NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents

Digital-Signal-Processor-Based ADPCM Codecs

Tân Thị Bùi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Electrical Engineering at
Concordia University
Montréal, Québec, Canada

March 1989

ISBN 0-315-49106-X

Canada

ABSTRACT

Digital-Signal-Processor-Based ADPCM Codecs

Tân Thị Bùi

The 32 kbits/s Adaptive Differential Pulse Code Modulation (ADPCM) algorithm approved by the International Telegraph and Telephone Consultative Committee (CCITT) as an international standard has been implemented in several systems. So far, most modifications to the algorithm have been to facilitate the implementation.

This thesis attempts to reduce the system speed and the bandwidth usage through the introduction of more look-up tables and fixed poles. It first reviews several ADPCM algorithms. Techniques to improve the system execution time are then discussed and the structure of a processor optimized for the CCITT algorithm is outlined. Test results reveal that the modified algorithm as realized on the digital signal processor TMS32020 can save as much as 34.51% of the total execution time. Furthermore, the 24 kbits/s ADPCM system shows to reduce both the bandwidth and the execution time. Finally, the performance of ADPCM systems in tandem, and with voiceband data signals is discussed.

ACKNOWLEDGEMENTS

I wish to thank Dr. Thọ Lê-Ngọc for his invaluable guidance and assistance without which it would have been impossible to complete this thesis.

Thanks are due to Messrs. M.T. Võ and Y. Shayan whose participations in the modification of the system hardware have enabled the implementation and, thus, the performance evaluation conducted in this thesis.

I also wish to acknowledge use of library services at Teleglobe Canada Inc.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF SYMBOLS

Symbol	Meaning
$\{a_i\}$	Predictor coefficient (pole filter)
B	Number of bits in quantizer
$\{b_i\}$	Predictor coefficient (zero filter)
$b^{(i)}(k)$	Error of an i^{th}-order backward predictor
$e(k)$	Quantizer input in DPCM system
$e_q(k)$	Quantizer output in DPCM system
$e^{(i)}(k)$	Error of an i^{th}-order forward predictor
$K^{(i)}$	Partial Correlation Coefficient
$n_q(k)$	Quantization noise at time k
$p(s)$	Probability density function of $s(k)$
S	State of source in Markov process
$s(k)$	Input signal sampled at time k
$\tilde{s}(k)$	Predicted value of $s(k)$
$\hat{s}(k)$	Reconstructed value of $s(k)$
$s_a(k)$	Analog signal
$s_c(k)$	Compressed signal of A- or μ-law quantizer
$s_d(k)$	Decoded signal
$s_{ez}(k)$	Output of zero filter
$\sigma^2_{n_q}$	Mean square value of $n_q(k)$
σ_s	Root mean square value of $s(k)$
σ^2_s	Mean square value of $s(k)$
T	Cycle of signal
W_0	Frequency of signal

CHAPTER I

INTRODUCTION

Coding of signals into digital form in telecommunications offers several advantages which have generally been cited as ruggedness, efficient signal regeneration, easy encryption, the possibility of combining transmission and switching functions, and a uniform format for different types of signals [1]. On the other hand, digital technologies have evolved to a stage such that the reliability, efficiency, economies, just to mention a few attractive points, of digital systems have brought about increasing interest in digital signal processing.

Generally, digital coding techniques use waveforms or parameters to represent the original signal. Hence, a signal can have waveform representation or parametric representation depending upon whether its waveform is represented directly or whether the representation is in terms of time-varying parameters of the basic signal model. In waveform coding, the shape of the signal is preserved as much as possible so that the analog waveform can be reconstructed from its digital representation [2].

Waveform coders based on different techniques and different algorithms have been built, tested and compared with one another [3], [4], [5]. The ultimate goal is to find the one which can transmit signals with the highest possible quality over the least possible channel capacity, and with the least cost [6].

This thesis reviews several algorithms performing Adaptive Differential Pulse Code Modulation (ADPCM) coding/decoding method, which is a form of waveform coding. It then looks at the implementation of an ADPCM algorithm on digital signal processor (DSP). The performance of the specific codec as well

as of ADPCM codecs in general is then discussed. Attempts are also made to reduce the system execution time and to identify the structure of a processor optimized for the algorithm.

Chapter II introduces the concepts of Pulse Code Modulation (PCM), Differential Pulse Code Modulation (DPCM), and Delta Modulation (DM). It describes the basic processes in a waveform coder, namely the sampling, the quantization and the coding processes. Different types of quantizers, uniform, non-uniform and optimum quantizers, are then presented with their respective characteristics. Both DM and DPCM deal with the difference between a signal and its predicted value, with DM usually considered as the 1-bit version of DPCM. Linear prediction plays an important role here. The methods to solve for the predictor coefficients are detailed : direct methods (autocorrelation, covariance, lattice), and iterative methods.

Chapter III features the adaptive schemes in DPCM. The notions of forward/backward and instantaneous/syllabic adaptation underlie the different ADPCM techniques. Each has its own advantages and disadvantages. More often, it is the aim of an application that dictates the specific scheme to be implemented in the system. ADPCM systems have adaptive quantizer and/or adaptive predictor. The chapter concludes with a literature survey on the adaptive quantization and adaptive prediction algorithms, and their implementation.

The implementation of the 32 kbits/s ADPCM algorithm, which is approved by the International Telegraph and Telephone Consultative Committe (CCITT) as an international standard, is the topic of chapter IV. First of all, the elements of a CCITT ADPCM system are described. The full-duplex implementation of this algorithm on the DSP TMS32010 is then studied. The processing requirements, both hardware and software, are identified which lead

to the consideration as to what techniques, modifications could be applied to improve the system execution time. Finally, the structure of a processor optimized for the implementation of the CCITT ADPCM algorithm is proposed.

Chapter V reports the performance of ADPCM codecs. The TMS32010-based ADPCM codec is stated to have satisfactory performance. The TMS32020-based codec with modified software proves to reduce execution time. The performance of ADPCM in tandem and with voice-band data (VBD) has been the subject of several studies. In general, between two and four ADPCM links in tandem would be tolerated, and ADPCM coder can handle VBD signals of up to 4800 bits/s. When 3-bit code is used in the codec, the bit rate is reduced to 24 kbits/s. It is observed that with similar modified software, 24 kbits/s ADPCM can improve the execution time as 32 kbits/s ADPCM.

The final chapter, chapter VI, goes over the main points of the thesis. It then concludes with suggestions for further studies.

CHAPTER II

PCM AND DPCM

Waveform coding techniques employ discrete-time, discrete-amplitude representations of signals for transmission or digital storage. Of these, we can cite Pulse Code Modulation (PCM), Delta Modulation (DM), Differential Pulse Code Modulation (DPCM), and their associated adaptive version.

Suggested by Reeves [7] in 1938, PCM is one of the simplest forms of waveform coding. To provide "facsimile" reproduction of the signal waveform, PCM takes the samples of the signal, quantizes them into a number of discrete levels, and uses a code to designate each level at each sample time. Although PCM can provide good transmission quality, it is inefficient in its bandwidth utilization. By transmitting the difference between samples, instead of the samples themselves, DM and DPCM can retain the same performance level as PCM at a lower bit rate. Therefore, bandwidth reduction without sacrifice on performance can be achieved through the use of such redundancy-removal schemes.

2.1. PCM

PCM systems use a series of discrete code symbols to convey information. Therefore, the transmission of signals using PCM technique involves the processes of sampling, quantization, and coding.

2.1.1. Sampling

In waveform coding, the shape of the analog signal is preserved by applying Nyquist's sampling theorem which states that any band-limited signal can be reconstructed from samples taken periodically in time if the sampling rate is at least twice the highest frequency of the signal [8]. In fact, a signal which is band-limited to a certain frequency W_0 can assume exactly $2W_0$ independent values per second. As a result, to convey the information contained in a band-limited signal of duration T, it is necessary to send only a finite set of $2W_0T$ independent values. These can be obtained by sampling the instantaneous amplitude of the signal at a regular rate of $2W_0$ samples per second (the Nyquist rate). Instead of transmitting the complete signal in analog form, we really need to transmit only a discrete number of samples produced from the sampling process [9].

Needless to say, the characteristics of the signals to be transmitted must be considered, in order to determine the sampling rate. The most frequently encountered signal in a telephone channel is speech the long-term power density spectrum of which is shown in Fig.2. Other signals carried by telephone channels include data signals, amongst which the most critical are those with the highest speed 4800 bits/s in public switched networks [11]. It can be observed, from Fig.2, that most frequency components in a telephone transmission path are typically below 4 kHz. Hence, a sampling rate of 8 kHz is generally used.

2.1.2. Quantization

The output of the sampling process is a continuous variable, i.e. it can take on a continuum of values. However, it is possible to allow only certain discrete levels of amplitude or position of the transmitted pulse. Hence, when the signal is sampled, the level nearest to the true signal level is yielded. Representing the signal by certain discrete allowed levels only is called quantizing. Obviously,

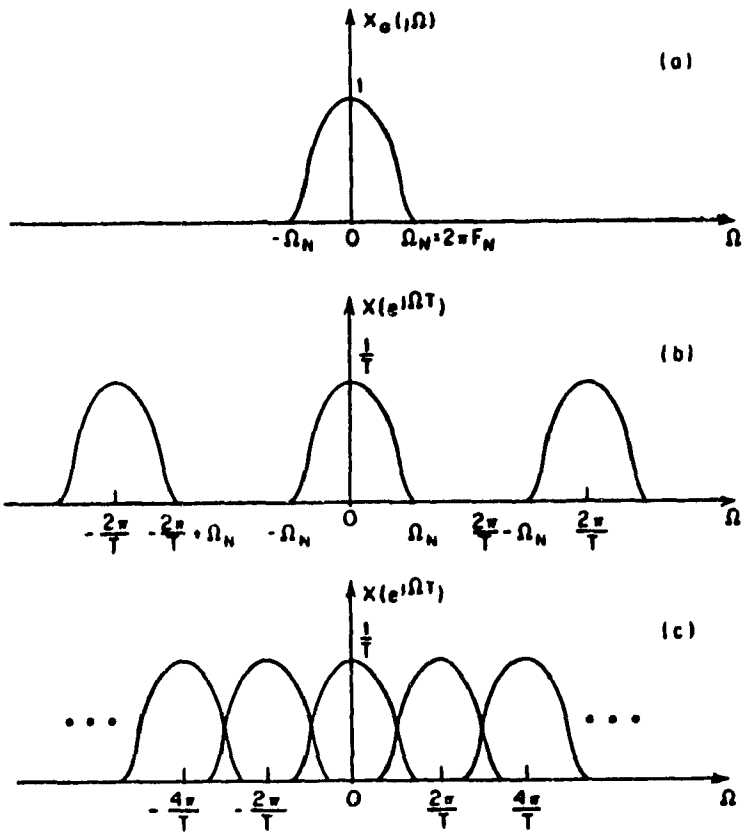


Fig.1. Illustration of sampling. (After Rabiner and Schafer [10].)

this process of quantization introduces some fluctuations about the true value. These fluctuations can be regarded as noise, and they are usually called quantization error (or quantization noise). This kind of error, "granular" error, is inherent in quantization. Meanwhile, "overload" errors occur when the step size between successive levels is too small and a signal falls outside the quantizer range. Between these two basic types of quantization error, "granular" errors tend to be less harmful to the quality of the output signals than "overload" errors [1].

The quantized sample is not transmitted directly. Usually, it is passed through a coder which assigns a different codeword to each level. In a majority of digital communication systems, the actual form chosen for codewords is a

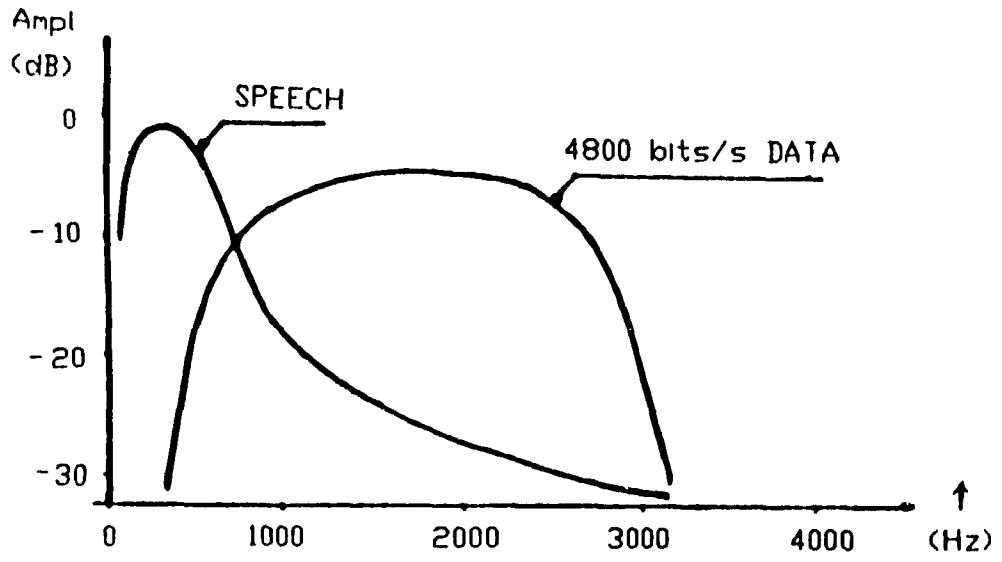


Fig.2. Spectra of speech and 4800 bits/s data. (After Raulin, Bonnerot, Jeandot, and Lacroix [11].)

binary number composed of 1's and 0's. If the binary codewords have B bits each, the code then can represent 2^B different quantization levels, and the bit rate is $2BW_0$ bits/s.

A simple waveform coder would incorporate the processes just described.

The block diagram in Fig.3 depicts a Pulse Code Modulation (PCM) system. Since the 1960's, 64 kbits/s PCM, where 8 bits are used to code signals sampled at 8 kHz, has been the only coding technique defined as a standard for worldwide use [8].

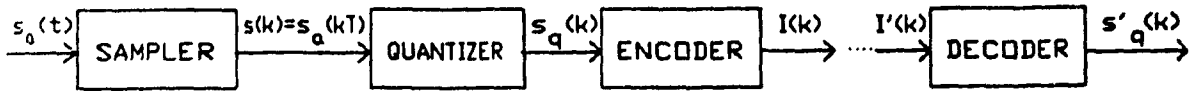


Fig.3. A PCM system.

A frequently used indicator of performance in waveform coder systems is the signal-to-quantization-noise ratio (SNR) which is defined as :

$$SNR = \frac{\sigma_s^2}{\sigma_{n_q}^2} = \frac{E[s^2(k)]}{E[n_q^2(k)]} = \frac{\sum_k s^2(k)}{\sum_k n_q^2(k)}$$

where σ_s^2 is the mean square value of the input signal s,

$\sigma_{n_q}^2$ is the mean square value of the quantization noise n_q,

$s(k)$ is the input signal sampled at time k,

$n_q(k)$ is the quantization noise at time k.

The quantity is often expressed in decibel (dB) as $10\log_{10} SNR$

In telephone channels, the term "toll quality" implies $SNR \geq 30$dB [6].

---

* An equation will not be numbered when it is:

- a definition,
- a step in an algorithm,
- not a key equation, or not referred to in other part of the text.

The SNR of any system can be increased by decreasing the quantization noise $n_q(k)$ that can be realized through an increase in the number of quantization levels assigned.

Quantization can be uniform or non-uniform.

2.1.2.1. Uniform Quantizers

A *uniform quantizer* provides the same spacing between successive levels, in other words, the quantization step size Δ is constant.

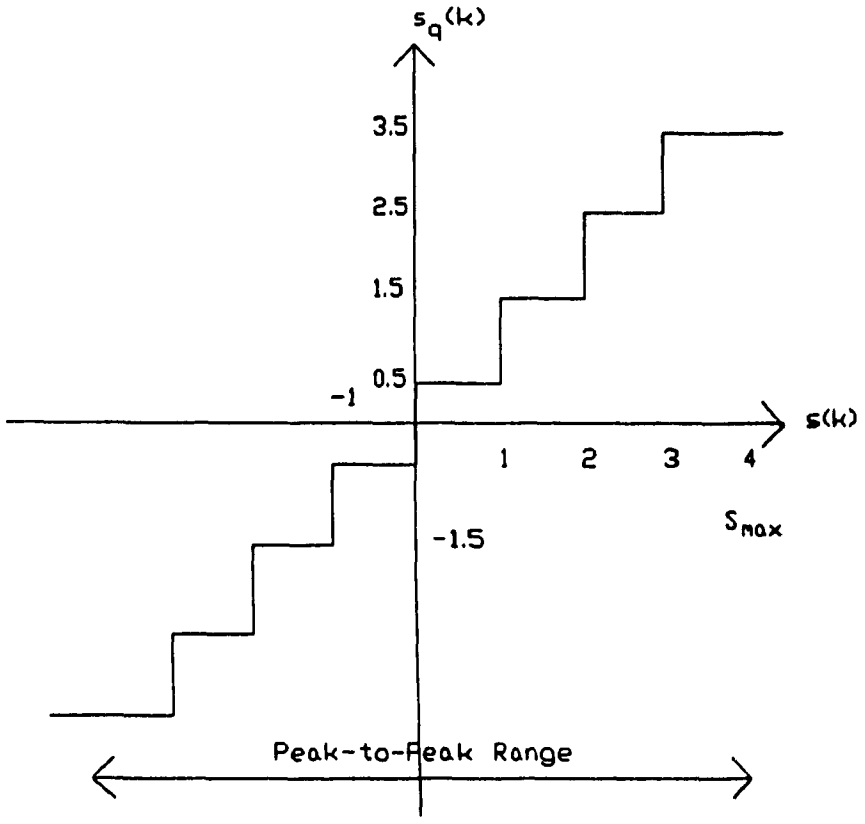


Fig.4. Uniform quantizer.

If the peak-to-peak quantizer range is assumed to be $2S_{max}$, then, for a B-bit quantizer,

$$\Delta = \frac{2S_{max}}{2^B}$$

and

$$SNR(dB) = 6B + 4.77 - 20\log_{10}[\frac{S_{max}}{\sigma_s}] \qquad (1)$$

where σ_s is the root mean square (rms) value of the input signal s.

2.1.2.2. Non-uniform Quantizers

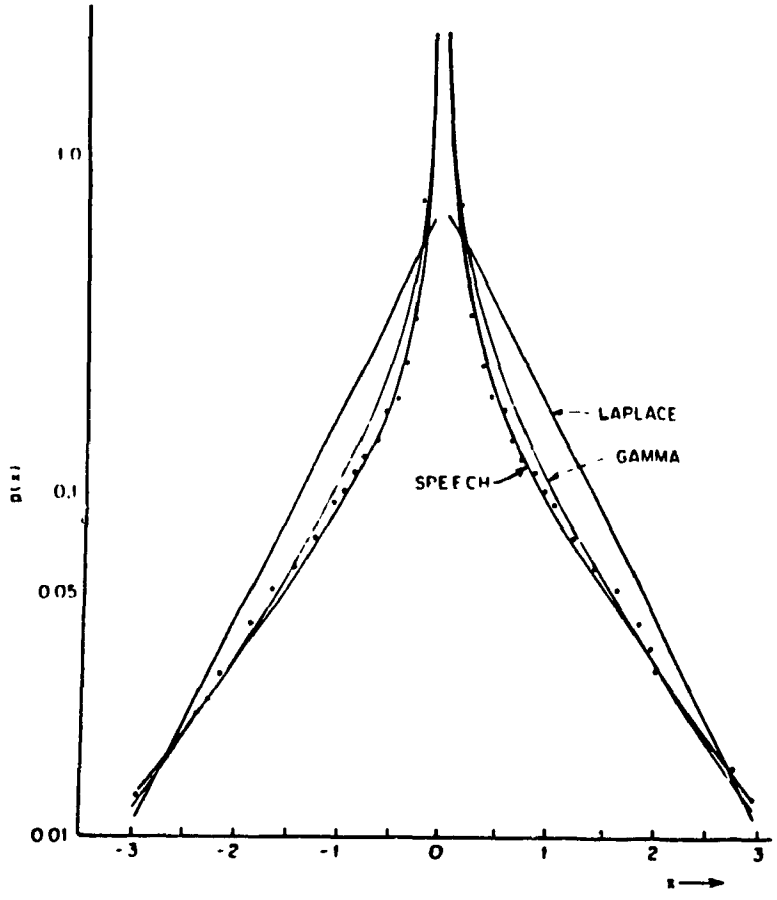


Fig.5. Amplitude density for speech. (After Rabiner and Schafer [10].)

Many source signals have the statistical characteristic that small signal amplitudes occur more frequently than large signal amplitudes (see Fig.5). A better approach is to have more closely spaced levels (characterized by fine quantizing steps) at the low signal amplitudes and more widely spaced levels (characterized by much coarser quantizing steps) at the large signal amplitudes [12].

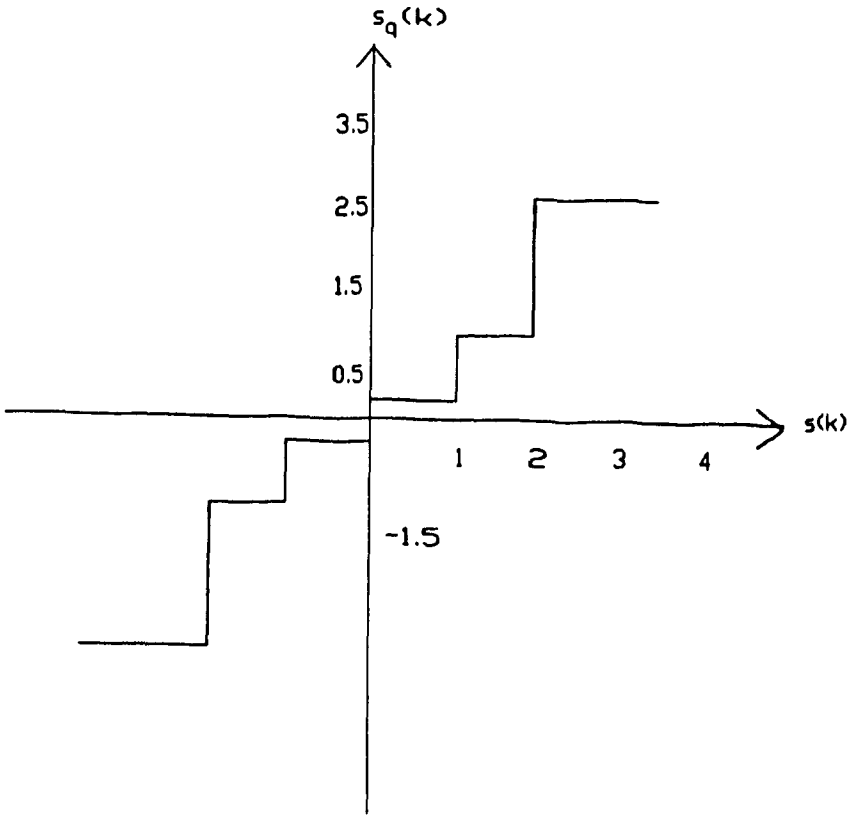


Fig.6. Non-uniform quantizer.

The advantage of such a quantizer is that, without increasing the total number of quantization levels, one can allow large end steps in the quantizer to take care of possible excursions of the input signal into the relatively infrequent large amplitude ranges.

A non-uniform quantizer characteristic can be obtained by passing the signal through a non-linear device that compresses the signal amplitude, followed by a uniform quantizer. For example, in commercial telephony, a logarithmic compressor is used in the Smith quantizer. In the reconstruction of the signal from the quantized values, the inverse logarithmic relation is used to expand the signal amplitude. The combined compressor-expandor pair is termed a compandor. The amplitude compression characteristics used in log-quantization follow either the μ-law or the A-law. For $s(k) > 0$ and $S_{max} = 1$, the compressed signals $s_c(k)$ are defined as follows :

$$\mu - law \qquad s_c(k) = \frac{\ln[1 + \mu s(k)]}{\ln(1+\mu)} \qquad \mu > 0$$

$$A - law \qquad s_c(k) = \frac{As(k)}{1 + \ln A} \qquad 0 \leq s(k) \leq A^{-1}$$

$$= \frac{1 + \ln As(k)}{1 + \ln A} \qquad A^{-1} \leq s(k) \leq 1$$

For a μ-law quantizer, the signal-to-quantization-noise ratio is derived :

$$SNR(dB) = 6B + 4.77 - 20\log_{10}[\ln(1+\mu)] - 10\log_{10}\left[1 + \left(\frac{S_{max}}{\mu\sigma_s}\right)^2 + \sqrt{2}\left(\frac{S_{max}}{\mu\sigma_s}\right)\right] \qquad (2)$$

Compared with Eq. (1), Eq. (2) indicates a much less severe dependence of SNR upon the quantity (S_{max}/σ_s). As μ increases, the SNR becomes less and less sensitive to changes in (S_{max}/σ_s).

A uniform quantizer would require about 11 bits to obtain the same dynamic range (ratio between largest and smallest values) as 7-bit log PCM which is generally taken as a standard for "toll quality" representation of the speech waveform.

Equations (1) and (2) state that each bit used contributes 6 dB to the SNR of the system.

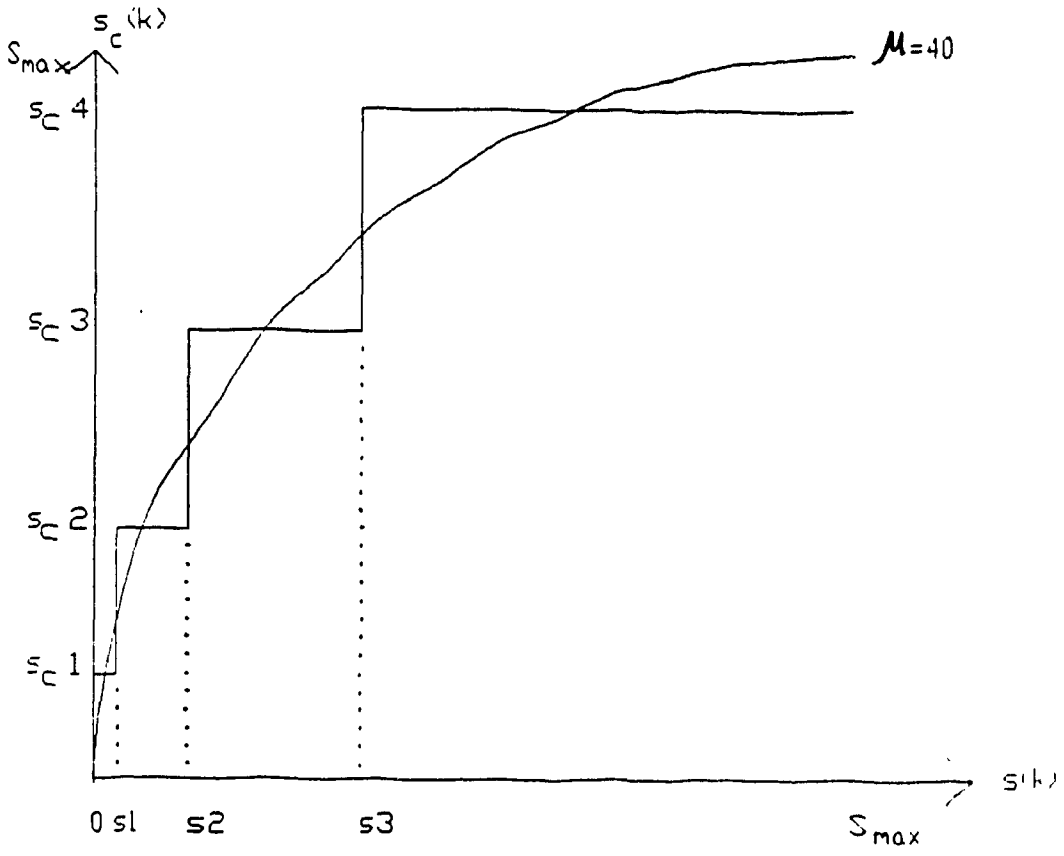


Fig.7. An 8-level μ-law quantizer with $\mu = 40$.

2.1.2.3. Optimum Quantization

From Eq. (2), it is observed that the SNR of a μ-law quantizer is relatively insensitive to σ_s, the rms value of the input s, and it is also relatively insensitive to the amplitude density of s. Consequently, constant SNR can be achieved over a wide range of signal variances. Nevertheless, the SNR performance can be improved if the quantizer step size is matched to the variance of the signal. In cases where the signal variance is known, quantizer levels can be chosen so as to minimize the quantization error variance, and hence, maximize the SNR. Consider a quantizer with L quantization levels, assuming L is even. The quantization level associated with the interval s_{i-1} to s_i is denoted as s_{q_i}.

For a symmetric, zero mean amplitude distribution, let the central boundary point $s_0 = 0$ and the extremes of the outer range $s_{\pm L/2} = \pm\infty$.

The variance of the quantization noise is given by

$$\sigma_{n_q}^2 = E\left[[s(k) - s_q(k)]^2\right]$$

$$= 2\sum_{i=1}^{L/2} \int_{s_{i-1}}^{s_i} (s - s_{q_i})^2 p(s) ds$$

where $p(s)$ is the probability density function of s.

In order to determine the sets of parameters $[s_i]$ and $[s_{q_i}]$ minimizing $\sigma_{n_q}^2$, we differentiate $\sigma_{n_q}^2$ with respect to each parameter and set the derivative to zero. The optimum (or minimum mean-squared-error or maximum signal-to-noise-ratio) quantizer would satisfy the following conditions:

$$s_0 = 0 \qquad s_{\pm L/2} = \pm\infty$$

$$\int_{s_{i-1}}^{s_i} (s - s_{q_i})\, p(s)\, ds = 0 \qquad i = 1, 2, \cdots, L/2 \tag{3}$$

$$s_i = \frac{1}{2}\left(s_{q_i} + s_{q_{i+1}}\right) \qquad i = 1, 2, \cdots, L/2 - 1 \tag{4}$$

Eq. (4) shows that the optimum boundary point s_i lies midway between the adjacent output levels, while Eq. (3) indicates that the optimum output levels are the centroids of the area under $p(s)$ and the adjacent input boundaries [10].

The above equations are not soluble without recourse to numerical methods. Max [13] suggested an iterative procedure and tabulated the values of s_i and s_{q_i} of both uniform and non-uniform quantizers for signals with Gaussian density

$$p(s) = \frac{1}{\sqrt{2\pi}} e^{-s^2/2}.$$

Following a numerical procedure similar to the one suggested by Max, Paez and Glisson [14] determined the optimum (both uniform and non-uniform) quantizer characteristics for Laplacian-distributed signals

$$p(s) = \frac{\alpha}{2} e^{-\alpha|s|}$$

and gamma-distributed signals

$$p(s) = \frac{\sqrt{k}}{2\sqrt{\pi}} \frac{e^{-k|s|}}{\sqrt{|s|}}$$

When matched to the variance and amplitude distribution of the signal, optimum quantizers yield minimum mean squared error. In practice, the input signals need not be stationary. The SNR will fall off rapidly as the level of the signal deviates from the level to which the quantizer is matched. In addition to this, even under the assumption of a constant mean level, the subjective performance of optimum quantizers is poor, as the idle channel noise is high for those quantizers. Consequently, according to Noll, optimum quantizers, per se, cannot be used [3].

2.2. DPCM and Linear Prediction

Most source signals sampled at the Nyquist rate or faster exhibit significant correlation between successive samples. The meaning of this high correlation is that the average change in amplitude between successive samples is relatively small so that the difference between adjacent samples should have a lower variance than the variance of the signal itself. As differences between samples are expected to be smaller than the actual sampled amplitudes, fewer bits would be required to encode the differences. This would lead to straightforward bandwidth reduction. The scheme where signals are represented not in terms of waveform amplitudes, but in terms of the differences between waveform

amplitudes is called Differential Pulse Code Modulation (DPCM). The origins of DPCM stem from patents by the N.V.Phillips Company in 1951 and by C.C.Cutler in 1952 [15].

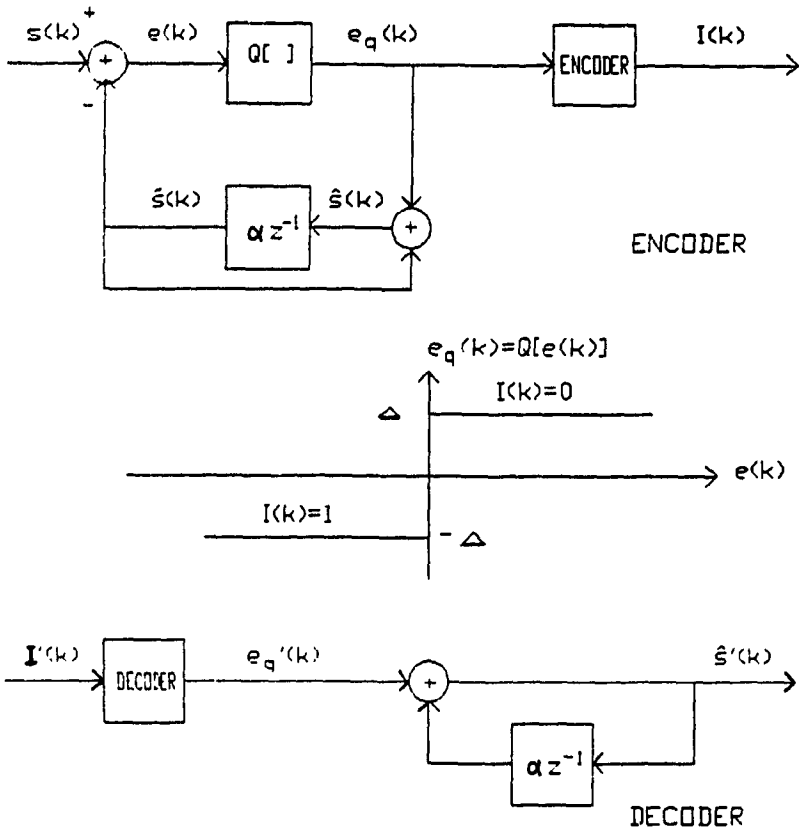


Fig.8. A DM system.

The simplest application of the concept of DPCM is in Delta Modulation (DM). In this scheme, the sampling rate is chosen to be many times of the Nyquist rate for the input signal. As a result, the correlation between adjacent samples is very close to one, and it is possible to use a one-bit quantizer to obtain a good approximation to the input samples. In this two-level quantizer, the difference signal $e(k)$ is quantized to only one of two values, a positive or a

negative one.

$$e_q(k) = \begin{cases} \Delta & \text{if } e(k) \geq 0 \\ -\Delta & \text{if } e(k) < 0 \end{cases}$$

The main advantage of DM is its simplicity. As only a one-bit code is required, no synchronization of bit patterns is required between transmitter and receiver. On the other hand, the limitations on the performance of DM systems stem mainly from the crude quantization of the difference signal.

Generally, the term DPCM is reserved for differential quantization systems in which the quantizer has more than two levels. And usually, the coder does not send the difference between adjacent samples, but instead uses the difference between a sample and its predicted value. With this refined approach, each successive coding compensates for the quantization error in the previous coding. Hence, the reconstructed signal can be prevented from drifting from the input signal. Besides, if a good predictor can be constructed, the difference waveform would have a lower dynamic range.

Fig.9 shows the block diagram of a DPCM system.

Generally, in linear prediction, a signal $s(k)$ is considered to be a linear function of past outputs, and present and past inputs :

$$s(k) = \sum_{i=1}^{n} a_i \, s(k-i) + G \sum_{j=0}^{m} b_j \, u(k-j) \qquad b_0 = 1 \qquad (5)$$

where a_i, $1 \leq i \leq n$; b_j, $1 \leq j \leq m$; and the gain G are the parameters of the system.

Taking the z-transform of $s(k)$ in Eq. (5) and letting $H(z)$ denote the transfer function of the system, we have :

$$H(z) \triangleq \frac{S(z)}{U(z)} = G \frac{1 + \sum_{j=1}^{m} b_j z^{-j}}{1 - \sum_{i=1}^{n} a_i z^{-i}} \qquad (6)$$

where $S(z) = \sum_{l=-\infty}^{\infty} s(l) z^{-l}$ is the z-transform of $s(k)$,

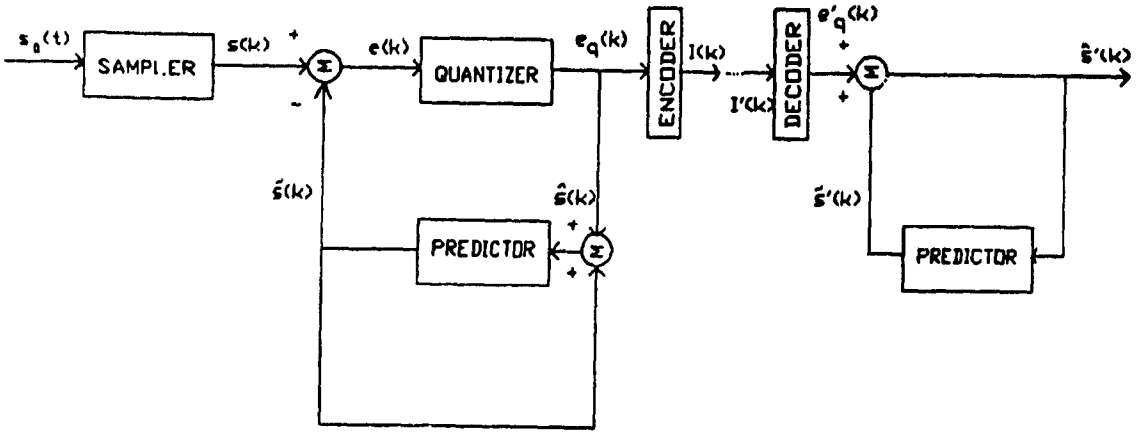


Fig.9. A DPCM system.

$U(z)$ is the z-transform of $u(k)$.

$H(z)$ in Eq. (6) is the general pole-zero model. The roots of the numerator and denominator polynomials are the zeros and poles of the model, respectively.

There are two special cases of the model :

1) all-zero model : $a_i = 0$, $1 \leq i \leq n$. This is known as the moving average (MA) model.

2) all-pole model : $b_j = 0$, $1 \leq j \leq m$. This is the autoregressive (AR) model.

The pole-zero model is known as the autoregressive moving average (ARMA) model.

Being described as simple, straightforward, inexpensive, and effective, the all-pole model has by far been the most widely used model [16] :

$$s(k) = \sum_{i=1}^{n} a_i \; s(k-i) + G \; u(k)$$

If we assume that the input $u(k)$ is totally unknown, which is the case in many applications, the signal $s(k)$ can be predicted only approximately from a linearly weighted summation of past samples. Let $\tilde{s}(k)$ denote the approximation of $s(k)$, we get :

$$\tilde{s}(k) = \sum_{i=1}^{n} a_i \; s(k-i)$$

The $\{a_i\}$ are the predictor coefficients or tap gains, and they are selected to minimize some function of the prediction error $e(k)$ between $s(k)$ and $\tilde{s}(k)$. Usually, we select $\{a_i\}$ to minimize the total squared error E :

$$E = \sum_{k} e^2(k) = \sum_{k} [s(k) - \sum_{i=1}^{n} a_i \; s(k-i)]^2 \qquad (7)$$

The range of the summation in Eq. (7) is of importance. For the moment, without specifying the range of the summation, let us first try to minimize Eq. (7) by differentiating with respect to $\{a_i\}$ and setting the derivatives equal to zero. We then obtain the normal equation :

$$\sum_{i=1}^{p} a_i \sum_{k} s(k-i) \; s(k-j) = \sum_{k} s(k) \; s(k-j) \qquad i \leq j \leq p \qquad (8)$$

For any definition of the signal $s(k)$, Eq. (8) forms a set of p equations with p unknowns which can be solved for the predictor coefficients $\{a_i, 1 \leq i \leq p\}$.

In the computation of $\{a_i\}$, we distinguish direct methods from iterative methods [16].

2.2.1. Direct Methods

Using analytical design technique, the direct methods would determine a set of linear equations and compute the predictor coefficients $\{a_i\}$ by solving the linear equations simultaneously.

2.2.1.1. Autocorrelation Method

In this method, Eq. (7) is minimized over the infinite duration $-\infty < k < \infty$. Eq. (8) then becomes :

$$\sum_{i=1}^{p} a_i \, R(j-i) = R(j) \qquad 1 \le j \le p \tag{9}$$

where $R(j) = R(-j) = \sum_{k=-\infty}^{\infty} s(k) \; s(k+j)$ is the autocorrelation function of the signal $s(k)$. Since the coefficients $R(j-i)$ form what often is known as an autocorrelation matrix, this method is called autocorrelation method. An autocorrelation matrix is a symmetric Toeplitz matrix where all the elements along each diagonal are equal.

2.2.1.2. Covariance Method

Eq. (7) is minimized over a finite interval, say, $0 \le k \le K-1$, so that Eq. (8) now is reduced to :

$$\sum_{i=1}^{p} a_i \, \varphi_{ij} = \varphi_{0j} \qquad 1 \le j \le p \tag{10}$$

where $\varphi_{ij} = \varphi_{ji} = \sum_{k=0}^{K-1} s(k-i) \; s(k-j)$ is the covariance of the signal $s(k)$ in the given interval. The coefficients φ_{ij} form a covariance matrix; Therefore, this method is called covariance method. The matrix is also symmetric.

Both methods, autocorrelation and covariance, lead to a set of p equations with p unknowns. There exist several standard methods to perform the necessary computations, e.g. the Gauss reduction or elimination method, and

the Crout reduction method which require $p^3/3 + O(p^2)$ operations and p^2 storage locations. Because of the symmetric characteristic of the coefficient matrices in both methods, Eq. (9) and (10) can be solved more efficiently by the square root or Cholesky decomposition method which requires about half the computation ($p^3/6 + O(p^2)$) and about half the storage ($p^2/2$) of the standard methods.

By exploiting the Toeplitz nature of the coefficient matrix in autocorrelation method, several efficient recursive procedures have been devised for solving the system of equations. The most popular and well-known procedures are the Levinson and Robinson algorithms. However, another method attributed to Durbin is the most efficient one which is twice as fast as Levinson's. This method requires only $p^2 + O(p)$ operations and $2p$ storage locations.

Durbin's recursive procedure can be specified as follows :

$$E^{(0)} = R(0)$$

$$\alpha_i = \frac{[R(i) - \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j)]}{E^{(i-1)}} \qquad 1 \le i \le p \qquad (11a)$$

$$a_i^{(i)} = \alpha_i \qquad (11b)$$

$$a_j^{(i)} = a_j^{(i-1)} + \alpha_i \; a_{i-j}^{(i-1)} \qquad 1 \le j \le i-1 \qquad (11c)$$

$$E^{(i)} = (1 - \alpha_i^2) \; E^{(i-1)} \qquad (11d)$$

Equations (11a) - (11d) are solved recursively for $i = 1, 2, \cdots, p$. The final solution is given by

$$a_j = a_j^{(p)} \qquad 1 \le j \le p$$

In obtaining the solution for a predictor of order p, one actually computes the solutions for all predictors of order less than p, i.e. $a_j^{(i)}$ is the j^{th} predictor coefficient for a predictor of order i.

2.2.1.3. Lattice Formulations and Solutions [10]

Both the autocorrelation and covariance methods consist of two steps :

1) Computation of a matrix of correlation values

2) Solution of a set of linear equations.

Lattice methods combine the above two steps into a recursive algorithm for determining the linear prediction coefficients.

Recall that $\{a_j^{(i)}, j = 1, 2, \cdots, i\}$ are the coefficients of the i^{th}-order linear predictor derived from the Durbin's algorithm. From Eq. (7), the prediction error is

$$e^{(i)}(k) = s(k) - \sum_{j=1}^{i} a_j^{(i)} s(k-j)$$

In terms of z-transform, we have :

$$E^{(i)}(z) = A^{(i)}(z)\ S(z) \tag{12}$$

$$\text{where} \qquad A^{(i)}(z) = 1 - \sum_{j=1}^{i} a_j^{(i)} z^{-j}$$

The recurrence formula for $A^{(i)}(z)$ in terms of $A^{(i-1)}(z)$ is given by :

$$A^{(i)}(z) = A^{(i-1)}(z) - K^{(i)}\ z^{-i}\ A^{(i-1)}(z^{-1}) \tag{13}$$

Then Eq. (12) can be rewritten as

$$E^{(i)}(z) = A^{(i-1)}(z)\ S(z) - K^{(i)}\ z^{-i}\ A^{(i-1)}(z^{-1})\ S(z)$$

Defining

$$B^{(i)}(z) = z^{-i}\ A^{(i)}(z^{-1})\ S(z) \tag{14}$$

and noting that

$$E^{(i-1)}(z) = A^{(i-1)}(z)\ S(z),$$

we obtain :

$$E^{(i)}(z) = E^{(i-1)}(z) - K^{(i)}\, B^{(i-1)}(z)\, z^{-1} \tag{15}$$

Substituting Eq. (13) in Eq. (14), we get :

$$B^{(i)}(z) = z^{-i}\, A^{(i-1)}(z^{-1})\, S(z) - K^{(i)}\, A^{(i-1)}(z)\, S(z)$$

or

$$B^{(i)}(z) = z^{-1}\, B^{(i-1)}(z) - K^{(i)}\, E^{(i-1)}(z) \tag{16}$$

In the time domain, Eq. (15) and (16) become

$$e^{(i)}(k) = e^{(i-1)}(k) - K^{(i)}\, b^{(i-1)}(k-1)$$

$$b^{(i)}(k) = b^{(i-1)}(k-1) - K^{(i)}\, e^{(i-1)}(k)$$

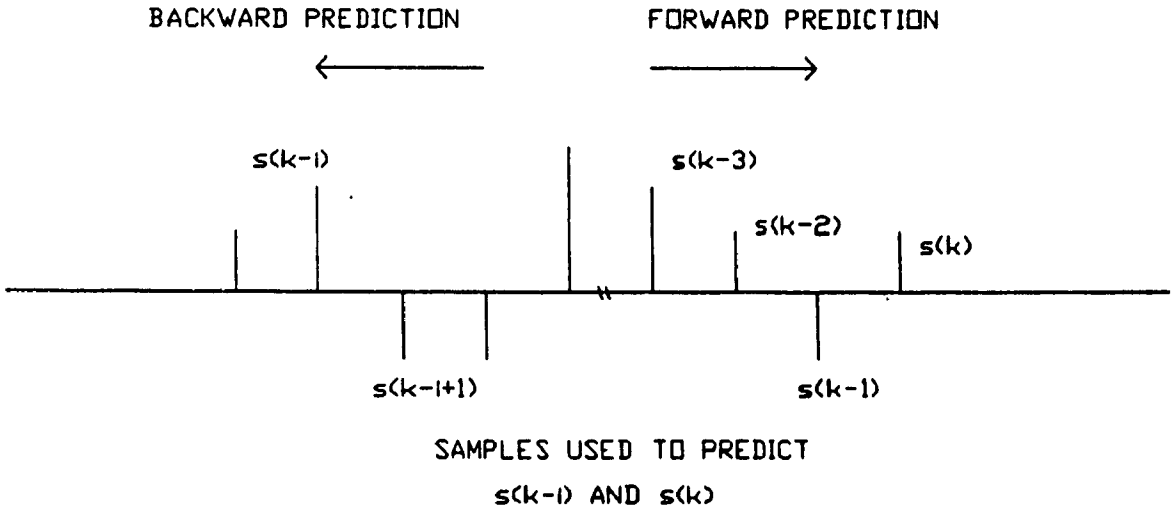


Fig.10. Forward and backward prediction using an i^{th}-order predictor.

$e^{(i)}(k)$ represents the error of an i^{th}-order forward predictor,i.e. the error which occurs when we try to predict $s(k)$ from the i samples of the input $\{s(k-i+j), j = 1, 2, \cdots, i\}$. $b^{(i)}(k)$ represents the error of an i^{th}-order backward predictor, or in other words, the error occurring when we attempt to predict $s(k-i)$ from the same i samples.

The flow graph of Fig.11 depicts the structure of a lattice network.

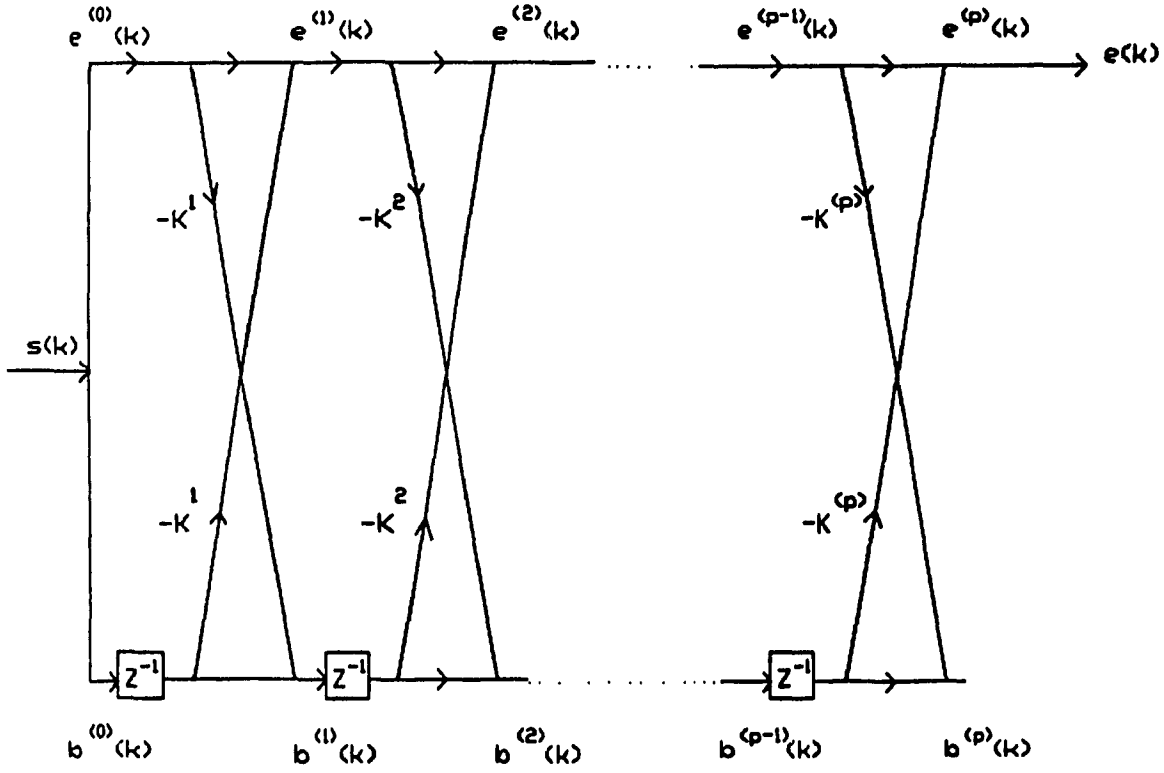


Fig.11.A lattice network.

