns.

Subband coders produce near toll quality speech at 16 kbit/s, whereas at 9.6 kbit/s the quality is reduced as the "effective bandwidth" of the recovered signal is significantly decreased. At a bit rate of 4.8 kbit/s, speech of communication quality (acceptable to certain applications) can be obtained by using sophisticated time domain coding for the subband signals and high frequency regeneration of "inactive" frequency bands [21].

3.2.3 Harmonic Coding (HC)

The major difference between basic ATC coding and Harmonic Coding (HC) is that a residual spectrum is encoded only after subtracting a simplified spectrum from the full speech spectrum [22-24]. Figure 3.4 illustrates a basic block diagram of the Harmonic Coder. As in ATC, a block of N speech samples is transformed and analyzed with the aid of a pitch estimator to identify the harmonics, which are then modeled as windowed sinusoids. Optimal amplitudes and phases for these sinusoids are chosen to minimize the energy in the difference between the spectrum of the original speech and that of the harmonic model. These parameters are then transmitted, along with the pitch estimate, to the receiver as side information. Using the coded versions of the harmonic amplitudes and phases, an "estimate spectrum" is reconstructed and subtracted from the original speech spectrum. The resulting spectrum is then encoded using ATC techniques. The receiver constructs the same "estimate spectrum" from the transmitted side information, adds that to the decode residual spectrum, and inverse-transforms the sum back into a block of speech samples.

In some low bit rate (4.8 kb/s) implementation of harmonic coding schemes [22, 25], linear predictive coding is used instead of the DFT or DCT for spectral analysis. The residual spectrum is obtained by inverse LPC filtering and a subsequent DFT. This scheme achieves good communication quality speech, with some tonal change and
Fig. 3.4 Block Diagram of a Harmonic Coder [22]
harshness as well as slight buzziness for male speech. The harshness can be reduced by allocating more bits to code the harmonic phases. The buzziness is a result of harmonics receiving no coding bits, of which there are more for male voices since male voices have more harmonics to code [6].

In a recent study [26], vector quantization was introduced into the harmonic coder for quantizing the harmonic parameters. This made it possible to quantize voiced speech below 8 kb/s, and still obtain high quality synthetic signals. In general, to improve the coding performance at 6 and 4.8 kb/s, modifications to the basic HC model and dynamic quantization strategies are needed.

3.2.4 Sinusoidal Transform Coding (STC)

Sinusoidal Transform Coding (STC) is an alternative approach to the problem of representing speech signals. The most common approach is to use the speech production model [27] in which speech is viewed as the result of passing a glottal excitation waveform through a time-varying linear filter that models the resonant characteristics of the vocal tract. The aim of the STC system is to generalize the model for the glottal excitation by assuming that the excitation waveform is composed of sinusoidal components of arbitrary amplitudes, frequencies, and phases. These parameters are estimated from the short-time Fourier transform using a simple peak-picking algorithm. Rapid changes in the highly resolved spectral components are tracked using the concept of "birth" and "death" of the underlying sine waves. For a given frequency track a cubic function is used to unwrap and interpolate the phase such that the phase track is maximally smooth. This phase function is applied to a sinewave generator, which is amplitude modulated and added to other sinewaves to give the final speech output (see figure 3.5). The resulting synthetic waveform preserves the general waveform shape and is essentially perceptually indistinguishable from the original speech. In addition, the STC system performs well in the presence of environmental disturbances due to noise, multiple speakers, or music, and could be used to successfully reproduce certain marine biological sounds [28-30].

It has been demonstrated that speech of high quality can be generated using a synthesizer based on an harmonic model for the sinewave frequencies, and a pitch-onset, voicing dependent model for the sinewave phases [31, 32]. In a recent evaluation using a non-real-time floating point harmonic sinewave system operating at 10 ms per frame, a
PHASES

INVERSE TAN(·)

INPUT TO DFT

FREQUENCIES

CHANNELSPEECH

PEAK PICKING

AMPLITUDES

MULTIPLEXER

WINDOW

INPUT

SPEECH

X

DFT

1.1

Fig. 3.5 Block Diagram of the Sinusoidal Analysis and Synthesis System [28]
Diagnostic Acceptability Measure (DAM) of 63.0 was achieved. Since the excitation for this system consists of only the pitch (7 bits) and voicing (3 bits), the possibility of achieving high-quality speech at low rates (1.2 - 2.4 kb/s) appears to be feasible provided efficient techniques can be developed for coding the sinewave amplitudes. Many different methods of efficiently encoding the amplitude data have been explored; (i) DPCM , (ii) DPCM on the down sampled envelope, (iii) DPCM with positive slope overload protection , (iv) DPCM with dynamic programming, and (v) all-pole modelling [23]. Another approach for achieving low data rates is to employ the zero-phase harmonic synthesis model [34] in which the sinewave amplitudes are obtained by sampling an envelope at the harmonics of the estimated pitch. An onset time is obtained from the sequence of pitch measurements that determine the time at which all of the sinewaves come into phase. Unvoiced speech is obtained by adding a uniformly distributed variable on (-7C , n) to the sinewave phases above a frequency cutoff that is dependent on the likelihood that the speech is voiced.

In general, STC coders can produce good communication quality speech at 8 kb/s and can be modified (using Harmonic frequency models) to operate at 4.8 and 2.4 kb/s [32, 33, 35]. An extension of STC is the Analysis-by-Synthesis Sinusoidal Coder [36] which uses a polynomial representation of the time evolution of the amplitudes and phases of the sinusoidal components. The polynomial coefficients are determined by minimizing the energy of an error signal formed as the difference between the input and the decoded speech signal. As a closed form solution is not available, an analysis-by-synthesis procedure is used to minimize the energy of the error signal. The minimization procedure is appropriately constrained to produce "smooth" parameter tracks and signal continuity at the frame boundaries.

3.3 Time Domain Coding

Most of the time domain coding schemes are based on a simplified model of human speech production. In this model, speech is broadly classified as either voiced or unvoiced. Voiced speech (e.g. the vowel sound /a/) is produced when air from the lungs is forced through the vocal cords causing them to vibrate. The vocal cords are two flaps of soft tissues which are held under tension for the production of voiced speech. The frequency of vibration of the vocal cords is dependent on the air pressure in the trachea and the degree of tension. The frequency of vibration is known as the fundamental
frequency and the perceived frequency is known as pitch. The pulses of air emitted from the vocal cords excite the vocal tract giving rise to resonant frequencies in the radiated signal. For unvoiced speech the vocal cords are loose and air from the lungs passes unaffected into the vocal tract. In this case the excitation occurs by placing a constriction at some point along the vocal tract. A model of this speech production process which has been widely used is illustrated in figure 3.6. Voiced excitation is modelled by a train of unipolar, unit amplitude impulses at the desired pitch frequency. Unvoiced excitation is modelled as the output from a pseudo-random noise generator; the voiced/unvoiced switch selects the appropriate excitation [37].

![Fig. 3.6 Simplified Model of Speech Production](image)

Based on this model, low bit rate predictive coding of speech usually uses two nonrecursive prediction error filters to process the input signal before coding. The prediction operations are motivated by the fact that the input speech exhibits a high degree of intersample correlations. These correlations occur between adjacent samples (near-sample redundancy) and for voiced speech, between samples separated by the pitch period (distant-sample redundancy). Near-sample redundancies (or short-term correlation) can be attributed to the filtering action of the vocal tract. The resonances of the vocal tract corresponds to the formant frequencies in speech. Far-sample redundancies (or long-term correlations) can be attributed to the pitch excitation of voiced speech. Two filters, the formant and pitch predictors (see figure 3.7) are used to remove the near-sample and far-sample redundancies, respectively. The resulting prediction-residual signal is of small amplitude and can be coded more efficiently than the original speech waveform. The predictor coefficients are adapted by updating them at fixed
Fig. 3.7 Basic Structure of Time Domain Coding (a) Analysis  (b) Synthesis
intervals to follow the time-varying correlations of the speech signal [38]. In fact, the way in which the remaining residual signal is quantized distinguishes different time coding schemes from one another. Most time domain decoders have a common structure as shown in figure 3.7 (b).

Efficient removal of long-term correlations is essential in faithful reproduction of the speech signal and numerous methods [39] of pitch detection exist. Recently, a more complex but efficient pitch prediction scheme has been reported [40].

In this section, some of the time domain coding techniques that are employed to achieve low bit rate coding, are reviewed.

3.3.1 Adaptive Predictive Coding (APC)

Adaptive Predictive Coding (APC) [41, 42] is one of the early and most successful, differential time domain speech coding systems for bit rates below 16 kbit/s. A simplified block diagram of an APC system is shown in figure 3.8. In conventional APC, two predictors are placed in a feedback loop around the quantizer. The redundancies of the speech signal are removed in two stages by using short-term (formant) and long-term (pitch) prediction filters. The difference signal between the input speech and its predicted value (residual signal) is quantized, encoded and transmitted to the receiver. In the synthesis phase, the excitation signal (the coded residual) is passed through a pitch synthesis and a formant synthesis filter to produce the decoded speech. The synthesis operation can be viewed in the frequency domain as first inserting the periodic structure due to pitch and then inserting the spectral envelope (formant structure). Due to non-stationarities of the speech signal, both of the predictors are made adaptive varying on a block basis.

With this configuration, it can be shown that the quantization noise is not only due to the difference between the residual and its quantized value but also the difference between the original speech signal and its reconstructed value. The perceptual distortion of the output speech can be reduced by adding a noise shaping filter which redistributes the quantization noise spectrum [43, 44]. The noise shaping filter increases the noise energy in the formant regions but decreases the noise power at frequencies in which the energy level is low [38].
Fig. 3.8 Block Diagram of an Adaptive Predictive Coder (APC) with Noise Shaping [45]
The performance of long-term prediction and efficient quantization of the residual signal have a strong influence on the overall quality of speech in the APC system. The APC coder is capable of producing high quality speech (equivalent to 7-bit log-PCM) at 16 kbit/s [43]. In this system, a third order pitch predictor is used with a parameter update rate of 10 ms. By using a "centre clipping" quantizer [46] the bit rate of the system can be reduced to 10 kbit/s with very little distortion (12 dB segmental SNR) in the decoded speech [42]. The quantization of the residual signal can be pitch adaptive [47]. In pitch adaptive quantization, the number of bits per sample are increased with large residual pitch pulses. Information regarding the position of these large amplitude regions must be transmitted.

Another APC system, which can be regarded as a Time-Frequency hybrid coder, was developed to produce high speech quality at 16 kbit/s (equivalent to 7-bit log-PCM) and 9.6 kbit/s (equivalent to 6-bit log-PCM) [48]. In this system, a split-band predictive coding scheme and a bit allocation scheme are employed to remove redundancies due to the periodic conservation of the residual energy as well as the non uniform nature of the spectrum. By using the QMF technique the speech signal is split into a number of subbands and then APC is applied to encode each subband. Quantization bits are allocated both at the subbands (frequency domain) and the subintervals (time domain) according to the distribution of energies in time-frequency domains. The major disadvantage of this system is its excessive complexity.

In order to improve the quantization of the difference signal (residual) at lower bit rates, Delayed Decision Coding [49] or Vector Quantization [50] is applied. In VAPC the scalar quantizer in APC is replaced by a vector quantizer. The motivation for using vector quantizer (VQ) is two fold. First, although linear dependency between adjacent speech samples is essentially removed by linear prediction, adjacent prediction residual samples may still have nonlinear dependency which can be exploited by VQ. Secondly, VQ can operate at rates below one bit per sample. This is not achievable by scalar quantization, but it is essential for speech coding at low bit rates [51].

### 3.3.2 Residual Excited Linear Predictive Coding (RELP)

Residual Excited Linear Predictive Coding (RELP) is known as a variant of APC coding [52]. A block diagram of a basic RELP coder is shown in figure 3.9. The fact that the lowest speech frequencies carry the highest perceptual importance is the
Fig. 3.9 Block Diagram of a Residual Excited Linear Predictive Coder (RELP)
fundamental aspect of this system. The baseband portion of the speech signal (typically 0 - 800 Hz) provides a major contribution to naturalness of speech and contains most of the energies in voiced sounds. In RELP coding, standard methods of LPC analysis provides the spectral coefficients, which need to be transmitted as side information and also used to inverse filter the speech signal to obtain the residual. In some cases, long-term prediction is also employed at this stage to remove the far-sample redundancies of the speech signal. A baseband of the residual is then extracted by a lowpass filter, decimated and waveform coded. The bandwidth of the lowpass filter (which depends on the decimation factor) should cover the whole range of the fundamental frequency of speech. The ratio of the fullband to baseband signal determines the decimation factor. Forward adaptive quantizer or Pitch-Predictive DPCM (PP-DPCM) are usually used to encode the decimated signal. The RELP decoder interpolates the residual back into the original sampling rate and attempts to reconstruct the fullband residual through nonlinear schemes. This is known as High Frequency Regeneration (HFR) and the aim is to generate the high frequencies such that the fullband signal has flat spectrum and fine structure for the voiced sounds. It could be said that most of the distortions associated with RELP coding are introduced at this stage and therefore much effort has been spent in finding an efficient HFR method [53]. Simple nonlinear functions such as squaring or rectification are used to generate high frequencies from low frequencies but do not yield a flat spectrum. More complex methods include spectral folding and spectral translation, in which the baseband spectrum is repeatedly copied into the high frequency range [54].

Improved RELP systems capable of producing communication quality speech at rates 4.8 - 9.6 kbit/s, have been proposed, which employ either pitch-aligned HFR techniques or fullband pitch prediction in the time domain, to remove the pitch information from the residual signal prior to band limitation and decimation [55-58].

Recently, a speech coder operating at 13 kbit/s has been adapted as a standard for the first generation of digital mobile radio system operating in Europe [59]. The chosen coder, Regular Pulse Excitation with Long Term Prediction (RPE-LTP) is a hybrid of RELP and Multipulse Excited coding. In this coder, the prediction residual is replaced by an optimized sequence consisting of a reduced number of samples (with 3:1 decimation factor) in which a grid selection procedure determines the transmitted baseband excitation signal [60]. This, however, means that the position of the selected sequence needs to be transmitted.
3.3.3 Analysis-By-Synthesis Linear Predictive Coding

The coders that have so far been reviewed in this section are classified as Analysis and Synthesis (Vocoding) type of systems. These obtain the residual signal by an analysis procedure and then directly quantize and transmit this residual. During the quantization process the error between the residual and its quantized value is minimized. Hence the quality of the synthesized speech is very dependent on the accuracy of quantization of the residual signal. These coders can produce high quality speech at bit rates as low as 8 kbit/s with moderate complexity [61]. In order to achieve good quality speech beyond these rates, the simplified and inadequate "excitation models" used in these systems significantly limit the coding performance. Improved speech quality can be achieved however, at bit rates in the range of 9.6 to 4.8 kbit/s by combining Analysis by Synthesis (AbS) optimization techniques with predictive coding.

A block diagram of an analysis-by-synthesis system is illustrated in figure 3.10 [62]. In these system both the "filter" and "excitation" models are parametrically defined on a short-term basis. Furthermore, the model parameters are estimated using a closed-loop optimization process, which minimizes a "perceptually" weighted mean-square (WMSE) measure found between the input and the decoded speech signals. A local decoder is employed alongside the encoder to synthesize the encoded signal. Coders based on the AbS principles are robust to background noise and non-speech signals in the same way as waveform coders. Two of the most successful Analysis-by-Synthesis LPC systems developed as a result of extensive research into low bit rate speech coding are the Multipulse Excited and the Codebook Excited systems. The main difference between the two schemes lies in the way in which the excitation sequence is defined and coded, with the code excited showing much greater potential for near-toll quality speech at or below 4.8 kbit/s [63, 64].

Multipulse Excited Linear Predictive Coding (MPE-LPC)

Multipulse Excitation Linear Predictive Coding (MPE-LPC) is the first generation of analysis-by-synthesis predictive coders [65]. In MPE-LPC, the residual signal is replaced with an excitation waveform having the following properties [37]:

- the ability to produce an output from the LPC synthesis filter similar to that obtained using the unquantized residual;
Fig. 3.10 Block Diagram of a General Analysis-By-Synthesis Predictive Coder
The ability to be sufficiently accurately represented by much fewer bits than is required for the residual.

The multipulse LPC model replaces the traditional pitch-pulse and white-noise excitation with a sequence of pulses with no distinction being made between voiced and unvoiced speech. The excitation signal is a sequence of \( P \) irregular spaced pulses i.e. given a frame of \( n \) speech samples the \( n \)-sample excitation sequence contains \( P \) non-zero pulses. For good quality speech several pulses are required per pitch period. However, as all the pulse positions and amplitudes must be transmitted a quality versus bit rate trade-off has to be made. The derivation of the appropriate pulse positions and amplitudes at the encoder is very crucial to the coder performance and many different schemes have been reported.

Figure 3.11 illustrates a Multipulse coder including long-term prediction. The excitation analysis procedure requires the partitioning of the input speech into small block (32-128) samples, and a search for the pulse positions and amplitudes which minimizes the error between the input and synthesized speech, over the block. The error signal is further processed to produce a measure of perceptual error. This processing includes linear filtering of the objective error to attenuate frequencies where the error is perceptually less important and amplifies the frequencies where the error is perceptually more important. The locations and amplitudes of pulses are obtained, one pulse at a time [66, 67]. After the first pulse has been determined, a new error is computed by subtracting the first pulse's contribution to the error, and the next pulse's location is determined by finding the minimum of the new error. The process of locating new pulses continues until the error is reduced to acceptable values or the number of pulses reaches the maximum value that can be encoded at the specified bit rate [68].

With a moderate increase in complexity, the performance of the sequential pulse search can be improved considerably, using a technique known as "amplitude re-optimization" [69]. This entails jointly optimizing the amplitudes of all the pulses in a frame once all their positions have been found. An alternative technique, which offers some advantage over amplitude re-optimization for lower bit rate coders, is known as "position re-optimization" [70].

The Multipulse coder can be extended to include long-term prediction [71-73]. The long-term prediction included in the encoding process takes advantage of the long-term correlation in speech which arise primarily as a result of pitch related correlations in
Fig. 3.11 Block Diagram of a Multipulse Excited Linear Predictive Coder (MPE-LPC)
voiced speech. With the inclusion of long-term prediction fewer pulses are required per pitch period to obtain the same speech quality. The long-term parameters then need to be accurately quantized and transmitted.

Multipulse coders provide near toll quality speech at 16 and 9.6 kbit/s. Their performance deteriorates rapidly however at bit rates below 8 kbit/s. At these bit rates acceptable performance can only be achieved by drastically modifying the basic Multipulse Excitation Model [72].

A special case of MPE-LPC coding is the Regular Pulse Excitation LPC (RPE-LPC) coder [74, 75] which models the excitation signal with a sequence of equally spaced pulses. The performance of the RPE-LPC system is similar to that obtained from MPE-LPC coders. Various computationally efficient RPE schemes have been proposed and one of them [76] has been chosen as the standard for the GSM European Digital Mobile Radio system.

**Codebook Excitation Linear Predictive Coding (CELP)**

Codebook Excitation Linear Predictive Coding (CELP), also known as Stochastic Coding, may be regarded as the second generation of analysis-by-synthesis predictive coders [77, 78]. A block diagram of the coder is shown in figure 3.12. The LPC (vocal tract) filter, long-term predictor and perceptual weighting filter are as described for the Multipulse coder. The difference is in the excitation function, the pulses being replaced by an "innovative sequence". The excitation sequence, via an exhaustive search, is selected from a codebook (which is available at both the encoder and the decoder) of random vectors (sequences) each of which is constructed using samples from a set of Independent Identical Distributed Gaussian random variables having zero mean and unit variance. The system selects that excitation sequence which minimizes the mean squared error between the input and the decoded signals. For perceptual reasons the error signal is weighted in such a way that the coding noise, in the recovered signal, is concentrated in the formant regions and is therefore masked by the powerful speech components in these regions. Again, the same method of analysis being used to determine the excitation waveform for all speech segments with no distinction being made between voiced and unvoiced speech.
Fig. 3.12 Block Diagram of a Codebook Excited Linear Predictive Coder (CELP Coder)
The error minimization scheme generates a codebook index and an optimum gain parameter corresponding to the selected optimum innovation sequence which are then transmitted to the decoder. At the decoder, the transmitted index and gain parameter are used to compute the identical innovation sequence. Each block of $M$ reconstructed speech samples is then produced by filtering this sequence, which consists of $M$ samples of white Gaussian noise, through the long-term filter and then the LPC vocal tract filter.

The major problem with CELP coding is the excessive number of computations required to select the optimum excitation sequence and various search strategies suitable for real-time implementation have been proposed. The codebook search simplifications are based on approximations in the way that the mean squared error is calculated [79, 80] and on modifications of the structure and the introduction of regularity into the CELP codebook [81-84].

The original proposed CELP coding algorithm has undergone numerous refinements such as efficient partitioning of the excitation signal into periodic and stochastic components [85], and efficient quantization of excitation and LPC parameters [86-88]. Further, proposals have been made to enhance the output speech quality of 4.8 kbit/s CELP by means of time-varying bit allocations to the LPC and excitation parts of the model [89], and by a careful use of adaptive postfiltering [50, 90, 91].

A computationally efficient CELP coder has recently been selected as the U.S. Government Standard 4.8 kbit/s voice coder [92, 93]. The coder produces high quality speech comparable to 32 kbit/s CVSD (DRT = 93, DAM = 67) and is robust in acoustic noise, channel errors, and tandem coding conditions. This standard forms the core of the Land Mobile Radio Standard that will include additional forward error correction for a total rate of 8 kbit/s.

CCITT is currently investigating the standardization of a 16 kbit/s speech coding scheme for use within the public switched telephone network. The two proposed coding schemes [94, 95] are both variants of CELP coding and use backward adaptation to track the spectral characteristics of the signal without requiring any buffering of the input speech, thereby allowing a very low encoding delay to be achieved.

The first candidate [96] is basically a CELP coder with a backward adaptive predictor and a robust backward gain adaptive vector quantizer [97] for excitation
coding. Gray-code Vector Quantization (VQ) index assignment is used to enhance the coder's robustness against channel errors [98, 99]. This Low-Delay CELP (LD-CELP) coder uses a block size (or vector dimension of only 5 samples) so as to meet what is the most demanding objectives of the CCITT 16 kbit/s standard; a one-way coding delay of less than 2 ms. In this system, the long-term predictor is eliminated due to its high sensitivity to channel errors. The order of the short-term predictor is increased from 10 to about 40 or 50 to improve speech quality in the absence of the long-term predictor. Formal subjective tests indicate that the speech quality of 16 kbit/s LD-CELP is similar to or better than that of G.721 for single encoding.

The second candidate [100, 101] has all the features of a conventional CELP coder except the use of backward adaptive linear prediction for modelling the time-varying short-term and long-term correlations of speech. The coder provides very good speech quality at 16 kbit/s, moderate complexity, a delay of under 2 ms, and a gentle degradation of quality with transmission errors. The coder also includes an adaptive postfilter to further enhance the coded speech quality.

Another variant of CELP coding, Base-Band Code Excited Linear Predictive Coding (CELP-BB) combines baseband coding, multipulse excitation coding and CELP coding schemes. This scheme will not only reduce the computational complexity significantly but also improves the quality achievable by MPE-LPC and CELP coders [102-105]. The coder is robust against both the background noise and channel errors. The built-in robustness of the coder makes the coder immune to error rates of up to 2% [106].

**Backward Excitation Recovery LPC Coding(BER-LPC)**

Backward Excitation Recovery Linear Predictive Coding (BER-LPC) [63] or Self Excited Vocoding (SEV) [107, 108] are the latest generation of the AbS type of coders. These coders employ a single or a multi-input speech synthesis filter and define the excitation signals from past information which is already available at both the encoder and the decoder. The excitation signal itself therefore need not be transmitted; instead the excitation is derived at the decoder, from past information, using the same strategy as that used by the encoder. The BER-LPC schemes offer low bit rate (4.8 kbit/s) combined with very small encoding delays (~ 3 ms). Low bit rate together with low delay is not feasible in the SEV case. The BER-LPC schemes and the SEV are equally susceptible to transmission errors. These types of backward adaptive coders require periodic resetting
of both encoder and decoder to mitigate the effects of transmission errors. BER-LPC and the SEV are both aimed at bit rates in the range 4 - 6 kbit/s [109-111]. The speech quality obtained from both types of coders, in the region of 8 to 4.8 kbit/s, is comparable to that obtained from CELP systems.

3.4 Concluding Remarks

In this chapter, some of the modem techniques of digital speech coding were reviewed. It is clear that in recent years, rapid progress has been made in producing high quality speech at low bit rates. The present generation of speech coders, using either Multipulse or Codebook Excitation in time domain and Sinusoidal Transform Coding in frequency domain, are able to generate speech of reasonably good quality at bit rates as low as 4.8 kbit/s. The progress in speech coding is leading to new standards for voice communication at rates of 4.8 - 16 kbit/s for digital cellular radio (Pan-European GSM DMR System, INMARSAT-M standard, U.S. DoD DMR System) and for Public Switched Telephone Network (CCITT 16 kbit/s standard). With the move to establish these standards, new initiatives are needed in speech coding research to push the frontiers for high quality speech coding even lower - down to 2.4 kbit/s [112]. This may need a complete re-thinking of modelling the speech production as high speech quality at such low bit rates cannot be achieved by incremental improvements of the existing coding concepts. It is the author's view that researchers in the field of low bit rate speech coding should no longer think of "computational complexity" as a constraint, as dramatic progress is being made in the DSP/VLSI field.

Some researchers [113] have suggested that the future work at low bit rates should attempt to devise algorithms that combine the flexibilities and capabilities of alternative approaches such as CELP and harmonic coding. In this process, it will be necessary to quantify the importance of phase fidelity in speech coding (a subject which has been totally ignored by the majority of researchers in speech coding), to incorporate refined noise audibility models in coder design, and to fully exploit the phenomenon of noise-masking in frequency and time domains, as in recent wideband audio work [114, 115]. The same author believes that the drop in speech quality at low bit rates should no longer be an acceptable argument and researchers should try to maintain the same speech quality at all bit rates.
Development of speech coders must no longer simply concentrate on improving speech quality. Lower delays and better robustness to both acoustic and transmission environment should be sought. Since these features cannot easily be added to an algorithm after development, the demands of the real world must be considered from the outset [37].

In future, there will be more demand on networks that combine data and speech for packetized type of communication [116]. This will require the design and development of efficient variable rate speech coding schemes for asynchronous communications [117].
References


42. B.S. Atal, "Predictive Coding of Speech at Low Bit Rates", IEEE Transactions on Communications, pp. 600-614, April 1982.


69. S. Singhal, B.S. Atal, "Improving the Performance of Multipulse LPC Coders at Low Bit-Rates", Proceedings of ICASSP, pp. 1.3.1-1.3.4, 1984.


4.1 Introduction

A 'real-time' process is a task which needs to be performed within a specified time scale. The time scale may vary from an order of 200 ns to values of up to 20-30 ms. In digital speech coding, assuming an 8 kHz sampling rate, the real-time processing needs to be performed within 125 μsec, for sample-by-sample analysis. The real-time process is usually performed on a block of data increasing the allowed time scale. This, however, does not realise any pressure on the amount of computations needed to be performed. This implies that the real-time execution of a task requires a combination of efficient programming and a high speed processor. As table 4.1 indicates, real-time analysis of voiceband and audio signals may easily be possible with today's VLSI technology but we may still be distant from real-time processing of video signals, at least with use of only a small number of processors.

<table>
<thead>
<tr>
<th>Application Area</th>
<th>Sample Rate (kHz)</th>
<th>Sample Period</th>
<th>MAC/Sample @100 ns MAC</th>
<th>MAC/Sample @ 50 ns MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voiceband</td>
<td>8 kHz</td>
<td>125 μsec</td>
<td>1,250</td>
<td>2,500</td>
</tr>
<tr>
<td>Audio</td>
<td>44 kHz</td>
<td>22.7 μsec</td>
<td>227</td>
<td>454</td>
</tr>
<tr>
<td>Video</td>
<td>5 MHz</td>
<td>200 nsec</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4.1 Real-Time Processing of Various Digital Signals
We are, however, only concerned herein with real-time coding of speech signals. Our aim in this chapter is to discuss the recent efforts in this direction and to assess the current available VLSI technology for such purpose. We aim to show that programmable digital signal processors (DSP) are currently the clear winners in this respect, as compared to the old-time array processors or today's transputers. The major differences between fixed and floating-point processors are highlighted, together with a review of the available DSP chips in both categories. The software development cycle is also considered. Finally, a detailed description of the architecture of two floating-point DSP chips, AT&T WE-DSP32 and WE-DSP32C, selected for the purpose of implementing our speech coders, is given.

4.2 Implementation Strategies

The speech coding field has undergone a significant advancement in the past decade. There was a time when evaluation of speech coders at rates as high as 16 kbit/s was only restricted to nonreal-time computer simulations due to the difficulty of developing specialized hardware or lack of high-speed processors. Nowadays, in contrast to this, highly complex coding schemes operating at rates as low as 4.8 kbit/s are implemented on a single processor [1,2]. Although, there exists numerous media for real-time realization of a digital process, such as array processors, transputers, application specific DSP (ADSP); programmable DSPs are the most suitable for signal processing applications, in particular speech coding. Array processors were used extensively in the past but they have minimum level of portability and are very expensive. Transputers, on the other hand, may be a good replacement for array processors but are best suited to 2-dimensional problems where a high degree of parallelism exists, such as image processing [3-5]. Use of programmable processors has become an attractive alternative to custom VLSI not only in the speech coding field but in many other real-time digital signal processing applications. The success and widespread use of this alternative, however, will depend on the development of efficient software and hardware development environments. This is a very crucial part of using programmable DSPs and one which is most often shadowed by other aspects of a DSP chip. It is for this reason that the reader is strongly advised, when comparing the features of DSP chips, to study the available software and hardware tools in detail as well as the ease of programming and availability of the chips.
In the next section, the architecture, main differences between fixed and floating-point and the methods sought to increase the speed and efficiency as well as the software and hardware development cycle of DSP chips are studied.

4.3 DSP Implementation of Speech Coders

VLSI implementation of sophisticated speech coders is essential in order to realize small-size, low-power-consumption hardware, resulting in economical and efficient system configurations. From the standpoint of technology, there are three basic alternatives for digital speech coder implementations [6]:

- a general-purpose digital signal processor,
- a DSP tuned for a specific coding algorithm, and
- a dedicated LSI.

The development of a DSP-based coder may require a short-time, mainly for programming the coding algorithm. Therefore, a DSP would be useful in implementing various coding algorithms within a short-lead time. The DSP implementation (programmable) is most suitable for developing speech coder prototypes, whereas, a 'tuned-up' DSP or dedicated LSI (or custom LSI) is preferable for implementation of a particular speech coder e.g. CCITT standard [7], when the codec is in great demand.

**Programmable DSPs**

Programmable DSPs are specialized microcomputers for real-time number crunching. Target applications require extensive computations, usually with strict real-time constraints. DSPs are traditionally designed for performance, not extensive functionality nor programmer convenience. Because of their specialized applications, programmable DSPs have evolved architectures that are significantly different from conventional microprocessors. A number of architectural innovations have been used to achieve this impressive performance. The most basic is the integration of fast multiplier/accumulator hardware into the main data path. Obviously, this is advantageous when most of the instructions involve arithmetic. But integrated fast arithmetic whilst necessary, is not sufficient. Today's DSPs use extensive pipelining, several independent memories, parallel function units, and hardwired designs (not microprogrammed). [8]
The classical Von-Neumann processor architecture, characterized by the sequential execution of various tasks, reached its limits in number crunching at frequency ranges as low as a few kHz. More efficient processor architectures had to be invented. The Harvard architecture, characterized by pipelining and parallel execution of several tasks, copes with digital signal processing nowadays at frequency ranges breaking the Megahertz barrier. The introduction of multiple memory banks and buses into the basic Harvard architecture increases the memory bandwidth. The modified Harvard architecture of figure 4.1 achieves high memory bandwidth, where up to six simultaneous memory accesses can occur.

A typical programmable DSP has instructions that will fetch two operands from memory, multiply them, add them to an accumulator, write the results to memory, and post-increment three address registers. It is obvious that if all these operations had to be performed sequentially within one instruction cycle, the instruction cycle times would be much longer than they in fact are. Fast execution is accomplished using 'pipelining'. In a pipelined structure, the processor divides the instruction execution task into instruction fetch, decode, operand fetch, and execute stages. Pipelining effectively speeds up the computation, but it can have a serious impact on programmability. [10]

The following example will help the reader to understand better the concept of pipelining. The DSP32 executes the multiply-accumulate instruction [11]:

\[ aN = aM + Y \times X \]  \hspace{1cm} (4.1)

in three stages (an optional fourth stage to perform a write to memory is not considered); fetch of X and Y, multiplication of X and Y, and accumulation of the Y*X product with the contents of an accumulator aM. This instruction executes as follows:

step 1) XY fetch,
step 2) multiply Y and X,
step 3) accumulate product with aM.
If several instructions of this form are executed one after another, the processor automatically pipelines the instructions so that one instruction is completed every instruction cycle. Consider the following set of instructions:

1) \( aN = aM + Y_1 \times X_1 \)
2) \( aN = aM + Y_2 \times X_2 \)
3) \( aN = aM + Y_3 \times X_3 \)
4) \( aN = aM + Y_4 \times X_4 \)

The pipelined instruction flow is as shown below:

1) \( aN = aM + Y_1 \times X_1 \)
   \( \text{accumulate}_1 \)\n   \( \text{multiply } Y_1 \)\n   \( \text{XY fetch}_1 \)
2) \( aN = aM + Y_2 \times X_2 \)
   \( \text{multiply } Y_2 \)\n   \( \text{XY fetch}_2 \)
3) \( aN = aM + Y_3 \times X_3 \)
   \( \text{multiply } Y_3 \)\n   \( \text{XY fetch}_3 \)
4) \( aN = aM + Y_4 \times X_4 \)
   \( \text{multiply } Y_4 \)\n   \( \text{XY fetch}_4 \)

This means that the execution of the first instruction will finish 2 instructions later. This is referred to as 'latency' and it is the programmer's responsibility to be aware of its effects when writing the software. In some DSPs the pipelining is invisible to the programmer and in some it can be disabled, at the cost of reduced throughput.

The programmable DSPs are categorized by precision and arithmetic types; a) fixed-point, and b) floating-point. Fixed-point DSP chips are available with 16-, 24- and 32 bits of precision. DSP chips have recently been introduced with full floating-point arithmetic capabilities. In the next sections the main differences between a selection of the available chips, in both categories, are considered.

### 4.3.1 Fixed-Point Implementation

In the opinion of a DSP expert [9], the first chips that qualify as 'DSP chips' are the Bell Labs DSP1 [12] and NEC 7720 [13], both with fixed-point architectures. The fixed-point DSPs tend to be faster and cheaper but more difficult to programme and provide less precision. Fixed-point DSPs exist with data precision of up to 32-bit word length. The overall number of gates in a DSP system is a function of the bits in its
numeric representation, the gate count increases with the available range and precision of the numeric representation used. Therefore, one tradeoff that must be made is mathematical performance versus circuit and system complexity. It is not always necessary to use a numeric format which provides a level of performance not required for overall system functionality.

Most programmers are familiar with floating-point numbers and restrict their use of fixed-point numbers to array indices, loop counters, or other variables. This, however, changes when processing fixed-point DSPs, where the user must pay close attention to inherent issues such as underflow, overflow. Uniform precision, throughout the implementation, must be maintained, by constant checking of numbers and appropriate scaling.

