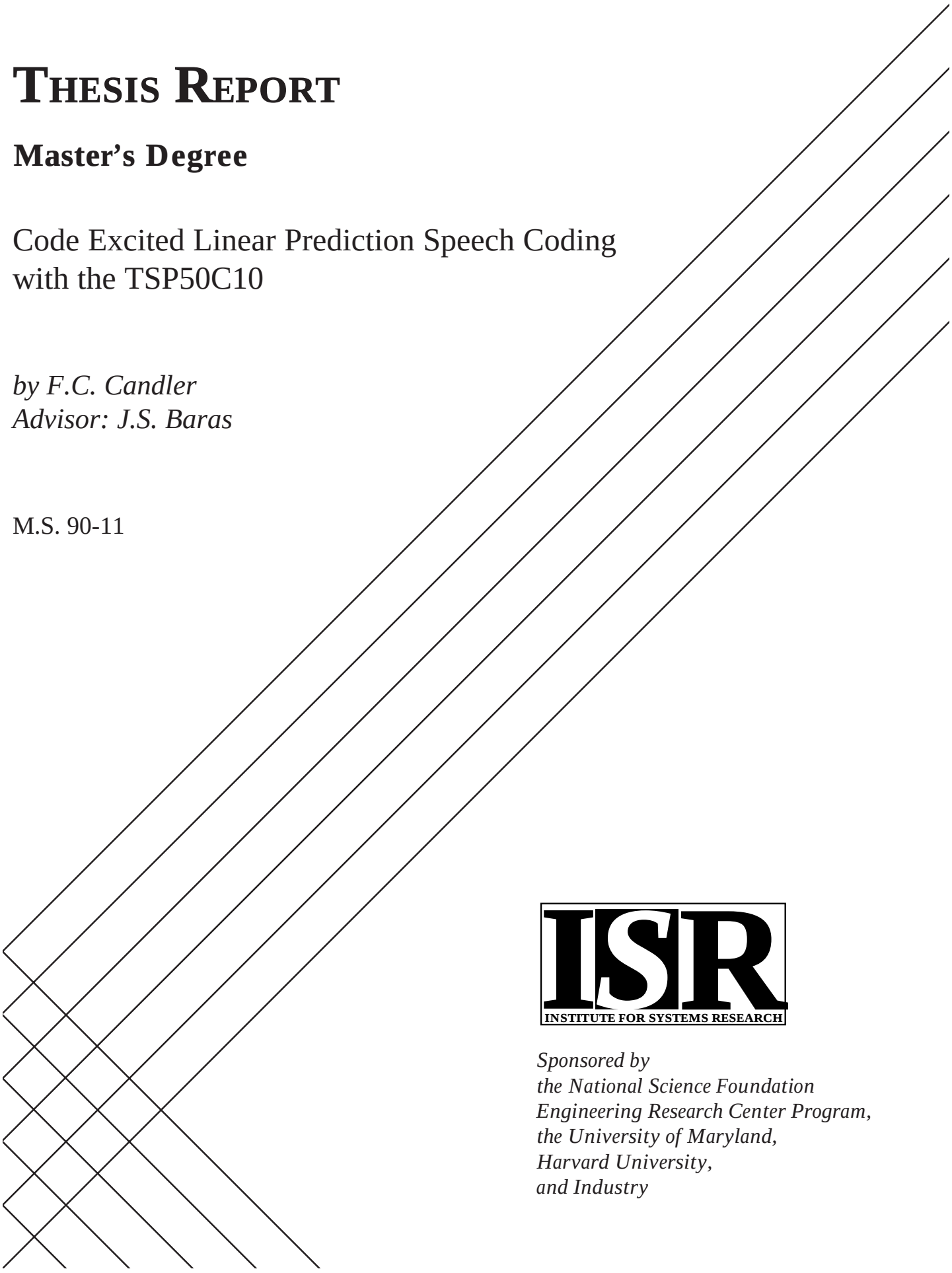
THESIS REPORT

Master's Degree

Code Excited Linear Prediction Speech Coding with the TSP50C10

by F.C. Candler
Advisor: J.S. Baras

M.S. 90-11



Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Code Excited Linear Prediction
Speech Coding with the TSP50C10

by

Frederick Christopher Candler

Thesis submitted to the Faculty of The Graduate School
of The University Of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1990

Advisory Committee:

Professor John S. Baras, Advisor
Professor Steven A. Tretter
Professor Thomas E. Fuja

ABSTRACT

Title of Thesis: Code Excited Linear Prediction
Speech Coding with the TSP50C10

Name of Degree Candidate: Frederick Christopher Candler

Degree and Year: Master of Science, 1990

Thesis directed by: John S. Baras
Professor
Electrical Engineering Department
&
Systems Research Center

Code Excited Linear Predictive Speech Coding and the TSP50C10 single-chip speech synthesizer are presented. An analysis and simulation are performed to determine the speech quality possible with the TSP50C10 and how the architecture can be modified to improve the speech quality.

Contents

List of Figures

Chapter 1

Introduction

Code Excited Linear Prediction (CELP) coding is a speech data compression technique which, at 4.8 kbps, can achieve speech quality comparable to other 32 kbps speech coding techniques.

The CELP algorithm is composed of two steps. The first step, analysis, is performed at the transmitter. The CELP parameters from the analysis are then transmitted over a communications channel. At the receiver the second step, synthesis, is performed to regenerate the speech. Figure 1.1 illustrates the CELP encoding, transmitting, and decoding.

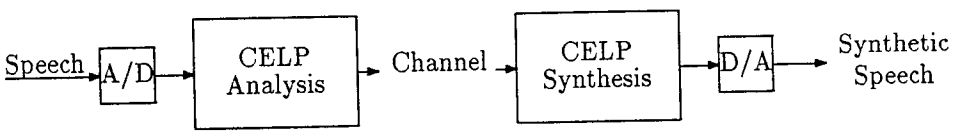


Figure 1.1: CELP Encoding, Transmission, and Decoding.

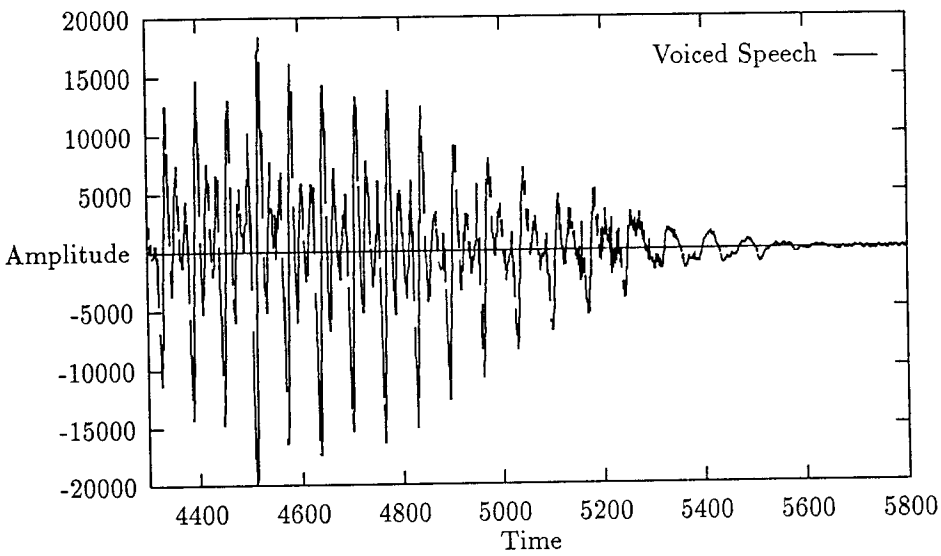


Figure 1.2: A voiced speech waveform.

1.1 Introduction to Speech Coding

In this section, a brief description of voice waveforms is given. Figures 1.2 and 1.3 show voiced and unvoiced signals respectively.

We can see that the first speech waveform (voiced) is almost periodic and is thus called quasi-periodic. Nasal and vowel sounds are voiced. An example of a voiced sound is the "long e" in *even*. The unvoiced signals come from sounds such as /f/ or /s/ in *fix* and *Sunday*. By breaking sounds into voiced and unvoiced categories, we arrive at an early model for speech synthesis known as the VOice CODER (VOCODER). Figure 1.4 shows the VOCODER speech synthesis technique.

The VOCODER was one of the early speech coding techniques which

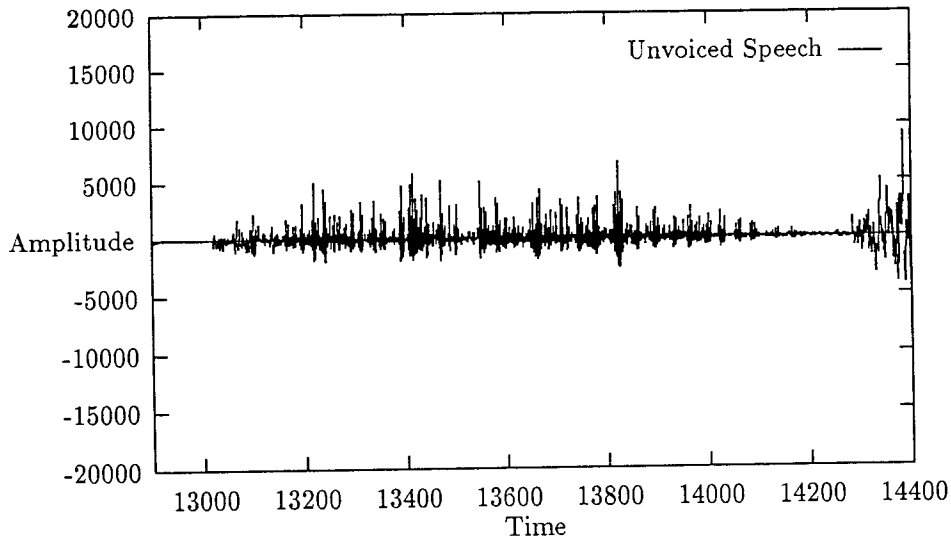


Figure 1.3: An unvoiced speech waveform.