The parameters $K^{(i)}$ are called partial correlation coefficients or PARCOR coefficients and are directly related to the forward and backward prediction errors :

$$K^{(i)} = \frac{\sum_{k=0}^{N-1} e^{(i-1)}(k)\, b^{(i-1)}(k-1)}{\left\{\sum_{k=0}^{N-1}(e^{(i-1)}(k))^2 \sum_{k=0}^{N-1}(b^{(i-1)}(k-1))^2\right\}^{1/2}} \qquad (17)$$

If Eq. (17) replaces Eq. (11a) in the Durbin algorithm, the predictor coefficients can be computed recursively.

The lattice formulation has become an important and viable approach in linear predictive analysis as it guarantees a stable filter. Moreover, it yields the predictor coefficients directly from the signal samples without an intermediate calculation of an autocorrelation function.

2.2.2. Iterative Methods

In these methods, one begins by an initial guess for the solution. The solution is then updated by adding a correction term that is usually based on the gradient of some error criterion. In general, iterative methods require more computation to achieve a desired degree of convergence than the direct methods. Nevertheless, in some applications, one often has a good initial guess, which might lead to the solution in only a few iterations. This can be a big saving over direct methods if the number of equations is large. Some of the iterative methods are the gradient method, the steepest descent method, Newton's method, conjugate gradient method, and the stochastic approximation method [16].

However the coefficients are calculated, the SNR gain of DPCM systems depends on the number p of predictor coefficients used. Noll [3] has computed the optimum gain $(G_p)_{opt}$ as a function of p for a 55-s speech sample that is either low-pass-filtered (LPF : 0 to 3400 Hz) or band-pass-filtered (BPF : 300 to 3400 Hz). The sampling rate in either case is 8 kHz. The results are reproduced in Fig.12. The shaded region shows the amount of variation obtained for four speakers, with the central curve representing the average over four speakers. It can be observed that the use of even the simplest DPCM logic should provide an SNR gain over PCM where $p = 0$. In fact, it is possible to

realize about a 6 dB improvement in SNR with the simplest predictor $p = 1$. This implies that a DPCM system can achieve a given SNR using one less bit than would be required when using the same quantizer directly on the input waveform. However, $(G_p)_{opt}$ never reaches a value of 12 dB which would signify a 2-bit advantage over PCM. According to Fig.12, the $(G_p)_{opt}$ function typically saturates for all practical purposes at $p = 2$. It should also be noted that a fixed predictor cannot be optimum for all speakers and for all speech material.

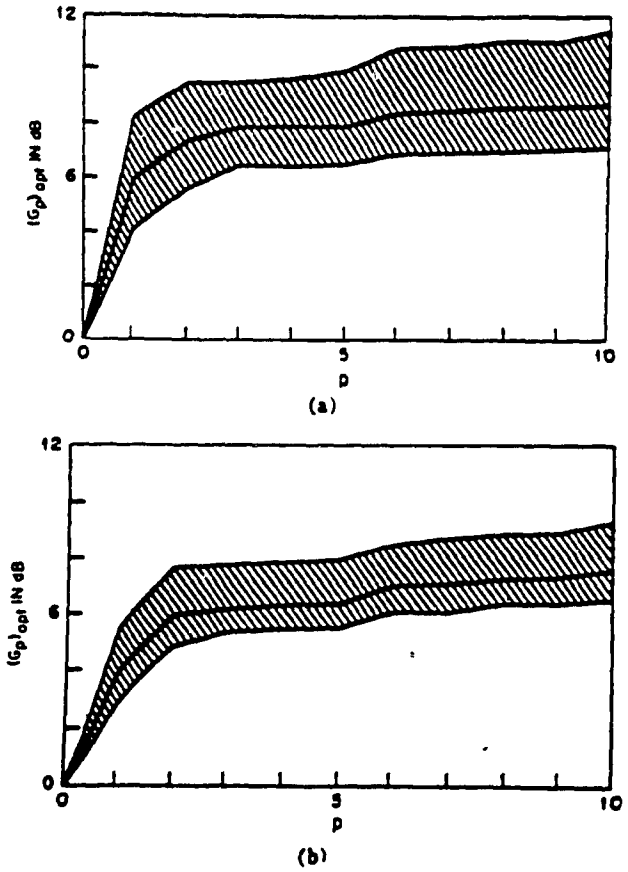


Fig.12. Optimum SNR gain G versus number of predictor coefficients. (a) Lowpass filtered speech (b) Bandpass filtered speech (After Noll [3].)

CHAPTER III

ADAPTIVE SCHEMES IN DPCM

Many real sources are quasistationary in nature. One aspect of the quasistationary characteristic is that the variance and the autocorrelation function of the source output vary slowly with time. Considerable improvement in efficiency and performance can be obtained in DPCM coders if the quantizer, or both the quantizer and the predictor adapt themselves to match the signal to be coded. Coders with these adaptive features have been referred to in the literature as Adaptive DPCM (ADPCM) coders. The general block diagram of an ADPCM system is similar to that of DPCM in Fig.9, except that the adaptive quantizer and/or the adaptive predictor replace the fixed quantizer and/or the fixed predictor, respectively.

3.1. Adaptive Quantizer

3.1.1. Adaptive Quantization Techniques

The primary purpose of adaptive quantization is to maintain a nearly optimum step size over a wide variety of input signal conditions. This can be realized by varying either the step size or the gain of the quantizer. In the first case, the quantizer step size $\Delta(k)$ should increase and decrease with increases and decreases of the input, or in other words, it is scaled linearly to match the variance of the signal. In the second case, the gain $G(k)$ changes inversely with changes in the variance of the input so as to keep the variance of the quantizer input relatively constant. In either case, the dynamic range is extended, and the coder can operate well with a wide range of signal levels.

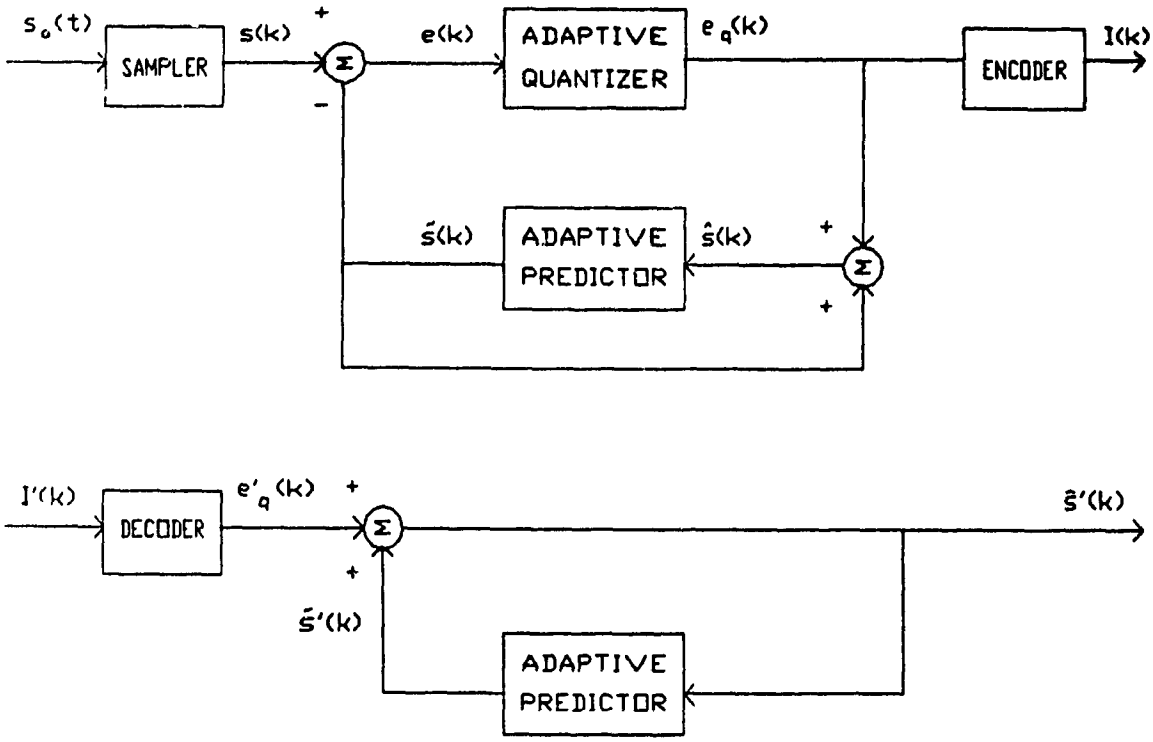


Fig.13. An ADPCM system.

This leads to some improvement in the performance of the system at any given signal level, and dramatically attenuates idle channel noise [17].

3.1.1.1. Instantaneous versus Syllabic Adaptation

From the time-scale point of view, adaptive quantization is of two types : instantaneous and syllabic.

An instantaneously adaptive quantizer relies on short-term statistics of the input, and changes either its step size or its gain continuously from sample to sample, or within a few samples. This is a fast-acting approach.

Based on long-term variations of the input, a syllabic adaptive quantizer varies the step size or the gain slowly, usually at relatively long time intervals.

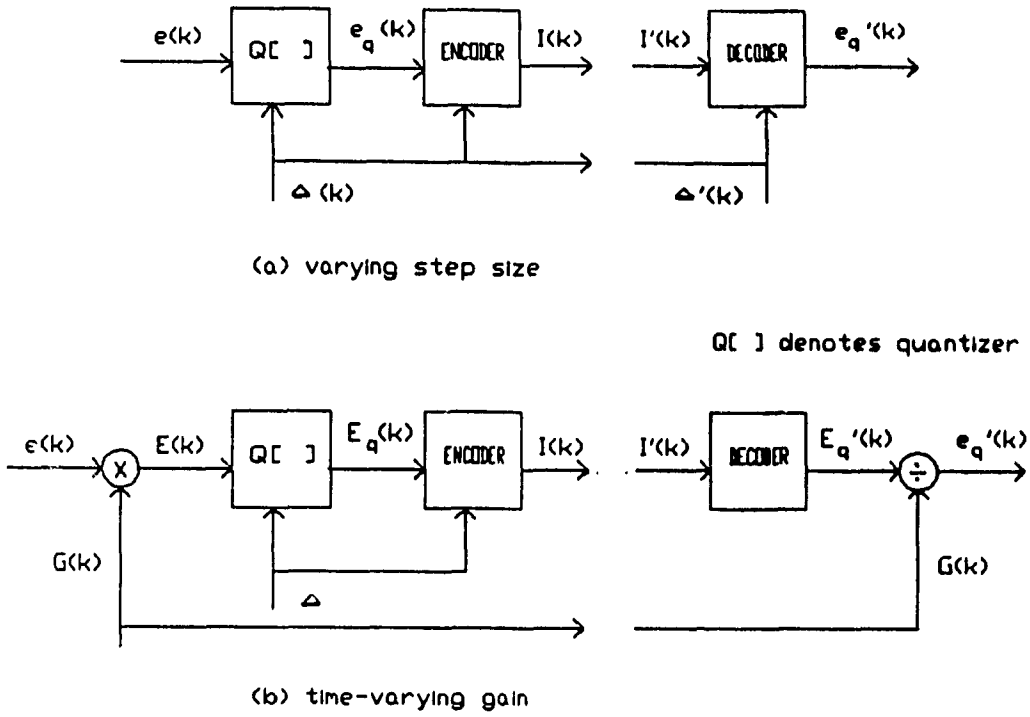
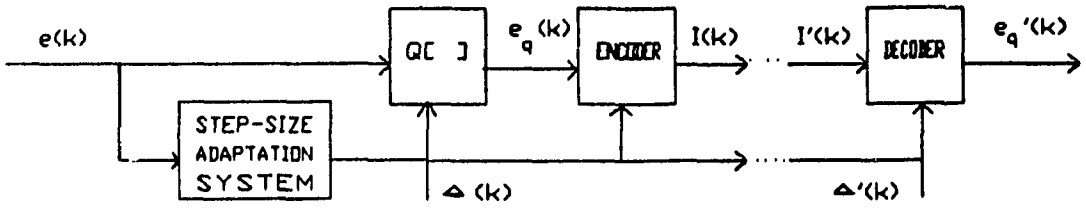


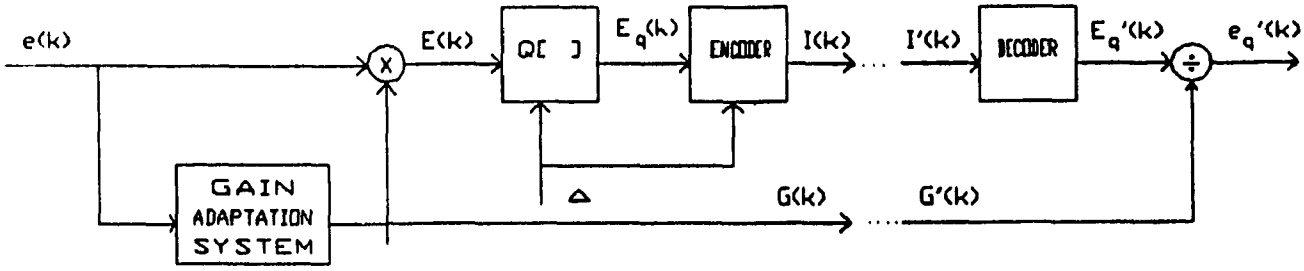
Fig.14. Adaptive quantizer.

Although instantaneous quantizers have good attack and decay times, they, unlike the slower syllabic adaptive systems, are prone to a little more quantizing noise [18].



(a) varying step size

Q[ ] denotes quantizer



(b) time-varying gain

Fig.15. Feed-forward adaptive quantizer.

3.1.1.2. Forward versus Backward Adaptation

A convenient classification of quantizers that has system mechanization implications is to separate quantizers as to whether they are forward adaptive or backward adaptive [17].

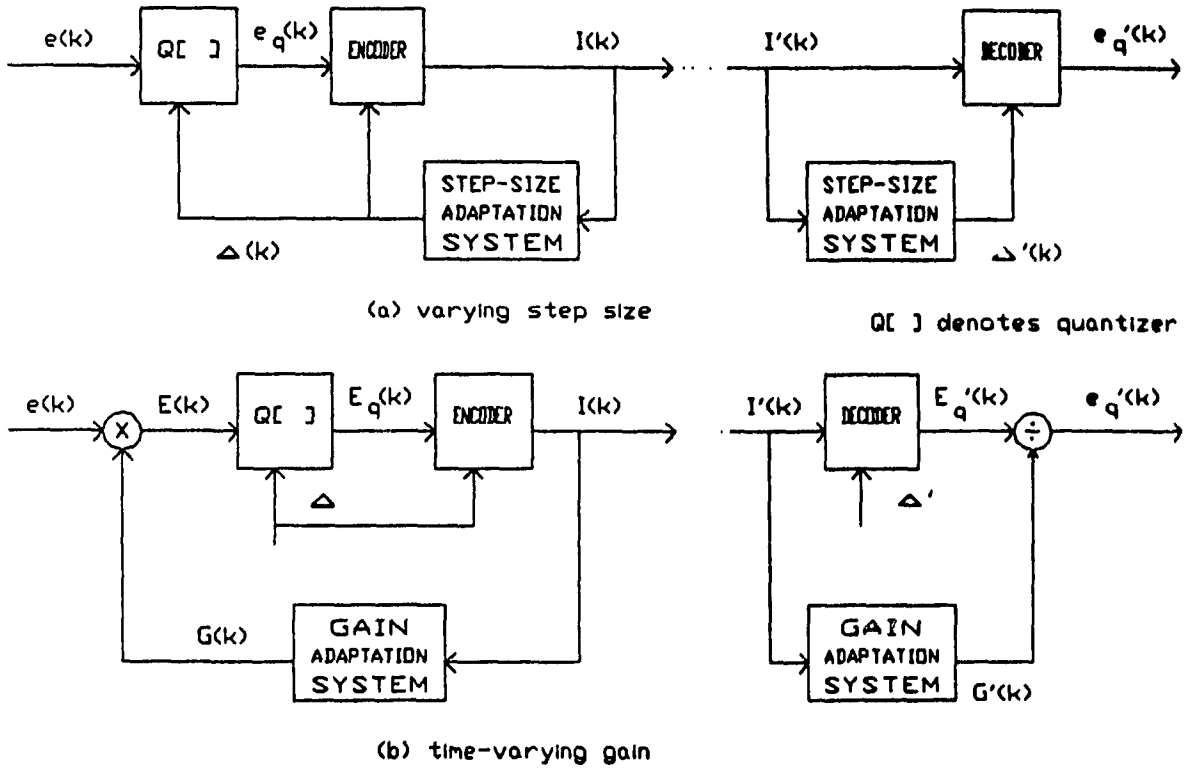


Fig.16. Feedback adaptive quantizer.

In forward adaptive quantizers, the variance of the input is estimated from the quantizer input to determine the step size or the gain. As the quantizer input is not available at the receiver side, the step size or the gain information has to be transmitted.

Backward adaptive quantizers adapt the step size or the gain basing on the variance of the input estimated from the quantized error signal. No additional information needs to be sent to the receiver.

The backward adaptation scheme has the advantage that the step size or the gain needs not be explicitly retained or transmitted since they can be derived from the sequence of codewords. The disadvantage of such system is increased sensitivity to errors in the codewords, since such errors imply not only an error in the quantizer level but also in the step size.

Noll [3] has studied several quantization schemes for speech encoding. Comparing PCM systems having logarithmic quantizer with coders using other algorithms, he derived G^*, the gain in SNR, over PCM.

For fixed predictor, fixed quantizer, $G^* \approx 7$ dB.

For fixed predictor, adaptive quantizer, $G^* \approx 12$ dB.

These would imply a gain in SNR of 5 dB of adaptive quantization over fixed quantization.

3.1.2. Adaptive Quantization Algorithms

In "A Switched Quantizer for Markov Sources Applied to Speech Signals", Cohen [19] described a statistic sensitive quantizer used in a system which exploits the intersample dependencies of speech signal. Assuming an m^{th}-order Markov process for the input signal $e(k)$, the conditional probability density distribution of the quantizer input can be written as :

$$p(e(k)|e(k-1),\cdots,e(1)) = p(e(k)|e(k-1),\cdots,e(k-m)) = p(e(k)|S)$$

where the state S is defined by the m preceding samples. For each state S of the Markov process, we could have conditional distributions as in Fig.17.

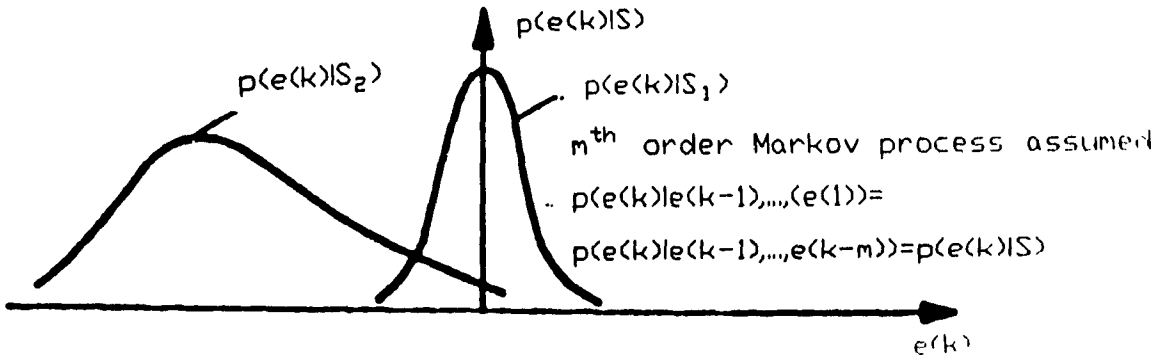


Fig. 17. Conditional distributions.

Less quantizing noise would occur if the quantizer matches its characteristic to each of these conditional distributions. In order to do this, the quantizer must have memory to know which state the Markov source is in. Cohen called such a quantizer which switches to a characteristic matching to the input conditional distribution a switched quantizer. According to Cohen, the switched quantizer, when matching to a particular distribution, forms its characteristic in two stages. First, there is a shifting of the quantizer steps along the $e(k)$-axis so that the steps of the quantizer are placed on the distribution to be quantized. Secondly, the step widths of the quantizer are adjusted so that each step gives the same amount of quantizing noise. A complete switched quantizer system is shown in Fig.18.

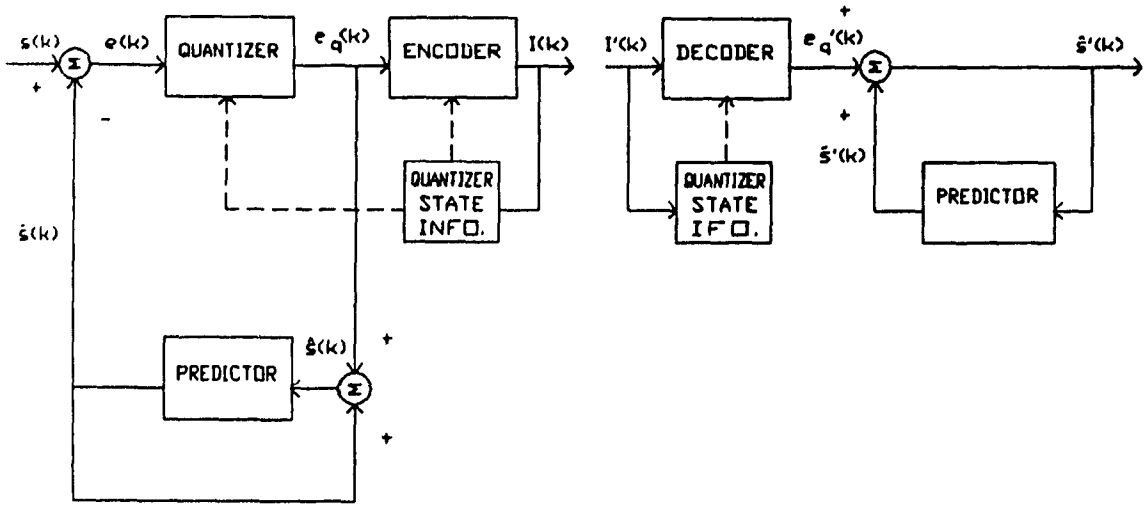


Fig.18. A switched quantizer system.

The DPCM system is used to accomplish the horizontal shifting of the quantizer , and additional state information is fed to the quantizer to determine how the quantizer step widths are to be adjusted.

A computer simulation of the system was carried out with an input of 60 seconds of 300 to 3400 Hz band-limited speech, 12-bit linearly quantized, with three feedback loops for the DPCM predictor and a 64-level switched quantizer. The switched quantizer state information was constrained to 6 bits, and the best 6 bits to use were determined to be the two most significant bits from the three previous quantized prediction error samples $I(k-1)$, $I(k-2)$, $I(k-3)$. No additional information is required to be transmitted for decoding purposes. The

theoretical gain in SNR with this system over that of a conventional DPCM system using a log quantizer is 7 dB.

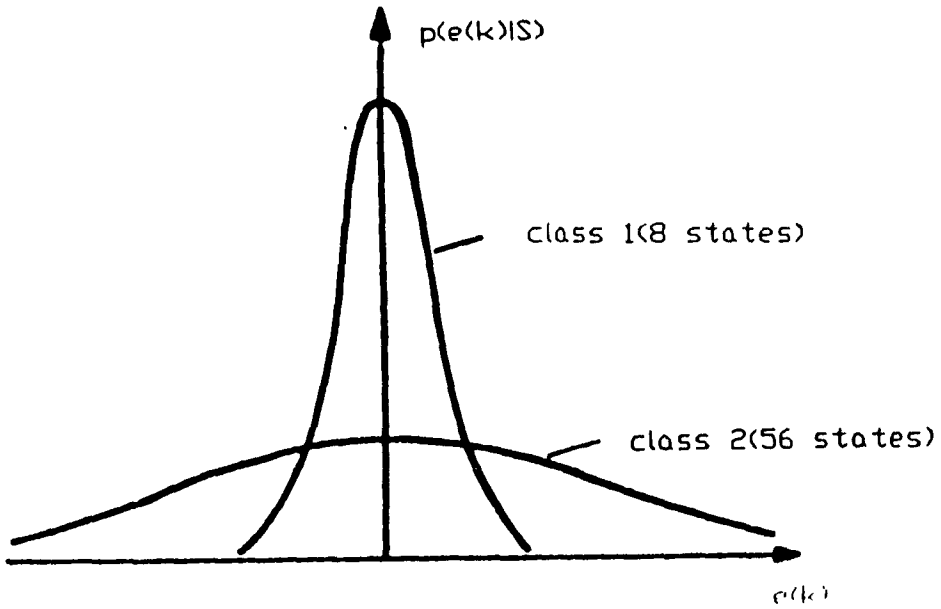


Fig.19. Representative distribution from the two classes containing the 64 $p(e(k)|S)$ densities.

Observing that the shapes of the 64 conditional distributions $p(e(k)|S)$ fall into two general classes as shown in Fig.19, Cohen suggested that two composite conditional distributions $p(e(k)|S_1, \cdots, S_8)$, and $p(e(k)|S_9, \cdots, S_{64})$ be calculated so that a system using only two quantizing characteristics for the switched quantizer can be realized and achieve most of the 7 dB gain.

In their 60-channel PCM-ADPCM converter, Cointot and De Passoz [20] used an automatic gain control (AGC) and a switched quantizer. The switched

quantizer used is as described by Cohen, and three characteristics were selected from the conditional probability distribution of $\hat{e}(k)$ for three system states.

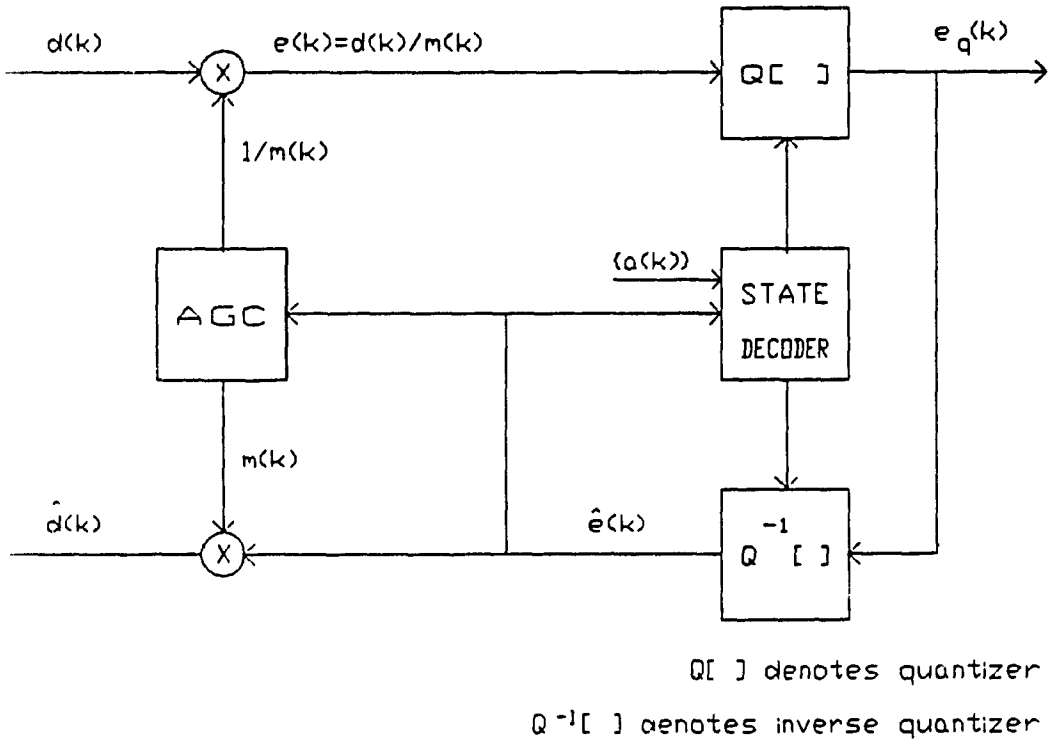


Fig.20. Quantizer stage of Cointot and De Passoz's coder.

In this system, the quantizer input $e(k)$ is derived from $d(k)$, the difference between the actual system input $s(k)$ and its predicted value $\tilde{s}(k)$, and the compression ratio $m(k)$. The AGC is to provide the value of $m(k)$ through the formula :

$$m(k+1) = \{(1-K)\, m(k) + K\, C|\hat{e}(k)|\}^{\gamma}$$

where K is a parameter establishing the AGC time constant ($K = 31/32$),
C is a constant determined by the variance of $\hat{e}(k)$ at the output of the inverse quantizer,

γ allows a memory loss for protection against errors (γ=127/128).

As $e(k)$ depends on $m(k)$, the AGC controls the step size adaptation.

As mentioned previously, three system states are defined in the switched quantizer. C_1 and C_2 are optimized characteristics for the speech signal. C_1 represents the prevailing state S_1; and C_2, the fast transients and hardly predictable non-voiced periods of speech (approximately 10% of the speech time). C_1 and C_2 are selected at time k by examining the values of $\hat{e}(k)$ according to the following criterion :

$$\left.\begin{array}{ll} & |\hat{e}(k-1)| < \hat{e}_{max}/4 \\ \text{and} & |\hat{e}(k-2)| < \hat{e}_{max}/2 \end{array}\right\} \quad \text{then} \quad C_1 \quad \text{otherwise} \quad C_2.$$

C_3 is a quantizing characteristic selected for 4800 bits/s modem signals that have an amplitude distribution (S_3) which is quite different from that of speech signals.

Speech/data discrimination is performed by examining the predictor coefficients. Through a careful selection of coefficient threshold levels, it is possible for the system to switch to the optimum quantizer.

Viewing DPCM and AGC as special solution of the switched quantizer, Dietrich [21] designed a switched quantizer system which is a cascade of three stages.

In stage I, DPCM estimates $\tilde{s}$ for the signal s through time-invariant linear prediction from two preceding transmitted and reconstructed samples $\hat{s}$. The difference $d = s - \tilde{s}$ is fed to stage II which is a digital version of the AGC. This stage ensures the required dynamic range of the coding system. Power variations of the speech signal s and of the difference d are compensated by dividing d by a power estimate m, the state information. Four source states are distinguished. The values of m are calculated by forming a weighted mean of

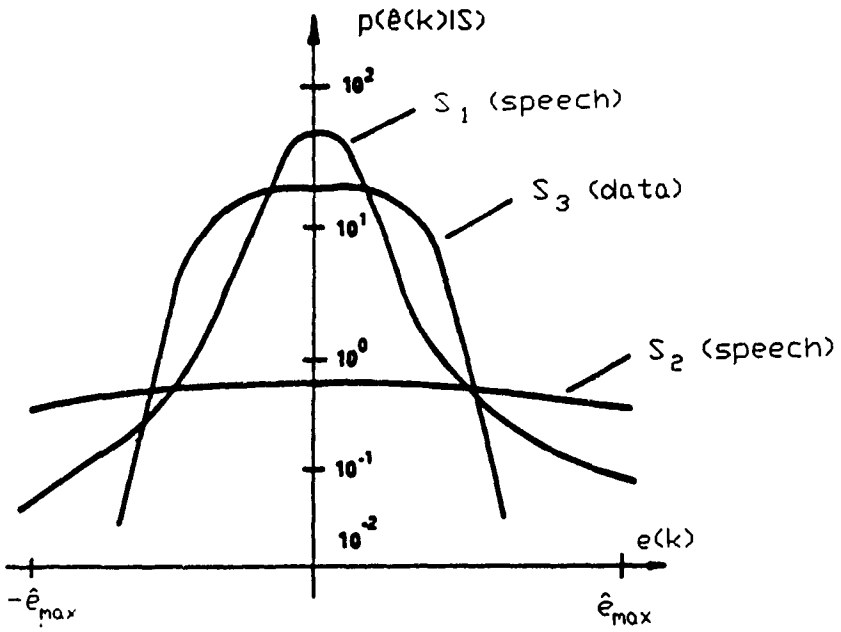


Fig.21. $\hat{e}(k)$ amplitude distribution and quantization curves.

the reconstructed difference signal $\hat{d}$. For simplicity, Dietrich chose m as powers of 2 : $m_i = 2^{-i}, i = 0, \cdots, 3$.

The output c of stage II is passed through a switched quantizer, the last stage of the cascade. As the statistical properties of the input signal e show a fundamental change depending on whether there is active speech at the system input or not, the criterion for quantizing characteristic switching is designed with regard to the case of absence of speech, which is indicated by the minimum signal power state m_3 of the AGC stage. To process the present sample, $e(k)$, three source states S_a, S_b, S_c are calculated from two preceding transmitted and reconstructed samples. If $\hat{e}(k-2)$ and $\hat{e}(k-1)$ have been both within fixed limits, the source is in the dominant state S_a. If not, the polarity of the

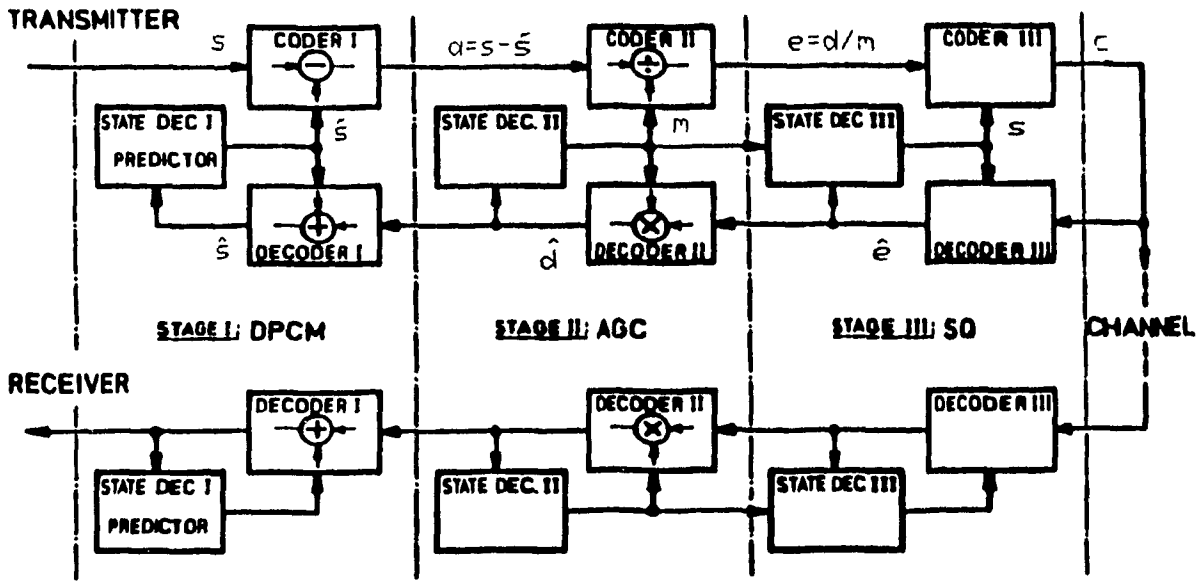


Fig.22. The switched quantizer as a cascade of three stages.

difference $\Delta\hat{e} = \hat{e}(k-1) - \hat{e}(k-2)$ takes a decision about either state S_b ($\Delta\hat{e} > 0$) or state S_c ($\Delta\hat{e} < 0$). For the quantizing characteristic switching, a combination of the state information m_3 together with S_a, S_b, and S_c is used. Five source states are distincted in this stage of the system :

$$S_1 = S_a.\overline{m}_3$$

$$S_2 = S_b.\overline{m}_3$$

$$S_3 = S_c.\overline{m}_3$$

$$S_4 = S_a.m_3$$

$$S_5 = \overline{S}_a.m_3$$

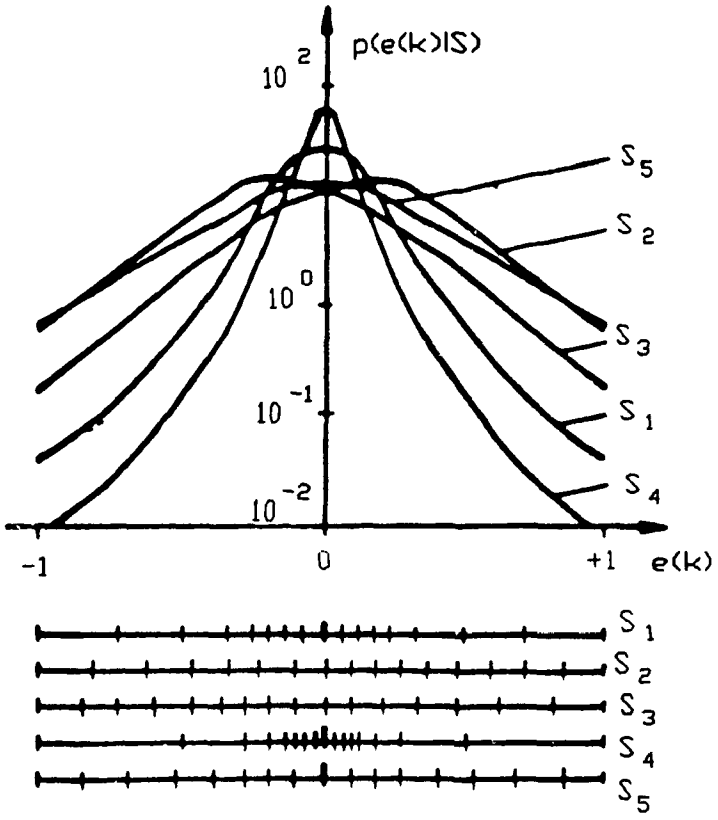


Fig.23. Signal probability and quantizing characteristics for source states.

Fig.23 shows the measured conditional probability distributions of these source states and the corresponding decision levels.

Compared to a PCM system of the same number bit rate (4 bits), this system yields a gain of 19 dB. According to Dietrich, its performance at 32 kbits/s can be compared with that of a 64 kbits/s PCM.

Noll [22] has described several adaptive quantization algorithms, where a local estimate $\hat{G}^2$ of the input variance is calculated. This estimation value controls the gain of an amplifier which is followed by a quantizer optimized for signals with zero mean and unit variance.

In the scheme of forward estimation, segments of NS speech samples are considered locally stationary. The samples are buffered, and the estimation

value

$$\hat{G}^2 = \frac{1}{NS} \sum_{k=1}^{NS} e^2(k)$$

is calculated leading to the optimum gain $G = \hat{G}^{-1}$ of the amplifier.

If the segment is short ($NS < 128$), the probability density function (PDF) is nearly Gaussian, while the PDF tends to become a Laplacian density for $NS > 512$.

For backward estimation, each input sample $e(k)$ is amplified with the gain factor $G(k) = \hat{G}^{-1}(k)$ which is calculated from quantized samples $e_q(k-j), j = 1, 2, \cdots$.

In algorithm BE1, the last N quantized samples are used for the computation of $\hat{G}(k)$:

$$\hat{G}^2(k) = C \frac{1}{N} \sum_{j=1}^{N} e_q^2(k-j)$$

The constant C is optimized to get an unbiased estimator.

Algorithm BE2 is a modified version of BE1 : the last N samples are weighted. Only small improvements are obtained.

In algorithm BE3, the gain of the amplifier is changed if the smallest or the largest reconstruction level appears once or more than once.

$$\hat{G}(k) = a.\hat{G}(m) \qquad if\ |J(k-j)| = J_{max} \qquad for\ j = 1, 2, \cdots, NA$$

$$\hat{G}(k) = b.\hat{G}(m) \qquad if\ |J(k-j)| = 1 \qquad for\ j = 1, 2, \cdots, NB$$

$$\hat{G}(k) = \hat{G}(m) \qquad otherwise$$

$J(k-j)$ is the index of the occupied quantizer step at the time $k-j$. The smallest and largest reconstruction levels are 1 and J_{max}, respectively. For a B-bit quantizer, $J_{max} = 2^{B-1}$. The factors a, b, NA, and NB are to be optimized to get a high SNR value. Noll has used $a = 2$, and $b = 0.75$. A simple solution is given for $NA = NB = 1$. Then, the gain $G(k) = \hat{G}^{-1}(k)$ is readjusted whenever the smallest or the largest reconstruction level appears.

For algorithm BE4,

$$\hat{G}(k) = a_{|J(k-1)|}\, \hat{G}(k-1).$$

In other words, the last estimation value of the standard deviation is multiplied with a factor that depends on the index of the last occupied quantizer step. Noll derived the optimum multiplier set 0.8, 0.8, 1.3, 1.9 for a 3-bit quantizer.

As a matter of fact, algorithm similar to algorithm BE4 has been proposed by Cummiskey, Jayant, and Flanagan [23] to adapt the step size of a quantizer. In their system, the input samples, instead of being amplified, are fed directly to the quantizer the step size of which is adapted. The empirical adaptation rule is that, for every new input sample, the step size is changed by a factor depending only on the knowledge of which quantizer slot was occupied by the previous signal sample. Meaningful adaptation requires that the step size be increased on the detection of quantizer overload, and decreased during underload.

The specific quantizer configuration under consideration is characterized by a uniform spacing of non-zero output levels. Fig.24 shows the quantizer at sampling instant k for $B = 3$.

If the outputs of the B-bit quantizer ($B > 1$) are of the form :

$$e_q(k) = I(k)\,\frac{\Delta(k)}{2}; \qquad I(k) = \pm 1, 3, \cdots, 2^B - 1 \qquad \Delta(k) > 0,$$

the step size $\Delta(k)$ is given by the previous step size multiplied by a time-invariant function of the previous codeword magnitude $|I(k-1)|$

$$\Delta(k) = \Delta(k-1)\; M(|I(k-1)|)$$

It can be observed that the entire quantizer is "accordioned in" when $M < 1$, and stretched out when $M > 1$.

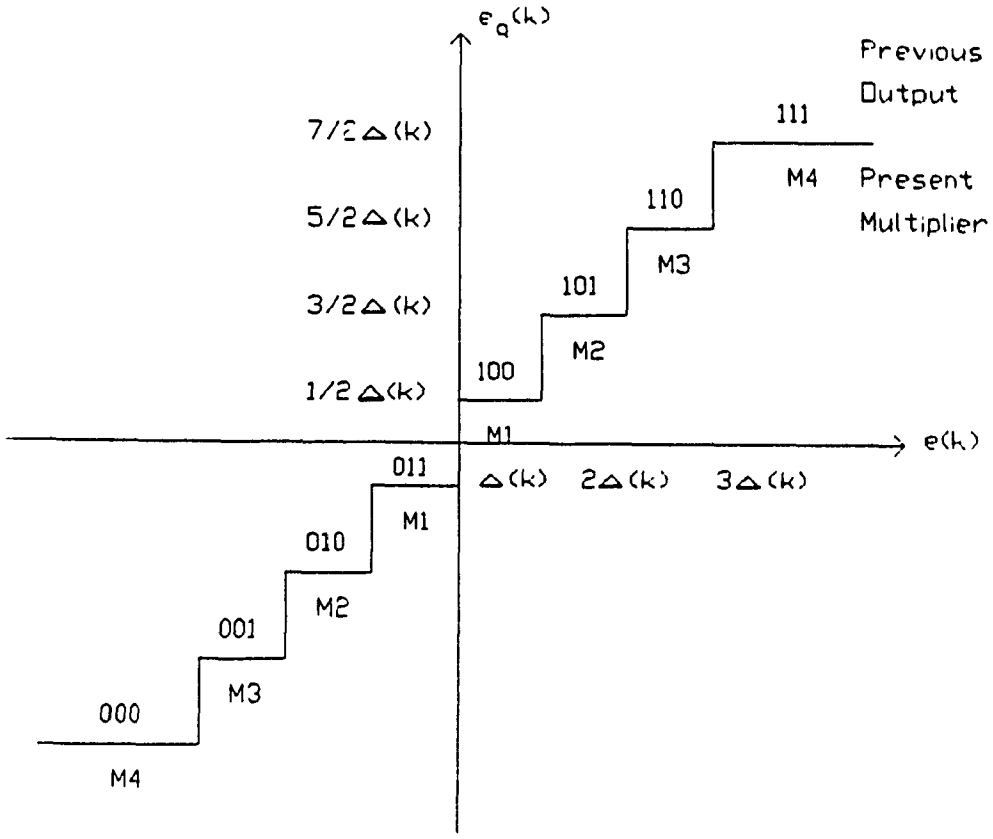


Fig.24. Uniform quantizer with 8 levels (B =3).

According to Jayant [24], the adaptation strategy represents a very simple, yet non-trivial, type of exponential adaptation. Moreover, the step size multiplier function $M(|I|)$ has the property that it demands step size decreases significantly slower than step size increases. This can be explained as follows. Quantization errors during overload tend to be more harmful than those during granularity in that the magnitude of granular error is restricted, by definition, to a half step size, while no such simple constraint exists for an overload error. Consequently, step sizes should be decreased relatively slowly to avoid unduly small step sizes leading to the harmful overload errors. The optimum multipliers

$M(k)^{OPT}$ have been derived theoretically for two cases :

$$M(k)^{OPT} = \left[\frac{1}{2} + \frac{K^2}{8} I^2(k-1)\right]^{1/2} + \delta^2(|I(k-1)|); \qquad C = 0$$

$$M(k)^{OPT} = \frac{|I(k-1)|}{2^{B-1}} + \delta^2(|I(k-1)|); \qquad C \to 1$$

where C is the correlation between adjacent samples,

$M(k)^{OPT}$ is the optimum multiplier at time k,

$I(k-1)$ is the magnitude of the codeword at time $k-1$,

δ^2 is a positive correction that is significant only for the last slot $|I(k-1)| = 2^B - 1$,

the constant K is a function of the number of quantizer levels and, hence, of B. Max's table II [13] specifies the values for $K(B)$.

Jayant [24] has tabulated step size multipliers found by search procedure for B = 2,3,4, and 5 for both PCM and DPCM coders.

Using adaptive quantizers as described above, usually referred to as Jayant quantizers, Jayant revealed that adaptive quantization, as incorporated into PCM, has the potential of outperforming the conventional technique of logarithmic companding. The advantages over log PCM of ADPCM have been reported by Cummiskey, Jayant, and Flanagan [23].

Indeed, several researchers have incorporated Jayant quantizer in their systems. In some quantizers, the multipliers used are those derived by Jayant, as reported by Gibson, Jones, and Melsa [25], Gibson and Cross [26], Gibson [27], Gibson [28], Gibson, Berglund, and Sauter [29], Un and Cynn [5], Evci, Xydeas, and Steele [30], Ramamoorthy and Jayant [31], Sayood [32]. In others, some modifications or adjustments are added, e.g. Fukasawa, Hosoda, Miyamoto, and Sugihara [33] used multiplication constants proposed by Jayant for speech

B	Coder Type	
	PCM	ADPCM
2	0.6, 2.2	0.8, 1.6
3	0.85, 1, 1, 1.5	0.9, 0.9, 1.25, 1.75
4	0.8, 0.8, 0.8, 0.8, 1.2, 1.6, 2.0, 2.4	0.9, 0.9, 0.9, 0.9, 1.2, 1.6, 2.0, 2.4
5	0.85, 0.85, 0.85, 0.85, 0.85, 0.85, 0.85, 0.85, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6	0.9, 0.9, 0.9, 0.9, 0.95, 0.95, 0.95, 0.95, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3.0, 3.3

Table I. Step size multipliers for speech signals.

signals, and derived values of the multiplication constants Md_i for voiceband data (VBD) signals according to the formula :

$$\sum_i P_i(x) \log Md_i = 0$$

where $P_i(x)$ is the probability of selecting Md_i against the range-to-signal ratio x.

According to the authors, by using optimum multiplication constants selected according to the signal identification result between speech and VBD signals, the adaptive quantizer could operate under different conditions. Excellent performance was reported to be attained for speech and VBD signals.

Nevertheless, Jayant quantizer has been shown to be sensitive to transmission errors. Using one-word memory, Jayant proposed :

$$\Delta(k) = \Delta(k-1)\, M(|I(k-1)|)$$

However, as pointed out by Goodman and Wilkinson [34], in the information theory sense, the memory is infinite : $\Delta(k)$ depends on the entire past of

the signal sequence :

$$\Delta(k) = \prod_{m=0}^{k-1} \Delta(0)\, M(|I(m)|)$$

Let Δ' and I' be the decoder version of Δ and I, respectively, and assume that at time $l < k$, $|I(l)| = i$ while a transmission error causes $|I(l)| = j$. And if there are no other errors, at the receiver side we have :

$$\Delta'(k) = \frac{M(j)}{M(i)}\, \Delta(k)$$

Each error causes a multiplicative offset between receiver and transmitter that can persist indefinitely.