The precision loss (or quantization errors) mostly occur when multiplying two numbers, because the number of bits required to specify the product with full precision is equal to the sum of the number of bits in the operand. Discarding any of these bits entails a loss of information. It is possible to store the full product as double precision numbers but this is expensive (in time and memory) and is usually not necessary. In some DSPs the overflow is controlled by simply setting the value of the result to the largest magnitude positive or negative numbers, as appropriate. In order for this to work, the Arithmetic Logic Unit (ALU) should have saturating hardware, and there must be saturation hardware between the accumulator and the data bus. Another overflow prevention scheme is to shift down (with sign extension) a product before adding to the accumulator, discarding the low order bits. This, of course, results in a loss of precision.

Many fixed-point DSPs exist and in table 4.2 we compare the features of a selection of these [14-16]. The most important parameters to consider are the data length, the instruction cycle time and the power dissipation.

AT&T's DSP16A 16-bit, integer DSP chips offer markedly higher speeds compared with the company's earlier DSP16 devices. The new chips processed in 0.75-μm CMOS technology, are available with 25 and 33 nsec instruction cycle times vs the older devices' 55 and 75 nsec. In addition, the DSP16A offers more on-chip memory than the earlier DSP16 units. The upgraded devices maintain source code, object code, and pin compatibility with their slower generation. The implementation of the CCITT wideband coder and an Adaptive Differential Pulse Code Modulation (ADPCM) Transcoder on the DSP16 have been reported [17,18].
<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Data Length (Bits)</th>
<th>MAC Input Precision (Bits)</th>
<th>MAC Output Precision (Bits)</th>
<th>Instruction Cycle Time (ns)</th>
<th>Clock Rate (MHz)</th>
<th>Internal Memory (Bits)</th>
<th>External Memory (Bits)</th>
<th>On-Chip Parallel I/O</th>
<th>On-Chip Serial I/O</th>
<th>Power Dissipation (mW)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog Devices</td>
<td>ADSP-2100A</td>
<td>16</td>
<td>16 x 16</td>
<td>40</td>
<td>80</td>
<td>12.5</td>
<td>None</td>
<td>None</td>
<td>No</td>
<td>No</td>
<td>790</td>
<td>Enhanced version of ADSP2100.</td>
</tr>
<tr>
<td></td>
<td>ADSP-2101</td>
<td>16</td>
<td>16 x 16</td>
<td>40</td>
<td>80</td>
<td>12.5</td>
<td>2k x 24 P</td>
<td>None</td>
<td>No</td>
<td>Yes</td>
<td>825</td>
<td>Able to load internal RAM from external EPROM.</td>
</tr>
<tr>
<td></td>
<td>ADSP-2102</td>
<td>16</td>
<td>16 x 16</td>
<td>40</td>
<td>80</td>
<td>12.5</td>
<td>2k x 24 P</td>
<td>1k x 16 D</td>
<td>Yes</td>
<td>Yes</td>
<td>825</td>
<td>Mask-ROM version of ADSP2101.</td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>DSP16A</td>
<td>16</td>
<td>16 x 16</td>
<td>32</td>
<td>25</td>
<td>40</td>
<td>2k x 16</td>
<td>4k x 16</td>
<td>Yes</td>
<td>Yes</td>
<td>450</td>
<td>Enhanced version of DSP16.</td>
</tr>
<tr>
<td>FUJITSU</td>
<td>MB8764</td>
<td>16</td>
<td>16 x 16</td>
<td>26</td>
<td>100</td>
<td>10</td>
<td>256 x 16</td>
<td>1k x 24 P</td>
<td>No</td>
<td>No</td>
<td>300</td>
<td>MB87064 version exists with fewer pins.</td>
</tr>
<tr>
<td>MICROCHIP TECHNOLOGY</td>
<td>DSC32C14</td>
<td>16</td>
<td>16 x 16</td>
<td>32</td>
<td>160</td>
<td>25.6</td>
<td>256 x 16</td>
<td>4k x 16</td>
<td>Yes</td>
<td>Yes</td>
<td>600</td>
<td>Full-Function Micro Controller based on TMS10.</td>
</tr>
<tr>
<td>MOTOROLA</td>
<td>DSP56001</td>
<td>24</td>
<td>24 x 24</td>
<td>56</td>
<td>74</td>
<td>27</td>
<td>1k x 24</td>
<td>544 x 24</td>
<td>Yes</td>
<td>Yes</td>
<td>450</td>
<td>A limited on-board DSP functions.</td>
</tr>
<tr>
<td>NEC</td>
<td>µPD77C25</td>
<td>16</td>
<td>16 x 16</td>
<td>31</td>
<td>122</td>
<td>8.192</td>
<td>256 x 16</td>
<td>None</td>
<td>Yes</td>
<td>Yes</td>
<td>250</td>
<td>Enhanced version of µPD7720/77C20.</td>
</tr>
<tr>
<td></td>
<td>µPD77P25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>µPD77220</td>
<td>24</td>
<td>24 x 24</td>
<td>47</td>
<td>122</td>
<td>16.384</td>
<td>1k x 24</td>
<td>2k x 32 P</td>
<td>8k x 24 D</td>
<td>Yes</td>
<td>1000</td>
<td>Slave or Master mode selectable.</td>
</tr>
<tr>
<td></td>
<td>µPD77P220</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGS-THOMSON</td>
<td>ST18930</td>
<td>16 or 32</td>
<td>16 x 16</td>
<td>32</td>
<td>80</td>
<td>20</td>
<td>320 x 16</td>
<td>3k x 32 P</td>
<td>512 x 16</td>
<td>No</td>
<td>No</td>
<td>Enhanced version of ST18930.</td>
</tr>
<tr>
<td></td>
<td>ST18931</td>
<td>16 or 32</td>
<td>16 x 16</td>
<td>32</td>
<td>80</td>
<td>20</td>
<td>320 x 16</td>
<td>None</td>
<td>64k x 32 P</td>
<td>4k x 16 D</td>
<td>No</td>
<td>Enhanced version of ST18931.</td>
</tr>
<tr>
<td></td>
<td>ST18940</td>
<td>16 or 32</td>
<td>16 x 16</td>
<td>32</td>
<td>100</td>
<td>20</td>
<td>512 x 16</td>
<td>3k x 32 P</td>
<td>512 x 16</td>
<td>No</td>
<td>Yes</td>
<td>Enhanced version of ST18930.</td>
</tr>
<tr>
<td></td>
<td>ST18941</td>
<td>16 or 32</td>
<td>16 x 16</td>
<td>32</td>
<td>100</td>
<td>20</td>
<td>512 x 16</td>
<td>128 x 16</td>
<td>64k x 32 P</td>
<td>64k x 16 D</td>
<td>No</td>
<td>Enhanced version of ST18931.</td>
</tr>
<tr>
<td>Texas Instruments</td>
<td>TMS320C25</td>
<td>16</td>
<td>16 x 16</td>
<td>32</td>
<td>80</td>
<td>40</td>
<td>544 x 16</td>
<td>4k x 16 P</td>
<td>No</td>
<td>Yes</td>
<td>925</td>
<td>Enhanced version of TMS320C25.</td>
</tr>
<tr>
<td></td>
<td>TMS320C26</td>
<td>16</td>
<td>16 x 16</td>
<td>32</td>
<td>100</td>
<td>40</td>
<td>1.6k x 16</td>
<td>256 x 16</td>
<td>No</td>
<td>Yes</td>
<td>925</td>
<td>Triples on-chip RAM capacity of TMS320C25.</td>
</tr>
<tr>
<td></td>
<td>TMS320C50</td>
<td>16</td>
<td>16 x 16</td>
<td>32</td>
<td>50/35</td>
<td>57</td>
<td>8.7k x 16</td>
<td>2k x 16</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>Not available yet.</td>
</tr>
</tbody>
</table>

P = Programme,  D = Data

Table 4.2 Comparison of Fixed-Point DSP Chips [14-16]
A recent speed increase in Motorola's DSP56001 24-bit integer processor provides a 30% increase in throughput. The device's 27 MHz operating frequency computes at 13.5 million instructions per second, making the device suitable for such applications as CD-quality sound processing, 2-D graphics, and digital television functions. The 56001's triple bus architecture simultaneously handles two pieces of data and an instruction. The implementation of a 16 kbit/s Adaptive Subband Excited Transform (ASET) coding scheme on a single DSP56000 has been reported [19].

The µPD77C25 is an upgraded version of the NEC's µPD7720/77C20 family of 16-bit fixed-point devices. The new chips are pin compatible with the 7720 devices, and they execute instructions twice as fast (122 vs 250 nsec). The NEC fixed-point DSPs have been used for the implementation of numerous speech coding schemes. As an example, a single NEC µPD7720 is used to implement a 16 kbit/s subband coder [20].

The second generation DSPs produced by Texas Instruments, TMS320C25-50, reduce the instruction cycle time from 100 to 80 nsec, corresponding to an increase in maximum clock rate of 40 to 50 MHz. Object code and pin-compatible with the TMS320C25-50, the recently TMS320C26 includes three times the amount of on-chip RAM. The TMS320 series of fixed-point DSPs have been used for the real-time realization of digital speech coders operating at rates as low as 800 bps to rates as high as 32 kbit/s [21-23].

### 4.3.2 Floating-Point Implementation

In 1984, AT&T developed the first single-chip floating-point programmable processor, the DSP32 [24]. It was a first in two important respects: the first DSP that AT&T marketed, and the first 32-bit floating-point DSP. This was shortly followed by NEC introducing their floating-point DSP, µPD77230, a 32-bit processor with an instruction cycle time of 150 ns [25].

A floating-point number, \( x \), is made up of a mantissa, \( M_x \), and an exponent, \( E_x \), such that:

\[
x = M_x \cdot 2^{E_x}
\]

The multiplication of two floating-point numbers, \( x \) and \( y \), results in a product, \( z \):
\[ z = M_x \cdot M_y \cdot 2^{E_x+E_y} \]  

This means that a hardware floating-point multiplier must contain both a multiplier for the mantisa and an adder for the exponent. Extra precision is usually provided to store the product of the two mantisas. The precision and dynamic range of a floating-point processor depends on the data lengths of the mantisa and exponent. Although, an IEEE standard on the floating-point data format exists, the DSP manufacturers usually select their own format (see table 4.3).

The advantages of floating-point implementation over fixed-point are many [26]. These advantages often lead to great savings in the development effort for a product or programme. The precision of a floating-point number remains constant throughout a programme because of automatic normalization of the mantisa by the processor. Whereas, the precision of fixed-point data varies with the size of the stored data or resulted value (see figure 4.2). Due to this normalization of the mantisa, rounding or truncation leads to smaller overall errors that would be found in fixed-point implementations. This constant precision, coupled with the ability to represent very large or very small numbers allows much more precise placement of poles and zeros, eliminating much of the trial-and-error implementation of filters.

![Fig. 4.2 Precision of Fixed- and Floating-Point DSPs [11]](image-url)
The much larger dynamic range possible in floating-point formats (see figure 4.3) allows much larger and smaller values to be maintained. This is especially necessary in intermediate computations of FFTs, and high-order recursive filters. The high dynamic range in turn leads to the elimination of intermediate scaling necessary to prevent overflow or underflow of data on a fixed-point device. Because of this, the use of a floating-point frequently reduces the size and complexity of the programmes that implement DSP algorithms.

The floating-point format also simplifies programming in another way. Most DSP algorithms are implemented on some sort of Mainframe or minicomputer to test their feasibility. These machines generally have floating-point capabilities. Rewriting these programmes for a floating-point device is much simpler and straightforward than the convolution of code necessary for use on a fixed-point device.

Table 4.3 compares the features of a number of selected floating-point DSPs. The important parameters to consider are the data length (the precision of mantissa and exponent), the instruction cycle time and the processor power dissipation, not forgetting the customer support, availability and the provided support software and hardware tools (not included in the table). The three most powerful processors, available to date, are the

![Fig. 4.3 Dynamic Range of Fixed- and Floating-Point DSPs [11]](image-url)
<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Data Length (Bits)</th>
<th>MAC Input Precision (Bits)</th>
<th>MAC Output Precision (Bits)</th>
<th>Instruction Cycle Time (nsec)</th>
<th>Clock Rate (MHz)</th>
<th>Internal Memory (RAM ROM)</th>
<th>External Memory (Bits)</th>
<th>On-Chip Parallel I/O</th>
<th>On-Chip Serial I/O</th>
<th>Power Dissipation (mW)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T</td>
<td>DSP32C</td>
<td>24 M 8 E</td>
<td>32 x 32</td>
<td>40</td>
<td>80</td>
<td>50</td>
<td>1k x 32 2k x 32</td>
<td>16M x 8 4M x 32</td>
<td>Yes</td>
<td>Yes</td>
<td>1250</td>
<td>Enhanced version of DSP32 with interrupt facility.</td>
</tr>
<tr>
<td>FUJITUSU</td>
<td>MB86220</td>
<td>18 M 6 E</td>
<td>24 x 24</td>
<td>30</td>
<td>150</td>
<td>40</td>
<td>512 x 24 2k x 30</td>
<td>64k x 30 128k x 30 D</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>Semicustom DSP development.</td>
</tr>
<tr>
<td></td>
<td>MB86224</td>
<td>18 M 8 E</td>
<td>24 x 24</td>
<td>30</td>
<td>150</td>
<td>40</td>
<td>512 x 24 2k x 30</td>
<td>128k x 24</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>PC Serial Bus Interface.</td>
</tr>
<tr>
<td></td>
<td>MB86232</td>
<td>24 M 8 E</td>
<td>32 x 32</td>
<td>32</td>
<td>150</td>
<td>40</td>
<td>512 x 32 1k x 32</td>
<td>64k x 32 1M x 32 D</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>IEEE Data Format Conversions.</td>
</tr>
<tr>
<td>MOTOROLA</td>
<td>DSP96002</td>
<td>23 M 8 E</td>
<td>32 x 32</td>
<td>44</td>
<td>74</td>
<td>27</td>
<td>2k x 24 1088 x 32</td>
<td>4G x 32 8G x 32 D</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>Very powerful chip.</td>
</tr>
<tr>
<td>NEC</td>
<td>µPD77230</td>
<td>24 M 8 E</td>
<td>32 x 32</td>
<td>55</td>
<td>150</td>
<td>13,333</td>
<td>1k x 32 2k x 32 P 4k x 32 P 8k x 32 D</td>
<td>Yes</td>
<td>Yes</td>
<td>1500</td>
<td>3-stage pipelining.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>µPD777P230</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OKI Semiconductors</td>
<td>MSM699210</td>
<td>16 M 6 E</td>
<td>22 x 22</td>
<td>22</td>
<td>100</td>
<td>40</td>
<td>512 x 22 2k x 32 P 64k x 32 P 64k x 22 D</td>
<td>Yes</td>
<td>No</td>
<td>400</td>
<td>Enhanced version of MSM6992.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MSM699215</td>
<td>16 M 6 E</td>
<td>22 x 22</td>
<td>22</td>
<td>100</td>
<td>40</td>
<td>512 x 22 2k x 32 P 64k x 32 P 64k x 22 D</td>
<td>Yes</td>
<td>Yes</td>
<td>400</td>
<td>Serial I/O version of MSM699210.</td>
<td></td>
</tr>
<tr>
<td>Texas Instruments</td>
<td>TMS320C30</td>
<td>24 M 8 E</td>
<td>32 x 32</td>
<td>40</td>
<td>60</td>
<td>33</td>
<td>2k x 32 4k x 32</td>
<td>16M x 32</td>
<td>Yes</td>
<td>Yes</td>
<td>1500</td>
<td>Real-time Operating System (SPOX).</td>
</tr>
<tr>
<td>ZORAN</td>
<td>ZR34325</td>
<td>24 M 8 E</td>
<td>32 x 32</td>
<td>44</td>
<td>80</td>
<td>25</td>
<td>128 x 32 1k x 32</td>
<td>16M x 32 16M x 32 D</td>
<td>No</td>
<td>No</td>
<td>1000</td>
<td>Direct process of IEEE 754 formatted data.</td>
</tr>
</tbody>
</table>

P = Programme, D = Data, M = Mantisa, E = Exponent.

Table 4.3 Comparison of Floating-Point DSP Chips [14-16]
AT&T DSP32C, the Motorola DSP96002 and the Texas Instruments TMS320C30, with the DSP32C being the most favoured floating-point processor at the present time.

The architecture and features of the AT&T DSP32 and DSP32C are considered in details in the following sections. Numerous coding schemes have been implemented using the two chips either in a single or double DSP structure. Among the coding algorithms implemented are a 16 kbit/s Multipulse coder and a 16 kbit/s Subband coder using vector quantization on a single DSP32, a Self-Excited Vocoder, a Vector Adaptive Predictive coder (VAPC) and a Vector Excitation coder, all operating at 4.8 kbit/s, on a single DSP32, a full-duplex system based on Multiband Excitation (MBE) coding operating at rates 2.4, 4.8 and 8 kbit/s on a single DSP32, and a 16 kbit/s low-delay CELP coder (CCITT candidate) on two DSP32Cs [27-32]. The DSP32 has also been used in areas other than speech coding [33-36].

4.3.3 Real-Time Software Development

The theoretical algorithm development and its implementation are of equal significance. A coding scheme which requires a number of DSPs to realize is not attractive in terms of cost, size and power consumption. Thus, it is very important to implement the algorithm in an optimized manner and this in turn means a combination of optimized software and hardware.

Real-time implementation of a specific task on a DSP chip involves several stages (see figure 4.4); (i) theory development and refinements, (ii) verification of the algorithm by computer simulations using a high-level language (such as C-language), (iii) translating the high-level code into corresponding DSP code, (iv) verification of the real-time code using the available development tools, (v) designing the appropriate hardware and finally (vi) the integration of the software and hardware. Each stage of the development is as important as the next and may prove to be equally tedious.

In the second phase, the verification of the algorithm may lead to refinements of the theory or a complete re-study. In the fourth phase, the computer simulation results are compared against the equivalent output of the programme. This is usually done for short test files of the input data since it is costly and inconvenient to process large amounts of data through the simulator. Once agreement is obtained between these two outputs, we are assured that the DSP software is essentially correct. The software is then downloaded into the appropriate hardware and tested in real-time. The
Fig. 4.4 Real-Time Software Development Cycle
In-Circuit-Emulation (ICE) facility at this stage would prove absolutely invaluable and necessary to speed up the debugging process. Once the algorithm is running the parameters are 'tuned' in a real-time environment to achieve the optimum performance.

In the third phase, the transfer of high-level code to the DSP code can either be performed using the high-level cross-assemblers or 'hand-coded'. The first method, although, much faster does not produce the optimized code. This is due to the lack of efficient cross-compilers and an area which needs considerable attention by both industry and researchers if we were to spread the use of DSPs. Currently, hand-coding of the software remains the only efficient solution both in terms of memory usage and speed of execution.

4.4 AT&T WE-DSP32 Floating-Point Processor

The WE-DSP32 Digital Signal Processor [37,38] is a single chip VLSI circuit with 32-bit instructions, 32-bit data path and full 32-bit hardware floating-point arithmetic. The DSP32 is implemented in 1.5 μm NMOS technology and operated with a system clock of 25 MHz (a slower version operates at 16 MHz). Each DSP32 instruction requires four machine cycles to execute, so the instruction execution rate is 6.25 million instructions per second (MIPS). Since each instruction can perform one floating-point multiply and one floating-point addition, the DSP32 is capable of 12.5 million floating-point operations per second (MFLOPS). The DSP32 is available in a 40-pin DIP or a 100-pin rectangular pin-grid-array (PGA) package. The pin-array package provides pins for an external 32-bit data bus and a 14-bit address bus for use with external memory expansion.

4.4.1 Architecture

Figure 4.5 shows the block diagram of the DSP32. The architecture of the DSP32 consists of a number of specialized subsections that work in parallel to achieve a high throughput rate. The main hardware features of the DSP32 are highlighted in the following sections. A more detailed description can be found in [39].
Fig. 4.5 WE-DSP32 Digital Signal Processor Block Diagram [39]
Memory

On-chip memory includes 2048 bytes of ROM for instructions and fixed data and 4096 bytes of RAM for variable data or instructions. The ROM is mask-programmed for application programmes. Memory can be addressed as 8-, 16-, or 32-bit words, and is organized to access 32-bit data at the same speed as 8-bit data. With the 100-pin package, memory can be expanded off-chip with an additional 56 kbytes of directly accessible data. The memory is divided into two banks and access can be made without regard to either banks. However, to achieve maximum throughput, memory accesses must alternate between the two banks.

Control Arithmetic Unit (CAU)

The Control Arithmetic Unit (CAU) is used to generate addresses for memory, perform address arithmetic and execute 16-bit integer operations. It has twenty-one 16-bit general purpose static registers, a 16-bit programme counter (PC) and a full function Arithmetic Logic Unit (ALU).

Data Arithmetic Unit (DAU)

The Data Arithmetic Unit (DAU) is the main execution unit for signal processing algorithm. It performs 32-bit floating-point multiplication and addition. It consists of a 32-bit floating-point multiplier, a 40-bit adder, four static 40-bit accumulators and a DAU Control Register (DAUC).

Serial Input and Output Unit (SIO)

The Serial I/O Unit (SIO) serially transmits and receives data either under programme control or DMA (Direct Memory Access). The control signals which can be generated on-chip or provided from external circuitry allow a direct interface to a time-division-multiplexed (TDM) line, another DSP32, or standard codecs.

Parallel Input and Output Unit (PIO)

The Parallel I/O Unit (PIO) allows communication with an external microprocessor. Bidirectional transfer is through the 8-bit parallel I/O data bus and can be either under programme or DMA control. The PIO DMA permits the microprocessor to download an application programme without interrupting another programme in progress. It also contains circuitry for refreshing the on-chip RAM and error trappings such as DAU floating-point overflow and memory parity error.
4.4.2 Software Development

The architecture of the DSP32 is highly pipelined and parallel. During a particular processor cycle, up to six different instructions may be in various stages of fetch-decode-execute. This advanced architecture creates certain latency and pipeline effects. These should be carefully taken into considerations when writing assembly programmes for the DSP32.

The DSP32 instruction set consists of five groups and are divided into two main types: (i) Data Arithmetic (DA) instructions, and (ii) Control Arithmetic (CA) instructions. DA instructions perform 32-bit floating-point multiply/accumulate operations. DA instructions also include special functions for data type conversions such as floating-point to-and-from integer and floating-point to-and-from \( \mu \)-law and A-law. CA instructions are 16-bit integer microprocessor instructions that include arithmetic and logic instructions, and control statements such as conditional 'goto' and 'call'. CA instructions also include data move statements so that data can be moved between memory, I/O registers, and any of the CAU registers.

The assembly language syntax is similar to the C programming language making it much easier to write and understand the software developed for the DSP32.

4.5 AT&T WE-DSP32C Floating-Point Processor

The WE-DSP32C high-performance, programmable digital signal processor supports 32-bit floating-point arithmetic and is upwardly compatible with its predecessor, the WE-DSP32. Because it is implemented in 0.75-\( \mu \)m (effective channel length) CMOS technology, the second generation device achieves high functional density with low power consumption. While remaining upwardly compatible with the DSP32, the DSP32C offers the several enhancements listed in table 4.4. The processor contains 405,000 transistors in an area of 88 mm\(^2\) that is enclosed in a standard 133-pin-square PGA package. [40]

At a clock rate of 50 MHz (an 80 ns instruction cycle time), the DSP32C executes 12.5 million instructions per second. This implies that it is capable of performing 25 MFLOPS. The device also performs 24-bit integer operations at the rate of 12.5 million operations per second.
Fig. 4.6 WE-DSP32C Digital Signal Processor Block Diagram [11]
Figure 4.6 shows the block diagram of the DSP32C. The overall architecture of the DSP32C remains identical to that of the DSP32. The main features of the DSP32C processor can be summarized as follows:

- 25 Mflop operation,
- 16 Mbit/s serial input and output ports,
- 16-bit parallel I/O port for control and data transfer,
- interrupt facilities,
- single-instruction μ-law and A-law data conversions,
- single-instruction conversion between integers and floating-point data,
- a byte addressable, on-chip memory that is expandable off-chip,
- direct memory access to and from internal and external memory via parallel and serial I/O ports,
- 16 Mbytes of address space, and
- IEEE Std. 754 floating-point format conversion.

<table>
<thead>
<tr>
<th>DSP32</th>
<th>DSP32C</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.5 million floating-point operations per second (160 ns instruction cycle)</td>
<td>25 million floating-point operations per second (80 ns instruction cycle)</td>
</tr>
<tr>
<td>16-bit address space</td>
<td>32-bit address space</td>
</tr>
<tr>
<td>12.5 Mbit/s Serial I/O Port</td>
<td>16 Mbit/s Serial I/O Port</td>
</tr>
<tr>
<td>8-bit Parallel I/O Port</td>
<td>16-bit Parallel I/O Port</td>
</tr>
<tr>
<td>25 MHz Operating Frequency</td>
<td>50 MHz Operating Frequency</td>
</tr>
<tr>
<td>-</td>
<td>Interrupt Capabilities</td>
</tr>
<tr>
<td>-</td>
<td>IEEE Std. 754 Floating-point Format Conversion</td>
</tr>
<tr>
<td>-</td>
<td>LSB- or MSB-first Format on SIO Port</td>
</tr>
</tbody>
</table>

Table 4.4 DSP32C Enhancements [40]

The most useful enhancement is the provision of the interrupts. A single-level interrupt facility can respond to four internal and two external, individually maskable
sources. A relocatable vector table controls programme flow based on the source of the interrupt.

The structure and operation of the memory, CAU and DAU units remains the same as the DSP32 and the reader is referred to [11] for the detailed description of these units.

4.6 Concluding Remarks

A good real-time system is a result of a well-developed and refined theoretical algorithm, a highly efficient and fast processor (hardware) and an optimized software. A decade ago, the implementation of complex real-time digital systems, such as digital speech coders, was only possible on expensive, bulky and power hungry media. The advent of digital signal processing chips has dramatically changed the state of the affairs. It is now possible to real-time implement complex processing tasks on a small-size printed circuit board which consumes only a few watts.

Fixed and floating-point programmable DSPs have become commercially available, capable of performing several million operations per second (up to 50 MOPS). A number of architectural innovations have been introduced into the classical Harvard structure to reach at this impressive performance. Floating-point DSPs that only became commercially available six years ago (in 1984) have been used extensively in implementing numerous speech coding techniques as well as in many other applications. Fixed-point DSPs remain cheaper and faster at the present time but in this author's opinion will soon be replaced by the more powerful and cheaper floating-point DSPs. The progress of the floating-point devices in the past six years, both in terms of speed and cost, makes this claim more substantial. The advantages of floating-point over fixed-point are so numerous that the manufacturers of these devices are being forced to look at ways of increasing their performance thus making them the user's absolute first choice. It is also anticipated that programmable floating-point DSPs will soon become standard peripherals in personal computers and workstations [41,42], handling real-time and compute-intensive tasks. Such widespread use of these devices is bound to have a great impact on their price.

The main disadvantage of using DSPs is that the software development is very time consuming even though the final code may be very small. Existing high-level cross-
compilers are inefficient and even the eventual optimized compiler may not be the complete solution. The Block-Diagram DSP Programming Environments, such as the one being developed at Berkeley [43], where the user graphically constructs a block diagram of his chosen algorithm carry high promises of a neat and fast software development environment.

Programmable DSPs have always been used in the initial stages of evaluating the performance of digital speech coding schemes in real-time. It is easier to develop prototypes and make algorithm or software changes (algorithm 'tuning'). As speech coding techniques have become more and more complex, there has been a matching increase in the processing power of the DSPs. The simultaneous performance increase of the DSPs has, in fact, played an important role in the design and development of new complex coding schemes. Following the same trends, we are, hopefully, going to see better quality speech coders, at much lower bit rates, implemented on single-chip processors, in the near future.
References


42. J. Shandle, "DSPs Start their Move to the Motherboard", Electronics Magazine, pp. 93-95, May 1990.

CHAPTER 5

Speech Coder For Pan-European Digital Mobile Radio System

5.1 Introduction

Background

In 1991, a new Pan-European Digital Mobile Cellular Telephone system will be opened for service, enabling people travelling in Europe to make, and receive telephone calls (and data communications) with people anywhere in the 'telephone world'. This will mark the end of almost a decades coordinated development and implementation programme. In 1982, the European Conference of Post and Telecommunications Administration (CEPT) initiated a study towards standardization of the second generation of Mobile Radio System, allocating the frequency bands 890-915 MHz and 935-960 MHz for this system. It was also decided to create a committee known as "Group Special Mobile" (GSM) to establish the specifications of a harmonized public mobile radio communication system [1-3].

The main drive for this initiative was the tremendous increase in the demand for the mobile service which was demonstrated in the first generation of cellular system, deployed throughout Europe in the 80's. Currently, there are more than 1 million cellular phones operating in Europe, 75% of these being concentrated in U.K. and Scandinavia. It is expected that the number of mobile radio users in Europe to reach 10-20 million or even more. There are currently nine types of cellular-telephone systems operating in 17 European countries, most of them incompatible with each other [1,3].
The main objective of the new Pan-European system can be summarized as follows [3];

- To define a common air interface to allow the subscriber to make and receive calls all over Europe. A common system specification with defined interfaces allows exchange of equipment from different manufacturers and will give the users of the system as well as the network operators a large number of potential suppliers.

- To achieve significantly better spectrum efficiency than for the first generation system so as to overcome the problems of overload and congestion, especially in larger cities.

- To be competitive in terms of functionality, performance and not the least cost, since it would have to be introduced in parallel to existing and well established first generation system.

- To allow for the design and use of cheap, compact and efficient handheld terminals. This lead to rather severe limitations on the allowed power consumption and the implementation complexity that could be allowed for the speech codec and the associated speech functions.

**Development of the System**

During the early stages of the study (in 1986) it became clear that a digital system would offer a number of advantages compared to an analogue solution;

- A digital system can tolerate higher co-channel interference levels, partly due to the fact that effective forward error protection techniques may be applied to increase error robustness.

- The evolution of digital transmission in the PSTN and the expanding penetration of the Integrated Services Digital Network (ISDN) in the European countries by the early 90's was another driving force. It is hoped that the ISDN services could be extended to mobile subscribers.

- A digital system will provide a more secure communication link in two respects: i) with respect to listening to conversations (privacy) and (ii) unauthorized access and use of the system (non-paying user).
Since 1986, many European telecommunication administrations and private industries have undertaken a considerable development effort to establish the system specifications. Although, the system has provisions for a wide range of data services, but like many other communication systems, the transmission of the speech is the dominant service in the system. It was therefore necessary to devote a considerable amount of time and effort to develop the two basic speech functions:

- a low bit rate speech codec,
- a scheme for voice activated transmission intended to increase the spectrum efficiency even further and to save battery power in handheld portable terminals [3].

In this chapter, an overview of the major components of the system will first be considered. The theory and implementation of the selected speech coding algorithm will then be reported. The coder was implemented on the AT&T DSP32 which is a floating-point Digital Signal Processor. The GSM recommendations are based on fixed-point implementation and this work was carried out as part of a "Test-Bed" for British Telecom Research Laboratories (BTRL). The final part of this chapter considers the theory and implementation of a Voice Activity Detection (VAD) algorithm, which at the time of carrying out this work was one of the candidate algorithms. The GSM later decided to use a more efficient VAD algorithm which was proposed by BTRL.

5.2 Overview of the System

5.2.1 The Access Scheme

The Time Domain Multiple Access (TDMA) scheme is chosen as the radio access strategy. Eight "Full-Rate Channels" are time multiplexed into each radio channel; each radio channel supports a bit stream of 271 kb/s which is partitioned into time frames each consisting of eight time slots. Figure 5.1, illustrates the composition of the basic TDMA frame and the time slot [2,3].

Figure 5.2, shows the mapping of the speech traffic channel on the physical channel. In order to support various monitoring and control functions, such as channel observation for handover operations, each speech Traffic Channel (TCH) is associated with a so-called Slow Associated Control Channel (SACCH), transmitted over the same
Figure 5.1
Basic TDMA Frame, Time Slot, and Burst Structures [3]

Figure 5.2
Mapping of Traffic Channels on the Physical Channel [3]
physical user channel. Part of the frame is left unused in preparation for a future speech channel using half the bit rate of the current one (see part B of figure 5.2). Two TCH/SACCH combinations will later be accommodated without any change to the system corresponding to a bit rate of 11.4 kb/s per channel. Work has already started throughout Europe to develop a speech codec capable of offering a comparable performance at 7-8 kb/s. This will effectively double the traffic capacity of the system.

5.2.2 Speech Coding, Error Protection and Interleaving

The transmission path of a mobile radio system may suffer from transmission error probabilities of up to $10^{-1}$, due to rapid fluctuations in the multi-path propagation conditions. Typically, the errors will occur in bursts. The GSM system employs three basic techniques to minimize the subjective effects of the transmission errors: interleaving, error correction and error detection. Speech and channel coding are embedded in the digital transmission chain as shown in figure 5.3. The speech coder [6,7], produces 260 bits every 20 ms, resulting in a bit rate of 13 kb/s. These bits are organized into 3 classes of importance according to their sensitivity to transmission errors and thus given different protection. A half-rate convolutional code is applied to the 50 most important bits whereas the less important ones are left unprotected. In addition, a small block code is applied to the 50 most sensitive bits for error detection. This will enable the channel decoder to provide to the speech decoder certain reliability information (R; see figure 5.3) for each frame. In case of detected but uncorrected severe transmission errors, the complete speech frame is discarded. A prediction based on the previous frame has been found to be subjectively more acceptable. A certain Channel State Information (CSI), such as instantaneous field strength, can also be used to increase the error correction capability [6].

![Fig. 5.3 Digital Transmission Link](image)

CSI = Channel State Information ; R = Reliability

Fig. 5.3 Digital Transmission Link [6]
The encoding process results in a speech frame ready for transmission consisting of 456 bits, corresponding to a raw bit rate per channel of 22.8 kb/s. In order to "randomize" the effects of the bursts of errors, an interleaving (semi-diagonal) scheme over 8 time slots is applied. The transmission delay corresponds to the duration of two speech frames. In other words, each burst carries information for two speech frames [2].

5.2.3 Network Organization

The main units of the Digital Mobile Radio (DMR) system are shown in figure 5.4. It mainly consists of:

- Mobile Station (MS): Speech encoding and decoding,
- Radio Sub-System: Channel coding, interleaving and deinterleaving, radio access, modulation and demodulation, and error correction,
- Mobile Switching Centre (MSC).

In a cellular network, the spectrum efficiency or capacity of the network is obtained by reusing frequencies in defined areas (cells) with a defined distance between them. A cell is controlled by a Base Transceiver System (BTS) and several BTSs may be under the management of a Base Station Controller (BSC). The BSC and its BTSs constitute a Base Station System (BSS), which is connected to a Mobile Switching Centre (MSC). The MSC controls one or several BSSs and mainly performs normal switching functions. The MSC is very similar to an ISDN exchange, and is a point of interconnection of the GSM system to the ISDN/PSTN.

In GSM, there are 4 types of handover;

(i) within a BTS (frequency change),
(ii) between two BTSs within a BSC,
(iii) between 2 BSCs within an MSC,
(iv) and between 2 MSCs.