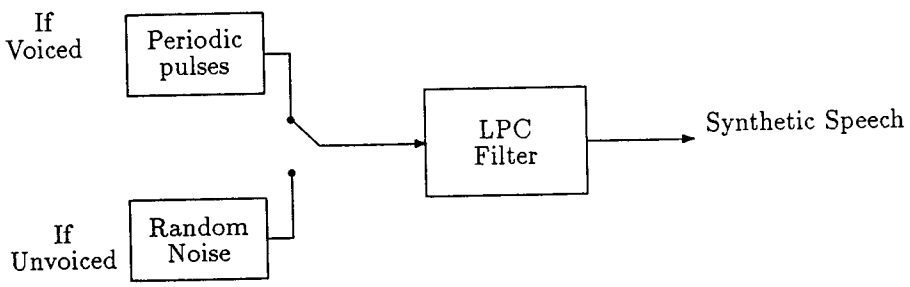


Figure 1.4: Block diagram of a VOCODER.

performs surprisingly well because the LPC filter reproduces the spectral envelope. The spectral envelope has power peaks at the formant frequencies where the speech wave resonates in the vocal tract. The idea in the VOCODER is to develop a linear predictive filter which models the vocal tract. The filter is then excited by either periodic pulses or random noise for voiced and unvoiced sounds respectively. The VOCODER works well for bit rates from 2kbps to 5kbps; however, the speech quality does not improve much at bit rates above 5kbps[Atal2]. The LPC filter reproduces the spectral envelope but does not reproduce the *spectral fine structure.* The spectral fine structure is the local peaks and valleys of the speech power spectrum which are smoothed out in the spectral envelope[Flanagan].

CELP makes two improvements to the VOCODER model. The CELP algorithm attempts to capture the spectral fine structure by extracting the pitch information using a pitch predictor. Secondly, the CELP algorithm excites the LPC filter with a noise-like vector derived from a residual created from the actual speech waveform.

CELP speech coding can achieve high quality speech at 4.8 kbps. Also by increasing the bit rate, we can achieve higher quality speech as opposed to the VOCODER whose quality is limited by the model.

The basic block diagram for CELP speech synthesis is shown in figure 1.5. Recall that the noise-like vector in figure 1.5 is a vector which approximates a speech residual created from the original speech (as discussed in the section on the stochastic code book search).

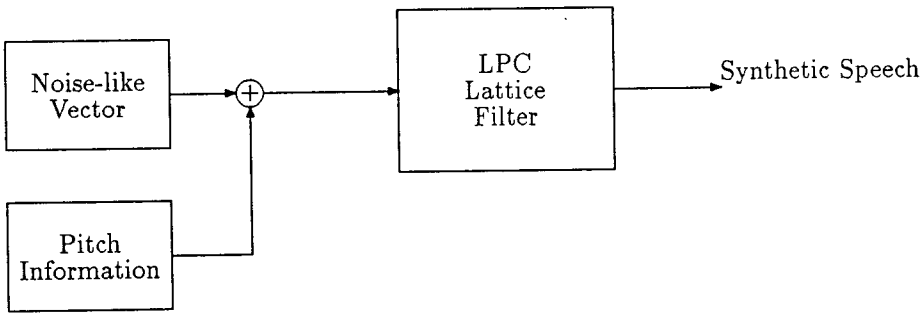


Figure 1.5: Simplified block diagram of CELP synthesis.

1.2 Objective

Very high quality CELP speech is possible using a high-end floating point DSP chip [Campbell2]. The objective herein is to analyze the speech quality that can be generated by the TSP50C10. The TSP50C10 is an inexpensive integer DSP chip produced by Texas Instruments.

The motivation for this research comes from inadequacies in the current low bit rate techniques for speech synthesis. Using the LPC VOCODER model, "speech editing" must be performed to optimize the VOCODER parameters, specifically whether the segment is voiced or unvoiced. Typically, the voiced or unvoiced nature of speech must be determined by a listener. Therefore, the analysis process within the VOCODER model is very time consuming and thus very expensive. CELP, on the other hand, does not rely on the voiced or unvoiced property which means that there is no need to "speech edit" the parameters. Hence, CELP is a much more attractive technique for speech data compression.

In this application, the CELP parameters will be stored in ROM (Read Only Memory) and read sequentially to produce the speech. As a result,

real-time analysis is not an issue while real-time synthesis is.

Therefore, for our purposes, we will concentrate our discussion on CELP synthesis; however, for completeness, the CELP analysis is also presented in the next chapter. This thesis is an example of the interplay between hardware architecture and algorithmic complexity and performance. We shall see how our analysis indicates the necessary modifications for the TSP50C10 in order to implement acceptable CELP synthesis.

Chapter 2

CELP-The Algorithm

As mentioned previously, the CELP algorithm consists of an analysis routine and a synthesis routine. The analysis routine is broken into LPC analysis, perceptual weighting, and code book searches. These are described below, followed by a description of how they are used as blocks in the CELP algorithm.

2.1 CELP Analysis-The Ingredients

2.1.1 LPC Analysis

Linear Predictive Coding is a technique used to model the human vocal tract as a filter. The filter is an all-pole filter with a transfer function:

$$H(z) = \frac{1}{\sum_{i=0}^{p} \alpha_i z^{-p}} \tag{2.1}$$

where $p = 10$ for LPC-10 analysis. LPC analysis is the process by which the LPC coefficients $\{\alpha_i\}$ are determined from the digitized speech.

As the name implies, linear prediction analysis uses the last p samples of speech to make a linear estimate of the next speech value. Therefore we

can write our estimate $\hat{s}(n)$ as

$$\hat{s}(n) = \alpha_1 s(n-1) + \alpha_2 s(n-2) + ... + \alpha_p s(n-p) \tag{2.2}$$

$$\hat{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k) \tag{2.3}$$

where $p = 10$ for LPC-10 analysis. To calculate $\{\alpha_i\}$, we choose to minimize the mean-squared prediction error defined as

$$E_n \equiv \sum_{\text{all } m} e(n+m)^2 \tag{2.4}$$

where

$$e(n+m) \equiv s(n+m) - \hat{s}(n+m). \tag{2.5}$$

Note that $\hat{s}(n+m)$ is the prediction m units ahead of the current sample. Ideally, we would like to minimize E_n for all values of m. However, this is obviously not possible. Therefore we minimize E_n for a specific number m. Thus E_n is called the *short-time average prediction error.* Equation 2.4 becomes

$$E_n = \sum_{\text{finite } m} (s(n+m) - \hat{s}(n+m))^2. \tag{2.6}$$

By substituting equation (2.3) into (2.6) we get

$$E_n = \sum_{m} \left(s(n+m) - \sum_{k=1}^{p} \alpha_k s(n+m-k) \right)^2. \tag{2.7}$$

To minimize (2.7), with respect to the coefficients $\{\alpha_i\}$, we set $\frac{\partial E_n}{\partial \alpha_i} = 0$ for $i = 1, 2, ...p$.

$$\sum_{m} \left(\left[s(n+m) - \sum_{k=1}^{p} \alpha_k s(n+m-k) \right] \cdot 2s(n+m-i) \right) = 0. \tag{2.8}$$

$$\Rightarrow \sum_m s(n+m)s(n+m-i) = \sum_m \left(\sum_{k=1}^{p} \alpha_k s(n+m-k)s(n+m-i) \right). \quad (2.9)$$

$$\Rightarrow \sum_m s(n+m)s(n+m-i) = \sum_{k=1}^{p} \alpha_k \sum_m s(n+m-k)s(n+m-i). \quad (2.10)$$

By defining the short-time autocorrelation function

$$R_n(i,k) = \sum_m s(n+m-i)s(n+m-k) \quad (2.11)$$

we can rewrite (2.10) as

$$R_n(i,0) = \sum_{k=1}^{p} \alpha_k R_n(i,k) \qquad \text{for } i = 1,2,...p. \quad (2.12)$$

The short-time autocorrelation function $R_n(i,k)$ is summed over a finite number m. Justification for equation (2.11) is given below.

As an approximation, we multiply $s(m+n)$ by a Hamming window to make the signal $s(m+n) = 0$ for m outside the window, i.e. for m not in $0 \leq m \leq N-1$. Thus after multiplying our digitized speech by a finite length data window, we can write

$$E_n = \sum_{m=0}^{N-1+p} e^2(n+m). \quad (2.13)$$

Due to the data window, we know that $s(n+m) = 0$ for m outside of $0 \leq m \leq N-1$, i.e.

$$s(n+m-i) = 0 \qquad \text{for } i > m;\ m > N-1+i \quad (2.14)$$

and

$$s(n+m-k)=0 \qquad \text{for } k>m;\ m>N-1+k \tag{2.15}$$

We can rewrite equation (2.11) as

$$R_n(i,k)=\sum_{m=0}^{N-1+p} s(n+m-i)s(n+m-k) \tag{2.16}$$

for $m \geq i$, $m \geq k$, $i=1,2,..p$, and $k=0,1,...p$. By letting $v=m-i$ we get

$$R_n(i,k)=\sum_{v=-i}^{N-1+p-i} s(n+v)s(n+v+i-k) \tag{2.17}$$

but

$$s(n+v)=0 \qquad \text{for } v<0 \tag{2.18}$$

and

$$s(n+v+i-k)=0 \qquad \text{for } v+i-k>N-1. \tag{2.19}$$

Therefore, we can write

$$R_n(i,k)=\sum_{v=0}^{N-1-(i-k)} s(n+v)s(n+v+i-k). \tag{2.20}$$

By setting $\tau=i-k$ we see that (2.20) is the short-time autocorrelation function defined as

$$R_n(\tau)=\sum_{j=0}^{N-1-\tau} s(n+j)s(n+j+\tau). \tag{2.21}$$

This justifies our previous approximation.

By noting that $R_n(\tau)$ is an even function, we can rewrite (2.12) as

$$R_n(i,0)=\sum_{k=1}^{p} \alpha_k R_n(\mid \tau \mid) \tag{2.22}$$

where $\tau = i - k$, $i = 1, 2...p$. Equation (2.22) can be solved efficiently for the linear predictive coefficients, $\{\alpha_i\}$, by using Durbin's algorithm.