Goodman and Wilkinson recommended an adaptation procedure in which the effect of a transmission error diminishes with time :

$$\begin{aligned} \Delta(k) &= \Delta^{\beta}(k-1)\, M(|I(k-1)|) \\ &= \prod_{m=0}^{k-1} \Delta^{\beta^k}(0) \Big[M(|I(m)|) \Big]^{\beta^{(k-m-1)}} \end{aligned} \tag{18}$$

If at time l, j is received instead of i,

$$\frac{\Delta'(k)}{\Delta(k)} = \left[\frac{M(j)}{M(i)} \right]^{\beta^{(k-l-1)}}$$

When $\beta < 1$, the offset due to each error decays exponentionally with time, causing the quantizer to be more robust in the presence of transmission errors than a quantizer operating according to Jayant algorithm, where implicitly $\beta = 1$. This error-dissipating mechanism, nevertheless, degrades to some extent the performance of a coder operating with an error-free channel. Theoretically, the degradation was shown to be limited. This kind of robust adaptive quantizer has found itself in systems reported by Gibson and Berglund [35], Moore and Gibson [36], Reininger and Gibson [37]. Kim and Un [38] used a table look-up method to implement the quantizer digitally. The possible step sizes are stored in consecutive read only-memory (ROM) locations in increasing order. Instead

of calculating the step size as given in Eq. (18) at each sampling instant, the address of ROM which indicates the memory location of corresponding step size is changed. In this manner, the calculation required for determination of step size is simplified. The authors have derived the address adaptation logic as follows :

$$ADR(i+1) = [\beta ADR(i) + m_l]$$

where [.] represents truncation,

ADR(i+1) and ADR(i) are addresses of ROM,

β is the leak factor proposed by Goodman and Wilkinson. It was chosen to be 127/128,

$m_l \triangleq \log_Q M(|I_l|)$ is the log multiplier with the base Q =1.055645, $1 \leq l \leq 2^{B-1}$. Values of the multipliers and their corresponding log transforms are listed in Table II.

Symbol	Value	$\log_Q M_l$
$M_1 - M_4$	0.9	-1.9
M_5	1.2	3.4
M_6	1.6	8.7
M_7	2.0	12.8
M_8	2.4	16.2

Table II. Log transformed multiplier.

Scagliola [39] evaluated ADPCM coders under noisy channel conditions. The system used in the experiment has the step size $\Delta(k)$ adapted according

to Goodman and Wilkinson's robust algorithm. The multipliers M_i are related by a linear relationship :

$$M_i = [\alpha + C\ (1-\alpha)\ (i-0.5)]\ \widehat{\Delta}^{(1-\beta)} \qquad 1 \leq i \leq 2^{B-1}$$

where α is a parameter controlling the speed of adaptation,

C is a parameter determining the mixture of granular noise and clipping distortion in the decoded signal at the nominal input level,

$\widehat{\Delta}$ is the step size that gives optimum performance at the desired nominal input level,

β is the decay constant as specified in Eq. (18).

Castellino, Modena, Nebbia, and Scagliola [40] studied a quantizer having the step size $\Delta(k)$ adapted proportionally to a short-time estimate $\sigma_{e_d}(k)$ of the rms of the decoded prediction error $e_d(k)$. The estimate $\sigma_{e_d}(k)$ is performed by two different algorithms :

1) The estimate $\sigma_{e_d}(k)$ is an exponential average of the magnitude of the past samples :

$$\sigma_{e_d}(k) = (1-2^{-t})\ \sigma_{e_d}(k-1) + 2^{-t}\ |e_d(k-1)| \tag{19}$$

where $e_d(k)$ and $e_d(k-1)$ are decoded prediction errors,

t is related to the time constant τ and the sampling period T :

$\tau = 2^t T$,

The step size

$$\Delta(k) = C\sigma_{e_d}(k) \tag{20}$$

2) The estimate is the square root of an exponential average of the squared past samples :

$$\sigma^2_{e_d}(k) = (1-2^{-t^*})\sigma^2_{e_d}(k-1) + 2^{-t^*} e^2_d(k-1) \tag{21}$$

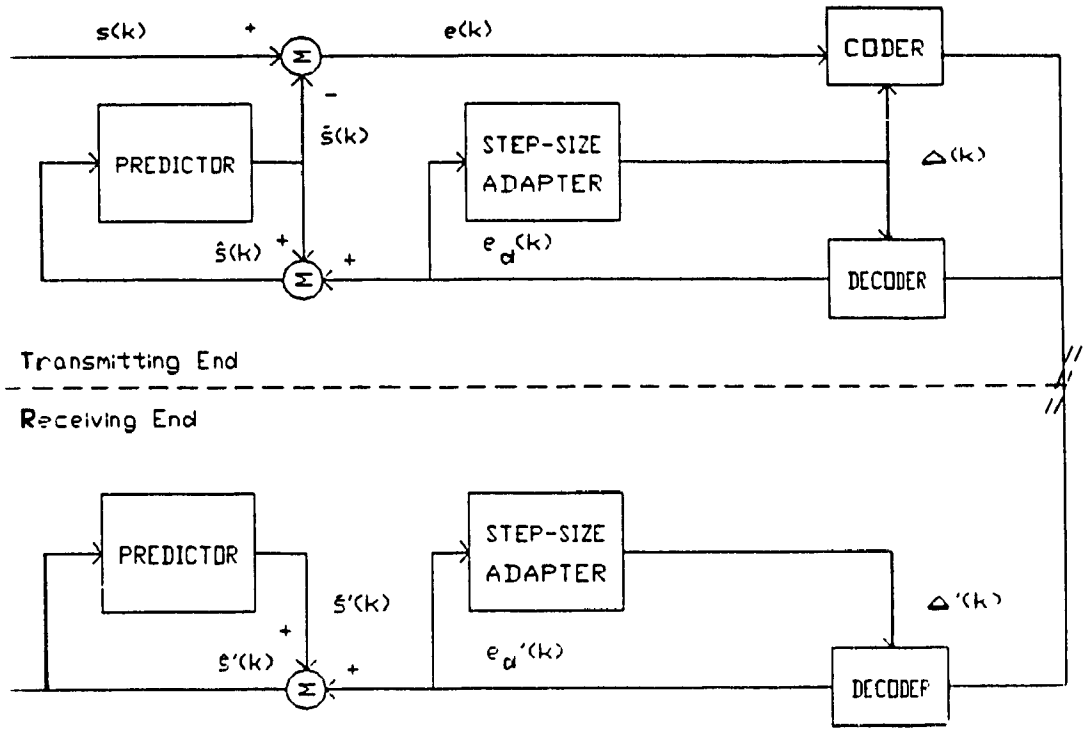


Fig.25. DPCM system with adaptive quantization.(After Castellino, Modena, Nebbia, and Scagliola [40].)

The step size

$$\Delta(k) = C^* \sigma_{e_d}(k) \tag{22}$$

The constants C and C^* in Eq. (20) and (22) have been derived theoretically and calculated by the authors to minimize the quantized error power e_q^2.

With optimum values of C and C^*, and $t = t^* = 2$, the quality of DPCM with adaptive quantizer at 32 kbits/s is reported to be comparable to a 64 kbits/s log PCM one.

When the channel is noisy, a wrong decoded prediction error $e'_d(k-1)$ at the receiver produces a wrongly estimated $\sigma'_{e_d}(k)$. Moreover, as the step size

$\Delta(k)$ is proportional to $\sigma_{e_d}(k)$, the next decoded samples $e_d(k)$ will be wrong, proportionally to $\Delta(k)$.

Scagliola [41] suggested modifying the right hand side of Eq. (19) as follows:

$$\sigma_{e_d}(k) = (1 - 2^{-t})\ \sigma_{e_d}(k-1) + 2^{-t}\ |e_d(k-1)| + a \qquad a > 0$$

a is a positive bias term.

Let $e_d(k) = I(k)\ \Delta(k)$, where $I(k)$ is the value corresponding to the codeword transmitted at time k, $\frac{1}{2} \leq |I(k)| \leq \frac{(2^B-1)}{2}$, Eq. (20) becomes :

$$\begin{aligned}\Delta(k) &= (1 - 2^{-t} + C\ 2^{-t}\ |I(k-1)|)\ \Delta(k-1) + C\ a \\ &= \alpha(k-1)\ \Delta(k-1) + C\ a\end{aligned}$$

where, with $\alpha(k-1) = 1 - 2^{-t} + C\ 2^{-t}\ |I(k-1)|$ and $a = 0$, the adaptation algorithm assumes the same form as Jayant's.

According to Scagliola, the modified algorithm would permit the quantizer to forget the effect of channel errors. And the possibility of doing so is intrinsically due to the fact that adaptation is not perfect : the quantizer step size is over-estimated for the lowest signal levels and under-estimated for the highest ones. As a result, the faster we make the error recovery, the greater is the impairment in SNR at low and high signal levels.

A method called pseudo-syllabic adaptation has been reported by Raulin, Bonnerot, Jeandot, and Lacroix [11]. This is a peak detection technique. The step size $\Delta(k)$ is restricted to 12 values :

$$\Delta(k) = \Delta_0\ 2^{i(k)} \qquad 0 \leq i \leq 11$$

where Δ_0 is the basic step size,

every step size is specified by its index i.

The all-zero code is not used in transmission; therefore, only 15 values are available for the 4-bit ADPCM code $e_q(k)$

$$-7 \leq e_q(k) \leq 7$$

If $|e_q(k)|$ exceeds a predetermined threshold, a constant value of 256 is fed to a first-order low-pass filter with transfer function

$$H(z) = \frac{1}{1 - \frac{511}{512} z^{-1}} .$$

Otherwise, the filter input is zero.

The effect is to increase the filter output $p(k)$ by 256 when the threshold is exceeded or to reduce it by a factor of 511/512 otherwise.

Then, the step size index $i(k)$ at time k is taken as

$$i(k) = \frac{p(k)}{2048}$$

This allows the step size to vary over the range of Δ_0 to $2048\Delta_0$.

An increase of the index $i(k)$ by one unit can be achieved in eight samples. Consequently, the step size can double in 1 ms for a sampling rate of 8 kHz, while the step size reduces by half in a time that varies between 4 and 16 ms, depending on the initial step size.

To make the transition smoother, four non-linear quantizing laws have been used, in which, for a fixed index $i(k)$, the step size is doubled for $e_q(k)$ greater than 5, 4, 3, and 2, respectively. Look-up tables and simple logic circuits were used to realize the quantization part of the system.

In a pseudo-syllabic system, the quantizer is fixed during the time of eight samples required for step increase or during the time required for step size decrease.

Chakravarthy, Georganas, and Shiva [42] proposed a step size adaptation algorithm for Incremental Adaptive Quantizers (IAQ). The feature of this design is that the step size can be increased or decreased by a fixed amount at each sampling instant. This fixed value is also the minimum allowed step size. There is also a provision for keeping the step size constant.

Such a quantizer can be designed using the following rule. Consider a B-bit quantizer with 2^{B-1} levels of either polarity. The levels (of either polarity) can be partitioned into three groups G1/G2/G3, with G1 containing the n lowest levels, G3 comprising the n uppermost levels, and G2 having the rest. An increment in the step is effected whenever an input sample is quantized to a level in G3, and the step size is reduced when the quantized level is in G1. Otherwise, no change is made.

Thus, considering a 4-bit quantizer (with eight levels of either polarity), the different partitions that are possible are 0/8/0, 1/6/1, 2/4/2, 3/2/3, and 4/0/4.

The step size change, itself, can be implemented in several ways. Take the 2/4/2 partition. When a sample falls into either of the two lowest levels, we can decrease the step by an amount equal to the minimum step value Δ_0. The same applies to the increment also. We call this the (1,1) weighting. Alternately, the decrease for the lower of the two levels can be $2\Delta_0$, while the decrement for the other level is Δ_0. Similarly, an increase of $2\Delta_0$ is effected for the highest level, and Δ_0 for the second level. This is termed the (1,2) weighting. In a similar fashion, for the 3/2/3 partition, we can have weights (1,1,1), (1,2,3), (1,2,4), etc... In general, for the different partitions, there are various possible weightings. After studying the simulation results, the authors suggested the

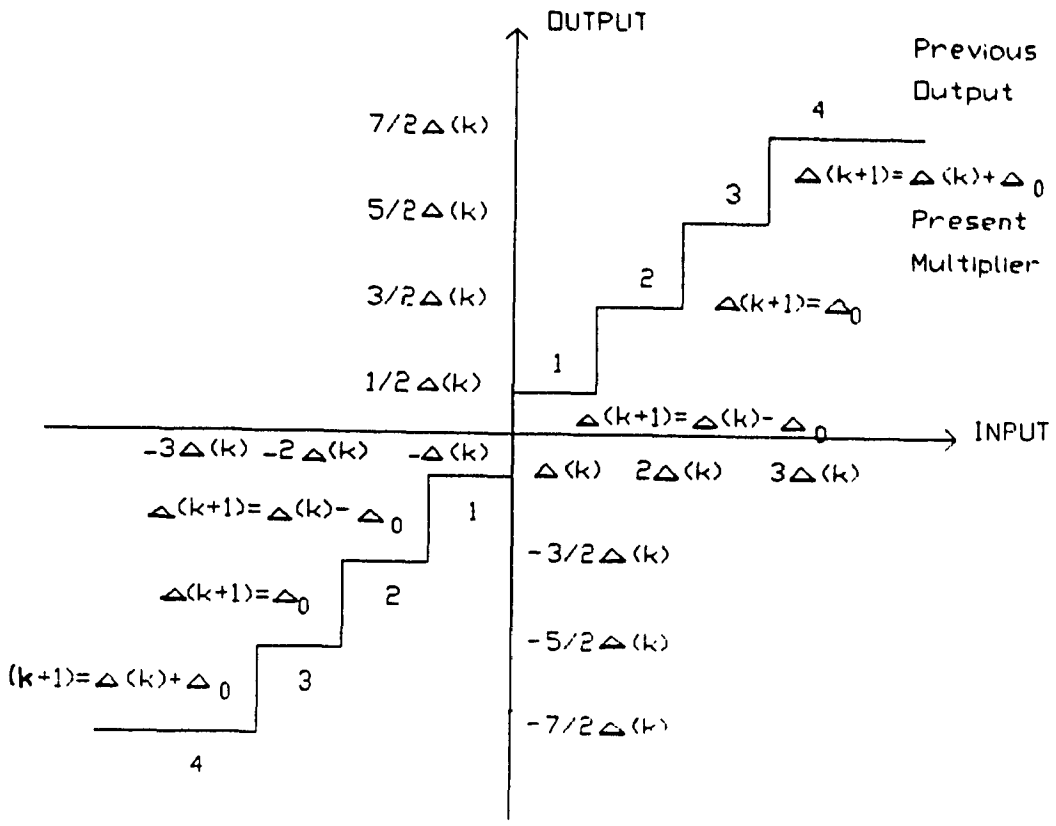


Fig.26. An Incremental Adaptive Quantizer.

following step size adaptation rule that applies to any number of bits :

$$\Delta(k) = \begin{cases} \Delta(k-1) - \Delta_0 & \text{if } |b(k)| < (2^{B-1})/4 \\ \Delta(k-1) + \Delta_0 & \text{if } |b(k)| \geq (2^{B-1}) * 3/4 \\ \Delta(k-1) & \text{otherwise} \end{cases}$$

where Δ_0 stands for the minimum step size,

$b(k)$ is the magnitude of the codeword.

Computer simulations showed that the quantizer can perform as well as Jayant adaptive quantizer for PCM or DPCM coders. As the proposed scheme does not change the step size by large amounts, it produces less granularity. However, this would obviously lead to excessive slope overload when the input changes very fast.

Nasr and Chakravarthy [18] described Hybrid Adaptive Quantization (HAQ) for speech coding. This technique uses instantaneous as well as syllabic adaptation of the step size. According to the authors, both rules acting in conjunction provide a more accurate reproduction of the waveform. The adaptive quantizer increments or decrements the step size at each instant by a constant value Δ_0. To vary the steps quickly, this incremental change must be large, whereas to accomodate low level signals, Δ_0 must be small. Therefore, it seems that Δ_0 must be made proportional to the speech envelope for the best result. In the HAQ, the envelope is extracted from the output of the quantizer, and this signal is used to control the increment change. The schematic diagram of the HAQ is shown in Fig.27.

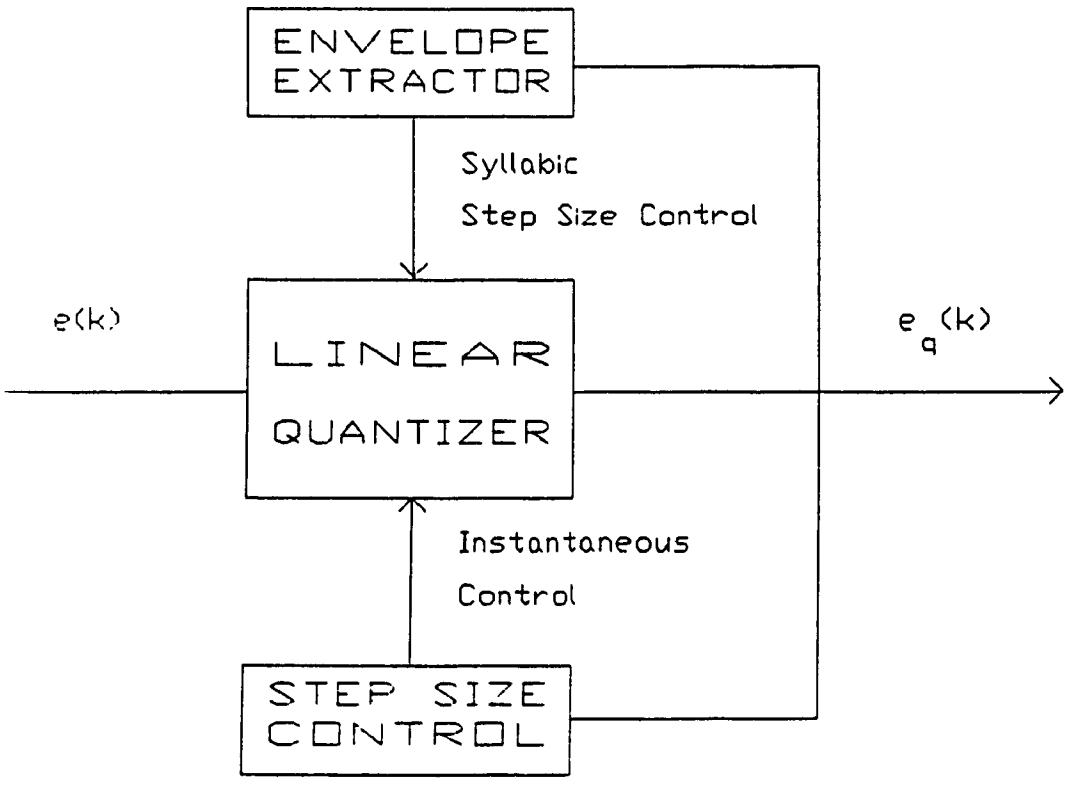


Fig.27. Schematic block diagram of HAQ.

The envelope detector is a full wave rectifier followed by a low-pass filter. Jayant quantizer as well as the IAQ described earlier have been used in the system. For the IAQ, the adaptation rule is modified as follows :

$$\Delta(k) = \begin{cases} \Delta(k-1) - \Delta_0(k) & \text{if } |b(k)| < (2^{B-1})/4 \\ \Delta(k-1) + \Delta_0(k) & \text{if } |b(k)| \geq (2^{B-1}) * 3/4 \\ \Delta(k-1) & \text{otherwise} \end{cases}$$

$\Delta_0(k)$ is the minimum step size at the k^{th} sampling instant, and is calculated as

$$\Delta_0(k) = K\ P(k)$$

where K is a constant,

$P(k)$ is the value of the envelope at that instant.

Computer simulations have been performed for a sine wave, correlated Gaussian signal, and digitized speech with 3-bit and 4-bit PCM and DPCM quantizers. SNR computation indicates that the hybrid technique is superior to the normal adaptive quantizer when both have the same ratio of maximum to minimum step size.

The performance of ADPCM system with HAQ has been studied by the same two authors [43]. Compared with CCITT 32 kbits/s ADPCM (refer to chapter IV) and another ADPCM, the HAQ is seen to perform much better. Simulation results reveal that with 3- and 4-bit quantizers, a dynamic range of 55-60 dB can be obtained.

Qureshi and Forney [44] developed a quantizer adaptation strategy which relates directly to syllabic companding. The algorithm is claimed to result in a smoothly varying step size but have the capability of rapid expansion upon indication of overload.

Consider a B-bit quantizer with output levels l_i, $i = 1, 2, \cdots, 2^B$. With the quantized difference signal $e_q(k)$ falling into an outer level as a cue to overload,

the step size $\Delta(k)$ is increased through a decaying correction factor :

$$\Delta(k) = \lfloor 2^{\lfloor G(k) \rfloor} \ (1 + G(k)) - |G(k)| \rfloor$$

where $\lfloor \ . \ \rfloor$ means "integer part of",

$G(k)$ is defined as $\log_2(\Delta(k))$, and is calculated as :

$$G(k) = G'(k) + C(k) + G_{min}$$

where G_{min} is the least value of $G(k)$,

the correction factor $C(k)$ is adjusted according to

$$C(k+1) = \gamma \ C(k) + f_2[|l_i(k)|]$$

$f_2[\ .\ ]$ is zero for all but the outer most levels.

In the absence of overload, $G(k)$ varies as $G'(k)$ which is slowly updated according to :

$$G'(k+1) = max(P \ G'(k) + f_1[|l_i(k)|] \ , \ 0)$$

where $f_1[\ .\ ] = \log_2 M[\ .\ ]$ close to zero except for the outer most levels,

$M[\ .\ ]$ is a set of time-invariant expansion/contraction factors,

a decay is introduced in $G'(k)$ with a time constant $1/(1-\rho)$ samples.

This quantizer, sometimes referred to as Pitch Compensating Quantizer (PCQ), has been used by Gibson and Berglund [35], Gibson, Berglund, and Sauter [29].

In DPCM-AQF coders, where AQF stands for adaptive quantization with forward estimation, the scaling of the quantizer is adjusted once for every block of W input samples. As the step size of the quantizer is evaluated from the input signal before it is passed forward to the DPCM encoder, the received

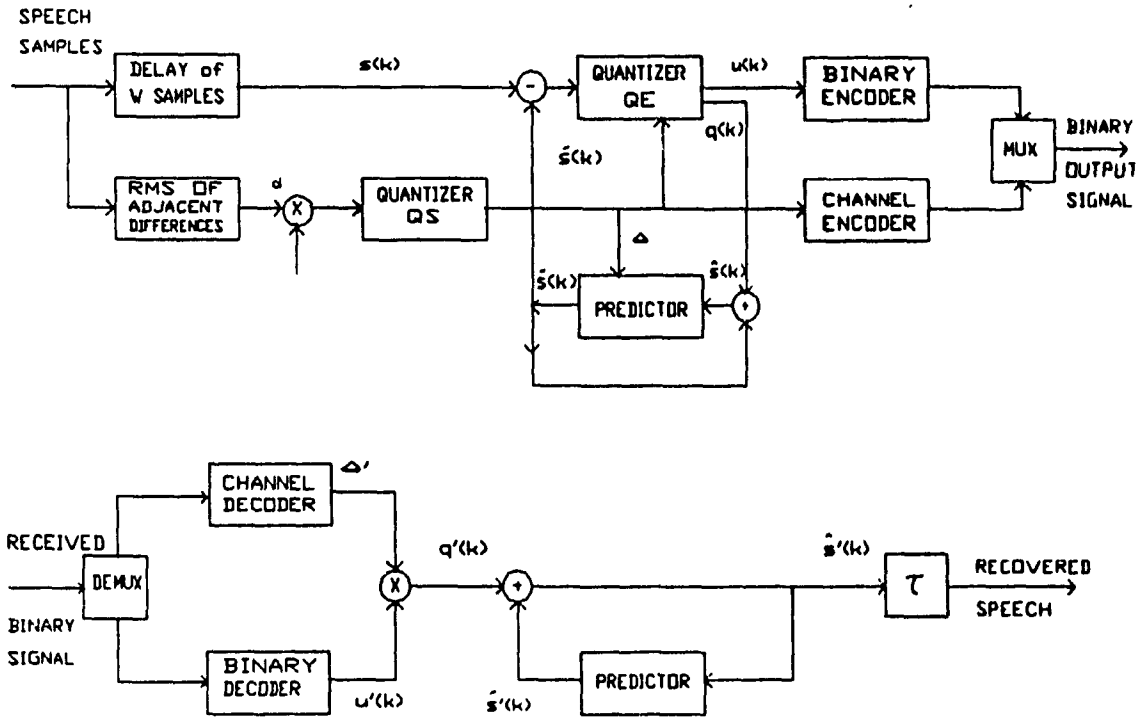


Fig.28. Block diagram of DPCM-AQF system (a) Encoder (b) Decoder.

signal is delayed by W-sample periods. Evci, Steele, and Xydeas [45] studied a DPCM-AQF system the block diagram of which is shown in Fig.28.

The rms value σ_d of the differences between adjacent samples in a block of W samples is calculated as :

$$\sigma_d = \left[\frac{1}{W} \sum_{k=2}^{W} [s(k) - s(k-1)]^2\right]^{1/2}$$

and the step size Δ used in the DPCM quantizer QE is formed by quantizer QS as :

$$\Delta = Q[\alpha\ \sigma_d]$$

where $Q[\ .\ ]$ means the quantization of $[\ .\ ]$, α is a system parameter. For 4-bit DPCM-AQF, α is chosen to be 0.33. The DPCM quantizer QE accepts the signal $(s(k) - \tilde{s}(k))$ to be quantized, namely the difference between the delayed input sample $s(k)$ and its predicted value $\tilde{s}(k)$, the step size Δ, and produces a level number $u(k)$ and a quantization level $q(k)$ where the k subscript means the k^{th} sampling instant in a particular block :

$$q(k) = \Delta\ u(k) \qquad\qquad k = 1, 2, \cdots, W$$

and the values that $u(k)$ can assume are

$$u = \pm(2r - 1)/2 \qquad\qquad r = 1, 2, \cdots, 2^{B-1}$$

where B is the number of bits in the DPCM-AQF codeword,

$u(k)$ is binary encoded and multiplexed with the channel-encoded Δ.

At the receiver, the binary signals are demultiplexed to yield Δ' and $u'(k)$. In the absence of transmission errors, $u(k) = u'(k)$, $q(k) = q'(k)$, $\tilde{s}(k) = \tilde{s}'(k)$. If sufficient channel protection coding is employed, Δ' may be assumed to be received with negligible error.

A quantizer adaptation algorithm has been proposed by Petr [46] for both speech and non-speech signals (such as VBD). While speech input produces difference signal the power of which varies rather rapidly, VBD generally

produces a difference signal with relatively constant power. On one hand, the large dynamic range of speech signals calls for adaptive quantization. On the other hand, significantly better VBD performance can be achieved with a properly scaled fixed quantizer. In addition to these, compromise adaptation speeds degrade both speech and VBD performance. Petr described an ADPCM-based 32 kbits/s coder with Dynamic Locking Quantizer (ADPCM-DLQ). The specific quantizer has two speeds of adaptation, "unlocked" for speech, and "locked" for VBD and tone signals.

Fig.29 is a block diagram of the DLQ scale factor (Δ) adaptation technique.

The upper portion of the figure shows the generation of two scale factors, Δ_U and Δ_L, which are then combined to form the new scale factor Δ. This Δ is then used to scale a non-linear quantizer characteristic.

The "unlocked" scale factor Δ_U is generated as suggested by Goodman and Wilkinson [34] :

$$\Delta_U(k+1) = \Delta^{\beta}(k)\ M(I(k))$$

where $M(I)$ is a constant function of the quantizer step number, and k is the time index. The exponent $\beta < 1$ introduces finite memory to mitigate the effect of transmission errors.

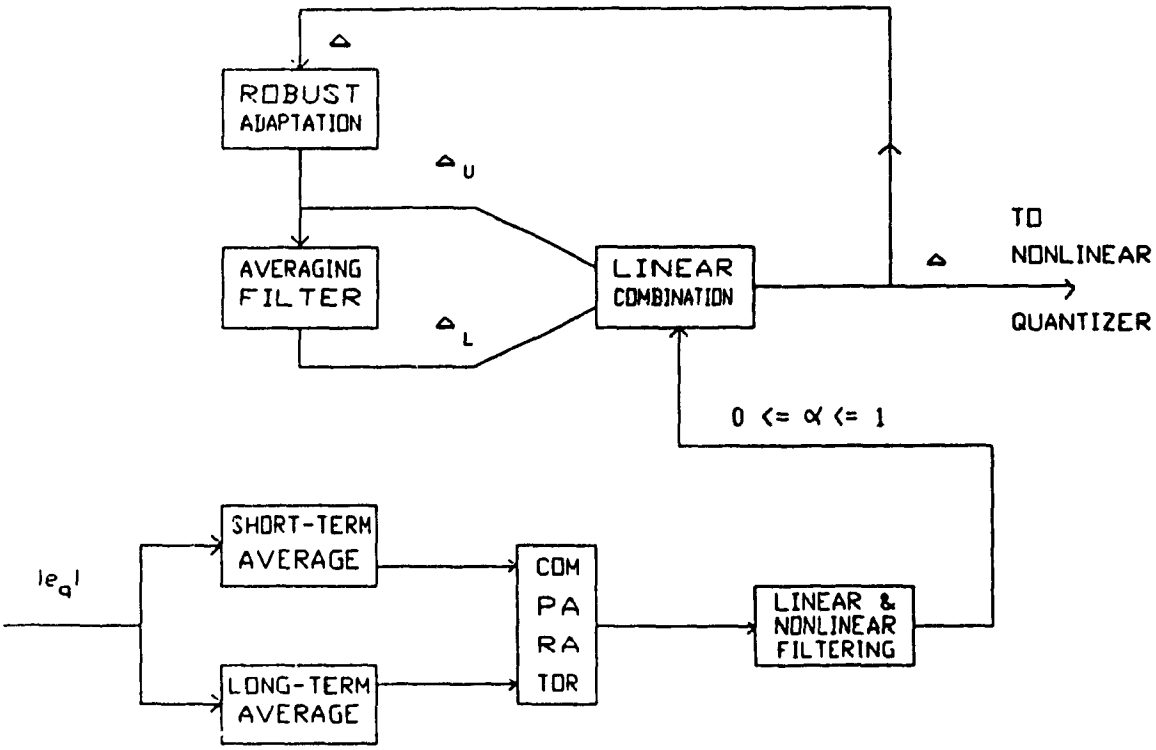


Fig.29. DLQ scale factor adaptation.

The "locked" scale factor Δ_L is produced by averaging Δ_U to effectively produce a fixed scale factor of the proper magnitude when the input signal is VBD or a tone signal. The linear combination of Δ_U and Δ_L is as follows :

$$\Delta = \alpha\ \Delta_U + (1 - \alpha)\ \Delta_L$$

The controlling parameter α is generated as in the lower portion of the figure. The operation of the circuit is such that when the average magnitude of the quantized difference signal ϵ_q is rapidly varying (e.g. speech input), α is driven to 1, unlocking the quantizer ($\Delta = \Delta_U$). Conversely, when the average magnitude of ϵ_q is relatively constant (e.g. VBD input), α is driven to 0, locking the quantizer ($\Delta = \Delta_L$).

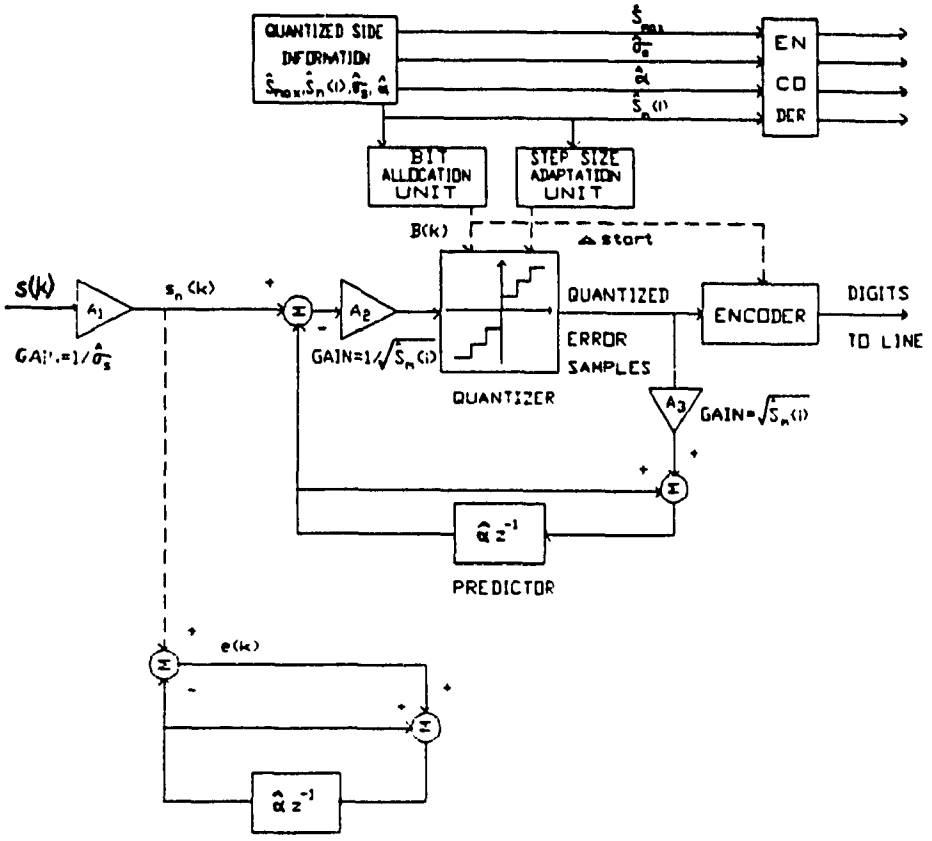


Fig.30. Encoder for ADPCM-AB system.

The DLQ, which switches from fast tracking for speech to slow tracking for VBD signals, has been used in Millar and Mermelstein's coder [47].

In ADPCM with adaptive bit allocation (ADPCM-AB), the number of quantization bit per sample is not equally allocated over time intervals of speech signals. This algorithm was described by Frangoulis, Yoshida, and Turner [48].

The encoder for the ADPCM-AB system is shown in block schematic form in Fig.30, and the associated decoder is shown in Fig.31.

The system is basically similar to a conventional ADPCM system, except for the provision of a duplicate prediction unit, which operates in a non-quantizing mode, and an adaptive bit allocation unit. Information for use in the adaptive bit allocation unit is derived from the duplicate prediction unit.

Input speech samples $s(k)$ are handled in blocks of $N(=128)$ samples. It is the block nature of the processing that causes the encoding and decoding delays introduced by the system. The sequence of input speech samples is normalised on a block basis, with the gain G of the amplifier A_1 adjusted from one block to the next in accordance with the relation $G = \hat{\sigma}_s^{-1}$ where

$$\sigma_s^2 = \frac{1}{N} \sum_{k=1}^{N} s^2(k)$$

and $\hat{\sigma}_s$ is a quantized version of σ_s.

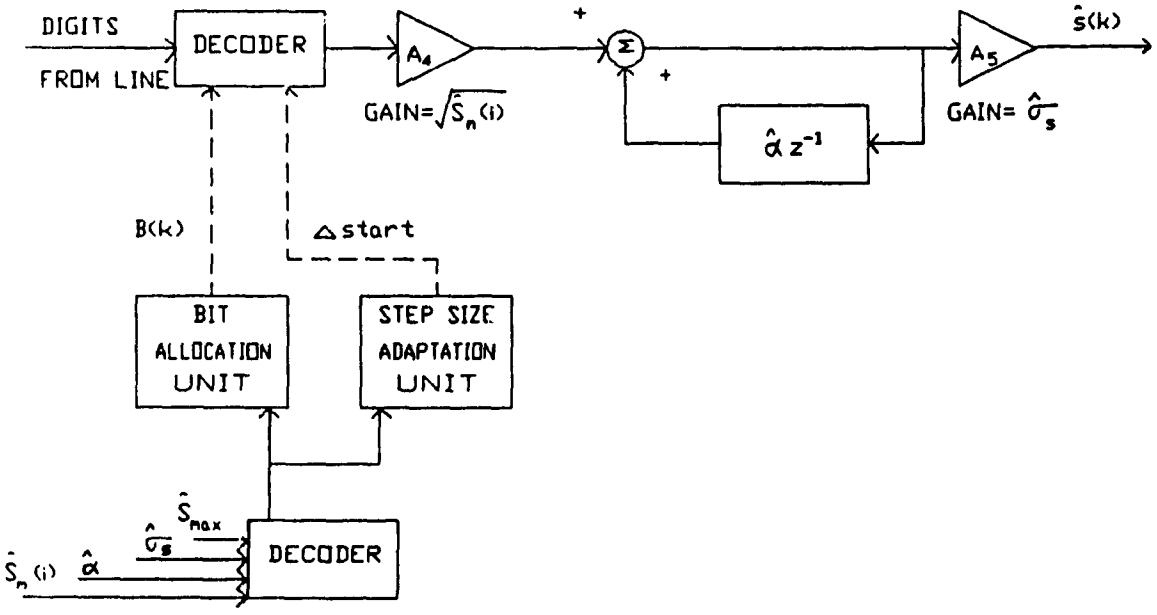


Fig.31. Decoder for ADPCM-AB system.

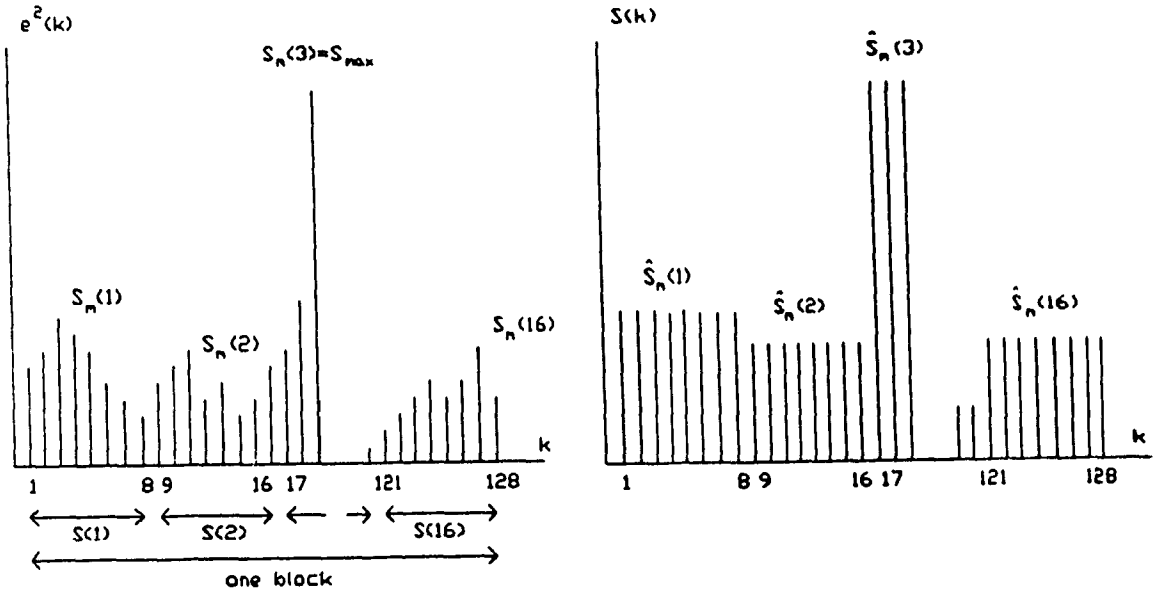


Fig.32. Error sample adjustment for use in adaptive bit allocation.

The normalised speech samples $s_n(k)$ are fed to a duplicate ADPCM unit where they are used to derive information relating to the number of bits to be used in the adaptive quantizer part of the main ADPCM system. First-order prediction is used in both the main ADPCM loop and the duplicate non-quantizing ADPCM unit. The prediction coefficient a is derived by the normal autocorrelation method. The unquantized prediction error $e(k)$ is squared, and the block of 128 such squared error samples $e^2(k)$ are subdivided into 16 equal blocks of eight samples. This process is illustrated in Fig.32.

The maximum squared-error value within the i^{th} block is denoted $S_m(i), i = 1, \cdots, 16$, and the largest of these 16 maximum values is denoted by S_{max}. The values of the $S_m(i)$ are used for each sub-block to adjust the gain of the amplifier

$A_2 = \frac{1}{\sqrt{\hat{S}_m(i)}}$ where $\hat{S}_m(i)$ is the quantized version of $S_m(i)$, so that the prediction error values fed to the quantizer of the quantizing ADPCM part of the system are normalised to have a maximum value of unity during each sub-block.

In the adaptive bit allocation unit, the squared samples within each sub-block are set equal to $\hat{S}_m(i)$, the quantized version of $S_m(i)$. The adjusted squared samples are denoted by $S(k)$.

The number of bits $B(k)$ to be used in the main ADPCM loop to quantize the k^{th} error sample $e(k)$ is derived from :

$$B(k) = \overline{B} + 0.5 \log_2 S(k) - \frac{1}{2N} \sum_{k=1}^{N} \log_2 S(k) - T \qquad (23)$$

where $\overline{B}$ is the average number of bits per sample used to quantize each error sample,

T is a small adjustment term which is necessary to achieve a constant transmission bit rate.

In the process of determining $B(k)$, an intermediate value $B'(k)$ is determined from Eq. (23) with T being initially equal to zero. If $\epsilon = N\overline{B} - \sum_{k=1}^{N} B'(k)$ is negative, which indicates that too many bits have been allocated, T is increased from 0 to 0.1. The calculation of $B'(k)$ is repeated, with the process being continued until ϵ is either zero or positive, which indicates that either the correct number of bits have been allocated or that a residual integer number of bits remain for allocation, respectively. If ϵ is positive, one further bit is allocated to each of the ϵ variables, $B'(k)$, which differ from their respective $B(k)$ by the greatest amount.

The quantizer used in the quantizing ADPCM loop was Jayant's, with the additional feature that the starting step size was adjusted every sub-block in accordance with $S_m(i)$.

An amplifier A_3 with gain equal to the inverse of that of amplifier A_2 was used in deriving the appropriate quantized predicted value for comparison with normalised input samples $s_n(k)$.

In this system, besides the quantized error samples, other quantized information has to be sent to the receiver : the quantized versions of σ_s, S_{max}, $S_m(i), i = 1, \cdots, 16$, and the prediction coefficient α.

It was reported that at 32 kbits/s, the performance of the proposed system is indistinguishable from that of conventional ADPCM system. At 16 kbits/s, the performance of the system is found to be superior to that of the conventional ADPCM systems.

3.2. Adaptive Predictor

It is suboptimum to use a single predictor for encoding quasistationary signal. As the statisctics of the input signal are changing, the performance of a DPCM system operating with a predictor designed on the basis of a certain signal statistics will degrade. By achieving the maximum possible value of SNR for all possible input signals, adaptive prediction can definitely improve DPCM performance. This requires dynamically adjusting the predictor coefficients $\{a_i\}$ to match the characteristics of the incoming signal.

3.2.1. Adaptive Prediction Techniques

Analogous to adaptive quantizers, adaptive predictors can be classified as either forward adaptive or backward adaptive. For feedforward control, the predictor adaptation is based upon measurements on the system input $s(k)$ while feedbackward control of adaptive predictors uses the quantized signal $\hat{s}(k)$.

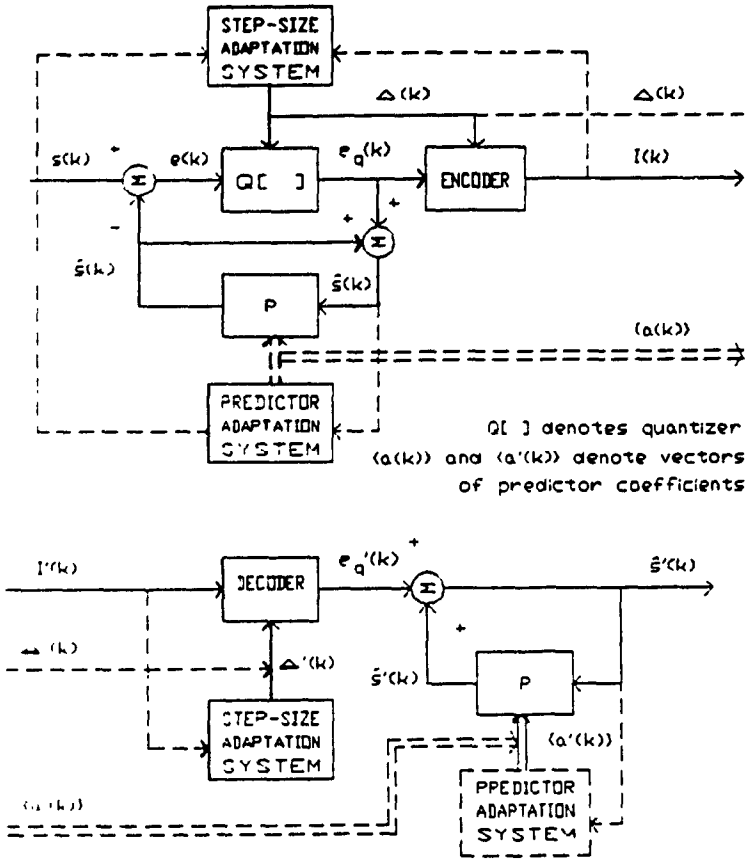


Fig.33. ADPCM system with both adaptive quantization and adaptive prediction.

Forward prediction has the disadvantage that predictor information must be sent to the receiver; Thereby, the number of bits allocated to the prediction error $e_q(k)$ is reduced. Besides, as quantization of the predictor coefficients can cause stability problems, the predictor coefficients are usually transformed into another set of parameters, prior to transmission. These coefficients must be combined with the $e_q(k)$ data stream. This combined stream is not homogeneous, and thus, synchronization may be difficult. Further, the coefficients and the quantized error signal have different sensitivities to transmission errors; Hence, two different coding schemes are required. On the other hand, forward

prediction has the advantages that the coefficients are exactly matched to the input signal $s(k)$, and that the receiver is not overly complex.

Backward adaptive prediction offers the advantages of not requiring additional information to be transmitted to the receiver other than $e_q(k)$, and rapid response to non-stationary input signal behavior. The disadvantages of backward prediction are sensitivity to transmission errors and receiver complexity, since both transmitter and receiver must have the same adaptation logic.

Gibson [49] analysed and compared forward and backward adaptation logic for the predictor coefficients in ADPCM. The conclusion was that no clear-cut preference for forward or backward adaptive prediction can be stated, although backward adaptation can outperform forward adaptation in some cases of practical interest.

Experimental comparison of forward and backward adaptive prediction in ADPCM has been carried out by Gibson and Sauter [4]. It was found out that if the additional data rate required for side information is ignored, forward prediction is preferred over backward prediction. However, if the data rate required for side information is included, then backward prediction outperforms forward prediction for equal data rates.

The performance of DPCM systems with fixed, and adaptive prediction has been compared. Noll [3] has shown that the maximum prediction gain is 10.5 dB for fixed prediction, and 14 dB for adaptive prediction.

3.2.2. Adaptive Prediction Algorithms

Numerous schemes for adaptive prediction have been proposed to improve the performance of ADPCM systems. The schemes may differ from one to another because of the transfer function (all poles, all zeros, poles and zeros) or of the way of implementing the predictor (transversal, lattice). More often, the

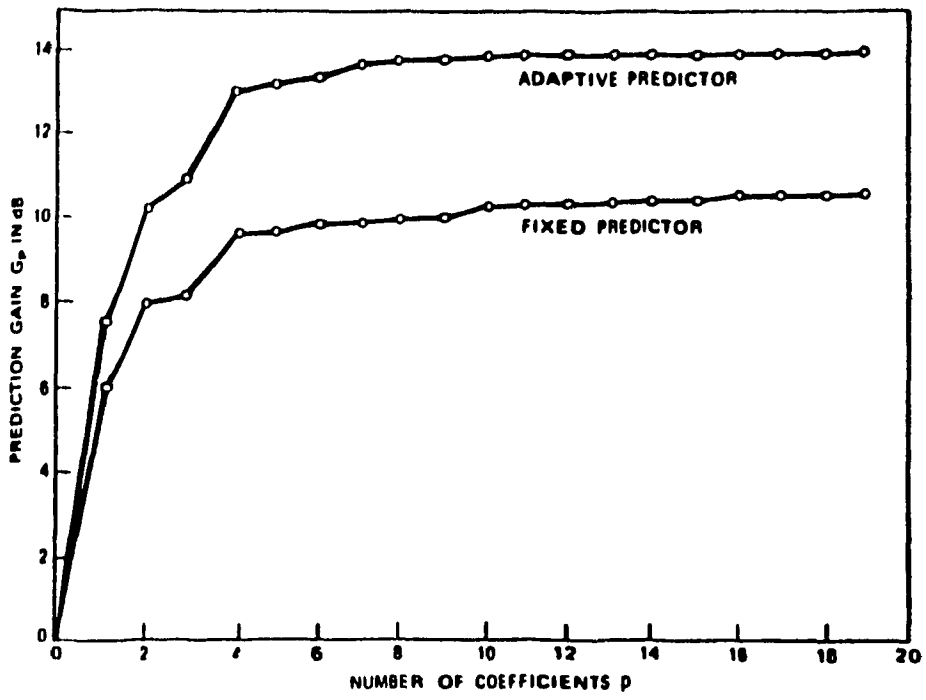


Fig.34. Prediction gain versus number of predictor coefficients.

prediction coefficients are calculated to match the sampled inputs. Nevertheless, sometimes the system just switches from one pre-calculated set of coefficients to another in accordance with the variation of the input signal. Following is a description of some adaptive prediction algorithms.