This allows the mobile station to make and receive calls anywhere in the GSM service area without any special actions taken by the mobile subscriber.
Fig. 5.4 Block Diagram of the Digital Mobile Radio System [3]
5.3 Speech Coding Algorithm

Background

The speech coder is a crucial component of a digital mobile radio (DMR) system. The speech quality it provides over the wide range of conditions under which it must operate largely determines the user's perception of the whole system. As a public cellular radio system, the GSM system will interface to the national telephone network (at the MSC) and as such must be regarded as part of the international telephone network. This fact resulted in certain requirements (and constraints) in the design of the GSM speech coder. In October 1985, a GSM specialized sub-group, the Speech Coding Experts Group (SCEG), was established to coordinate the European research program towards a suitable digital speech coding scheme [7]. The SCEG decided that the speech codec should meet the following performance objectives [7-9]:

- **Speech Quality**: As a minimum requirement, the overall quality of the codec shall, on average, have better speech performance than companded FM analog systems in the 900 MHz band currently in operation in Western Europe;

- **Bit Rate**: For the DMR system to make efficient use of the available radio spectrum the bit rate for the speech coder should not exceed 16 kb/s, the aim being to offer a capacity superior to that of existing system;

- **Delay**: The speech codec algorithm delay, interleaving delay, speech codec processing delay and other delay of the radio subsystems, all contribute towards the overall delay in a DMR system. To minimize problems associated with long delays such as echoes, the back-to-back delay of the speech coder shall not exceed 65 ms;

- **Error Robustness**: The speech quality should not degrade noticeably with a bit error ratio (BER) of $10^{-3}$, and intelligibility should be maintained up to a BER of $10^{-2}$.

Many other requirements [10], such as robustness to environmental noise, wide dynamic range, wide range of speakers, little degradation on tandeming, low complexity and power consumption, were also placed on the design of the speech coder. Transmission of voice band data through the codec is believed to reduce the speech
quality. Voice band data transmission will, therefore, be provided by the system by means of specialized terminal adaptors, i.e. bypassing the codec [8].

Initially, more than 20 speech coders were proposed for use in the DMR system [11-18]. In order to reduce this number to a manageable value, allowing formal listening tests to be made, SCEG decided to limit the number of candidate coders to 6, a maximum of one per country. As a result of national tests and selections, six coders were elected. The features and performance of each coder are summarized in Table 5.1. The selected coders can be grouped into two classes:

1) Two Pulse Excited Coders:
   . a multipulse excited coder with long term prediction (MPE-LTP), and
   . a simplified regular pulse excited LPC coder (RPE-LPC).

2) Four Subband Coders:
   . three variants of subband coders with block forward adaptive PCM coding of subband signals (SBC-APCM), and
   . one subband coder using backward adaptive ADPCM coding of subband signals (SBC-ADPCM).

As a result of many extensive subjective tests conducted in several different laboratories, by subjects listening to their native language, the GSM finally decided to work on a compromise solution, based on the two pulse excited coders. The French-German hybrid coder is known as RPE-LTP, as it is based on the original Regular-Pulse Excitation coder (RPE-LPC) which has been modified by adding a Long-Term Prediction (LTP) loop. This modification reduced the net bit rate of the original coder to 13 kb/s but increased the computational complexity and memory allocation by 40 percent [19]. The compromise coder, at 13 kb/s, achieves a speech quality equivalent to that of RPE-LPC at 14.8 kb/s in error free conditions. The inclusion of the LTP loop results in a minor increase in the sensitivity to errors. However, if the bit rate reduction of 1.8 kb/s is utilized for additional error protection, a net increase in error robustness can be assumed [7].

In the following sections, the original and the compromise coding schemes will be reviewed. The floating-point implementation of the compromise coder will then be presented. This work was carried out in early 1988, based on the documentation available and there may have been some slight changes of some of the parameters since then.
<table>
<thead>
<tr>
<th>Country</th>
<th>Coding Scheme</th>
<th>Average M.O.S.</th>
<th>Net Bit Rate (kb/s)</th>
<th>Gross Bit Rate (kb/s)</th>
<th>FEC Scheme</th>
<th>Delay*</th>
<th>Frame Length</th>
<th>Memory Requirements (Bytes)</th>
<th>Computational Complexity (MOPS)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>France</td>
<td>MPE-LTP</td>
<td>3.27</td>
<td>13.2</td>
<td>16</td>
<td>Extended Hamming</td>
<td>36 ms</td>
<td>20 ms</td>
<td>25.88 k</td>
<td>4.9 M</td>
<td>8th order LPC Analysis with long-term prediction</td>
</tr>
<tr>
<td>F.R.G.</td>
<td>RPE-LPC</td>
<td>3.54</td>
<td>14.77</td>
<td>16</td>
<td>Reed Solomon</td>
<td>&lt; 40 ms</td>
<td>19.5 ms</td>
<td>10.1 k</td>
<td>1.5 M</td>
<td>12th order LPC Analysis overlapping Hamming Window</td>
</tr>
<tr>
<td>Italy</td>
<td>SBC-APCM</td>
<td>2.98</td>
<td>15</td>
<td>16</td>
<td>Golay + Hamming</td>
<td>&lt; 40 ms</td>
<td>15 ms</td>
<td>8.44 k</td>
<td>1.2 M</td>
<td>8 sub-bands using Max Quantizers</td>
</tr>
<tr>
<td>Norway</td>
<td>SBC-APCM</td>
<td>2.46</td>
<td>15</td>
<td>16</td>
<td>BCH Truncated</td>
<td>&lt; 45 ms</td>
<td>20 ms</td>
<td>14.32 k</td>
<td>1.4 M</td>
<td>16 sub-bands using Max Quantizers</td>
</tr>
<tr>
<td>Sweden</td>
<td>SBC-APCM</td>
<td>3.14</td>
<td>13</td>
<td>16</td>
<td>Extended Golay</td>
<td>35 ms</td>
<td>16 ms</td>
<td>7.7 k</td>
<td>1.5 M</td>
<td>16 sub-bands using Max Quantizers</td>
</tr>
<tr>
<td>U.K.</td>
<td>SBC-ADPCM</td>
<td>2.92</td>
<td>15</td>
<td>15</td>
<td>None</td>
<td>7 ms</td>
<td>1 ms</td>
<td>5.3 k</td>
<td>1.9 M</td>
<td>8 sub-bands using Forward ADPCM</td>
</tr>
<tr>
<td>Reference</td>
<td>Companded FM</td>
<td>1.95</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

* (assuming serial transmission)

Table 5.1 Performance and Features of the Six Candidate Coders for DMR System
5.3.1 Algorithm Description of Original Coder (RPE-LPC)

The initial coder chosen is the "Regular Pulse Excitation - Linear Predictive Coder" (RPE-LPC), which is closely related to the "Multi Pulse Excitation - LPC" (MPE-LPC). The difference being that in an MPE coder the prediction residual is basically replaced by an optimized sequence consisting of a reduced number of samples i.e. combined optimization of pulse amplitudes and pulse positions whereas in the RPE coder the position of the pulse amplitudes is restricted to a regular grids and thus only the pulse amplitudes have to be optimized for a few different grid positions [20].

Figure 5.5 shows the block diagram of the simplified RPE coder designed for a gross transmission rate of 16 kb/s (speech coding: 14.77 kb/s; error protection: 1.23 kb/s). It is based on 8 kHz sampling and non-linear quantization of the input and output samples according to A-law companding rules. The error protection scheme is based on the Reed-Solomon code. The analysis section of the coder (see Figure 5.5a) consists of the following five subblocks:

- segmentation, including windowing and normalization,
- computation of the autocorrelation function (ACF),
- linear prediction analysis by Schur recursion algorithm,
- transformation of the reflection coefficients using approximated log.-area-ratios (LAR),
- quantization of the transformed coefficients.

The reflection coefficients as well as the approximated LARs have different dynamic ranges and different asymmetric amplitude distributions. For this reason the LAR coefficients are quantized individually with different uniform quantizers [16].

In order to avoid the unwanted transients of the residual signal due to abrupt changes of the filter coefficients, linear interpolation is applied in the transition between two successive parameter sets. Each residual frame of length K is divided into four subsegments consisting of K/4 samples. These subsegments are then applied to an 11-tap Finite Impulse Response (FIR) weighting filter. The conventional convolution of a
Figure 5.5 Block Diagram of the RPE-LPC Coder
<table>
<thead>
<tr>
<th>Coding Scheme</th>
<th>RPE-LPC (Regular Pulse Excitation - LPC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling Frequency</td>
<td>8 kHz</td>
</tr>
<tr>
<td>A/D and D/A Conversion</td>
<td>8 bit PCM A-law (CCITT G.711 and G.712)</td>
</tr>
<tr>
<td>Block Lengths</td>
<td></td>
</tr>
<tr>
<td>Analysis Window</td>
<td>24.375 ms or 195 Sample Periods</td>
</tr>
<tr>
<td>Signal Frame</td>
<td>19.500 ms or 156 Sample Periods</td>
</tr>
<tr>
<td>RPE-Subsequence</td>
<td>4.875 ms or 39 Sample Periods</td>
</tr>
<tr>
<td>Number of Subsequences</td>
<td>4</td>
</tr>
<tr>
<td>Sample Rate Decimation</td>
<td>3</td>
</tr>
<tr>
<td>Number of RPE-Samples</td>
<td>13 * 4</td>
</tr>
<tr>
<td>Order of Predictor</td>
<td>12</td>
</tr>
<tr>
<td>Order of Weighting Filter</td>
<td>10</td>
</tr>
<tr>
<td>Error Protection</td>
<td>Reed Solomon Codes</td>
</tr>
<tr>
<td></td>
<td>(N, K) Codes: N Bits, K Information Symbols</td>
</tr>
<tr>
<td></td>
<td>(15, 9) Code for protection of the 36 most important bits</td>
</tr>
<tr>
<td></td>
<td>No protection of the remaining 252 bits.</td>
</tr>
<tr>
<td>Bit Allocation</td>
<td>(Bits per Frame)</td>
</tr>
<tr>
<td>12 LOG.-Area Ratios</td>
<td>52</td>
</tr>
<tr>
<td>52 RPE Samples</td>
<td>204</td>
</tr>
<tr>
<td>4 Block Amplitudes</td>
<td>24</td>
</tr>
<tr>
<td>4 RPE Positions</td>
<td>8</td>
</tr>
<tr>
<td>Error Protection</td>
<td>24</td>
</tr>
<tr>
<td>Total</td>
<td>312</td>
</tr>
<tr>
<td>Bit Rate</td>
<td></td>
</tr>
<tr>
<td>Speech</td>
<td>14.77 kbit/s (288 Bits per Frame)</td>
</tr>
<tr>
<td>Error Protection</td>
<td>1.23 kbit/s (24 Bits per Frame)</td>
</tr>
<tr>
<td>Total</td>
<td>16.00 kbit/s (312 Bits per Frame)</td>
</tr>
<tr>
<td>Theoretical Delay</td>
<td>(with infinite fast processing and transmission)</td>
</tr>
<tr>
<td>Speech Codec</td>
<td>25 ms (24.375 exactly)</td>
</tr>
<tr>
<td>Channel Codec</td>
<td>0 ms</td>
</tr>
<tr>
<td>Interleaving</td>
<td>15 ms (Serial Transmission)</td>
</tr>
<tr>
<td>Additional Delay</td>
<td>(Due to Actual Realization)</td>
</tr>
<tr>
<td>Speech Codec</td>
<td>8 ms</td>
</tr>
<tr>
<td>Channel Codec</td>
<td>8 ms</td>
</tr>
<tr>
<td>Total Delay</td>
<td>56 ms (measured analogue in to analogue out)</td>
</tr>
</tbody>
</table>

Table 5.2 Summary of Basic Features of RPE-LPC Coder [22]
sequence having \( K/4 = 39 \) samples with the impulse response of length \( n=11 \) would result in a segment of \((K/4)+n-1 = 49\) samples. Due to the fact that the RPE optimization is applied individually to the subsegments, a "block filtering" operation is used which produces the \( K/4=39 \) central samples of the conventional operation. The filtered segments, \( x(t) \), is then decomposed into 3 interleaved regular pulse candidate sequences of length \( K/12 \) i.e. a decimation factor of 3. According to the explicit solution of the RPE mean squared error criterion the optimum candidate sequence, \( X_m(l) ; m=0,1,2 \) denotes the position of the decimation grid, is selected which is the one with the maximum energy. The selected sequence is then quantized by block adaptive PCM (APCM). Each block of \( K/12 \) samples is normalized by the block maximum. The scaled samples are quantized with four bits using non-uniform Max quantizers, whereas the block maximum is coded logarithmically with 6 bits.

At the decoder, the excitation signal which is fed to the synthesis filter is formed by decoding and denormalization of the RPE samples and by placing them in the correct temporal position, \( M \) (grid). At this stage the sample rate is increased by a factor of 3 by inserting 2 zero-valued samples in between the RPE samples. Prior to the speech encoder, after A-law decoding and down scaling by a factor of 0.5, the speech signal is applied to a first order FIR pre-emphasis filter, with a prediction factor of 0.86. Therefore, the output of the synthesis filter is de-emphasized using an IIR filter and then scaled up by 2. Finally, Table 5.2 summarizes the basic features of the RPE-LPC coding scheme.

### 5.3.2 Algorithm Description of Compromise Coder

As figure 5.6 shows, the modified coder is based on the analysis-by-synthesis scheme and can be divided into five major sub-sections:

- Pre-processing,
- LPC analysis,
- Short-term filtering,
- Long-term prediction,
- RPE encoding.

The major difference between the original and the compromise coders are as follows:
Figure 5.6 Block Diagram of the RPE-LTP Encoder
Figure 5.7 Block Diagram of the RPE-LTP Decoder
In the pre-processing stage, a notch filter is added in order to remove the offset of the input speech signal.

In the LPC analysis stage, the prediction order is reduced from 12 to 8, and non-overlapping frames of 160 samples are used.

The long-term correlations are removed from the residual signal prior to decimation and quantization.

A feedback loop is formed to reconstruct an estimate of the first residual signal.

The first residual signal is found in the same manner as explained in previous section with the above exceptions. It is then segmented into four sub-segments, each of 5 ms duration, and a long-term correlation lag, and an associated gain factor is found for each sub-segment. The determination of these two parameters is implemented in three steps [22]:

1) The current sub-segment, \( d(i) \), is cross-correlated with the previous 120 samples of the reconstructed residual signal, \( d'(i) \):

\[
R(p) = \sum_{i=0}^{39} d(i) \cdot d'(i - p) \quad ; \quad p = 40, \ldots, 120
\]  

(5.1)

The cross-correlation is evaluated for lags, \( p \), greater than or equal to 40 and less than or equal to 120 i.e. corresponding to samples outside the current sub-segment and not delayed by more than two sub-segments.

2) The position, \( N \), of the peak of the cross-correlation function within this interval is searched for:

\[
R(N) = \max [R(p)] \quad ; \quad p = 40, \ldots, 120
\]  

(5.2)

3) The gain factor, \( b \), is evaluated according to:

\[
b = \frac{R(N)}{S(N)}
\]  

(5.3)

where
<table>
<thead>
<tr>
<th>Coding Scheme</th>
<th>RPE-LTP (Regular Pulse Excitation - LTP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling Frequency</td>
<td>8 kHz</td>
</tr>
<tr>
<td>A/D and D/A Conversion</td>
<td>8 bit PCM A-law (CCITT G.711 and G.712)</td>
</tr>
<tr>
<td>Block Lengths</td>
<td></td>
</tr>
<tr>
<td>Analysis Window</td>
<td>20.0 ms or 160 Sample Periods</td>
</tr>
<tr>
<td>Signal Frame</td>
<td>20.0 ms or 160 Sample Periods</td>
</tr>
<tr>
<td>RPE-Subsequence</td>
<td>5.0 ms or 40 Sample Periods</td>
</tr>
<tr>
<td>Number of Subsequences</td>
<td>4</td>
</tr>
<tr>
<td>Sample Rate Decimation</td>
<td>52 out of 160 samples</td>
</tr>
<tr>
<td>Number of RPE-Samples</td>
<td>13 * 4</td>
</tr>
<tr>
<td>Order of Predictor</td>
<td>8</td>
</tr>
<tr>
<td>Order of Weighting Filter</td>
<td>10</td>
</tr>
<tr>
<td>Bit Allocation</td>
<td>(Bits per Frame)</td>
</tr>
<tr>
<td>8 LOG.-Area Ratios</td>
<td>36</td>
</tr>
<tr>
<td>52 RPE Samples</td>
<td>156</td>
</tr>
<tr>
<td>4 LTP Gains</td>
<td>8</td>
</tr>
<tr>
<td>4 LTP Delays</td>
<td>28</td>
</tr>
<tr>
<td>4 Block Amplitudes</td>
<td>24</td>
</tr>
<tr>
<td>4 RPE Positions</td>
<td>8</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>260</strong></td>
</tr>
<tr>
<td>Bit Rate</td>
<td>13.0 kbit/s (260 Bits per Frame)</td>
</tr>
<tr>
<td>Theoretical Delay</td>
<td>(with infinite fast processing and transmission)</td>
</tr>
<tr>
<td>Speech Codec</td>
<td>20 ms</td>
</tr>
</tbody>
</table>

Table 5.3 Summary of Basic Features of RPE-LTP Coder [22]
\[ S(N) = \sum_{i=0}^{39} d^2(i-N) \] (5.4)

The second residual signal, \( e(k) \), is found by subtracting the output of the long-term analysis filter, \( d''(k) \), from the first residual signal, \( d(k) \):

\[ e(k) = d(k) - d''(k) ; \quad k = 0, ..., 39 \] (5.5)

where the output of the long-term analysis filter is computed from previously reconstructed residual samples, \( d'(k) \), adjusted to the current sub-segment LTP lag, \( N \), and weighted by the quantized sub-segment LTP gain, \( b' \):

\[ d(k) = b'. d'(k - N) ; \quad k = 0, ..., 39 \] (5.6)

The baseband part of the second residual is quantized and transmitted to the decoder. The reconstructed second residual signal, \( e'(k) \), is then added to the output of the long-term filter, \( d''(k) \), to give the reconstructed first residual signal, \( d'(k) \):

\[ d'(k) = e'(k) + d''(k) ; \quad k = 0, ..., 39 \] (5.7)

In the decoder the excitation signal, \( d'(k) \), is found exactly as in the encoder and then applied to the short-term synthesis filter (see Figure 5.7). Finally, the output of the synthesis filter is fed into the Infinite Impulse Response (IIR) de-emphasis filter leading to the output speech signal. Table 5.3 summarizes the basic features of the RPE-LTP coding algorithm.

5.4 Software Implementation of RPE-LTP Encoder

Both the original and the compromise coders were simulated using software written in the C-language. However, only the RPE-LTP coder was real-time implemented using the first generation of the AT&T floating-point digital signal processors (DSP), the WE-DSP32 [26]. The DSP32 is capable of multi operations in single instructions due to its pipelining architecture and performs 6.25 million instructions per second (i.e. instruction cycle of 160 ns). After it was verified that the simulated programmes were error-free and produced the expected results, they were used to write the corresponding DSP32 code. Two DSP32s were used for encoding and
Figure 5.8 Flow Chart of Speech Encoding Process
decoding. In this section the implementation of main stages of the encoder will be discussed. The processing power as well as memory allocation for each stage of the encoding process are also reported.

Figure 5.8 shows the flow chart of the speech encoding program. The speech coding algorithms that require LPC analysis are classified as block processing algorithms. This usually means that the processing is performed on an array of input data rather than on individual input samples. The real-time implementation of this algorithm requires a dual input and output buffering system. This is usually achieved by employing the DMA (Direct Memory Access) techniques.

5.4.1 Dual Input and Output Buffering System

The DMA controlled serial input and output allows the transfer of data between external interfaces, such as analogue-to-digital (A/D) or digital-to-analogue (D/A) converters, and memory without programme intervention. By operating in this mode, the programme may operate on a previously stored block of data while the current samples are automatically stored in another. The main constraint is that the process time of the block of data must be less than the time required to fill the buffer, if the system is to operate successfully in real-time. The same procedure applies to the output buffers (see Figure 5.9). This means that two input and two output buffers are needed. In DSP32, two special purpose registers in the CAU, r20 (PIN) and r21 (POUT), are used as pointers for the DMA transfers. PIN, input pointer, is used as the serial input DMA pointer, and POUT, output pointer, is used as the serial output DMA pointer [23]. Each time the processing of one buffer is finished, the programme has to check to see if the input pointer (i.e. r20 register) has reached the end of the other buffer. If this is the case, swap the tasks (ping-pong effect), otherwise wait till the buffer is filled up and then swap processing and storing tasks (see Figure 5.10).

5.4.2 Pre-Processing of Input Speech

For reasons that will be discussed in section 5.4.8, it was necessary to normalize the input speech samples to ± 1.0, after A-law to float conversion. It was found that the extreme values of the A-law code correspond to ± 4096.0 [24]. Each input sample was therefore divided by this value as well as constant factor, a = 0.079. The former normalization is necessary for fixed-point implementation and can be avoided for
Figure 5.9 Dual Input and Output Buffering System
Figure 5.10 Flow Chart of Dual Input/Output Buffering System
floating-point implementation. However, in our implementation, we tried to follow the documentation as closely as possible. The normalized input array was then pre-emphasized and stored for next processing stage.

5.4.3 Autocorrelation Function

An 8th order LPC analysis requires the first 9 values of the autocorrelation function. These are calculated by:

\[
ACF(k) = \sum_{i=0}^{159-k} S(i) \cdot S(i+k) \quad ; \quad k = 0, ..., 8
\]

(5.8)

where \( S(i) \) are the speech samples.

5.4.4 Schur Recursion

The reflection coefficients are calculated as shown in Figure 5.11, using the Schur recursion algorithm. The term "reflection coefficient" comes from the theory of linear prediction of speech, where a vocal tract representation consisting of a series of uniform cylindrical sections is assumed. Such a representation can be described by the reflection coefficients or the area ratios of connected sections [22].

5.4.5 Log.-Area-Ratio (LAR) Transformation

In order to efficiently quantize the reflection coefficients, different transformations are used. One of these representations is the log.-area-ratio (LAR) transformation which is defined as:

\[
LAR(i) = \log_{10} \left[ \frac{1 + r(i)}{1 - r(i)} \right]
\]

(5.9)

The real-time implementation of this transformation will require 8 divisions and 8 calls of the logarithmic subroutines which would need 600 operations and consequently 0.096 ms (160 ns/instruction) of processing power. The following segmented approximation is therefore used in which it is merely necessary to multiply, add, and compare these values:
Schur Recursion

\[ K(9-i) = ACF(i); \quad i = 1 \ldots 7 \]
\[ P(j) = ACF(j); \quad j = 0 \ldots 8 \]

\[ n = 1 \]

\[ P(0) \leq |P(1)|? \]
- Yes: \( r(i) = 0; \quad i = n \ldots 8 \)
- No:
  - If \( P(0) \geq 0.5 \):
    - Yes: \( r(n) = \frac{|P(1)|}{P(0)} \)
    - No: \( P(0) = 2 \cdot P(0) \)
    - If \( |P(1)| = 2 \cdot |P(1)| \)
  - No: \( P(1) > 0 ? \)
    - Yes: \( r(n) = -r(n) \)
    - No: \( n = 8 ? \)
      - Yes: \( P(0) = p(0) + p(1) \cdot r(n) \)
      - No: \( m = 1 \)

\[ P(m) = P(m+1) + r(n) \cdot k(9-m) \]
\[ k(9-m) = k(9-m) + r(n) \cdot p(m+1) \]

\[ n = n + 1 \]
\[ m = 8 - n ? \]
- Yes: \( m = m + 1 \)
- No: \( m = m + 1 \)

Figure 5.11 Flow Chart of Schur Recursive Algorithm [23]
\[
LAR(i) = \begin{cases} 
    r(i) & \text{if } |r(i)| < 0.675 \\
    \text{sign}[r(i)] \cdot [2|r(i)| - 0.675] & \text{if } 0.675 \leq |r(i)| < 0.950 \\
    \text{sign}[r(i)] \cdot [8|r(i)| - 6.375] & \text{if } 0.950 \leq |r(i)| < 1.000 
\end{cases}
\]

(5.10)

Equation 5.11 gives the inverse transformation:

\[
r'(i) = \begin{cases} 
    LAR(i) & \text{if } |LAR'(i)| < 0.675 \\
    \text{sign}[LAR'(i)] \cdot [0.500|LAR'(i)| + 0.337500] & \text{if } 0.675 \leq |LAR'(i)| < 1.220 \\
    \text{sign}[LAR'(i)] \cdot [0.125|LAR'(i)| + 0.796875] & \text{if } 1.225 \leq |LAR'(i)| < 1.625 
\end{cases}
\]

(5.11)

The approximated LARs are then quantized using different quantizers because of different dynamic ranges and distribution densities. Equations 5.12 and 5.13 were used for quantization and decoding, respectively. The determination of coefficients \(a_i\) and \(b_i\) were based on the analysis of real speech signals and subjective tests.

\[
c(i) = \text{Nint}[a_i \cdot LAR(i) + b_i] 
\]

(5.12)

\[
LAR'(i) = \frac{c(i) - b_i}{a_i} 
\]

(5.13)

Function "Nint" defines the rounding to the nearest integer value and \(c(i)\) is the code value. Table 5.4 shows the values of coefficients \(a_i\), \(b_i\) and the bit assignment of each LAR.

<table>
<thead>
<tr>
<th>LAR No.</th>
<th>(a_i)</th>
<th>(b_i)</th>
<th>Minimum (c(i))</th>
<th>Maximum (c(i))</th>
<th>Bits / LAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20.000</td>
<td>0.000</td>
<td>-32</td>
<td>+31</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>20.000</td>
<td>0.000</td>
<td>-32</td>
<td>+31</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>20.000</td>
<td>4.000</td>
<td>-16</td>
<td>+15</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>20.000</td>
<td>-5.000</td>
<td>-16</td>
<td>+15</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>13.637</td>
<td>0.184</td>
<td>-8</td>
<td>+7</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>15.000</td>
<td>-3.500</td>
<td>-8</td>
<td>+7</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>8.334</td>
<td>-0.666</td>
<td>-4</td>
<td>+3</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>8.824</td>
<td>-2.235</td>
<td>-4</td>
<td>+3</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5.4 Quantization of the Approximated LARs [23]
5.4.6 Short-Term Inverse Filter

High order direct-form filters are very sensitive to quantization, particularly as to how much the filter poles deviate from their ideal positions in response to quantization of coefficients, whereas lattice filters have good quantization properties [25].

\[
\begin{align*}
LAR_j^i(i) &= 0.75 \cdot LAR_{j-1}^i(i) + 0.25 \cdot LAR_j^i(i) \\
&= 0.50 \cdot LAR_{j-1}^i(i) + 0.50 \cdot LAR_j^i(i) \\
&= 0.25 \cdot LAR_{j-1}^i(i) + 0.75 \cdot LAR_j^i(i) \\
&= LAR_j^i(i)
\end{align*}
\]

Table 5.5 Interpolation of LAR Parameters [23]

To avoid spurious transients which may occur if the filter coefficients are changed abruptly, two subsegment sets of LARs are linearly interpolated, as given in Table 5.5. The interpolated LARs are then converted back to reflection coefficients after quantization and used in the lattice structure of Figure 5.12, to produce the prediction error (first residual signal).

Figure 5.12 Structure of Short-Term Inverse Filter
5.4.7 LTP Analysis

The LTP analysis and synthesis is implemented exactly as explained in section 5.3.2. The LTP lags and gains were coded with 7 bits and 2 bits each, respectively, according to Table 5.6 and the following algorithm:

\[ [b_j] = \begin{cases} 
1 & \text{if } DLB(l-1) < b_j \leq DLB(l) \\
0 & \text{if } b_j \leq DLB(0) \\
3 & \text{if } b_j > DLB(2) 
\end{cases} \] (5.14)

<table>
<thead>
<tr>
<th></th>
<th>Decision Level DLB (l)</th>
<th>Quantization Level QLB (l)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
<td>0.10</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.35</td>
</tr>
<tr>
<td>2</td>
<td>0.8</td>
<td>0.65</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 5.6 Quantization of the LTP Gain [22]

5.4.8 Quantization of RPE Sequences

Adaptive Pulse Code Modualtion (APCM) is used to quantize the 13 samples within the RPE sequence. The pipelined execution of the DSP32 instructions result in several latency effects, one being a Data-Arithmetic-Unit (DAU) conditions tested by a conditional branch instruction should be established by the last DAU instruction four instructions prior to the test. This, however, does not apply to Control-Arithmetic-Unit (CAU) instructions which use 16-bit registers. If look-up tables are to be used for quantization of the RPE sequence, which means comparing the input samples with different threshold levels, a much smaller processing time is needed. This requires the floating-point data to be converted into integer numbers (i.e. 16 bits) and the CAU to be used. The look-up tables given in the GSM documentation (Tables 5.7 and 5.8), assumes 16-bit implementation, therefore it was necessary to normalize the input speech samples to ± 1.0 in the pre-processing stage.
<table>
<thead>
<tr>
<th>$I_{x'}$ (Interval limits)</th>
<th>$I_{\hat{x}'}$</th>
<th>$I_{xn}$ (Coded Value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-32768 ... -24576</td>
<td>-28672</td>
<td>0 = 000</td>
</tr>
<tr>
<td>-24577 ... -16384</td>
<td>-20480</td>
<td>1 = 001</td>
</tr>
<tr>
<td>-16385 ... -8192</td>
<td>-12288</td>
<td>2 = 010</td>
</tr>
<tr>
<td>-8191 ... 0</td>
<td>-4096</td>
<td>3 = 011</td>
</tr>
<tr>
<td>1 ... 8192</td>
<td>4096</td>
<td>4 = 100</td>
</tr>
<tr>
<td>8193 ... 16384</td>
<td>12288</td>
<td>5 = 101</td>
</tr>
<tr>
<td>16385 ... 24576</td>
<td>20480</td>
<td>6 = 110</td>
</tr>
<tr>
<td>24577 ... 32767</td>
<td>28672</td>
<td>7 = 111</td>
</tr>
</tbody>
</table>

*Table 5.7 Quantization of the Normalized RPE Samples [23]*
<table>
<thead>
<tr>
<th>$I_{x_{\text{max}}}$</th>
<th>$I^\text{c}<em>{x</em>{\text{max}}}$</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ... 31</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>32 ... 63</td>
<td>63</td>
<td>1</td>
</tr>
<tr>
<td>64 ... 95</td>
<td>95</td>
<td>2</td>
</tr>
<tr>
<td>96 ... 127</td>
<td>127</td>
<td>3</td>
</tr>
<tr>
<td>128 ... 159</td>
<td>159</td>
<td>4</td>
</tr>
<tr>
<td>160 ... 191</td>
<td>191</td>
<td>5</td>
</tr>
<tr>
<td>192 ... 223</td>
<td>223</td>
<td>6</td>
</tr>
<tr>
<td>224 ... 255</td>
<td>255</td>
<td>7</td>
</tr>
<tr>
<td>256 ... 287</td>
<td>287</td>
<td>8</td>
</tr>
<tr>
<td>288 ... 319</td>
<td>319</td>
<td>9</td>
</tr>
<tr>
<td>320 ... 351</td>
<td>351</td>
<td>10</td>
</tr>
<tr>
<td>352 ... 383</td>
<td>383</td>
<td>11</td>
</tr>
<tr>
<td>384 ... 415</td>
<td>415</td>
<td>12</td>
</tr>
<tr>
<td>416 ... 447</td>
<td>447</td>
<td>13</td>
</tr>
<tr>
<td>448 ... 479</td>
<td>479</td>
<td>14</td>
</tr>
<tr>
<td>480 ... 511</td>
<td>511</td>
<td>15</td>
</tr>
<tr>
<td>512 ... 575</td>
<td>575</td>
<td>16</td>
</tr>
<tr>
<td>576 ... 639</td>
<td>639</td>
<td>17</td>
</tr>
<tr>
<td>640 ... 703</td>
<td>703</td>
<td>18</td>
</tr>
<tr>
<td>704 ... 767</td>
<td>767</td>
<td>19</td>
</tr>
<tr>
<td>768 ... 831</td>
<td>831</td>
<td>20</td>
</tr>
<tr>
<td>832 ... 895</td>
<td>895</td>
<td>21</td>
</tr>
<tr>
<td>896 ... 959</td>
<td>959</td>
<td>22</td>
</tr>
<tr>
<td>960 ... 1023</td>
<td>1023</td>
<td>23</td>
</tr>
<tr>
<td>1024 ... 1151</td>
<td>1151</td>
<td>24</td>
</tr>
<tr>
<td>1152 ... 1279</td>
<td>1279</td>
<td>25</td>
</tr>
<tr>
<td>1280 ... 1407</td>
<td>1407</td>
<td>26</td>
</tr>
<tr>
<td>1408 ... 1535</td>
<td>1535</td>
<td>27</td>
</tr>
<tr>
<td>1536 ... 1663</td>
<td>1663</td>
<td>28</td>
</tr>
<tr>
<td>1664 ... 1791</td>
<td>1791</td>
<td>29</td>
</tr>
<tr>
<td>1792 ... 1919</td>
<td>1919</td>
<td>30</td>
</tr>
<tr>
<td>1920 ... 2047</td>
<td>2047</td>
<td>31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$I_{x_{\text{max}}}$</th>
<th>$I^\text{c}<em>{x</em>{\text{max}}}$</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>2048 ... 2303</td>
<td>2303</td>
<td>32</td>
</tr>
<tr>
<td>2304 ... 2559</td>
<td>2559</td>
<td>33</td>
</tr>
<tr>
<td>2560 ... 2815</td>
<td>2815</td>
<td>34</td>
</tr>
<tr>
<td>2816 ... 3071</td>
<td>3071</td>
<td>35</td>
</tr>
<tr>
<td>3072 ... 3327</td>
<td>3327</td>
<td>36</td>
</tr>
<tr>
<td>3328 ... 3583</td>
<td>3583</td>
<td>37</td>
</tr>
<tr>
<td>3584 ... 3839</td>
<td>3839</td>
<td>38</td>
</tr>
<tr>
<td>3840 ... 4095</td>
<td>4095</td>
<td>39</td>
</tr>
<tr>
<td>4096 ... 4607</td>
<td>4607</td>
<td>40</td>
</tr>
<tr>
<td>4608 ... 5119</td>
<td>5119</td>
<td>41</td>
</tr>
<tr>
<td>5120 ... 5631</td>
<td>5631</td>
<td>42</td>
</tr>
<tr>
<td>5632 ... 6143</td>
<td>6143</td>
<td>43</td>
</tr>
<tr>
<td>6144 ... 6655</td>
<td>6655</td>
<td>44</td>
</tr>
<tr>
<td>6656 ... 7167</td>
<td>7167</td>
<td>45</td>
</tr>
<tr>
<td>7168 ... 7679</td>
<td>7679</td>
<td>46</td>
</tr>
<tr>
<td>7680 ... 8191</td>
<td>8191</td>
<td>47</td>
</tr>
<tr>
<td>8192 ... 9215</td>
<td>9215</td>
<td>48</td>
</tr>
<tr>
<td>9216 ... 10239</td>
<td>10239</td>
<td>49</td>
</tr>
<tr>
<td>10240 ... 11263</td>
<td>11263</td>
<td>50</td>
</tr>
<tr>
<td>11264 ... 12287</td>
<td>12287</td>
<td>51</td>
</tr>
<tr>
<td>12288 ... 13311</td>
<td>13311</td>
<td>52</td>
</tr>
<tr>
<td>13312 ... 14335</td>
<td>14335</td>
<td>53</td>
</tr>
<tr>
<td>14336 ... 15359</td>
<td>15359</td>
<td>54</td>
</tr>
<tr>
<td>15360 ... 16383</td>
<td>16383</td>
<td>55</td>
</tr>
<tr>
<td>16384 ... 18431</td>
<td>18431</td>
<td>56</td>
</tr>
<tr>
<td>18432 ... 20479</td>
<td>20479</td>
<td>57</td>
</tr>
<tr>
<td>20480 ... 22527</td>
<td>22527</td>
<td>58</td>
</tr>
<tr>
<td>22528 ... 24575</td>
<td>24575</td>
<td>59</td>
</tr>
<tr>
<td>24576 ... 26623</td>
<td>26623</td>
<td>60</td>
</tr>
<tr>
<td>26624 ... 28671</td>
<td>28671</td>
<td>61</td>
</tr>
<tr>
<td>28672 ... 30719</td>
<td>30719</td>
<td>62</td>
</tr>
<tr>
<td>30720 ... 32767</td>
<td>32767</td>
<td>63</td>
</tr>
</tbody>
</table>

Table 5.8 Quantization of the Block Maximum [23]
Flow charts of Figures 5.13 and 5.14 show how the APCM quantizer and decoder were implemented. As the flow chart of the logarithmic quantizer (Fig. 5.14) shows, the implementation is done in such a way that there is no need to store the look-up table and by breaking it down into smaller segments, the execution speed is increased.