In matrix form, equation (2.22) is written as:

$$\begin{bmatrix} R(0) & R(1) & R(2) & \cdots & R(p-1) \\ R(1) & R(0) & R(1) & \cdots & R(p-2) \\ R(2) & R(1) & R(0) & \cdots & R(p-3) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R(p-1) & R(p-2) & R(p-3) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ R(3) \\ \vdots \\ R(p) \end{bmatrix}$$

By noting that the above matrix is symmetric and Toeplitz, we can apply Durbin's algorithm to solve for $\{\alpha_i\}$ [Makhoul].

Durbin's algorithm is a recursive procedure which solves for $\{\alpha_i\}$ in $p^2 + O(p)$ operations [Makhoul]. Durbin's algorithm is stated by the equations given below [Rab&Sch].

$$E^{(0)} = R(0) \tag{2.23}$$

$$k_i = \frac{R(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} R(i-j)}{E^{(i-1)}} \qquad \text{for } 1 \le i \le p \tag{2.24}$$

$$\alpha_i^{(i)} = k_i \tag{2.25}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \qquad \text{for } 1 \le j \le i-1 \tag{2.26}$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \tag{2.27}$$

$$\tag{2.28}$$

Note that the superscripts denote the iteration of the recursion. Also, the values of $\{\alpha_i\}$ are set to the values of $\{\alpha_i^{(i)}\}$ at the last iteration (i.e, when $i = p$).

As an example of Durbin's algorithm, a 2^{nd} order predictor is considered. The initialization is given by (2.29).

$$E^{(0)} = R(0) \tag{2.29}$$

The recursion starts with $i = 1$. Hence we get equations (2.30 - 2.33).

$$k_1 = \frac{R(1) - 0}{E^{(0)}} \tag{2.30}$$

which reduces to

$$k_1 = \frac{R(1)}{R(0)} \tag{2.31}$$

$$\alpha_1^{(1)} = \frac{R(1)}{R(0)}. \tag{2.32}$$

$$E^{(1)} = (1 - k_1^2)E^{(0)} \tag{2.33}$$

which simplifies to

$$E^{(1)} = \frac{R^2(0) - R^2(1)}{R(0)}. \tag{2.34}$$

The recursion, for an order 2 predictor, finishes with $i = 2$ and $j = 1, 2$.

$$k_2 = \frac{R(2) - \alpha_1^{(1)} R(1)}{E^{(1)}} \tag{2.35}$$

which works out to

$$k_2 = \frac{R(2)R(0) - R^2(1)}{R^2(0) - R^2(1)}. \tag{2.36}$$

$$\alpha_2^{(2)} = k_2 \tag{2.37}$$

$$\alpha_1^{(2)} = \alpha_1^{(1)} - k_2\alpha_1^{(1)} \tag{2.38}$$

which yields

$$\alpha_1^{(2)} = \frac{R(1)R(0) - R(2)R(1)}{R^2(0) - R^2(1)}. \tag{2.39}$$

The actual values of the linear predictive coefficients (LPC's) are assigned during the last iteration. In the above example, the LPC's are assigned as in equations (2.40) and (2.41).

$$\alpha_1 = \alpha_1^{(2)} \tag{2.40}$$

$$\alpha_2 = \alpha_2^{(2)}. \tag{2.41}$$

The quantity $E^{(i)}$ in equation (2.27) is the predictor error at stage i [Rab&Sch]. It can be shown that the error $E^{(i)}$ decreases or stays the same as i increases [Makhoul]. Therefore, from equation (2.27) we are guaranteed stability if the magnitude of each k_i is bounded by 1. The parameters $\{k_i\}$ are called PARtial CORrelation (PARCOR) coefficients. The name PARCOR is derived from the fact that they represent the correlation between the forward and backward prediction error in the LPC lattice filter [Tretter]. PARCOR coefficients are also known as *reflection coefficients* because they represent the reflection coefficients in an acoustical tube. By denoting the cross-sectional area of an acoustical tube by A_m we can define the PARCOR coefficients as [Mar&Gra].

$$k_i = \frac{A_{m-1} - A_m}{A_{m-1} + A_m}. \tag{2.42}$$

2.1.2 Perceptual Weighting

Speech has certain frequencies, called formants, which have more power than most frequencies. The idea behind perceptual weighting is to allow the error

vector to have more power at the formant frequencies. The added power of the noise vector will not be noticed at the formant frequencies because it will be masked by the power of the signal.

The transfer function of the perceptual weighting filter is given in equation (2.43)

$$H(z) = \frac{1 + \sum_{k=1}^{p} \alpha_k z^{-k}}{1 + \sum_{k=1}^{p} \gamma^k \alpha_k z^{-k}} \tag{2.43}$$

where $\{\alpha_k\}$ are the linear predictive coefficients and γ is a constant. If $\gamma = 1$ then H(z) is an all pass filter. For perceptual weighting, γ is usually taken between 0.7 and 0.9. A typical value is 0.73.

2.1.3 Stochastic Code Book Search

The purpose of the code book search is to find the optimum gain and corresponding vector from the stochastic code book. The stochastic code book is a time-invariant list of 512 60-dimensional vectors. Note that the dimension of each code book vector is dependent upon the subframe length. Here subframes are 60 consecutive speech samples (7.5 msec). The measure for optimality is the total squared error between the code book vector and the speech residual.

The stochastic code book search is illustrated in figure 2.1 where the filter, denoted by its impulse response f_n is the combined response of the LPC filter $H(z)$ and the perceptual weighting filter $W(z)$, i.e. F(z) = H(z)W(z). Using superscript k to designate code book vector k, we can write the output from the filter as

$$y_i^{(k)} = \sum_{j=-\infty}^{\infty} f_{i-j} v_j^{(k)} \tag{2.44}$$

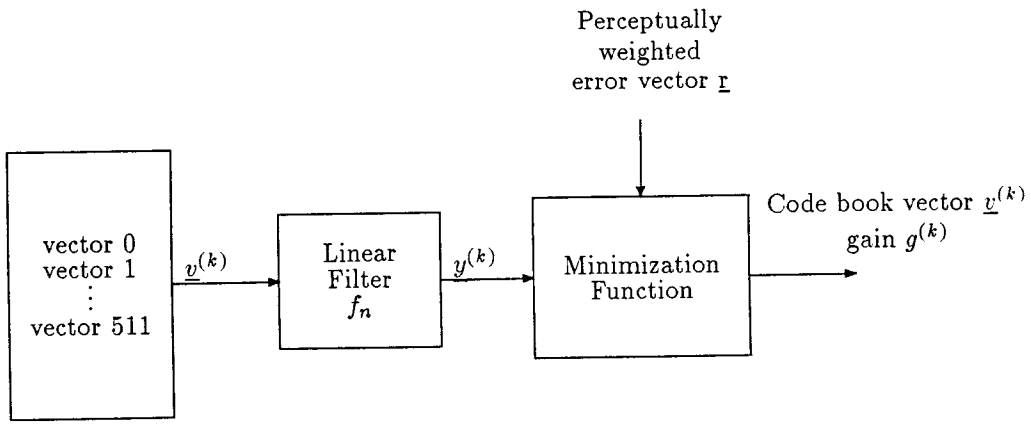


Figure 2.1: Block diagram of the search procedure.

where $\underline{y}^{(k)} = (y_0^{(k)}, y_1^{(k)}, ..., y_i^{(k)}, ..., y_{59}^{(k)})^T$. By using causality of the filter, $f_{i-j} = 0$ for $i < j$ and we can write

$$y_i^{(k)} = \sum_{j=-\infty}^{i} f_{i-j} v_j^{(k)}. \tag{2.45}$$

But $\underline{v}^{(k)} = (v_0, v_1, ..., v_{59})^T$,

$$\Rightarrow y_i^{(k)} = \sum_{j=0}^{i} f_{i-j} v_j^{(k)}. \tag{2.46}$$

Hence, in matrix form,

$$\underline{y}^{(k)} = \mathbf{F}\underline{v}^{(k)} \tag{2.47}$$

where F is an NxN (i.e., 60x60) lower triangular matrix. The matrix elements $f_{i,j}$ are given by f_{i-j}.

Next we derive a relation for determining the optimum code book vector. Physically, we want the code book vector that, once scaled, LPC filtered and perceptually weighted, most resembles the perceptually weighted error

vector. Let $g^{(k)}$ be the gain associated with code book vector $\underline{v}^{(k)}$. From figure 2.3, $\underline{r}$ is the perceptually weighted version of $\underline{e}_0$ which is the residual created by subtracting the linear prediction generated by the sum of the pitch and stochastic code book elements.

Then we can write $E^{(k)}$, the norm-squared of the error vector, as

$$E^{(k)} = \|\underline{r} - g^{(k)}\mathbf{F}\underline{v}^{(k)}\|^2 \tag{2.48}$$

$$E^{(k)} = (\underline{r} - g^{(k)}\mathbf{F}\underline{v}^{(k)})(\underline{r} - g^{(k)}\mathbf{F}\underline{v}^{(k)})^T \tag{2.49}$$

$$E^{(k)} = \underline{r}\underline{r}^T - 2g^{(k)}\underline{r}^T\mathbf{F}\underline{v}^{(k)} + (g^{(k)})^2\mathbf{F}\underline{v}\underline{v}^T\mathbf{F}^T \tag{2.50}$$

To minimize the norm-squared error, we set $\frac{\partial E^{(k)}}{\partial g^{(k)}} = 0$.

$$\Rightarrow -2\underline{r}^T\mathbf{F}\underline{v}^{(k)} + 2g^{(k)}\|\mathbf{F}\underline{v}^{(k)}\|^2 = 0 \tag{2.51}$$

$$\Rightarrow g^{(k)} = \frac{\underline{r}^T\mathbf{F}\underline{v}^{(k)}}{\|\mathbf{F}\underline{v}^{(k)}\|^2}. \tag{2.52}$$

From (2.52) we know the optimum scaling for vector k. By substituting (2.52) in equation (2.50), we get

$$E^{(k)} = \|\underline{r}\|^2 - (g^{(k)})^2\|\mathbf{F}\underline{v}^{(k)}\|^2 \tag{2.53}$$

or equivalently,

$$E^{(k)} = \|\underline{r}\|^2 - \frac{(\underline{r}^T\mathbf{F}\underline{v}^{(k)})^2}{\|\mathbf{F}\underline{v}^{(k)}\|^2}. \tag{2.54}$$