Fukasawa, Hosoda, Miyamoto, and Sugihara [33] based their ADPCM codec on adaptive prediction for speech, modem, and VBD signals. The adaptive predictor is designed by a set of non-recursive cascaded filter blocks, and is described as stable and having minimum phase shift response in each iteration of adaptation.

The predictor consists of zero-point and pole filters. The output $\tilde{s}(k)$ is the sum of the output $\tilde{s}_z(k)$ of the zero-point filter and the output $\tilde{s}_p(k)$ of the

pole filter. The prediction signal is written by the following equations :

$$\tilde{s}(k) = \tilde{s}_p(k) + \tilde{s}_z(k)$$

$$\tilde{s}_p(k) = \sum_{i=1}^{m} a_i \, \hat{s}(k-i)$$

$$\tilde{s}_z(k) = \sum_{i=1}^{n} b_i \, \hat{e}(k-i)$$

$$\text{and} \qquad e(k) = s(k) - \tilde{s}(k)$$

where $\{a_i\}$ and $\{b_i\}$ are the tap coefficients of the pole filter and zero filter respectively.

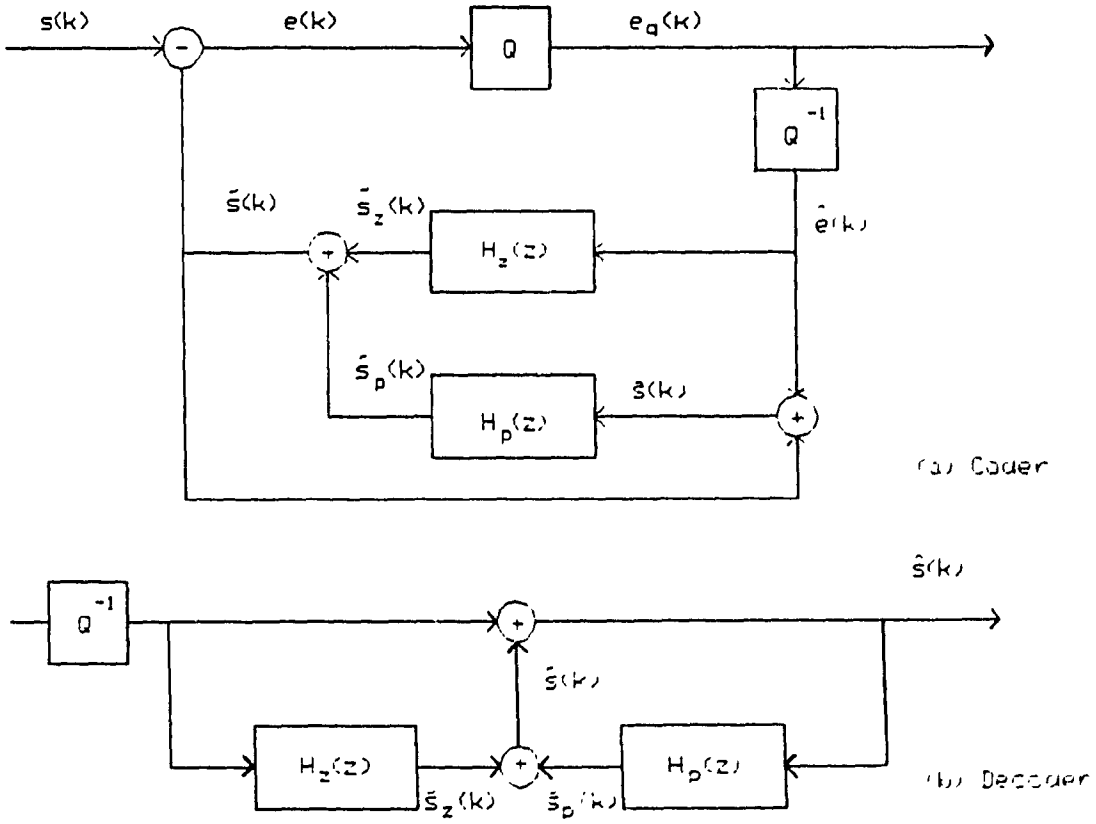


Fig.35. Block diagram of Fukasawa, Hosoda, Miyamoto, and Sugihara's ADPCM encoder and decoder.

The transfer function of the predictor is an approximation function of the spectrum structure of the input signal. A Tschebyscheff function is introduced as the transfer function in the sense of "minimax approximation".

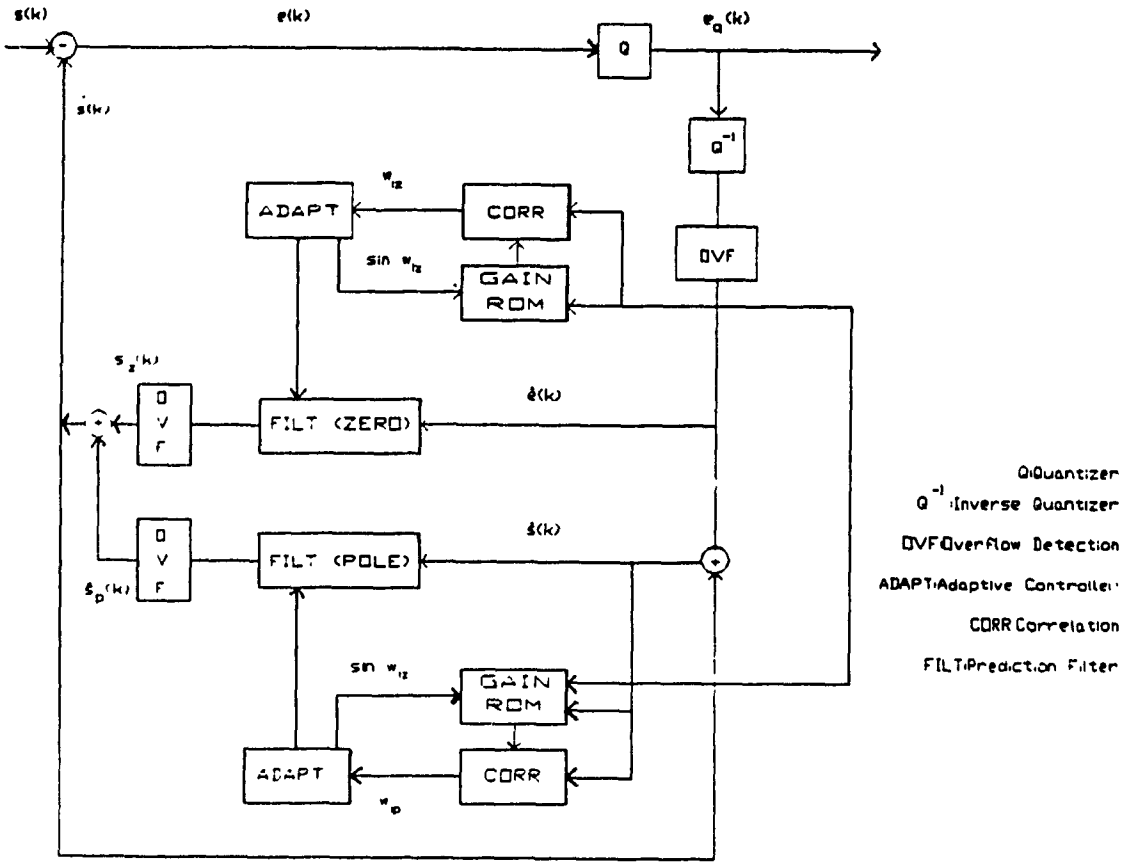


Fig.36. Block diagram of Fukasawa, Hosoda, Miyamoto, and Sugihara's ADPCM encoder.

The overall transfer function of the predictor is given by :

$$H(z) = \frac{1 + H_z(z)}{1 - H_p(z)}$$

where $H_z(z)$ and $H_p(z)$ are the transfer functions of the zero-point and pole filters, respectively.

For the zero-point filter,

$$H_z(z) = \frac{1}{2}(C_z(z) + D_z(z)) - 1$$

and

$$C_z(z) = (1 + z^{-1}) \prod_{i=1}^{\frac{m}{2}} (1 - 2c_{iz}z^{-1} + z^{-2})$$

$$D_z(z) = (1 - z^{-1}) \prod_{i=1}^{\frac{m}{2}} (1 - 2d_{iz}z^{-1} + z^{-2})$$

where tap coefficients c_{iz} and d_{iz} are as follows :

$$\left.\begin{aligned} c_{iz} &= \cos\omega_{iz} \\ d_{iz} &= \cos\nu_{iz} \end{aligned}\right\} \quad i = 1 \cdots \frac{m}{2}$$

For the pole filter,

$$H_p(z) = \frac{1}{2}(C_p(z) + D_p(z)) - 1$$

and

$$C_p(z) = (1 + z^{-1}) \prod_{i=1}^{\frac{n}{2}} (1 - 2c_{ip}z^{-1} + z^{-2})$$

$$D_p(z) = (1 - z^{-1}) \prod_{i=1}^{\frac{n}{2}} (1 - 2d_{ip}z^{-1} + z^{-2})$$

where tap coefficients c_{ip} and d_{ip} are as follows :

$$\left.\begin{aligned} c_{ip} &= \cos\omega_{ip} \\ d_{ip} &= \cos\nu_{ip} \end{aligned}\right\} \quad i = 1 \cdots \frac{n}{2}$$

According to the authors, the stability and minimum-phase-shift conditions are always ensured for each iteration of adaptations. Hence, the predictor becomes very robust to channel error disturbance. Their specific 32 kbits/s ADPCM codec has a predictor consisting of zero and pole adaptive filters with six orders.

A tenth-order linear prediction adaptive filter realized as a cascade of five second-order sections has been described by Raulin, Bonnerot, Jeandot,

and Lacroix [11], and Raulin and Jeandot [50]. The filter order $P = 10$ has been chosen as a good trade off between performance and complexity : even though the prediction gain increases with the filter order, there is little to be gained with P greater than 10. The authors have cited several advantageous features of the cascade of second-order sections : easy control of gain and stability, minimum number of multiplications, high level of modularity, and small coefficient accuracy.

Starting with the z transfer function :

$$H(z) = 1 - \sum_{i=1}^{P} b_i z^{-i}$$

where $b_i, (1 \leq i \leq P)$ are the prediction filter coefficients, the authors derive an expression for the prediction filter in the form :

$$H(z) = \frac{1}{1 + \dfrac{\sum_{i=1}^{P} b_i z^{-i}}{1 - \sum_{i=1}^{P} b_i z^{-i}}} \tag{24}$$

If P is an even integer, the factorization of $H(z)$ on second-order sections is given by :

$$\begin{aligned}
H(z) = 1 - \sum_{i=1}^{P} b_i z^{-i} &= \prod_{i=1}^{\frac{P}{2}} (1 - b_{1i} z^{-1} - b_{2i} z^{-2}) \\
&= \prod_{i=2}^{\frac{P}{2}} (1 - b_{1i} z^{-1} - b_{2i} z^{-2}) \\
&\quad - (b_{11} z^{-1} + b_{21} z^{-2}) \prod_{i=2}^{\frac{P}{2}} (1 - b_{1i} z^{-1} - b_{2i} z^{-2}) \\
&= \prod_{i=3}^{\frac{P}{2}} (1 - b_{1i} z^{-1} - b_{2i} z^{-2}) \\
&\quad - (b_{12} z^{-1} + b_{22} z^{-2}) \prod_{i=3}^{\frac{P}{2}} (1 - b_{1i} z^{-1} - b_{2i} z^{-2}) \\
&\quad - (b_{11} z^{-1} + b_{21} z^{-2}) \prod_{i=2}^{\frac{P}{2}} (1 - b_{1i} z^{-1} - b_{2i} z^{-2})
\end{aligned}$$

If the procedure is carried on, the following expression is obtained :

$$H(z) = 1 - \sum_{i=1}^{P} b_i z^{-i}$$

$$= 1 - \sum_{j=1}^{\frac{P}{2}} (b_{1j} z^{-1} + b_{2j} z^{-2}) \prod_{i=j+1}^{\frac{P}{2}} (1 - b_{1i} z^{-1} - b_{2i} z^{-2})$$

Hence, Eq. (24) can be written as :

$$H(z) = \frac{1}{1 + \sum_{j=1}^{\frac{P}{2}} \frac{b_{1j} z^{-1} + b_{2j} z^{-2}}{\prod_{i=1}^{j} (1 - b_{1i} z^{-1} - b_{2i} z^{-2})}}$$

Fig.37 shows the prediction filter for $P = 10$ with five second-order sections cascaded. The predicted signal sequence $\tilde{s}(k)$ is obtained by adding the appropriate signals.

The coefficients are adapted, using the gradient method, with the increments db_{ji} given by :

$$db_{ji} = -\delta \; e(k) \; \frac{\partial e(k)}{\partial b_{ji}} \qquad \begin{matrix} j = 1,2 \\ i = 1, \cdots, \frac{P}{2} \end{matrix}$$

where δ is a positive constant controlling the speed of convergence.

$$e(k) = \frac{1}{2\pi j} \int_C z^{k-1} H(z) \; S(z) \; dz$$

is the output sequence with $S(z)$, the z transform of the input sequence $s(k)$,and C, an appropriate closed contour.

The gradient g_{ji} $(j = 1,2;\; i = 1, \cdots, \frac{P}{2})$ is given by :

$$g_{ji} = \frac{\partial e(k)}{\partial b_{ji}}$$

$$= \frac{1}{2\pi j} \int_C z^{k-1} (-z^{-j}) \prod_{\substack{l=1 \\ l \neq i}}^{\frac{P}{2}} (1 - b_{1l} z^{-1} - b_{2l} z^{-2}) \; S(z) \; dz$$

or $$g_{ji} = \frac{1}{2\pi j} \int_C z^{k-1} \frac{-z^{-j}}{1 - b_{1i} z^{-1} - b_{2i} z^{-2}} H(z) \; S(z) \; dz$$

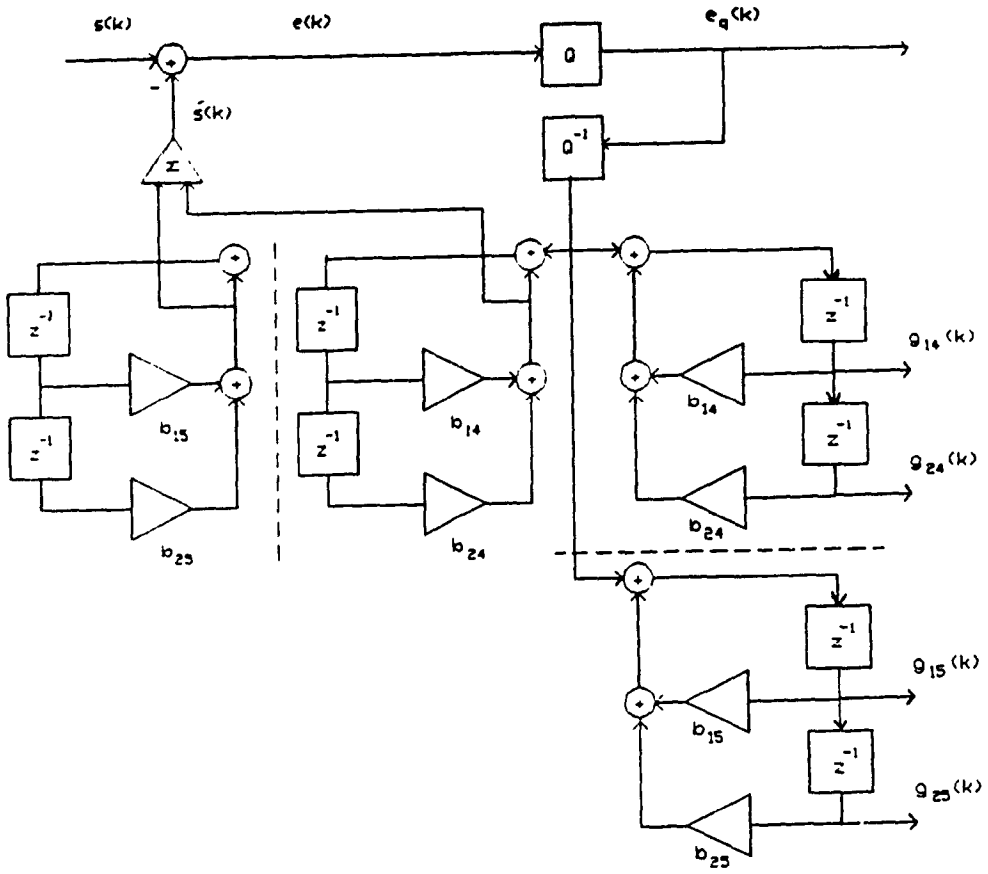


Fig.37. ADPCM coder prediction filter in cascade form.

Consequently, the gradients are obtained by a set of recursive second-order sections, with the same coefficients as those of the prediction filter, fed by the output sequence $e(k)$.

Nishitani, Aikoh, Araseki, Ozawa, and Maruta [51] proposed a robust prediction algorithm which never generates unstable poles in a decoder transfer function. In this algorithm, the decoder has a form of pole-zero speech production model, where only zeros are adaptively modified.

By introducing an infinite power series, the decoder transfer function is written as :

$$H(z) = (1 - \sum_{i=1}^{N} a_i z^{-i})^{-1}(1 + \sum_{i=1}^{\infty} b_i(k) z^{-i}) \qquad (25)$$

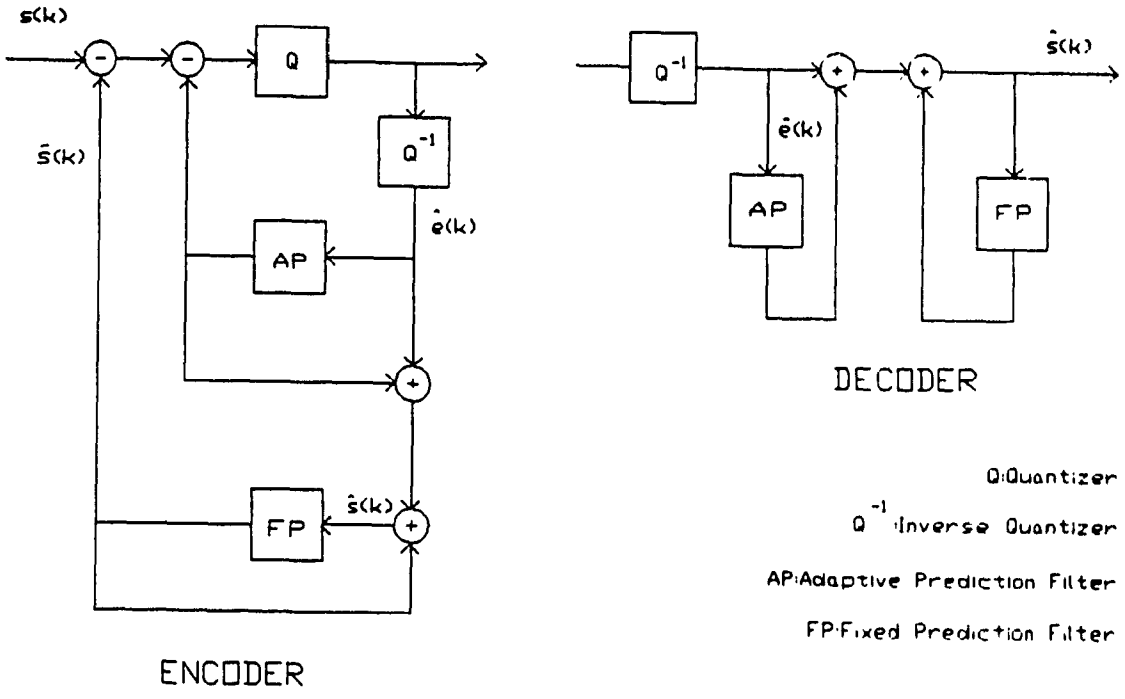


Fig.38. Nishitani,Aikoh, Araseki, Ozawa, and Maruta's ADPCM system.

where $\{a_i | i = 1, 2, \cdots, N\}$ is a set of fixed coefficients specifying fixed pole locations, and $\{b_i(k) | i = 1, 2, \cdots, \infty\}$ is an infinite set of adaptive coefficients. When fixed pole positions are chosen to be optimized for average speech signals, the infinite power series in Eq. (25) can be replaced by a truncated power series with a limited number of terms. If an M term power series are used, the approximated transfer function has N fixed poles and M adaptive zeros.

The decoder is composed of a cascade connection of an adaptive non-recursive filter and a fixed recursive filter, realizing the approximated transfer function. The prediction value $\tilde{s}(k)$ for the incoming signal $s(k)$ is calculated

by :

$$\tilde{s}(k) = \sum_{i=1}^{N} a_i \, \hat{s}(k-i) + \sum_{i=1}^{M} b_i(k) \, \hat{e}(k-i)$$

The coefficient adaptation algorithm used in AP is based on the general *gradient method* where the difference signal $e(k)$ is sequentially minimized :

$$b_i(k+1) = (1-\delta) \, b_i(k) + \frac{\alpha \, \hat{e}(k) \, \hat{e}(k-i)}{\sum_{i=1}^{M} \hat{e}^2(k-i)}$$

where δ has a value less than 1,

α is a positive constant chosen by experiment.

In the implementation of the algorithm, the coefficient adaptation is simplified so as not to include divisions such that a normalized error signal $d(k)$, uniquely determined by the quantized codeword $I(k)$, replaces the error signal term :

$$b_i(k+1) = (1-\delta) \, b_i(k) + g \; sgn[d(k)] \; d(k-i)$$

where g is a positive constant,

$sgn[\,.\,]$ indicates the sign of $[\,.\,]$.

A codec with 10^{th}-order adaptive prediction filter and 4^{th}-order fixed filter has been realized. The test results showed that the developed codec has high quality speech coding capability, comparable to that of the standard 64 kbits/s PCM.

Approaches similar to that of Nishitani, Aikoh, Araseki, Ozawa, and Maruta have been considered by other researchers. Maruta, Aikoh, Nishitani, and Kawayachi [52] described an ADPCM scheme using two predictor stages. It is combined of a fixed pole predictor-stage and an adaptive zero predictor-stage.

The portion surrounded by dotted lines indicate the adaptive zero predictor stage at the coder, and its corresponding decoder function. The remaining portions are identical to the conventional ADPCM codec.

The adaptive zero predictor-stage has no pole in its transfer function. Therefore, it is stable intrinsically. Its function is to improve the total prediction capability by further predicting the prediction error signal $d(k)$ from the fixed pole predictor-stage. The total prediction gain becomes the product of the gain for each stage. The prediction gain of the adaptive zero predictor-stage is considerably high and independent of input signal frequency.

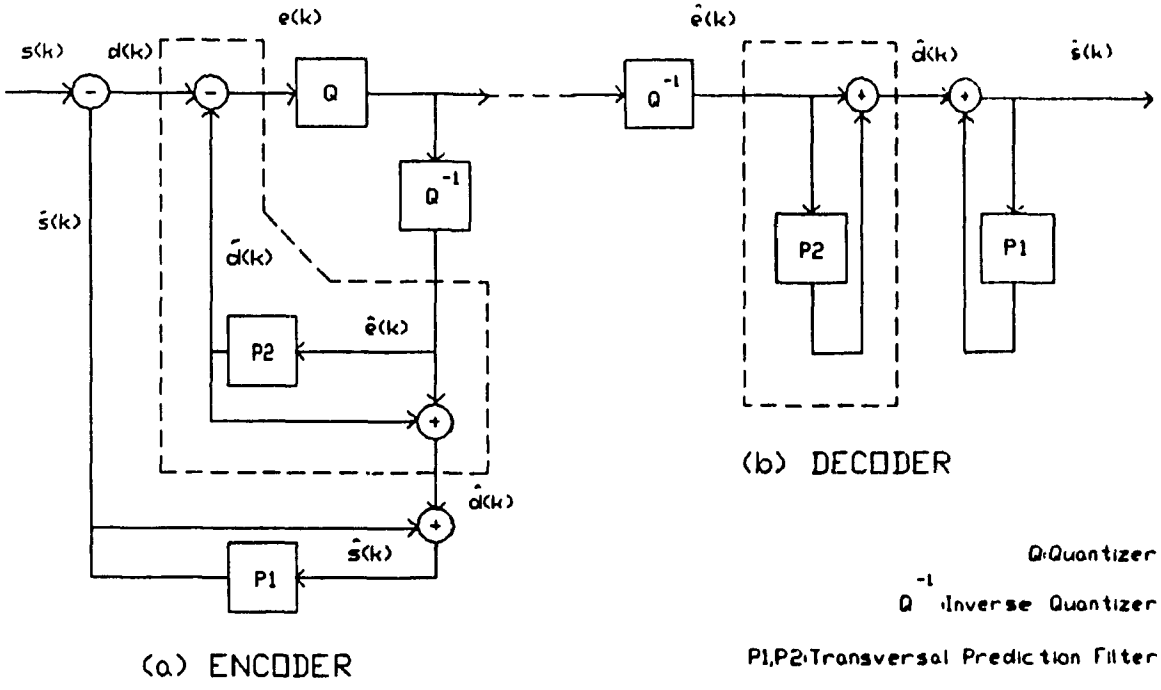


Fig.39. Maruta, Aikoh, Nishitani, and Kawayachi's per-channel ADPCM codec.

Since the voiceband modem signal spectrum differs largely from the average speech spectrum, the best data transmission performance is incompatible with the best speech transmission performance, when a fixed predictor optimized

for speech is employed. Although this incompatibility could be resolved if an adaptive control is also employed in the prediction filter P_1, it would reduce the robustness in regard to withstanding to transmission bit errors to some extent, even if oscillation is avoided by an appropriate stability control. Thus, there is a trade off between the adaptibility degree and the robustness degree regarding transmission bit errors. Experiment results suggested that the 32 kbits/s ADPCM optimized for speech can be used to transmit data signals at up to 4800 bits/s.

The predicted value $\tilde{s}(k)$ of the input ( ) in Ramamoorthy and Jayant's predictor [31] is a combination of two components, the output $\tilde{s}_z(k)$ and $\tilde{s}_p(k)$ of an all-zero predictor $B(z)$ and an all-pole predictor $A(z)$. Formally,

$$\tilde{s}(k) = \tilde{s}_z(k) + \tilde{s}_p(k)$$

$$\tilde{s}_p(k) = \sum_{i=1}^{2} a_i(k)\ \hat{s}(k-i)$$

$$\tilde{s}_z(k) = \sum_{i=1}^{6} b_i(k)\ e_q(k-i)$$

where $e_q(k)$ is the quantized version of the prediction error,

$\hat{s}(k)$ is the reconstructed output.

$$e_q(k) = Q[s(k) - \tilde{s}(k)]$$

$$\hat{s}(k) = \tilde{s}(k) + e_q(k)$$

Adaptation of the predictor coefficients a_i and b_i follows the updating algorithms :

$$a_i(k) = \lambda_i\ a_i(k-1) + \mu_i\ sgn[e_q(k-1)]\ sgn[\hat{s}(k-1-i)]$$

$$i = 1,2; \qquad \lambda_1 = 511/512; \qquad \lambda_2 = 255/256; \qquad \mu_1 = \mu_2 = 0.008$$

$$b_i(k) = \lambda'_i\, b_i(k-1) + \mu'_i\, sgn[e_q(k-1)]\; sgn[e_q(k-1-i)]$$

$$i = 1 \text{ to } 6; \qquad \lambda'_i = 255/256\ ,\ and\ \mu'_i = 0.008\ for\ all\ i.$$

The coefficients of the all-pole predictor are further controlled, for stability reasons, by the following constraints :

$$-0.75 \leq a_2 \leq 0.97$$

$$|a_{1,max}| = 0.97 - a_2 \qquad |a_1| = min\{|a_1|, |a_{1,max}|\}$$

$$a_1 = |a_1|\ sgn[a_1]$$

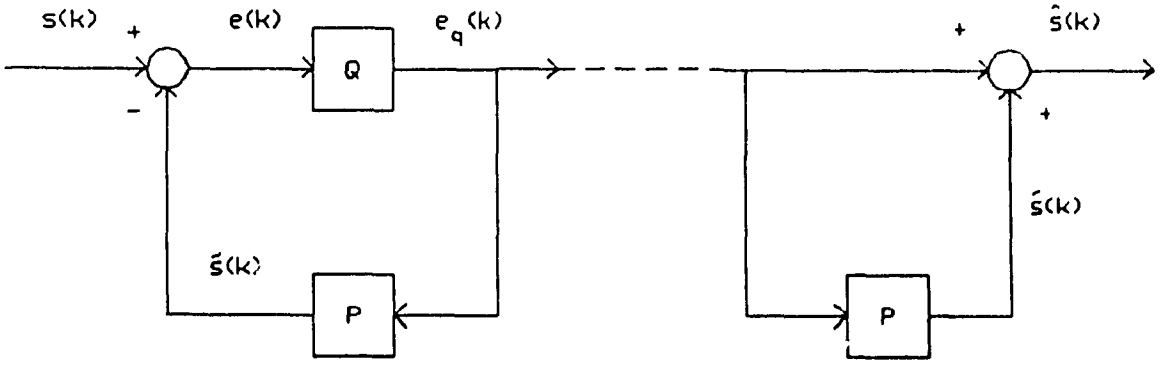


Fig.40. Cointot and De Passoz's all zero predictor.

An all-zero predictor has been described by Cointot and De Passoz [20], and Cointot [53]. The design is claimed to be robust to channel errors. The solution adopted consists in inverting the encoder-decoder structures so as to obtain an absolutely stable transversal filter at the receive end.

The predicted signal $\tilde{s}(k)$ is calculated from M reconstructed difference samples :

$$e(k) = s(k) - \sum_{i=1}^{M} a_i(k)\ e_q(k-i)$$

The coefficient updating algorithm is as follows :

$$a_i(k+1) = \beta_1 \{a_i(k) + \alpha_1\ e_q(k)\ e_q(k-i)\}$$

where α_1 is a constant,

β_1 is a leakage parameter ($\beta_1 \approx 1$) allowing the transmitting and receiving ends to fully resynchronize shortly after errors have occurred over the transmission line.

A first-order prewhitening filter P_2 which is also adaptive is added in order to improve system behavior with respect to low-frequency signals.

$$\tilde{s}_2(k) = b(k)\ \hat{s}(k)$$

$$b(k+1) = \beta_2 \{b(k) + \alpha_2\ \hat{s}(k)\ (e_q(k) + \tilde{s}_1(k))\}$$

This poleless filter improves the predictor performance for speech signals, while not altering the stability of the decoder.

Observing that predictors with zeros do not give rise to mistracking, Millar and Mermelstein [47] proposed representing the transfer function of the two-pole filter $H_p(z)$ in terms of an infinite number of zeros :

$$H_p(z) = (1 - \sum_{i=1}^{2} a_i(k) z^{-i})^{-1} \qquad = \qquad 1 + \sum_{i=1}^{\infty} c_i(k) z^{-i}$$

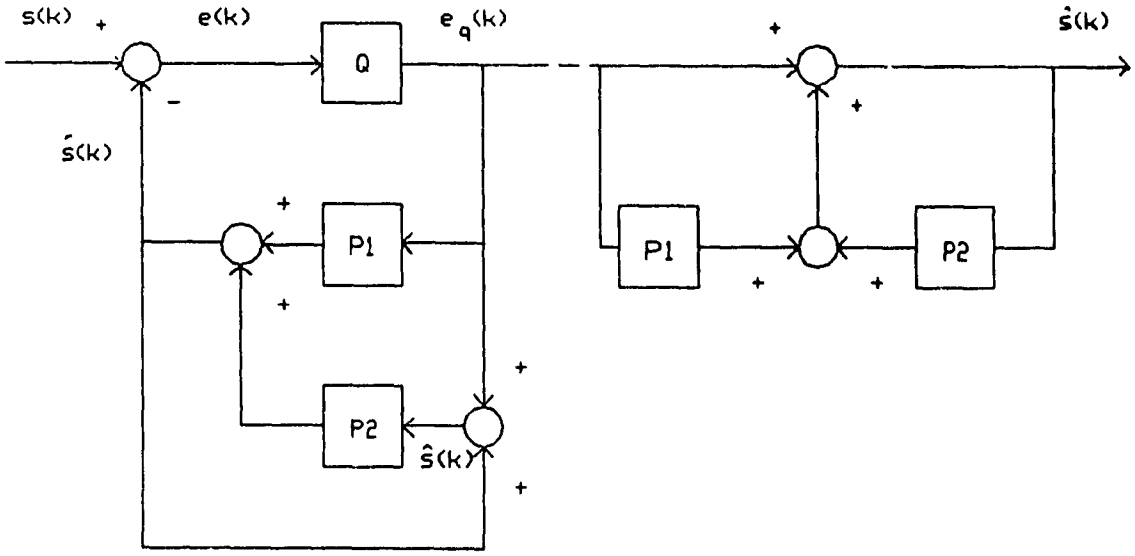


Fig.41. All zero predictor with prewhitening filter [53].

The coefficients c_i of an M^{th}-order zero-based prediction filter with quantized residual samples $\hat{e}(k)$ as input are adapted according to :

$$c_i(k+1) = (1-\delta)\ c_i(k) + \frac{\alpha'\ \hat{e}(k)\ \hat{e}(k-i)}{\sum_{i=1}^{M} \hat{e}^2(k-i)} \tag{26}$$

where δ is a small positive constant to ensure that the predictor has finite memory,

α' is a positive constant chosen by experiment.

Eq. (26) can be approximated as

$$c_i(k+1) = (1-\delta)\ c_i(k) + \alpha\ sgn[\hat{e}(k)]\ sgn[\hat{e}(k-i)]$$

In order to modify the pole coefficients so that the modified transfer function approximates the modified zero-based transfer function that results from appropriate updates to c_i, $i = 1,2$ when $\delta = 0$, we seek $\Delta a_i(k)$ such that

$$[1 - \sum_{i=1}^{2}(a_i(k) + \Delta a_i(k))z^{-i}]^{-1} \cong 1 + \sum_{i=1}^{2}(c_i(k) + \Delta c_i(k))z^{-i} + \sum_{i=3}^{\infty} c_i^*(k)z^{-i}$$

By equating like powers of z, a set of predictor updates for a_i is derived from :

$$\Delta a_1(k) = \Delta c_1(k)$$

$$\Delta a_2(k) = \Delta c_2(k) - 2a_1(k)\ \Delta c_1(k)$$

which is as follows :

$$a_1(k+1) = a_1(k)\ (1-\delta) + \alpha\ sgn[\hat{e}(k)]\ sgn[\hat{e}(k-1)]$$

$$a_2(k+1) = a_2(k)\ (1-\delta) + \alpha\Big\{sgn[\hat{e}(k)]\ sgn[\hat{e}(k-2)] - 2a_1(k)\ sgn[\hat{e}(k)]\ sgn[\hat{e}(k-1)]\Big\}$$

Millar and Mermelstein have simulated a 2-pole 6-zero predictor ADPCM structure using a dynamic locking quantizer as suggested by Petr [46]. The zero predictor has the transfer function

$$H_z(z) = \sum_{i=1}^{6} b_i(k)z^{-i}$$

The adaptation of the pole predictor is driven by the sum of the quantized residual signal $\hat{e}(k)$ and its zero-based reconstruction

$$\hat{e}(k) + \sum_{i=1}^{6} b_i(k)\ \hat{e}(k-i)$$

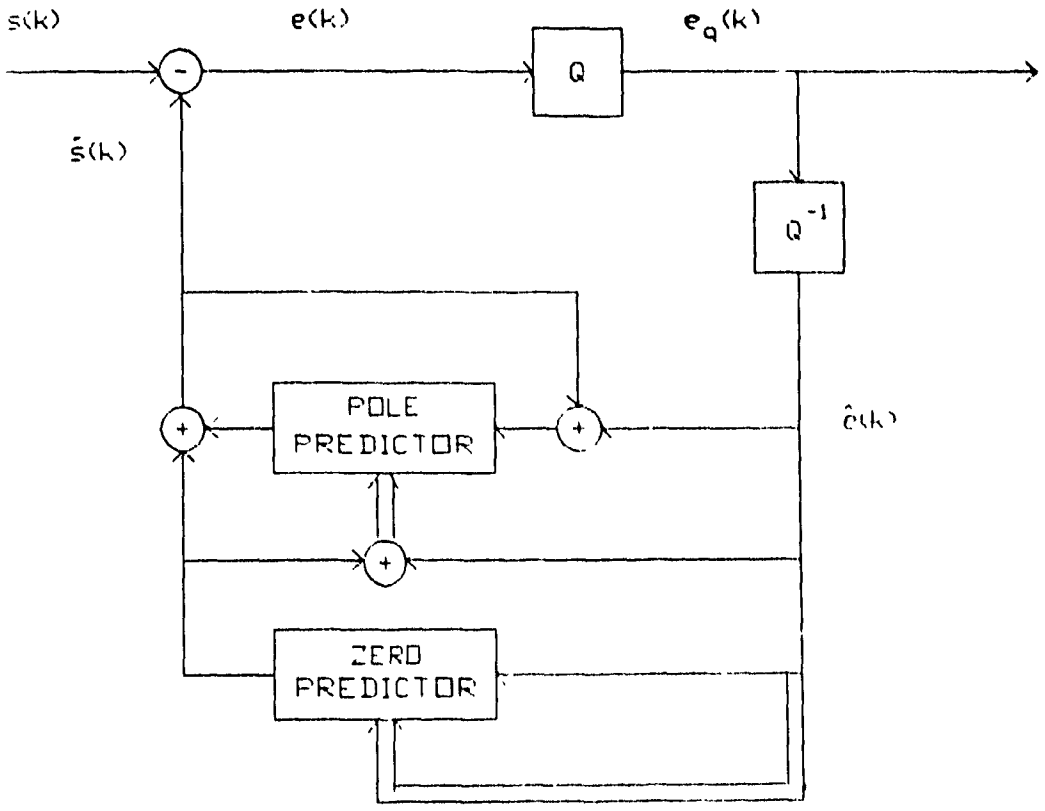


Fig.42. Predictor with feedback driven by quantized residual signal and zero-based reconstruction [47].

Simulation results revealed that pole adaptation based on quantized residual signal and its zero-based reconstruction can prevent predictor mistracking, with a small increase in complexity of the pole update equations.

This coefficient update algorithm has been used for the adaptive predictor in the CCITT 32 kbits/s ADPCM.

Johnson, Lyons, and Heegard [54], and Heegard, Johnson, and Lyons [55] suggested an algorithm in which the predictor is a linear, autoregressive, moving average (ARMA) filter of fixed degree. The coefficients of this filter are adapted by a recursive maximum likelihood (RML) algorithm.

Let $s(k)$ be the real-valued symbol of a source at time k, and

$$\tilde{s}(k) = \sum_{i=1}^{L} b_i \, c_q(k-i) + a_i[\tilde{s}(k-i) + c_q(k-i)]$$

be a prediction of $s(k)$.

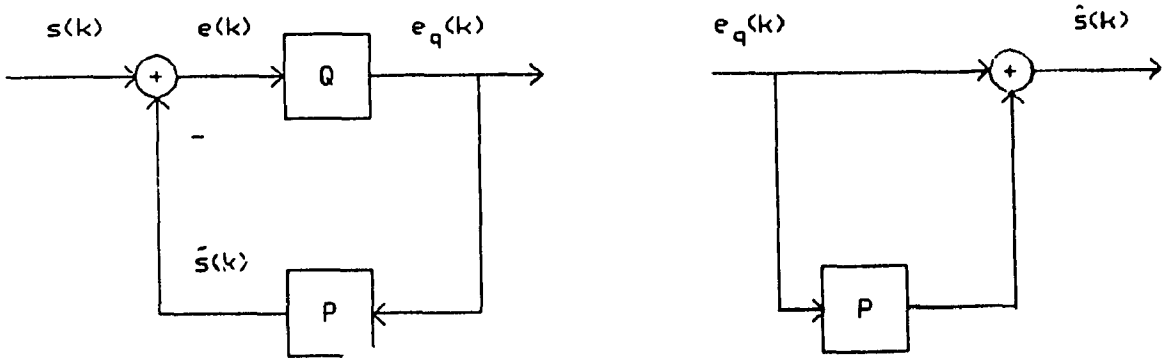


Fig.43. RML-based ADPCM.

This prediction is the output of an ARMA filter of order L with input $e_q(k) = Q[e(k)]$ where $e(k) = s(k) - \tilde{s}(k)$ is the prediction error.

The current prediction $\tilde{s}(k)$ consequently is a linear combination of the L past prediction $\tilde{s}(k-i)$ and the L past quantized prediction errors.

Denoting $\theta(k)$ as the parameter estimate vector :

$$\theta(k-1) = [b_1(k-1), b_2(k-1), \cdots, b_L(k-1), a_1(k-1), a_2(k-1), \cdots, a_L(k-1)]^T$$

$\phi(k)$ as the information vector :

$$\phi(k) = [e_q(k-1), \cdots, e_q(k-L), \tilde{s}(k-1) + e_q(k-1), \cdots, \tilde{s}(k-L) + e_q(k-L)]^T$$

$P(k)$ as the direction (or covariance) matrix,

$\psi(k)$ as an autoregressively filtered version of the information vector,

$0 < \lambda \leq 1$ as the data forgetting factor,

the authors described the predictor coefficient adaptation algorithm as follows :

1. Compute a priori estimate $\bar{s}(k) = \phi^T(k)\theta(k-1)$
2. Measure $\bar{e}_q(k) = Q[s(k) - \bar{s}(k)]$
3. Update $\psi(k)$:

$$\psi(k) = \phi(k) - \sum_{i=1}^{L} b_i(k-1)\psi(k-i)$$

4. Update $P(k)$:

$$P(k) = \frac{1}{\lambda(k)}\left[P(k-1) - \frac{P(k-1)\ \psi(k)\ \psi^T(k)\ P(k-1)}{\lambda(k) + \psi^T(k)\ P(k-1)\ \psi(k)}\right]$$

5. Update $\theta(k)$:

$$\theta(k) = \theta(k-1) + P(k)\ \psi(k)\ \bar{e}_q(k)$$

6. Compute a posteriori estimates

$$\tilde{s}(k) = \phi^T(k)\ \theta(k)$$

$$e_q(k) = \bar{e}_q(k)[1 - \phi^T(k)\ P(k)\ \psi(k)]$$

7. $k = k + 1$; go to step 1.

At step 5 of the algorithm, a stability check is made on the roots of the polynomials $1 - A(z)$ and $1 + B(z)$ where

$$A(z) = \sum_{i=1}^{L} a_i z^{-i}$$

$$B(z) = \sum_{i=1}^{L} b_i z^{-i}$$

If $1 + B(z)$ or $1 - A(z)$ is found to be unstable, the coefficients of $B(z)$ or $A(z)$ are adjusted to project the roots of $1 + B(z)$ or $1 - A(z)$ into the unit circle.

The stability and projection of $1 + B(z)$ or $1 - A(z)$ is required for the boundedness of $e_q(k)$ and the stable generation of $\psi(k)$ from $\phi(k)$.

Simulation results showed that this ADPCM scheme based on the RML predictor can converge to the optimum solution.

Gibson, Jones, and Melsa [25] studied ADPCM systems using Kalman and stochastic approximation algorithms.

Both algorithms try to sequentially minimize the square of the residual error which is defined as :

$$[\hat{e}(k)]^2 = [s(k) - \hat{S}_N^T(k-1)\ A(k-1)]^2$$

where $A(k)$ is a vector of predictor coefficients,

N is the number of predictor coefficients,

$$\hat{S}_N(k) = [\hat{s}(k-1), \hat{s}(k-2), \cdots, \hat{s}(k-N)]$$

Taking the derivative of $[\hat{e}(k)]^2$ with respect to $A(k)$ yields

$$\frac{\partial[\hat{e}(k)]^2}{\partial A(k-1)} = -2\ \hat{S}_N^T(k-1)[s(k) - \hat{S}_N^T(k-1)\ A(k-1)] \qquad (27)$$

To minimize $[\hat{e}(k)]^2$, $A(k)$ is corrected in the direction of the negative of the gradient given by Eq. (27). Consequently, the coefficients are updated according to :

$$A(k+1) = A(k) + K(k+1)\ \hat{e}(k+1)$$

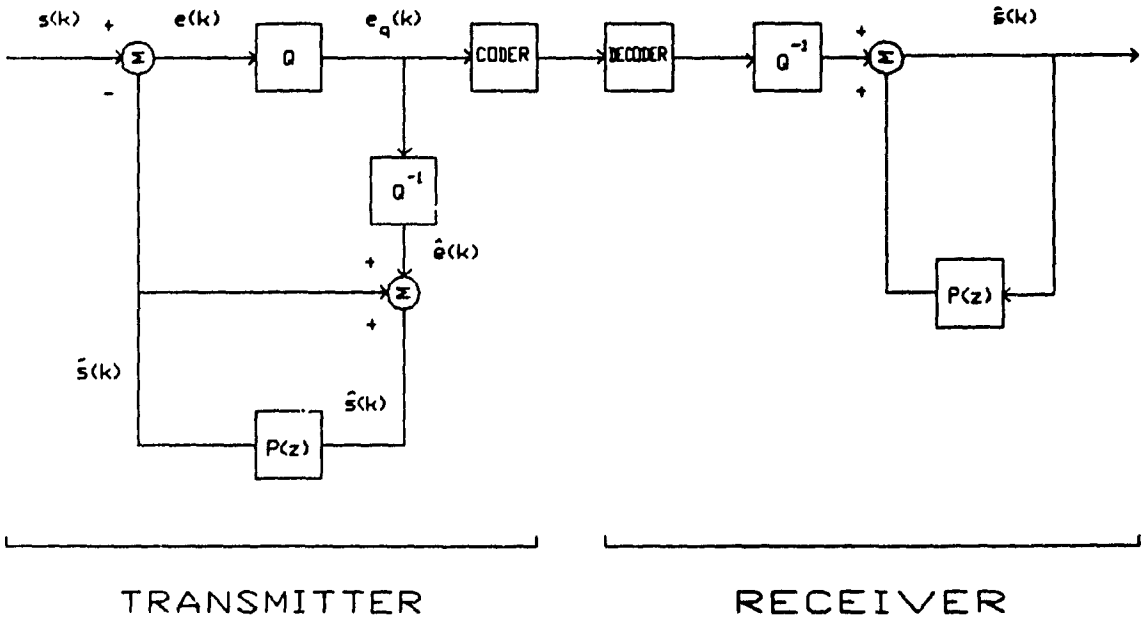


Fig.44. Transmitter and receiver of ADPCM system. (Gibson, Jones, and Melsa [25].)

The gain $K(k)$ can be chosen as either the stochastic approximation gain

$$K(k) = \frac{g\ \hat{S}_N(k-1)}{100 + \hat{S}_N^T(k-1)\ \hat{S}_N(k-1)}$$

where g is a scalar gain selected to optimize system performance,

g was chosen to be 0.08,

$A(0) = 0$,

$N = 4$

or the Kalman filter gain :

$$K(k) = V_a(k)\ \hat{S}_N(k-1)[\hat{S}_N^T(k-1)\ V_a(k)\ \hat{S}_N(k-1) + V_v]^{-1}$$

where $V_v = 100$,

$N = 4$,

$V_a(k)$ is a symmetric N by N matrix representing the covariance matrix of the error between the "true" value and the best estimate of the predictor coefficients. $V_a(k)$ has the initial value $V_a(0) = diag(0.01)$, and evolves according to

$$V_a(k) = [I - K(k-1)\ \hat{S}_N^T(k-2)]V_a(k-1) + V_w$$

with I, the identity matrix,

V_w is an $N \times N$ matrix which has the value selected by experiment and is related to the steady state value of the gain.

It was chosen to be diag(10^{-7}).

From simulation results, the Kalman filter performs better than the stochastic approximation algorithm.

In another study, Gibson [27] concluded that the Kalman predictor maintains a significant advantage over the adaptive gradient algorithm (which is the stochastic approximation algorithm described above) for all bit rates from 12.8 to 32 kbits/s. Indeed, Kalman predictor outperforms the adaptive gradient predictor by 1.57 to 3 dB.

Gibson and Berglund [35] compared ADPCM systems using a second-order fixed-tap predictor and PCQ (as described by Qureshi and Forney [44]) with those using Kalman predictor coefficient adaptation and PCQ. System simulation results indicated that the latter provides an increase in average SNR of approximately 2 dB over the former.

Gibson, Berglund, and Sauter [29] modified PCQ to eliminate the oscillations during silent intervals. They found that the combination of a Kalman predictor with a modified PCQ in ADPCM produce high-quality output speech.

Reininger and Gibson [37], and Gibson and Reininger [56] studied several transversal and lattice predictors. Two transversal algorithms, a gradient algorithm and a Kalman algorithm, were considered. The Kalman adaptive algorithm is similar to what has been described (see pp. 86-88) except that a damping term w has been introduced in the calculation of $V_a(k)$.

$$V_a(k) = w^{-1}[I - K(k-1)\ \hat{S}_N^T(k-2)]V_a(k-1)$$

The gradient or least mean square (LMS) transversal algorithm is given by:

$$K(k) = \frac{\hat{S}_N(k-1)}{10 + D(k-1)}$$

where $D(k)$ is the exponentially weighted sum of the squared reconstructed sequence, and acts as the automatic gain control in the algorithm.

$$D(k) = w\ D(k-1) + \hat{s}^2(k)$$

$$D(0) = 0$$

The value "10" in the denominator prevents a possible division by zero.