**Fig. 5.13 Flow Chart of Quantization of RPE Sequence (ADPCM)**
Logarithmic Quantizer

Is \( X > 511 \)? No → 16 Level Quantizer (step-size = 31)

Yes → Is \( X > 1023 \)? No → 8 Level Quantizer (step-size = 63)

Yes → Is \( X > 2047 \)? No → 8 Level Quantizer (step-size = 127)

Yes → Is \( X > 4095 \)? No → 8 Level Quantizer (step-size = 255)

Yes → Is \( X > 8191 \)? No → 8 Level Quantizer (step-size = 511)

Yes → Is \( X > 16383 \)? No → 8 Level Quantizer (step-size = 1023)

Yes → Is \( X > 32767 \)? No → 8 Level Quantizer (step-size = 2047)

Yes → \( X = 32767 \)

END

Fig. 5.14 Flow Chart of Logarithmic Quantizer
5.4.9 Bit Mapping

The different parameters of the encoded speech and their individual bits have unequal importance with respect to subjective quality. In Table 5.9, the speech coder bits have been grouped into 6 different classes according to their sensitivity to channel errors. Class 1 is the most sensitive and errors in these bits result in unintelligible speech whereas single errors in the class 3 bits will result in moderate distortion. This ranking has been determined by subjective testing.

By dividing the bits into different classes it is possible to protect the most sensitive bits against channel errors with more bits and the least sensitive ones with less, or none at all. Since all the encoded parameters are stored in integer format, a subroutine is needed to fetch one bit at a time from each parameter and place it in a binary string in the appropriate order. The most efficient way of implementing this was to index each parameter as well as indexing the bits in each parameter with index zero corresponding to the least significant bit. The "Bit Mapping" was, therefore, implemented in the following manner:

1- Set up two look-up tables, one containing the parameter indices and another containing the bit indices.

2- Find the "mask":

\[ \text{MASK} = 2^{(\text{bit index})} \]

3- Set up the appropriate offset using the parameter index and then find the corresponding value to be converted (say V).

4- Perform an "AND" operation:

\[ r = V \& \text{MASK} \]

if \( r = 0 \) then \( r \) remains as zero else \( r = 1 \).

5- Shift the binary string to the left by one bit and add \( r \) to it.

6- Repeat the last 4 routines for the next bit.
<table>
<thead>
<tr>
<th>Importance Class</th>
<th>Parameter</th>
<th>Parameter Index</th>
<th>Bit Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LARI 1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Block Amplitude</td>
<td>21, 22, 23, 24</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>LAR 1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>LAR 2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>LAR 3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>LAR 1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>LAR 2, 4</td>
<td>2, 4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>LAR 3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>LTP Lag</td>
<td>9,10,11,12</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>LAR 2, 5,6</td>
<td>2,5,6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>LTP Lag</td>
<td>9,10,11,12</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>LTP Lag</td>
<td>9,10,11,12</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>LTP Lag</td>
<td>9,10,11,12</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>LTP Lag</td>
<td>9,10,11,12</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Block Amplitude</td>
<td>21, 22, 23, 24</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>LAR 1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>LAR 4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>LAR 7</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>LTP Lag</td>
<td>9,10,11,12</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>LAR 5,6</td>
<td>5,6</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>LTP Gain</td>
<td>13,14,15,16</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>LTP Delay</td>
<td>9,10,11,12</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Grid Position</td>
<td>17, 18, 19, 20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Block Amplitude</td>
<td>21, 22, 23, 24</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LAR 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>LAR 2,3,4,8</td>
<td>2,3,4,8</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>LAR 5,7</td>
<td>5,7</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>LTP Gain</td>
<td>13,14,15,16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RPE Pulses</td>
<td>25 ... 37</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>RPE Pulses</td>
<td>38 ... 50</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>RPE Pulses</td>
<td>51 ... 63</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>RPE Pulses</td>
<td>64 ... 76</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Grid Position</td>
<td>17, 18, 19, 20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Block Amplitude</td>
<td>21, 22, 23, 24</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>LAR 1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>LAR 2,3,4,6,8</td>
<td>2,3,4,6,8</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>LAR 7,8</td>
<td>7,8</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Block Amplitude</td>
<td>21, 22, 23, 24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RPE Pulses</td>
<td>25 ... 37</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RPE Pulses</td>
<td>38 ... 50</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RPE Pulses</td>
<td>51 ... 63</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RPE Pulses</td>
<td>64 ... 76</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>LAR 2,3,5,6</td>
<td>2,3,5,6,4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Block Amplitude</td>
<td>21, 22, 23, 24</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.9 Bit Mapping of Encoded Bits
The parameter indices of Table 5.9 are different from those given in the documentation. This is due to the fact that the parameters are stored differently in our implementation.

5.4.10 Computational Complexity and Memory Usage

Tables 5.10 and 5.11 show the computational complexity of individual subroutines and memory allocation of the encoding programme. The two most complex subroutines are the LTP analysis and the 11th order weighting filter. The encoding

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>No. of Instructions</th>
<th>Processing Time (ms)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-processing</td>
<td>1,857</td>
<td>0.29712</td>
</tr>
<tr>
<td>Autocorrelation Function</td>
<td>5,007</td>
<td>0.80108</td>
</tr>
<tr>
<td>Schur Recursion</td>
<td>907</td>
<td>0.14512</td>
</tr>
<tr>
<td>Log.-Area-Ratio Conversion</td>
<td>203</td>
<td>0.03248</td>
</tr>
<tr>
<td>LAR Quantization</td>
<td>96</td>
<td>0.01536</td>
</tr>
<tr>
<td>LAR Decoding</td>
<td>560</td>
<td>0.0896</td>
</tr>
<tr>
<td>Interpolation</td>
<td>141</td>
<td>0.02256</td>
</tr>
<tr>
<td>Inverse LAR</td>
<td>779</td>
<td>0.12464</td>
</tr>
<tr>
<td>Short-Term Inverse Filter</td>
<td>15,029</td>
<td>2.40464</td>
</tr>
<tr>
<td>LTP Analysis</td>
<td>49,707</td>
<td>7.95312</td>
</tr>
<tr>
<td>LTP Parameter Quantization</td>
<td>69</td>
<td>0.01104</td>
</tr>
<tr>
<td>APCM Quantization</td>
<td>4,564</td>
<td>0.73024</td>
</tr>
<tr>
<td>Long-Term Inverse Filter</td>
<td>2,404</td>
<td>0.38464</td>
</tr>
<tr>
<td>Weighting Filter</td>
<td>20,320</td>
<td>3.2512</td>
</tr>
<tr>
<td>RPE Grid Selection</td>
<td>1,008</td>
<td>0.16128</td>
</tr>
<tr>
<td>Bit Mapping</td>
<td>5,177</td>
<td>0.82832</td>
</tr>
<tr>
<td>Others</td>
<td>153</td>
<td>0.02448</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>107,981</strong></td>
<td><strong>17.27692</strong></td>
</tr>
</tbody>
</table>

*(@ 160 ns / Instruction)

Table 5.10 Computational Complexity of RPE-LTP Encoder
process takes 17.3 ms per speech frame (at 160 ns/instruction) i.e. 86.5% of the allowed time. The data storage requires 4 kbytes of memory as compared to 3 kbytes of storage for the instructions. This is due to the fact that a dual input and output buffering system is employed.

<table>
<thead>
<tr>
<th>Encoder: Memory Allocation (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programme</td>
</tr>
<tr>
<td>Data</td>
</tr>
<tr>
<td>TOTAL</td>
</tr>
</tbody>
</table>

Table 5.11 Memory Allocation of RPE-LTP Encoder

5.5 Software Implementation of RPE-LTP Decoder

Figures 5.7 and 5.15 show, respectively, the block diagram and the flow chart of the implementation of the decoder. The decoder consists of four main subsections:

- RPE decoding,
- Long-Term Synthesis,
- Short-Term Synthesis,
- Post Processing.

5.5.1 Bit Decoding

The transmitted bits are captured using the same dual input and output buffering system as of the encoder. The received binary string needs processing to decode the parameter values. This is implemented in the following manner:

1. Reset all parameter values to zero.
2. Perform an "AND" operation on the transmitted string, \( T_s \):

\[
\text{r} = T_s \; \& \; 0x0001
\]
Fig 5.15 Flow Chart of Speech Decoding Process
if \( r = 0 \) then \( r \) remains as zero, else \( r = 1 \).

3 - Set up the appropriate address of parameter.

4 - Shift present value of integer by one bit to left.

5 - Add 'r' to shifted value.

6 - Repeat the last 4 routines for the next bit.

In the decoder, only a table containing the parameter indices is needed and there is no need for storing the bit indices.

5.5.2 Bad Frame Correction

Prior to the decoder, the Error Detection System receives the 260 transmitted bits and corrects the erroneous bits. If, however, the transmitted frame is totally corrupted, it sets bit 261 of the transmitted string to "1", indicating a bad frame. In the decoder, every time a frame is received, bit 261 is checked and the parameters are adjusted accordingly.

As was mentioned in section 5.3.2, that there are four subsegments in each frame and thus four LTP lags L1, L2, L3 and L4, one for each subsegment. The simulations showed that the best adjustment for a bad frame was muting the input excitation to the LTP synthesis filter and repeating the very last LTP lag i.e. L4, four times with a small LTP gain of 0.35. In this way the memory contents of the LTP filter was output and there was no noticeable discontinuity in the output speech. The short-term synthesis filter uses the LAR values of the previous frame.

It should be noted that this scheme was used for the initial BTRL Test-Bed and GSM has now decided on a slightly different but similar scheme [10]. In the current GSM system, the Speech Frame Substitution (SFS) is activated when the most vulnerable bits of the speech coder parameters are so heavily corrupted that they can not be corrected by the channel decoder. When this situation is detected by the channel decoder, the SFS replaces the corrupted speech frame by the preceding (uncorrupted) speech frame and submits this to the speech decoder. If subsequent frames are unusable then the speech decoder output is progressively (after a maximum of 320 ms) muted to indicate channel breakdown to the user [27, 28].
5.5.3 Short-Term Synthesis Filter

The excitation signal, $d'(k)$, for the short-term synthesis filter is reconstructed applying the identical procedure to that in the encoder and the filter is implemented according to the lattice structure of Figure 5.16.

![Figure 5.16: Structure of Short-Term Synthesis Filter](image)

5.5.4 Post Processing

The output of the synthesis filter is fed into an IIR deemphasis filter, the resulting output signal is then scaled up and converted into A-law format.

5.5.5 Computational Complexity and Memory Usage

Tables 5.12 and 5.13 show the computational complexity and memory allocation of the decoding programme. The decoding process takes only 4.65 ms per speech frame (at 160 ns / instruction) i.e. 23% of the allowed time as compared to 17.3 ms for encoding. This is however expected since all of the analysis are performed during the encoding stage. The data storage occupies 4k bytes of memory. The programme only occupies 1.6k bytes of memory. This is again expected since dual input and output buffering system is used for serial transmission and receiving of data.
### Decoder: Computational Complexity (per frame)

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>No. of Instructions</th>
<th>Processing Time (ms)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Decoding</td>
<td>4,032</td>
<td>0.64512</td>
</tr>
<tr>
<td>APCM Decoding</td>
<td>1,229</td>
<td>0.19664</td>
</tr>
<tr>
<td>LAR Decoding</td>
<td>572</td>
<td>0.09152</td>
</tr>
<tr>
<td>Interpolation</td>
<td>141</td>
<td>0.02256</td>
</tr>
<tr>
<td>Inverse LAR Conversion</td>
<td>846</td>
<td>0.13536</td>
</tr>
<tr>
<td>Long-Term Synthesis Filter</td>
<td>2,936</td>
<td>0.46976</td>
</tr>
<tr>
<td>Short-Term Synthesis Filter</td>
<td>16,275</td>
<td>2.6040</td>
</tr>
<tr>
<td>Post Processing</td>
<td>2,884</td>
<td>0.46144</td>
</tr>
<tr>
<td>Others</td>
<td>157</td>
<td>0.02512</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>29,072</strong></td>
<td><strong>4.65152</strong></td>
</tr>
</tbody>
</table>

*(@ 160 ns / Instruction)*

Table 5.12 Computational Complexity of RPE-LTP Decoder

### Decoder: Memory Allocation (Bytes)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Programme</td>
<td>1,640</td>
</tr>
<tr>
<td>Data</td>
<td>4,104</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>5,744</strong> Bytes</td>
</tr>
</tbody>
</table>

Table 5.13 Memory Allocation of RPE-LTP Decoder
5.6 Speech Coder Hardware

The speech codec hardware, illustrated in Figure 5.17, was designed and built at BTRL. Two WE-DSP32 processors execute the speech coding algorithm (one for encoding and one for decoding) and the remaining hardware elements provide the required analogue-to-digital and digital-to-analogue conversions and the appropriate timing signals. All housekeeping and overall system supervision is performed by an MC68000 computer. Communication with the outside world is implemented using two serial RS232 posts, serving the user terminal and the host computer (downloading object programmes).

At the encoder the timing derivation block generates all the required clock frequencies phase-locked (when required) to an externally applied 16 kHz clock. The clock frequencies generated are:

- 8 kHz sampling clock,
- 50 Hz frame synchronization signal,
- 2 kHz clock to the DSP32, controlling the digital input/output of the coded data,
- 1536 kHz serial input and output clock.

The 8 kHz sampling rate clock is connected to the PCM codec device which samples the incoming speech signal and generates an 8 bit A-law encoded PCM sample every 125 us. The 8 bit samples are clocked out serially from the codec output to the DSP32 serial digital input at a clock rate of 1536 kHz. An 8 kHz clock is also connected to the input load (ILD) pin of the DSP32. Every active transmission of the sampling rate clock initiates a serial input to the DSP32.

As the coded speech samples are transmitted in frames of 20 ms duration, the 50 Hz clock signal is used by the DSP32 for frame synchroniaztion. As soon as some encoded data has been derived for the current frame, the DSP32 commences to output data by loading the output serial register one byte (8 bits) at a time. The 2 kHz clock (one eighth of the 16 kHz rate) is used as the load control (OLD) for this output serial register. The serial data is clocked out of the DSP32 using the 1536 kHz clock. Once 8 bits have been clocked out, the output shift register empty (OSE) signal becomes active. This signal controls the 8 bit serial-to-parallel (S/P) and parallel-to-serial (P/S) registers at the
Figure 5.17 Speech Codec Hardware for GSM Test-Bed (16 kHz)
output of the DSP32, such that the burst of eight valid data bits at 1536 kHz is converted to a continuous data stream at 16 kb/s. At the decoder a frame synchronization signal and a 16 kHz clock signal are assumed to be present (derived by the radio system). The interface to the speech decoder DSP32 essentially mirrors the encoder interface [29, 30].

5.7 Voice Activity Detection (VAD)

Background

Digital speech coders achieve the compression of the original 64 kb/s speech signal down to 16 kb/s (and less) with telephone quality. Additional gain, however, in the transmission can be achieved by taking advantage of the half-duplex effect of any normal telephone conversation, when each subscriber speaks for less than half of the time. The remainder of the time is composed of listening, gaps between words and syllables, and pauses. One can use this idle time to interpolate additional talkers up to twice the overall channel capacity if a number of channels are available [31].

During the effort to maximize the spectrum efficiency of the GSM system, it was found that a significant increase in spectrum efficiency could be achieved by utilizing voice activated transmission. The basic principle, known as Discontinuous Transmission (DTX), is to switch the transmitter on only for those periods when there is active speech to transmit. In this way; (i) the average interference on the "air" is reduced thus allowing a smaller frequency re-use cluster size (reduced co-channel interference), (ii) the spectrum efficiency could be doubled given an average activity of 50%, (iii) and the power drain in hand portable equipment is reduced [3, 32]. The DTX system basically consists of two parts [33, 34]:

- a Voice Activity Detector (VAD) on the transmit side,

- comfort noise generator at the receive side.

The function of the VAD is to distinguish between speech superimposed on noise and noise without speech present. The output of the VAD is used to control a transmitter switch. If the VAD fails to detect every speech event then the transmitted speech will be degraded by clipping. On the other hand, false classification of noise as speech must be minimized since an increased activity factor would increase the interference on the air.
At the receiving end, the background acoustic noise which is transmitted with the speech abruptly disappears whenever the radio transmitter is switched off. Since the switching can take place very rapidly, within words as well as between words, it has been found that this noise modulation can be very annoying for the listener, especially in mobile environments with high background noise levels. In extreme cases, the speech may be hardly intelligible. This problem can be overcome by generating, on the receive side, the synthetic noise ("Comfort Noise") similar to the background noise on the transmit side. The parameters of this so called comfort noise are estimated on the transmit side and transmitted to the receive side before the radio transmission is cut and at regular low rates afterwards. This allows the comfort noise to adapt to the changes of the noise on the transmit side. The transmission of comfort noise information to the receive side is achieved by means of a special frame, named Silence Descriptor (SID) [27, 35].

In the following section, the implementation of a low complexity speech detector, which can be operated in conjunction with the implementation of RPE-LTP coder is discussed. The VAD was implemented as part of the BTRL Test-Bed. The performance of the VAD in conjunction with the GSM speech coder was investigated by BTRL and was found to be unsatisfactory, especially in the presence of mobile environmental noise. Consequently, the current GSM voice detector [36] was developed by BTRL. Recently, the latest GSM speech coder and the complete DTX system has been implemented in the Speech Research Group at Surrey University, on the second generation of AT&T floating-point DSPs, namely DSP32C [37].

5.7.1 Description of Preliminary VAD Algorithm

The activity decision is primarily based on the evaluation of the signal energy, and on the comparison of this energy with an adaptive threshold. The proposed algorithm is capable of detecting even the low energy segments like fricatives in utterance attacks by monitoring the changes in the power spectrum.

In order to reduce the overhead computational complexity of the algorithm when used in conjunction with the RPE-LTP coder, the following assumptions are made [38]:

- The short-term power spectrum of the signal in a block of samples is directly related to the autocorrelation function of this signal.
- The energy of the signal is well approximated by the maximum magnitude of the
samples within the block.

Both of these parameters are already being computed in the RPE-LTP coder, and
only a small extra processing is needed to execute the VAD.

5.7.2 Software Implementation of VAD

Figure 5.18 shows the flow chart of the speech detector. For each block of
samples:

\[
X(n) ; \quad n = 1, ..., 159 \quad (5.15)
\]

the inputs to the VAD are:

- the signal magnitude, \(X_{\text{max}}\):

\[
X_{\text{max}} = \text{MAX}\{ |X(n)| \} \quad ; \quad n = 1, ..., 159 \quad (5.16)
\]

- the normalized autocorrelation values:

\[
R(i) \quad ; \quad i = 0, ..., 8 \quad (5.17)
\]

Since there are 4 block maximum values, one for each 13 samples, a routine is
needed to select the largest. This value is then compared with an adaptive threshold,
\( \text{VAD}_{\text{TH}} \). Three different cases are then considered:

1) If the signal maximum is much greater than the threshold \((X_{\text{max}} > a \cdot \text{VAD}_{\text{TH}})\,
where 'a' is called the upper threshold constant), then the activity decision is
immediately taken, while a counter is set up for hangover purpose. A hangover
time out is used to bridge short intersyllabic silences, while it does not increase
significantly the speech activity, and it avoids the possible intersyllabic clipping
which is unpleasant in the case of high level background noise.

2) If the signal magnitude is less than the threshold \((X_{\text{max}} < b \cdot \text{VAD}_{\text{TH}})\,
where 'b' is called the lower threshold constant), then the segment is definitively declared non-active.
Speech Detector

- **Yes** if \(VADTH > THMAX\)
  - Clamp Threshold To Upper Level
  - \(VADTH = THMAX\)
- **No**
  - **Yes** if \(VADTH < THMIN\)
    - Clamp Threshold To Lower Level
    - \(VADTH = THMIN\)
  - **No**
    - **Yes** if \(TOUT < -1\)
      - Clamp Time-Out
      - \(TOUT = -1\)
    - **No**
      - **Yes** if \(Xmax > VADTH\)
        - Spectrum Variations
        - \(SUM = \frac{1}{8} \sum |R(i) - Rold|\)
        - **No**
          - **Yes** if \(SUM > C\)
            - **No**
              - **Yes** if \(TOUT \geq 0\)
                - \(VADFLAG = 0\)
                - Silent Frame
          - **No**
            - **Yes** if \(TOUT < 0\)
              - \(VADFLAG = 1\)
              - Speech Frame

- Increment Threshold Set Time-Out
  - \(VADTH = VADTH + d\)
  - \(TOUT = TMAX\)

- Update Autocorrelation Values
  - \(Rold(i) = R(i); \: i = 0 ... 9\)

Fig. 5.18 Flow Chart of the Speech Detector
3) If the signal magnitude is not significantly higher than the VAD threshold, then the segment may be either an active segment or a non-speech segment. This secondary decision is taken by looking at the spectrum variations from the previous block to the current block. If the current block was actually a silence block, then the spectrum will vary significantly only if the current block is a talkspurt attack. It was found that the comparison of the averaged variation in absolute value with a fixed threshold 'c' (called frequency threshold constant) provides a robust criterion for the detection of speech, even for low energy fricatives in background noise.

In addition to this basic strategy, in order to track possible increasing of the background noise level, the VAD threshold is either updated by $X_{\text{max}}$ or incremented by a small amount, $d$.

Finally, Table 5.14 shows the operating conditions of the VAD algorithm. The real-time implementation is performed in such a manner that no call to "divide" subroutine is needed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Speech at a Nominal Level</td>
<td>-12 dB</td>
</tr>
<tr>
<td>Maximum Level of Background Noise</td>
<td>-40 dB (TH_{\text{Max}} = 320)</td>
</tr>
<tr>
<td>Minimum Level of Background Noise</td>
<td>-54 dB (TH_{\text{Min}} = 64)</td>
</tr>
<tr>
<td>Hangover Time</td>
<td>$T_{\text{Max}} = 60$ ms</td>
</tr>
<tr>
<td>Upper Time Threshold Constant</td>
<td>$a = 2$</td>
</tr>
<tr>
<td>Lower Time Threshold Constant</td>
<td>$b = 1$</td>
</tr>
<tr>
<td>Frequency Threshold Constant</td>
<td>$c = 0.2$</td>
</tr>
<tr>
<td>Increment for Time Threshold</td>
<td>$d = 1$</td>
</tr>
</tbody>
</table>

*(on a 16-Bit Scale)*

Table 5.14 Operating Conditions of the VAD Algorithm
5.8 Concluding Remarks

In 1991, a decade of coordinated development, throughout Europe, will result in the replacement of the first generation of mobile cellular telephone systems (analogue) with the more efficient digital mobile radio system. The tremendous increase in the demands for the mobile services and non-compatibility of nine different types of cellular-telephone systems, currently in operation in 17 European countries, were the main drive for this initiative. The new Pan-European system, employing the latest state of technology, as well as providing a common air interface and better spectrum efficiency, is aimed to be cheap, compact and power efficient with a more secure communication link.

Voice communication remains the dominant part of the new DMR system. In an extensive programme to select an optimum speech coding technique, the initial 20 proposed coding schemes were narrowed down to 6. A more detailed performance analysis of the remaining coders resulted in the selection of a hybrid coder (the coders proposed by France and Germany). The final coding algorithm, a variant of Regular Pulse Excitation, operates at 13 kbit/s with a moderate complexity. The inclusion of long-term prediction improves the speech quality but imposes a higher capacity FEC to survive the high channel errors of the mobile link (a total bit rate of 22.8 kbit/s).

In this chapter, we have reported on the real-time implementation of the proposed coding scheme. The GSM recommendations of the algorithm give a functional description as well as computational details on the fixed-point implementation. As part of the BTRL contract, the speech coder was implemented on a floating-point processor, AT&T DSP32. The functional description of the coding scheme was used to achieve this. The most difficult part of the implementation proved to be the quantization of the residual signal (ADPCM coding) where the given quantizer threshold and output levels only apply to the fixed-point implementation. Many different solutions to this problem were investigated during the simulation stage and we finally decided that the most suitable way would be to normalize the input speech samples to \( \pm 1.0 \). This will not have any significant effects on the dynamic range and precision of the implementation due to the DSP32 representation of floating-point numbers (32-bit data length; 24-bit mantisa, 8-bit exponent).
The only shortcoming of this chapter is the un-availability of performance measurements of the implemented coder. As mentioned, the implemented coder was part of the BTRL Test-Bed and on-going tests are being carried out within BTRL. It is hoped to report on the resulting performance measurements in future publications.
References


24. CCITT, "Recommendation G.711 and G.712".


33. CEPT/CCH/GSM, "Speech Processing Functions: General Description", GSM Recommm. 06.01, Draft 1.0.0, January, 1989.

34. CEPT/CCH/GSM, "Discontinuous Transmission (DTX) for Full Rate Speech Traffic Channels", GSM Recommm. 06.31, January, 1989.

36. ETSI/GSM, "Voice Activity Detector", GSM Recommn. 06.32, Version 2.0.0.


6.1 Introduction

Satellite technology has come a long way since the days that Arthur C Clarke first described the concept of achieving world wide communications by placing three satellites around the geostationary orbit, in 1945. Today, large organizations such as INTELSAT and INMARSAT through the satellite networks are making available transoceanic international communications by providing links between national telephone networks. With advances in technology, the traditional role of satellites is now changing and there is a tremendous drive towards satellite networks that provide direct user to user services. The earth stations have also reduced considerably in size, making the possibility of such services a reality [1,2].

Since 1984, after the deregulation of public telecommunications in the USA, there has been an increasing number of manufacturers who have produced a range of small satellite terminals, in Ku band, which are designed to provide medium speed data channels to individual locations without the need to install dedicated terrestrial circuits.
Some of these systems also provide voice and video transmission capabilities [3,4]. These terminals often called VSATs (Very Small Aperture Terminals) use small diameter antennas (under 2 meters) with an offset fed parabolic reflector [5]. There is also an ongoing study on providing an experimental Ka band VSAT network, as part of the OLYMPUS Utilization Programme which will operate amongst Technical centres, Universities and Research Establishments [6,7].

New Ku band VSAT networks are now commercially available as alternative networks, in parts of the world where a reliable terrestrial infrastructure does not exist, or for small private business networks or mobile and defence networks. One of such systems developed by MULTIPOINT COMMUNICATIONS and SPL (in UK), MP-SP2300, is a Single Channel per carrier (SCPC) Demand Assigned Multiple Access (DAMA) VSAT system, suitable for a mix of high and low capacity terminals, requiring voice or data communications. This network assigns a transparent bi-directional channel on demand from a pool of channels and automatically routes this between VSAT and HUB or VSAT to VSAT via the HUB DAMA switch. Twenty DAMA networks serve a network of 500-1000 VSATs with a low probability of queuing. The channels interface directly with CCITT voice or X25 packaged data networks. The SCPC-DAMA VSAT network offers an assigned channel uncorrupted by random clashes of data as encountered on some VSAT systems. The network incorporates a HUB-VSAT broadcast mode, management software, monitoring and billing facilities [4].

Early VSAT systems were designed exclusively for low data rate traffic. However, developing countries have seen them as the answer to their rural telephony problem and thus the inclusion of a voice channel has become of paramount importance.

The MP-SP2300 VSAT system required the development and implementation of a robust, efficient and self-synchronous digital speech coder operating at 9.6 kbit/s, producing good quality speech. The hybrid coder, Base-Band CELP [8], developed in the Speech Research Group, was therefore modified and refined for this purpose. The resultant speech coder operates at 7 kbit/s (an extra 2.6 kbit/s is used for signalling and synchronization) and is robust against channel errors as well as background noise. Two AT&T WE-DSP32 floating-point processors (one for encoding, and the other for decoding) are used to realize the real-time coder on a double extended Eurocard board. The hardware is fully synchronized and offers the user many flexibilities.

In this chapter, a brief overview of the MP-SP2300 VSAT system will be followed by detailed description of the Base-Band CELP coding algorithm. The software
and hardware implementation of the real-time coder will be discussed. Finally, we will report on the performance of the coder.

6.2 Overview of the System

6.2.1 Network Operation

The MP-SP2300 SCPC DAMA VSAT (MP-SDV) network development was a joint venture between Multipoint Communications, Signal Processors and the Royal Signals and Radar Establishments under the auspices of the BNSC satcoms programme [9]. The network is entirely digital using Phase Shift Keying (PSK) modulation for both the VSAT to HUB SCPC and HUB to VSAT Time Division Multiplexed (TDM) channels. The basic MP-SDV network (see figure 6.1) offers 20 x 9.6 kbit/s SCPC trunks from the VSAT to the HUB and a single TDM frame containing 20 x 9.6 kbit/s time slots (plus overhead and framing) on a single carrier from HUB to VSAT. The VSAT to HUB channels compromise one random access channel and nineteen SCPC DAMA message channels. The TDM channels from HUB to VSAT compromise one signalling time slot and nineteen time slots for TDM-DAMA channels.

The assignment of a bi-directional channel from VSAT to HUB is as follows. The VSAT accesses the HUB via the random access channel, F1, requesting a HUB connected line or port. The HUB informs the VSAT as to the frequency of the assigned channel via the signalling time slot, T1, in the HUB to VSAT TDM frame. The VSAT transmits on Fx and receives on Time slot Tx. The channel is transparent and can be connected to an X25 data or CCITT voice network. For the transferral of X25 data a buffer is incorporated at the VSAT to enable data packages to be present, on request, some 600 ms after transmission.

For VSAT to VSAT data communications the originating VSAT accesses the HUB requesting another VSAT. The HUB maps the message received on the assigned channel, Fx, to the TDM slot, Tx. The HUB informs the Tx VSAT via the TDM signalling time slot, T1, as to time slot, Tx, from which the message can be extracted. The request to re-send data for X25 operation is returned from the Rx VSAT via the random access channel, F1, to the HUB and signalling time slot, T1, to the originating VSAT. This mode avoids the use of two SCPC DAMA channels for data and the risk of queuing when requesting a re-send.
Fig. 6.1 The MP-SP2300 SCPC DAMA VSAT Network
VSAT to VSAT voice requires two DAMA channels Fx, Fy and time slots Tx and Ty. VSAT networks are best suited for transmission of data with voice confined to VSAT to HUB channels.

At 9.6 kbit/s the MP-SDV has twice the throughput of slotted aloha systems and also offers low bit rate voice. The MP-SDV can be programmed for higher bit rates of 19.2 kbit/s and 64 kbit/s at the expense of fewer DAMA channels. For Telex and Fax, standard multiplexers or store-and-forward are employed.

6.2.2 Network Size

The number of VSATs which can be supported by a pool of DAMA channels is determined in the same way as the number of household or business telephones that can be supported by telephone exchange lines. The ERLANG loading of each VSAT is computed and summed for the network loading. The network size is adjusted for a grade of service i.e. probability of queuing for a channel. Twenty DAMA channels will support 500-1000 VSATs with 2% queuing probability.

Some VSATs support a single terminal while others support a network of telephones and computers. VSATs supporting a local network may require simultaneous access to more than one DAMA channel.

6.2.3 Advantages of MP-SDV System

The MP-SDV network requires low power from Satellite to HUB, enabling a smaller HUB to be used. The HUB may be an existing 5.5m E2 or 7m F2 type. The network is configured with 0.9-1.5m VSAT terminals and is compliant with the INTELSAT and EUTELSAT specifications. The MP-SDV network employs minimal HUB software leading to a simple rugged network. Two separately located HUBs can be used to operate the network giving redundancy with security.
6.3 Speech Coding Algorithm

Background

Of the coders that were reviewed in chapter 3, Multipulse coding (MPLPC) and Code Excited coding (CELPC), in the time domain, were seen to be the two most promising schemes for achieving good quality speech at low bit rates. For coding speech at rates below 8 kbit/s, MPLPC loses its attractions due to the fact that both amplitude and position of pulses need to be transmitted. In order to reduce the bit rate of a Multipulse coder, the number of selected pulses per speech frame needs to be reduced, consequently limiting the speech quality performance. CELPC coding, on the other hand, is more successful in reducing the bit rate down to 4.8 kbit/s or even lower. This is, however, at the expense of excessive computational complexity which is not feasible for certain applications. The other major disadvantage of conventional CELPC at rates below 6 kbit/s, is the effect of extreme quantization noise resulting from the use of large vector dimensions.

Since 1987, there has been extensive research on the development of low bit rate speech coders in the Speech Research Group of University of Surrey [10,11]. A new coding scheme, Base-Band Code Excited Linear Predictive Coding (CELPC-BB) [8] was developed which combines Base-Band coding, Multipulse coding and CELPC coding techniques. It enjoys features of both MPLPC and CELPC schemes and also by operating on the base-band signal, both the computational complexity and the bit rate are reduced.

Figure 6.2 illustrates the block diagram of the CELP-BB Coder.

Figure 6.2 illustrates the block diagram of the CELP-BB coder. The encoding process compromises the following major sections:
- LPC Analysis and Filtering
- Base-Band Extraction (or Decimation)
- LTP Analysis and Filtering
- Pattern Matching (or Codebook Search)

and the decoder is formed as a cascade of the following sections:

- Pattern Reconstruction
- LTP Synthesis
- Full-Band Reconstruction (Up Sampling)
- LPC Synthesis.

In the encoding stage, conventional LPC analysis methods are employed to
remove the short-term correlations from the input block of speech. The remaining signal
(LPC residual) is then divided into sub-blocks, the number of sub-blocks determines the
total bit rate of the coder. Each sub-block is separately filtered by a weighting
(smoothing) filter and down sampled. The extracted base-band will then pass through an
analysis-by-synthesis procedure containing the LTP analysis and codebook search. By
excluding the LPC synthesis and noise shaping filter (present in conventional CELP)
from the AbS loop, the considerable amount of computations required by the IIR LPC
filter are eliminated.

At the decoder, the reverse operation of each stage of the encoder is performed.
These include forming the exact pattern (or excitation) as the encoder and placing the
short- and long-term correlations back into the reconstructed full-band signal.

In the past few years, the CELP-BB coding algorithm has undergone many
modifications and refinements. The performance of the coder at various bit rates has been
investigated [8, 12]. In recent work, the coder has been modified in such a way as to
posses a limited amount of built-in robustness against random channel errors, through
the use of Line Spectral Frequency (LSF) Transformations of the LPC parameters [13-
15]. The real-time implementation of the coder, at rates of 4.8 to 8 kbit/s, have also been
realized by using floating-point digital signal processors [16-19].