We use (2.54) to determine if we have found the smallest error among all codevectors $k = 0, 1, ...511$. Equation (2.54) is minimized if the second term on the right hand side is maximized. Thus we have a criterion for

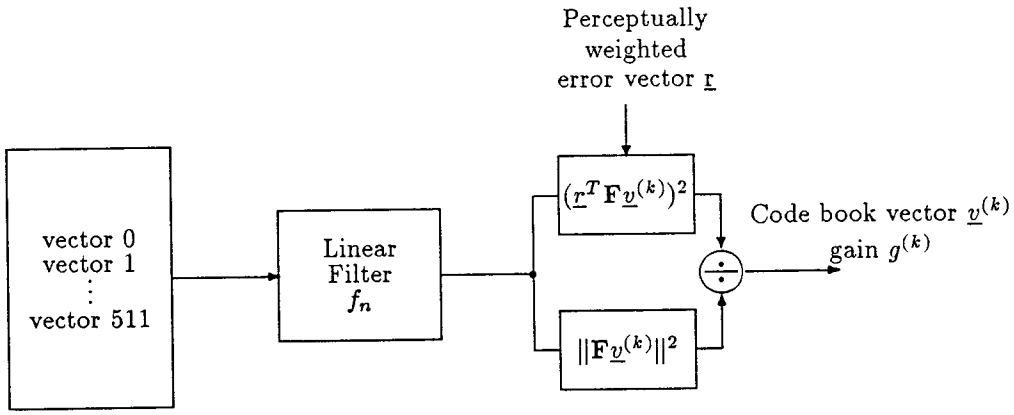


Figure 2.2: Block diagram of the search procedure. The detailed diagram for the analysis is given in figure 2.3.

determining the optimum codevector and the corresponding gain. In block diagram form we get the algorithm as shown in figure 2.2.

We see from figure 2.2 that the upper branch of the minimization function can be written as $(\sum_{i=1}^{N=60} r_i \underline{f}_i \underline{v}^{(k)})^2$, where $\underline{f}_i$ is the i^{th} row of $\mathbf{F}$, which is just the square of the correlation of $\underline{r}$ with $\mathbf{F}\underline{v}^{(k)}$. Similarly, the lower branch is $\sum_{i=1}^{N=60}(\underline{f}_i \underline{v}^{(k)})^2$ which is the energy.

2.1.4 Pitch Code Book Search

The pitch code book search algorithm is exactly the same as the stochastic code book search except for two changes. The first is that the residual $\underline{r}$ changes to the input speech vector minus the perceptually weighted prediction from the LPC lattice filter (given by $\underline{u}$ in figure 2.3). Secondly, the code book changes from a time-invariant stochastic code book to an adaptive pitch code book. The adaptive pitch code book is made up of 128

different pitch lags. By a pitch lag, we mean that, for instance if the lag $l = 25$, the pitch vector corresponding to l would be the speech vector that was produced 25 samples ago. The pitch code book is very intuitive in that the pitch information is the quasi-periodic part of the speech signal. Therefore, if the pitch period is $l = 25$, then we would like to repeat the signal that was generated 25 samples ago. Hence the pitch code book search would choose $l = 25$.

As stated, the pitch code book consists of 128 different pitch lags. The lags chosen in the NSA standard are $l = 20$ to $l = 147$. Also, when implementing the full floating point CELP, the NSA standard specifies that there should also exist another 128 fractional pitch lags, thus making 256 different pitch lags. The fractional pitch lags of course give a finer selection of the actual pitch period.

As a note, the original CELP algorithm suggested by M. Schroeder and B. Atal used a long time pitch predictor as opposed to the NSA adaptive code book. The pitch predictor was specified by

$$P(z) = b_1 z^{-m+1} + b_2 z^{-m} + b_3 z^{-m-1} \tag{2.55}$$

where m is the pitch period. We can see that (2.55) gives an interpolated value of the sample m lags ago. By increasing the order of the pitch predictor we can improve the interpolation.

2.2 CELP Analysis

CELP analysis is the process by which the digitally sampled speech is compressed to 4.8 kbps.

The digital speech is broken into frames and subframes. The NSA standard specifies that a frame is composed of 240 speech samples (30 msec) and a subframe is composed of 60 speech samples (7.5 msec).

The basic idea behind CELP analysis is to extract and code separately the spectral envelope information, the pitch information, and the innovations sequence. The spectral envelope is extracted by using LPC-10 analysis on the incoming digitized speech. The LPC parameters are coded as PARtial CORrelation(PARCOR) coefficients and then packed into the CELP bitstream. As a note, the LPC analysis is performed $\frac{1}{2}$ a frame in advance of the pitch and innovations code book analysis. Also, the physical interpretation of the LPC-10 filter is that it models the human vocal tract. Due to the slowly time varying nature of the vocal tract, the LPC-10 analysis is performed only on a frame-by-frame basis as opposed to every subframe.

The pitch information is extracted from the incoming speech by simply subtracting off the spectral envelope information. The spectral envelope information is contained in the previous LPC-10 parameters due to the fact that the LPC-10 analysis is performed $\frac{1}{2}$ a frame ahead of the pitch analysis. The residual (i.e., the digitized speech minus the spectral envelope information) is perceptually weighted and then compared to all the sequences in the pitch code book (see the section on the pitch search). The pitch search determines the optimum vector from the adaptive pitch code book. The pitch search generates a pitch index into the pitch code book and a pitch gain. These two parameters are the second and third CELP parameters. To achieve higher quality speech, the pitch analysis is performed every sub-

frame.

After extracting the pitch information and the spectral envelope information, there remains a noise-like sequence known as the *innovations sequence.* It represents the information that could not be extracted using either the pitch analysis or spectral envelope analysis. Physically, the innovations sequence is the residual created by subtracting the spectral envelope and the pitch from the incoming digitized speech. The sequence is then perceptually weighted and the result is compared to a time invariant stochastic code book (see the section on the stochastic code book search). The code book search results in a code book index to the optimum codeword and a code book gain. These form the last two CELP parameters.

Therefore the CELP bitstream is made up of:

1. 10 PARCOR coefficients per frame.
2. a pitch index per subframe.
3. a pitch gain per subframe.
4. a code book index per subframe.
5. a code book gain per subframe.

The CELP analysis algorithm is summarized in figure 2.3.

2.3 LPC Filter Parameterization

To achieve better data compression, *Line Spectral Pairs* (LSP's) can be used to parameterize the LPC lattice filter. The transfer function of the

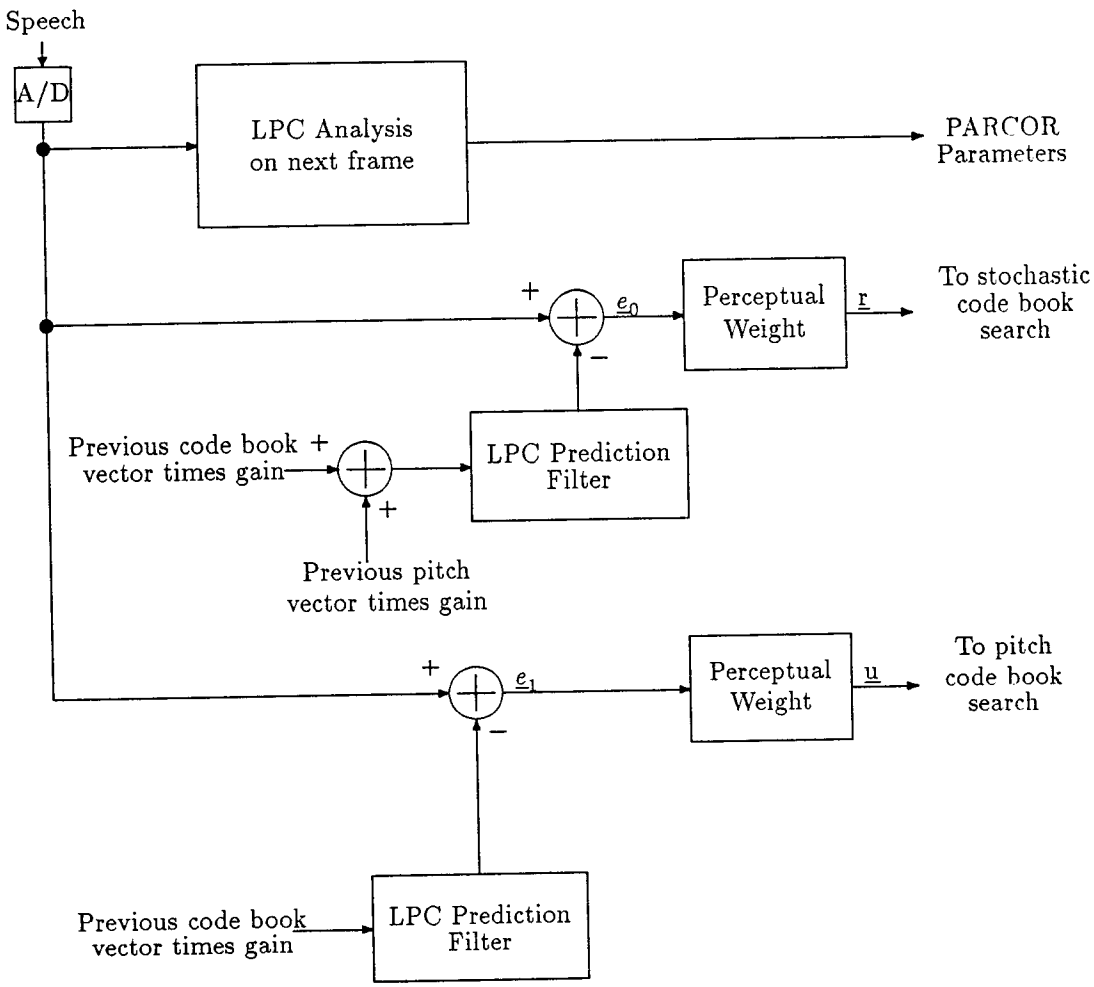


Figure 2.3: The CELP analysis algorithm.

vocal tract filter stays the same whether you are using PARCOR or LSP's. However, LSP's have better properties [Sug&Itak]:

1. The LSP's are ordered and thus lend themselves to better quantization with fewer bits.

2. The LSP's perform well when interpolated between subframes of speech.

3. They have uniform spectral sensitivity and hence we can use uniform bit allocation when quantizing.

2.3.1 From LSP to LPC

The CELP standard specifies that the lattice filter should be parameterized using line spectral pairs. As mentioned previously, the TSP50C10 lattice filter is controlled by PARCOR coefficients. The conversion from LSP's to LPC's is straightforward. LPC parameters are easily changed to PARCOR as shown in the next section. The LPC filter is specified by equation (2.56).

$$H(z^{-1}) = \frac{\sigma}{A_p(z^{-1})} \tag{2.56}$$

where $p = 10$ for LPC-10, σ is the gain, and A_p is given by equation (2.57).

$$A_p = 1 + \sum_{i=1}^{p} \alpha_i z^{-i} \tag{2.57}$$

The α_i in equation (2.57) are the LPC coefficients for the filter. In LSP parameterization, the filter again has the same form as in equation (2.56). Now we introduce two new polynomials $P(z^{-1})$ and $Q(z^{-1})$ as in equation (2.58).

$$A_p(z^{-1}) = \frac{1}{2}(P(z^{-1}) + Q(z^{-1})) \tag{2.58}$$

where

$$P(z^{-1}) = A_p(z^{-1}) - z^{-(p+1)}A_p(z) \tag{2.59}$$

and

$$Q(z^{-1}) = A_p(z^{-1}) + z^{-(p+1)}A_p(z). \tag{2.60}$$

Since $P(z^{-1})$ and $Q(z^{-1})$ both have real coefficients, the roots are complex conjugates and thus the polynomials can be written as in equations (2.61) and (2.62).

$$P(z^{-1}) = (1 - z^{-1}) \prod_{i=1}^{\frac{1}{2}p} (1 - 2cos\omega_i z^{-1} + z^{-2}) \tag{2.61}$$

$$Q(z^{-1}) = (1 + z^{-1}) \prod_{i=1}^{\frac{1}{2}p} (1 - 2cos\theta_i z^{-1} + z^{-2}) \tag{2.62}$$

The parameters ω_i and θ_i are known as *line spectral pairs*. By expanding equation (2.58) where $P(z^{-1})$ and $Q(z^{-1})$ are given by equations (2.61) and (2.62) the LPC parameters α_i may be obtained as functions of the LSP parameters θ_i and ω_i. These functions are very straightforward to derive but are quite lengthy in form.

2.3.2 From LPC to PARCOR

The TSP50C10 uses PARCOR parameters to describe the lattice filter. Therefore, we must convert the LPC parameters to PARCOR parameters.

The LPC and PARCOR parameters are derived from Durbin's algorithm from which the following equations arise:

$$k_i = \alpha_i^{(i)} \tag{2.63}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \tag{2.64}$$

where $i = 1, 2..., p$ and $j = 1, 2, ..., i-1$. By solving equation (2.64) for $\alpha_j^{(i-1)}$ we obtain:

$$\alpha_j^{(i-1)} = \alpha_j^{(i)} + k_i \alpha_{i-j}^{(i-1)} \tag{2.65}$$

Equation 2.65 is in the correct form in that we can find $\alpha_j^{(i-1)}$ in terms of $\alpha_j^{(i)}$ and k_i. However, we also need to know $\alpha_{i-j}^{(i-1)}$ which we don't know because it is from the previous stage (i.e superscript is $i-1$). Therefore, we manipulate the above equations to elliminate the $\alpha_{i-j}^{(i-1)}$ term. By using equation 2.65 with subscript j replaced by $i-j$, we obtain:

$$\alpha_{i-j}^{(i-1)} = \alpha_{i-j}^{(i)} + k_i \alpha_{i-(i-j)}^{(i-1)} \tag{2.66}$$

By substituting equation (2.66) into (2.65) we obtain:

$$\alpha_j^{(i-1)} = \alpha_j^{(i)} + k_i [\alpha_{i-j}^{(i)} + k_i \alpha_j^{(i-1)}] \tag{2.67}$$

solving for $\alpha_j^{(i-1)}$ we get:

$$\alpha_j^{(i-1)} = \frac{\alpha_j^{(i)} + k_i \alpha_{i-j}^{i}}{1 - k_i^2} \tag{2.68}$$

where $i = p, .., 1$, $j = 1, 2, 3, .., i-1$. Thus by using the above equation and (2.63) we are able to solve for successive k-parameters, i.e. starting with k_{10} we are able to find $k_9 ... k_1$.

2.4 CELP Synthesis

CELP synthesizes speech by exciting a lattice filter which approximates the vocal tract. The excitations to the filter are generated at the speech sample rate. Therefore, the objective in CELP synthesis is to process packed speech data to produce an excitation, at an 8KHz rate, and write it to a lattice filter.

The TSP50C10 single-chip speech synthesizer is appropriate for the CELP algorithm in that it incorporates the lattice filter in firmware. Thus CELP synthesis is reduced to generating an excitation at an 8KHz rate(while updating the lattice filter coefficients). To generate the excitation the program executes the following steps:

1. Extract code book element.
2. Multiply element by code book gain.
3. Extract pitch value.
4. Multiply pitch value by pitch gain.
5. Add results 2 & 4.(code book element * gain + pitch * pitch gain)
6. Write result to filter.

These steps are shown in figure 2.4. The figure shows that the following signals are necessary for each excitation:

- Code book index
- Code book gain

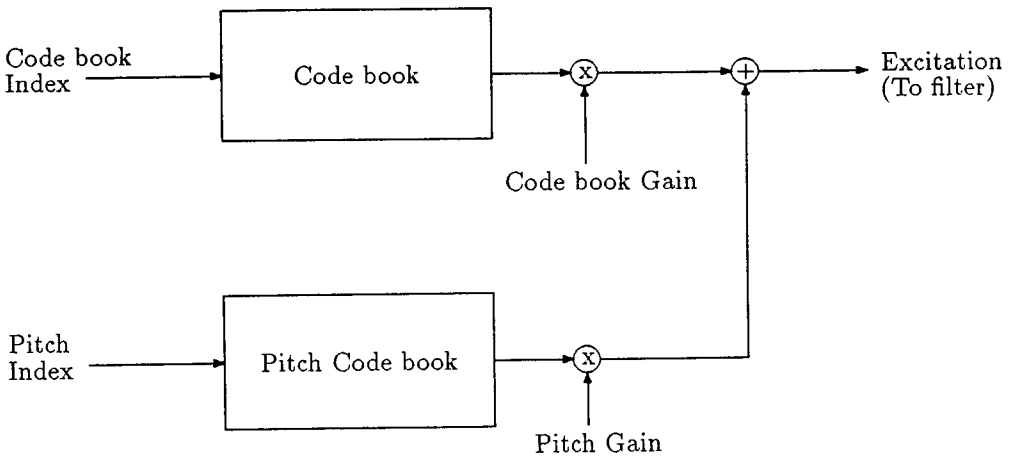


Figure 2.4: Sequence of operations to generate the excitation.

- Pitch index
- Pitch gain

The LPC lattice filter, which is implemented in firmware in the TSP50C10, is shown in figure 2.5.

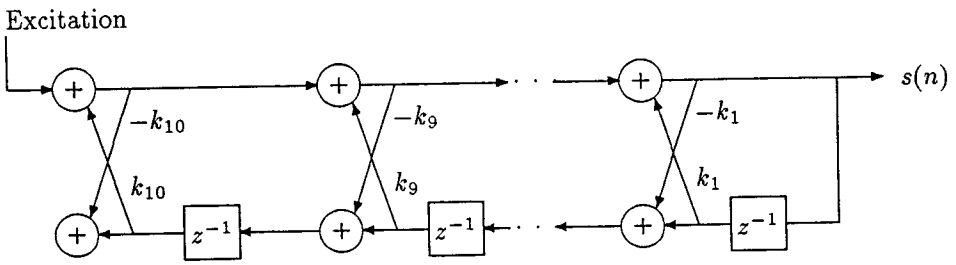


Figure 2.5: LPC synthesis filter using PARCOR coefficients.

Chapter 3

The TSP50C10

The TSP50C10 is a first generation Texas Instruments speech synthesizer. The chip incorporates an internal microprocessor as well as a lattice filter for speech synthesis.

3.1 The Hardware

Figure 3.1 shows a block diagram of the hardware internal to the TSP50C10.

The function of each block is :

ALU (Arithmetic and Logical Unit). Arithmetic and logical operations, specifically:

1. Add
2. Subtract
3. Multiply
4. Shift left and right
5. Logical AND

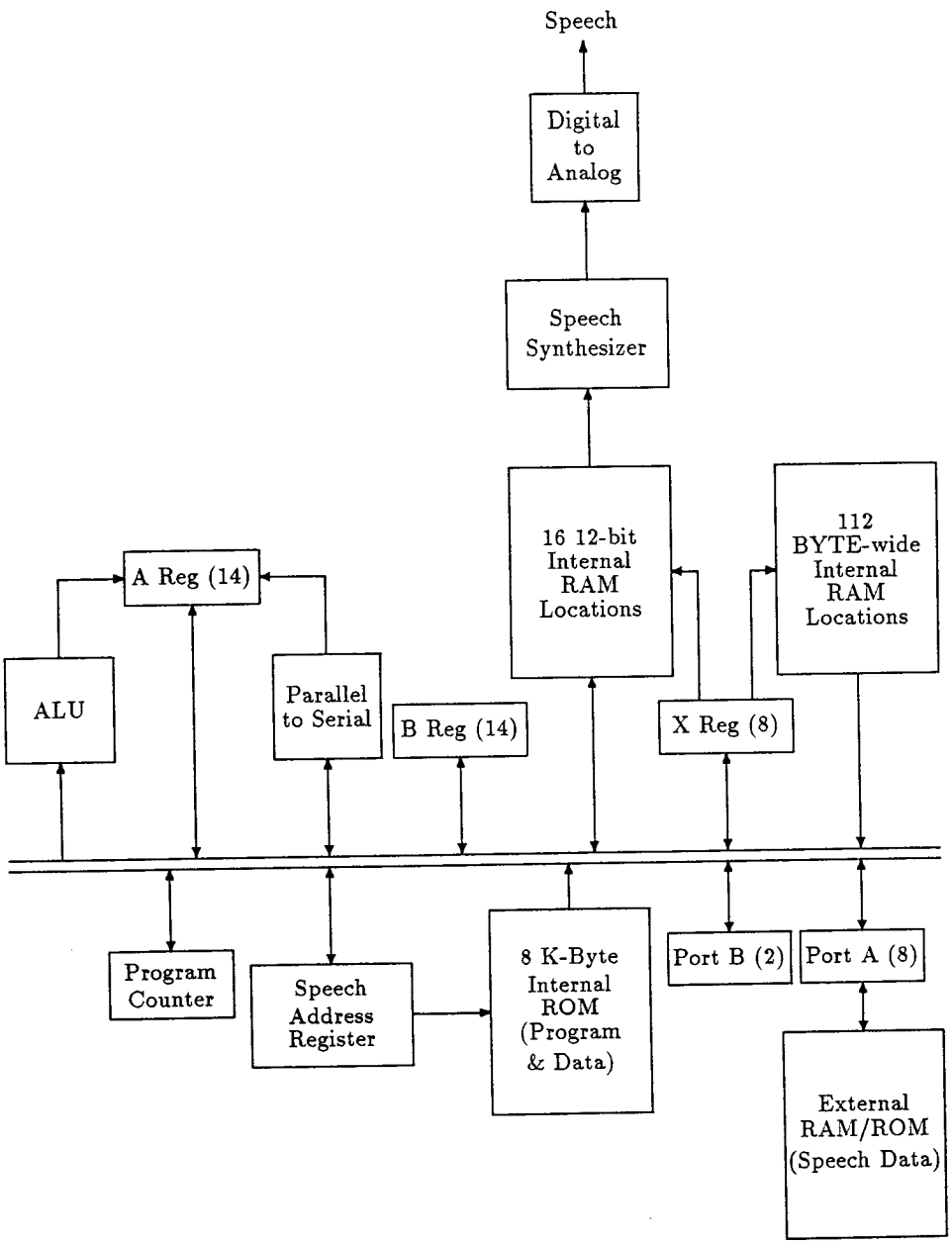


Figure 3.1: Block diagram of the internal hardware of the TSP50C10.

6. Logical OR

A register - the most important storage register for intermediate calculations. Almost all data movement must go through the 14-bit A register.

B register - A 14-bit wide intermediate storage location.

X register - An 8-bit wide register used to index internal RAM or as temporary storage.

Bus - 14-bit wide data bus used to transfer data within the TSP50C10.

Internal RAM - The internal RAM is made up of two blocks as shown in figure 3.2:

1. 16 12-bit locations which are used to store the K-parameters and two constants for a postfilter.
2. 112 byte-wide locations which are available to the user .

Speech Address register - 13-bit register which indexes the 8-Kbyte internal ROM for speech data. Initially, the SAR is loaded with the address of the beginning of the packed speech data.

Internal ROM - 8 Kilobytes of ROM which is used to store the program and the speech data. The speech data is accessed by the *GET* instruction.

Port A - An 8-bit wide port which is available to the user to access external RAM or ROM.

loc 0 – unused
loc 1 – Energy parameter for Speech Synthesizer
loc 2 – k-12 for LPC filter
loc 3 – k-11 for LPC filter
.
.
.
loc 13 – k-1 for LPC filter
loc 14 – Constant 1 for low pass filter
loc 15 – Constant 2 for lowpass filter
loc 16 – first byte available to the programmer
.
.
.
loc 127 – last byte available to the programmer

Figure 3.2: The internal RAM.

Speech Synthesizer - A lattice filter which is implemented in firmware. The coefficients are taken from the 12-bit wide internal RAM locations and is thus user programmable.

3.2 Other Features

The TSP50C10 was not designed explicitly with CELP synthesis in mind. As a result, the chip has capabilities which are not needed by the CELP algorithm. The TSP50C10 has a *mode* register which controls how it operates. By changing the bits in the mode register, the chip can be used as:

1. a microprocessor with no instruction cycles lost to the synthesizer.

2. an LPC vocoder with a digital-to-analog converter. 53% of the instruction cycles are used by the speech synthesizer. The program writes the

LPC values to the RAM and unpacks the speech data from the ROM. Depending on the voiced or unvoiced nature of the speech, a white noise value or a pitch controlled value is written to the filter.

3. a microprocessor and a digital-to-analog converter. The program generates the speech value and then writes it to the DAC.

4. as a microprocessor with a filter and a DAC (CELP mode). 50% of the instruction cycles are used by the speech synthesizer.

When in mode 2, the program uses an internal timer to interpolate the LPC values as it advances through the frame.

3.3 Program Execution

The program instructions are addressed from the internal ROM by the program counter. The current version of the TSP50C10 runs at 7.68 MHz (for an 8KHz sampling rate). An instruction cycle is executed at $\frac{1}{16}$ the oscillator frequency (480,000 instruction cycles per second). However, when the speech synthesizer is running, it uses 50% of the instruction cycles. Therefore only 240,000 instruction cycles per second are available to the CELP program. Furthermore, some instructions, such as moving a constant into the A register, require 2 instruction cycles.

3.4 The Next Generation

Texas Instruments plans to upgrade the current TSP50C10 to obtain better speech quality. The next generation TSP50C10 will have a couple of very

important improvements. First, it will run at twice the current oscillator frequency but allow 90 instruction cycles per sample. Secondly, it will have 240 bytes of internal RAM instead of the current 128 locations. As will be discussed later, these improvements make CELP speech synthesis feasible on a TSP50C10.

3.5 CELP Synthesis Using the TSP50C10

Conceptually, CELP speech synthesis is very straight forward using a TSP50C10. The TSP50C10 runs the CELP synthesis program, taking the CELP parameters from a data bank stored in external or internal ROM. The TSP50C10 hardware generates an interrupt at the speech sampling rate (i.e. every 125 μSec). The interrupt:

1. pushes (i.e. stores) the program counter onto the stack,
2. stores the A, B and X registers in dedicated storage locations,
3. stores the register of flags (*mode* register).

In the interrupt, the program generates an excitation and writes it to the lattice filter. The filter output goes through a D/A converter to produce speech. As a note, once the excitation is generated, the program lets the hardware take care of the speech generation.

3.6 Log Area Ratios

Markel and Gray showed that the spectrum of the vocal tract filter is more sensitive to PARCOR coefficients near ± 1 [Mar&Gra]. To optimize the

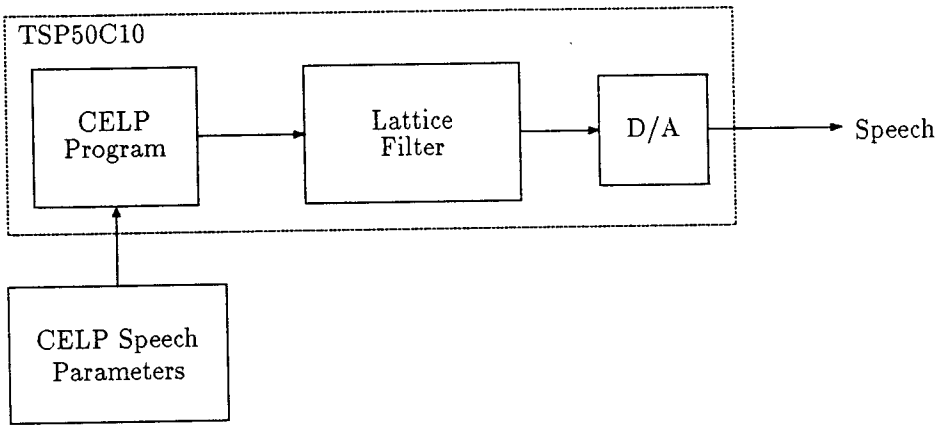


Figure 3.3: Block diagram of CELP synthesis using a TSP50C10.

quantization of the PARCOR coefficients, log area ratios were used. Log area ratios are a transformation of the PARCOR coefficients. We define A_m as the cross-sectional area of the m^{th} stage of the acoustic-tube model of the vocal tract. The ratio of the area of two adjacent sections of the tube $\frac{A_m}{A_{m-1}}$ can be written as $\frac{1-K_m}{1+K_m}$ [Mar&Gra]. By taking the log we get $LAR_m = log\frac{1-K_m}{1+K_m}$.

The transformed K-parameters are then linearly quantized. The resulting table is then inversely transformed back to a table of K-parameters using:

$$K_m = \frac{1 - e^{LAR_m}}{1 + e^{LAR_m}} \tag{3.1}$$

Of course, the analyzer must also have a table of log area ratios with which it determines the K-parameter that is closest to the unquantized K-value.

It is important to note that, from experimentation, log area ratios are the best known quantization scheme for PARCOR coefficients [Mar&Gra].

However, as mentioned in chapter 2, LSP's have uniform spectral sensitivity and interpolate better than PARCOR coefficients. Thus, LSP's are preferred over PARCOR for the filter parameterization. However, since the TSP50C10 has a firmware filter where the coefficients are PARCOR, we are forced to use PARCOR coefficients.

Chapter 4

TSP50C10 Speech Synthesis Program

4.1 Program Format

The TSP50C10 CELP synthesis program can be analyzed in three sections:

- Initialization
- Interrupt Service Routine (ISR)
- Parameter Update

4.1.1 Initialization

The initialization performs all the tasks necessary to prepare the processor for synthesis. The first task is to clear the internal RAM which stores all the variables and values necessary for the program. Secondly, the first frame data is read from the packed data which is stored in external ROM. At this point, the program is ready to start synthesis so the interrupt is enabled. The interrupt causes the program to branch to the ISR at the sampling

frequency. For 8 KHz sampling, the ISR is executed every T_{sample} seconds ($T_{sample} = \frac{1}{8000}$ sec = 125 μSec).

4.1.2 Interrupt Service Routine

The ISR is responsible for generating an input to the lattice filter at the 8 KHz rate. It uses the values written in the update loop (as shown in figure 2.4) and generates the excitation.

After generating the excitation, the ISR writes the excitation into an area of memory called the *workspace* where it is stored until the pitch code book is updated. The pitch code book update is discussed in section 4.1.4.