As can be observed, both algorithms have a damping term w, which exponentially discounts past data, that allows the algorithms to track the time variation in the input parameters. This damping factor is normally less than but very close to 1.

The update for the transversal coefficients is of the form :

$$A(k+1) = v\ A(k) + (1-v)\ A^* + K(k+1)\ \hat{e}(k+1)$$

where A^* is a constant, target set of predictor coefficients, A^* was chosen to be [1.515 $\quad -0.752$].

The damping factor v allows the system to forget past values of the coefficients, and causes the coefficients to decay to the fixed set A^* during

silent periods when $\hat{e}(k) = 0$. Without damping ($v = 1$), it is possible for the receiver to go unstable after only one transmission error.

The lattice methods have been described in section 2.2.1.3. of this thesis. Two adaptive lattice algorithms were simulated by the authors. The gradient or least mean square (LMS) lattice algorithm is defined by :

$$K^{(i)}(k) = \frac{K'^{(i)}(k)}{D^{(i)}(k)}$$

where $K'^{(i)}(k+1) = w\ K'^{(i)}(k) + v\ e^{(i)}(k)\ b^{(i)}(k-1)$

$D^{(i)}(k+1) = w\ D^{(i)}(k) + e^{(i)^2}(k)$

v is a damping factor to improve the system noisy channel performance,

w is an exponential fading memory term.

Being more complex but more precise, the least squares (LS) lattice algorithm differentiates backward and forward coefficients $K'^{(i)b}(k)$ and $K'^{(i)f}(k)$. The algorithm is as follows :

$$K'^{(i)}(k) = w\ K'^{(i)}(k-1) + \frac{e^{(i)}(k)\ b^{(i)}(k-1)}{1 - \gamma^{(i-1)}(k-1)}$$

$$K^{(i)f}(k) = \frac{K'^{(i)}(k)}{R^{(i)f}(k)}$$

$$K'^{(i)b}(k) = \frac{K'^{(i)}(k)}{R^{(i)b}(k-1)}$$

$$R^{(i+1)f}(k) = \left[R^{(i)f}(k) - \frac{K'^{(i)^2}(k)}{R^{(i)b}(k-1)}\right] v^{-2}$$

$$R^{(i+1)b}(k) = \left[R^{(i)b}(k-1) - \frac{K'^{(i)^2}(k)}{R^{(i)f}(k)}\right] v^{-2}$$

$$\gamma^{(i)}(k-1) = \gamma^{(i-1)}(k-1) + \frac{b^{(i)^2}(k-1)}{R^{(i)b}(k-1)}$$

v is a damping factor, for stability and tracking for a noisy channel,

$R^{(i)f}$ and $R^{(i)b}$ represent the weighted sum of prediction error for the i-stage forward and backward predictors, respectively.

The recursions are started at $i = 0$ with the initial conditions :

$$e^{(0)}(k) = b^{(0)}(k) \;=\; \hat{s}(k)$$

$$R^{(0)b}(k) = R^{(0)f}(k) = w\, R^{(0)f}(k-1) + e^{(0)^2}(k-1)$$

$$\gamma^{(-1)}(k) = 0$$

The recursions are performed in the order listed for each stage i, and are followed by the prediction error updates

$$e^{(i)}(k) = e^{(i-1)}(k) - K^{(i)b}(k)\; b^{(i-1)}(k-1)$$

$$b^{(i)}(k) = b^{(i-1)}(k-1) - K^{(i)f}(k)\; e^{(i-1)}(k)$$

Simulation results indicated that adaptive lattice predictors are less sensitive to channel errors than the adaptive transversal predictors. The adaptive lattice predictors are shown to have better performance for both noiseless and noisy channels.

Making use of the same transversal and lattice predictors, Honig and Messerschmitt [57] reported that the difference in system performance using the different adaptive algorithms is negligible. Their results even indicated that in the context of ADPCM, the extra computational burden associated with more complex adaptive linear prediction algorithms outweights the accompanying improvement in system performance. Hence, the predictor having the simplest implementation might be the best!

The ADPCM-DLQ described by Petr [46] utilized an adaptive transversal predictor with four taps. The tap coefficients a_i update is a simplified version of the adaptive gradient algorithm :

$$a_i(k+1) = l^i\; a_i(k) + c\; sgn[e_q(k)]\; sgn[\hat{s}(k-i)] \qquad\qquad i = 1, \cdots, 4$$

where $sgn[\,.\,]$ is the signum function,

$l < 1$ introduces finite memory to mitigate the effect of transmission errors,

c is a gain factor controlling the speed of adaptation.

Speech and VBD performance of the coder shows that a single 32 kbits/s ADPCM-DLQ encoding is really equivalent to a single 64 kbits/s PCM encoding. Moreover, in the presence of digital transmission errors, ADPCM-DLQ is actually rated better than PCM.

Evci, Steele, and Xydeas [45] described two adaptive predictors used in their speech coders. The schematic diagram representing both types of adaptive predictors is shown in Fig.45.

The difference between the predictors resides in the updating of the coefficients a_1 and a_2. The initial values of a_1 and a_2 are the long-time average autocorrelation function of the speech signal with a lag of one sampling instant, and zero, respectively.

The predicted output at the k^{th} instant is :

$$\tilde{s}(k) = a_1(k)\ \hat{s}(k-1) + a_2(k)\ \hat{s}(k-2) \tag{28}$$

and the decoded output is

$$\hat{s}(k) = \tilde{s}(k) + e_q(k) \tag{29}$$

The stochastic approximation predictor (SAP), also called adaptive gradient predictor, has the two coefficients updated according to :

$$a_1(k) = a_1(k-1) + P\ e_q(k-1)\ \hat{s}(k-2) \tag{30}$$

$$a_2(k) = a_2(k-1) + P\ e_q(k-1)\ \hat{s}(k-3) \tag{31}$$

where P is introduced to control the convergence rate of the predictor

$$P = \frac{A}{B + (\Delta/\alpha)^2} \tag{32}$$

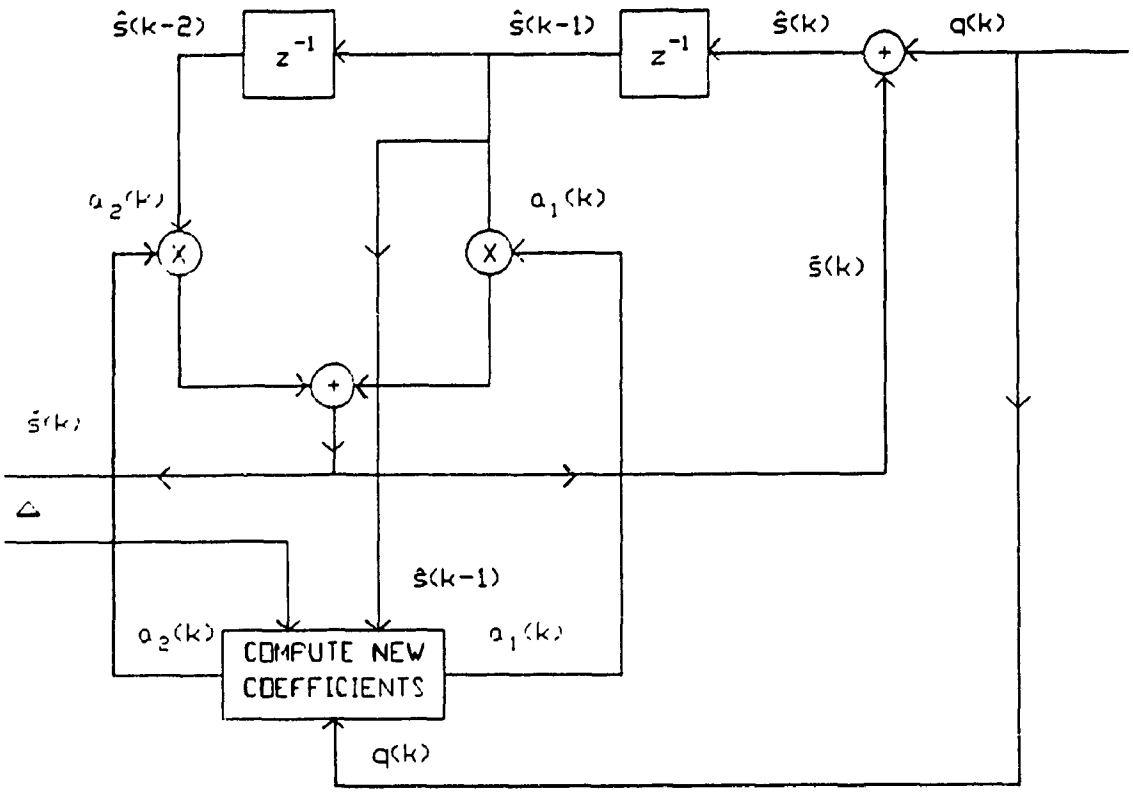


Fig.45. Two-coefficient adaptive predictor [45].

A and B are system parameters, and are known at the receiver,

Step size Δ is transmitted.

The quantized prediction error at the $(k-1)^{th}$ instant is

$$e_q(k-1) = Q[s(k-1) - \tilde{s}(k-1)] \tag{33}$$

Predictor coefficients $a_1(k)$ and $a_2(k)$ are computed using Eq. (30) - (33), enabling $\tilde{s}(k)$ and thence $\hat{s}(k)$ to be found using Eq. (28) and (29), respectively.

The sequential gradient estimation predictor (SGEP) attempts to minimize an error function G. In order to update each coefficient, two values of G are

determined. Then, the difference between the two G values associated with a particular coefficient is calculated.

$$\delta_i(k) = G_{2i-1}(k) - G_{2i}(k) \qquad (34)$$

$$i = 1, 2, \cdots, N$$

And this $\delta_i(k)$ is used to form the i^{th} predictor coefficient at the $(k+1)^{th}$ sampling instant according to

$$a_i(k+1) = a_i(k) - \delta_i(k)\,\frac{P(s)}{i^{\gamma}} \qquad (35)$$

$$i = 1, 2, \cdots, N$$

where

$$P(s) = \frac{A}{B + \dfrac{1}{M}\displaystyle\sum_{l=k-1-M}^{k-1} s^2(l)}$$

A and B are constants,

By setting γ to a value less than unity, good prediction is obtained.

The function $P(s)$ is to attempt to equalize the adaptation rate of the prediction algorithm to the variation in the mean square value σ_s^2 of the immediate past M samples. Thus, when σ_s^2 increases, the second term in Eq. (35) is reduced, and overcorrections of the $a_i(k+1)$ coefficient avoided, preventing the occurrence of a large prediction error.

The constant B maintains a finite value of $P(s)$ during silent intervals. The values of A, B, and M are different for SAP and SGEP.

$\delta_i(k)$ plays an important role in this algorithm, and is determined as follows. The first coefficient $a_1(k)$ in the coefficient vector $\{a_i(k)\}$ is increased by a positive number $c_i(k)$ calculated from

$$c_i(k) = \frac{1}{D_i^{\beta}} \qquad i = 1, 2, \cdots, N$$

where $i = 1$, $D > 1$, and $\beta < 1$. With the other coefficients unchanged, a predicted output $\tilde{s}_1(k)$ is obtained. Then, $a_1(k)$ is decreased by $c_1(k)$, and another predicted output $\tilde{s}_2(k)$ obtained. Modifying $a_2(k)$ by $\pm c_2(k)$ yields $\tilde{s}_3(k)$ and $\tilde{s}_4(k)$. This process of modifying sequentially one term in $\{a_i(k)\}$ by $\pm c_i(k)$, $i = 1, 2, \cdots, N$ continues to yield a predictive sequence $\{\tilde{s}(k)\}$ having $2N$ components. The prediction error between the input sample $s(k)$ and each of the $2N$ predicted values in $\{\tilde{s}(k)\}$ is then found. The error function G, chosen to be the absolute error criterion $\{G(k)\} = \{|e(k)|\}$, has consequently $2N$ values :

$$G_{2i-1}(k) = |s(k) - \tilde{s}_{2i-1}(k)|$$

$$G_{2i}(k) = |s(k) - \tilde{s}_{2i}(k)| \qquad i = 1, 2, \cdots, N$$

By substituting $G_{2i-1}(k)$ and $G_{2i}(k)$ into Eq. (34), N values of $\delta_i(k)$ are found. These $\delta_i(k)$ values are the difference between two prediction errors, where the prediction sample in each case was found by either adding or subtracting a constant $c_i(k)$ to the $a_i(k)$ coefficient. Should $G_{2i-1}(k) > G_{2i}(k)$, $\tilde{s}_{2i}(k)$ is a better prediction than $\tilde{s}_{2i-1}(k)$, $\delta_i(k) > 0$ and consequently $a_i(k+1) < a_i(k)$. When $G_{2i-1}(k) < G_{2i}(k)$, $a_i(k+1)$ is increased, as $\delta_i(k) < 0$. Experiments revealed that the improvement in SNR of ADPCM when SGEP is used, compared to SAP, is approximately 3.5 dB.

The same authors also studied DPCM-AQF [30] with switched predictor, which can be either fixed second-order predictor (FSOP) or SGEP. In both cases, the correlation coefficient of the speech signal is determined and compared with a set of thresholds T_j, $j = 1, 2, \cdots, z$ which divide the range of this coefficient, namely -1 to +1, into $(z + 1)$ zones. A unique set of prediction coefficients is assigned to each zone, thereby ensuring that a high prediction gain is achieved for the block of speech samples being encoded. The prediction

coefficients are also stored at the receiver; hence, they are not transferred. Only the value of j needs to be transmitted, using a word of $\log_2(z+1)$ bits. Table III provides the values of the coefficients a_1 and a_2 for the 3-order correlation switch predictor (CSP). For SGEP, the prediction coefficients of each zone given in Table III constitute the initial values of the SGEP prediction coefficients for a block of w samples, thereby facilitating a faster coefficient convergence rate. For $z = 3$, the DPCM-AQF/SGEP codec using 3-order CSP gave overall performance better than SGEP by itself, and FSOP with 3-order CSP.

j	Threshold T_j	Correlation Zone	Coefficient a_1	Coefficient a_2
1	0.7	0.7 to 1.0	1.53	-0.72
2	0.4	0.4 to 0.7	0.95	-0.34
3	0.0	0.0 to 0.4	0.48	-0.21
		-1.0 to 0.0	-0.63	-0.36

TABLE III. Coefficients for 3-order CSP.

Adoul, Debray, and Dalle [58] studied intermediate solutions between fixed and adaptive prediction. The proposed ADPCM system uses forward adaptive prediction which consists of a finite number L of preselected predictors. These L predictors are assumed to be of an identical structure : linear and of the same order M :

$$e(k) \quad = \quad s(k) - \tilde{s}(k) \quad = \quad \sum_{i=0}^{M} a_i \, s(k-i) \qquad\qquad a_0 \triangleq 1$$

Consider a reference spoken sentence composed of N_B blocks of N samples. Let S be the set of the ordered blocks B_j of the reference sentence :

$$S = B_j \; \{ \; j = 1, 2, \cdots, N_B \; \}$$

For a given block B_j, one out of L predictors A_i which offers the minimum residual energy $\delta_j(i)$ will be selected. The set S can be partitioned in L subsets S_i in the following manner :

$$S_i = \{ B_j | \ \delta_j(i) < \delta_j(k) \ ; \ \forall i, \ k \} \tag{36}$$

Let E_i be the total contribution to the residual energy from blocks of subset S_i

$$E_i = \sum_{B_j \in S_i} \left\{ \sum_{k=0}^{M} \sum_{l=0}^{M} a_k^i \ a_l^i \ R_j(| \ k-l \ |) \right\}$$

$$\text{or} \qquad E_i = \sum_k \sum_l a_k^i \ a_l^i \left\{ \sum_{B_j \in S_i} R_j(| \ k-l \ |) \right\}$$

where R is the autocorrelation function of the input process.

For a given S_i, the conditions for minimizing E_i are obtained by differentiation technique, yielding a solution :

$$(37) \begin{cases} a_0^i & = 1 \\ \sum_{l=0}^{M} a_l^i \sum_{B_j \in S_i} R_j(| \ k-l \ |) & = 0 \qquad ;k = 1, M \end{cases}$$

The filter A_i which is the solution to the above equations can be seen as the best filter for class S_i. It is referred to as the centroid filter for set S_i.

The algorithm to select the best predictor among the given set is as follows :

1. Initial choice : a set P of L predictors.

2. Determination of the partition of the sentence in L classes according to Eq. (36).

3. Determination of the L centroid filters for the classes of step 2, using Eq. (37).

4. Compute $E(i)$ where i denotes the i^{th} iteration. If $| \ E(i) - E(i-1) \ |$ is less than some threshold, stop the procedure, otherwise go to step 2.

This approach was described as drastically reducing the amount of forward information required to update the predictor coefficients since only the index of the predictor needs to be sent, namely $\log_2 |L|$ bits of forward side information. The relative prediction-gain improvement has been computed for 3-second speech sample, and for several values of L, M and block size. Results showed that more than 1/2 of the adaptive over fixed-prediction improvement in dB is reached with only $L = 4$, and 2/3 with $L = 8$.

O'Neal and Stroh [59] proposed several schemes for compromise predictors, which, though not optimum for any particular signal, perform better than PCM for a variety of speech and data signals.

Denoting σ_s^2 and σ_d^2 as the mean-square value of the resulting error sequences for speech and data, respectively, the two authors suggested optimizing the predictor in several ways :

- minimize $\delta\sigma_s^2 + (1 - \delta)\sigma_d^2$ where $0 \leq \delta \leq 1$. The quantity δ is called the compromise coefficient.

- minimize σ_s^2 or σ_d^2 subject to the constraint $\sigma_s^2 = \sigma_d^2$.

- minimize σ_s^2 or σ_d^2 or $\sigma_s^2 + \sigma_d^2$ subject to the constraint that $\sigma_s^2/\sigma_{s_{min}}^2 = \sigma_d^2/\sigma_{d_{min}}^2$ where $\sigma_{s_{min}}^2$ is the minimum value of σ_s^2 that could be obtained by designing the predictor to minimize σ_s^2 for speech regardless of the effect on data ($\sigma_{d_{min}}^2$ has a similar interpretation).

- minimize σ_s^2 subject to the constraint that σ_d^2 be some constant value.

Numerical results for the four compromise designs revealed the cost of the compromise, i.e. the degradation on performance suffered by the speech and data signals in a compromise system designed to accomodate both signals.

CHAPTER IV

IMPLEMENTATION OF ADPCM CODECS

The ADPCM algorithms described in chapter III represent only a small portion of those which have been reported in the literature. Nevertheless, from these we can already realize that more and more ADPCM algorithms have been proposed, more and more ADPCM systems realized and their performance evaluation reported.

In order to avoid chaos in the networks as a result of non-compatible systems being introduced, International Telegraph and Telephone Consultative Committee (CCITT) has formally approved a 32 kbits/s ADPCM algorithm as an international standard. At 32 kbits/s, ADPCM coder systems can transmit telephone signals at a quality comparable to that for 64 kbits/s PCM coders, and enable doubling the existing channel capacity for standard 64 kbits/s PCM transmission. The per-channel telephone transmission costs are therefore reduced.

Because of the complex nature of the ADPCM coding algorithm, it would not be possible to have economical designs of the encoder and decoder units with standard TTL chips. Still the computation rate required is often far beyond the ability of general purpose microprocessors. On the other hand, ADPCM implementation through custom-designed chips can be justified only if the production volume is large. With the advent of single chip signal processors, the implementation of ADPCM algorithm becomes feasible, efficient, and economic as these processors have special arithmetic functions, and allow flexibility for specification modifications. The second characteristic enables short development period.

4.1. CCITT ADPCM Algorithm

Within CCITT, the group that has the primary responsibilty for standardizing the speech encoding algorithm and recommending uniform performance characteristics is Study Group XVIII. During the period 1981-1984, the study on speech coding fell under the question : encoding of speech and voiceband signals using methods other than PCM in accordance with Recommendation G.711 [60]. In June 1982, Study Group XVIII chartered a small seven-country expert group to recommend a single 32 kbits/s coding algorithm as a candidate for international standardization. A number of contributions have been submitted from various Administrations. Three candidate codecs from France, Japan (NTT), and USA (AT&T) were submitted for evaluation. A fourth codec, from Canada, was withdrawn prior to testing. An analysis of the test results indicated the AT&T codec exhibits better overall performance than the other two. The codecs of AT&T and NTT employed similar algorithm structures. These organizations then worked in collaboration with the view of achieving a best performing "compromise algorithm". The French delegation offered to withdraw its algorithm from further assessment. After eighteen months of work, the expert group drafted an ADPCM algorithm which is defined as a digital transcoding technique from 64 kbits/s PCM. At the October 1984 plenary session, CCITT formally approved the algorithm as an international standard.

The algorithm, which employs both adaptive quantization and adaptive prediction, has been described in details elsewhere [61], [62]. It has been designed with several constraints taken into consideration :

- adequate transmission performance is to be maintained for both voice and VBD. This implies 8 kHz sampling and 4-bit per sample coding.

- transmission delay should be minimized through backward adaptation.

- the algorithm should enable encoder/decoder tracking recovery and elimination of cumulative distortion in synchronous tandem codings.

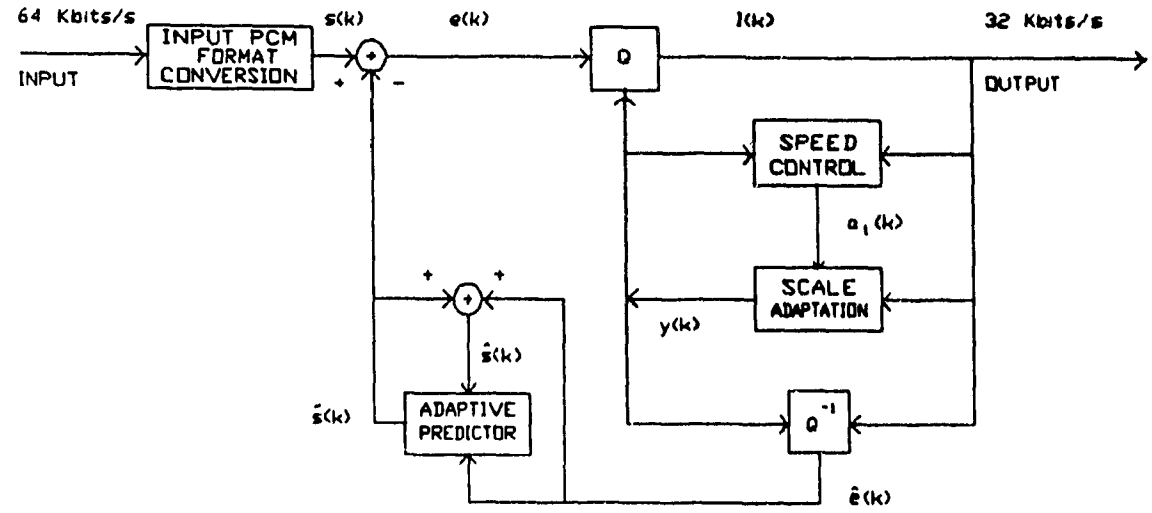
The overall block diagrams of the encoder and decoder are shown in Fig.46. The only differences between the encoder and decoder are found in the input and output PCM format conversion blocks, and the synchronous coding adjustment which appears only in the decoder.

In the encoder, the A-law or μ-law PCM input signal is converted into uniform PCM. A difference signal is then obtained by subtracting an estimate of the input signal from the input signal itself. An adaptive 16-level quantizer is used to assign four binary digits to the value of the difference signal for transmission to the decoder. An inverse quantizer produces a quantized difference signal from these same four binary digits. The signal estimate is added to this quantized difference signal to produce the reconstructed version of the input signal. Both the reconstructed signal and the quantized difference signal are operated upon by an adaptive predictor which produces the estimate of the input signal, thereby completing the feedback loop.

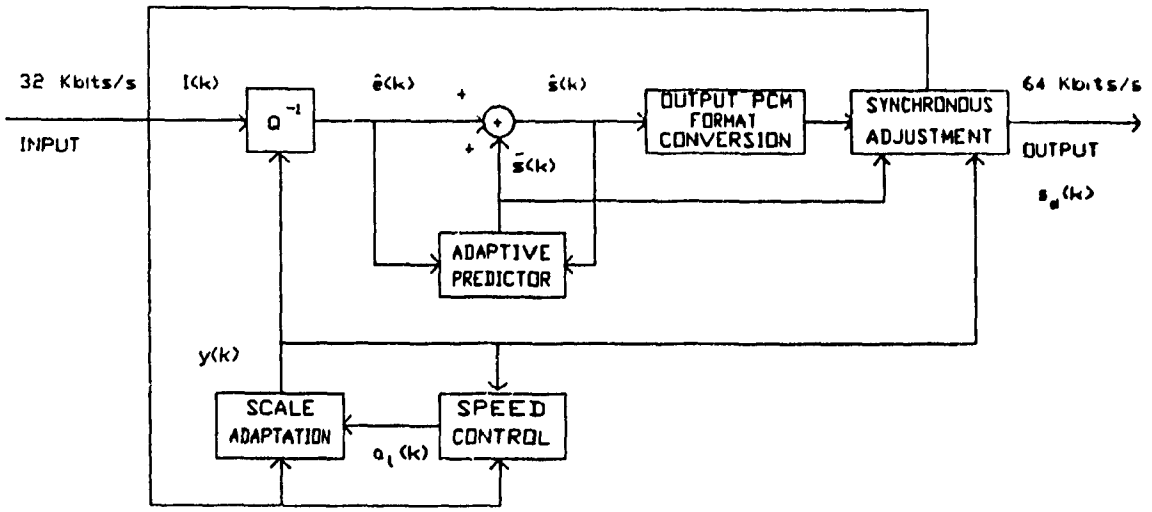
As mentioned above, the decoder includes a structure identical to the feedback portion of the encoder, together with a uniform PCM to A-law or μ-law conversion and a synchronous coding adjustment.

4.1.1. PCM Format Conversion

Conversion between 8-bit A- or μ-law PCM and 13-bit uniform PCM complies to CCITT Recommendation G.711 [60].



ENCODER BLOCK DIAGRAM



DECODER BLOCK DIAGRAM

Fig.46. CCITT ADPCM system.

4.1.2. Adaptive Quantizer

A sixteen level backward adaptive quantizer is used to quantize the difference signal $e(k)$, obtained by subtracting the signal estimate from the

uniform PCM input signal.

$$e(k) = s(k) - \tilde{s}(k) \tag{37}$$

The quantization characteristic is gaussian in nature. The decision and output levels used in the quantization process are basically taken from Max table [13] as shown below :

e(k)	$e_q(k)$
0.	0.1284
0.2582	0.3881
0.5224	0.6568
0.7996	0.9424
1.099	1.256
1.437	1.618
1.844	2.069
2.401	2.733

Table IV. Max quantizer input/output.

and normalized :

e(k)	$e_q(k)$
0.	0.5
1.0	1.5
2.02	2.54
3.09	3.65
4.25	4.86
5.56	6.27
7.14	8.01
9.29	10.58

Table V. Normalized quantizer input/output.

In order to maintain a wide dynamic range and minimize complexity, quantization and its adaptation are performed in the base 2 logarithmic domain. Hence table VI :

$\log_2 \lvert e(k) \rvert$	$\log_2 \lvert e_q(k) \rvert$	$\lvert I(k) \rvert$
$-\infty$	-1.05	0
-0.05	0.53	1
0.96	1.29	2
1.58	1.81	3
2.04	2.23	4
2.42	2.59	5
2.78	2.95	6
3.16	3.34	7

Table VI. Normalized quantizer input/output (log value).

The input to the quantizer is scaled by subtracting the logarithmic scale factor $y(k)$ (discussed in section 4.1.3. of this chapter) from $\log_2 |e(k)|$.

$$|I(k)| \leftarrow \log_2 |e(k)| - y(k) \tag{38}$$

4.1.3. Scale Factor Adaptation

It is desirable for $y(k)$ to change rapidly for difference signals with large fluctuations but slowly for difference signals with small fluctuations. This kind of bimodal adaptation is realized by forming a linear combination of fast and slow adaptive scale factors.

A fast scale factor $y_u(k)$ is computed from the present value of $y(k)$:

$$y_u(k) = (1 - 2^{-5})\, y(k) + 2^{-5}\, W[I(k)] \tag{39}$$

where $1.06 \leq y_u(k) \leq 10$,

the leak factor $(1 - 2^{-5})$ introduces finite memory to aid encoder/decoder tracking recovery following transmission errors, the discrete function $W[I(k)]$ is defined :

$\|I\|$	7	6	5	4	3	2	1	0
$W[I(k)]$	69.25	21.25	11.50	6.12	3.12	1.70	0.25	-0.75

The values of $W[I(k)]$ are actually calculated by taking 2 and raising it to the power of the multipliers,$M[I(k)]$, respectively.

$\|I\|$	7	6	5	4	3	2	1	0
$M[I(k)]$	4.482	1.585	1.283	1.142	1.070	1.037	1.006	0.984

A slow scale factor is then derived from $y_u(k)$:

$$y_l(k) = (1 - 2^{-6})\ y_l(k-1) + 2^{-6}\ y_u(k) \tag{40}$$

The scale factor $y(k)$ is formed by :

$$y(k) = a_l(k)\ y_u(k-1) + (1 - a_l(k))\ y_l(k-1) \tag{41}$$

where $0 \leq a_l(k) \leq 1$.

It can be observed that the derivation of $y(k)$ follows Petr's ADPCM-DLQ algorithm [46].

4.1.4. Adaptive Speed Control

The controlling parameter $a_l(k)$ is derived from a rate of change measure of the difference signal.

First, short and long term measures of $I(k)$ are computed :

$$d_{ms}(k) = (1 - 2^{-5})\ d_{ms}(k-1) + 2^{-5}\ F[I(k)] \tag{42}$$

$$d_{ml}(k) = (1 - 2^{-7})\ d_{ml}(k-1) + 2^{-7}\ F[I(k)] \tag{43}$$

where $F[I(k)]$ is defined by :

$\lvert I\rvert$	7	6	5	4	3	2	1	0
$F[I(k)]$	7	3	1	1	1	0	0	0

$F[I(k)]$ is a simple, empirically defined function that amplifies differences in the $I(k)$ sequence in order to reduce the transition time between fast and slow modes of adaptation.

The intermediate variable $a_p(k)$ is defined

$$a_p(k) = \begin{cases} [1 - 2^{-4}]\ a_p(k-1) + 2^{-3} & |d_{ms}(k) - d_{ml}(k)| > 2^{-3}\ d_{ml}(k) \\ & \text{or } y(k) < 3 \\ [1 - 2^{-4}]\ a_p(k-1) & \text{otherwise} \end{cases} \tag{44}$$

Finally, $a_l(k)$ is obtained :

$$a_l(k) = \begin{cases} 1 & a_p(k-1) > 1 \\ a_p(k-1) & a_p(k-1) \leq 1 \end{cases} \tag{45}$$

4.1.5. Adaptive Predictor

The adaptive predictor is composed of a sixth-order section that models zeros in the input signal, and a second-order section that models poles in the input signal. The pole-zero structures have been studied previously by Nishitani, Aikoh, Araseki, Ozawa, and Maruta [51], and Cointot [53].

The combination of both poles and zeros allows the filter to model more effectively any general input signal. The sixth-order all-zero section helps to stabilize the filter, and prevents it from drifting into oscillation. Millar and Mermelstein's coefficient update algorithm [47] is employed in the second-order section to minimize mistracking.

The signal estimate is computed by :

$$\tilde{s}(k) = \sum_{i=1}^{2} a_i(k-1)\ \hat{s}(k-i) + s_{ez}(k) \tag{46}$$

where the reconstructed signal is defined by :

$$\hat{s}(k-i) = \tilde{s}(k-i) + \hat{e}(k-i) \tag{47}$$

$$\text{with} \qquad s_{ez}(k) = \sum_{i=1}^{6} b_i(k-1)\ \hat{e}(k-i) \tag{48}$$

A simplified $sgn[\,].sgn[\,]$ gradient algorithm is used to update both sets of predictor coefficients.

$$b_i(k) = (1 - 2^{-8})\; b_i(k-1) + 2^{-7}\; sgn[\hat{e}(k)]\; sgn[\hat{e}(k-i)] \qquad (49)$$

$$i = 1, \cdots, 6$$

$$a_1(k) = (1 - 2^{-8})\; a_1(k-1) + 3\;.\;2^{-8}\; sgn[\hat{d}(k)]\; sgn[\hat{d}(k-1)] \qquad (50)$$

$$a_2(k) = (1 - 2^{-7})\; a_2(k-1) + 2^{-7}\Big\{sgn[\hat{d}(k)]\; sgn[\hat{d}(k-2)]$$

$$- f[a_1(k-1)]\; sgn[\hat{d}(k)]\; sgn[\hat{d}(k-1)]\Big\} \qquad (51)$$

where $$\hat{d}(k) = \hat{e}(k) + s_{ez}(k) \qquad (52)$$

$$f(a_1) = \begin{cases} 4\; a_1 & |a_1| \le 2^{-1} \\ 2\; sgn[a_1] & |a_1| > 2^{-1} \end{cases} \qquad (53)$$

with the stability constraints on a_1 and a_2 :

$$|a_2(k)| \le 0.75$$

$$|a_1(k)| \le 1 - 2^{-4} - a_2(k)$$

4.1.6. Synchronous Adjustment

The synchronous adjustment function prevents the occurrence of cumulative distortion in synchronous tandem codings (i.e. ADPCM-PCM-ADPCM-PCM ..., with no intermediate analog conversion), provided that transmission errors and digital signal processing devices are not found on the intermediate 32 and 64 kbits/s streams.

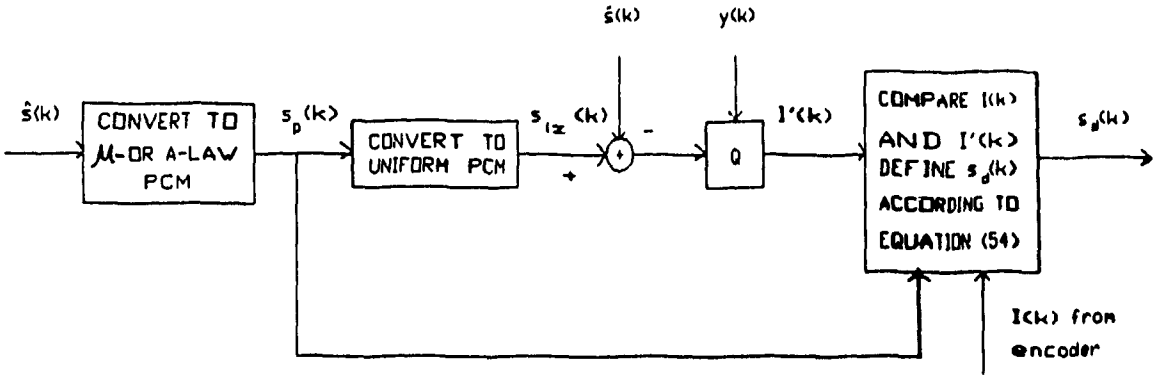


Fig.47. Synchronous coding adjustment diagram.

The processing that a PCM sample would undergo between a decoder and the ADPCM quantizer in the next encoder is simulated in the decoder before the PCM sample is transmitted. If that processing results in a different $I(k)$ value (than that received by the decoder), the output PCM sample is changed to eliminate the difference. This guarantees that values of all state variables in successive ADPCM stages are identical, hence no cumulative distortion. First, the A- or μ-law PCM decoder output signal $s_p(k)$ is converted to a uniform PCM signal $s_{lx}(k)$, and a difference signal is formed :

$$e_x(k) = s_{lx}(k) - \tilde{s}(k)$$

The PCM output signal $s_d(k)$ is produced by comparing this difference to the ADPCM quantizer decision level defined by the received $I(k)$ value and scaled by $y(k)$.

$$s_d(k) = \begin{cases} s_p^+(k) & e_x(k) < \text{lower interval boundary} \\ s_p^-(k) & e_x(k) \geq \text{upper interval boundary} \\ s_p(k) & \text{otherwise} \end{cases} \tag{54}$$

where $s_p^+(k)$ = the adjacent PCM codeword that represents the next more positive PCM output level,

$s_p^-(k)$ = the adjacent PCM codeword that represents the next more negative PCM output level.

Evidently, the CCITT ADPCM algorithm requires a great number of complex operations. The whole description of the algorithm includes 37 different sub-blocks, which represent about 250 equations, and some tables for the conversions. The format of the variables can be two's complement, signed magnitude, and floating point, from 1-bit to 19-bit word length [63]. In the algorithm, computational detail is strictly defined. However, strict specifications not only cover operation sequence orders in some arithmetic processing but also number representation in every variable. These were determined basically through block-by-block optimization for multichannel-oriented hardware. Thus, a different bit length and format are adopted for each computation. Exact computation must be executed in order to assure full compatibility. This is mainly due to the fact that in ADPCM system, only the differential information is sent from the encoder to the decoder, and the signal is reconstructed basically by an accumulation process. Therefore, a slight difference in computational detail may cause a large error in the reconstruction process [64].

4.2 Processing Requirements

Digital Signal Processors (DSP) are an emerging subset of general purpose microcomputers. Generally, their architectures are optimized for manipulating a large amount of data in a given amount of time, determined by the bandwidth of the signals to be processed. Speed and software support are two main features of DSP which stimulate researchers towards implementing digital signal algorithms using this kind of processors.

The CCITT ADPCM algorithm has been realized with or without some modifications. By some researchers, the LSI technology has been utilized to produce codecs [65], [66]. On the other hand, processors such as FDSP-3, TMS32010 ... have been reported as the DSP used in other systems [64], [63], [67]. Most of the implementations use one chip for encoder and another for decoder [64], [63], [67]. Recently, a full-duplex implementation of ADPCM on a TMS32010 has been detailed in [68]. Let us study this implementation and determine the possibilities of improving the system execution time. Thence, the structure of a software programmable signal processor for the implementation of CCITT ADPCM algorithm will be derived.

4.2.1. TMS32010 ADPCM Codec

In this implementation, the codec reads and writes μ-law PCM samples without using the synchronous coding adjustment recommended by CCITT.

4.2.1.1. Hardware

With a modified Harvard architecture where program memory and data memory lie in two separate spaces, the TMS32010 has an instruction cycle time of 200 ns. It utilizes off-chip program memory, and the maximum amount of program memory that can be directly addressed is $4K$ x 16-bit words.

Instructions in off-chip program memory are executed at full speed. There is on-chip data RAM of size 144 bytes. Its set of instructions has been described as speeding the execution of DSP algorithms.

The full-duplex implementation of ADPCM using TMS32010 has been described by Reimer, McMahan, and Arjmand [68]. The corresponding circuit was given in "Design Example : 32-kbit/s ADPCM Using The TMS32010" [69] and realized with some minor modifications by Vo [70].

The circuit block diagram of the implementation is shown in Fig.48. A detailed graph can be found in Appendix 1. The input signal is sampled at a rate of 8 kHz and converted into 8-bit μ-law PCM data by a codec (TCM2914). The serial output of the codec is converted into parallel form through the SN74ALS164 and stored in a buffer (SN74ALS574). At each interrupt, the TMS32010 reads the buffer and converts the 8-bit PCM data into a 4-bit ADPCM sample which is stored in another buffer, the SN74ALS574. The TMS32010 also reads in 4-bit ADPCM data and converts to 8-bit μ-law PCM data. The output PCM sample is converted into serial form through the SN74ALS165 and sent to the codec TCM2914 which converts it back to analog form.

The generation of several clock signals for different parts of the circuit is illustrated in Fig.49. A 20.48 MHz crystal and a SN74HC04 hex inverter provide a 20.48 MHz clock signal CLKIN which is fed to the clock input of the TMS32010. A SN74HC390 decade counter divides the signal down to 2.048 MHz which is used as the master clock (MCLK) for the codec. The 128 kHz data clock (DCLK) is obtained by dividing MCLK by 16. Finally, DCLK is further divided by 16 to provide the 8 kHz framing pulse $\overline{FP}$ and the frame synchronization pulse $\overline{FS}$.

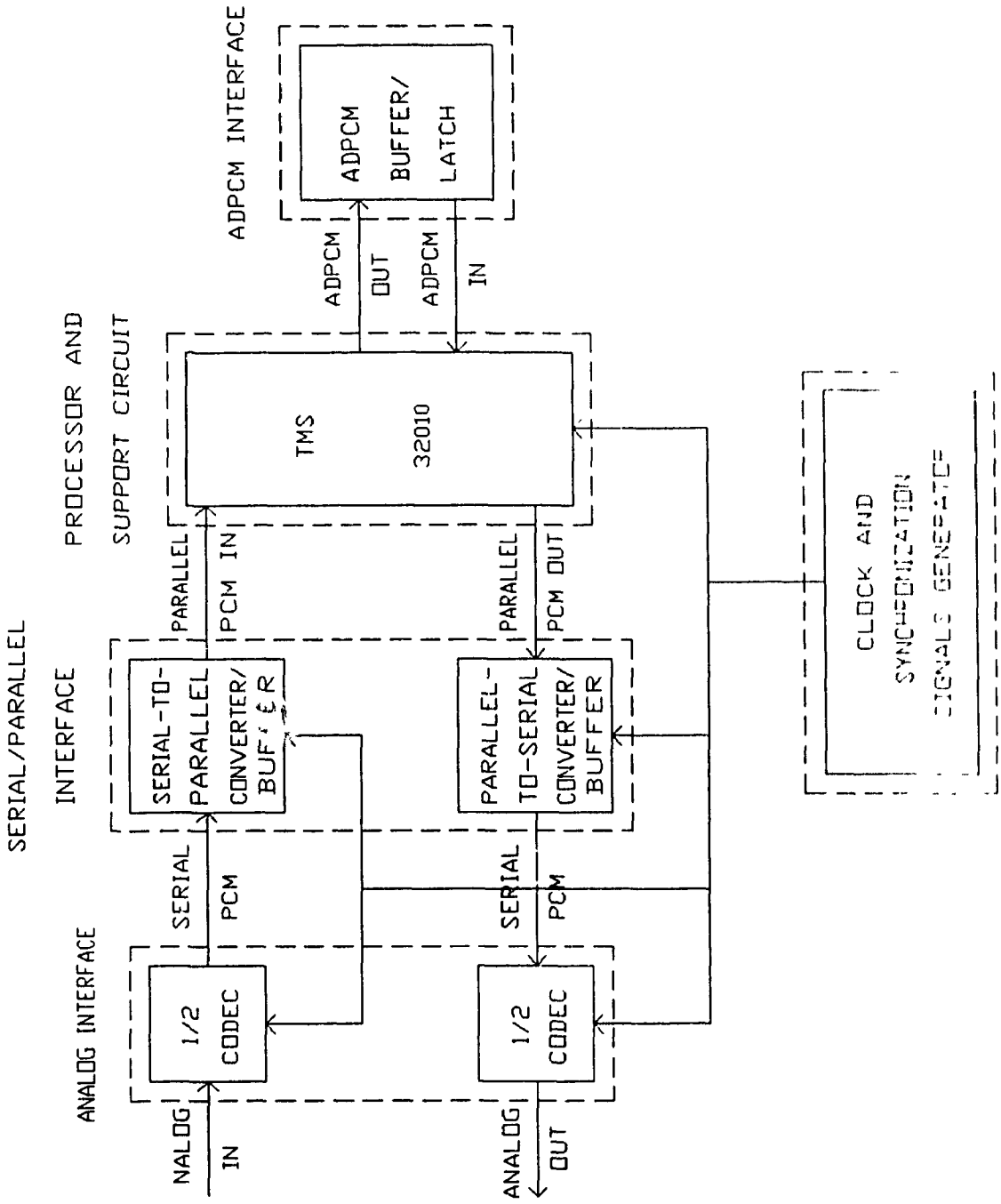


Fig.48. Circuit block diagram.

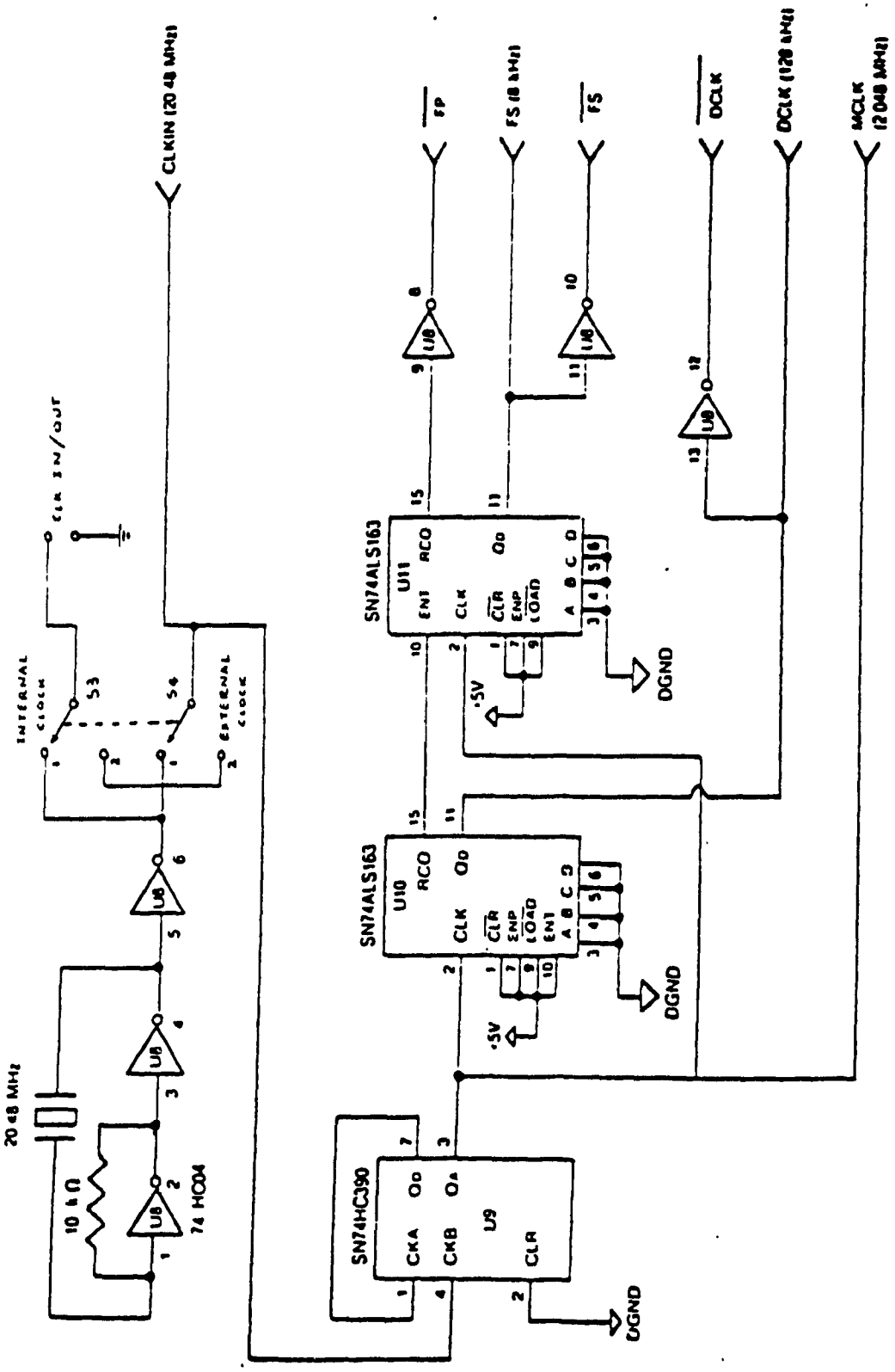


Fig.49. Clock circuit.

Fig.50 shows the TMS32010 and the support circuit. Two SN74ALS138 decoders decode the I/O port addresses and generate the signals $\overline{RDP1}$, $\overline{RDP3}$, $\overline{WRP2}$, $\overline{WRP4}$, where the I/O port assignment is : Port1 for PCM IN, Port2 for PCM OUT, Port3 for ADPCM IN, and Port4 for ADPCM OUT. The program is stored in two TBP38L165 $2K \times 8$-bit PROM's.

The circuit allows for three different outputs depending on the position of the switch S1 which can be regulated externally (switch S2 follows S1). At position 1, we have analog loop back where the output of the TLO72 op-amp (U20) is routed to the LM386 audio amplifier (U21) without passing through the TCM2914 codec (U12). Position 2 is for PCM loop back : the PCM output of U12 is immediately fed back into U12 to be converted to analog form. Position 3 is the normal ADPCM connection.

4.2.1.2. Software

A listing of the source program in TMS32010 assembly language can be found in Appendix 2.

The program works on an interrupt basis.

At power start up or at reset, the TMS32010 begins execution at program memory location 0. A branch instruction transfers control to the RESET routine which initializes the constants and variables. Then it spins in a loop waiting for interrupts.