For the VSAT application, the CELP-BB coder is designed to operate at 7 kbit/s
with an additional 2.6 kbit/s for signalling and synchronization. The following section
gives a detailed description of the coding algorithm at this bit rate.
6.3.1 Algorithm Description of Encoder (7 kbit/s)

A block diagram of the CELP-BB encoder is shown in figure 6.3. A block of 240 speech samples, $s(t)$, is first preemphasized and then analyzed using Durbin's method and 10 LPC reflection coefficients are computed. These are transformed into Log.-Area-Ratios (LARs) and scalar quantized. The quantized parameters are then used to inverse filter the input block of speech, for removal of short-term correlations. The remaining signal (first residual), $r(t)$, is then divided into 8 sub-blocks of 30 sample lengths and passed through a weighting filter (smoothing filter), separately. Filtered sub-blocks are split into a number of sequences equal to the decimation factor (3 in this case). These sequences (each of length 10) are computed in terms of their energies and the one with the highest energy is selected as the base-band sequence, $r_B(t)$. This approximates base-band coding to MPLPC and eliminates the high frequency regeneration (HFR) noise significantly. The position of the selected sequence needs to be transmitted to the decoder in order to place the base-band sequence in the correct location in the full-band reconstruction stage.

The remaining stages of encoding, long-term prediction and codebook search, are performed inside an analysis-by-synthesis loop. To remove the long-term correlations from the extracted base-band, a long-term prediction (LTP) analysis, based on a simplified cross-correlation scheme, is performed. The LTP delay and gain are computed and used within the LTP inverse filter to obtain the second residual signal, $e(t)$. The remaining signal should ideally resemble a very randomized signal and is usually modelled as being Gaussian Noise. An 8-bit codebook (256 entries) filled by Gaussian sequences (each of 10 sample length) is exhaustively searched to find the best match for the error signal, $e(t)$. The error minimization scheme used for codebook search in CELP-BB is far simpler than that used in conventional CELP coding. It is still, nevertheless, the most complex part of the encoder. The codebook search will produce a codebook index and an optimum gain parameter (for each sub-block) which are also transmitted (after quantization) to the decoder.

Table 6.1 shows the bit allocation for the various parameters of the CELP-BB coder operating at 9.6 kbit/s. The net bit rate of the speech coder is 7 kbit/s (6,967 bits/s
Fig. 6.3 Block Diagram of CELP-BB Encoder
### Table 6.1 Bit Allocation of Parameters of VSAT Speech Coder

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Bits / Frame</th>
<th>Bit Rate (bits / s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPC via LAR</td>
<td>40</td>
<td>1,333</td>
</tr>
<tr>
<td>LTP Delay</td>
<td>33</td>
<td>1,100</td>
</tr>
<tr>
<td>LTP Gain</td>
<td>24</td>
<td>800</td>
</tr>
<tr>
<td>Codebook Index</td>
<td>64</td>
<td>2,133</td>
</tr>
<tr>
<td>Codebook Gain</td>
<td>32</td>
<td>1,067</td>
</tr>
<tr>
<td>Grid Position</td>
<td>16</td>
<td>533</td>
</tr>
<tr>
<td>Frame Synchronization</td>
<td>79</td>
<td>2,633</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>288</strong></td>
<td><strong>9,600</strong></td>
</tr>
</tbody>
</table>

exactly) and the remainder of the bits (79 bits per frame corresponding to 2,633 bits/s) are used for frame synchronization. The frame synchronization is base on "Bit Stuffing" and use of unique words.

### 6.3.2 Algorithm Description of Decoder (7 kbit/s)

Figure 6.4 illustrates the block diagram of the CELP-BB decoder. The decoder contains an identical codebook to that present in the encoder. The transmitted codebook index is used to select the corresponding sequence. The selected sequence is then scaled up by its corresponding gain factor. The LTP synthesis filter replaces the long-term correlations back into the sequence, thus recovering the base-band signal, \( \hat{f}_b(t) \). The recovered base-band signal is then shifted to the correct location (using the transmitted grid position) and interpolated with zero insertions. This forms the excitation signal, \( \hat{f}(t) \), to the LPC synthesis filter, the output of which, is the final quantized output speech signal, \( \hat{s}(t) \).
Fig. 6.4 Block Diagram of CELP-BB Decoder
6.4 Software Implementation of Encoder

The complete coding scheme was simulated using high-level software written in C/Unix. These simulations were used to verify the operation of the coder and later translated (manually) into the real-time software appropriate for AT&T WE-DSP32 processors. The encoding process is performed on a single DSP32 processor and extensive optimization solutions were sought to realize the real-time software. In this section, we will discuss the software implementation of the VSAT Encoder, especially the algorithm simplifications that were needed to achieve single chip implementation.

6.4.1 Data Acquisition and Preprocessing

Data Acquisition

CELP-BB coder, similar to the coder reported in chapter 5, is an LPC based coder and requires a dual input and output buffering system. This is implemented in the same way as discussed in section 5.4.1 (see flow chart of figure 5.10), with the exception that no external Frame Synchronization Signals are provided. This is not, however, a major problem as registers r20 (PIN) and r21 (POUT) can be continuously monitored to control the DMA flow of input and output data. The encoder input receives blocks of 240 speech samples (at sampling rate of 8 kHz corresponding to a frame rate of ~33 Hz) and compresses the input block to 288 bits of data (inclusive of frame synchronization information for the decoder).

Preprocessing

The input block of speech will pass through a preprocessing stage before being passed on to the speech encoder. The preprocessing stage includes:

- Conversion of A-law coded samples to floating-point,
- Removal of DC-offset,
- Preemphasis.

The analogue-to-digital (A/D) conversion of the input speech signal is performed in accordance with CCITT A-Law coding (recommendations G.711/2). The DSP32 being a floating-point processor requires the floating-point representation of the input speech samples. This is achieved by using one of the "Special Functions" on the DSP32.
Any DC offset present in the input speech needs to be removed. This is essential for two reasons. Firstly, the presence of the DC offset reduces the resolution of the input speech signal (reducing the dynamic range). Secondly, during the high frequency regeneration stage (spectral folding) the DC component will be reflected into higher bands, causing degradations in the output speech. A simple notch high-pass filter is applied in order to remove the offset of the input signal, $S_i(k)$, to produce the offset-free signal, $S_{of}(k)$:

$$
S_{of}(k) = S_i(k) - S_i(k-1) + a \cdot S_{of}(k-1) ; \quad k = 0, ... , 239
$$

(6.1)

where "a" is a fixed constant (= 0.9989929).

To ensure that Durbin's method produces stable reflection coefficients, the offset-free signal is preemphasized using a first order FIR filter with a preemphasis factor of 0.86:

$$
S(k) = S_{of}(k) - 0.86 S_{of}(k-1) ; \quad k = 0, ... , 239
$$

(6.2)

### 6.4.2 LPC Analysis

The LPC analysis is a cascade of the following stages:

- Computation of 10 reflection coefficients using Durbin's Recursive Method,
- LAR Transformation of reflection coefficients,
- Quantization and Coding of LARs,
- LPC Inverse Filtering,

and due to its important contribution towards the final output speech quality, needs efficient and effective implementation.

#### Durbin's Recursive Method

One of the well known LPC analysis procedures is the autocorrelation method. This method forms a matrix equation for solving for the prediction coefficients. The matrix is of a Toeplitz nature i.e. is symmetric and all the elements along a given diagonal are equal [20]. Several efficient recursive procedures have been devised for solving this system of equations. Durbin's recursive procedure is one of the most efficient methods for solving this particular system of equations [21].
Durbin's method, however, needs a preprocessing stage to guarantee stable coefficients. During the simulation stage, we found that the best preprocessing system was preemphasis as compared to 'windowing'. Overlapping windows may prove to be superior to preemphasizing the speech signal but introduces extra delay which was not desirable in this system. A fast and efficient real-time code was written for implementing the Durbin's method. For a frame length of 240 samples and a 10th order LPC analysis, the code only requires 2p+1 locations of memory for 'scratch pad' (where p is the order of the LPC system). The total number of operations adds up to 6,411 instructions, 5282 instructions for the autocorrelation and 1129 instructions for the Durbin's method.

LAR Transformation of Reflection Coefficients

The 10 computed reflection coefficients are transformed into LARs for the reasons explained in section 5.4.5. In this implementation, no approximations were used for the transformation and equation 5.9 was directly implemented.

Quantization and Coding of LARs

Accurate quantization of the LPC parameters plays an important role in the overall performance of the coder. A large data base (120 seconds of 8 kHz sampled speech from a wide range of both male and female talkers) was used to design individual uniform quantizers for each order of the LPC. Using the statistics of the parameters, various bit assignments, within the allowed capacity (40 bits per frame), were experimented and we found that \{6,5,5,4,4,4,3,3,3,3\} bit allocation pattern provided the best noise shaping characteristics.

Table 6.2 shows the bit allocation and statistics of each LPC parameter (input speech preemphasized with a 0.86 factor). The individual scalar quantizers are based on the average mean value and allocated number of bits. Figure 6.5 illustrates the flow chart of the real-time code for quantization and coding of the LARs.

All the quantizers used in our implementation are of symmetrical nature. The implementation only considers one half of the quantizer (positive) with the only difference that for negative values an appropriate offset is added to the corresponding code value. This will enhance the execution speed of the implementation, both in coding and decoding stages.
LAR Quantization

\[ i = 0 \]

Code Value = 0
\[ n = 0.5 \]
Maxcode\[i\] = \[2^\text{A (no. of bits-1)}\]-1

\[ d = |\text{LAR} - \text{Mean}| \]
\[ S = \text{sign}(\text{LAR}-\text{Mean}) \]

\[ d = d - (\text{Step Size}) \]

Is Code\[i\] \(\geq\) Maxcode\[i\] ?

Yes

n = n + 1
Code = Code + 1

No

Is d \(\leq\) 0 ?

Yes

Compute Quantized Value
\[ \text{LAR} = \text{Mean} + (n.S.\text{Step Size}) \]

Is S > 0 ?

Yes

Add Offset To Code Value
\[ \text{Code}[i] = \text{Code}[i] + (\text{Maxcode}[i]+1) \]

No

i = i + 1

Is i > Order ?

Yes

END

Fig. 6.5 Uniform Scalar Quantization of LARs
<table>
<thead>
<tr>
<th>Order</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Bits</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Mean Value</td>
<td>0.237</td>
<td>-0.298</td>
<td>0.142</td>
<td>-0.165</td>
<td>0.040</td>
<td>-0.211</td>
<td>-0.021</td>
<td>-0.117</td>
<td>0.008</td>
<td>-0.052</td>
</tr>
<tr>
<td>Step-Size</td>
<td>0.052</td>
<td>0.060</td>
<td>0.056</td>
<td>0.069</td>
<td>0.061</td>
<td>0.067</td>
<td>0.107</td>
<td>0.145</td>
<td>0.087</td>
<td>0.062</td>
</tr>
<tr>
<td>SNR (dB)</td>
<td>31.1</td>
<td>25.1</td>
<td>26.9</td>
<td>19.5</td>
<td>19.2</td>
<td>20.3</td>
<td>14.1</td>
<td>13.0</td>
<td>13.4</td>
<td>12.6</td>
</tr>
</tbody>
</table>

Table 6.2 Bit Allocation and Statistics of LPC Parameters

**LPC Inverse Filtering**

The LPC inverse filtering, like the previous coder, uses the lattice structure implementation (see figure 5.12). The lattice structure requires the reflection coefficients and therefore the 10 quantized LARs are transformed back into reflection coefficients according to equation 6.3:

\[
 r(i) = \frac{10LAR(i) - 1}{10LAR(i) + 1} ; \quad i = 0, ..., 9
\]  

(6.3)

The implementation of the lattice structure is performed in such a way that it exploits the pipeline latencies of the DSP32 to its advantage during the memory update of each iteration. This results in an enhancement of the execution speed as well as minimizing the number of memory locations required for the scratch pad. For a filter order of \( p \), only \( (p+1) \) locations are needed as scratch pad.

**6.4.3 Base-Band Extraction**

The base-band extraction procedure comprises the two following stages;
- Weighting Filter (Smoothing),
- Optimum Sequence Selection.
This technique is identical to the one employed in the GSM coder (see section 5.3.1) and was first used as part of a Regular Pulse Excitation coder [22]. The combination of the weighting filter and optimum sequence selection is an approximation of the pulse search in MPLPC coding.

**Weighting Filter**

The LPC residual, \( r(i) \), is divided into subsegments of 30 sample length. Each subsegment is convoluted with the impulse response, \( w(i) \), of an 11th order FIR filter:

\[
x(i) = \sum_{k=0}^{10} w(k) r(5+i-k) ; \quad i = 0, \ldots, 29
\]

where the filter has the impulse response of table 6.3 [23].

<table>
<thead>
<tr>
<th>i</th>
<th>0, 10</th>
<th>1, 9</th>
<th>2, 8</th>
<th>3, 7</th>
<th>4, 6</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>W(i)</td>
<td>1.000000</td>
<td>0.700790</td>
<td>0.250793</td>
<td>0.000000</td>
<td>-0.045649</td>
<td>-0.016356</td>
</tr>
</tbody>
</table>

*Table 6.3 Impulse Response of the Weighting Filter*

The convolution of the subsegment (30 samples) with the impulse response of length 11 would result in a segment of 40 samples. A 'block filtering' operation is used such that the number of samples is not increased i.e. only centre samples of the convolution sequence are computed. The real-time code for implementing this task can consume a large portion of the allowed processing power, if not written efficiently. Therefore, much effort was spent in minimizing the complexity of the code.

**Optimum Sequence Selection**

Each filtered segment is decomposed into 3 (the decimation factor) candidate sequences of length 10:

\[
x_m(i) = x(m + 3. i) \quad i = 0, 1, \ldots, 9 \quad ; \quad m = 0, 1, 2
\]

where \( m \) denotes the phase of the decimation grid.
A mean squared error criterion is employed to select the optimum down-sampled sequence, \( x_M(i) \), i.e. the sequence with the maximum energy:

\[
E_M = \max_m \sum_{i=0}^{9} x_m^2(i) \quad ; \quad m = 0, 1, 2
\]  

(6.6)

The position of the selected sequence, \( m \), is coded with 2 bits and transmitted to the decoder.

### 6.4.4 Long-Term Analysis

In conventional CELP, in order to cover the full range of pitch periods present in the human speech, 7 bits are usually allocated to encoding the LTP delays (20 - 147 samples at 8 kHz sampling rate). The advantage of base-band coding with respect to this point is two fold. In CELP-BB, the residual is decimated by a factor, \( D \), consequently the range of LTP delays is reduced by a number of samples related to, \( D \). This also reduces the computational complexity (during the cross correlation), since both the sequences and the LTP memory are of smaller lengths. During our extensive simulations, we found that a 32 delay location search performs adequately well for a wide range of male and female talkers and can be coded with only 5 bits (a saving of 2 bits per sub-block). Further improvements on both the bit reduction and complexity is also achieved through a scheme that we have termed "N-Point Windowing". The scheme is based on the assumption that the speech will remain reasonably stationary for durations of up to 30 ms. This means that the LTP delays that are calculated at the beginning of each 30 ms frame are very unlikely to change by much during the rest of the frame. We found that the LTP delays for the following sub-blocks were very likely to fall by not more than 4 samples on either side of that found for the first sub-block.

Based on the cross correlation method, the LTP delay, \( L \), and gain, \( g \), parameters of the first sub-block are computed:

\[
R(i) = \sum_{k=0}^{9} r(k).r'(k-i) \quad ; \quad i = \text{Min Lag}, ... , \text{Max Lag}
\]  

(6.7)

\[
R(L) = \max [R(i)] \quad ; \quad i = \text{Min Lag}, ... , \text{Max Lag}
\]  

(6.8)
\[ g = \frac{R(L)}{S(L)} \]  \hspace{1cm} (6.9)

where

\[ S(L) = \sum_{k=0}^{9} r'(k)^2 (k-L) \]  \hspace{1cm} (6.10)

where minimum and maximum lags have values of 10 (= size of the decimated sub-block) and 41 (= minimum lag + 31), respectively, corresponding to 32 lag locations. The LTP analysis is placed within the AbS loop and \( r'(i) \), represents the reconstructed LPC residual signal. The LTP delay is the location which produces the maximum correlation (equation 6.8).

For the next 7 sub-blocks, an 8-point windowing scheme is employed to search only the 8 points nearest the first computed lag, \( L \). The \( N \)-windowing scheme is simply a slight modification of equation 6.7, as demonstrated below:

\[ R_{NW}(i) = \sum_{k=0}^{9} r(k).r'(k-i) \]  \hspace{1cm} (6.11)

where

\[
\begin{align*}
  i &= \text{Min Lag, \ldots, Min Lag+(N-1)} & \text{if } L < (\text{Min Lag}+(N/2-1)) \\
  i &= \text{Max Lag-(N-1), \ldots, Max Lag} & \text{if } L \geq (\text{Max Lag}-N/2) \\
  \text{else} & \\
  i &= L-N/2-1, \ldots, L+N/2 
\end{align*}
\]

where \( N \) is the length of the window (=8 in this case).

The first two conditions need to be considered, to modify the window for extreme cases of the first LTP delay values i.e. beginning and end of the LTP register. If the first lag happens to be at the start of the LTP register, the same location and the next (N-1) locations need to be searched and if the first lag occurs at the end of the LTP register, the same location and the (N-1) points before this location are searched.

This scheme significantly reduces the computational complexity with no noticeable degradation in the output speech quality.
Quantization and Coding of LTP Parameters

The first LTP lag is coded with 5 bits and the remaining 7 lags with only 3 bits each. This, however, means that the LTP synthesizer, at the decoder, has to compute the corresponding lags by adding an offset to the first lag value.

<table>
<thead>
<tr>
<th>Input Threshold</th>
<th>Output</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\leq 0.10$</td>
<td>0.01</td>
<td>0 = 000</td>
</tr>
<tr>
<td>$\leq 0.30$</td>
<td>0.20</td>
<td>1 = 001</td>
</tr>
<tr>
<td>$\leq 0.50$</td>
<td>0.40</td>
<td>2 = 010</td>
</tr>
<tr>
<td>$\leq 0.70$</td>
<td>0.60</td>
<td>3 = 011</td>
</tr>
<tr>
<td>$\leq 0.90$</td>
<td>0.80</td>
<td>4 = 100</td>
</tr>
<tr>
<td>$\leq 1.10$</td>
<td>1.00</td>
<td>5 = 101</td>
</tr>
<tr>
<td>$\leq 1.30$</td>
<td>1.20</td>
<td>6 = 110</td>
</tr>
<tr>
<td>$&gt; 1.30$</td>
<td>1.40</td>
<td>7 = 111</td>
</tr>
</tbody>
</table>

Table 6.4: Input Thresholds and Output Levels of LTP Gain Quantizer

A uniform scalar quantizer was used to quantize and code the LTP gain parameters. Each gain parameter was coded with 3 bits. The quantizer threshold and output levels were computed using a large training data, and based on subjective testing. Table 6.4 shows the input and output levels of the quantizer and their corresponding code values.

LTP Filtering

LTP filtering is performed in the same manner as described in section 5.3.2, using the quantized values of the LTP gain parameters.

6.4.5 Codebook Search

Once the long and short-term correlations are removed from the input speech signal, the remaining signal, $e(i)$, should ideally be a very randomized signal. Based on this assumption, a codebook populated with Gaussian random sequences is generated.
The codebook is then exhaustively searched to find the best match for the input sequence.

In conventional CELP, the matching pattern is the one which maximizes $E_{\text{opt}}(i)$, [23]:

$$E_{\text{opt}}(i) = \max \frac{\left[ \sum_{i=1}^{N} e(i) \cdot (v_j(i) \ast f_j(i)) \right]^2}{\sum_{i=1}^{N} (v_j(i) \ast f_j(i))^2} ; \quad j = 0, ..., \ (\text{book size}-1) \quad (6.12)$$

and an optimum scale parameter, $\alpha_{\text{opt}}$, is calculated as:

$$\alpha_{\text{opt}} = \frac{\sum_{i=1}^{N} e(i) \cdot (v_j(i) \ast f_j(i))}{\sum_{i=1}^{N} (v_j(i) \ast f_j(i))^2} \quad (6.13)$$

where $v_j(i)$ are the codebook sequences, $f_j(i)$ is the truncated LTP filter response for the $j$th sequence, $N$ is the length of the sequence and (*) denotes the convolution process.

$(f_j(i) \ast v_j(i))$ is the response at the output of the LTP synthesis filter caused by the $n$th vector (selected sequence) in the codebook. If the minimum LTP lag is made equal to or greater than the length of the codebook sequence, the truncated response of the LTP filter will have a value of 1 in the first location and zero anywhere else. This will result in:

$$v_j(i) = f_j(i) \ast v_j(i) \quad j = 0, ..., \ (\text{book size}-1) \quad (6.14)$$

thus eliminating the convolutional process completely. Codebook search is, therefore, much simpler and consequently faster:
\[ E_{opt}(j) = \text{MAX} \frac{\left[ \sum_{i=1}^{N} e(i) \cdot v_j(i) \right]^2}{\sum_{i=1}^{N} v_j^2(i)} ; \quad j = 0, ..., \text{(book size-1)} \quad (6.15) \]

and

\[ \alpha_{opt} = \frac{\sum_{i=1}^{N} e(i) \cdot v_j(i)}{\sum_{i=1}^{N} v_j^2(i)} \quad (6.16) \]

The complexity of the search is further reduced by storing normalized codebook sequences i.e. \( \sum_{i=1}^{N} v_j^2(i) = 1 \), resulting in equations 6.17 and 6.18:

\[ E_{opt}(j) = \text{MAX} \left[ \sum_{i=1}^{N} e(i) \cdot v_j(i) \right]^2 ; \quad j = 0, ..., \text{(book size-1)} \quad (6.17) \]

\[ \alpha_{opt} = \sum_{i=1}^{N} e(i) \cdot v_j(i) \quad (6.18) \]

An 8-bit codebook containing sequences of 10 sample length each was generated off-line. The real-time code for implementing the two equations 6.17 and 6.18 was written in the most efficient manner possible.

When operating the DAU set of instructions on the DSP32, the flags corresponding to the current DAU instruction is set 3 instructions later. This is due to the pipelining architecture of the DSP32. There are two solutions to this problem. One, which is the non-optimized solution, is the insertion of the "No Operation" (NOP) instructions in between where the DAU instruction is executed and the instruction that
checks the flag. The other solution is the insertion of non-relevant instructions. In implementing the codebook search, we embedded equations 6.17 and 6.18 together and spent much effort in overcoming this latency. There were, however, few NOP instructions left in the code that can not be avoided.

**Parameter Quantization and Coding**

The 8 codebook indices are coded with 8 bits each. The quantization and coding of the codebook gain is important with respect to the output speech quality and thus needs certain considerations. If the threshold and the output levels are set low, the output speech will suffer loss and if set high a clipping effect is introduced. Different schemes of quantizing the gain parameters were investigated and implemented.

The method that provided the most accurate quantization was the one in which the gain parameter is normalized (before quantization) using the energy of the most recent segment of the LTP register. We observed that the optimum gain has a very similar magnitude response to the LTP memory energy. By normalizing the gain parameter using the energy of the LTP memory, some form of adaptive quantization is achieved which not only produces better quantization but also will require fewer number of levels (i.e. fewer bits per gain parameter). This scheme, however, assumes that the LTP memory at the encoder and decoder will always be the same at any given instant of time. This assumption cannot be fulfilled in reality due to the breakdowns of transmission or effects of channel errors on the transmitted parameters. This problem can, however, be overcome if voice activity detection is placed in the link and the LTP memories are reset during the silence periods.

During the simulation stages, based on a large training database, we noticed that the gain parameter may vary from very small values (~0.01) to values in the order of 100. In order to cover such a large range with only 4 bits (1 bit for sign) an efficient quantizer is needed. The non-uniform quantizer of figure 6.6 was therefore implemented in which both the step-sizes and the threshold levels are incremented (by using a fixed multiplication factor) from one level to the next. As shown in table 6.5, by setting the initial step-size and threshold to 1.0 and the multiplication factor, M, to 1.7 the quantizer is able to cover a range of input values up to 57.19, which is adequate for A-law coded input speech samples.
Fig. 6.6 Flow Chart of Non Uniform Scalar Quantization

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Positive Values</td>
</tr>
<tr>
<td>&lt; 1.0</td>
<td>0.500</td>
<td>0 = 0000</td>
</tr>
<tr>
<td>&lt; 2.7</td>
<td>1.850</td>
<td>1 = 0001</td>
</tr>
<tr>
<td>&lt; 5.59</td>
<td>4.145</td>
<td>2 = 0010</td>
</tr>
<tr>
<td>&lt; 10.50</td>
<td>8.047</td>
<td>3 = 0011</td>
</tr>
<tr>
<td>&lt; 18.86</td>
<td>14.679</td>
<td>4 = 0100</td>
</tr>
<tr>
<td>&lt; 33.05</td>
<td>25.954</td>
<td>5 = 0101</td>
</tr>
<tr>
<td>&lt; 57.19</td>
<td>45.122</td>
<td>6 = 0110</td>
</tr>
<tr>
<td>&gt; 57.19</td>
<td>77.708</td>
<td>7 = 0111</td>
</tr>
</tbody>
</table>

Table 6.5 Quantization and Coding of Codebook Gain
6.4.6 Bit Packing

The digital compression of voice signals, based on LPC analysis or methods employing vector quantization, result in a block or frame of parameters with a period of 10 to 30 ms. Frames of parameters are transported to the decoder, where synchronization with the frame is required in order to decode and reconstruct the voice signals. The synchronization information must be coded to leave as much of the available transport bandwidth free for the actual coded voice signal as possible, whilst still providing bit error immunity. In this case, input speech frames of 240 sample lengths are processed and compressed into 209 bits of coded data (every 30 ms) and an extra 79 bits are used for synchronization.

![Transmitted Speech Frame Diagram]

Fig. 6.7 Structure of Transmitted Speech Frame

The implementation of frame synchronization is described in detail in section 6.6 and here it suffices to say that the use of the two unique words of length 12 (one for marking the start and the other marking the end of the frame) in conjunction with 'Bit Stuffing' scheme achieves a fast and robust synchronization.

For this application, the Bit Packing procedure of the GSM coder (see section 5.4.9) is modified, eliminating the bit classification. This is to say that code values (for
example) 001, 101 and 111 ... are packed into 001101111... pattern and transmitted. The bit stuffing is widely used in packetized transmission of data and although not very efficient in terms of minimizing 'wasted bits' has the advantages of being simple and providing fast acquisition. The Bit Packer places the two unique words 0xffff and 0xff8 at the start and end of the actual coded data (see figure 6.7). In addition, in order to avoid the coded data from mimicking the unique words, each code value is 'stuffed' with an extra bit (a zero bit in the LSB position). This results in a total frame size of 288 bits corresponding to a bit rate of 9.6 kbit/s (with 2 unused bits).

6.5 Software Implementation of Decoder

A single AT&T WE-DSP32 (running at 25 MHz) is dedicated to the real-time execution of the speech decoder. The real-time code only occupies a fraction of the DSP processing power and the entire encoding and decoding can be performed on a single DSP32, but lack of interrupt lines makes this task very difficult, leading to a more expensive solution due to the need for extra components. Although, there was not such a strict constraint on the processing power as in the case of the encoder, but great care and effort was taken to deliver an optimized code for the speech decoder. This makes it possible to include other additional functions such as echo cancellation or noise suppression at the decoder, when the need arises.

6.5.1 Data Acquisition

Every 30 ms the decoder will receive 288 bits of coded information which are then decoded into 240 reconstructed speech samples. The acquisition of the data and the transfer of speech samples to the D/A is based on the modified dual input and output buffering system. For the VSAT coder, frame synchronization information is transmitted by the encoder and the decoder will process the received frame of coded information and align the received frame (see figure 6.8). The synchronization procedure is fully documented in section 6.6 of this chapter.

Once the received frames are aligned, the 'Bit Unpack' procedure will turn the continuous string of data into corresponding code values for each parameter. Every code value is bit stuffed with an extra '0' which needs to be discarded.
6.5.2 Long-Term Synthesis

Input Excitation

The input excitation to the LTP synthesis filter is formed by selecting a codebook sequence and scaling it appropriately. For every sub-block, the decoder will receive a code value ranging between 0 and 255, corresponding to a codebook entry (in the same order). It also receives a code value ranging between 0 and 15 for the codebook gain parameter. The simplest and fastest way of decoding the code values is to store arrays of
the quantizer output levels (look-up tables) at the decoder and then by setting up a floating-point pointer (i.e. pointer value = code value x 4) and adding it to the base address of the array, the corresponding quantized output level is found. For the codebook gain, the 8 values in the second column in table 6.5 are stored in the memory. Once a value is selected, the MSB of the code value will determine the sign of the gain parameter. Each sub-block sequence is therefore multiplied by a gain value to form the excitation signal of the LTP synthesis filter.

**Decoding of LTP Parameters**

The 8 floating-point values of the second column of table 6.4 are stored in the memory. For each sub-block, one of these values is selected as the LTP gain parameter in the same manner as explained in the previous section.

For the first sub-block, the LTP lag code will have a value between 0 and 31 corresponding to the decimated lag values of 10 to 41 (lag value = size of sub-block + Lag code). For the other 7 sub-blocks, a code value between 0 and 7 is received corresponding to the 8-points of the window. Based on the conditions of equation 6.11, an appropriate shift is computed and added to the lag value of the first sub-block, producing the lag value of the present sub-block.

**LTP Synthesis Filter**

The decoded values of LTP gain and lag parameters, in conjunction with equation 5.6, are used to implement the LTP synthesis filter.

**6.5.3 Short-Term Synthesis**

**Input Excitation**

The input excitation to the LPC synthesis filter is the up-sampled version of the output of the LTP synthesis filter. For every sub-block a code value between 0 and 2 is received corresponding to the 3 grid positions. An up-sampled sequence of length 30 is formed by inserting two zeros in between each sample value of the LTP synthesis filter output. The first sample value is placed in locations 0, 1 or 2 according to the received grid position.

In terms of computational complexity, it is best to find the complete excitation (240 sample length) for the LPC synthesis filter rather than for every sub-block.
Decoding of LARs

The 10 LAR code values can again be decoded by employing the look-up table approach. We found that it was best to implement the decoder in a more universal way. This not only reduces the computational load but also has the flexibility of tuning the quantizers more easily. Like the encoder, the mean values and step-sizes of table 6.2 are also stored in the decoder and the reverse operation of the flow chart in figure 6.5 produces the 10 corresponding LAR values. These are then transformed back into reflection coefficients (see equation 6.3).

LPC Synthesis Filter

The implementation of the LPC synthesis filter is based on the lattice structure and like the inverse filter the pipeline latencies of the DSP32 are used to advantage in reducing the complexity. The real-time code will only require (p+1) locations as a scratch pad for a pth order filter.

6.5.4 Post Processing

The post-processing stage comprises execution of the following procedures:

- Deemphasis,
- Post Filtering,
- Conversion of floating-point representation of speech samples into A-law Codes.

Deemphasis

The preemphasis of the input speech samples at the encoder will require the deemphasis of the reconstructed speech samples at the decoder. A first order IIR filter with a deemphasis factor of 0.86 is therefore implemented:

\[ S_{DE}'(k) = S'(k) + 0.86 \cdot S_{DE}'(k-1) \quad ; \quad k = 0, \ldots, 239 \]  

(6.19)

where \( S'(k) \) is the output of the short-term synthesis filter.

Post Filtering

In order to further enhance the quality of the output speech, an adaptive post filter is added. The post filter modifies the envelope of the quantization noise contaminating the speech, making it less noticeable to the human ear.
Full details of the post filter operation and its implementation are to be found in the next chapter.

A-law Code Conversion

Finally, the floating-point output speech samples are transformed into their corresponding A-law codes (8-bit each) according to the CCITT recommendations. This is achieved by using one of the 'Special Functions' present on the DSP32 set of instructions.

6.6 Frame Synchronization

Coding techniques such as CELP operate on a block of input speech samples and usually have long frame periods (10-30 ms). Therefore, it is necessary for an implementation of a coder of this type to incorporate a means of self synchronization to the coded signals so that various data elements in the frame can be identified. Synchronism with the coded signal must be established within a reasonable period of time for voice communication, preferably a fraction of a second. The synchronizing mechanism must be continuous so as to deal with bit errors and other causes of loss of synchronization.

Methods for determining alignment in a digital bit stream are well known [24-26]. One of the simplest forms of frame synchronization is to include one extra bit per frame, which carries a known data pattern that is unlikely to occur in any other position in the frame. The decoder then searches for a regular occurrence of this pattern on one particular bit in the frame and labels that one as the sync bit. A steady zero is not a good pattern since it will often occur in the speech data bits during pauses in speech, and a steady one is equally bad since it will often occur during large increases in speech energy. An alternating one-zero pattern is also quite likely because of the nature of the quantizers which tend to generate samples with alternating amplitude levels when the input level is constant. Therefore, the coded bit string needs to be carefully considered before the selection of the unique pattern [27]. This synchronization technique has the disadvantage that it requires one bit per frame of additional data, and the time to acquire synchronization can be quite significant if errors are present on received data, perhaps 30 to 100 frames of data need to be analyzed taking up to half a second. In order to improve the acquisition time, two frame bits can be used with normally one bit set to the inverse of the other [28].
An alternative frame synchronization technique which is quite often employed in HF radio communications, where acquisition time is important, is achieved using 'silence' frames [29, 30]. This technique avoids the need for any additional sync bits in the frame, while retaining the facility to monitor and resynchronize if necessary, during the speech pauses. It allows a short synchronization acquisition time in half-duplex operation, as only a few frames of silence are required at the beginning of each transmission. The disadvantage of this technique is that silence frames are not generated during speech pauses if the acoustic operating environment is noisy, and loss of synchronization may not be detected at the receiver.

A method that is capable of providing fast synchronization acquisition time and is relatively robust to the effects of random channel errors, is the 'Bit Stuffing' method. In this technique, one or two unique patterns (unique words) will mark the start and end of the frame and the actual coded bits are 'stuffed' with extra bits in order not to mimic the unique patterns. Unlike, the two previous techniques, this will require a larger number of frame bits ('wasted bits') but proves to be a more reliable solution. The VSAT coder is designed to operate in a mobile environment where environmental noise and channel errors will affect the transmitted speech data. We, therefore, elected to use the stuffed synchronization technique, with unique words 0xfff (UWs) marking the start and 0xff8 (UWe) marking the end of the frame. The use of two unique words has the advantage of providing a shorter acquisition time and a more robust synchronization scheme.

In this section, we will discuss how this synchronization technique is implemented and consider the robustness and acquisition time of the scheme.

6.6.1 Modified Dual Buffering System

Figure 6.9 illustrates the flow chart of the modified dual buffering system. In this system, every 30 ms, a decision has to be made as to whether the captured frame of data (288 bits) is aligned or not. If the input data is synchronized, the data flow will continue in the normal way, otherwise alignment is required. The synchronization technique takes a period of two frames (60 ms), due to the dual buffering system, used to align the input data buffers. During this time, the output speech buffers can either be reconstructed or muted. We decided to mute the output buffers since long breakdowns of the transmission may produce undesired signals at the output, in the case of the frame reconstruction.
Modified Dual Buffering System

Capture 288 Bits & Store in Input Data Buffer
Output 240 Speech Samples From Output Buffer

No

Is Input Data Buffer Full?

Yes

Swap Input Data Buffers
Swap Output Speech Buffers
Find the Shift (d, bytes) From Stored Input Data Buffer

No

Is There Any Shift?

Yes

Mute the Output

No

Is Input Data Buffer Full?

Yes

Delay Input Data String For d Bytes
Store 288 Bits Into Input Data Buffer
Mute The Output

No

Is Input Data Buffer Full?

Yes

Swap the Input & Output Buffers
Process The Data
Store Samples in Output Buffer

Process The Data
Store Samples in Output Speech Buffer
In order to align the input data string, the first frame is analyzed and a delay value (or shift, in number of bytes), \( d \), is measured. The next frame is then delayed by \( d \) bytes i.e. is written into a dummy location and the rest transferred into the appropriate input data buffer.