Before exiting the ISR, the program updates a counter, `samp_done`, which keeps track of the number of samples executed during the current subframe. The counter, `samp_done`, is checked in the update loop.

4.1.3 Parameter Update

The *update loop* performs subframe and frame updates. The subframe updates include:

- read in pitch gain.
- read in pitch index.
- read in code book gain.
- read in code book index.
- interpolate the K-parameters.

- update the adaptive pitch code book.

One approach to subframe updates is to wait until the end of the subframe (i.e., when `Samp_done` = 60) and then perform all the updates. This approach is fine when time is not a constraint. However, when time is a constraint, this approach is inefficient because:

1. reading in new data from ROM is very slow.

2. there exist unused instruction cycles between interrupts.

A more efficient technique is to create a second set of memory locations to store the new parameters. The *free* cycles between interrupts are used to read in the new data and store the values in the second set of memory locations. The new parameters are not used until the next subframe.

To distinguish between the new parameters and the parameters being used by the program, an A/B subframe flag was created. While an 'A' subframe is being executed in the ISR, the program reads new data into the 'B' subframe locations. Then at the end of the subframe, the flag is flipped so that the ISR goes to the 'B' locations to get the data and the program reads in the new data into the 'A' subframe locations. The ping-pong pointer movement is shown in figure 4.1.

Similarly, the K-parameter interpolation for the next subframe is performed during the extra cycles that are not used in the ISR. The interpolated values are written into user-defined RAM locations and then are transferred to the dedicated RAM locations at the beginning of the subframe.

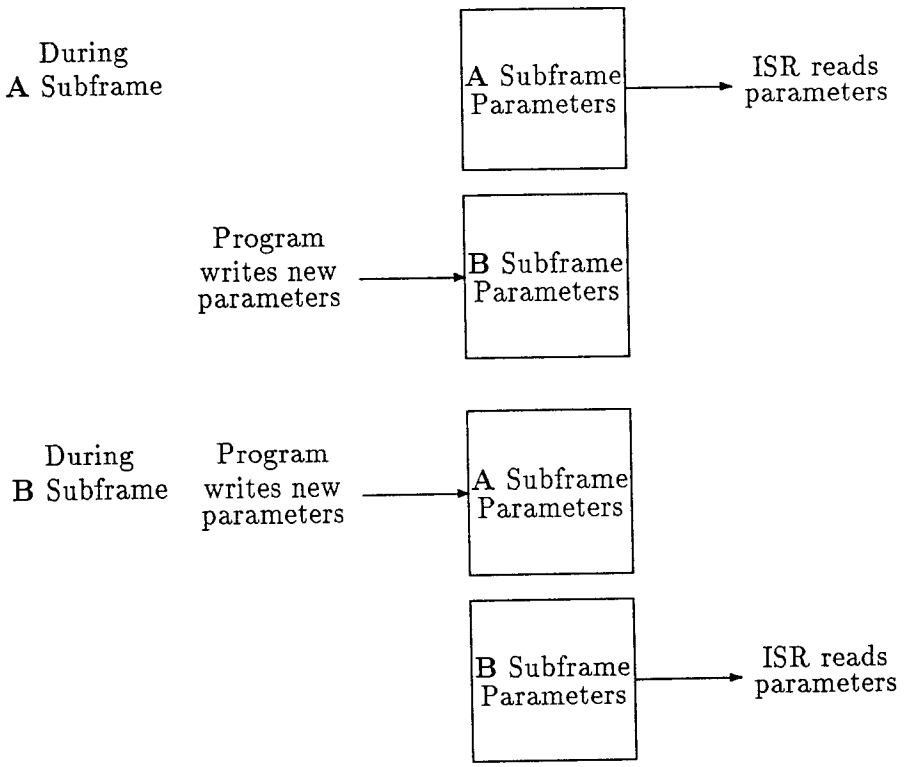


Figure 4.1: The ping-pong pointer technique to time-multiplex the access to the subframe parameters.

4.1.4 Pitch Code Book - A Circular Buffer

The pitch code book is stored as a circular buffer. Therefore it has pointers which indicate the beginning and the end of the code book. The circular buffer resides in memory locations 92 through 239. The pitch code book has 148 locations. 60 locations(called the *workspace*), store the most recent excitation values. Note that the circular buffer is made up of the pitch code book and the *workspace*. Also, for the purpose of discussion, the *workspace* is not part of the pitch code book (see figure 4.2). By careful pointer movement, the *workspace* is included within the same memory used to store the pitch code book. The program variables and constants used for the pitch code book are:

1. `P_base` is a constant equal to 5C Hex. It designates the absolute address of the beginning of the circular buffer. Note that the beginning of the circular buffer is not necessarily the beginning of pitch code book.

2. `P_start` is a variable which stores the address of the beginning of the code book itself. It is the only parameter which is updated when the pitch code book is updated.

3. `delay` is a variable which at the beginning of the subframe, is equal to the pitch index. However, it is used as the index into the pitch code book. The address of the pitch element is generated by adding the `delay` variable to the `P_start` variable. The `delay` variable is decremented after each excitation. Thus a different pitch value is used

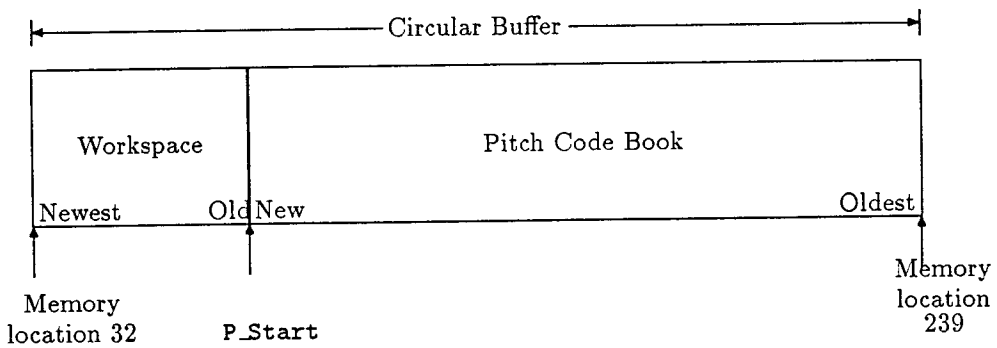


Figure 4.2: Modified circular buffer to illustrate how the pitch code book is updated. The modified buffer uses 60 locations for the *workspace* and 148 locations for the pitch code book

every sample.

4. `P_index` is the actual lag value as written by the CELP analysis routine (i.e., one of the CELP parameters).

It is illustrative to consider a modified circular buffer where 60 locations are used for the *workspace* and 148 locations are used for the pitch code book. Figure 4.2 shows the modified circular buffer which uses 208 locations.

To generate the address into the pitch code book, we add the `delay` variable to the `P_Start` variable. The address is used to indirectly address the actual pitch value in internal RAM. The pitch value is then used to generate the excitation value which is written to the filter and then the *workspace*. The location used to store the excitation in the *workspace* is created by

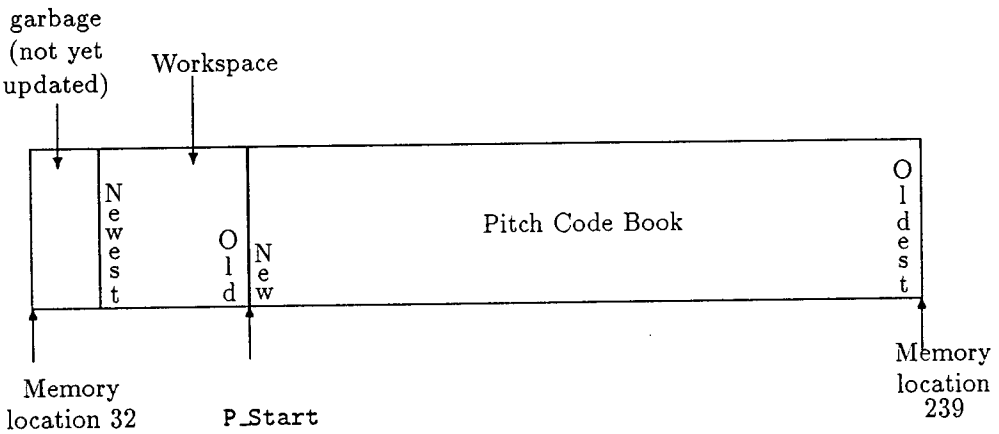


Figure 4.3: Modified circular buffer to illustrate how the pitch code book is updated.

subtracting the number of samples completed from `P_Start` (i.e. `P_Start - Samp_done`). The new excitations are stored in decreasing addresses within the internal RAM. In other words, the *workspace* "grows" from right to left within the internal RAM.

From figure 4.2 and figure 4.3, we see that, within the pitch code book, the oldest value has an address 147 locations "to the right" of `P_Start`. The newest value within the pitch code book is at the address pointed to by `P_Start`. The values within the *workspace* are, of course, even newer than any of those in the pitch code book. The address of the newest value within the *workspace* is `P_Start - Samp_done` (i.e. the last excitation written to internal RAM).

The order in which samples are stored is shown in figure 4.3. Using the

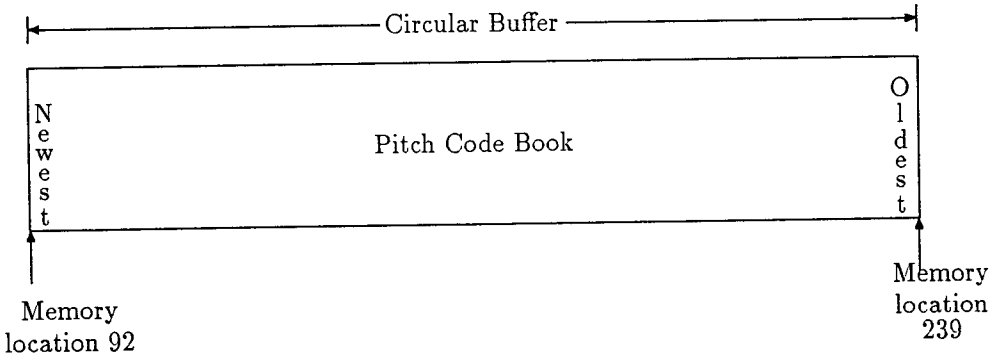


Figure 4.4: Circular buffer as it appears at the beginning of the subframe. The *workspace* has no data at the beginning of the subframe. The circular buffer uses just 148 locations.

modified pitch code book as described above requires 148 memory locations for the pitch code book and 60 locations for the *workspace*. The 208 locations needed makes the algorithm infeasible on even the new TSP50C10. However, by making a slight modification to the above buffer, the number of locations needed can be reduced to 148. By studying figure 4.3 it is seen that when the oldest location is accessed by the program, it becomes free to store the new excitation. Hence, the *workspace* can grow into the pitch code book as shown in figures 4.4-4.7.