At each interrupt which happens every 125 μs, the TMS32010 begins execution at location 2 : it reads in an 8-bit PCM sample from Port1, converts into a 4-bit ADPCM word, and sends the result to Port4. It then reads in and converts a 4-bit ADPCM sample from Port3 to send the output 8-bit PCM sample to Port2. Fig.51 shows the flow chart of the program. Tables VII and VIII depict the various blocks in the algorithm in the order in which

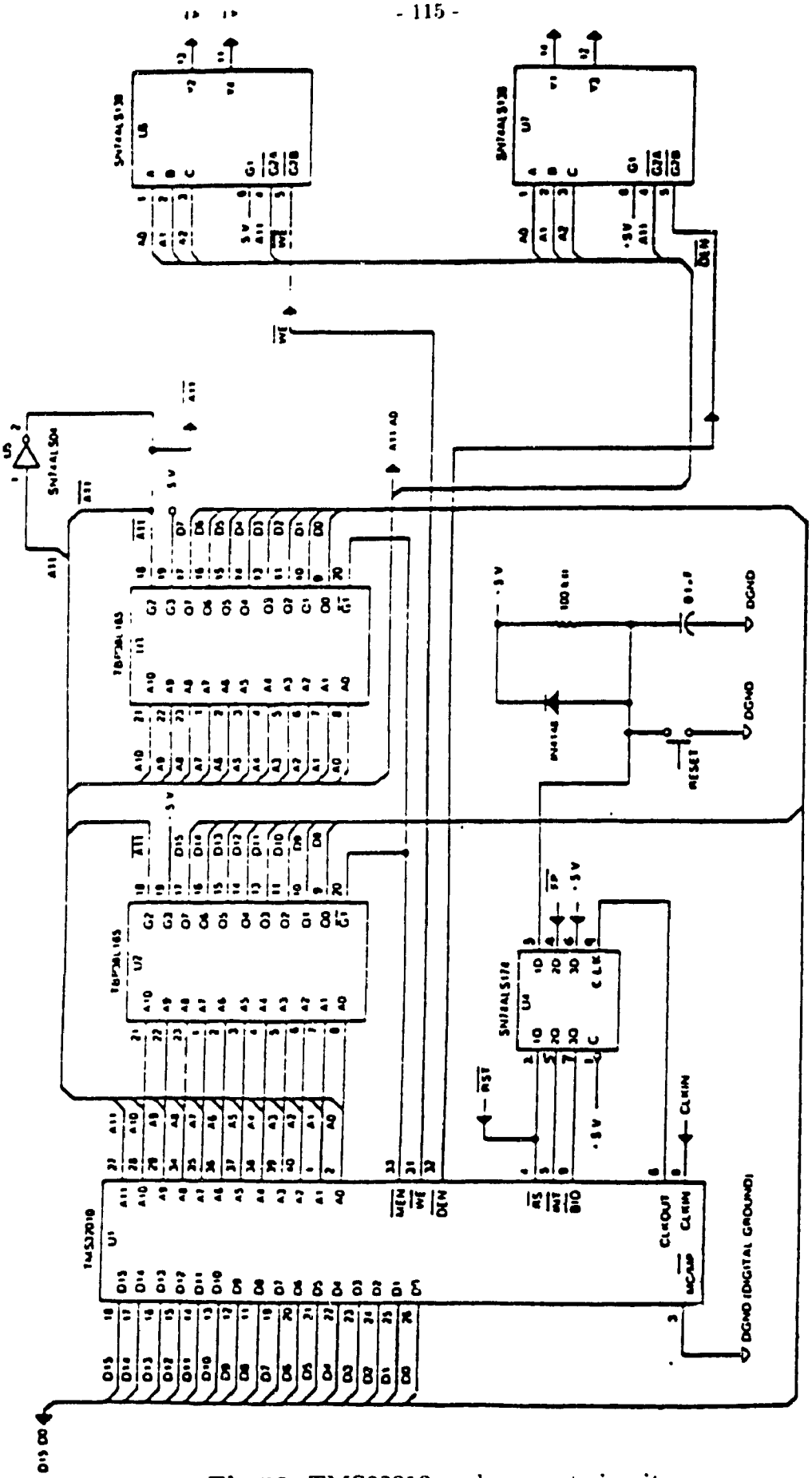


Fig.50. TMS32010 and support circuit.

they are executed. Also listed is the processor loading and demand which consists of the amount of program memory used to implement the given function and the number of instruction cycles in the worst case. Each block has been implemented using the equations given in the previous section concerning the ADPCM algorithm.

Constants and variables occupy the first 128 locations in the data RAM. The program is 1093 bytes long which include several look-up tables. As speed is essential in the algorithm, tables are quite indispensable. Specifically, the program utilizes tables for μ-law-to-linear-PCM conversion, reconstructed signal values,$F[I(k)]$ and $W[I(k)]$ values, and the shift values for antilogarithm calculations. Finding the logarithm and antilogarithm values is simplified through the approximating formulas :

$$\log_2(1+x) = x \qquad 0 \leq x \leq 1$$

$$\text{and} \qquad \log_2^{-1}(x) = 1 + x$$

The technique of binary search is applied in the calculation of logarithm values, the ADPCM sample to be output as well as log PCM value.

As the TMS32010 instruction set supports numeric-intensive signal processing applications, it facilitates several functions in the algorithm. For example, the calculation of signal estimate and predictor coefficients, which requires multiply-and-accumulate operation, can be performed effectively through instructions such as LTD (Load T Register, Accumulate Previous Product, and Move Data), MPY (Multiply), and APAC (Add P Register to Accumulator). Besides, faster execution of computations involving operands of different representations is enabled thanks to the hardware shift which is built into the ADD, SUB (Subtract), and LAC (Load Accumulator) instructions.

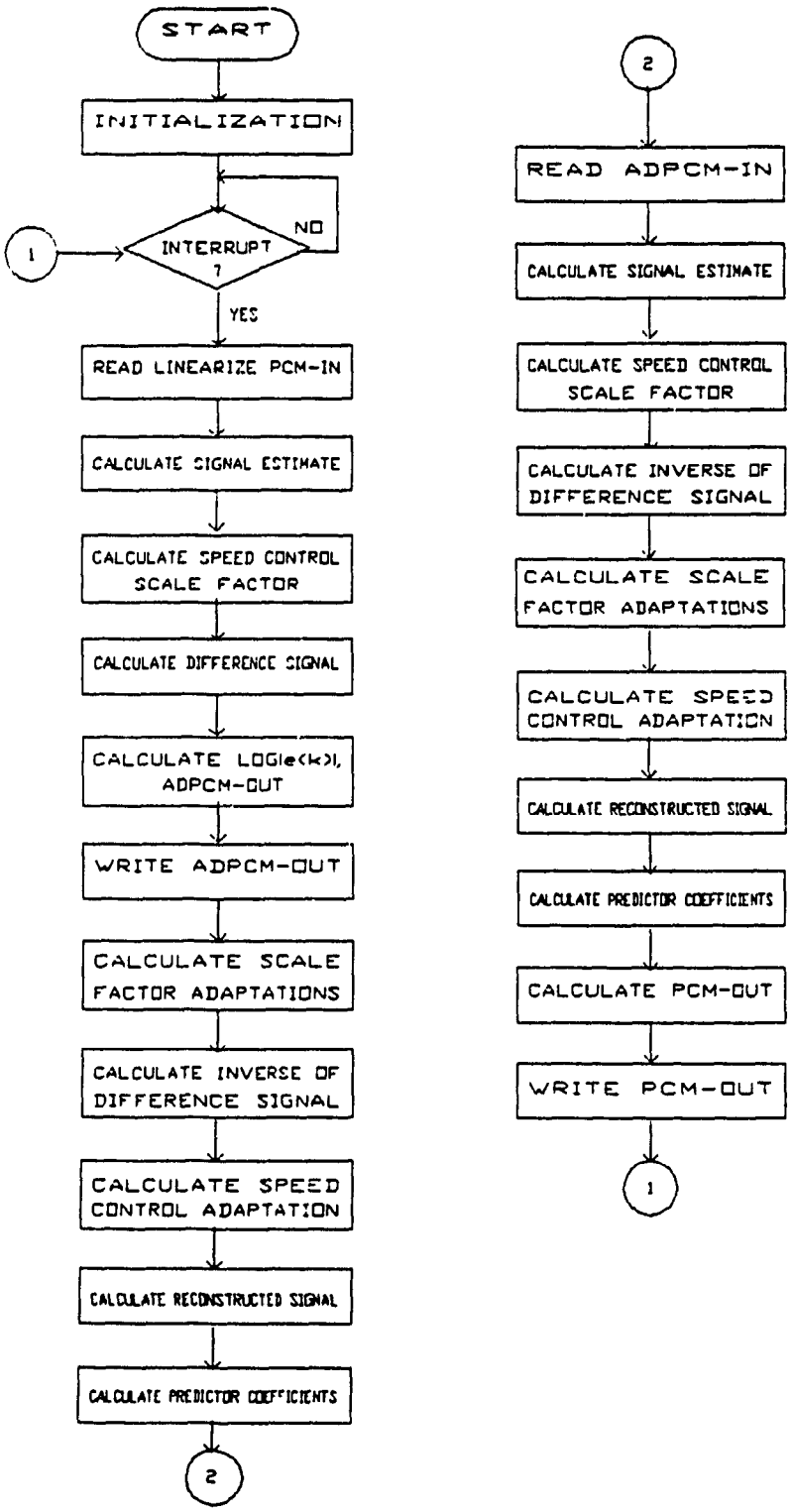


Fig.51. ADPCM flow chart.

ORDER	FUNCTION	DESCRIPTION	CYCLES	WORDS
1	INPUT PCM	Read 8-bit μ-law PCM and linearize it to 12 bit $s(k)$	8	0005
2	COMPUTE SIGNAL ESTIMATE	Calculate signal estimate $\tilde{s}(k)$ $\tilde{s}(k) = \sum_{i=1}^{2} a_i(k-1)\,\hat{s}(k-i) + s_{ez}(k)$ $s_{ez}(k) = \sum_{i=1}^{6} b_i(k-1)\,\hat{e}(k-i)$	30	001E
3	COMPUTE ADAPTIVE QUANTIZER	Calculate speed control $a_l(k)$,scale factor $y(k)$ $a_l(k) = \begin{cases} 1 & a_p(k-1) > 1 \\ a_p(k-1) & a_p(k-1) \le 1 \end{cases}$ $y(k) = a_l(k)\,y_u(k-1) + (1 - a_l(k))\,y_l(k-1)$	19	0013
4	COMPUTE DIFFERENCE SIGNAL	Calculate difference signal $e(k)$ from current sample $s(k)$ and signal estimate $\tilde{s}(k)$ $e(k) = s(k) - \tilde{s}(k)$	3	0003
5	COMPUTE ADPCM OUTPUT	Calculate $\log_2 \lvert e(k)\rvert$ and ADPCM output $I(k)$ $I(k) \leftarrow \log_2 \lvert e(k)\rvert - y(k)$	39	00A8
6	OUTPUT ADPCM	Write ADPCM output $I(k)$	2	0001
7	COMPUTE QUANTIZED DIFFERENCE	Calculate inverse of $e_q(k)$ and its linear domain value $\hat{e}(k)$	27	0017
8	COMPUTE SCALE FACTOR	Calculate updates for scale factor adaptation $y(k)$ $y_u(k) = (1 - 2^{-5})\,y(k) + 2^{-5}\,W[I(k)]$ $y_l(k) = (1 - 2^{-6})\,y_l(k-1) + 2^{-6}\,y_u(k)$	27	001C
9	COMPUTE SPEED CONTROL	Calculate updates for speed control adaptation $a_p(k)$ $d_{ms}(k) = (1 - 2^{-5})\,d_{ms}(k-1) + 2^{-5}\,F[I(k)]$ $d_{ml}(k) = (1 - 2^{-7})\,d_{ml}(k-1) + 2^{-7}\,F[I(k)]$ $a_p(k) =$ $\begin{cases} [1 - 2^{-4}]\,a_p(k-1) + 2^{-3} & \lvert d_{ms}(k) - d_{ml}(k)\rvert > 2^{-3}\,d_{ml}(k) \\ & \quad or\ y(k) < 3 \\ [1 - 2^{-4}]\,a_p(k-1) & otherwise \end{cases}$	26	001B
10	COMPUTE RECONSTRUCTED SIGNAL	Calculate reconstructed signal $\hat{s}(k)$ $\hat{s}(k-i) = \tilde{s}(k-i) + \hat{e}(k-i)$	3	0003
11	COMPUTE PREDICTOR COEFFICIENTS	Calculate updates for predictor coefficients $b_i(k) = (1 - 2^{-8})\,b_i(k-1) + 2^{-7}\,sgn[\hat{e}(k)]\,sgn[\hat{e}(k-i)]$ $i = 1, \cdots, 6$ $a_1(k) = (1 - 2^{-8})\,a_1(k-1) + 3.2^{-8}\,sgn[\hat{d}(k)]\,sgn[\hat{d}(k-1)]$ $a_2(k) = (1 - 2^{-7})\,a_2(k-1) + 2^{-7}\{sgn[\hat{d}(k)]\,sgn[\hat{d}(k-2)]$ $-f[a_1(k-1)]\,sgn[\hat{d}(k)]\,sgn[\hat{d}(k-1)]\}$ $\hat{d}(k) = \hat{e}(k) + s_{ez}(k)$ $f(a_1) = \begin{cases} 4\,a_1 & \lvert a_1\rvert \le 2^{-1} \\ 2\,sgn[a_1] & \lvert a_1\rvert > 2^{-1} \end{cases}$	96 280	0066

Table VII. Full-duplex transmitter.

Notes on authors contributed in functions/ equations listed in Table VII

Order	Author
2	Nishitani et al.
3	Petr
8	Petr (y_1)
	Goodman & Wilkinson (y_m)
9	Petr
10	Nishitani et al.
11	Millar & Mermelstein (Poles)
	Nishitani et al. (Zeros)

ORDER	FUNCTION	DESCRIPTION	CYCLES	WORDS
1	INPUT ADPCM	Read ADPCM input $I(k)$	2	0001
2	COMPUTE SIGNAL ESTIMATE	Calculate signal estimate $\tilde{s}(k)$ $\tilde{s}(k) = \sum_{i=1}^{2} a_i(k-1)\,\hat{s}(k-i) + s_{ez}(k)$ $s_{ez}(k) = \sum_{i=1}^{6} b_i(k-1)\,\hat{e}(k-i)$	30	001E
3	COMPUTE ADAPTIVE QUANTIZER	Calculate speed control $a_l(k)$,scale factor $y(k)$ $a_l(k) = \begin{cases} 1 & a_p(k-1) > 1 \\ a_p(k-1) & a_p(k-1) \le 1 \end{cases}$ $y(k) = a_l(k)\,y_u(k-1) + (1 - a_l(k))\,y_l(k-1)$	19	0013
4	COMPUTE QUANTIZED DIFFERENCE	Calculate inverse of $e_q(k)$ and its *linear domain value* $\hat{e}(k)$	36	0020
5	COMPUTE SCALE FACTOR	Calculate updates for scale factor adaptation $y(k)$ $y_u(k) = (1 - 2^{-5})\,y(k) + 2^{-5}\,W[I(k)]$ $y_l(k) = (1 - 2^{-6})\,y_l(k-1) + 2^{-6}\,y_u(k)$	27	001C
6	COMPUTE SPEED CONTROL	Calculate updates for speed control adaptation $a_p(k)$ $d_{ms}(k) = (1 - 2^{-5})\,d_{ms}(k-1) + 2^{-5}\,F[I(k)]$ $d_{ml}(k) = (1 - 2^{-7})\,d_{ml}(k-1) + 2^{-7}\,F[I(k)]$ $a_p(k) =$ $\begin{cases} [1 - 2^{-4}]\,a_p(k-1) + 2^{-3} & \lvert d_{ms}(k) - d_{ml}(k)\rvert > 2^{-3}\,d_{ml}(k) \text{ or } y(k) < 3 \\ [1 - 2^{-4}]\,a_p(k-1) & \text{otherwise} \end{cases}$	29	001B
7	COMPUTE RECONSTRUCTED SIGNAL	Calculate reconstructed signal $\hat{s}(k)$ $\hat{s}(k-i) = \tilde{s}(k-i) + \hat{e}(k-i)$	3	0003
8	COMPUTE PREDICTOR COEFFICIENTS	Calculate updates for predictor coefficients $b_i(k) = (1 - 2^{-8})\,b_i(k-1) + 2^{-7}\,sgn[\hat{e}(k)]\,sgn[\hat{e}(k-i)]$ $i = 1, \cdots, 6$ $a_1(k) = (1 - 2^{-8})\,a_1(k-1) + 3.2^{-8}\,sgn[\hat{d}(k)]\,sgn[\hat{d}(k-1)]$ $a_2(k) = (1 - 2^{-7})\,a_2(k-1) + 2^{-7}\{sgn[\hat{d}(k)]\,sgn[\hat{d}(k-2)]$ $- f[a_1(k-1)]\,sgn[\hat{d}(k)]\,sgn[\hat{d}(k-1)]\}$ $\hat{d}(k) = \hat{e}(k) + s_{ez}(k)$ $f(a_1) = \begin{cases} 4\,a_1 & \lvert a_1\rvert \le 2^{-1} \\ 2\,sgn[a_1] & \lvert a_1\rvert > 2^{-1} \end{cases}$	96	0066
9	COMPUTE LOG-PCM	Convert $\hat{s}(k)$ to μ-law signal $s_p(k)$	41	0075
10	OUTPUT PCM	Write μ-law output $s_p(k)$	2	0001
11	WAIT	Spin until the next interrupt	285	0005

Table VIII. Full-duplex receiver.

Notes on authors contributed in
functions/ equations listed in Table VIII

Order	Author
2	Nishitani et al.
3	Petr
5	Petr (y_l)
	Goodman & Wilkinson (y_u)
6	Petr
7	Nishitani et al.
8	Millar & Mermelstein (Poles)
	Nishitani et al. (Zeros)

From tables VII and VIII, it can be observed that during each interrupt, it takes the TMS32010 280 instruction cycles to transmit an ADPCM sample, and 285 cycles to convert ADPCM into PCM samples. Thus, for the worst case, the total number of instruction cycles in each interrupt would be 565.

In this codec, modification to the CCITT ADPCM algorithm have been made to facilitate the implementation. Instead of floating-point, fixed-point arithmetic has been used in the calculation of the predictor coefficients. Besides, in the updating of the parameters, the order of computation has been altered. As a matter of fact, similar changes have been proposed by Hangartner and Jain [67] to improve the performance of their codec. Originally, they considered using a mixed predictor with full integer tap coefficients and floating-point format signal samples as an alternate to floating-point predictor. However, the authors have chosen to use a full integer predictor "since no overwhelming reason could be found to retain the higher complexity mixed predictor". In the computations of parameter adaptations, which have the general form

$$y(n) = y(n-1) + 2^{-k}\ (2^m\ x(n) - y(n-1))$$

where $x(n)$ and $y(n)$ are the input and output variables respectively, and $k \geq m$, the order of calculation has been changed from

$$y(n) = y(n-1) + Int[2^{m-k}\ x(n)] - Int[2^{-k}\ y(n-1)]$$

to

$$y(n) = Int[y(n-1) + 2^{m-k}\ x(n) - 2^{-k}\ y(n-1)]$$

This way, there are fewer right shifts to be performed.

According to the authors, this change improves performance and implementability because it leads to a faster convergence, more accuracy, and more

efficiency. (The DSP typically executes left shifts more efficiently than right shifts).

Even though tables have been used in the program, because of the limit of the TMS32010 program memory capacity, most values have to be computed. Given a DSP with a larger memory, more look-up tables can be added to reduce computation time, and thus, improve execution time. Besides, if we study tables VII and VIII carefully, we can observe that the calculation of predictor coefficients take up about 34% of the total execution time during each interrupt. If some of the predictor coefficients are fixed, it would take the DSP less time to complete the process.

4.2.2. Techniques to Improve the Execution Time

4.2.2.1. Table Look-up Technique

There is a trade-off between memory space and speed. Look-up tables can always be used to save the time required for any calculation. However, the tables would take up space. Although system designer would in certain cases sacrifice one for the other, it is not possible all the times to make the choice because of system limits and/or requirements.

In implementing the CCITT algorithm, when program memory space allows, system designer can install more look-up tables, such as those of logarithm, antilogarithm, quantization, and unlocking scale factor values. Table IX lists the tables and their size.

Table	Words
μ-law-to-linear PCM	256
Reconstructed signal	8
$F[I(k)]$	8
$W[I(k)]$	8
Shift value	15
Logarithm value	256
Antilogarithm value	1823
Quantization value	1556
Unlocking scale factor	36608

Table IX. Tables and their size.

4.2.2.2. Fixed Poles

The predictor, as recommended by CCITT, consists of a sixth-order section that models zeros, and a second-order section that models poles. In order to effectively cater for the variety of input signals which might be encountered, both zeros and poles are adaptive. As can be observed in tables VII and VIII, the machine cycles it takes to compute the predictor coefficients are 96 at either transmitter or receiver side. Indeed, it is the most time-consuming part of the algorithm. It has been stated that predictors with zeros do not give rise to maistracking [47]. Moreover, of the 96 instruction cycles, only 31 are for the calculation of zero coefficients. Therefore, either an all-zero system or a system with adaptive zeros and fixed poles would already save 130 instruction cycles (or 23% of the total execution time) during each interrupt. Systems with adaptive zeros and fixed poles have been reported to yield good performance for speech signal [51], [52]. Therefore, modifications to the CCITT algorithm to realize such a system would result in a codec having acceptable performance with a

shorter execution time. In this case, although the codec performance in speech signal transmission may remain the same, the truth will not hold for VBD transmission.

4.3. Structure of an ADPCM Codec Processor

It is to be realized that processors which have been designed for general purpose cannot always be expected to function as an optimum machine for any specific target application. Rather, special purpose processors should be utilized in the specific applications they have been built for, if satisfactory performance is to be achieved. We now attempt to identify the structure of a processor which is optimized for the implementation of the CCITT algorithm.

The algorithm requires different sizes for different variables with 19 bits as the largest word length. However, most variables have bit length of 16 or less. And for the two variables of size 19 bits, two 16-bit words can always be used to store the value of each. Therefore, 16-bit data word would be sufficient. Nevertheless, a 32-bit ALU/accumulator is required to handle computations which result in values occupying more than 16 bits.

Several additions or subtractions in the algorithm involve operands which have different bit lengths or different binary point positions. In those cases, binary position adjustment needs to be performed by a shifter before the actual arithmetic operation. Besides, the multibit position shift function is also required in power of 2 multiplication. Consequently, multibit position shifter is a key element for CCITT ADPCM codec implementation. An 0 to 15-bit barrel shifter can be used to fulfill this requirement.

Beside addition and subtraction, an often encountered arithmetic operation in the algorithm is multiplication which appears in the calculation of

reconstructed signal, scale factor, difference signal, and predictor coefficients. As such, a 16 × 16 bit multiplier is indispensable in our processor.

The data RAM of the processor does not need to have more than 144 words as the RAM is used only to store constants and values of defined variables. However, the program memory should be large enough to store look-up tables. A 64K of program space would enable the implementation of tables outlined on page 122.

The instruction set should support arthmetic operations : addition, subtraction, and multiplication. To speed up the calculations, single cycle multiply/accumulate instruction is essential. In general, fixed-point arithmetic would be sufficient. Logical operations such as AND, XOR, which are used to examine or manipulate bits of a word, to find the one's and two's complement value of a certain variable, or to process the PCM codewords received or to be transmitted, should also have their own instruction codes. In addition to this, there should be instructions for unconditional and conditional branches which are used quite extensively in the binary tree search technique. This technique is applied in locating the most significant digit of a word, in finding the ADPCM and the PCM codewords to be transmitted. Finally, the processor should support sign extension as well as two's complement representation. It should also provide for I/O interface and control. As speed is essential in our application, the processor should have fast cycle time. 200 ns would be the maximum instruction cycle time.

CHAPTER V

PERFORMANCE

Once an algorithm has been implemented, it is essential that the performance of the corresponding system be evaluated so that its capability, efficiency as well as the impact of the implementation can be determined. This chapter describes the test results of the ADPCM codec realized on a TMS32010 processor. It then reports the improvement in execution time obtained through added look-up tables and fixed poles introduced in the TMS32020-based system. The performance of ADPCM codec in tandem is examined next, through reference to reported studies. The incorporation of ADPCM in communication network implies that ADPCM codec is to process voice as well as non-voice signals. Measurements have been taken to verify whether the performance of ADPCM coder is satisfactory for VBD signals. Finally, attempts have been made to reduce the bit rate to 24 kbits/s. When more look-up tables are added, and fixed poles are used, 3-bit ADPCM decreases the execution time by the same amount as 4-bit ADPCM.

5.1. Performance Evaluation

5.1.1. ADPCM Codec Using TMS32010

The TMS32010-based ADPCM codec has been realized and its performance has been measured by Vo [70].The test results were obtained with a 4945A Transmission Impairment Measuring Set (TIMS) and a 8903A Audio Analyzer, both manufactured by Hewlett Packard.

In all the tests, the basic setup is as follows :

Clock : External, 20 MHz

Trans. impedance : 600 Ω

Rec. impedance : 600 Ω

Input level : -40 dBm

Measurements have been taken for ADPCM, PCM as well as analog output signals so that any bad performance due to the analog portion of the circuit would not be wrongly attributed to the ADPCM system.

5.1.1.1. Frequency Response

The output signal level for different frequencies is tabulated in Table X and plotted in Fig.52.

The zero reference level is at 6.9 dBm, which corresponds to the output level at 1000 Hz in the analog loop back mode. The level for PCM and ADPCM is higher because of the gain of the TCM2914 codec.

The curve shows a good response from 500 Hz to over 3000 Hz. In this range, the frequency distortion is minimum. This frequency range is adequate for telephone speech. The sudden drop in signal level (in PCM and ADPCM modes) for high frequencies is due to the low-pass-filtering action of the codec.

5.1.1.2. Distortion

Total Harmonic Distortion (THD)

Table XI shows the obtained data and Fig.53 is the graph for the total harmonic distortion of the system.

Frequency (Hz)	Level (dB)		
	Analog	PCM	ADPCM
200	-7.9	-1.6	-1.6
300	-4.5	1.2	1.2
400	-2.7	2.3	2.3
500	-1.7	2.8	2.9
600	-1.1	3.1	3.1
700	-.7	3.2	3.3
800	-.4	3.4	3.4
900	-.2	3.5	3.5
1000	0.0	3.6	3.5
1100	.2	3.6	3.6
1200	.2	3.7	3.6
1300	.3	3.7	3.7
1400	.4	3.8	3.7
1500	.5	3.8	3.8
1600	.5	3.8	3.8
1700	.6	3.8	3.8
1800	.6	3.8	3.8
1900	.6	3.8	3.8
2000	.6	3.7	3.8
2100	.6	3.7	3.7
2200	.6	3.7	3.7
2300	.6	3.7	3.7
2400	.7	3.7	3.7
2500	.7	3.7	3.7
2600	.7	3.7	3.7
2700	.7	3.7	3.7
2800	.7	3.7	3.7
2900	.8	3.8	3.7
3000	.8	3.8	3.8
3100	.8	3.8	3.7
3200	.7	3.6	3.6
3300	.7	2.9	3.0
3400	.7	1.4	1.4
3500	.7	-1.6	-1.6
3600	.7	-6.0	-6.0

Table X. Frequency response.

In the range 800-3000 Hz, ADPCM has the highest distortion; PCM is next; and analog has the lowest value. The maximum difference among the three modes, in term of distortion, is less than 2 dB, which indicates a pretty good performance of the ADPCM system.

Intermodulation Distortion

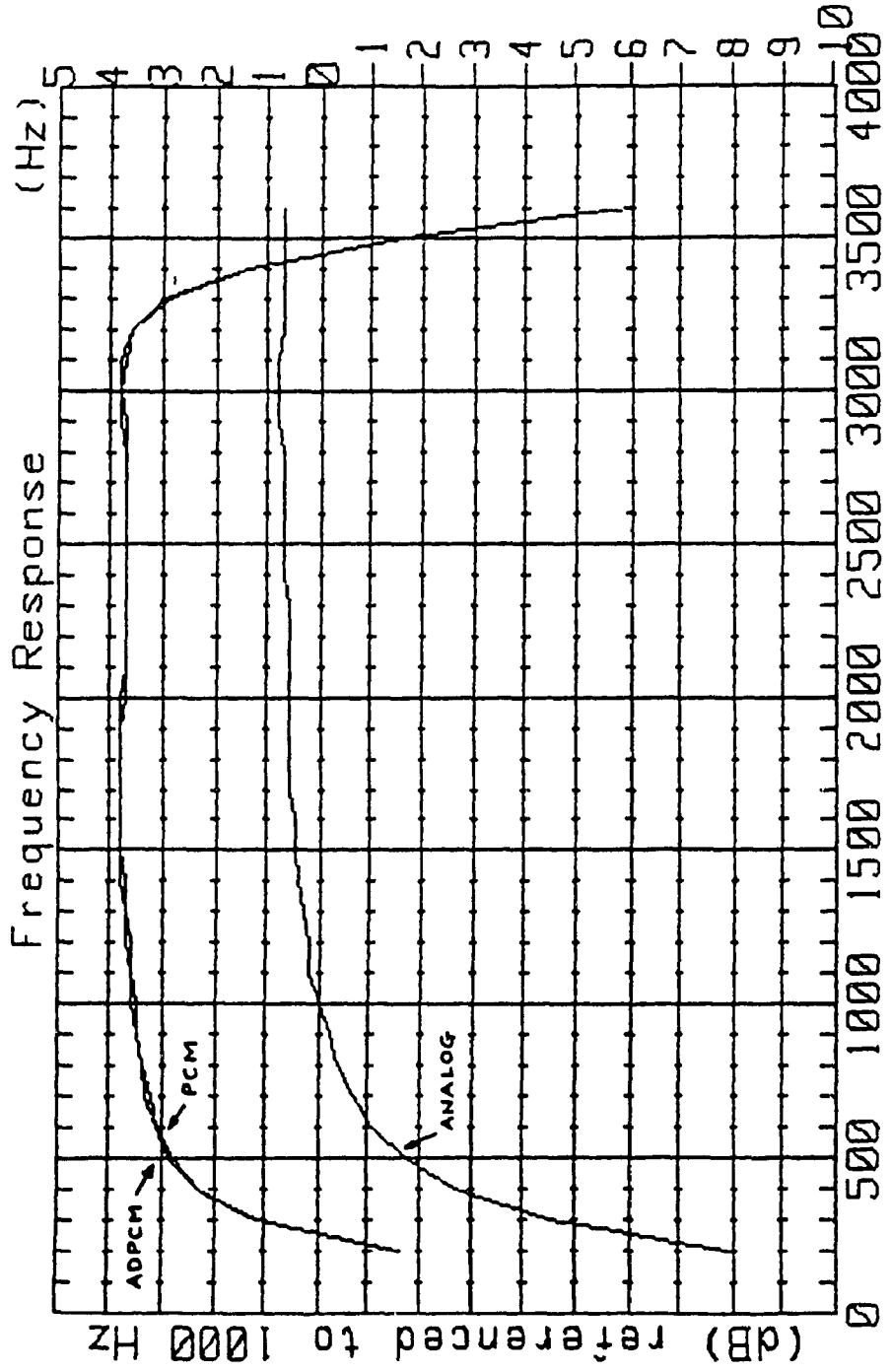


Fig.52. Frequency response curves.

Frequency (Hz)	Level (dB)		
	Analog	PCM	ADPCM
200	-18.3	-21.2	-20.4
300	-21.9	-24.4	-23.9
400	-24.5	-26.1	-25.5
500	-25.9	-26.9	-26.4
600	-25.8	-26.7	-26.0
700	-25.9	-26.1	-25.8
800	-26.0	-26.1	-25.4
900	-26.3	-26.2	-25.7
1000	-26.7	-26.6	-25.8
1100	-26.8	-26.4	-26.0
1200	-26.7	-26.4	-25.8
1300	-26.7	-26.5	-25.8
1400	-26.7	-26.2	-25.8
1500	-26.6	-26.4	-25.8
1600	-26.2	-26.0	-25.6
1700	-26.4	-26.0	-25.7
1800	-26.5	-25.8	-25.6
1900	-26.4	-25.8	-25.3
2000	-26.6	-25.9	-25.5
2100	-26.4	-25.8	-25.5
2200	-26.6	-25.8	-25.7
2300	-26.5	-25.7	-25.2
2400	-26.6	-25.5	-25.3
2500	-26.7	-25.4	-25.4
2600	-26.6	-25.4	-25.3
2700	-26.6	-25.4	-25.2
2800	-26.7	-25.4	-25.5
2900	-26.6	-25.3	-25.2
3000	-26.7	-25.5	-25.3
3100	-26.6	-25.7	-25.7
3200	-26.7	-25.6	-25.7
3300	-26.7	-25.0	-25.1
3400	-26.7	-23.5	-23.7
3500	-26.8	-20.2	-20.7
3600	-26.7	-15.7	-15.7

Table XI. Total harmonic distortion.

This type of distortion appears as harmonics of individual frequencies plus intermodulation (mixing) products of the input frequencies. This generates new signal components not present in the original transmitted signal.

5.1.1.3. Noise

Noise-with-Tone

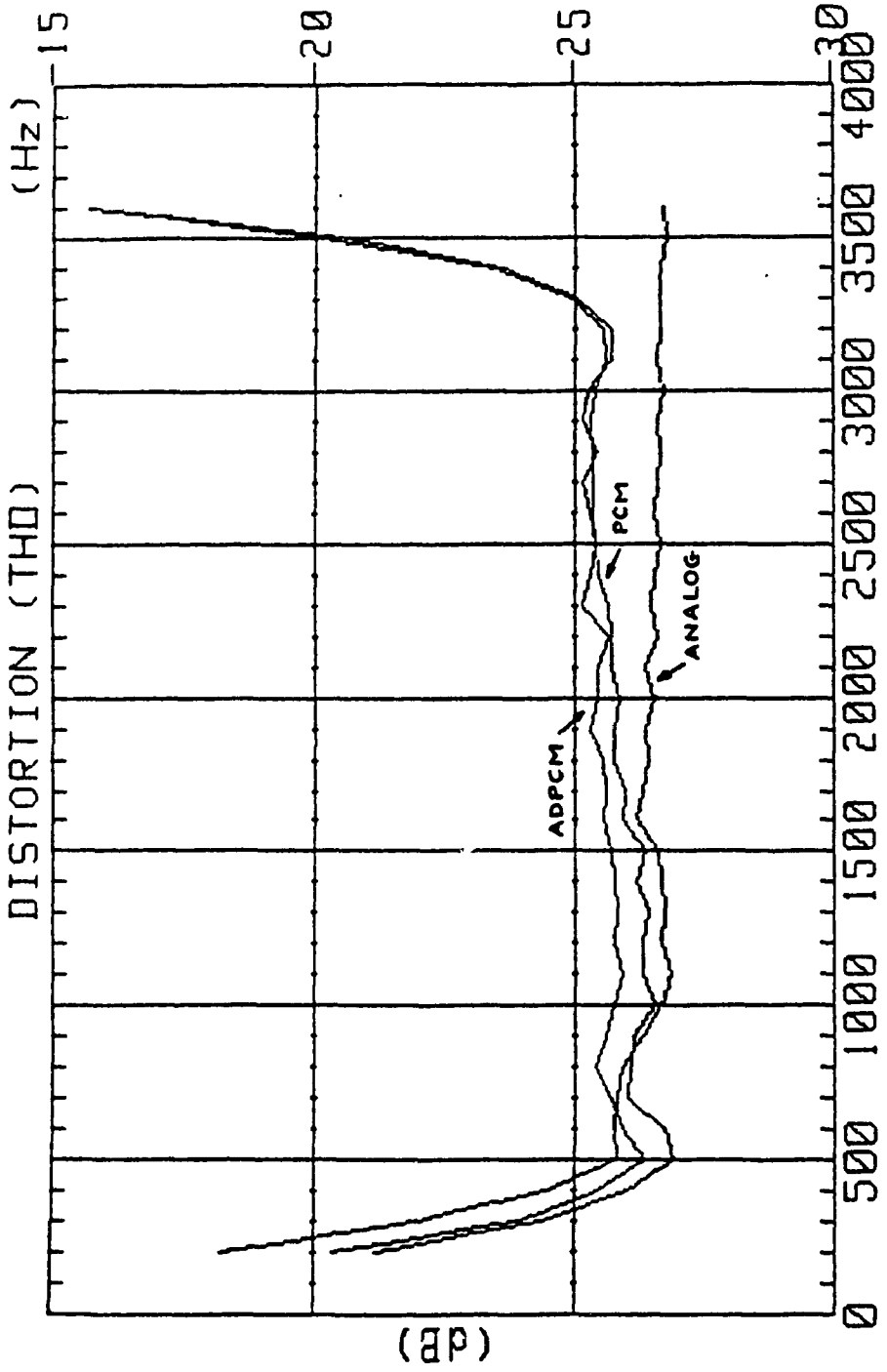


Fig.53. Total harmonic distortion curves.

	Analog	PCM	ADPCM
Signal level	7.0 dBm	9.8 dBm	9.9 dBm
Signal 2^{nd} order	42 dB	41 dB	40 dB
Signal 3^{rd} order	45 dB	26 dB	26 dB

Table XII. Intermodulation distortion.

	Analog	PCM	ADPCM
Noise-with-tone	62 dBrn	66 dBrn	67 dBrn
Signal-to-noise	35 dB	34 dB	33 dB
Noise to ground	56 dBrn	58 dBrn	59 dBrn
Message CKT noise	62 dBrn	65 dBrn	65 dBrn

Table XIII. Noise levels

The noise-with-tone level is measured after the output signal has been passed through a notch filter to take out the input frequency component.

Signal-to-Noise Ratio

This ratio is obtained by comparing the noise-with-tone level with the holding tone (output) level.

The noise levels measured at 1004 Hz are tabulated in Table XIII.

Although the noise level of the whole system is very high (20-30 dBrn would be more acceptable values), the noise levels for PCM and ADPCM modes are greater than that of analog mode by 3 to 5 dB only, which proves that most of the noise come from the analog portion of the circuit. The signal-to-noise ratio obtained (33-35 dB) is very close to the normal signal-to-noise ratio for 7-bit μ-law PCM, which is about 34 dB [10].

	Analog	PCM	ADPCM
20-300 Hz amp.	6.5% PK	6.3% PK	6.0% PK
20-300 Hz phase	6.8 deg	6.8 deg	6.8 deg
4-300 Hz amp.	6.7% PK	6.5% PK	7.2% PK
4-300 Hz phase	7.2 deg	7.5 deg	8.2 deg
4-20 Hz amp.	0.1% PK	0.7% PK	0.5% PK
4-20 Hz phase	0.5 deg	0.6 deg	0.6 deg

Table XIV. Phase and amplitude jitters.

5.1.1.4. Jitters

The phase jitter is a measurement of the phase deviation of a 1004 Hz holding tone on a voice channel.

The amplitude jitter is the summation of incidental amplitude modulation and effects of interference and noise. It is measured by examining the amplitude disturbances on a 1004 Hz test tone.

5.1.1.5. Transients

	Analog	PCM	ADPCM
Impulse noise :			
Low 68 dBrn	421 cnts	421 cnts	420 cnts
Mid 72 dBrn	420 cnts	420 cnts	420 cnts
High 76 dBrn	360 cnts	417 cnts	417 cnts
Phase hits ±10 deg	0	0	0
Gain hits ±5 dB	0	0	0
Dropouts−12 dB	0	0	0

Table XV. Transients.

The data in Table XV were obtained by counting the inpulse noise (spikes), the phase hits (sudden change in phase), the gain hits (sudden change in level), and the dropouts (sudden drop in level) for one minute with a count rate of

7/second, and with the thresholds set at 68 dBrn for impulse noise, 10 degrees for phase hits, and 5 dB for gain hits.

Once again, most of the noise can be attributed to the analog portion of the circuit, rather than to PCM and ADPCM.

5.1.1.6. Envelope Delay

Frequency (Hz)	Envelope delay (μs)		
	Analog	PCM	ADPCM
200	3800	5660	6000
400	3347	4179	4560
600	3186	3805	4190
800	3112	3677	4060
1000	3072	3623	4006
1200	3049	3598	3979
1400	3033	3587	3967
1600	3024	3589	3974
1800	3016	3596	3984
2000	3010	3610	3996
2200	3007	3630	4015
2400	3003	3661	4043
2600	3001	3702	4089
2800	2999	3772	4158
3000	2997	3897	4278
3200	2996	4126	4508
3400	2995	4370	4760
3600	2994	4296	4677

Table XVI. Envelope delay.

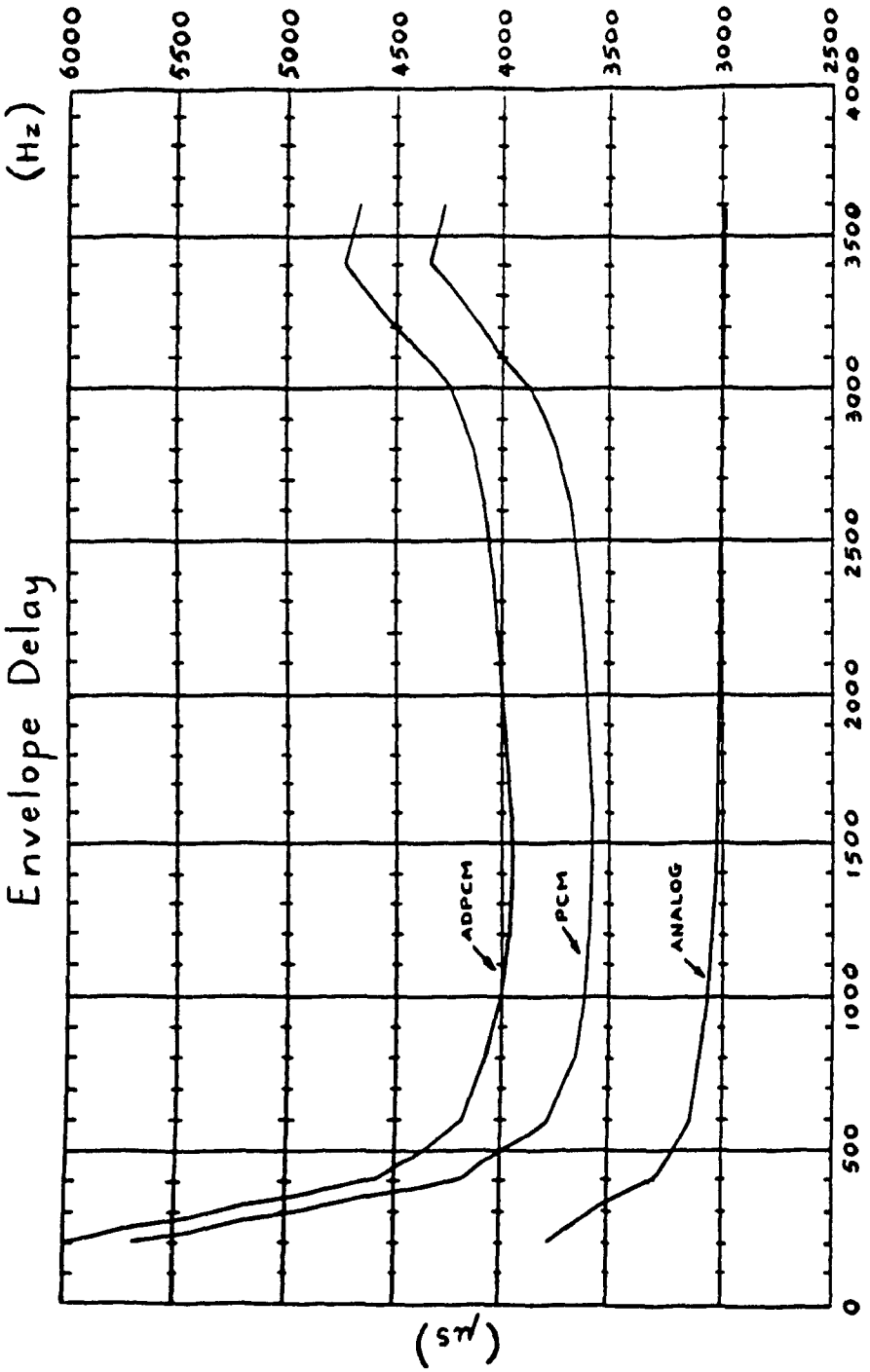


Fig.54. Envelope delay curves.

The envelope delay is a measurement of the linearity of the phase versus frequency of a voice channel.

Ideally, the phase should shift linearly when the frequency is increased. In that case the envelope delay would be constant for all frequencies in the relevant range, and we would get a horizontal envelope delay curve.

As Fig.54 shows, we have a pretty good curve for the analog mode, and U-shaped curves for PCM and ADPCM. However, non-linear phase shift does not have any significant effect over voice transmission, although it seriously affects data transmission.

5.1.2. ADPCM Codec Using TMS32020

The TMS32020 is the second member of the TMS320 family. Its architecture is based upon that of the TMS32010. Similar to the TMS32010, it has 200-ns instruction cycle time. However, the on-chip data RAM provides more flexibility in system design in that of the 544 words 256 words may be configured as either data or program memory. Besides, it allows larger program memory space, 64K words, which can operate at full speed. Its set of instructions includes the TMS32010 instruction set plus several extra instructions which help improve the performance of DSP algorithms. Switching from the TMS32010 to the TMS32020 has consequently many-sided advantages. Not only that the hardware is in pace with the continuously changing technology but also the software helps improve the system performance.

In order for the TMS32020 to replace the TMS32010 in the ADPCM system, the original hardware and software have to be modified as described in Appendix 3. Note that the changes in the software are just to allow the ADPCM system to yield similar performance as reported in 5.1.1. Further modifications are then made to reduce the system execution time.

ORDER	FUNCTION	CYCLES	WORDS
1	INPUT PCM	8	0005
2	COMPUTE SIGNAL ESTIMATE	22	0016
3	COMPUTE ADAPTIVE QUANTIZER	18	0012
4	COMPUTE DIFFERENCE SIGNAL	3	0003
5	COMPUTE ADPCM OUTPUT	34	0052
6	OUTPUT ADPCM	2	0001
7	COMPUTE QUANTIZED DIFFERENCE	23	0011
8	COMPUTE SCALE FACTOR	18	0010
9	COMPUTE SPEED CONTROL	24	0018
10	COMPUTE RECONSTRUCTED SIGNAL	3	0003
11	COMPUTE PREDICTOR COEFFICIENTS	94 --- 249	0063

Table XVII. Full-duplex transmitter (Tables added).

5.1.2.1. Tables versus Execution Time

With 64K of program memory, the TMS32020 allows ADPCM system designers to install more look-up tables. When the tables listed in Table IX were used, the total number of instruction cycles during each interrupt decreased to 500 (refer to Tables XVII and XVIII). Compared to the original program, the improvement in speed is 11.50% with an increase in program size of 97.35% in exchange.

ORDER	FUNCTION	CYCLES	WORDS
1	INPUT ADPCM	2	0001
2	COMPUTE SIGNAL ESTIMATE	22	0016
3	COMPUTE ADAPTIVE QUANTIZER	18	0012
4	COMPUTE QUANTIZED DIFFERENCE	32	001A
5	COMPUTE SCALE FACTOR	18	0010
6	COMPUTE SPEED CONTROL	24	0018
7	COMPUTE RECONSTRUCTED SIGNAL	3	0003
8	COMPUTE PREDICTOR COEFFICIENTS	94	0063
9	COMPUTE LOG-PCM	36	005E
10	OUTPUT PCM	2	0001
11	WAIT	----- 251	0005

Table XVIII. Full-duplex receiver (Tables added).

5.1.2.2. Tables and Fixed Poles versus Execution Time

As discussed in 4.2.2.2., if fixed poles are used in the algorithm, we can save 23% of the total execution time. Moore and Gibson [36] have used two fixed predictor coefficients $a_1 = 1.515$, and $a_2 = -0.752$ in their ADPCM system, and reported to obtain good performance. As these coefficients were calculated from McDonald's average speech data [15], some degradation might occur when non-speech data are encountered. However, for speech processing

applications, ADPCM systems using those fixed coefficients can be expected to function satisfactorily.

When both fixed poles and look-up tables are installed in the ADPCM system, the TMS32020 spends 370 machine cycles during each interrupt to transcode PCM and ADPCM signals. This signifies an improvement of 34.51% of execution time.

It can be observed that when more look-up tables are used, more memory locations will be taken up. Naturally, it is the improvement in execution time which can justify such usage of memory spaces. The time saved during each interrupt could be used for the TMS32020 to perform another task, should multitasking be intended for the system. Even with the tables installed, there still rest roughly 24K bytes of program memory which could be used for programming purposes. Besides, if the ADPCM program (with tables and fixed poles) is to run on a machine with faster instruction cycle time, say 100 ns, the time saved would amount to 69%. This implies that after processing one set of PCM and ADPCM signals, the system still has ample time to transcode another set. In other words, with proper setup, a channel capacity gain of 2-to-1 could be achieved. In fact, the third member of the TMS320 family, namely the TMS320C25 which is built with 100-ns instruction cycle time, can be utilized in our ADPCM system to enable channel doubling. Further work on this implementation can be carried out to study system performance as well as possible improvements.

5.2. Performance of ADPCM in Tandem

A signal may undergo a number of tandem codings on its path to its final destination. This gives rise to the question : what will happen when signals are coded successively in tandem? Generally, cumulative distortion will occur

because of the addition of both quantization and channel noises in each coding stage. As a result, the SNR at each coding stage successively reduces. It is this characteristic of a coder, the capability to provide quality transmission performance when connected one after another over a certain number of coding stages, which determine the effect of tandem coding on signals.

The tandem property of ADPCM codecs is of importance for network planning as the 32 kbits/s ADPCM coding will be introduced while Time-Division Multiplexing (TDM) switch still operates with 64 kbits/s PCM. Moreover, because of the slow transition from the analog network to the digital network, it is quite likely that a part of the network becomes digital while the other part still trails behind with the conventional analog system.