In the following sections, we will explain (i) how a synchronization loss is detected, (ii) how the delay or shift is computed and (iii) how the alignment is performed.

### 6.6.2 Frame Boundary Detection

Every input data buffer (containing 288 bits) is first analyzed to determine a synchronization loss. In case of misalignment, the amount of shift needs to be measured and then the input string is delayed by this amount. Figure 6.10 shows all the possible states of the input data buffers and figure 6.11 illustrates the flow chart of the implementation of synchronization loss detection. A search routine is implemented in which the input buffer is scanned bit by bit to detect the presence and location of the 'Start Unique Word' (UWs) and 'End Unique Word' (UWe). Two flags, \( S_{flag} \) and \( E_{flag} \), are set to 'True' or 'False' depending on the presence or absence of the unique words. Two other parameters, \( SH_s \) and \( SH_e \), are also computed corresponding to the distance of the UWs and UWe, respectively (in number of bits), from the beginning of the input buffer. Based on these two parameters and the two flags, a decision is made as to whether a misalignment of the data exists and if so, a 'SHIFT' value is computed. The SHIFT value is then used in a 'Frame Adjustment' procedure to align the input data buffers. If both UWs and UWe are not found (case (a) of figure 6.10), it is safe to assume that a breakdown in the transmission or a burst error condition has occurred and therefore output is muted until the acquisition of a good link is obtained.

In all cases except state (d), the received coded data in the input buffers can be processed after alignment. In case (d), part of the data belongs to the previous frame (or frames) and has to be discarded.

### 6.6.3 Frame Adjustment

The implementation of the synchronization technique for this particular application proved to be extremely tedious due to the fact that the coder (i) uses a
Input Data Buffer

(a) Both Unique Words Start and End are corrupted. Output Muted.

(b) Unique Word Start not detectable. Data can be processed.

(c) Unique Word End not detectable. Data can be processed.

(d) Both Unique Words are detectable. Part of Data belongs to another frame.

(e) Aligned Frame (synchronized).

(f) Both Unique Words are detectable. Data can be processed.

Fig. 6.10 Possible States of Input Data Buffers
Fig. 6.11 Flow Chart of Synchronization Loss Detection
DMA dual input buffering system, (ii) needs fast synchronization and (iii) the input and output data is received and transmitted in bytes.

As the flow chart in figure 6.12 demonstrates, the frame adjustment routine initially mutes the output data by setting all sample values to zero (or a very small value) and then converts the SHIFT value into the corresponding number of bytes, B. The input buffer, operating under the DMA, is then delayed for B bytes. Finally, the input and output buffers are swapped and the execution flow returns to the main dual buffering system.
6.7 Computational Complexity and Memory Usage

The DSP32 provides on-chip memory which includes 2k bytes of ROM and 4k bytes of RAM. In addition, up to 56k bytes of directly accessible memory can be added off-chip. The entire memory (both internal and external) can be arranged in four different configurations as demonstrated in figure 6.13. The DSP32 memory is divided into two banks, an upper bank (1) and a lower bank (0). Although, memory accesses can be made without regard to the upper and lower banks, to achieve maximum throughput, memory accesses must alternate between the two banks. As the one memory bank is accessed, the other bank is being addressed. This form of pipelining reduces the effective memory access time by one-half [31].

Although, the maximum range of memory addresses that can be accessed by the DSP32 is 64k bytes, the actual amount of memory available is less and is dependant on the mode of operation. In all of our implementations, memory mode 2 is chosen since this mode allows the maximum number of accessible addresses (62k bytes).

In order to increase the throughput i.e. increasing the execution speed of the programme, the number of 'Conflict Wait States' needs to be minimized. If two consecutive accesses are directed toward the same bank of memory, the DSP32 control unit automatically inserts one conflict wait state, invisible to the user. Each wait state equals 1/4 of an instruction cycle (stealing one clock cycle). It is virtually impossible to write a DSP32 code with zero number of wait states, however, in our implementations, this number is minimized by directing the code corresponding to the programme into bank (0) and all the data storage into bank (1).

In the following sections, the computational complexity and memory usage of the encoder and the decoder are documented. The computational complexity measurements are given both in terms of total clock cycles and number of operations for each subroutine. Each instruction takes 4 clock cycles to execute, assuming there are no conflict wait states. The columns showing the number of clock cycles include the DSP32 inserted number of conflict wait states.
Fig. 6.13 DSP32 Memory Address Configurations [31]
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>No. of Clock Cycles</th>
<th>No. of Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-Processing</td>
<td>5,817</td>
<td>1,214</td>
</tr>
<tr>
<td>Preemphasis</td>
<td>5,322</td>
<td>1,210</td>
</tr>
<tr>
<td>Autocorrelation</td>
<td>26,323</td>
<td>5,282</td>
</tr>
<tr>
<td>Durbin</td>
<td>5,047</td>
<td>1,130</td>
</tr>
<tr>
<td>LAR Conversion</td>
<td>3,062</td>
<td>663</td>
</tr>
<tr>
<td>LAR Quantization</td>
<td>3,508</td>
<td>822</td>
</tr>
<tr>
<td>Inverse LAR Conversion</td>
<td>2,835</td>
<td>614</td>
</tr>
<tr>
<td>Inverse Filtering(lattice)</td>
<td>93,388</td>
<td>18,967</td>
</tr>
<tr>
<td>Weighting Filter</td>
<td>13,804 x 8</td>
<td>3,286 x 8</td>
</tr>
<tr>
<td>Grid Selection</td>
<td>650 x 8</td>
<td>141 x 8</td>
</tr>
<tr>
<td>LTP Analysis</td>
<td>4974 x 8</td>
<td>1,055 x 8</td>
</tr>
<tr>
<td>Codebook Search</td>
<td>36,280 x 8</td>
<td>7,723 x 8</td>
</tr>
<tr>
<td>Optimum Gain Quantization</td>
<td>207 x 8</td>
<td>49 x 8</td>
</tr>
<tr>
<td>LTP Memory Update</td>
<td>739 x 8</td>
<td>148 x 8</td>
</tr>
<tr>
<td>Bit Packing</td>
<td>6,719</td>
<td>1,675</td>
</tr>
<tr>
<td>Post-Processing</td>
<td>492</td>
<td>114</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>605,745</strong></td>
<td><strong>123,935</strong></td>
</tr>
</tbody>
</table>

\[= 24.23 \text{ ms}^* \]

\[= 19.83 \text{ ms}^* \]

* @ 25 MHz Clock Rate

Table 6.6 Computational Complexity of VSAT Encoder

### 6.7.1 Encoder Complexity and Memory Requirements

Table 6.6 shows the computational complexity of the VSAT encoder. A major part of the processing power is spent in the analysis-by-synthesis stage where each subroutine is executed 8 times per frame, corresponding to the 8 sub-segments. The codebook search in the AbS procedure is responsible for consuming a large part of this power due to its exhaustive search nature. The other major complex subroutine, is the LPC inverse filter, even though the lattice structure of this filter is realized in a very
efficient manner. The entire encoding process takes 605,745 clock cycles per frame to execute, corresponding to a total processing period of 24.23 ms with the DSP32 operating at a rate of 25 MHz. Excluding the number of conflict wait states, 123,935 operations (or instructions) are needed to process a 30 ms frame of speech i.e. 19.83 ms at 160 ns/instruction. A tolerance of ± 5% needs to be added to this figures, as the processing time varies form one frame to another. The serial input and output DMA also adds an extra processing time to these figures.

<table>
<thead>
<tr>
<th>Section</th>
<th>No. of Bytes</th>
<th>Percentage of Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programme</td>
<td>3,136</td>
<td>8.5 %</td>
</tr>
<tr>
<td>Data</td>
<td>15,008</td>
<td>40.7 %</td>
</tr>
<tr>
<td>TOTAL</td>
<td>18,144</td>
<td>49.2 %</td>
</tr>
</tbody>
</table>

Table 6.7 Memory Requirements of VSAT Encoder

Table 6.7 shows the memory requirements of the VSAT encoder. Of the total available memory (32k bytes of external memory + 4k bytes of internal RAM), less than 50% (17,960 bytes) are used for storing encoder instructions and data. The memory allocation of the data far exceeds that of the programmes due to the dual input and output buffering system.

6.7.2 Decoder Complexity and Memory Requirements

Table 6.8 illustrates the computational complexity of each stage of the VSAT decoder. The two most complex subroutines are the LPC synthesis filter and the post filtering. The large complexity of the LPC synthesis filter is due to its lattice structure implementation and that of the post filtering is due to the fact that the reconstructed speech frame first passes through an LPC inverse filter and then through an LPC synthesis filter, using the modified LPC coefficients. The decoding process requires
204,588 clock cycles of the DSP32 per frame to be executed, corresponding to a processing time of 8.18 ms (of the allowed 30 ms period), with the DSP32 running at a rate of 25 MHz. Assuming no conflict wait states are generated, 48,575 instructions are executed every 30 ms corresponding to 7.77 ms at 160 ns/instruction. Once again, the

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>No. of Clock Cycles</th>
<th>No. of Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Word Search</td>
<td>15,102 x 2</td>
<td>3,775 x 2</td>
</tr>
<tr>
<td>Output Mute</td>
<td>2,208</td>
<td>492</td>
</tr>
<tr>
<td>Bit-to-Byte</td>
<td>46</td>
<td>11</td>
</tr>
<tr>
<td>Pre-Processing</td>
<td>512</td>
<td>119</td>
</tr>
<tr>
<td>Byte Alignment</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>Bit Unpacking</td>
<td>7,278</td>
<td>1,806</td>
</tr>
<tr>
<td>LTP Gain Decoder</td>
<td>62 x 8</td>
<td>15 x 8</td>
</tr>
<tr>
<td>LTP Lag Decoder</td>
<td>50 x 8</td>
<td>12 x 8</td>
</tr>
<tr>
<td>Optimum Gain Decoder</td>
<td>107 x 8</td>
<td>26 x 8</td>
</tr>
<tr>
<td>LTP Synthesis Filter</td>
<td>149 x 8</td>
<td>32 x 8</td>
</tr>
<tr>
<td>Up-Sampling</td>
<td>219 x 8</td>
<td>44 x 8</td>
</tr>
<tr>
<td>LTP Memory Update</td>
<td>388 x 8</td>
<td>92 x 8</td>
</tr>
<tr>
<td>LAR Decoder</td>
<td>604</td>
<td>141</td>
</tr>
<tr>
<td>Inverse LAR Conversion</td>
<td>2,835</td>
<td>614</td>
</tr>
<tr>
<td>LPC Synthesis Filter</td>
<td>84,308</td>
<td>19,217</td>
</tr>
<tr>
<td>Deemphasis</td>
<td>3,882</td>
<td>970</td>
</tr>
<tr>
<td>Post Filtering</td>
<td>62,835</td>
<td>15,369</td>
</tr>
<tr>
<td>Post Processing</td>
<td>2,036</td>
<td>508</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>204,588</td>
<td>48,575</td>
</tr>
<tr>
<td></td>
<td>( \equiv ) 8.18 ms*</td>
<td>( \equiv ) 7.77 ms*</td>
</tr>
</tbody>
</table>

* @ 25 MHz Clock Rate

Table 6.8 Computational Complexity of VSAT Decoder
VSAT coder as with the GSM coder needs more processing power at the encoder stage. This is more or less a characteristic of all the LPC based low bit rate speech coders, in which all the analysis is performed during the encoding stage. The figures quoted in table 6.8 have to be considered with a tolerance of $\pm 5\%$ to allow for the variations in the processing flow.

<table>
<thead>
<tr>
<th>Section</th>
<th>No. of Bytes</th>
<th>Percentage of Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programme</td>
<td>3,096</td>
<td>8.4 %</td>
</tr>
<tr>
<td>Data</td>
<td>14,864</td>
<td>40.3 %</td>
</tr>
<tr>
<td>TOTAL</td>
<td>17,960</td>
<td>48.7 %</td>
</tr>
</tbody>
</table>

Table 6.9 Memory Requirements of VSAT Decoder

Table 6.9 shows the memory allocation of the VSAT decoder. The total size of the memory used is 18,144 bytes i.e. 49.2% of the total available size. The programme storage occupies 8.5% of the memory and the data 40.7%. As with the encoder, serial input and output DMA is responsible for the major part of the data storage requirements.

6.8 VSAT Speech Coder Hardware

The hardware occupies a double extended Eurocard and is based on the AT&T WE-DSP32 digital signal processor. The board comprises two halves; the encoder and the decoder. Each half is independent of the other and consists of a DSP32, 8031 micro controller, A-law PCM codec, and data input and output circuitry. Although, a small number of components are shared between the two functions, each half will operate independently of the other. For example, a loss of decoder functionality will not necessarily lead to the loss of the encoder functionality.
Figures 6.14 shows the circuit block diagram of the speech coder. The description of either half of the board is equally applicable to the other half. Each half of the board comprises the following units:

(i) Micro Controller Unit,
(ii) Digital Signal Processor Unit,
(iii) Input and Output Unit,
(iv) Phase Locked Loop (PLL) Unit.

6.8.1 Micro Controller Unit

The DSP32 is controlled by an 8031 micro controller. The controller comprises an 8031, EPROM, RAM, glue logic, crystal, RS232 interface, lost clock circuit, and reset and watchdog. A 3 wire RS232 interface is available for downloading application programmes into the board, which is an extremely useful tool for debugging of the software during the software-hardware integration phase. The reset/watchdog device provides a reliable power up reset pulse of 50 ms duration from the power supply reaching its operating level. This device also forces a reset of the 8031 if the power supply falls below 4.6 volts, ensuring no processor activity near to the operating limits. The watchdog timer facility has a timeout period of 1.4 seconds and ensures reliable operation of the hardware in the field.

The lost clock circuit monitors the 9.6 kHz data clock, which must be 9.6+ 0.13 kHz at all times, and interrupts the 8031 if the clock fails. The lost clock detection consists of a monostable continually triggered by the 9.6 kHz clock, if the trigger is lost or extended beyond the timeout period of the monostable, the output of the device changes state and an interrupt is forced onto the 8031.

6.8.2 Digital Signal Processing Unit

The digital signal processing unit comprises an AT&T WE-DSP32, 32k bytes of fast static RAM, 24 MHz oscillator module, reset circuit, and a serial interface detailed in the next section. The DSP32 programme is downloaded from the 8031 micro controller.
Fig. 6.14. Block Diagram of XSAT Speech Codec Hardware
to the DSP32 static RAM at power up or reset. The 24 MHz oscillator module enables a peak floating point performance of 12 million floating-point operations per second (MFLOPS).

6.8.3 Input and Output Unit

The analogue speech signals are converted into 64 kb/s A-law coded samples via the on-board or an external PCM codec. Similarly, the 64 kb/s A-law coded speech samples, at the output of the decoder, are converted into analogue form using the on-board or external PCM codec.

The encoder DSP32 accepts the 64 kb/s PCM coded speech samples and after processing, the coded data appears on the DSP32 serial output port at a nominal rate of 2.048 Mb/s and therefore requires translation into a slower rate of 9.6 kb/s. The frequency translation is performed by a serial to parallel and parallel to serial converter. The decoder DSP32 receives the coded data at a nominal rate of 2.048 Mb/s that has been translated from the incoming 9.6 kb/s channel. The decoder DSP32 outputs 64 kb/s PCM coded data through the serial port for conversion to an analogue speech signal.

6.8.4 Phase Locked Loop Unit

The 9.6 kHz transmit and receive clocks are independent and thus two phase locked loops (PLL) are needed. The PLL is implemented using a digital phase locked loop (DPLL) device. The locking range of the DPLL can be digitally programmed and hence gives greater flexibility.

The DPLL is used to lock together the incoming 9.6 kb/s data clock and the 2.048 MHz PCM codec clock. In the absence of any 9.6 kHz clock, the output of the DPLL is 4.096 MHz. This clock is divided by 2 to produce the 2.048 MHz codec clock and then divided by 213 to produce a 9.6 kHz clock to compare against the incoming 9.6 kHz data clock. Any changes in frequency between the two clocks forces the DPLL to modify the 4.096 MHz output frequency in order to bring the two clocks back into synchronism.
Fig. 6.15 V-SAT Speech Coder Hardware (9.6 kbit/s)
6.8.5 Electrical and Mechanical Interfaces

Figure 6.15 illustrates the VSAT speech coder board. The board is a double extended Eurocard, 233 mm by 220 mm, with a three row fully populated DIN41612 connector providing the board interface to the outside world. The total power consumption of the board is 7.5 + 1.25 Watts and requires +5, ± 15 V power supplies.

There are two LEDs on the board, one for each function of encoder and decoder. The alarm flag on the edge connector is valid whenever either of the LEDs show an active alarm state. The LEDs are active under the following conditions;

(i) failure of the micro controller to power up,
(ii) failure of the DSP32 and micro controller interface,
(iii) loss of the 9.6 kHz clock, and
(iv) absence of the 9.6 kHz clock at power up.

6.9 Coder Performance

Currently the coder is undergoing field trials in the Far East where the coder has been integrated into a complete VSAT system. The initial results are satisfactory. The coder was laboratory tested and proved to be conforming to all the stated requirements.

<table>
<thead>
<tr>
<th>BER</th>
<th>Quality (MOS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-6}$</td>
<td>3 Min.</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>3 Min.</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>2 Min.</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>1 Min.</td>
</tr>
</tbody>
</table>

Table 6.10 Speech Quality Requirements of the VSAT Coder
Table 6.10 shows the speech quality requirements of the coder, in a back-to-back configuration, under random error conditions, with the Mean Opinion Score (MOS) scale of 1 to 5. A limited subjective test was performed and it was found that the coder outperformed all the requirements of table 6.10.

The coder was also required to cope with the variations in the external data clocks with a nominal rate of 9.6 kHz, minimum of 8.6 kHz and a maximum of 10.6 kHz. This is easily accommodated with the inclusion of the DPLL on the board.

The coder has a back-to-back delay of 90 ms as compared to the required maximum delay of 100 ms. This delay corresponds to 3 speech frames (30 ms each) and is partly due to the LPC based nature of the coder and the dual input and output buffering system.

The board was also tested mechanically and survived dropping from a height of 10 cm onto a hard surface (at any corner).

6.10 Concluding Remarks

The simultaneous rapid advances in both satellite communication technology and applicable electronics hardware is changing the traditional role of satellites. The great demand, by small private business and defence organizations in particular, for a compact, power efficient, and reliable portable communication link has lead to development of an alternative usage for satellites. The new communication systems, with VSAT Networks, are now manufactured and researched upon all over the world. Every day, new VSAT systems are produced offering the user a wider choice of facilities.

In this chapter, an overview of a VSAT system (MP-SP2300) manufactured in the UK, was briefly presented. The system required the development of a robust and efficient speech coder at 9.6 kbit/s. We reported on the development and refinements of a new speech coding algorithm, developed in the Surrey University research group, to suit such a system. The speech coder was then real-time implemented, using a recently available floating-point DSP (AT&T WE-DSP32), and was integrated into a very flexible hardware. Many functions of the hardware are user-configurable allowing total flexibility for different set-ups.
The three usual stages of (i) algorithm development, (ii) software implementation and optimization, and (iii) hardware realization are fully documented. The study of the coder performance provided results superior to those required of the coder. The speech coding hardware has been successfully integrated into a VSAT network and is currently operating satisfactorily in the Far East.
References


9. -, "VSAT SCPC DAMA Networks", Multipoint Communications Ltd.


CHAPTER 7

Speech Coder For INMARSAT-M System

7.1 Introduction

The origins of the International Maritime Satellite Organization, commonly known as INMARSAT, date back to 1966, when the Inter-governmental Maritime Consultative Organization (IMCO), an agency of the United Nations began studying the operational requirements for a maritime satellite system. Although, the INMARSAT organization was officially formed in 1979, it was 3 years later, in 1982, that it assumed operational responsibility with the introduction of the Standard-A maritime communication system. Some 54 countries own the organization with its initial mission of providing international mobile ocean communications via satellites to ship operators and owners in the Atlantic, Indian and Pacific regions. The organization is now in the process of expanding into the provision of sea to air and land communication services [1].

INMARSAT, through its first communication system (Standard-A), has provided high quality telephony, facsimile, data and telex services to ships via its satellites since 1982. The system has proved very popular and currently an estimated 9000 Standard-A Mobile Earth Stations (MES) are in operation, with approximately 10% of these being transportable and used on land. The speech transmission technique used is Companded FM (CFM) in a SCPC mode [2].
INMARSAT later introduced a smaller ship earth station known as Standard-B. The system provides the same services as the Standard-A with the same quality objectives. The analogue voice channel is replaced by a digital one, employing 16 kbit/s Adaptive Predictive Coding (APC) with 4-phase Phase-Shift Keying (PSK) and 1/2 rate Forward Error Correction (FEC).

Standard-C is a new type of service using a very small terminal for a more limited range of activities, including emergency back up capacity to Standard-A and B terminals. It will be more attractive to operators of small vessels due to its lower cost. Standard-C is basically a store-and-forward message transfer system using a low-gain near omnidirectional antenna and capable of two-way teletext and low speed data [1].

A new Aeronautical Telephone Service, called Skyphone, will soon be operational providing telephony via the INMARSAT satellites to and from aircraft. The service will be open to the use of passengers as well as the flight crew as a means of reliable voice communication with Air Traffic Control. The speech coding system is a digital system and employs 9.6 kbit/s Multipulse Excited Coding (MPE-LPC) [4].

Currently, INMARSAT is in the process of developing a new standard, INMARSAT-M, to provide communications quality voice, data and facsimile services via the PSTN network or via closed networks to land-mobile, land transportable and maritime users (see figure 7.1). The system is intended to address large new land-mobile and maritime markets which are constrained by equipment size and costs. The main objective of the system is to provide a compact system with high efficiency in its use of satellite resources, whilst maintaining acceptable service quality. It is anticipated that some 340,000 INMARSAT-M terminals will be operational by the year 2000, of which approximately 9% would be maritime. The system is digital thus allowing efficient use of the space segment (hundreds of channels can be supported) resulting in a good basis for reasonable end user charges [2].

During the initial stages of developing INMARSAT-M system, INMARSAT required, as a proof of concept, the development and implementation of a digital speech coder operating at 4.8 kbit/s. The coding scheme needed to be robust, produce acceptable speech quality and to be integrated into a compact and power efficient hardware. As a result, the CELP-BB coding scheme, described in chapter 6, underwent considerable amount of modifications and refinements with emphasis put on the channel error robustness. A robust coding scheme at 4.8 kbit/s was achieved, and the second
Fig. 7.1 Inmarsat-M Network Configuration [7]
generation of the AT&T floating-point DSPs, WE-DSP32C, which had just been released was used to realize the coder in real-time. The latest state of technology was then used to design a very small hardware, a single Eurocard size board, to implement both speech encoder and decoder into one single DSP32C [5]. INMARSAT later decided on the adoption of a 6.4 kbit/s (combined source and channel coding). The coding algorithm at 4.8 kbit/s was modified with the addition of channel coding and put forward for the INMARSAT Selection Procedure [6].

In chapter 6, the basic principles of CELP-BB coding are documented in detail. In this chapter, after a brief system overview, we will report only on the modification made to the scheme and consider the software and hardware implementations of the coder. Finally, the performance of the coder will be considered.

### 7.2 Overview of the System

The followings are the major elements of the baseline INMARSAT-M system [7]:

- Space Segment,
- Mobile Earth Stations (MES),
- Land Earth Stations (LES),
- Network Coordination Stations (NCS).

The function of these four system elements are combined to form the following major INMARSAT-M sub-systems;

(a) Communication Sub-System: providing the demand assigned satellite SCPC communication links between MESs and LESs, with extensions into the terrestrial networks. The same physical SCPC channel can be used either to support digital voice services, optional data services, and in-band signalling under the control of the Access Control and Signalling System (ACSE).

(b) Access Control and Signalling System (ACSE): providing the satellite signalling links between MESs, LESs and NCSs. INMARSAT-M will use the identical ACSE system as Standard-B telephony services.
Network Coordination Station (NCS)

- Return SCPC Channel
- SCPC MESV and SUB
- SCPC MESD and SUB
- SCPC MES-SIG and SUB

Forward SCPC Channel

<table>
<thead>
<tr>
<th>Channel Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCPC</td>
<td>Single Channel Per Carrier</td>
</tr>
<tr>
<td>NCSA</td>
<td>NCS Assignment Channel</td>
</tr>
<tr>
<td>NCSC</td>
<td>NCS Common Channel</td>
</tr>
<tr>
<td>NCSS</td>
<td>NCS Spot-beam Channel</td>
</tr>
<tr>
<td>NCSI</td>
<td>NCS Inter-station Channel</td>
</tr>
<tr>
<td>LESI</td>
<td>LES Inter-station Channel</td>
</tr>
<tr>
<td>SUB</td>
<td>Subband Signalling Channels</td>
</tr>
<tr>
<td>MESV</td>
<td>MES Voice Channels</td>
</tr>
<tr>
<td>MESD</td>
<td>MES Data Channels</td>
</tr>
<tr>
<td>MES-SIG</td>
<td>MES In-band Signalling Channels</td>
</tr>
<tr>
<td>MESRQ</td>
<td>MES Request Channel</td>
</tr>
<tr>
<td>MESRP</td>
<td>MES Response Channel</td>
</tr>
<tr>
<td>LESV</td>
<td>LES Voice Channels</td>
</tr>
<tr>
<td>LESD</td>
<td>LES Data Channels</td>
</tr>
</tbody>
</table>

Fig. 7.2 Inmarsat-M System Functional Channel Configuration [7]
Figure 7.2 illustrates the functional satellite channels used for communication services and signalling in the INMARSAT-M system. A number of these channels could share the same physical carrier amongst themselves and also with Standard-B channels, especially during the initial years of INMARSAT-M system operation. The following is a brief description of each functional channel;

(a) SCPC Channel: mandatory SCPC channel used in both directions to support the baseline 6.4 kbit/s full-duplex digital voice services and the optional 2.4 kbit/s data services. The SCPC channels in both directions can operate in either one of the three exclusive modes under the control of the INMARSAT-M ACSE; (i) Voice, (ii) Data (optional), and (iii) In-Band Signalling mode.

(b) LES Assignment (LESA) Channel: (optional) TDM channel having identical channel format as the NCS TDM, used in the forward direction to carry LES signalling messages to MESs including channel assignments for mobile-originated calls and selective clearing of malfunctioning MESs.

(c) LES Inter-station (LESI) Channel: used by each LES to carry signalling information to the NCS in the satellite network.

(d) NCS Common (NSCC) Channel: TDM channel, used in the forward direction to carry NCS signalling messages including call announcements, network status information (Bulletin Board) and selective channel clearing.

(e) NCS Assignment (NCSA) Channel: TDM channel, used in the forward direction to carry channel assignment messages for both mobile-originated and fixed-originated calls.

(f) NCS Inter-station (NCSI) Channel: used by the NCS to carry signalling information to the LESs in the satellite network.

(g) NCS Spot-Beam (NCSS) Channel: TDM channel, transmitted in the forward direction (one frequency per spot beam) to enable MESs to identify the spot beam in which they are located.

(h) MES Request (MESRQ) Channel: random access (slotted Aloha) channel used in the return direction to carry MES signalling information, specifically to request signals which initiate a mobile-originated call.
(i) MES Response (MESRP) Channel: TDMA channel used in the return direction to carry MES signalling information to the NCS in the network, specifically the response information required for a fixed-originated call, and for acknowledgments of MES Group ID down-loading messages to the NCS.

The basic modulation and coding techniques for the SCPC communication channels are filtered offset-quadrature phase-shift keying (O-QPSK), with convolutional coding at rate-3/4 FEC codes. Punctured coding is used for rate-3/4 FEC code. The baseline speech coder rate supported by SCPC channels operating in the 'Voice' mode is 6.4 kbit/s and INMARSAT currently, under an extensive codec evaluation process, is determining a suitable speech coding algorithm.

### 7.3 Speech Coding Algorithm

The speech coding scheme, described in chapter 6, was optimized to suit the land-mobile satellite requirements of the INMARSAT-M system. The modifications to the algorithm come in the form of introduction of Line Spectrum Frequencies (LSF) to represent the LPC parameters, the optimization of a Gaussian codebook (generation and search) and the inclusion of post filtering. The details of each of the modifications and their optimization to the INMARSAT-M channel are given in this section. The inclusion of the LSFs was critical in meeting the robustness to channel error criterion, and great care was taken in their optimization so as to preserve a low complexity order. The full details of the reduced complexity scheme for LSF transformation and its implementation are given in section 7.6.

#### 7.3.1 Algorithm Description of the Encoder (4.8 kbit/s)

The encoder structure is depicted in figure 6.3 with a difference from the original that LSF transformations of LPC parameters are used rather than the LARs. A non-overlapping 240-point biased Hamming window \((0.50, \ldots, 1.0, \ldots, 0.5065)\) is applied to the input frame of speech prior to the 10th order LPC analysis. Durbin's recursive method resolves the autocorrelation matrix, finding the 10 transversal filter coefficients. Quantization of the LPC parameters via the LAR transformation is not an optimum solution for high channel error rates (up to 1%) which is a non-avoidable
characteristic of satellite-land mobile channels. The LSF transformation provides better robustness to channel errors and was therefore used as a media for quantization.

In this coder, the 240 sample length of speech residual is split into 5 sub-blocks of 48 samples each. Each sub-block passes through the weighting filter, down-sampling, long-term prediction and codebook search cycle.

During the simulation stage, different combinations of codebook size (8 and 9-bits) and structure (centre-clipped and Gaussian) and decimation factors (3 and 4) were experimented upon and the performance of each combination, in terms of subjective quality, was measured. We found that the decimation factor of 3 and a 9-bit codebook (512x16) populated with Gaussian sequences produced the optimum speech quality [8,9].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Bits / Frame</th>
<th>Bit Rate (bits / s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPC via LSF</td>
<td>37</td>
<td>1,233</td>
</tr>
<tr>
<td>LTP Delay</td>
<td>17</td>
<td>567</td>
</tr>
<tr>
<td>LTP Gain</td>
<td>15</td>
<td>500</td>
</tr>
<tr>
<td>Codebook Index</td>
<td>45</td>
<td>1,500</td>
</tr>
<tr>
<td>Codebook Gain</td>
<td>20</td>
<td>667</td>
</tr>
<tr>
<td>Grid Position</td>
<td>10</td>
<td>333</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>144</strong></td>
<td><strong>4,800</strong></td>
</tr>
</tbody>
</table>

Table 7.1 Bit Allocation of Parameters of 4.8 kbit/s Speech Coder

Table 7.1 shows the bit allocation for the various parameters of the coder operating at 4.8 kbit/s. A total of 144 bits per speech frame are required to represent the six different sets of parameters, with LPC parameters needing more bits (37 bits = 1.233 kbit/s). The bit allocation of table 7.1 is an optimum choice for INMARSAT-M system. If the error rate of the transmission channel was less i.e. $10^{-4}$, then less bits could be used for the LPC parameter quantization in a differential encoding stage.

As table 7.2 suggests, the LPC parameters are the most sensitive to channel errors followed by the LTP delays, LTP gains, codebook gains, codebook indices and
### Parameter Performance at BER

<table>
<thead>
<tr>
<th>Parameter</th>
<th>10^{-3}</th>
<th>10^{-2}</th>
<th>2x10^{-2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSF</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>LTP Delay</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>LTP Gain</td>
<td>B</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Codebook Index</td>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Codebook Gain</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Grid Position</td>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

A = No Noticeable Degradation ; B = Noticeable Degradation  
C = Significant Degradation ; D = Just Intelligible

|**Table 7.2 Parameter Sensitivity Measure of 4.8 kbit/s Speech Coder [10]** |

The grid positions [10]. These results were later used as a reference for devising an optimum channel coding strategy for the 6.4 kbit/s coder (source and channel coding).

### 7.3.2 Algorithm Description of the Decoder (4.8 kbit/s)

The decoder has the same structure as that depicted in figure 6.4 with the addition of the post filtering and removal of deemphasis. The 5 reconstructed sub-blocks (codebook selected entries) after passing through an LTP synthesis and up-sampling cycle, are combined together to form the input excitation to the LPC synthesis filter. The quality of the recovered output speech is further enhanced through a pole-zero post filter. Parameters of the post filter are the weighted LPC coefficients. Once again, many simulations with different weighting factors were run and we found that weighting factors of 0.6 and 0.8, respectively, for inverse and synthesis filters of the post filter gave the optimum enhancement of the output speech.

### 7.4 Software Implementation of Encoder

The entire algorithm was simulated using high-level software written in C/Unix. Once the correct operation of the algorithm was verified, the software was translated (manually) into the instruction set of the DSP32C in the most optimized way possible.
Fortunately, the problem of frame synchronization that existed in the implementation of the VSAT coder was no longer a specification, but we were faced with a different problem, single-chip implementation of the coder. This proved to be a non-trivial problem and many different design strategies needed to be considered before arriving at the method proposed here. The DSP32C has only one input and output serial port which magnified the difficulty.

The design strategy of figure 7.3 was found to be the best solution, in which the speech samples are read in and out of the processor in the DMA mode and the 4.8 kbit/s stream of encoded data is read in and out using the two external interrupt lines and memory mapped locations.

![Diagram](image_url)

**Fig. 7.3 Single Chip Implementation of Speech Coder on DSP32C**

Figure 7.4 shows the execution cycle of the 4.8 kbit/s speech coder as implemented on a single chip. The algorithm is a block processing scheme and therefore at reset or power up after going through a set of initialization routines (such as A-law or μ-law setting and initializing the input and output pointers), the programme first decodes the received frame of data and then having gathered a full frame of speech samples, performs the encoding process. This means that, provided both the transmit and receive channels are operational, every 30 ms the programme acquires 240 speech samples and encodes them into 144 bits and then decodes the received bits into 240 reconstructed speech samples.
Figure 7.5 shows the flow chart of the encoding process. In this section, we will report on the implementation of the modified stages of the coder (as compared to the VSAT speech coder) and highlight the methods used to reduce the complexity and memory requirements of the real-time code.
Fig. 7.5 Execution Cycle of 4.8 kbit/s Speech Encoder
7.4.1 Data Acquisition and Preprocessing

Encoder Interrupt Service Routine

Input serial DMA and dual buffering procedure are used to collect input blocks of 240 A-law or μ-law (8-bit) coded speech samples. External interrupt line and a memory mapped location are used to output the encoded data at a rate of 4.8 kbit/s. This results in the generation of 600 interrupts per second (for encoding), since the data is transmitted out as bytes. Each time an interrupt is received, the 'Interrupt Service Routine' of figure 7.6 is executed. The routine first checks to see if a frame sync pulse is present and if so resets the input and output pointers otherwise it collects the received byte and increments the pointer.

![Diagram of Encoder Interrupt Service Routine]

Fig. 7.6 Execution Cycle of Encoder Interrupt Service Routine
Whenever an interrupt is generated, the DSP32C freezes all the states of the execution flow including the accumulators and flags, in shadow registers, except the CAU registers. This means that the programmer needs to store the contents of all the CAU registers that are used within the interrupt service routine and restore them at the end of the routine. This is an overhead and we therefore attempted to minimize the use of the CAU registers and took advantage of the DSP32C latencies to reduce the overhead.

**Preprocessing**

The input block of samples will pass through a preprocessing stage that includes:

- Conversion of A-law or μ-law coded samples to floating-point,
- Removal of DC-offset,
- Applying a non-overlapping Hamming window (biased) for LPC analysis.

The first two stages are implemented as explained in section 6.4.1. In order to reduce the complexity, it was decided to store the 240-point window in the memory (as floating-point values) and thus the windowed signal is simply found by a one to one multiplication of each sample value by the corresponding stored value.

**7.4.2 LPC Analysis**

The LPC analysis is performed in the following order:

- Computation of 10 transversal filter coefficients using Durbin's recursion method,
- LSF transformation of LPC parameters,
- Quantization and coding of LSFs,
- LPC inverse filtering (direct form structure).

**LPC parameters: Transformation, Quantization and Coding**

Different methods of transforming the LPC coefficients into LSF parameters exist, offering different levels of accuracy and complexity. Based on our simulation results, we decided on the Discrete Fourier Transform (DFT) method. A much reduced complexity scheme was devised and implemented as reported in section 7.6.

In order to quantize the LSF parameters as accurately as possible, a large training data, containing a wide range of male and female talkers, was used to compute an optimum bit allocation, and to help us to design individual scalar quantizers for each
<table>
<thead>
<tr>
<th>Order</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Bits</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>188</td>
<td>401</td>
<td>753</td>
<td>1041</td>
<td>1439</td>
<td>2006</td>
<td>2287</td>
<td>2775</td>
<td>3151</td>
<td></td>
</tr>
<tr>
<td>195</td>
<td>215</td>
<td>464</td>
<td>844</td>
<td>1175</td>
<td>1584</td>
<td>2116</td>
<td>2410</td>
<td>2909</td>
<td>3272</td>
<td></td>
</tr>
<tr>
<td>222</td>
<td>239</td>
<td>510</td>
<td>911</td>
<td>1275</td>
<td>1672</td>
<td>2177</td>
<td>2480</td>
<td>3000</td>
<td>3354</td>
<td></td>
</tr>
<tr>
<td>251</td>
<td>261</td>
<td>554</td>
<td>968</td>
<td>1341</td>
<td>1741</td>
<td>2223</td>
<td>2528</td>
<td>3086</td>
<td>3415</td>
<td></td>
</tr>
<tr>
<td>286</td>
<td>284</td>
<td>590</td>
<td>1017</td>
<td>1408</td>
<td>1804</td>
<td>2261</td>
<td>2574</td>
<td>3160</td>
<td>3474</td>
<td></td>
</tr>
<tr>
<td>329</td>
<td>311</td>
<td>626</td>
<td>1064</td>
<td>1466</td>
<td>1856</td>
<td>2298</td>
<td>2613</td>
<td>3235</td>
<td>3531</td>
<td></td>
</tr>
<tr>
<td>391</td>
<td>341</td>
<td>656</td>
<td>1110</td>
<td>1515</td>
<td>1906</td>
<td>2334</td>
<td>2651</td>
<td>3332</td>
<td>3581</td>
<td></td>
</tr>
<tr>
<td>Quantized Levels</td>
<td>483</td>
<td>375</td>
<td>687</td>
<td>1155</td>
<td>1560</td>
<td>1947</td>
<td>2365</td>
<td>2689</td>
<td>3453</td>
<td>3677</td>
</tr>
<tr>
<td>408</td>
<td>721</td>
<td>1202</td>
<td>1612</td>
<td>1989</td>
<td>2394</td>
<td>2724</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>445</td>
<td>754</td>
<td>1250</td>
<td>1659</td>
<td>2035</td>
<td>2428</td>
<td>2759</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>495</td>
<td>794</td>
<td>1295</td>
<td>1715</td>
<td>2081</td>
<td>2464</td>
<td>2791</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>552</td>
<td>849</td>
<td>1350</td>
<td>1773</td>
<td>2135</td>
<td>2502</td>
<td>2830</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>615</td>
<td>924</td>
<td>1410</td>
<td>1834</td>
<td>2194</td>
<td>2551</td>
<td>2880</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>684</td>
<td>1029</td>
<td>1499</td>
<td>1906</td>
<td>2267</td>
<td>2625</td>
<td>2957</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>769</td>
<td>1166</td>
<td>1616</td>
<td>2009</td>
<td>2370</td>
<td>2729</td>
<td>3050</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>935</td>
<td>1392</td>
<td>1809</td>
<td>2167</td>
<td>2477</td>
<td>2852</td>
<td>3197</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.3: Bit Allocation and Corresponding Quantized Levels of LSF Parameters

order of the LSF coefficient vector. Table 7.3 shows the bit allocation and the corresponding quantized levels of each order. We found that the bit allocation pattern \{3, 4, 4, 4, 4, 4, 4, 3, 3\} gave the optimum performance when measured in terms of the subjective quality of the output speech (of the fully quantized coder at 4.8 kbit/s).

The quantization of the LSF parameters required extra care and consideration as compared to LAR quantization. During the quantization, if the ascending order of the LSF array is disturbed causing instabilities in both the inverse transformation filter and the LPC inverse and synthesis filtering, consequently resulting in undesirable high-pitched sounds at the coder output. Figure 7.7 illustrates the way in which the quantization and coding of the LSF parameters are achieved. Each computed LSF is compared against the stored quantized levels (of table 7.3) and the level which produces the minimum absolute difference is selected as the quantized value. This value is then compared with the quantized value of the previous order and if greater, is pushed up to the next quantizer level. The corresponding code value also changes.
LSF Quantization

\[ m = 0 \]

\[ i = 0 \]
Maxcode_\[m\] = \[2^{\text{(no. of bits)\,-\,1}}\]
MinDif = 5000
\[ n = -1 \]

\[ d = |\text{LSF}[m] - \text{Level}[i]| \]
\[ n = n + 1 \]

Is \[ d < \text{MinDif} \]?

Yes
MinDif = d
QLSF[m] = Level[i]
Code[i] = n

No
Is \[ n \geq \text{Maxcode}[m] \]?

Yes

Is \[ m = 0 \]?

No
Code[i] = Code[i] + 1
\[ i = i + 1 \]
QLSF[m] = Level[i]

Is \[ \text{QLSF}[m] > \text{QLSF}[m-1] \]?

No
Is \[ \text{Code}[i] = \text{Maxcode}[m] \]?

Yes
\[ m = m + 1 \]
Store Code Value and Quantized Level

Is \[ m < \text{Order} \]?

No

END

Fig. 7.7 Quantization and Ordering of LSF Parameters
LPC Inverse Filtering

The quantized LSF parameters are transformed back into transversal filter coefficients as documented in section 7.6. The LPC residual signal, \( r(i) \), is computed by implementing the LPC inverse filter in the direct form:

\[
r(i) = s(i) - \sum_{n=1}^{\text{Order}} a_i \cdot s(i-n) ; \quad i = 0, ..., 239 \tag{7.1}
\]

where \( s(i) \) is the block of input speech and \( a_i \) are the quantized LPC coefficients. The direct form implementation of the filter is slightly less complex than the lattice structure and since the stability of the filter coefficients has already been established, via the LSF order check, it is not necessary to use the lattice structure implementation.

7.4.3 LTP Analysis

The LTP analysis is performed on the down-sampled LPC residual. The baseband residual signal is extracted in an identical manner to that of the VSAT speech coder. A decimation factor of 3 is chosen corresponding to 5 sub-segments of 16 sample length each and 3 possible grid positions.

<table>
<thead>
<tr>
<th>Input Threshold</th>
<th>Output</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \leq 0.10 )</td>
<td>0.05</td>
<td>0 = 000</td>
</tr>
<tr>
<td>( \leq 0.40 )</td>
<td>0.25</td>
<td>1 = 001</td>
</tr>
<tr>
<td>( \leq 0.60 )</td>
<td>0.45</td>
<td>2 = 010</td>
</tr>
<tr>
<td>( \leq 0.80 )</td>
<td>0.65</td>
<td>3 = 011</td>
</tr>
<tr>
<td>( \leq 1.00 )</td>
<td>0.85</td>
<td>4 = 100</td>
</tr>
<tr>
<td>( \leq 1.20 )</td>
<td>1.05</td>
<td>5 = 101</td>
</tr>
<tr>
<td>( \leq 1.40 )</td>
<td>1.25</td>
<td>6 = 110</td>
</tr>
<tr>
<td>&gt; 1.40</td>
<td>1.45</td>
<td>7 = 111</td>
</tr>
</tbody>
</table>

Table 7.4 Input Thresholds and Output Levels of LTP Gain Quantizer

An 8-p. windowing scheme is again employed for the LTP lag computations of the 4 sub-segments (3 bits each) and the first lag value is coded with 5 bits (32 decimated
lag locations). The LTP gain parameter is uniformly scalar quantized according to table 7.4. Each parameter is coded with 3 bits. The computation of the threshold levels and the output levels are based on a large training data (during the simulation phase) and those levels are selected that produced the least noticeable degradation in the output speech.

### 7.4.4 Sequential Codebook Search

The theory of the codebook search for CELP-BB coding and its mathematical derivations are documented in section 6.4.5. We realized that the direct implementation of equations 6.17 and 6.18 would be expensive in terms of memory usage. We therefore decided on a 'Sequential Codebook Search' where a much smaller number of sequences need to be stored. In a sequential search, with codebook sequences of length $q$, and a shift of $s$, the codebook index pointer is incremented by $s$ for each codebook entry search as compared to an increment of $q$ in the case of normal search procedures. This means that a 9-bit Gaussian codebook of 16-sample sequence, with non-sequential search, occupies a memory of size (floating-point storage):

$$\text{Book Size} = 2^{\text{no. of bits}} \cdot \text{Sequence Size} \quad (7.2)$$

\begin{align*}
\text{no. of bits} &= 9 \\
\text{Sequence Size} &= 16
\end{align*}

$$\text{Book Size} = 8,192 \equiv 32,768 \text{ Bytes.}$$

We experimented with codebook shifts of 1 to 4 and saw no noticeable discrepancy in the quality of the output speech for any of the shifts as compared with the normal search. It was therefore decided to design a codebook with a minimum shift i.e. $s=1$ which only requires a memory storage:

$$\text{Book Size} = [2^{\text{no. of bits}} \cdot \text{shift}] + [\text{Sequence Size} - \text{shift}] \quad (7.3)$$

$$\text{Book Size} = 1,038 \equiv 4,152 \text{ Bytes.}$$

In a sequential codebook search we can no longer benefit from the pre-normalization of the codebook sequences (as suggested by equations 6.17 and 6.18) and equations 6.15 and 6.16 have to be implemented. In developing the real-time software, it is advantageous to turn all the divisions into multiplications (by the inverse of the
denominator) whenever possible. This is due to the fact that a multiplication is normally a one-instruction routine whereas division would usually require the execution of many instructions. In order not to cause a substantial overhead on the codebook search complexity, the inverse of the codebook sequence energies (denominator of equation 6.15) are also stored in the memory thus requiring no on-line processing and calls to division subroutine. This means a total memory requirement of:

\[
\text{Total Book Size} = 1,038 + 512 = 1,550 \approx 6,200 \text{ Bytes}
\]

which is still a fraction of the normal search (a ratio of 5.3 to 1). Therefore, although, we have incurred a small penalty in terms of increased complexity, we have managed a memory saving of 26,568 bytes.

**Parameter Quantization and Coding**

The 5 codebook indices need no quantization. The codebook gain parameters are non-uniformly quantized with 4 bits each (1 bit for sign). The quantization scheme of the VSAT coder is employed with initial step size and threshold set to 10 and the multiplication factor, M, is set to 1.2.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Code Positive Values</th>
<th>Code Negative Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 10.0</td>
<td>5.00</td>
<td>0 = 0000</td>
<td>8 = 1000</td>
</tr>
<tr>
<td>&lt; 22.0</td>
<td>16.0</td>
<td>1 = 0001</td>
<td>9 = 1001</td>
</tr>
<tr>
<td>&lt; 36.4</td>
<td>29.2</td>
<td>2 = 0010</td>
<td>10 = 1010</td>
</tr>
<tr>
<td>&lt; 53.68</td>
<td>45.0</td>
<td>3 = 0011</td>
<td>11 = 1011</td>
</tr>
<tr>
<td>&lt; 74.42</td>
<td>64.1</td>
<td>4 = 0100</td>
<td>12 = 1100</td>
</tr>
<tr>
<td>&lt; 99.30</td>
<td>86.9</td>
<td>5 = 0101</td>
<td>13 = 1101</td>
</tr>
<tr>
<td>&lt; 129.2</td>
<td>114.2</td>
<td>6 = 0110</td>
<td>14 = 1110</td>
</tr>
<tr>
<td>&gt; 129.2</td>
<td>147.1</td>
<td>7 = 0111</td>
<td>15 = 1111</td>
</tr>
</tbody>
</table>

*Table 7.5 Quantization and Coding of Codebook Gain*

The quantizer covers an input range of up to 129.2. Table 7.5 shows the threshold and output levels of the quantizer and their corresponding code values.
7.4.5 Data Output and Post processing

The 4.8 kbps encoder generates six sets of parameters resulting in 35 code values. The Bit Packer routine of the VSAT speech coder is modified (the bit stuffing is removed) to pack the coded values into a continuous string of 144 bits. The packed bits are then transmitted as bytes of data, through the encoder interrupt service routine (as explained in section 7.4.1).

7.5 Software Implementation of Decoder

Figure 7.8 shows the execution flow of the decoding process. Every 30 ms, the decoder reads in, through the 'Decoder Interrupt Service Routine', the encoded 144 bits and the corresponding status flags (through the parallel port). If the VAD flag suggests a silent frame, the received bits are ignored and the decoder output is muted. Otherwise, if the LOST flag indicates a corrupted frame, the received block of data is once again discarded and the parameters of the previous frame are used for reconstruction of this frame. The coder is able to tolerate burst errors of up to 90 ms duration (i.e. 3 frames). If, however, a GOOD frame is received, the received bits are unpacked and decoded. The long-term correlations are put back into appropriate codebook sequences and the full-band signal is reconstructed. This is repeated 5 times for each LPC frame. The short-term correlations are then placed into the signal and post filtering is performed to enhance the speech quality. The post processing stage simply converts the floating-point values into their corresponding A-law or µ-law codes, ready to be transferred to the PCM codec by the DMA routine.

The LPC and LTP synthesis procedures are identical to the VSAT speech coder and a description of their implementation will not be repeated here. In this section, only the implementation of the decoder interrupt service routine, Frame Reconstruction and Post Filtering procedures are considered.

7.5.1 Data Acquisition

The Decoder Interrupt Service Routine is responsible for the transfer of the coded data (the 4.8 kbit/s data stream) to the decoder processing block. The data is received in bytes, resulting in 18 interrupts to be generated every 30 ms.
Fig. 7.8 Execution Cycle of 4.8 Speech Decoder
Figure 7.9 shows the execution flow of the interrupt routine for the decoder. Every time an interrupt is received, the DSP32C freezes its present execution flow before entering the interrupt routine. If the decoder frame sync flag indicates the start of a new speech frame, the input and output pointers are reset to the beginning of new buffers. Otherwise, the transmitted byte of data is collected and the pointer is incremented.

### 7.5.2 Frame Reconstruction

The speech transmission channel, in a mobile environment, is subject to both random and burst errors. The burst errors have a severe effect on the quality of the
output speech if not dealt with. One method of rectifying the problem would be to employ high capacity FEC on interleaved frames of speech. This will naturally increase both the coder bit rate and its delay. The other method, which is very popular with low bit rate speech coders, is to use a lower order FEC combined with an efficient scheme of frame reconstruction. The channel decoder examines a set of sensitive bits and if in error will indicate this as a 'Bad' or 'Lost' frame to the speech decoder. The speech decoder will then replace the lost information with information that minimizes the subjective discomfort at the output. For lost frame indications, we used a set of data that was recorded in a mobile receiver, around two busy areas of central London, from the INMARSAT satellite at L-band. Each lost frame decision had been taken by matching the average received signal power over 10 ms to a given threshold.

By conducting a number of informal listening tests, the following frame reconstruction strategy proved to be the optimum:

(i) Replace all 5 LTP lags with the very last LTP lag value (5th sub-segment) of the previous frame.

(ii) LTP gain values set to a small level ( = 0.25, the second output level of the LTP gain quantizer).

(iii) Codebook indices remain the same as the previous frame.

(iv) Codebook gain values set to a small level ( = 5.0, 1st output level of the codebook gain quantizer) but their signs stay the same as those of the previous frame.

(v) LSF parameters remain the same as the previous frame.

(vi) Grid positions remain the same as the previous frame.

This causes a gradual decay of the coder output, avoiding an abrupt break or unpleasant 'clicks' in the output speech for long durations of bursty channel conditions, while successfully filling the short gaps (up to 90 ms).
7.5.3 Post Filtering

In enhancing the output quality of low bit-rate speech coders, advantage can be made of the human auditory system where the ear does not perceive noise only on the basis of its rms value but takes into account the relation between the quantization noise spectrum and the speech spectrum. There are two different approaches to reducing the perceptual effect of the quantization noise; noise spectral shaping and noise suppression.

![Diagram](image)

**Fig. 7.10 Operation of an Idealized Post Filter [11]**

The philosophy of the post filtering technique (noise spectral shaping) can be best explained using the simple illustration in figure 7.10. Part (a) of the figure shows a signal spectrum with two narrow-band components in the frequency regions, $W_1$ and $W_2$, and a flat noise spectrum that is 15 dB below the first signal component but 5 dB above the second signal component. Part (b) represents the spectra of the post filtered signal and noise, when the post filter transfer function is identical to the signal spectrum.
in (a). Although, the resulting SNRs in regions W₁ and W₂ are the same as before post filtering, the noise in the rest of the frequency range is much lower than the signal levels. This post filtering operation perceptually enhances the signal.

$$F(z) = \frac{1 + B(z)}{1 + A(z)}$$

Fig. 7.11 Block Diagram of Post Filtering Function

An adaptive 10th order post filter, F(z), was implemented. The filter coefficients, aᵢ, are the received LPC parameters:

$$A(z) = \sum_{i=1}^{10} (a_i \alpha^i). z^{-i} ; \quad 0 \leq \alpha \leq 1 \quad (7.5)$$

and

$$B(z) = \sum_{i=1}^{10} (a_i \beta^i). z^{-i} ; \quad 0 \leq \beta \leq 1 \quad (7.6)$$

The selection of the two weighting coefficients, α and β, was based on a number of informal listening tests and we concluded that with α = 0.6 and β = 0.8, an optimum level of enhancement is reached.
It is also necessary to ensure that the signal energy of the output speech remains at the same level as the energy of the input speech signal to the post filter. Therefore, an 'equalization factor', \( e \), is derived and each sample of the block, \( S_O(i) \), is scaled up by this factor:

\[
\varepsilon = \left( \frac{\sum_{j=0}^{239} S_i(j)^2}{\sum_{j=0}^{239} S_O(j)^2} \right)^{1/2}
\]  

(7.7)

This will provide a 'Block Equalization' of the signal and if a 'Sample-by-Sample Equalization' [13] is performed, at the expense of extra computational complexity, a further enhancement is achieved.

7.5.4 Data Output and Postprocessing

The reconstructed 240 speech samples, in floating-point format, are transformed into their corresponding companded code values and transferred out to the PCM codec. The transfer of the coded speech samples to the PCM codec makes use of the serial DMA output and dual buffering system.

7.6 Line Spectrum Frequency Transformation

Efficient quantization of the LPC parameters and the 'safe' delivery of these parameters to the decoder, has always been the subject of much research in the past and will continue to be so in the future. Several types of representation (PARCOR, LAR, Sinusoidal Transform) have been proposed. Most recently, Line Spectrum Frequency (LSF), also known as Line Spectrum Pair (LSP), coefficients have been the subject of extensive research and application, especially in the field of low bit-rate mobile voice communications. Although, LSF coefficients were initially devised as a means of bit rate reduction [14, 15], nowadays, the general belief is that the use of optimally quantized LSP coefficients offers little or no advantage in comparison to the use of optimally
quantized PARCOR coefficients. However, the inherent error protection properties of directly quantized LSF coefficients can be used to devise a simple LPC coefficient transmission scheme, that is more robust to channel errors than conventionally error protected transmission schemes [16]. Indeed, the initial claim is still valid if one used the quantization schemes that are based on exploiting the high degree of correlation that exists between successive LSF vectors [17, 18].

The LSFs have a built-in stability criterion which requires the elements in a single vector to occur in an ascending order. Violations of this criterion for any vector indicates the presence of an error. Using appropriate error location techniques, the corrupted received elements can be identified and adjusted accordingly and possibly corrected [19].

In this section, a brief description of a reduced complexity LSF transformation [20] and its implementation are given. A simple method in which the erroneous LSF parameters are detected and adjusted is also reported.

7.6.1 LPC-To-LSF Transformation

Once it is assumed that the PARCOR filter is stable and the filter order is even, one of four methods; (i) Complex Root method, (ii) Real Root method, (iii) Ratio Filter method and (iv) DFT method, can be used to derive the LSF coefficients. The DFT method was selected for reasons of its low complexity.

An all-pole digital filter for speech synthesis, \( H(z) \), can be derived from linear predictive analysis as:

\[
H(z) = \frac{1}{A_p(z)} \tag{7.8}
\]

where

\[
A_p(z) = 1 + \sum_{k=1}^{p} \alpha_k \cdot z^{-k} \tag{7.9}
\]

The LPC filter can be written as PARCOR equivalent [20, 21]:
\[ A_{p-1}(z) = A_p(z) + k_p \cdot B_{p-1}(z) \] (7.10)

\[ B_p(z) = z^{-1} [B_{p-1}(z) - k_p \cdot A_{p-1}(z)] \] (7.11)

where \( A_0(z) = 1 \) and \( B_0(z) = z^{-1} \), and

\[ B_p(z) = z^{-(p+1)} \cdot A_p(z^{-1}) \] (7.12)

The polynomial \( A(z) \) is decomposed into anti-symmetric and symmetric polynomials \( P(z) \) and \( Q(z) \) (for \( k_{p+1} = 1 \) and \( k_{p+1} = -1 \), respectively):

\[ P(z) = A_p(z) - B_p(z) \] (7.13)

and

\[ Q(z) = A_p(z) + B_p(z) \] (7.14)

If an even order is assumed, \( P(z) \) and \( Q(z) \) are factorized as [21]:

\[ P_p(z) = (1 - z) \prod_{i=2}^{\text{even}} (1 - 2 \cos(\omega_i) \cdot z + z^2) \] (7.15)

and

\[ Q_p(z) = (1 + z) \prod_{i=1}^{\text{odd}} (1 - 2 \cos(\omega_i) \cdot z + z^2) \] (7.16)

where the coefficients \( \{\omega_i\} \) are referred to as the LSP parameters.

By performing a DFT on the coefficient sequence, \( A_k \) and \( B_k \), \( \{\omega_i\} \) can be solved as the zero-valued frequencies of a power spectrum. If the spectrum was to be obtained directly, it would involve a considerable amount of computation. Fortunately, a number of computation reductions are possible. The aim is to find the partial minima of the response, thus the absolute values of the response are not critical, but only the location of the minima are of value. A typical plot is shown in figure 7.12, where the partial minima of the spectrum are clearly visible.
Fig. 7.12 A Typical Zero Frequency Plot of $P(z)$ and $Q(z)$
As the input sequence, $A_k$ and $B_k$, are real we can move them from the start to the middle of the input matrix with zeros elsewhere. This produces an even spectrum, which means that only $f_s/2$ terms need to be computed. Also, the spectrum is real, thus only the cosine terms in the kernel require computing. The cosine terms are fixed for a particular transform size, therefore they can be pre-computed and stored as a look-up table.

Once the spectrum is found, the partial minima need to be located and this involves computationally expensive comparisons. As the LSFs are naturally ordered i.e. the frequencies alternate between $Q(z)$ and $P(z)$, they can be located in an efficient manner. Once the first LSF location is found (on the $Q(z)$ spectrum), the search pointer is moved to the $P(z)$ spectrum for locating the second LSF parameter and then back to $Q(z)$ spectrum. This alternation is repeated until all LSFs are found. Thus in total only one pass of the frequency range is made instead of two.

In our implementation, a 512-point DFT is performed giving a resolution of 8 Hz per point (approximately). Although, this is adequate for speech, it can cause problems for tone transmission, where the frequencies become closer than 8 Hz. In order to overcome this problem a higher order DFT is needed, but the limitations on the available processing power prevented us from making this modification.

### 7.6.2 LSF-To-LPC Transformation

Two very similar methods of inverse transforming the LSF coefficients back into the LPC exist; (i) Direct Expansion method, and (ii) LPC synthesis Filter method. The LPC synthesis filter method is usually preferred for its ease of visualization.

In the LPC synthesis filter, a digital filter, $H(z)$, is constructed based on the LSF parameters. The filter transfer function can be represented in the form:

$$H(z) = \frac{1}{1 + (A_p(z) - 1)}$$  \hspace{1cm} (7.17)

or

$$H(z) = \frac{1}{1 + 1/2 \left[ (P_{p+1}(z) - 1) + (Q_{p+1}(z) - 1) \right]}$$  \hspace{1cm} (7.18)
It can be shown that, when \( p \) is even [21]:

\[
A_p(z) - 1 = \frac{z}{2} \left[ \sum_{i=2}^{p} (C_i + z) \prod_{j=0}^{i-2} (1 + C_j z + z^2) - \prod_{j=2}^{p} (1 + C_j z + z^2) \right] + \sum_{i=1}^{p-1} (C_i + z) \prod_{j=1}^{i-2} (1 + C_j z + z^2) + \prod_{j=1}^{p-1} (1 + C_j z + z^2) \]

(7.19)

where \( C_i = -2 \cos(\omega_i) \), \( C_0 = C_1 = -z \).

The structure of a 10th order filter is shown in figure 7.13. The LPC coefficients are simply the impulse response of this filter. The computational complexity of the inverse transformation is much less than the forward transformation and in our implementation the following pipelined instruction of the DSP32C is used to speed up the execution time and also to reduce the memory usage (scratch pad memory):

\[
an_N = an_N + (*rM++ = an_N) * *rM++
\]

(7.20)

where \( an_N \) is one of the four accumulators and \( rM \) is a register used as a memory pointer.

### 7.6.3 LSF Error Detection and Correction

As was mentioned before, the ascending order of the LSF vector can be used as a built-in error detection strategy. At the decoder, the received LSF vector is examined and if the ascending order is disturbed, a transmission error is assumed. It is very important to correct or at least adjust the erroneous LSF parameters in order to maintain a stable LPC synthesis filtering. There is a high degree of correlation between the successive LSF vectors and this has been the basis of many successful schemes of LSF error correction [10,19].

In this implementation, we decided on a very simple strategy in which if there were only one single erroneous LSF parameter, it is replaced with the LSF parameter (of the same order) of the previous vector. Otherwise, the entire set is replaced with the LSF parameters of the previous speech frame. This scheme operates quite well and avoids the
generation of high-pitched sounds, especially during high random errors and burst errors. The overhead computational complexity of this scheme is negligible.

### 7.7 Voice Activity Detection

A simple Voice Activity Detection (VAD) system was developed and implemented [22]. Although, the proposed scheme operates adequately its performance can be improved further, especially in the presence of extreme background noise levels, at the expense of increased complexity. The VAD decision making is simply based on two parameters; (i) the energy of the input signal, and (ii) the LSF parameter variations. There exists a direct relationship between the input signal energy and the energy content of the LTP register.

<table>
<thead>
<tr>
<th>LSF Order</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Limit</td>
<td>-</td>
<td>640</td>
<td>960</td>
<td>1360</td>
<td>1720</td>
<td>2060</td>
<td>2440</td>
<td>2780</td>
<td>3140</td>
<td>3480</td>
</tr>
<tr>
<td>High Limit</td>
<td>-</td>
<td>820</td>
<td>1220</td>
<td>1580</td>
<td>1940</td>
<td>2260</td>
<td>2660</td>
<td>3040</td>
<td>3380</td>
<td>3740</td>
</tr>
</tbody>
</table>

Table 7.6: Low and High Limits of the LSF Parameters During Non-Speech Periods

Figure 7.14 shows the flow chart of the VAD system. In the absence of background noise, during the silence periods, the input signal energy is very low. During the simulation phase, limiting the dynamic range of the input signal to those of A-law and µ-law coding, two threshold levels, \(TH_1\) and \(TH_2\), were computed. The energy of the LTP register, \(E_c\), is compared against the first threshold level, \(TH_1\), and if smaller, it was decided that the frame is silent. Otherwise, the frame is either speech or carries background noise. The energy of the most recent segment of the LTP register (16 sample length), \(E_{LS}\), is used to give a closer estimate of the input signal energy. LSF parameters have well defined values when the input signal is silence or random noise. This means that we can use this information to distinguish between speech (correlated signal) and random noise (uncorrelated signal). When the input signal is equivalent to an uncorrelated signal, each value of the LSF vector moves to within two (high and low) limits which are very close to their mean values. During our simulations, we noticed that when speech was present, at least 8 of the LSFs were outside the high and
Voice Activity Detection

Comput Energy Content of LTP Register, $E_c$

Is $E_c < TH1$ ?

Yes

No

Comput Energy of Most Recent Segment of LTP Register, $E_{LS}$

Is $E_{LS} > TH2$ ?

Yes

No

Is $E_c > TH2$ ?

Yes

No

Compare LSF Values Against Given Range

Are More than 7 in Range ?

Yes

No

Count = 0

Count += 1

Is Count = 3 ?

Yes

Speech Frame Detected

Silence Frame Detected

Fig. 7.14 Flow Chart of Low Complexity Voice Activity Detection Scheme
low limits. Therefore, if 8 of the 10 LSFs are outside the precomputed ranges (see table 7.6) then the frame is declared as speech else it is classified as a non-carrying speech frame (may contain background noise). A hangover period of 3 frames i.e 90 ms is used to minimize clipping of the speech segments.

A major limitation of this scheme is the preset of the LSF limits for silence classification. In some cases, correlations in the background noise may fool the VAD system. In order to enhance the VAD performance, both the threshold levels and the LSF limits need to be computed adaptively. This will result in the VAD system operating more efficiently under correlated noise as well as random noise.

7.8 Computational Complexity and Memory Usage

Tables 7.7 and 7.8 show, respectively, the computational complexity of the encoder and decoder in terms of required clock cycles (including conflict wait states) and also in number of operations (excluding wait states). The encoding process, on average, takes 867,384 clock cycles per frame to execute, corresponding to a processing period of 21.7 ms with the DSP32C operating at a rate of 40 MHz. The decoding process takes 144,626 clock cycles per frame, corresponding to a processing time of 3.6 ms. This suggests a complexity ratio of 6 to 1 between the encoding and the decoding processes. Once again, the major complexity of the encoding process is attributed to the analysis-by-synthesis procedure (in particular the codebook search). The most computationally expensive stage of the decoder is the post filtering where a 10th order pole-zero filter is implemented. In total, as table 7.9 indicates, the encoding and decoding processes take 1,012,010 clock cycles of the 40 MHz master clock corresponding to 25.3 ms processing time i.e. 84.3% of allowed time (30 ms). A tolerance of ±5% should be considered for the variations in the execution flow of the programme.

The memory requirements of the coder are shown in table 7.10. A total of 38,912 bytes of memory are available (32 kbytes of external and 6 kbytes of internal RAM memory). The programme occupies 15% of this memory, equivalent of 1,461 instructions, residing in the external memory (bank 0 of the DSP32C). The data storage and reservations require 25,812 bytes of memory (66% of the total available memory). The large amount of memory requirements for data storage is due to the dual input and output buffering system of the encoder and decoder, the codebook storage and the simplified scheme of LSF transformation. The transformation is based on a reduced
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>No. of Clock Cycles</th>
<th>No. of Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt Service Routine</td>
<td>1,233 (=18 x 68)</td>
<td>265 (= 18 x 14)</td>
</tr>
<tr>
<td>Pre-Processing</td>
<td>8,714</td>
<td>1,699</td>
</tr>
<tr>
<td>ACF + Durbin</td>
<td>31,362</td>
<td>6,413</td>
</tr>
<tr>
<td>LPC-to-LSF</td>
<td>62,907</td>
<td>13,387</td>
</tr>
<tr>
<td>LSF Quantization</td>
<td>9,370</td>
<td>2,260</td>
</tr>
<tr>
<td>LSF-to-LPC</td>
<td>4,839</td>
<td>1,043</td>
</tr>
<tr>
<td>Inverse Filter (Direct Form)</td>
<td>36,995</td>
<td>6,729</td>
</tr>
<tr>
<td>Down-Sampling</td>
<td>23,136 x 5</td>
<td>5,477 x 5</td>
</tr>
<tr>
<td>LTP Analysis</td>
<td>7,606 x 5</td>
<td>1,575 x 5</td>
</tr>
<tr>
<td>Sequential Codebook Search</td>
<td>109,723 x 5</td>
<td>23,077 x 5</td>
</tr>
<tr>
<td>Optimum Gain Quantization</td>
<td>551 x 5</td>
<td>135 x 5</td>
</tr>
<tr>
<td>Pattern Reconstruction</td>
<td>285 x 5</td>
<td>54 x 5</td>
</tr>
<tr>
<td>Bit Packing</td>
<td>4,142</td>
<td>1,033</td>
</tr>
<tr>
<td>Voice Activity Detection</td>
<td>1,317</td>
<td>285</td>
</tr>
<tr>
<td>Total</td>
<td>867,384</td>
<td>184,704</td>
</tr>
</tbody>
</table>

Table 7.7 Computational Complexity of 4.8 kbit/s Encoder
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>No. of Clock Cycles</th>
<th>No. of Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt Service Routine</td>
<td>1,215</td>
<td>265</td>
</tr>
<tr>
<td>Bit Unpacking</td>
<td>4,700</td>
<td>1,140</td>
</tr>
<tr>
<td>Parameter Decoding</td>
<td>1,537</td>
<td>368</td>
</tr>
<tr>
<td>Pattern Reconstruction</td>
<td>283 x 5</td>
<td>54 x 5</td>
</tr>
<tr>
<td>LTP Synthesis</td>
<td>789 x 5</td>
<td>150 x 5</td>
</tr>
<tr>
<td>Up-Sampling</td>
<td>815 x 5</td>
<td>175 x 5</td>
</tr>
<tr>
<td>LSF-to-LPC</td>
<td>4,839</td>
<td>1,043</td>
</tr>
<tr>
<td>LSF Error Correction</td>
<td>482</td>
<td>111</td>
</tr>
<tr>
<td>LPC Synthesis Filter</td>
<td>36,764</td>
<td>6,731</td>
</tr>
<tr>
<td>Post Filtering</td>
<td>82,267</td>
<td>15,076</td>
</tr>
<tr>
<td>Post-Processing</td>
<td>3,387</td>
<td>727</td>
</tr>
<tr>
<td>Total</td>
<td>144,626</td>
<td>27,356</td>
</tr>
</tbody>
</table>

Table 7.8 Computational Complexity of 4.8 kbit/s Decoder
<table>
<thead>
<tr>
<th>Process</th>
<th>No. of Clock Cycles</th>
<th>No. of Operations</th>
<th>Processing Time*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder</td>
<td>867,384</td>
<td>188,493</td>
<td>21.7</td>
</tr>
<tr>
<td>Decoder</td>
<td>144,626</td>
<td>27,356</td>
<td>3.6</td>
</tr>
<tr>
<td>Total</td>
<td>1,012,010</td>
<td>215,849</td>
<td>25.3</td>
</tr>
</tbody>
</table>

* @ 40 MHz Clock Rate

Table 7.9 Computational Complexity of 4.8 kbit/s Speech Coder

<table>
<thead>
<tr>
<th>Section</th>
<th>Memory Used (Bytes)</th>
<th>Percentage of Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programme</td>
<td>5,844</td>
<td>15 %</td>
</tr>
<tr>
<td>Data</td>
<td>25,812</td>
<td>66 %</td>
</tr>
<tr>
<td>TOTAL</td>
<td>31,656</td>
<td>81 %</td>
</tr>
</tbody>
</table>

Table 7.10 Memory Requirements of 4.8 kbit/s Coder
complexity scheme (see section 7.6.2) which requires the storage of a large amount of pre-calculated cosine terms (in floating-point format). This in fact accounts for most of the memory usage (= 12 kbytes). In total, 31,656 bytes of memory are used with 19% remaining.