In studying the performance of ADPCM in tandem, we differentiate synchronous and asynchronous codings. Synchronous coding refers to multiple codings of ADPCM with 64 kbits/s PCM interfaces between each coding. Two ADPCM synchronous codings would have the configuration :

$$analog - 64kbits/s - 32kbits/s - 64kbits/s - 32kbits/s - 64kbits/s - analog.$$

By asynchronous coding is meant multiple coding of ADPCM with analog interfaces between each coding. The configuration of two ADPCM asynchronous codings is shown below :

$$analog - 64kbits/s - 32kbits/s - 64kbits/s - analog - 64kbits/s - 32kbits/s - 64kbits/s - analog.$$

Usually, the performance of an ADPCM coder in tandem is evaluated in terms of the performance of the same codec in single coding and/or in terms of the performance of PCM in multiple codings.

Raulin, Bonnerot, Jeandot, and Lacroix [11] showed that four ADPCM coders can be connected in tandem without accumulation of distortion.

According to Mermelstein and Williams, the quality for one ADPCM coding is roughly that of 7-bit PCM. For four asynchronous codings, the quality drops to 6-bit PCM [71]. This holds true for both speech and VBD.

In evaluating the subjective quality of 32 kbits/s ADPCM-DLQ [46], Petr reported that eight synchronous encodings introduce only a small amount of additional degradation over that of a single encoding, and are subjectively equivalent to eight asynchronous 64 kbits/s PCM encodings [72].

Looking at asynchronous codings, Cointot [53] found that for eight successive encoding operations, the SNR is about 23 dB which represents a degradation equivalent to that obtained with a PCM system under the same conditions.

The CCITT ADPCM codec used a synchronous tandem algorithm which eliminates cumulative distortion regardless of the number of tandem transcodings in the steady state mode [73]. In fact, Williams and Suyderhoud [74] concluded that for this coder in a synchronous mode of operation, no additional perceptible degradation will occur once synchronism has been acquired.

It is argued by Ramamoorthy [75] that the capacity of "tandem coding lock" depends on the bit rate and the predictor order. At lower rates, there is a good chance for the adaptive quantizer to get locked with the same set of step sizes, and hence results in the same sequence of coded output as the input to the coder. In this case of "locking", the coder functions as an identity mapping, and does not introduce any noise at all. According to the author, the memory in the coder reduces its chances of attaining the tandem coding lock. Therefore, a coder which has lower predictor order has less built-in memory but performs better in a large number of tandem codings. Further research in this direction

can be carried out to study the factors which determine the tandem property of ADPCM coder, and the extent of their effect.

Meanwhile, the Exchange Carriers Standards Association has proposed tandem coding limits for the 32 kbits/s ADPCM in order to ensure the capability of quality transmission performance between end users. These limits are proposed based on the test results which show the maximum number of asynchronously connected ADPCM links in tandem that enable the end-to-end performance criteria to be met [76]. For voice, the criterion is based on a measure called Q that captures the effect of quantizing noise on quality. The criterion of $Q \geq 20$ dB was used. For VBD, two criteria were used : Block Error Ratio ($BLER$) $\leq 10^{-2}$ (with a block size of 10^3 bits) and Bit Error Ratio (BER) $\leq 10^{-5}$. For voice applications, the standard provides a limit of at most four asynchronously connected ADPCM. For VBD signals at 2.4 kbits/s or less, it allows four ADPCM links in tandem. At 4.8 kbits/s, between two and four ADPCM links would be tolerated depending on the particular modem used.

5.3. Performance of VBD on ADPCM

The designer of a codec for use in the telephone trunks is faced with the challenge of encoding and recovering a variety of signals, including speech and VBD. Therefore, one of the important considerations in the choice of a particular coding technique is its performance with VBD.

Generally, efficient coders are designed to match the signal to be coded. As speech and data signals have quite different characteristics, coders that are optimum for speech are not necessarily optimum for data. Our question is would an ADPCM coder which can adapt its quantizer and its predictor in accordance with the incoming signals perform adequately with VBD as it does with speech? For an answer, we can refer to the VBD test results available

in the literature. Here again, as mentioned before, for coder performance with VBD, BER and/or $BLER$ are the appropriate performance measures.

Raulin, Bonnerot, Jeandot, and Lacroix [11] transmitted data signals at various rates, ranging from 1200 bits/s to 4800 bits/s, through their ADPCM transcoder. For both single and tandem (four-coder) codings, BER lower than 10^{-6} were observed in all cases.

Cointot's ADPCM coder [53] was reported to be capable of handling modem signals of up to 4800 bits/s. The transmission of 4800 bits/s data signals yields a $BER < 10^{-6}$ for the modem signal.

Similarly, Nishitani, Kuroda, Satoh, Katoh, and Aoki [66] could verify that voiceband modem signal transmission up to 4800 bits/s was possible through their ADPCM codec.

Fukasawa, Hosoda, Miyamoto, and Sugihara [33] used their coder to send data signals. They reported to obtain excellent performance. The transmission was free of error for 4.8 kbits/s modem. For 9.6 kbits/s modem, the error rate was estimated to be less than 10^{-7}.

Simulation results [77] revealed that Mermelstein and Millar's ADPCM coder can transmit 4800 bits/s VBD satisfactorily at 32 kbits/s, even with two or three stages of tandeming. However, they recommended 5 bits/sample or 40 kbits/s coding for 9.6 kbits/s data rates.

Mermelstein and Williams [71] reported that four stages of ADPCM asynchronously coding resulted in an error rate of 6.9×10^{-6}. The data error rate rised rapidly for additional asynchronous codings. Therefore, four stages would be a practical upper limit so as not to exceed 10^{-5}.

Petr's ADPCM-DLQ coder [46] could transmit VBD signals at 4800 bits/s with very low modem error rate. For four asynchronous tandem encodings,

$BER = 3.7 \times 10^{-6}$, $BLER = 9.6 \times 10^{-4}$ were reported. For four synchronous tandem encodings, $BER = 1.1 \times 10^{-7}$, $BLER = 2.4 \times 10^{-5}$ were noted.

When the performance of the CCITT ADPCM coder with VBD was measured [78], it was observed that for the modems operating at data rates less than or equal to 2400 bits/s, performance differences between ADPCM and PCM are considered minimum. For the 4800 bits/s modems tested, performance with ADPCM is comparable to PCM under most of the conditions; However, under some of the more stringent conditions (e.g. four asynchronous codings), degradations associated with ADPCM should be recognized. Besides, no significant performance differences were found between one and four synchronous ADPCM codings for all the VBD tests.

Most tests revealed performance limitations associated with the transmission of 9.6 kbits/s VBD signals of 32 kbits/s ADPCM. A possible approach to these limitations is to increase the bit rate to 40 kbits/s, as suggested by Mermelstein and Millar [77]. In fact, in its questionnaire sent to elicit industry opinion, the Exchange Carriers Standards Association mentioned 40 kbits/s ADPCM as one possibility to enhance the performance [79]. Benvenuto and Daumer [80] attempted to accomodate 9.6 kbits/s modem signals through 32 kbits/s ADPCM links. Reasoning that VBD signals have little spectral energy beyond 3.2 kHz, the authors decreased the sampling rate to 6.4 kHz, and used a 5-bit quantizer. Acceptable performance was found with two ADPCM codings in the presence of mild envelope delay distortion. They suggested to use a 40 kbits/s rate if acceptable performance under conditions of multiple ADPCM codings and analog impairments is required. The authors also proposed another approach where a few parameters of the standard ADPCM algorithm are changed. The bit rate remains to be 32 kbits/s. However, 2^{-2}, instead of 2^{-3}, is used in Eq. (44) to calculate $a_p(k)$. Moreover, 2^{-9} replaces 2^{-8} in Eq. (49)

for the computation of the zero coefficients of the predictor. Acceptable performance (i.e. $BLER \leq 10^{-2}$) was obtained in the presence of analog impairments after one ADPCM coding.

5.4. Lower Bit-Rate ADPCM

Low bit-rate coding remains to be one of the goals researchers in digital communications have ever wanted to achieve. Evidently, the technique can reduce per-channel cost of digital transmission facilities or reduce the required bandwidth in radio applications where storage cost is directly proportional to the coding bit rate. 32 kbits/s ADPCM already reduces the bit rate of PCM by half. In order to study the performance of 24 kbits/s ADPCM, where 3-bit, instead of 4-bit, code is used, modifications have been made to the quantization and the $F[I(k)]$ tables as well as the program code. For $B = 3$ and $N = 8$, the following Max table is used :

e(k)	$e_q(k)$
0.	0.2451
0.5006	0.7560
1.050	1.344
1.748	2.152

Table XIX. Max quantizer input/output (B=3).

The values are normalized and then converted to logarithm as shown in Tables XX and XXI.

e(k)	$e_q(k)$
0.	0.4896
1.	1.5101
2.0975	2.685
3.492	4.299

Table XX. Normalized quantizer input/output (B=3).

$\log_2 \|e(k)\|$	$\log_2 \|e_q(k)\|$	$\|I(k)\|$
0.	-1.03032	0
-0.05	0.594644	1
1.06867	1.42492	2
1.80405	2.104	3

Table XXI. Normalized quantizer input/output (log value) (B=3).

The $F[I(k)]$ values utilized are as follows :

$\|I\|$	3	2	1	0
$F[I(k)]$	7	3	1	0

The TMS32020 is again used as the processor.

With the look-up tables implemented, the 24 kbits/s ADPCM system's busy time has been decreased by 11.50%.

When both fixed poles and tables are used in the 24 kbits/s ADPCM system, the execution time is improved by 34.51%.

As can be observed, the 24 kbits/s ADPCM codec can save as much execution time as the 32 kbits/s ADPCM.

CHAPTER VI

CONCLUSION

In order to improve bandwidth efficiency in transmitting speech and other voiceband signals, and hence to reduce the costs of transport for voice and voiceband services, ADPCM algorithms have been proposed and implemented. Indeed, at 32 kbits/s, ADPCM solution provides double the channel capacity of the current 64 kbits/s PCM technique. This together with the superior performance and application flexibility of ADPCM, relative to other bandwidth reduction techniques such as Digital Speech Interpolation (DSI), are the prime reason for its selection [81].

This thesis has attempted to present a comprehensive overview of various ADPCM techniques. From this review, it can be inferred that ADPCM is going to play a major role in telecommunications. Especially, now that CCITT has introduced the standard 32 kbits/s ADPCM algorithm, it will be incorporated into new and existing telecommunications equipment for use in network and private line applications [82]. Moreover, there exists the potential to generate a whole new variety of services based on 32 kbits/s . For example, Integrated Services Digital Network (ISDN) is a telecommunication network composed of a wide range of network capabilities to offer services which involve both voice and non-voice applications [83], [84].

After describing the implementation of the CCITT algorithm on the processor TMS32010, the thesis suggested modifications to reduce the system execution time, namely the addition of more look-up tables and the use of fixed poles instead of adaptive poles. Results obtained showed that when the number of look-up tables is increased, the improvement in speed is 11.50%.

If fixed poles are used in the algorithm, we can save 23% of the execution time. When both the added look-up tables and fixed poles are used, 34.51% of the execution time is saved. The above-mentioned modifications have been implemented on a TMS32020, which has instruction cycle time of 200 ns. It is expected that if the same modified techniques are realized on a TMS320C25 with instruction cycle time of 100 ns, about 69% of the execution time would be saved. The significance of this is the system would need to spend at most half of the sampling period to process PCM and ADPCM input signals. As a result, the system has time to perform other tasks, or it can process the input signals of another channel. In the latter case, the channel is consequently doubled.

Attempts have also been made to study the possibility and feasibility of a reduced bit-rate version of the CCITT algorithm. It was observed that 24 kbits/s ADPCM with the same software modifications can save the same amount of execution time as 32 kbits/s.

The thesis has outlined the structure of a processor optimized for the implementation of the CCITT ADPCM algorithm.

Further work could be carried out to study the performance of 32 kbits/s ADPCM systems using the TMS320C25. It is expected that these systems perform as well as those using the TMS32010 or the TMS32020, at a faster speed. Tests could be performed to evaluate the performance of ADPCM systems using 3-bit code. If found satisfactory, 24 kbits/s ADPCM would help reduce the bandwidth and, consequently, the cost in digital communication applications. The factors influencing the tandem capability of ADPCM systems can be the topic of other studies. Once these factors have been identified, would a compromise design improve the system tandem property and still retain acceptable performance? Finally, further research can be done to realize the optimized processor outlined in this thesis.

REFERENCES

1. Jayant, N.S., "Digital Coding of Speech Waveforms : PCM, DPCM and DM Quantizers", *Proceedings of the IEEE*, Vol.62, No.5, May 1974, pp. 611-632.
2. O'Neal Jr., J.B., "Waveform Encoding of Voiceband Data Signals", *Proceedings of the IEEE*, Vol.68, No.2, February 1980, pp. 232-247.
3. Noll, P., "A Comparative Study of Various Schemes for Speech Encoding", *Bell System Technical Journal*, Vol.54, No.9, November 1975, pp. 1597-1614.
4. Gibson, J.D. and Sauter, L.C., "Experimental Comparison of Forward and Backward Adaptive Prediction in DPCM", *IEEE International Conference on Acoustics, Speech and Signal Processing* , Denver, April 1980, pp. 508-511.
5. Un, C.K. and Cynn, M.H., "A Performance Comparison of ADPCM and ADM Coders", *National Telecommunications Conference 80 Conference Record NTC*, Houston, Texas, November 30-December 4, 1980, pp. 50.1.1-50.1.5.
6. Flanagan, J.L., Schroeder, M.R., Atal, B.S., Crochiere, R.E., Jayant, N.S., and Tribolet, J.M., "Speech Coding", *IEEE Transactions on Communications*, Vol.COM-27, No.4, April 1979, pp. 710-736.
7. Narasimha, M.J. and Foster, S.R., "ADPCM Comes of Age", *Telephone Engineer and Management*, September 1984.
8. Bylanski, P. and Chong, T.W., "New Developments in Speech Coding for Communications", *Telephony*, October 1984.
9. Oliver, B.M., Pierce, J.R., and Shanon, C.E., "The Philosophy of PCM", *Proceedings of the IRE*, Vol.36, November 1948, pp. 1324-1331.

10. Rabiner, L.R. and Schafer, R.W., *Digital Processing of Speech Signals*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978.

11. Raulin, J.M., Bonnerot, G., Jeandot, J.L., and Lacroix, R., "A 60 Channel PCM-ADPCM Converter", *IEEE Transactions on Communications* , Vol.COM-30, No.4, April 1982, pp. 567-573.

12. Proakis, J.G., *Digital Communications*, McGraw-Hill Book Company, New York, 1983.

13. Max, J., "Quantizing for Minimum Distortion", *IRE Transactions on Information Theory*, Vol.IT-6, March 1960, pp. 7-12.

14. Paez, M.D. and Glisson, T.H., "Minimum Mean-Squared-Error Quantization in Speech PCM and DPCM Systems", *IEEE Transactions on Communications*, Vol.COM-20, April 1972, pp. 225-230.

15. McDonald, R.A., "Signal-to-Noise and Idle Channel Performance of Differential Pulse Code Modulation Systems - Particular Applications to Voice Signals", *Bell System Technical Journal*, September 1966, pp. 1123-1151.

16. Makhoul, J., "Linear Prediction : A Tutorial Review", *Proceedings of the IEEE*, Vol.63, No.4, April 1975, pp. 561-580.

17. Gibson, J.D., "Adaptive Prediction in Speech Differential Encoding Systems", *Proceedings of the IEEE*, Vol.68, No.4, April 1980, pp. 488-525.

18. Nasr, M.E.M. and Chakravarthy, C.V., "Hybrid Adaptive Quantization for Speech Coding", *IEEE Transactions on Communications* , Vol.COM-32, No.12, December 1984, pp. 1358-1361.

19. Cohen, F., "A Switched Quantizer for Markov Sources Applied to Speech Signals", *Nachrichtentechnische Zeitschrift*, Heft 11, 1973, pp. 520-522.

20. Cointot, D. and De Passoz, G., "A 60-Channel PCM-ADPCM Converter Robust to Channel Errors", *IEEE International Conference on Communications*, 1982, pp. A8.5.1-A8.5.5.

21. Dietrich, M., "Coding of Speech Signals Using a Switched Quantizer", *Proc. Int. Zürich Seminar on Digital Communications*, Zürich, 1974, pp. A4(1)-A4(4).

22. Noll, P., "Adaptive Quantizing in Speech Coding Systems", *Proc. Int. Zürich Seminar on Digital Communications*, Zürich, 1974, pp. B3(1)-B3(6).

23. Cummiskey, P., Jayant, N.S., and Flanagan, J.L., "Adaptive Quantization in Differential PCM Coding of Speech", *Bell System Technical Journal*, Vol.52, No.7, September 1973, pp. 1105-1118.

24. Jayant, N.S., "Adaptive Quantization With a One-Word Memory", *Bell System Technical Journal*, Vol.52, No.7, September 1973, pp. 1119-1144.

25. Gibson, J.D., Jones, S.K., and Melsa, J.L., "Sequentially Adaptive Prediction and Coding of Speech Signals", *IEEE Transactions on Communications*, Vol.COM-22, No.11, November 1974, pp. 1789-1796.

26. Gibson, J.D. and Cross, E.A., "Fixed-Tap ADPCM System Divergence and a Bound on the Robust Quantizer Overload Point", *IEEE Transactions on Communications*, Vol.COM-26, No.6, June 1978, pp. 827-832.

27. Gibson, J.D., "Sequentially Adaptive Backward Prediction in ADPCM Speech Coders", *IEEE Transactions on Communications*, Vol.COM-26, No.1, January 1978, pp. 145-150.

28. Gibson, J.D., "Quantization Noise Filtering in ADPCM Systems", *IEEE Transactions on Systems, Man and Cybernetics*, Vol.SMC-10, No.8, August 1980, pp. 529-536.

29. Gibson, J.D., Berglund, V.P., and Sauter, L.C., "Kalman Backward Adaptive Predictor Coefficient Identification in ADPCM with PCQ", *IEEE Transactions on Communications*, Vol.COM-28, No.3, March 1980, pp. 361-371.

30. Evci, C.C., Xydeas, C.S., and Steele, R., "Sequential Adaptive Predictors for ADPCM Speech Encoders", *National Telecommunications Conference 81 Conference Record NTC*, New Orleans, Louisiana, November 29-December 3, 1981, pp. E8.1.1-E8.1.5.

31. Ramamoorthy, V. and Jayant, N.S., "Enhancement of ADPCM Speech by Adaptive Postfiltering", *AT & T Bell Laboratories Technical Journal*, Vol.63, No.8, October 1984, pp. 1465-1475.

32. Sayood, K., "A Postfiltered DPCM Scheme", *IEEE Transactions on Communications*, Vol.COM-33, No.9, September 1985, pp. 1019-1021.

33. Fukasawa, A., Hosoda, K., Miyamoto, R., and Sugihara, H., "A 32 kbps ADPCM Codec Based on a New Algorithm", *IEEE Global Telecommunications Conference*, San Diego, November 28-December 1, 1983, pp. 204-208.

34. Goodman, D.J. and Wilkinson, R.M., "A Robust Adaptive Quantizer", *IEEE Transactions on Communications*, Vol.COM-23, November 1975, pp. 1362-1365.

35. Gibson, J.D. and Berglund, V.P., "Backward Adaptive Predictor Coefficient Identification in ADPCM with Robust Quantization and PCQ", *National Telecommunications Conference 78 Conference Record NTC* , Birmingham, Alabama, December 3-6, 1978, pp. 46.3.1-46.3.5.

36. Moore, C.C. and Gibson, J.D., "Self-Orthogonal Convolutional Coding for the DPCM-AQB Speech Encoder", *IEEE Transactions on Communications*, Vol.COM-32, No.8, August 1984, pp. 980-982.

37. Reininger, R.C. and Gibson, J.D., "Backward Adaptive Lattice and Transversal Predictors in ADPCM", *IEEE Transactions on Communications*, Vol.COM-33, No.1, January 1985, pp. 74-82.

38. Kim, H.D. and Un, C.K., "An ADPCM System with Improved Error Control", *IEEE Global Telecommunications Conference*, San Diego, November 28-December 1, 1983, pp. 1369-1373.

39. Scagliola, C., "Evaluation of Adaptive Speech Coders under Noisy Channel Conditions", *Conference Record 1978 International Conference on Communications*, Toronto, June 4-7, 1978, pp. 8.1.1-8.1.5.

40. Castellino, P., Modena, G., Nebbia, L., and Scagliola, C., "Bit Rate Reduction by Automatic Adaptation of Quantizer Step-Size in DPCM Systems", *Proc. Int. Zürich Seminar on Digital Communications* , Zürich, 1974, pp. B6(1)-B6(6).

41. Scagliola, C., "An Adaptive Speech Coder with Channel Error Recovery", *Conference Record 1977 International Conference on Communications*, Chicago, June 12-15, 1977, pp. 13.8.317-13.8.32

42. Chakravarthy, C.V., Georganas, N.D., and Shiva, S.G.S., "An Incremental Adaptive Quantizer : A Novel Quantization Scheme", *IEEE Transactions on Communications*, Vol.COM-29, No.7, July 1981, pp. 1056-1061.

43. Nasr, M.E.M. and Chakravarthy, C.V., "ADPCM for Speech with Hybrid Adaptive Quantization", *IEEE International Conference on Communications 1985*, Chicago, June 23-26, 1985, pp. 14.2.1-14.2.5.

44. Qureshi, S.U.H. and Forney, G.D., "A 9.6/16 Kb/s Speech Digitizer", *Conference Record 1975 IEEE International Conference on Communications*, June 1975, pp. 30.31-30.36.

45. Evci, C.C., Steele, R., and Xydeas, C.S., "DPCM-AQF Using Second-Order Adaptive Predictors for Speech Signals", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol.ASSP-29, No.3, June 1981, pp. 337-341.

46. Petr, D.W., "32 Kbps ADPCM-DLQ Coding for Network Applications", *IEEE Global Telecommunications Conference*, Miami, November 29- December 2, 1982, pp. A8.3.1-A8.3.5.

47. Millar, D. and Mermelstein, P., "Prevention of Predictor Mistracking in ADPCM Coders", *Proceedings of the International Conference on Communications - ICC84, Links for the Future*, 1984, pp. 1508-1512.

48. Frangoulis, E.D., Yoshida, K., and Turner, L.F., "Adaptive Differential Pulse-Code Modulation with Adaptive Bit Allocation", *IEE Proceedings*, Vol.131, Pt.F., No.5, August 1984, pp. 542-548.

49. Gibson, J.D., "Comparisons and Analyses of Forward and Backward Adaptive Prediction in ADPCM", *National Telecommunications Conference 78 Conference Record NTC*, Birmingham, Alabama, December 3-6, 1978, pp.19.2.1-19.2.5.

50. Raulin, J.M. and Jeandot, J.L., "A 32 kbit/s PCM to ADPCM Converter", *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 82*, Paris, 1982, pp. 968-971.

51. Nishitani, T., Aikoh, S., Araseki, T., Ozawa, K., and Maruta, R., "A 32 kb/s Toll Quality ADPCM Codec Using a Single Chip Signal Processor", *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 82*, Paris, 1982, pp. 960-963.

52. Maruta, R., Aikoh, S., Nishitani, T., and Kawayachi, N., "Per-Channel ADPCM Codec for Multi-Purpose Applications", *IEEE Global Telecommunications Conference*, Miami, November 29-December 2, 1982, pp. A8.4.1-A8.4.5.

53. Cointot, D., "A 32-Kbit/sec ADPCM Coder Robust to Channel Errors", *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 82*, Paris, 1982, pp. 964-967.

54. Johnson, C.R., Lyons, J.P., and Heegard, C., "A New Adaptive Parameter Estimation Structure Applicable to ADPCM", *IEEE International Conference on Acoustics, Speech and Signal Processing*, Boston, 1983, pp. 1.1.1-1.1.4.

55. Heegard, C., Johnson, C.R., and Lyons, J.P., "Quantizer Effects in RML-Based ADPCM", *The 22nd IEEE Conference on Decision and Control*, Texas, December 1983, pp. 709-714.

56. Gibson, J.D. and Reininger, R.C., "Adaptive Prediction with Quantized Data", *The 22nd IEEE Conference on Decision and Control* , Texas, December 1983, pp. 715-721.

57. Honig, M.L. and Messerschmitt, D.G., "Comparison of Adaptive Linear Prediction Algorithms in ADPCM", *IEEE Transactions on Communications*, Vol.COM-30, No.7, July 1982, pp. 1775-1785.

58. Adoul, J.P., Debray, J.L., and Dalle, D., "Spectral Distance Measure Applied to the Optimum Design of DPCM Coders with L Predictors", *IEEE International Conference on Acoustics, Speech and Signal Processing*, Denver, April 1980, pp. 512-515.

59. O'Neal, J.B. and Stroh, R.W., "Differential PCM for Speech and Data Signals", *IEEE Transactions on Communications*, Vol.COM-20, No.5, October 1972, pp. 900-912.

60. CCITT Red Books, Recommendation G.711, Pulse Code Modulation (PCM) of Voice Frequencies, Geneva, 1972.

61. CCITT Red Books, Recommendation G.721, 32 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM), Malaga-Torremolinos, October 1984, pp. 125-159.

62. Daumer, W.R., Mermelstein, P., Maître, X., and Tokizawa, I., "Overview of the ADPCM Coding Algorithm", *Proc. GLOBECOM '84*, November 1984, pp. 774-777.

63. Charbonnier, A., Maître, X., and Petit, J.P., "A Digital Signal Processor Implementation of the CCITT 32 kbit/s ADPCM Algorithm", *IEEE International Conference on Communications 1985*, Chicago, June 1985, pp. 1197-1201.

64. Matsumura, T., Gambe, H., and Murano, K., "Implementation of a 32 kbit/s ADPCM Codec Using a General-Purpose Digital Signal Processor", *IEEE Journal on Selected Areas in Communications*, Vol.SAC-4, No.1, January 1986, pp. 125-132.

65. Nakamura, M., Suzuki, H., Kamitake, T., Ohnishi, M., Sakai, T., Totoki, T., and Tsuchihashi, H., "A 32 KB/s ADPCM Codec Single-Chip LSI Based on CCITT's G.721 Standard", *IEEE International Conference on Communications 1986*, Toronto, June 1986, pp. 41.5.1-41.5.5.

66. Nishitani, T., Kuroda, I., Satoh, M., Katoh, T., and Aoki, Y., "A CCITT Standard 32 kbit/s ADPCM LSI Codec", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.ASSP-35, No.2, February 1987, pp. 219-225.

67. Hangartner, R.D. and Jain, V.K., "Efficient Micro processor Based Tandemable 32 kbit/s ADPCM/PCM Transcoder", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.ASSP-35, No.8, August 1987, pp. 1203-1207.

68. Reimer, J., McMahan, M., and Arjmand, M., "32-kbit/s ADPCM with the TMS32010", *Digital Signal Processing Applications with the TMS920 Family*, Texas Instruments Incorporated, 1986, pp. 469-497.

69. Texas Instruments Incorporated, "Design Example : 32-kbit/s ADPCM Using the TMS32010", (Private Communication), 1986.

70. Vo, M.T., "An ADPCM System Based on the TMS32010", Technical Report, Concordia University, 1986.

71. Mermelstein, P. and Williams, G., "Network Performance Issues in 32 Kb/s Coding of Speech and Voiceband Data", *IEEE Global Telecommunications Conference*, Miami, November 29-December 2, 1982, pp. 234-238.

72. CCITT COM XVIII (1981-1984), Question 7/XVIII, Contribution No. 120-E, (AT&T), "32 Kb/s ADPCM-DLQ Coding, September 1, 1982.

73. Taka, M., Maruta, R., and Le Guyader, A., "Synchronous Tandem Algorithm for 32 Kbit/s ADPCM", *Proc. GLOBECOM'84*, November 1984, pp.791-795.

74. Williams, G. and Suydcrhoud, H., "Subjective Performance Evaluation of the 32-kbit/s ADPCM Algorithm", *Proc. GLOBECOM'84*, November 1984, pp. 778-785.

75. Ramamoorthy, V., "On Tandem Coding of Speech", *IEEE International Conference on Communications 1985*, Chicago, June 23-26, 1985, pp. 1229-1231.

76. T1Q1/87-020R4, Draft of American National Standard for Telecommunications Network Performance Standards-32 kb/s ADPCM Tandem Encoding Limits, ECSA T1 Committee.

77. Mermelstein, P. and Millar, D.J., "Adaptive Predictive Coding of Speech and Voiceband Data Signals", *Proc. GLOBECOM'82*, pp.972-975.

78. Raulin, J.M., Belfield, W.R., and Nishitani, T., "Objective Test Results for the 32 kb/s ADPCM Coder", *Proc. GLOBECOM'84*, November 1984, pp. 786-790.

79. T1Y1.2/87-056, Questionnaire on the Utilization and Possible Enhancement of the 32 Kbit/s ADPCM Standard, ANSI T1.301-1987, ECSA T1 Committee.

80. Benvenuto, N. and Daumer, W.R., "Waveform Coding of 9.6 kb/s Voiceband Data Signals at 32 kb/s", *Proc. GLOBECOM'87*, 1987, pp. 1700-1703.

81. Psimenatos, N., Gruber, J., Goddard, G., and Mondor, D., "The Application of 32 kb/s ADPCM Systems in Telecommunications Networks", *Proc. GLOBECOM'84*, November 1984, pp. 796-802.

82. Kilm, T. and Sparrell, D., "Introduction of 32 Kb/s ADPCM Equipment into the North American Network", *Proc. GLOBECOM'84*, November 1984, pp. 803-807.

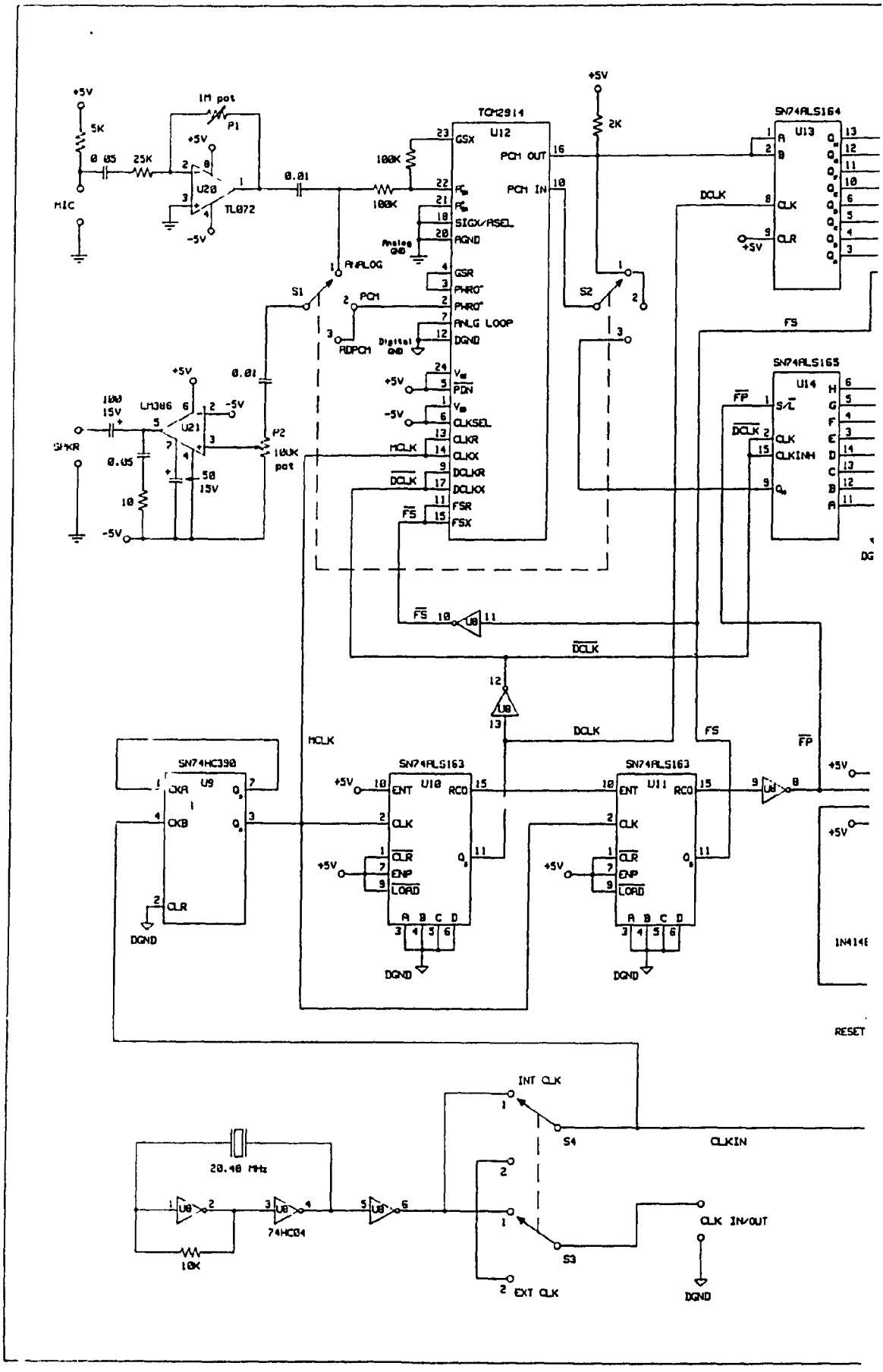
83. Kahl, P., "ISDN-Services. A Review of CCITT Standardization to Date", *IEEE International Conference on Communications 1985*, Chicago, June 23-26, 1985, pp. 1061-1066.

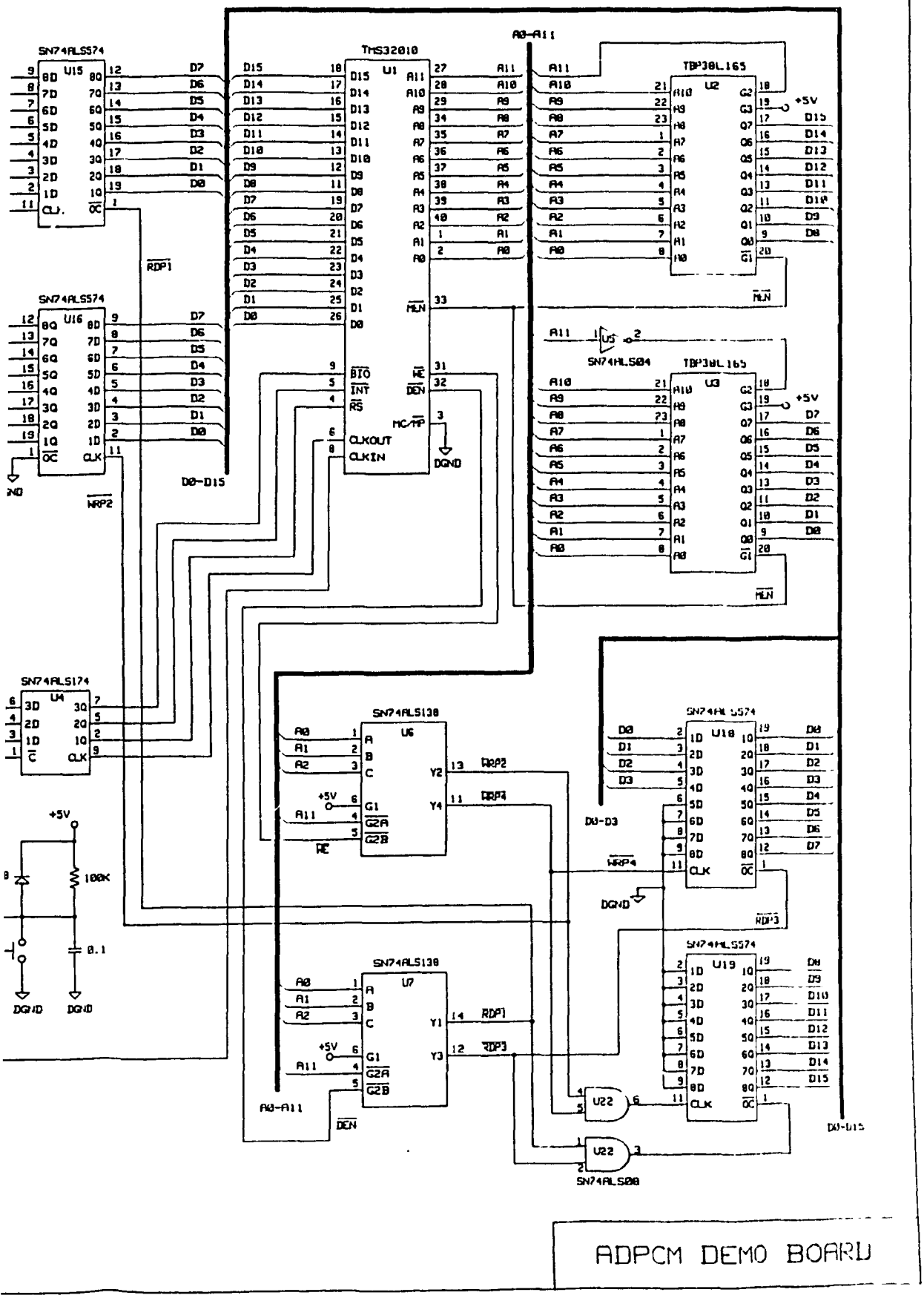
84. Rumsey, D.C., "Support of Existing Data Interfaces by the ISDN", *IEEE International Conference on Communications 1985*, Chicago, June 23-26, 1985, pp. 1088-1092.

APPENDIX 1

SCHEMATIC DIAGRAM

The next page shows the schematic diagram of the ADPCM system using the digital signal processor TMS32010.


+5V
1M pot
P1
5K
0.05
25K
MIC
U20
TL072
-5V
0.01
100K
TCM2914
U12
GSX
PCM OUT
PCM IN
SIGX/RSEL
AGND
GSR
FSR
FSX
ANLG LOOP
DGND
ANALOG
PCM
DPCM
S1
S2
2K
SN74ALS164
U13
DCLK
CLK
CLR
FS
SN74ALS165
U14
CLKINH
FP
SPKR
LM386
U21
P2
10K pot
10
50
15V
MCLK
SN74HC390
U9
SN74ALS163
U10
U11
ENT
RCO
ENP
LOAD
RESET
IN414E
20.48 MHz
74HC04
10K
INT CLK
S4
CLKIN
S3
CLK IN/OUT
EXT CLK