7.9 Speech Coder Hardware

The hardware is implemented on a 4-layer printed circuit board (PCB). The presence of high speed clocks on the board required a ground layer which divides into sections; a digital and an analogue layer. The board is of a single Eurocard size, 160 mm by 100 mm. The heart of the board is the AT&T WE-DSP32C Digital Signal Processor (DSP) which is responsible for both encoding and decoding.

Figure 7.15 illustrates the speech coder hardware. The hardware is a standalone single-board computer and as such requires no setting up if the on-board codecs are to be used for analogue speech input and output.

In the following sections the features of the main stages of the hardware are briefly considered.

7.9.1 The Processor Board

The processor board is a single Eurocard which provides a very high level of integration by using state of the art technology. As figure 7.16 shows, it uses the AT&T DSP32C, the high performance static RAM, an advanced microcontroller, and a reprogrammable gate array. It is capable of producing a full-duplex link, by taking analogue speech, or A-law/u-law coded PCM speech, at its input and produces a low bit rate (4.8 Kbps), TTL compatible data output locked to an externally provided frame sync. Similarly, it will accept the low bit rate data, locked to an externally provided frame sync, and produces at its output either analogue speech or A-law/u-law PCM coded speech. The DSP32C has only a single serial port. The single-chip implementation of a full-duplex link requires two serial ports if the coded data is to be sent and received serially to and from the channel. In our implementation, the serial input and output ports are used for input and outputing the speech samples in DMA mode, and the two external interrupt lines are used for input and outputing the coded data.
Fig. 7.15 4.8 kbit/s Speech Coder Hardware
The board has many features which offer the user total flexibility. The board can be configured for either an internal or external PCM codec. The coded data can be received or sent at full-rate or split into In-Phase and Quadrature (IQ) data streams for direct interface to a modem. Two digital phase locked loops (DPLL) are provided for transmitting and receiving data. The locking range of the DPLL can be digitally programmed (jumper select). The analogue output is mixed with a small proportion of the input signal to produce a sidetone.
A watchdog unit on the board monitors the correct operation of each stage of the board and an alarm is activated under the following conditions; (i) failure of the microcontroller to power up, (ii) failure of the DSP32C and microcontroller interface, (iii) failure of the gate array to power up or initialize.

The encoding and decoding processes are made totally independent of each other and thus the loss of either of receive clock, or receive frame syncs will halt the decoding process only and similarly the loss of either transmit clock, or transmit frame syncs will only halt the encoding process.

Four input flags and seven output flags can be passed on to the DSP32C. These flags can be used as voice activity detection (VAD) flags, lost frame indication or speech/data detection. To achieve a peak floating performance of 20 MFLOPS, the DSP32C is run below its specified maximum clock of 50 MHz, at 40 MHz. The programme and the data is downloaded from the microcontroller to the static RAM at power up or reset.

7.9.2 Audio Input and Output Unit

Figure 7.17 illustrates the implementation of the audio input and output stages of the hardware. Two AT&T T7500PC PCM codecs are used for analogue-to-digital and digital-to-analogue conversions. The two codecs can be configured for either A-law or µ-law through jumper selects.

The analogue input to the codec is buffered by an LF353 operational amplifier (op-amp). The op-amp stage includes a 3.5 KHz filter. The gain of the input stage is set by a 10K Ohm potentiometer and gives an amplification range from 1 to 6. The input stage has an impedance of 100K Ohm and an input voltage range of ± 2.5 volts.

The analogue output from the codec is mixed with a small portion of the input signal by a 100K Ohm potentiometer to produce a side-tone. The combined output and sidetone signal is conditioned and filtered before amplification by the audio amplifier. The output level is set by a 20K Ohms potentiometer. The output stage has an input range of ± 2.5 volts and a driving power of 250 mW into an 8 Ohm load.
Fig. 7.17 Audio Input and Output Stages of 4.8 kbit/s Speech Codec Hardware
7.9.3 Digital Input and Output Unit

Output

The encoded 4.8 kbit/s data is output through a parallel to serial (P/S) converter in the gate array. The parallel to serial converter is double-buffered and therefore can store one byte of data whilst another is in transmission. External interrupt 2 is used by the encoder routine to perform two tasks; (i) update the parallel to serial converter, and (ii) swap the encoder input and output buffers.

Interrupt 2 occurs on every eighth transmitted clock cycle and an additional interrupt 2 request is generated on every encoder frame period. The two interrupt conditions can be differentiated by reading the encoder frame flag, bit 15 of the memory location 0x8000. If the flag is set then a swap buffer condition is valid, otherwise, a normal update of the data output register is performed. The interrupt request is automatically cleared when the DSP acknowledges the interrupt and begins the interrupt service routine.

Input

The encoded 4.8 kbit/s data is input through a serial to parallel (S/P) converter in the gate array. The S/P converter is not buffered and hence must be read on every external interrupt 1 request. External interrupt 1 is used for two purposes; (i) reading the contents of the serial to parallel converter, and (ii) swaping the decoder input and output buffers.

External interrupt 1 occurs on every eighth receive clock cycle and an additional interrupt 1 request is generated on every decoder frame period (frame sync pulses). The two interrupt conditions are differentiated by reading the decoder frame flag, bit 15 of memory location 0x8004. If the flag is set then a swap buffer condition is valid, otherwise, a normal read of the data input register is required. The data input register is accessed by reading memory location 0x8004. The interrupt request is automatically cleared when the DSP acknowledges the interrupt and begins the interrupt service routine.
7.9.4 Power Supply Budget

The analogue part of the board needs a voltage supply of ±5.0 volts and the digital part requires +5.0 volts (±5%). The two supplies are isolated from each other on the board and can only be connected together on the DIN connector. If a single power supply is used for both digital and analogue supplies, the analogue and digital ground needs to be tied together on the connector and twisted wires used. Most of the ICs used are either CMOS or CMOS compatible and hence the total power consumption of the board is 2.5 watts (500 mA at 5.0 volts).

7.10 Coder Performance

The output speech quality was evaluated by conducting subjective tests on simulations of the CELP-BB algorithm (21 subjects), and in real-time (22 subjects). The subjective testing of the simulations was carried out, at the initial stages of the project, in order to decide on the optimum setting of the codebook size and structure and the decimation factor (basically comparison of 4 kbit/s and 4.8 kbit/s source coding). The output speech quality at 4.8 kbit/s scored a MOS of 3.1 (on a 1 to 5 scale) as compared to a MOS of 2.4 at 4 kbit/s. It was therefore decided to pursue the development and refinement of the 4.8 kbit/s source coding [9].

The final performance of the complete coder was assessed in real-time, both under clean and erroneous channels, by conducting a formal subjective test. The coder performance test set-up is shown in figure 7.18. During the testing, speech material was played in real-time through a single board, connected in a back-to-back configuration. A tape recorder was used to play the speech samples and the output from the board was directed to a sound-proof room, where the subjects were seated. Subjects were given a brief background information about the INMARSAT-M application and its tasks before the test conduct. To simulate the channel errors, a Random Error Injection Box (REIB) is placed in between the outgoing 4.8 kbit/s and incoming 4.8 kbit/s data streams. The error rate of the REIB can be set for rates between 0% and 100%.

A total of 22 subjects and 12 test conditions, with the addition of 4 MNRU conditions for reference, were included in the test [22]. The 16 conditions were randomized, as shown in table 7.11, and played to the subjects. The first 8 sequences
Fig. 7.18 4.8 kbit/s Speech Coder Performance Test Set-Up
Table 7.11 Randomization of Subjective Test Conditions [22]

<table>
<thead>
<tr>
<th>1</th>
<th>5</th>
<th>2</th>
<th>8</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>10</td>
<td>16</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>14</td>
<td>11</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>11</td>
<td>15</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>9</td>
<td>12</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>16</td>
<td>10</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>16</td>
</tr>
</tbody>
</table>

1 = Male + no error; 2 = Male + $10^{-4}$; 3 = Male + $10^{-3}$; 4 = Male + $10^{-2}$
5 = Female + no error; 6 = Female + $10^{-4}$; 7 = Female + $10^{-3}$; 8 = Female + $10^{-2}$
9 = Male + Truck; 10 = Male + Truck + $10^{-2}$; 11 = Female + Truck; 12 = Female + Truck + $10^{-2}$
13 = MNRU Q-10; 14 = MNRU Q-15; 15 = MNRU Q-20; 16 = MNRU Q-25

were played from top to bottom and the following 8 in reverse order, to achieve better randomization. Two source tapes were provided by INMARSAT containing clean speech spoken by a wide range of male and female talkers as well as conversations recorded in a truck travelling at 100 km/h. Tables 7.12 and 7.13 summarize the results of the test (on a MOS scale of 0 to 4).

Table 7.12 indicates that the coder is robust to channel errors of up to 1%, and also on average a MOS of 1.9 (Fair) is obtained for the speech recorded in the truck. In order to compare these results with a standard reference, tests were conducted using speech contaminated with modulated noise. Table 7.13 gives the MOS scores for MNRU
Table 7.12 MOS and Variance Scores of the 4.8 kbit/s Speech Coder [22]

<table>
<thead>
<tr>
<th>Condition</th>
<th>Clean Speech</th>
<th>Speech + Truck Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Rate</td>
<td>0%</td>
<td>10^{-4}</td>
</tr>
<tr>
<td>Male</td>
<td>2.6</td>
<td>2.5</td>
</tr>
<tr>
<td>Female</td>
<td>2.5</td>
<td>2.6</td>
</tr>
<tr>
<td>Average MOS</td>
<td>2.55</td>
<td>2.55</td>
</tr>
<tr>
<td>MOS Variance</td>
<td>0.25</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Table 7.13 MOS and Variance Score for the MNRU Conditions [22]

<table>
<thead>
<tr>
<th>MNRU</th>
<th>Q-25</th>
<th>Q-20</th>
<th>Q-15</th>
<th>Q-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>3.6</td>
<td>3.4</td>
<td>2.7</td>
<td>2.0</td>
</tr>
<tr>
<td>Female</td>
<td>3.2</td>
<td>3.2</td>
<td>2.6</td>
<td>2.1</td>
</tr>
<tr>
<td>Average MOS</td>
<td>3.4</td>
<td>3.3</td>
<td>2.65</td>
<td>2.05</td>
</tr>
<tr>
<td>MOS Variance</td>
<td>0.27</td>
<td>0.26</td>
<td>0.20</td>
<td>0.15</td>
</tr>
</tbody>
</table>

(Modulated Noise Reference Unit) Q values of 10, 15, 20 and 25. According to most subjective experts anything equal to or above Q15 is considered to be of good quality. When compared with the MOS scores of table 7.12, we can conclude that the coder, with no background noise and channel errors, achieves a score almost exactly equivalent to the MNRU Q15 and achieves a Q10 equivalent quality with 1% error rate.

In order to evaluate the performance of the Frame Reconstruction scheme i.e. burst error performance of the coder, the software package provided by INMARSAT was run on an IBM PC [23]. The package generates 'good' (logic level '0') and 'bad' (logic level '1') frame indications, based on real-time data gathered in busy areas of London, and outputs these through the parallel port of the PC. These frame indications were passed to the programme through the parallel port of the DSP32C. We saw no noticeable degradation at short burst (under 3 frames i.e 90 ms) and for long bursts, the output was gradually muted. This avoids the usual 'bangs and clicks' in the presence of burst errors.
7.11 Concluding Remarks

The introduction of Skyphone services and the development of INMARSAT-M system is changing the traditional role of the INMARSAT organization in the world of communications. INMARSAT is no longer restricting its services to the maritime environment but extending these to the land and air. The rapid development in digital speech coding, DSP technology advances and the perceived need for global communications have been the driving force behind this push. New digital technologies enable the mobile equipment to be smaller and at the same time use the satellite power and bandwidth resources efficiently.

Our contribution in developing the INMARSAT-M system has been in the form of developing and setting-up a 'test-bed' for evaluating the digital speech coding performance and assisting the INMARSAT personnel in their definition of the design parameters of the final coder. The aim of the project was to design a robust digital speech coder at 4.8 kbit/s and to real-time implement it inside a compact and flexible hardware conforming to the INMARSAT specifications.

The objectives have been met in that a CELP-BB algorithm has been optimized to suit the land-mobile satellite INMARSAT-M type environment. Subjective tests have indicated that the performance of the implemented coder in the presence of vehicle ambient noise and with 1% transmission errors is FAIR (a score of 2 on a 0-4 MOS scale) and in error free conditions is GOOD (2.6 MOS). Demonstrations have also taken place of a lost frame mechanism which copes with bursty type errors on measured data produced by INMARSAT. The coder has been implemented on a single Eurocard (encoder and decoder + VAD + Lost Frame) using a single DSP32C digital signal processor chip. The input and output interfaces of the hardware were designed and built to INMARSAT specifications. A basic VAD system has been implemented and shown to function well as has been a mechanism for dealing with lost frames. Our restriction to a single DSP chip implementation has resulted in some compromises:

(i) DTMF tones are not perfectly passed.
(ii) VAD operation is not perfect in all conditions.
(iii) Robustness could not be extended to 4%.
However, these restrictions are not part of the characteristics of the algorithm and are mainly due to the restrictions on the available processing power and memory of the hardware. If these were to be resolved, a higher order DFT (1024 or 2048-point) during the LSF transformation, a more intelligent VAD system and forward error correction coding would easily overcome the short-comings. It is to be noted, that we have already found solutions to these problems and these are being implemented and integrated into future coders designed for different applications.
References


Conclusions

We can conclude that most of the human being's social, cultural and technical developments are because of his ability to exchange information, particularly via speech. This is a unique characteristic of mankind which distinguishes him from animals. The great need by human beings to converse with one another has lead to the development of long-distance communication systems, evolving from basic fire-smoke on hill-tops, to Graham Bell's telephone system, to Pulse Code Modulation (PCM) and nowadays to complex digital coding of speech at rates as low as 4.8 kbit/s or less. It is not surprising to see voice communication as the dominant part of nearly all the emerging telecommunication networks. Researchers in the field of voice communication are striving to find more efficient speech coding schemes at lower bit rates in order to accommodate the ever increasing number of users on channels with inherent limitations of bandwidth or power, such as cellular radio or satellite links.

As a small contribution to this ocean of knowledge, we have reported on the real-time implementation of three coders at rates of 13, 9.6 and 4.8 kbit/s. After the introductory part, the more general aspects of speech coding as applied to the satellite, land-mobile communication systems were considered. This was then followed by chapter 3, reviewing a number of modern techniques of digital speech coding at rates 16 to 2.4 kbit/s. Chapter 4, addressed the issue of real-time implementation in its general form and has tried to indicate the best media for real-time evaluation of speech coding techniques. The next three chapters carried an in-depth report on the theory and
software/hardware developments of the three coders. The remaining part of this chapter lists the main conclusions drawn from each study and concludes with the thoughts on the future trends.

**Digital Telephony**

Speech communication via transmission of digitally encoded speech is becoming more prevalent in telecommunication networks. It provides numerous advantages such as compatibility with data transmission, permitting the use of modern transmission techniques, facilitating security measures including encryption, and maintaining a standard of audio quality throughout the digital network. In fact, the developments of ISDN exemplifies the global need for this type of service [1].

The requirements of digital speech coders are very much application dependent. Military type of applications and the PSTN system are the two extreme areas. Between, these two extremes, there is a need for coders with moderate amounts of robustness and speech quality with low delay and small manufacturing costs.

The performance of a speech coder is usually evaluated by both objective and subjective quality measurements. Subjective quality measurements are commonly indicated by Mean Opinion Scores (MOS) and objective quality is measured in terms of Signal-to-Noise Ratios (SNR) or segmental SNRs (segSNR). There is a general dissatisfaction with measurements in both domains, as not reflecting the true performance of the low bit rate speech coders.

**Modern Speech Coding Techniques**

The last decade has seen the introduction of new applications for the transmission and storage of digital speech and the simultaneous rapid advances in VLSI/DSP technology has encouraged numerous studies into complex and potentially efficient speech coding algorithms. The two powerful schemes, Codebook Excited (CELP) coding in the time-domain and Sinusoidal Transform Coding (STC) in frequency domain, are able to generate speech of reasonably good quality at bit rates as low as 4.8 kbit/s. The progress in speech coding is leading to new standards for voice communications at rates of 4.8 - 16 kbit/s for digital cellular radio (Pan-European GSM DMR system, INMARSAT-M land-mobile, U.S. DoD DMR system) and for the PSTN network (CCITT 16 kbit/s standard). Although, CELP coding remains the dominant scheme at rates 16-4.8 kbit/s, the standardization of speech coding at 2.4 kbit/s may need a complete re-thinking of modelling the speech production, as high speech quality at such low bit rates cannot be achieved by incremental improvements of the existing
coding concepts. The dramatic progress in the DSP/VLSI field is gradually removing the 'computational complexity' obstacle, paving the way for the introduction of more elaborate techniques.

The present time-domain coders are classified into two types of systems; (i) Analysis and Synthesis (Vocoding), (ii) Analysis-by-Synthesis (AbS). The analysis and synthesis type of systems obtain the residual signal by an analysis procedure and then directly quantize and transmit this residual. The simplified and inadequate 'excitation models' of this system has lead to the more successful AbS schemes. In these systems, the model parameters are estimated using a closed-loop optimization procedure, which minimizes a 'perceptually' weighted mean-square measure found between the input and the decoded speech signals. Two of the most successful analysis-by-synthesis LPC systems are the Multipulse Excited and the Codebook Excited systems, with CELP coding showing much greater potential for near-toll quality speech at or below 4.8 kbit/s [2,3].

Real-Time Speech Coding

The real-time execution of a task requires a combination of efficient programming and a high-speed processor (a number cruncher). The bandwidth of a signal is a dominant factor in the complexity of real-time digital coding of that signal. It is for this reason that the real-time analysis of voiceband and audio signals may easily be possible with the available VLSI technology but the real-time processing of video signals require further maturity of VLSI technology.

The real-time implementation of complex digital systems such as digital speech coders has experienced dramatic changes through the advent of powerful digital signal processing (DSP) chips. Fixed and floating-point programmable DSPs have become commercially available, capable of performing several million operations per second. Although, fixed-point DSPs remain cheaper and faster at the present time but the numerous advantages of floating-point chips will force the manufacturers to look at ways of increasing their performance together with their widespread use causing a drop in their prices, they will become the user's absolute first choice.

Programmable DSPs are the most suitable media for evaluating the performance of digital speech coding schemes in real-time. It is easier and faster to develop prototypes and make algorithm or software changes ('tuning'). The In-Circuit-Emulation (ICE) facility is of absolute necessity for this purpose.
Pan-European Digital Mobile Radio System

In 1991, a new Pan-European DMR system will assume operation in 17 European countries, employing the latest state of technology, providing a common air interface and better spectrum efficiency as well as being cheap, compact and power efficient with a more secure communication link. Although, the system has provisions for a wide range of data services, with voice communication remaining the dominant service in the system, a considerable amount of time and effort was devoted to develop the two basic speech functions:

- a low bit rate speech coder,

- a scheme for voice activated transmission (discontinuous transmission) to increase the spectrum efficiency even further and to save battery power in handheld terminals.

As a joint collaboration between Surrey University and British Telecom Research Labs (BTRL) to set up an independent 'Test-Bed' to evaluate the selected coding scheme, the GSM speech coder was real-time implemented on two floating-point processors, AT&T DSP32s. The entire real-time software was developed at the Surrey Speech Research Group with the hardware being developed at BTRL. The functional description of the coding scheme was very closely followed. The most difficult part of the implementation proved to be the quantization of the residual signal where the given quantizer threshold and output levels only apply to the fixed-point implementation. The best solution proved to be the normalization of the input speech samples to ± 1.0.

Although, the speech coder quality was evaluated informally and proved to be of the GSM standard but the in-depth performance measurements of the system is being carried out within BTRL. It is hoped to report on these results in the future publications.

V-SAT Business Satellite Communication System

The simultaneous advances in both satellite communication technology and applicable electronics hardware has made a dramatic impact on the traditional role of satellites. New communication systems are emerging offering small private business and defence organizations, a compact, power efficient, and reliable portable satellite communication link.

One of these systems is the VSAT network, MP-SP2300, manufactured in UK by Multipoint Communications Ltd. As part of a contract, the system required the
development of a robust and efficient speech coder at 9.6 kbit/s. A new speech coding algorithm, developed in the Surrey University Speech Research Group, was modified and refined to suit such a system. The system needed to be self-synchronous and as a result a robust frame synchronization strategy with fast acquisition time was developed and then integrated into the coder. The speech coder was then real-time implemented, using two DSP32s (one for encoding and the other for decoding), on a very flexible hardware. Many functions of the hardware, such as phase-locked loop (PLL) range, internal or external PCM coding, were made user-configurable allowing total flexibility for different set-ups.

The study of the coder performance showed results superior to those required of the coder. Currently, the speech coding hardware is operating satisfactorily within a VSAT network in the Far East.

INMARSAT Land-Mobile Digital Radio System

Currently, INMARSAT is in the process of extending its services to the land-mobile area, through the INMARSAT-M system. Our contribution in developing the INMARSAT-M system has been in the form of developing and setting up a 'Test-Bed' for evaluating the digital speech coding performance and assisting the INMARSAT personnel in their definition of the design parameters of the final coder. As a result a robust digital speech coder at 4.8 kbit/s was designed and later real-time implemented inside a compact, flexible and at the time probably the world's smallest hardware (single Eurocard board), conforming to the INMARSAT specifications. Emphasis was placed on the robustness of the coder in the presence of vehicle ambient noise and the transmission errors. The transformation of LPC parameters into Line Spectrum Frequency (LSF) domain provided the excellent built-in robustness of these parameters at the expense of increased complexity. Extensive subjective quality measurements were taken and have been reported.

The coder has been implemented on a single Eurocard (encoder and decoder + VAD + Lost Frame Replacement) using a single DSP32C. The input and output interfaces of the hardware were designed and built to INMARSAT specifications. Although, the real-time software was hand-coded in the most optimized manner possible, our restriction to a single DSP chip implementation resulted in some compromises. The hardware, once again, designed to provide total flexibility.
Future Trends

There is a need for more realistic techniques of evaluating the performance of the low bit rate speech coders. The present objective measurements do not truly reflect the perceived quality of the coded speech. The subjective quality measurements are very extensive and time consuming. A closer study of the human auditory system and its modelling may result in the more accurate and automated subjective testing of the speech quality.

As suggested [4], the future work at low bit rates should attempt to devise algorithms that combine the flexibilities and capabilities of alternative approaches such as CELP and harmonic coding. In this process, it will be necessary to quantify the importance of the neglected phase fidelity in speech coding. It is very clear that researchers in the field of narrowband speech coding and those involved in the wideband audio coding should no longer work in isolation, as a hybrid of ideas may prove very rewarding to both sides. It is also believed that the rapid advances in both the VLSI technology and DSP techniques is making it difficult to justify the drop in speech quality at lower bit rates and users will soon demand a uniformity in the output speech quality at all bit rates. As well as improving the speech quality, the demands of the real world such as lower delays and robustness, must be considered from the outset rather than after the algorithm development [5].

The future will see the emergence of networks that combine data and speech for packetized type of communication, thus, requiring the design and development of efficient variable rate speech coders for asynchronous transmissions [6,7].

In evaluating the real-time performance of speech coders, the main problem associated with the use of DSPs is the very time consuming software development cycle. Existing high-level cross-compilers remain to be inefficient and even the eventual optimized compiler may not prove to be the complete solution. More work on the development of the highly promising Block-Diagram DSP Programming Environments [8] needs to be carried out to arrive at the final version of this neat and fast software development environment.

The simultaneous performance increase of the DSPs has played an important role in the design and development of new complex speech coding schemes. Following the same trends, we will see better quality speech coders at much lower bit rates, real-time implemented on single-chip processors, in the near future.
References


MATERIAL REDACTED AT REQUEST OF UNIVERSITY
Appendix B

Sample Source Codes

1. 4.8 kbps Speech Coder (Main Body)

2. Autocorrelation Function Subroutine

3. LPC Inverse Filter Subroutine (Lattice Structure)

4. LPC Synthesis Filter Subroutine (Lattice Structure)
/*--------------------------------------*/
/*     === 4.8 KBPS Speech Coder ===    */
/*     (Using DMA and Interrupts)        */
/*     Version October 1989             */
/*--------------------------------------*/

#define encod_data 0x8000
#define decod_data 0x8004
#define frame1 240
#define frame2 48
#define frame3 16
#define order 10
#define SYMAX 512
#define dec 3
#define nogrids 3
#define minpitch (frame3)
#define maxpitch (frame3+32)
#define booksize 512
#define bookshft 1
#define loop (frame1/frame2)
#define magalfa 8 /* no. of output levels for alfa quantizer */
#define winpts 8
#define hangover 4

/*--------------------------------------*/
/* Initialization                       */
/*--------------------------------------*/
start:
    pin = encin1 /* DMA Input Pointer */
    pout = decout1 /* DMA Output Pointer */
    ivtp = itable /* Interrupt Vector Table Pointer */
    rl = encinl
    r2 = encin2
    r3 = encoutl
    r4 = encout2
*encinwrite = rl /* Encoder ip to be written into by DMA */
*encinread = r2 /* Encoder ip to be read by processing block */
*encoutread = r3 /* Encoder op to be read by Interrupt routine */
*encoutwrite = r4 /* Encoder op to be written into by proc. block */
*encoutptr = r3 /* ISR Encoder output pointer */
    r1 = decin1
    r2 = decin2
    r3 = decout1
r4 = decout2
*decinwrite = r1 /* Decoder ip to be written into by Interrupt */
*decinread = r2 /* Decoder ip to be read by processing block */
*decoutread = r3 /* Decoder op to be read by DMA */
*decoutwrite = r4 /* Decoder op to be written into by proc. block */
*decinptr = r1 /* ISR Decoder input pointer */

r51 = piop /* Parallel I/O port */
nop
r5 = r5 & 0x0040 /* check AULAW flag */

if (eq) goto start0

dauc = 3 /* if flag=0, then A-law PCM */

dauc = 0 /* else u-law PCM */

start0: ioc = 0x6447 /* DMA: 8-bit input, 8-bit output */
rl = 0x8400
pcw = rl /* enable interrupts: IREQ1 & IREQ2 */

rl = 0
*encflag = rl /* encoder flag */
*decflag = rl /* decoder flag */

main: rl = *decflag

nop
if (eq) goto main0 /* if decoder input buffer not full */
rl = 0

call decod (rl8) /* Else call Decoder Subroutine */
*decflag = rl /* reset the flag */

main0: rl = *encflag

nop
if (eq) goto main /* if encoder input buffer not full */
rl = 0

call encod (rl8) /* Else call Encoder Subroutine */
*encflag = rl /* reset the flag */
go to main

nop

itable: goto decod_isr /* Decoder Interrupt Service Routine */
*(istack) = rl /* store r1 */
8*nop /* unused Interrupts */
goto encod_isr /* Encoder Interrupt Service Routine */
*(istack) = rl /* store r1 */

/*--------------------------------------------*/
/* Encoder Subroutine */
/*--------------------------------------------*/
encod: *stack = rl8

call pre (rl4)
nop
int (frame1-2), 0
call win (rl4)
nop
int syin, pad1, (frame1-2), 0
call acf (rl4)
nop
int pad1, (frame1-2), (order-1), ACF
call durbin (rl4)
nop
int order-1, 0
call lpc_lsp (rl4)
nop
int acoef, lsf
call lsfchk (rl4)
r18 = r14
call lsfcomp (rl4)
nop
int lsf+4, count, lsfrange, (order-3)
call qlsf (rl4)
r18 = r14
int lsf, Lcode
int maxltab, maxcode
call lsp_lpc (rl4)
r18 = r14
int lsf, acoef
call ilfr (rl4)
nop
int syin, acoef, ilfrmem, 0

/*----------------------start of subsegmentation---------------------------*/

LTP: rl = 0 /* index of 1st subsegment */
*pointer = rl /* subsegment pointer */
*fcounter = rl /* subsegment counter */
ltP0:  call wgtflr (rl4)  /* Weighting Filter for decimation */
nop
int (frame2-2), padl

call grid (rl4)  /* Base-Band Extraction */
nop
int padl, (dec-2)

call ltpana (rl4)  /* Long-Term Analysis */
nop
call qpgain (rl4)  /* Long-Term Gain Quantization */
nop
call eLTPflr (rl4)  /* Long-Term Inverse Filter */
nop
call svquan (rl4)  /* Sequential Codebook Search */
nop
int error, bookshft

r1 = *fcounter
r2 = *code
r1 = r1 + address
*r1 = r2  /* posn. of selected sequence */
call qalfa (rl4)  /* Codebook Gain Quantization */
r18 = r14
call seqrsq (rl4)  /* reconstruct the error signal */
nop
int address, (bookshft*4)
call update (rl4)  /* update Encoder LTP memory */
nop
int eLTPmem, 0
call incptr (rl4)  /* increment pointer & frame counter */
nop
int ltp0, 0

/*!-------------------------------end of subsegmentation-------------------------------*/
call encvad (rl4)
r18 = r14
call intcnv (rl4)  /* Integer-to-Bit Conversion */
nop
int Lcode, nbits, (35-2), 0

r11 = *opflags
r18 = *stack
*decod_data = r1  /* write out output status flags */
return (r18)
nop

/*---------------------------------------------*/
/*  Decoder Subroutine */
/*---------------------------------------------*/
decod: r11 = piop
  *stack = r18
  *ipflags = r1
  /* read in input status flags */
  r1 & 0x0010
  if (ne) goto decod0
  /* check DESLNCE flag */
  nop  /* if 'high' then speech frame */
  call mute (r14)
  /* else mute output & exit decoder */
  nop

int muteconst, (frame1-2)

decod0: r1 & 0x0008
  if (ne) goto decod1
  /* check lost frame flag */
  nop  /* if 'high' then good frame */
  call badchan1 (r14)
  /* else reconstruct lost frame */
  nop
int dLTP, 0  /* then branch to dLTP */

decod1: call bitcnv (r14)
  nop

dltp0: call unidec (r14)  /* decode LTP gain parameter */
  nop

int dpgain, pgtable, bj, 0

call sftdec (r14)  /* decode LTP lag parameter */
  nop

call decalfa (r14)
  r18 = r14  /* decode code-book gain parameter */

call dLTPflr (r14)  /* Decoder Long-Term Filter */
  nop

call segrsq (r14)  /* reconstruct the error signal */
  nop

int daddress, (bookshft*4)
call update (rl4) /* update decoder LTP memory */
nop
int dLTPmem, 0

call upsample (rl4) /* reconstructing excitation signal */
nop
int dposn, 0

call incptr (rl4)
nop
int dltp0, 0

/*@------------------------end of subsegmentation----------------------------*/

call declsf (rl4)
nop
int lsf, dLcode, maxltab, 0

call errchk (rl4)
nop
int lsf, prelsf

call lsp_lpc (rl4)
r18 = r14
int lsf, acoef

call sflr (rl4)
nop
int syin, acoef, sflrmem, 0

call replace (rl4)
nop
int lsf, prelsf, (order-2), 0

call postflr (rl4)
r18 = r14

call post (rl4)
nop
int syin, (frame1-2)

r18 = *stack
nop
return (r18)
nop
/*

call acf (rl4)
nop
int input_array, (frame_size-2), (order-1), ACF
*/

/************************************
/* computing the values of the autocorrelation function */
/************************************
acf: r1 = *r14++  /* start address of input buffer */
r3 = *r14++    /* frame size */
r6 = *r14++    /* no. of ACF values-2 */
r7 = *r14++    /* start address of ACF array */
r10 = zero
a0 = *r10     /* a0 = 0.0 */
r2 = r1    
r8 = r1    
r4 = r3    
r5 = 4     /* pointer offset (flp) */

acf0: if(r3 -- >= 0) goto acf0
a0 = a0 + *r14++ * *r2++    /* a0 += data[t] x data[t+j] */
*r7++ = a1 = a0    /* store results in ACF array */
a0 = *r10  /* a0 = 0.0 */
r1 = r8    
r2 = r8    
r2 = r2 + r5   /* r2 points to data[t+j] */
r3 = r4 - 1
r4 = r3
if(r6 -- >= 0) goto acf0  /* loop for the next value of ACF */
r5 = r5 + 4    /* offset data pointer by 4 (flp) */

return(rl4)
nop

/************************************
zero: float 0.00000000e000
call iflr (r14)
nop
int input_array, (frame_size-2), (order-2), 0
*/

/*--------------------------------------------------------*/
/* Inverse filtering (PARCOR structure) subroutine */
/*--------------------------------------------------------*/
iflr: 
  r8 = *r14++  /* input and output array */
  r6 = *r14++  /* input frame size - 2 */
  r5 = *r14++  /* no. of ref. coeffs. - 2 */
  r7 = r8      /* output array */
  r10 = r5     /* */
iflr0: 
  r4 = pad3    /* new values of filter memory */
  *r4++ = a3 = *r8++  /* b[0] = cr[0] = data[k] */
  r2 = rcoef    /* reflectin coeff. array */
  r3 = iflr_mem /* memory of inverse filter */
iflr1: 
  r9 = r10    /* latency */
  a0 = *r3    /* a0 = pb[k] */
  *r3++ = a3 = a0 + *r2 * a3  /* b[k]=pb[k] + r[k-1].er[k-1] */
  if(r5-- >= 0) goto iflr1
  a3 = a3 + *r3++ + *r2++  /* er[k]=er[k-1] + pb[k-1].r[k-1] */
  r4 = pad3
  r3 = iflr_mem
  *r7++ = a3 = a3  /* residual[k] = er[8] */
iflr2: 
  if(r9-- >= 0) goto iflr2
  *r3++ = a0 = *r4++  /* pb[k] = b[k] */
  if(r6-- >= 0) goto iflr0
  r5 = r10
  goto r14+2
  nop

/*--------------------------------------------------------*/
iflr_mem: (order)*float 0.0
pad3: (order+1)*float 0.0
rcoef: (order)*float 0.0
/* call synflr (r14) */
int input_array, (frame_size-2), (order-2), 0

ในฐานะ פונקציה: Synthesis Filter (PARCOR structure) subroutine

// input and output array
r6 = *r14++
frame size 2
r5 = *r14++
order of filter 2
r7 = r8

// output array
r11 = r5 + 1
r11 = r11*2
r11 = r11*2
r12 = r11
r13 = r11 + 4
r11 = rcoef + r11
end of reflection coeff. array
r12 = sflr_mem + r12
present memory of filter
r13 = pad3 + r13
new values of memory
r15 = r5

sflr0: a3 = *r8++
s[0] = d'(k)
r2 = r11
r(n)
r3 = r12
r4 = r13

sflr1: a3 = a3 - *r2 * *r3
latency
r1 = pad3
if (r5-- >= 0) goto sflr1
*r4-- = a1 = *r3-- + a3 * *r2--

*speech(k) = s[n]
*r4 = a3 = a3
*d[0] = s[n]
*r7++ = a3 = a3
r10 = sflr_mem
r9 = r15

sflr2: if (r9-- >= 0) goto sflr2
*r10++ = a0 = *r1++
update filter memory
if (r6-- >= 0) goto sflr0
r5 = r15

goto r14+2
nop

sflr_mem: (order)*float 0.0
pad3: (order+1)*float 0.0
rcoef: (order)*float 0.0