ADPCM DEMO BOARD

APPENDIX 2

SOURCE CODE OF ADPCM PROGRAM

Following is a listing of the program used in the TMS32010-based ADPCM system. It has been written in TMS32010 assembly language.

```
     IDT  'ADPCM'
     OPTION    XREF
*;
*;*************************************************************
*;*************************************************************
*; This is the source module for a full duplex 32Kbps ADPCM
*; system. This system is modelled after the CCITT standard
*; but is not identically bit compatible and does not
*; include synchronous coding adjustment in the receiver
*; (this could be added). The loading on the system
*; averages about 98%. The system reads samples on an
*; interrupt basis from a mu-law codec (port 1) every 125
*; microsec and outputs an ADPCM sample to port 4. During
*; each interrupt, the system will also read a 4 bit ADPCM
*; sample from port 3 and output an 8 bit mu-law PCM sample
*; to port 2.
*;
*;*************************************************************
*;*************************************************************
     PAGE
*;
*; System I/O channel assignments
*;
ADC  EQU  1
DAC  EQU  2
IN32K     EQU  3
OUT32K    EQU  4
*;
     PAGE
*;*************************************************************
     AORG 0
*;
     B    RESET
*;
*;*************************************************************
*; INTERRUPT HANDLING ROUTINE -- SYSTEM HANDLES CODEC
*; INTERRUPTS ON A SAMPLE BY SAMPLE BASIS. DURING
*; EACH SAMPLE PERIOD BOTH A RECEIVE AND A TRANSMIT CYCLE
*; ARE EXECUTED.
*;*************************************************************
*;
*;first execute code to transmit a 4-bit adpcm sample
*;
INTRPT    IN   SAMPLE,ADC     ; Get sample
*;
*; patch to stop simulation
*;
     OUT  SCRACH,DAC
     LAC  SAMPLE         ; linearize
     XOR  M255
     ADD  CODEAD
```

```
c     TBLR SAMPLE
*;
      PAGE
*;
*;*************************************************************
*; SIGDIF
*;
*; COMPUTE SCALE FACTOR AND  BOTH PARTIAL AND FULL SIGNAL
*; ESTIMATE FROM PREVIOUS SAMPLES DATA -- THEN COMPUTE
*; DIFFERENCE SIGNAL
*;*************************************************************
*;
*; compute SEZ-- partial signal estimate
*;
*;  SEZ(k) = B1(k-1)*DQ(k-1) + ... + B6(k-1)*DQ(k-6)
*;
SIGDIF    ZAC
      LT    DQ5
      MPY   B6
      LTD   DQ4
      MPY   B5
      LTD   DQ3
      MPY   B4
      LTD   DQ2
      MPY   B3
      LTD   DQ1
      MPY   B2
      LTD   DQ
      MPY   B1
      APAC
*;
*; shift left by 1 to adjust decimal point
*;
      SACL TEMP1
      SACH TEMP2
      ADDH TEMP2
      ADDS TEMP1
      SACH SEZ,1
*;
*; now compute signal estimate as
*;
*;  A1(k-1)*SR(k-1) + A2(k-1)*SR(k-2) + SEZ(k)
*;
GETSE     LAC  SEZ,14
      LT    SR1
      MPY   A2
      LTD   SR
      MPY   A1
      APAC
*;
*; shift left by 1 to adjust decimal point
```

```
*;
     SACL TEMP1
     SACH TEMP2
     ADDH TEMP2
     ADDS TEMP1
     SACH SE,1
*;
*; limit speed control parameter: AL===> 1.Q6
*;
*; APP : 2.Q8   AL: Q6
*;
*;   AL = 1 (64)    if APP >  1 (256)
*;   AL = APP  if APP <= 1
*;
LIMA LAC  ONE,12
     SACL AL
     LAC  APP
     SUB  ONE,8
     BGEZ MIX        ;APP >= 256
     LAC  APP,4
     AND  MFFC0
     SACL AL         ;APP  < 256
*;
*; form linear combination of fast and slow scale factors
*;
*; Y(k) = (1-AL(k))*YL(k-1) + AL(k)*YU(k-1)
*;
MIX  LAC  YU
     SUB  YLH
     SACL TEMP1
     LT   AL
     MPY  TEMP1
     PAC
     ADD  YLH,12
     SACH Y,4
     LAC  Y,14
     SACH TEMP3
*;
*; get difference signal:   D = SAMPLE - SE
*;
     LAC  SAMPLE          ; compute difference sig
     SUB  SE
     SACH TEMP4
*;
*;*********************************************************
*;  ADAPTIVE QUANTIZER --- 62 CLOCKS
*;
*;   input:    differenced PCM speech sample--D
*;        scale factor--Y
*;    output:  4-b quantized diff signal--I
*;*********************************************************
*;
```

```
*; first get log of difference signal
*;
AQUAN     ABS
     SACL TEMP1
GETEXP    SUB  ONE,8      ; binary search to get exponent
     BGEZ C8TO14
C0TO7     ADD  M15,4      ; TEMP1-16      exp = 0-7
     BGEZ C4TO7
C0TO3     ADD  THREE,2 ; TEMP1-4    exp = 0-3
     BGEZ C2TO3
C0TO1     ADD  ONE,1      ; TEMP1-2 exp = 0-1
     BGEZ EXP1
EXP0 LARK 0,0  ; exp = 0
     LAC  TEMP1,7
     B    GETMAN     ; save exponent and get mantissa
EXP1 LARK 0,1  ; exp = 1
     LAC  TEMP1,6
     B    GETMAN
C2TO3     SUB  ONE,2      ; TEMP1-8 exp = 2-3
     BGEZ EXP3
EXP2 LARK 0,2  ; exp = 2
     LAC  TEMP1,5
     B    GETMAN
EXP3 LARK 0,3  ; exp = 3
     LAC  TEMP1,4
     B    GETMAN
C4TO7     SUB  THREE,4 ; TEMP1-64  exp = 4-7
     BGEZ C6TO7
C4TO5     ADD  ONE,5      ; TEMP1-32      exp = 4-5
     BGEZ EXP5
EXP4 LARK 0,4  ; exp = 4
     LAC  TEMP1,3
     B    GETMAN
EXP5 LARK 0,5  ; exp = 5
     LAC  TEMP1,2
     B    GETMAN
C6TO7     SUB  ONE,6      ; TEMP1-128     exp = 6-7
     BGEZ EXP7
EXP6 LARK 0,6  ; exp = 6
     LAC  TEMP1,1
     B    GETMAN
EXP7 LARK 0,7  ; exp = 7
     LAC  TEMP1
     B    GETMAN
C8TO14    SUB  M15,8      ; TEMP1-4096    exp = 8-14
     BGEZ CCTOE
C8TO11    ADD  THREE,10 ;TEMP1-1024      exp = 8-11
     BGEZ CATOB
C8TO9     ADD  ONE,9      ; TEMP1-512     exp = 8-9
     BGEZ EXP9
EXP8 LARK 0,8  ; exp = 8
     LAC  TEMP1,15
```

```
      SACH TEMP1
      LAC  TEMP1
      B    GETMAN
EXP9  LARK 0,9   ; exp = 9
      LAC  TEMP1,14
      SACH TEMP1
      LAC  TEMP1
      B    GETMAN
CATOB      SUB  ONE,10     ; TEMP1-2048    exp = 10-11
      BGEZ EXP11
EXP10      LARK 0,10 ; exp = 10
      LAC  TEMP1,13
      SACH TEMP1
      LAC  TEMP1
      B    GETMAN
EXP11      LARK 0,11 ; exp = 11
      LAC  TEMP1,12
      SACH TEMP1
      LAC  TEMP1
      B    GETMAN
CCTOE      SUB  THREE,12 ;TEMP1-16384    exp = 12-14
      BGEZ EXP14
CCTOD      ADD  ONE,13     ; TEMP1-8192    exp = 13-14
      BGEZ EXP13
EXP12      LARK 0,12 ; exp = 12
      LAC  TEMP1,11
      SACH TEMP1
      LAC  TEMP1
      B    GETMAN
EXP13      LARK 0,13 ; exp = 13
      LAC  TEMP1,10
      SACH TEMP1
      LAC  TEMP1
      B    GETMAN
EXP14      LARK 0,14 ; exp = 14
      LAC  TEMP1,9
      SACH TEMP1
      LAC  TEMP1
GETMAN     AND  M127
      SAR  0,TEMP1
      ADD  TEMP1,7
*;
*; scale by subtraction
*;
SUBTB      ADD  ONE,11
      SUB  TEMP3
*;
*; 4b quantizer
*;
*; QUANT TABLE FOR 32KB OUTPUT (OFFSET: 2048)
*;
ITAB1      EQU  2041
```

```
ITAB2     EQU  2171
ITAB3     EQU  2250
ITAB4     EQU  2309
ITAB5     EQU  2358
ITAB6     EQU  2404
ITAB7     EQU  2453
*;
QUAN SUB  K2309     ; ITAB4
     BGEZ CI4TO7
CI0TO3    ADD  K138 ; ITAB2   I = 0-3
     BGEZ CI2TO3
CI0TO1    ADD  K130 ; ITAB1   I = 0-1
     BGEZ IEQ1
IEQ0 LACK 0
     B    GETIM
IEQ1 LACK 1
     B    GETIM
CI2TO3 SUB     K79  ; ITAB3   I = 2-3
     BGEZ IEQ3
IEQ2 LACK 2
     B    GETIM
IEQ3 LACK 3
     B    GETIM
CI4TO7    SUB  K95  ; ITAB6   I = 4-7
     BGEZ CI6TO7
CI5TO6    ADD  K46  ; ITAB5   I = 5-6
     BGEZ IEQ5
IEQ4 LACK 4
     B    GETIM
IEQ5 LACK 5
     B    GETIM
CI6TO7    SUB  K49  ; ITAB6   I = 6-7
     BGEZ IEQ7
IEQ6 LACK 6
     B    GETIM
IEQ7 LACK 7
GETIM     SACL IM
     XOR  TEMP4
     AND  M15
     SACL I
OUTI OUT  I,OUT32K  ; XMITTER OUTPUT
*;
*;************************************************************
*; INVERSE ADAPTIVE QUANTIZER
*;
*;   input:    4b quantized samples     -- IM
*;        scale factor          -- Y
*;   output: reconstructed diff sig     -- DQ
*;************************************************************
*;
*; first convert quantized difference back to log domain
*;
```

```
IAQUAN    LAC  IM
     ADD  INQTAB    ; reconst table
     TBLR TEMP1
*;
*; add back scale factor
*;
ADDA LAC  TEMP1,5
     ADD  TEMP3,5   ; Y >> 2
     SACH TEMP1,4
     AND  M4095
*;
*; now covert to linear domain
*;
*;
ALOG ADD  ONE,12          ; 1+x
     SACL TEMP2           ; extract mantissa
     LT   TEMP2           ; prepare to shift
     LAC  TEMP1
     ADD  SHIFT           ; look up multiplier
     TBLR TEMP5
     MPY  TEMP5
     PAC
     SACH DQ,4       ; right shift--dqmag
     LAC  DQ
     XOR  TEMP4
     SUB  TEMP4
     SACL DQ
ADDSGN    LAC  TEMP4,12
     ADD  ONE,11
     SACL SDQ
*;
*;************************************************************
*; QUANTIZER SCALE FACTOR ADAPTATION
*;
*;   input:       I : 32KB coded samples
*;   output: YU,YLL: next sample scale factor
*;************************************************************
*;
*; First compute WI
*;
*;   input --IM
*;   output--TEMP1 (WI)
*;
QSFA LAC  WITAB           ; get table address and offset
     ADD  IM
     TBLR TEMP1           ; lookup WI (Q6)
*;
*; Update fast adaptation scale factor -- constant=1/32
*;
*;   YU(k)=(1-2**-5)*Y(k)+(2**-5)*WI(k)
*;
*;   input --TEMP1 (WI--- TC   6...Q4)
```

```
*;   output--YU    (YU---SM     3...-9)
*;
FILTD     LAC  Y,12       ; Y  (Q21)
     SUB  Y,7        ; Y/32    (Q21)
     ADD  TEMP1,12   ; WI/32 (Q21)
     SACH YU,4       ; YU (Q9)
*;
*; limit quant scale factor 1.06 <= YU <= 10.0
*;
*;   input: YU (3...-9)
*;   ouput: YU
*;
LIMB SUB  K544,12         ; check lo threshold
     BGEZ CHKHI
     LAC  K544
     B    STRLIM          ; go store limited value
CHKHI     SUB  K4576,12        ; check hi threshold
     BLEZ FILTE           ; within limits--continue
     LAC  K5120
STRLIM    SACL YU
*;
*; Update slow adaptation scale factor -- constant = 1/64
*;
*; YL(k) = (1-2**-6)*YL(k-1) + 2**-6 * YU(k)
*;
FILTE     LAC  YLH,15          ; shift yl left by 6
     ADD  YLL,9
     SUB  YLH,9
     SUB  YLL,3
     ADD  YU,9
     SACH YLH,1
     AND  M32767
     SACL TEMP2
     LAC  TEMP2,6
     SACH YLL,1
*;
*;*************************************************************
*; ADAPTATION SPEED CONTROL
*;
*;   input:    IM
*;   output: APP
*;*************************************************************
*;
*; first compute FI function: 5 CLOCKS
*;
SPDCRL    LAC  FITAB      ; look-up value of FI function
     ADD  IM
     TBLR TEMP1      ; FI*16
*;
*; now update short term average of FI
*;
*; DMS(k) = (1-2**-5)*DMS(k-1) + 2**-5 * FI(k)
```

```
*;
FILTA     LAC  TEMP1,15  ; FI/32 in q24
     ADD  DMS,15
     SUB  DMS,10         ; DMS/32
     SACH DMS,1
*;
*; update long term average of FI
*;
*; DML(k) = (1-2**-7)*DML(k-1) + 2**-7 * FI(k)
*;
FILTB     LAC  TEMP1,15  ; FI/128 in q26
     ADD  DML,15
     SUB  DML,8          ; DML/128
     SACH DML,1
*;
*; Compute mag of diff of short and long term
*; and perform threshold comparison to compute speed
*; control parameter--low-pass result.
*;
*; APP(k) = (1-2**-4)*APP(k-1) + 2**-3 , if y < 3 or
*;           if |DMS-DML| > 2**-3 * DML
*; else
*;
*; APP(k) = (1-2**-4)*APP(k-1)
*;
FILTC     ZALH APP
     SUB  APP,12
     SACH APP  ; (1-2**-4)*APP
     LAC  Y
     SUB  THREE,9
     BLZ  ADD18
     ZALH DMS        ; DMS (Q25)
     SUB  DML,14          ; DMS-DML
     ABS
     SUB  DML,11
     BLZ  APRED
ADD18     LAC  APP
     ADD  ONE,5
     SACL APP        ; +1/8 in Q8
*;
*;************************************************************
*; ADAPTIVE PREDICTOR: compute signal estimate from quantized
*; difference
*;
*;   input:    DQ   quantized difference
*;   output: SE     signal estimate
*;************************************************************
*;
*; compute reconstructed signal:  DQ + SE
*;
APRED     LAC  DQ
     ADD  SE
```

```
     SACL SR
*;
*; compute coefficients of 6th order predictor
*;
*; Bi(k) = (1-2**-8)*Bi(k-1)
*;      + 2**-7 * SGN[DQ(k)] * SGN[DQ(k-1)]
*;
*;     for i = 1...6
*;     and Bi is implicitly limited to +/- 2
*;
GETB6     LT   SDQ6
     LAC  B6,15      ; Q29
     SUB  B6,7 ; B6 * 2**-7  (Q29)
     MPY  SDQ  ; SGN(SDQ)*SGN(SDQ6)*2**-7 (Q29)
     LTD  SDQ5
     SACH B6,1 ; 1.Q14
GETB5     LAC  B5,15
     SUB  B5,7
     MPY  SDQ
     LTD  SDQ4
     SACH B5,1
GETB4     LAC  B4,15
     SUB  B4,7
     MPY  SDQ
     LTD  SDQ3
     SACH B4,1
GETB3     LAC  B3,15
     SUB  B3,7
     MPY  SDQ
     LTD  SDQ2
     SACH B3,1
GETB2     LAC  B2,15
     SUB  B2,7
     MPY  SDQ
     LTD  SDQ1
     SACH B2,1
GETB1     LAC  B1,15
     SUB  B1,7
     MPY  SDQ
     LTD  SDQ
     SACH B1,1
*;
*; Update coefficients of 2nd order predictor
*;
*; First get sign of sum of quantized diff
*; and partial sig estimate
*;
ADDC DMOV PK1        ; PK1==>PK2
     DMOV PK0        ; PK0==>PK1
     LAC  SEZ
     ADD  DQ
     SACH TEMP1
```

```
     LAC  TEMP1,10
     ADD  ONE,9
     SACL PK0
SUMGT0    LT   PK0
*;
*;  now calculate f[A1(k-1)]  ==> TEMP3 will get 1/2 F
*;
*;      = 4*A1        if |A1|   <= 1/2
*;      = 2*SGN(A1)   if |A1|   >  1/2
*;
GETF LAC  A1,1       ; 2*A1
     SACL TEMP5
     BLZ  GETF2
GETF1     SUB  ONE,14
     BLZ  GETA1
     LAC  ONE,14
     B    DONEF
GETF2     ABS
     SUB  ONE,14
     BLZ  GETA1
     LAC  MINUS,14
DONEF     SACL TEMP5
*;
*;  A1(k) = (1-2**-8)*A1(k-1)
*;        + (3*2**-8)*SGN[p(k)]*SGN[p(k-1)]
*;
GETA1     LAC  A1,12
     SUB  A1,4
     MPY  PK1        ; 3*SGN[p(k-1)*SGN[p(k)]
     APAC
     APAC
     APAC
     SACH A1,4
     PAC        ; save sign of SGN[p(k-1)]*SGN[p(k)]
*;
*; A2(k) = (1-2**-7)*A2(k-1)
*;       + (2**-7)*(SGN[p(k)]*SGN[(p(k-2)]
*;       - f[A1(k-1)]*SGN[p(k)]*SGN[p(k-1)])
*;
GETA2     BGEZ SUBF       ; if sign + --> subtract F
     ZAC             ;  else negate F and subtract
     SUB  TEMP5
     SACL TEMP5
*;
SUBF LAC  A2,12
     SUB  A2,5
     MPY  PK2
     APAC
     APAC
     SUB  TEMP5,6
     SACH A2,4
*;
```

```
*; now limit A2 to +/- .75 and prevent overflow
*;
LIMC LAC  A2
     ABS
     SUB  THREE,12   ; |value| must be < .75
     BLEZ LIMD
     LAC  A2
     BGEZ SATPOS
SATNEG    LAC  MINUS3,12 ; -.75
     B    DONEG
SATPOS    LAC  THREE,12
DONEG     SACL A2
*;
*; limit A1 to +/-  1-2**-4 - A2
*;
LIMD LAC  M15,10
     SUB  A2
     SACL TEMP1          ; 1-2**-4-A2P
     LAC  A1
     SACH TEMP2
     ABS
     SUB  TEMP1
     BLEZ GET32K         ; A1 <= LIMIT
     LAC  TEMP1
     XOR  TEMP2
     SUB  TEMP2
     SACL A1
     PAGE
*;*************************************************
*; now execute receive code
*;*************************************************
*;
GET32K    IN   RI,IN32K   ; read 4-bit sample
*;
*;*************************************************
*; RECEIVER SUBROUTINES
*;*************************************************
*; COMPUTE SCALE FACTOR AND  BOTH PARTIAL AND FULL
*; SIGNAL ESTIMATE FROM PREVIOUS SAMPLES DATA --
*; THEN COMPUTE DIFFERENCE SIGNAL
*;*************************************************
*;
*; compute SEZ-- partial signal estimate
*;
*;  RSEZ(k) = RB1(k-1)*RDQ(k-1) + ... + RB6(k-1)*RDQ(k-6)
*;
SEZSEY    ZAC
     LT   RDQ5
     MPY  RB6
     LTD  RDQ4
     MPY  RB5
     LTD  RDQ3
```

```
      MPY   RB4
      LTD   RDQ2
      MPY   RB3
      LTD   RDQ1
      MPY   RB2
      LTD   RDQ
      MPY   RB1
      APAC
*;
*; shift left by 1 to adjust decimal point
*;
      SACL TEMP1
      SACH TEMP2
      ADDH TEMP2
      ADDS TEMP1
      SACH RSEZ,1
*;
*; now compute signal estimate as
*;
*;  RA1(k-1)*RSR(k-1) + RA2(k-1)*RSR(k-1) + RSEZ(k)
*;
RGETSE  LAC    RSEZ,14
      LT    RSR1
      MPY   RA2
      LTD   RSR
      MPY   RA1
      APAC
*;
*; shift left by 1 to adjust decimal point
*;
      SACL TEMP1
      SACH TEMP2
      ADDH TEMP2
      ADDS TEMP1
      SACH RSE,1
*;
*; limit speed control parameter: AL===> 1.Q6
*;
*; RAPP : 2.Q8
*;
*;    RAL = 1 (64)       if RAPP >  1
*;    RAL = RAPP         if RAPP <= 1
*;
RLIMA      LAC  ONE,12
      SACL RAL
      LAC  RAPP
      SUB  ONE,8
      BGEZ RMIX       ;RAPP >= 256
      LAC  RAPP,4
      AND  MFFC0
      SACL RAL        ;RAPP  < 256
*;
```

```
*; form linear combination of fast and slow scale factors
*;
*; RY(k) = (1-RAL(k))*RYL(k-1) + RAL(k)*RYU(k-1)
*;
RMIX LAC  RYU
     SUB  RYLH
     SACL TEMP1          ; low half
     LT   RAL
     MPY  TEMP1
     PAC             ; RYL + RAL*(RYU-(RYLL>>6))
     ADD  RYLH,12
     SACH RY,4
     LAC  RY,14
     SACH TEMP3
*;
*;***********************************************************
*; INVERSE ADAPTIVE QUANTIZER
*;
*;   input:    4b quantized samples     -- RI
*;         scale factor (QADAPT)    -- RY
*;   output: reconstructed diff sig     -- RDQ
*;***********************************************************
*;
*; first convert received nibble back to log domain
*;
RIAQUA    LAC  RI
     SACL RIM
     XOR  M15
     SUB  ONE,3
     SACH TEMP4
     BGEZ POSRI
     ADD  ONE,3
     SACL RIM
POSRI     LAC  RIM
     ADD  INQTAB    ; reconst table
     TBLR TEMP1
*;
*; add back scale factor
*;
RADDA     LAC  TEMP1,5
     ADD  TEMP3,5   ; Y >> 2
     SACH TEMP1,4
     AND  M4095
*;
*; now covert to linear domain
*;
RALOG     ADD  ONE,12          ; 1+x
     SACL TEMP2          ; extract mantissa
     LT   TEMP2          ; prepare to shift
     LAC  TEMP1
     ADD  SHIFT          ; ptr to table of multipliers
     TBLR TEMP5
```

```
      MPY  TEMP5          ; integer multiply
      PAC
      SACH RDQ,4          ; dqmag
      LAC  RDQ
      XOR  TEMP4
      SUB  TEMP4
      SACL RDQ
      LAC  TEMP4,12
      ADD  ONE,11
      SACL RSDQ
*;
*;*****************************************************
*; QUANTIZER SCALE FACTOR ADAPTATION
*;
*;    input:      RI : 32KB coded samples
*;    output: RYU,RYLL: next sample scale factor
*;*****************************************************
*; First compute WI
*;
*;    input --RIM
*;    output--TEMP1 (WI)
*;
RQSFA      LAC  WITAB          ; get table address and offset
      ADD  RIM
      TBLR      TEMP1         ; lookup WI (Q6)
*;
*; Update fast adaptation scale factor -- constant=1/32
*;
*;    RYU(k)=(1-2**-5)*Y(k)+(2**-5)*WI(k)
*;
*;    input --TEMP1  (WI--- TC   6...Q4)
*;    output--RYU    (RYU---SM    3...-9)
*;
RFILTD     LAC  RY,12          ; RY      (Q21)
      SUB  RY,7       ; RY/32   (Q21)
      ADD  TEMP1,12   ; WI/32 (Q21)
      SACH RYU,4          ; RYU     (Q9)
*;
*; limit quant scale factor 1.06 <= RYU <= 10.0 : 9 CLOCKS
*;
*;    input: RYU     (3...-9)
*;    ouput: RYU
*;
RLIMB      SUB  K544,12        ; check lo threshold
      BGEZ RCHKHI
      LAC  K544
      B    RSTRLI         ; go store limited value
RCHKHI     SUB  K4576,12       ; check hi threshold
      BLEZ RFILTE         ; within limits--continue
      LAC  K5120
RSTRLI     SACL RYU
*;
```

```
*; Update slow adaptation scale factor -- constant = 1/64 : 28
CLOCKS
*;
*; RYL = (1-2**-6)*RYL + 2**-6 * RYU
*;
RFILTE  LAC    RYLH,15
     ADD  RYLL,9
     SUB  RYLH,9
     SUB  RYLL,3
     ADD  RYU,9
     SACH RYLH,1
     AND  M32767
     SACL TEMP2
     LAC  TEMP2,6
     SACH RYLL,1
*;
*;*********************************************************
*; ADAPTATION SPEED CONTROL
*;
*;   input:     RIM
*;   output:    RAPP
*;*********************************************************
*;
*; first compute FI function
*;
RSPDCR    LAC  FITAB     ; look-up value of FI function
     ADD  RIM
     TBLR TEMP1     ; FI*16
*;
*; now update short term ave age of FI
*;
*; RDMS(k) = (1-2**-5)*RDMS(k-1) + 2**-5 * FI(k)
*;
RFILTA    LAC  TEMP1,15  ; FI/32 in q24
     ADD  RDMS,15
     SUB  RDMS,10        ; RDMS/32
     SACH RDMS,1
*;
*; update long term average of FI
*;
*; RDML(k) = (1-2**-7)*RDML(k-1) + 2**-7 * FI(k)
*;
RFILTB    LAC  TEMP1,15  ; FI/128 in q26
     ADD  RDML,15
     SUB  RDML,8         ; RDML/128
     SACH RDML,1
*;
*; Compute mag of diff of short and long term functions
*; of quantizer output sequence and perform threshold
*; comparison to compute speed control parameter--low-pass
*; result.
*;
```

```
*; RAPP(k) = (1-2**-4)*RAPP(k-1) + 2**-3 , if ry < 3 or
*;         if |RDMS-RDML| > 2**-3 * RDML
*; else
*;
*; RAPP(k) = (1-2**-4)*RAPP(k-1)
*;
RFILTC ZALH    RAPP
     SUB  RAPP,12
     SACH RAPP
     LAC  RY
     SUB  THREE,9
     BLZ  RADD18
     ZALH RDMS       ; RDMS (Q25)
     SUB  RDML,14         ; RDMS-RDML
     ABS
     SUB  RDML,11
     BLZ  RAPRED
RADD18 LAC     RAPP
     ADD  ONE,5
     SACL RAPP       ; 1/8 in Q8
*;
*;************************************************
*; ADAPTIVE PREDICTOR: compute signal estimate
*;
*;   input:    RDQ  quantized difference
*;   output: RSE    signal estimate
*;************************************************
*;
*; compute reconstructed signal:  DQ + SE
*;
RAPRED    LAC  RDQ
     ADD  RSE
     SACL RSR
*;
*; compute coefficients of 6th order predictor
*;
*; RBi(k) = (1-2**-8)*RBi(k-1)
*;         + 2**-7 * SGN[RDQ(k)] * SGN[RDQ(k-1)]
*;
*;      for i = 1...6
*;      and RBi is implicitly limited to +/- 2
*;
RGETB6    LT   RSDQ6
     LAC  RB6,15     ; Q29
     SUB  RB6,7      ; RB6 * 2**-7  (Q29)
     MPY  RSDQ ; SCN(RSDQ)*SGN(RSDQ6) * 2**-7  (Q29)
     LTD  RSDQ5
     SACH RB6,1      ; 1.Q14
RGETB5    LAC  RB5,15
     SUB  RB5,7
     MPY  RSDQ
     LTD  RSDQ4
```

```
      SACH RB5,1
RGETB4     LAC  RB4,15
      SUB  RB4,7
      MPY  RSDQ
      LTD  RSDQ3
      SACH RB4,1
RGETB3     LAC  RB3,15
      SUB  RB3,7
      MPY  RSDQ
      LTD  RSDQ2
      SACH RB3,1
RGETB2     LAC  RB2,15
      SUB  RB2,7
      MPY  RSDQ
      LTD  RSDQ1
      SACH RB2,1
RGETB1     LAC  RB1,15
      SUB  RB1,7
      MPY  RSDQ
      LTD  RSDQ
      SACH RB1,1
*;
*; Update coefficients of 2nd order predictor
*;
*; First get sign of sum of quantized diff and
*; partial sig estimate
*;
RADDC      DMOV RPK1       ; RPK1==>RPK2
      DMOV RPK0       ; RPK0==>RPK1
      LAC  RSEZ
      ADD  RDQ
      SACH TEMP1
      LAC  TEMP1,10
      ADD  ONE,9
      SACL RPK0
RSUMGT     LT   RPK0
*;
*;  now calculate f[RA1(k-1)]  ==> TEMP3 will get 1/2 F
*;
*;         = 4*RA1        if |RA1|  <= 1/2
*;         = 2*SGN(RA1)   if |RA1|  >  1/2
*;
RGETF      LAC  RA1,1           ; 2*RA1
      SACL TEMP5
      BLZ  RGETF2
RGETF1     SUB  ONE,14
      BLZ  RGETA1
      LAC  ONE,14
      B    RDONEF
RGETF2     ABS
      SUB  ONE,14
      BLZ  RGETA1
```

```
     LAC  MINUS,14
RDONEF    SACL TEMP5
*;
*;  A1(k) = (1-2**-8)*A1(k-1)
*;        + (3*2**-8)*SGN[p(k)]*SGN[p(k-1)]
*;
RGETA1    LAC  RA1,12
     SUB  RA1,4
     MPY  RPK1       ; 3*SGN[rp(k-1)*SGN[rp(k)]
     APAC
     APAC
     APAC
     SACH RA1,4
     PAC        ; save sign of SGN[rp(k-1)]*SGN[rp(k)]
*;
*; RA2(k) = (1-2**-7)*RA2(k-1)
*;         + (2**-7)*{SGN[rp(k)]*SGN[(rp(k-2)]
*;         - f[RA1(k-1)]*SGN[rp(k)]*SGN[rp(k-1)]}
*;
RGETA2    BGEZ RSUBF           ; if sign + --> subtract F
     ZAC              ;  else negate F and subtract
     SUB  TEMP5
     SACL TEMP5
*;
RSUBF     LAC  RA2,12
     SUB  RA2,5
     MPY  RPK2
     APAC
     APAC
     SUB  TEMP5,6
     SACH RA2,4
*;
*; now limit RA2 to +/- .75 and prevent overflow
*;
RLIMC     LAC  RA2
     ABS
     SUB  THREE,12  ; |value| must be < .75
     BLEZ RLIMD
     LAC  RA2
     BGEZ RSATPO
RSATNE  LAC    MINUS3,12
     B    RDONEC
RSATPO    LAC  THREE,12
RDONEC    SACL RA2
*;
*; limit RA1 to +/-  1-2**-4 - RA2
*;
RLIMD   LAC    M15,10
     SUB  RA2
     SACL TEMP1          ; 1-2**-4-RA2
     LAC  RA1
     SACH TEMP2
```

```
     ABS
     SUB  TEMP1
     BLEZ GETSR          ; RA1 <= LIMIT
     LAC  TEMP1
     XOR  TEMP2
     SUB  TEMP2
     SACL RA1
*;
*; now convert linear output sample to mu-law PCM
*;
GETSR     ZAC
     SACL SCRACH
     LAC  RSR,2
     SACL SAMPLE         ; save for output
     BGEZ SAT
*;
*;
*;   Check for sign and taking absolute value
*;
     ABS                 ;Take absolute value of Sample
     SACL SAMPLE         ;Store absolute value in Sample
     LACK 128       ;Negative value handling
     SACL SCRACH
     LAC  SAMPLE
*;
SAT  SUB  T9        ; Sample - T9 to check for saturation
     BLEZ BYPASS         ; Branch to Bypass for non-saturation
     LACK      127
     SACL SAMPLE
     B    STEP      ; Saturation handling complete
*;
BYPASS    LAC  SAMPLE         ; Reset accumulator with absolute
value
     SUB  ONE,2          ;Subtract T1 =4 from Sample
     BLZ  STEP           ;Codec value at sign is the final value
*;
TT5  LAC  SAMPLE         ;Load absolute value back into accum.
     SUB  T5        ;Sample - T5
     BLZ  BOT       ;Bottom half of the mu-law curve
*;
TT7  SUB  ONE,12         ; Sample-T7 = Sample -T5 -(T7-T6) -
     SUB  ONE,11         ; (T6-T5)
     BLZ  TT6       ;Branch to compare with T6
*;
TT8  SUB  ONE,13         ; Sample-T8 = Sample -T7 -(T8-T7)
     BLZ  SEG7      ;7-th Segment identified
*;
     SACL SAMPLE
     LAC  SAMPLE,6
     SACH SAMPLE
     LACK 112       ; Segment = 8
     ADD  SAMPLE
```

```
      SACL SAMPLE
      B    STEP       ;Branch to stepcode computation
*;
SEG7  ADD  ONE,13           ; Sample-T7 = Sample-T8 +(T8-T7)
      SACL SAMPLE
      LAC  SAMPLE,7
      SACH SAMPLE
      LACK 96          ; Segment = 7
      ADD  SAMPLE
      SACL SAMPLE
      B    STEP       ;Branch to step code computation
*;
TT6   ADD  ONE,12           ; Sample - T6 = Sample -T7+(T7-T6)
      BLZ  SEG5       ;5-th segment identified
*;
      SACL SAMPLE
      LAC  SAMPLE,8
      SACH SAMPLE
      LACK 80          ; Segment = 6
      ADD  SAMPLE
      SACL SAMPLE
      B    STEP       ;Branch to step code computation
*;
SEG5  ADD  ONE,11           ; Sample - T5 = Sample -T6 + (T6-T5)
*;
      SACL SAMPLE
      LAC  SAMPLE,9
      SACH SAMPLE
      LACK 64          ; Segment = 5
      ADD  SAMPLE
      SACL SAMPLE
      B    STEP       ;Branch to step code computation
*;-------------------------------------------------------------
*;    Comparisons for upper half of the curve is complete.  The
*;    next set of comparisons is for searching in the lower half
*;    of mu-law curve.
*;-------------------------------------------------------------
*;
BOT   ADD  ONE,10           ; Sample-T3 = Sample -T5 +(T5-T4) +
      ADD  ONE,9            ; (T4-T3)
      BLZ  TT2        ;Branch to compare with T2
*;
TT4   SUB  ONE,9            ;Sample - T4 = Sample-T3 -(T4-T3)
      BLZ  SEG3       ;3-rd Segment identified
*;
      SACL SAMPLE
      LAC  SAMPLE,10
      SACH SAMPLE
      LACK 48          ; Segment = 4
      ADD  SAMPLE
      SACL SAMPLE
      B    STEP       ;Branch to stepcode computation
```

```
*;
SEG3 ADD  ONE,9          ; Sample-T3 = Sample-T4 +(T4-T3)
*;
     SACL SAMPLE
     LAC  SAMPLE,11
     SACH SAMPLE
     LACK 32         ; Segment =3
     ADD  SAMPLE
     SACL SAMPLE
     B    STEP       ;Branch to step code computation
*;
TT2  ADD  ONE,8           ;Sample-T2 = Sample-T3 +(T3-T2)
     BLZ  SEG1       ;First segment identified
*;
     SACL SAMPLE
     LAC  SAMPLE,12
     SACH SAMPLE
     LACK 16         ; Segment = 2
     ADD  SAMPLE
     SACL SAMPLE
     B    STEP       ;Branch to step code computation
*;
SEG1 ADD  ONE,7           ;Sample -T1 = Sample+(T2+4) -
     SUB  ONE,3           ;     (T1 +4) --T1=4,T2=124
*;
     SACL SAMPLE
     LAC  SAMPLE,13
     SACH SAMPLE
     LACK 1          ; Segment = 1. ADD 1 for Seg1 corrn.
     ADD  SAMPLE
     SACL SAMPLE
*;
STEP LAC  SCRACH
     ADD  SAMPLE
        XOR     M255
     SACL SCRACH
*;**************************************************************
INTRET    EINT
     RET
*;
*;
*;**************************************************************
*; SYSTEM INITIALIZATION
*;**************************************************************
*;
RESET     DINT            ; Disable interrupts
     LDPK 0          ; Initialize data page
*;
SETPAC    LARK 0,143
     LARP 0
     ZAC             ; Zero iram
ZRAMA     SACL *,0,0
```

```
      BANZ      ZRAMA
*;
      LACK 1          ; Initialize constant ONE
      SACL ONE
      SUB  ONE,1
      SACL MINUS          ; MINUS
*;
      CALL INIT1          ; initialize CCITT parms
      EINT            ; Enable TMS320 interrupts
WAIT  NOP
      B    WAIT       ; Loop to wait for command
      PAGE
*;
*;*************************************************************
*; INITIALIZE 32K CCITT VARIABLES
*;*************************************************************
*;
INIT1     LT   ONE
      LACK 3          ; 3
      SACL THREE
      SUB  THREE,1        ; -3
      SACL MINUS3
*;
      LACK 15         ; 15
      SACL M15
*;
      LACK 63         ; 63
      SACL M63
*;
      LACK 127        ; 127
      SACL M127
        LACK    255
        SACL    M255
      LAC  ONE,9          ; 544
      ADD  ONE,5
      SACL K544
*;
      LAC  ONE,10         ; 1023
      SUB  ONE
      SACL M1023
*;
      LAC  ONE,12         ; 4096
      ADD  ONE,10         ; + 1024 = 5120  (hi limit for LIMB)
      SACL K5120
      SUB  K544       ; 4576
      SACL K4576
*;
      LAC  ONE,11         ; 2047
      SUB  ONE
      SACL M2047
      ADD  ONE,11         ; 4095
      SACL M4095
```

```
      ADD  ONE,12          ; 8191
      SACL M8191
      ADD  THREE,13  ; 32767
      SACL M32767
*;
      LAC  ONE,6
      SUB  ONE,7
      SACL MFFC0
*;
      LAC  ONE,11          ; constant for lin/codec conv
      SUB  ONE,7
      SUB  ONE,2
      SACL T5
*;
      LAC  ONE,15          ; constant for lin/codec conv
      SUB  ONE,7
      SUB  ONE,3
      SACL T9
*;
      MPYK IQTAB           ; Init inverse quant table addr
      PAC
      SACL INQTAB
*;
      MPYK SHFT       ; Init shift mult table addr
      PAC
      SACL SHIFT
*;
      MPYK WTABLE          ; Init WI lookup table address
      PAC
      SACL WITAB
*;
      MPYK FITABL          ; Init FI lookup table address
      PAC
      SACL FITAB
*;
      MPYK CODADD          ; mu-law coding table
      PAC
      SACL CODEAD
*;
      MPYK ITAB4           ; constants for QUAN table
      PAC
      SACL K2309
      LACK 138
      SACL K138
      LACK 130
      SACL K130
      LACK 79
      SACL K79
      LACK 95
      SACL K95
      LACK 46
      SACL K46
```

```
      LACK 49
      SACL K49
*;
INIT2      LAC  ONE,9          ; 512
      SACL PK0        ; init sign of pk to 1
      SACL PK1
      SACL RPK0
      SACL RPK1
*;
      LAC  K544
      SACL YU         ; initial value
      SACL RYU
      SACL YLH
      SACL RYLH
*;
      ZAC
      SACL YLL        ; initial value
      SACL RYLL
;
      LAC  ONE,11
      SACL SDQ1
      SACL SDQ2
      SACL SDQ3
      SACL SDQ4
      SACL SDQ5
      SACL SDQ6
      SACL RSDQ1
      SACL RSDQ2
      SACL RSDQ3
      SACL RSDQ4
      SACL RSDQ5
      SACL RSDQ6
*;
         RET
      PAGE
*;
*;*********************************************************
*;    ROM                                  *
*;*********************************************************
ROMLOC     BSS  0
*;    CODEC-TO-LINEAR CONVERSION TABLE
CODADD     BSS  0
      DATA 0
      DATA 8/4
      DATA 16/4
      DATA 24/4
      DATA 32/4
      DATA 40/4
      DATA 48/4
      DATA 56/4
      DATA 64/4
      DATA 72/4
```

```
DATA 80/4
DATA 88/4
DATA 96/4
DATA 104/4
DATA 112/4
DATA 120/4
DATA 132/4
DATA 148/4
DATA 164/4
DATA 180/4
DATA 196/4
DATA 212/4
DATA 228/4
DATA 244/4
DATA 260/4
DATA 276/4
DATA 292/4
DATA 308/4
DATA 324/4
DATA 340/4
DATA 356/4
DATA 372/4
DATA 396/4
DATA 428/4
DATA 460/4
DATA 492/4
DATA 524/4
DATA 556/4
DATA 588/4
DATA 620/4
DATA 652/4
DATA 684/4
DATA 716/4
DATA 748/4
DATA 780/4
DATA 812/4
DATA 844/4
DATA 876/4
DATA 924/4
DATA 988/4
DATA 1052/4
DATA 1116/4
DATA 1180/4
DATA 1244/4
DATA 1308/4
DATA 1372/4
DATA 1436/4
DATA 1500/4
DATA 1564/4
DATA 1628/4
DATA 1692/4
DATA 1756/4
```

```
DATA 1820/4
DATA 1884/4
DATA 1980/4
DATA 2108/4
DATA 2236/4
DATA 2364/4
DATA 2492/4
DATA 2620/4
DATA 2748/4
DATA 2876/4
DATA 3004/4
DATA 3132/4
DATA 3260/4
DATA 3388/4
DATA 3516/4
DATA 3644/4
DATA 3772/4
DATA 3900/4
DATA 4092/4
DATA 4348/4
DATA 4604/4
DATA 4860/4
DATA 5116/4
DATA 5372/4
DATA 5628/4
DATA 5884/4
DATA 6140/4
DATA 6396/4
DATA 6652/4
DATA 6908/4
DATA 7164/4
DATA 7420/4
DATA 7676/4
DATA 7932/4
DATA 8316/4
DATA 8828/4
DATA 9340/4
DATA 9852/4
DATA 10364/4
DATA 10876/4
DATA 11388/4
DATA 11900/4
DATA 12412/4
DATA 12924/4
DATA 13436/4
DATA 13948/4
DATA 14460/4
DATA 14972/4
DATA 15484/4
DATA 15996/4
DATA 16764/4
DATA 17788/4
```

```
DATA 18812/4
DATA 19836/4
DATA 20860/4
DATA 21884/4
DATA 22908/4
DATA 23932/4
DATA 24956/4
DATA 25980/4
DATA 27004/4
DATA 28028/4
DATA 29052/4
DATA 30076/4
DATA 31100/4
DATA 32124/4
DATA 0/4
DATA -8/4
DATA -16/4
DATA -24/4
DATA -32/4
DATA -40/4
DATA -48/4
DATA -56/4
DATA -64/4
DATA -72/4
DATA -80/4
DATA -88/4
DATA -96/4
DATA -104/4
DATA -112/4
DATA -120/4
DATA -132/4
DATA -148/4
DATA -164/4
DATA -180/4
DATA -196/4
DATA -212/4
DATA -228/4
DATA -244/4
DATA -260/4
DATA -276/4
DATA -292/4
DATA -308/4
DATA -324/4
DATA -340/4
DATA -356/4
DATA -372/4
DATA -396/4
DATA -428/4
DATA -460/4
DATA -492/4
DATA -524/4
DATA -556/4
```

```
      DATA -588/4
      DATA -620/4
      DATA -652/4
      DATA -684/4
      DATA -716/4
      DATA -748/4
      DATA -780/4
      DATA -812/4
      DATA -844/4
      DATA -876/4
      DATA -924/4
      DATA -988/4
      DATA -1052/4
      DATA -1116/4
      DATA -1180/4
      DATA -1244/4
      DATA -1308/4
      DATA -1372/4
      DATA -1436/4
      DATA -1500/4
      DATA -1564/4
      DATA -1628/4
      DATA -1692/4
      DATA -1756/4
      DATA -1820/4
      DATA -1884/4
      DATA -1980/4
      DATA -2108/4
      DATA -2236/4
      DATA -2364/4
      DATA -2492/4
      DATA -2620/4
      DATA -2748/4
      DATA -2876/4
      DATA -3004/4
      DATA -3132/4
      DATA -3260/4
      DATA -3388/4
      DATA -3516/4
      DATA -3644/4
      DATA -3772/4
      DATA -3900/4
      DATA -4092/4
      DATA -4348/4
      DATA -4604/4
      DATA -4860/4
      DATA -5116/4
      DATA -5372/4
      DATA -5628/4
      DATA -5884/4
      DATA -6140/4
      DATA -6396/4
```

```
      DATA -6652/4
      DATA -6908/4
      DATA -7164/4
      DATA -7420/4
      DATA -7676/4
      DATA -7932/4
      DATA -8316/4
      DATA -8828/4
      DATA -9340/4
      DATA -9852/4
      DATA -10364/4
      DATA -10876/4
      DATA -11388/4
      DATA -11900/4
      DATA -12412/4
      DATA -12924/4
      DATA -13436/4
      DATA -13948/4
      DATA -14460/4
      DATA -14972/4
      DATA -15484/4
      DATA -15996/4
      DATA -16764/4
      DATA -17788/4
      DATA -18812/4
      DATA -19836/4
      DATA -20860/4
      DATA -21884/4
      DATA -22908/4
      DATA -23932/4
      DATA -24956/4
      DATA -25980/4
      DATA -27004/4
      DATA -28028/4
      DATA -29052/4
      DATA -30076/4
      DATA -31100/4
      DATA -32124/4
*; INVERSE QUANTIZING TABLE
IQTAB     BSS  0
      DATA 65401
      DATA 68
      DATA 165
      DATA 232
      DATA 285
      DATA 332
      DATA 377
      DATA 428
*; SHIFT MULT TABLE
SHFT BSS  0
      DATA 1
      DATA 2
```

```
      DATA 4
      DATA 8
      DATA >10
      DATA >20
      DATA >40
      DATA >80
      DATA >100
      DATA >200
      DATA >400
      DATA >800
      DATA >1000
      DATA >2000
      DATA >4000
*; WI TABLE
WTABLE    BSS  0
      DATA 65524
      DATA 4
      DATA 27
      DATA 50
      DATA 98
      DATA 184
      DATA 340
      DATA 1108
*; FI TABLE
FITABL    BSS  0
      BSS  0
      DATA 0
      DATA 0
      DATA 0
      DATA 16
      DATA 16
      DATA 16
      DATA 48
      DATA 112
      PAGE
*;
*;*******************************************************
*;    RAM                              *
*;*******************************************************
RAMLOC    BSS  0
      DORG 0
*;
*;  RAM Location # 000
D     DATA 0    ; difference signal q0
*;
*;  RAM Location # 001
I     DATA 0    ; 32kb output
*;
*;  RAM Location # 002
IM    DATA 0    ; 8 b version of I
*;
*;  RAM Location # 003
```

```
SE    DATA 0    ; signal estimate q0
*;
*;  RAM Location # 004
SEZ   DATA 0    ; partial signal estimate q0
*;
*;  RAM Location # 005
APP   DATA 0    ; unlimited speed control q8
*;
*;  RAM Location # 006
AL    DATA 0    ; limited speed control q6
*;
*;  RAM Location # 007
DMS   DATA 0    ; short term ave of F q9
*;
*;  RAM Location # 008
DML   DATA 0    ; long term ave of F q11
*;
*;  RAM Location # 009
YU    DATA 0    ; fast quant scale factor q9
*;
*;  RAM Location # 010
YLL   DATA 0    ; slow quant scale factor (lo word) q15
*;
*;  RAM Location # 011
YLH   DATA 0    ; slow quant scale factor (hi word)
*;
*;  RAM Location # 012
Y     DATA 0    ; quantizer scale factor
*;
*;  RAM Location # 013
B1    DATA 0    ; 6th order predictor coeff q14
*;
*;  RAM Location # 014
B2    DATA 0    ; 6th order predictor coeff q14
*;
*;  RAM Location # 015
B3    DATA 0    ; 6th order predictor coeff q14
*;
*;  RAM Location # 016
B4    DATA 0    ; 6th order predictor coeff q14
*;
*;  RAM Location # 017
B5    DATA 0    ; 6th order predictor coeff q14
*;
*;  RAM Location # 018
B6    DATA 0    ; 6th order predictor coeff q14
*;
*;  RAM Location # 019
SDQ   DATA 0    ; sign of dq
*;
*;  RAM Location # 020
SDQ1  DATA 0    ; sign of dq(k-1)
```

```
*;
*;  RAM Location # 021
SDQ2  DATA 0     ; sign of dq(k-2)
*;
*;  RAM Location # 022
SDQ3  DATA 0     ; sign of dq(k-3)
*;
*;  RAM Location # 023
SDQ4  DATA 0     ; sign of dq(k-4)
*;
*;  RAM Location # 024
SDQ5  DATA 0     ; sign of dq(k-5)
*;
*;  RAM Location # 025
SDQ6  DATA 0     ; sign of dq(k-6)
*;
*;  RAM Location # 026
DQ    DATA 0     ; quantized diff signal q0
*;
*;  RAM Location # 027
DQ1   DATA 0     ; dq(k-1)
*;
*;  RAM Location # 028
DQ2   DATA 0     ; dq(k-2)
*;
*;  RAM Location # 029
DQ3   DATA 0     ; dq(k-3)
*;
*;  RAM Location # 030
DQ4   DATA 0     ; dq(k-4)
*;
*;  RAM Location # 031
DQ5   DATA 0     ; dq(k-5)
*;
*;  RAM Location # 032
A1    DATA 0     ; 2nd order predictor coeff q14
*;
*;  RAM Location # 033
A2    DATA 0     ; 2nd order predictor coeff q14
*;
*;  RAM Location # 034
SR    DATA 0     ; reconstructed signal q0
*;
*;  RAM Location # 035
SR1   DATA 0     ; sr(k-1)
*;
*;  RAM Location # 036
PK0   DATA 0     ; sign of p(k)
*;
*;  RAM Location # 037
PK1   DATA 0     ; sign of p(k-1)
*;
```

```
*;  RAM Location # 038
PK2  DATA 0     ; sign of p(k-2)
*;
*;  RAM Location # 039
     DATA 0
*;
*;  RAM Location # 040
     DATA 0
*;
*; receiver variables reflect transmitter variables above
*;
*;  RAM Location # 041
RI   DATA 0
*;
*;  RAM Location # 042
RIM  DATA 0
*;
*;  RAM Location # 043
RSE  DATA 0
*;
*;  RAM Location # 044
RSEZ DATA 0
*;
*;  RAM Location # 045
RAPP DATA 0
*;
*;  RAM Location # 046
RAL  DATA 0
*;
*;  RAM Location # 047
RDMS DATA 0
*;
*;  RAM Location # 048
RDML DATA 0
*;
*;  RAM Location # 049
RYU  DATA 0
*;
*;  RAM Location # 050
RYLL DATA 0
*;
*;  RAM Location # 051
RYLH DATA 0
*;
*;  RAM Location # 052
RY   DATA 0
*;
*;  RAM Location # 053
RB1  DATA 0
*;
*;  RAM Location # 054
RB2  DATA 0
```

```
*;
*;  RAM Location # 055
RB3   DATA 0
*;
*;  RAM Location # 056
RB4   DATA 0
*;
*;  RAM Location # 057
RB5   DATA 0
*;
*;  RAM Location # 058
RB6   DATA 0
*;
*;  RAM Location # 059
RSDQ DATA 0
*;
*;  RAM Location # 060
RSDQ1      DATA 0
*;
*;  RAM Location # 061
RSDQ2      DATA 0
*;
*;  RAM Location # 062
RSDQ3      DATA 0
*;
*;  RAM Location # 063
RSDQ4      DATA 0
*;
*;  RAM Location # 064
RSDQ5      DATA 0
*;
*;  RAM Location # 065
RSDQ6      DATA 0
*;
*;  RAM Location # 066
RDQ   DATA 0
*;
*;  RAM Location # 067
RDQ1 DATA 0
*;
*;  RAM Location # 068
RDQ2 DATA 0
*;
*;  RAM Location # 069
RDQ3 DATA 0
*;
*;  RAM Location # 070
RDQ4 DATA 0
*;
*;  RAM Location # 071
RDQ5 DATA 0
*;
```

```
*;  RAM Location # 072
RA1  DATA 0
*;
*;  RAM Location # 073
RA2  DATA 0
*;
*;  RAM Location # 074
RSR  DATA 0
*;
*;  RAM Location # 075
RSR1 DATA 0
*;
*;  RAM Location # 076
RPK0 DATA 0
*;
*;  RAM Location # 077
RPK1 DATA 0
*;
*;  RAM Location # 078
RPK2 DATA 0
*;
*;  RAM Location # 079
TEMP4     DATA 0
*;
*;  RAM Location # 080
TEMP3     DATA 0
*;
*;  RAM Location # 081
MFFC0     DATA 0
*;
*;  RAM Location # 082
K2309     DATA 0    ; 2309
*;
*;  RAM Location # 083
K138 DATA 0    ; 138
*;
*;  RAM Location # 084
K130 DATA 0    ; 130
*;
*;  RAM Location # 085
K79  DATA 0    ; 79
*;
*;  RAM Location # 086
K95  DATA 0    ; 95
*;
*;  RAM Location # 087
K46  DATA 0    ; 46
*;
*;  RAM Location # 088
K49  DATA 0    ; 49
*;
*;  RAM Location # 089
```

```
MINUS3    DATA 0     ; -3
*;
*;  RAM Location # 090
THREE     DATA 0     ; 3
*;
*;  RAM Location # 091
     DATA 0
*;
*;  RAM Location # 092
     DATA 0
*;
*;  RAM Location # 093
     DATA 0
*;
*;  RAM Location # 094
     DATA 0
*;
*;  RAM Location # 095
     DATA 0
*;
*;  RAM Location # 096
TEMP1     DATA 0     ;    Temporary storage location
*;
*;  RAM Location # 097
TEMP2     DATA 0     ;    Temporary storage location
*;
*;  RAM Location # 098
TEMP5     DATA 0     ;    Temporary storage location
*;
*;  RAM Location # 099
     DATA 0
*;
*;  RAM Location # 100
     DATA 0
*;
*;  RAM Location # 101
     DATA 0
*;
*;  RAM Location # 102
SHIFT     DATA 0     ; address of shift table
*;
*;  RAM Location # 103
K4576     DATA 0     ; 4576
*;
*;  RAM Location # 104
FITAB     DATA 0     ; address of F table
*;
*;  RAM Location # 105
WITAB     DATA 0     ; address of W table
*;
*;  RAM Location # 106
INQTAB    DATA 0     ; address of inverse quant table
```

```
*;
*;  RAM Location # 107
K544 DATA 0    ; 544
*;
*;  RAM Location # 108
K5120     DATA 0    ; 5120
*;
*;  RAM Location # 109
M15  DATA 0    ; 15
*;
*;  RAM Location # 110
M63  DATA 0    ; 63
*;
*;  RAM Location # 111
M127 DATA 0    ; 127
*;
*;  RAM Location # 112
M1023     DATA 0    ; 1023
*;
*;  RAM Location # 113
M2047     DATA 0    ; 2047
*;
*;  RAM Location # 114
M4095     DATA 0    ; 4095
*;
*;  RAM Location # 115
M8191     DATA 0    ; 8191
*;
*;  RAM Location # 116
M16383    DATA 0    ; 16383
*;
*;  RAM Location # 117
M32767    DATA 0    ; 32767
*;
*;  RAM Location # 118
MINUS     DATA 0    ; -1
*;
*;  RAM Location # 119
ONE  DATA 0    ; 1
*;
*;  RAM Location # 120
M255 DATA 0
*;
*;  RAM Location # 121
     DATA 0
*;
*;  RAM Location # 122
     DATA 0
*;
*;  RAM Location # 123
SCRACH    DATA 0    ; scratch location
*;
```

```
*;  RAM Location # 124
T9    DATA 0     ; Linear to codec break point
*;
*;  RAM Location # 125
T5    DATA 0     ; Linear to codec break point
*;
*;  RAM Location # 126
SAMPLE    DATA 0     ; AD/DA sample
*;
*;  RAM Location # 127
CODEAD    DATA 0     ; Address of codec de-coding table
*;
*;
      DORG 0
*; INTERRUPT PARAMETER AREA
*;
*;  RAM Location # 128      PAGE 1
      DATA 0
*;
*;  RAM Location # 129      PAGE 1
      DATA 0
*;
*;  RAM Location # 130      PAGE 1
      DATA 0
*;
*;  RAM Location # 131      PAGE 1
      DATA 0
*;
*;  RAM Location # 132      PAGE 1
      DATA 0
*;
*;  RAM Location # 133      PAGE 1
      DATA 0
*;
*;  RAM Location # 134      PAGE 1
      DATA 0
*;
*;  RAM Location # 135      PAGE 1
      DATA 0
*;
*;  RAM Location # 136      PAGE 1
      DATA 0
*;
*;  RAM Location # 137      PAGE 1
*;
*;  RAM Location # 138      PAGE 1
      DATA 0
*;
*;  RAM Location # 139      PAGE 1
      DATA 0
*;
*;  RAM Location # 140      PAGE 1
```

```
     DATA 0
*;
*;  RAM Location # 141      PAGE 1
     DATA 0
*;
*;  RAM Location # 142      PAGE 1
     DATA 0
*;
*;  RAM Location # 143      PAGE 1
     DATA 0
```

APPENDIX 3

ADPCM SYSTEM USING TMS32020

Hardware

The circuit in Appendix 1 is reused with some small modifications. Because the TMS32010 is manufactured as a 44-pin chip, while the TMS32020 chip has 68 pins, an adaptor has to be built, mounted in the 44-pin socket so that the TMS32020 can be inserted in the same circuit board. Through the adaptor, the pins are connected as follows :

TMS32010	TMS32020
$D_0 - D_{15}$	$D_0 - D_{15}$
$A_0 - A_2$	$A_0 - A_2$
A_{11}	$\overline{IS}$
$\overline{RS}$	$\overline{RS}$
$\overline{INT}$	$\overline{INT_0}$
$\overline{BIO}$	BIO
$\overline{WE}$	$R/\overline{W}$
$\overline{DEN}$	$R/\overline{W}inverted$
$CLKIN$	$CLKIN$
$CLKOUT$	$CLKOUT$
V_{cc}	V_{cc}
V_{ss}	V_{ss}

Moreover, pins B8, A7, F2 of the TMS32020 are connected to Vcc.

Since the TMS32020 instruction cycle timing depend on program memory and I/O memory wait states when the instructions are executed from external program memory, a J-K flip-flop with Clear is to receive the original CLKIN

signal and generate the new clock to the whole system so that timing is not a problem.

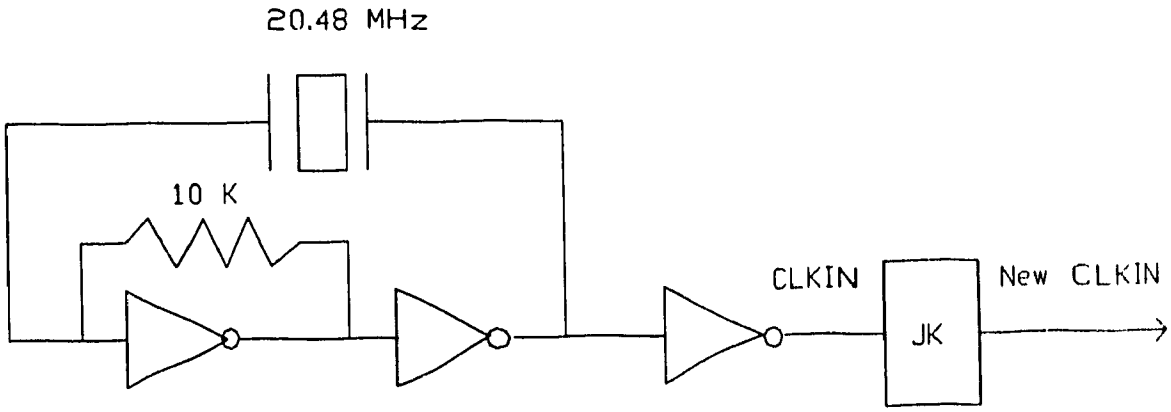


Fig.55. Clock circuit for TMS32020.

In the experiments, the programs are assembled using the Macro Assembler and then simulated on the Simulator. After the execution has been verified good, the executable object code is downloaded to the XDS/22 Emulator. As a result, the two PROM's TBP38L165 which are not needed any more are removed from the board.

Software

As the memory map of the TMS32020 is programmable and the interrupt structure is completely different from that of the TMS32010, the original

program written for the TMS32010 has to be upgraded before it can run on the TMS32020 and provide similar output. The changes are as described in Appendix D : TMS32010/TMS32020 System Migrartion of the TMS32020 User's Guide (Texas Instruments Incorporated, 1985).