For simplicity, the above figures have shown the pitch code book with a fixed `P_Start`. Since the pitch code book is actually stored in a circular buffer, the address of `P_Start` changes. The `P_start` index is decremented by 60 every subframe to keep up with the new values which are written to

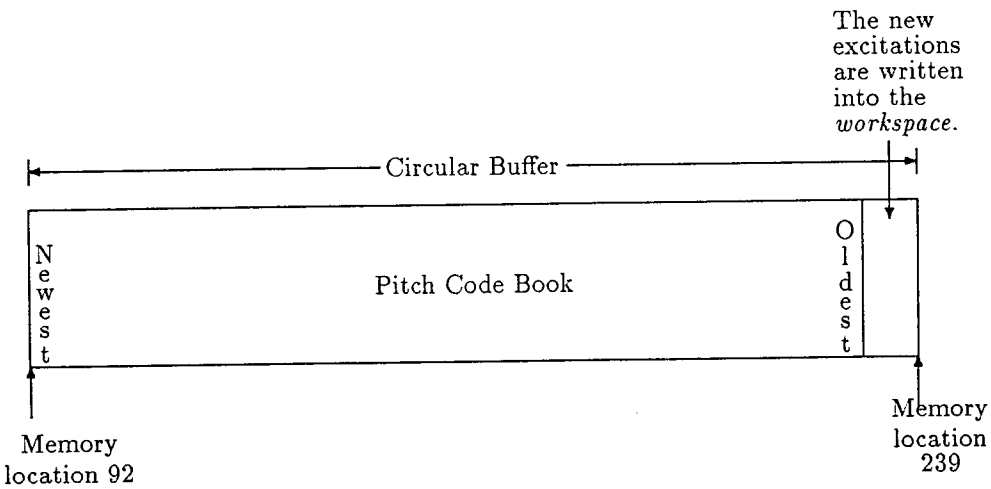


Figure 4.5: Circular buffer as it appears during a subframe. As the excitations are generated, they are written into the *workspace* which grows as the pitch code book locations become free.

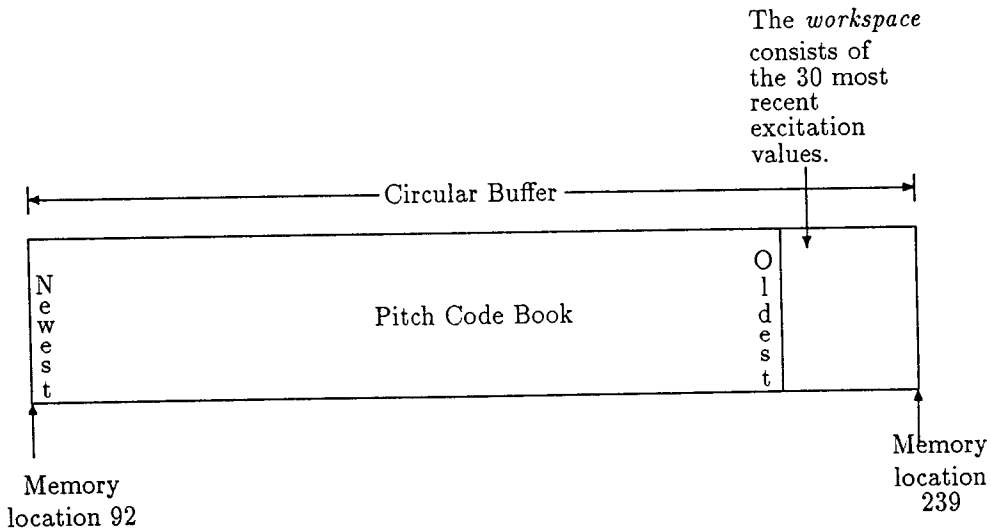


Figure 4.6: Circular buffer as it appears halfway through a subframe. The *workspace* consists of the 30 most recent excitation values.

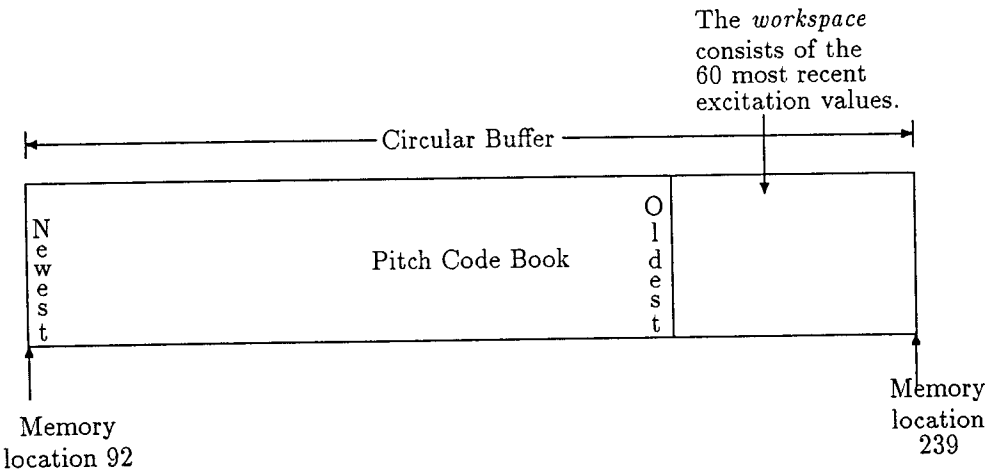


Figure 4.7: Circular buffer as it appears at the end of a subframe. The *workspace* consists of the 60 most recent excitation values. Just prior to the beginning of next subframe, the circular buffer pointers are updated and the workspace is incorporated into the pitch code book as shown in figure 4.4.

the pitch code book.

The P_Start index and the index to store the new excitation are generated by adding and subtracting and can thus underflow and overflow the memory allotted to the circular buffer. The underflow and overflow conditions cause the addresses to wrap around to the upper and lower addresses respectively of the pitch code book. The wrap around cases are:

1. At the end of the subframe, P_Start is updated during the pitch code book update by subtracting 60 (the subframe size). If P_Start is less than the beginning address of the circular buffer, then the length of the circular buffer (148) is added to P_Start and stored as the new P_Start.

2. The second wrap around occurs when the index delay is added to P_start and the result (the address of the pitch code book element) is greater than the address of the circular buffer. In this case, 148 is subtracted to produce the correct address within the circular buffer.

3. The third wrap around occurs due to when the pitch index is less than 60. As delay is decremented, it becomes negative. As specified by the NSA algorithm, the negative delay is added to the pitch index to get a valid pitch index.

4.2 Program Flow

The program flow can be viewed as illustrated in figure 4.8. The program starts by clearing the internal RAM and reading in the first subframe data.

Then, the ISR is enabled. The ISR generates an excitation for the filter. During a subframe, the update loop uses the extra instruction cycles between samples to read in the subframe parameters and interpolate the PARCOR coefficients for the next subframe. During the fourth subframe, the ISR reads in the subframe parameters and the new PARCOR coefficients. After getting the new PARCOR values, the ISR again interpolates them. The flowcharts for the *update* and ISR are given in figures 4.8 and 4.9.

4.3 Program Constraints

4.3.1 The Time Constraint

The current version of the TSP50C10 runs at 7.68 MHz, and as discussed earlier, this means that 240,000 instruction cycles/sec are available to the CELP synthesis program. Since the lattice filter excitation update must occur at 8KHz, only 30 instruction cycles per sample are available to the TSP50C10. Therefore, the ISR, the paramater updates, and the pitch code book are limited to 30 instruction cycles. Using the NSA standard for CELP, the CELP.asm program is able to do the tasks in 69 instruction cycles. Thus, CELP synthesis will not run on the current version of the TSP50C10. However, the program will run on the next generation TSP50C10 which will allow 90 instruction cycles per sample.

4.3.2 Memory Constraint

The current version of the TSP50C10 has 112 byte-wide memory locations available to the programmer. The pitch code book as specified by the NSA

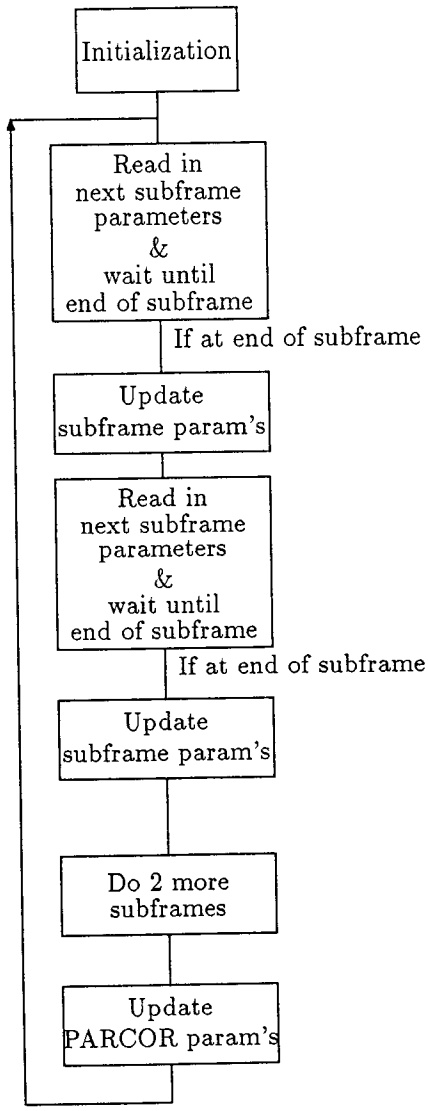


Figure 4.8: Flowchart of CELP synthesis update loop.

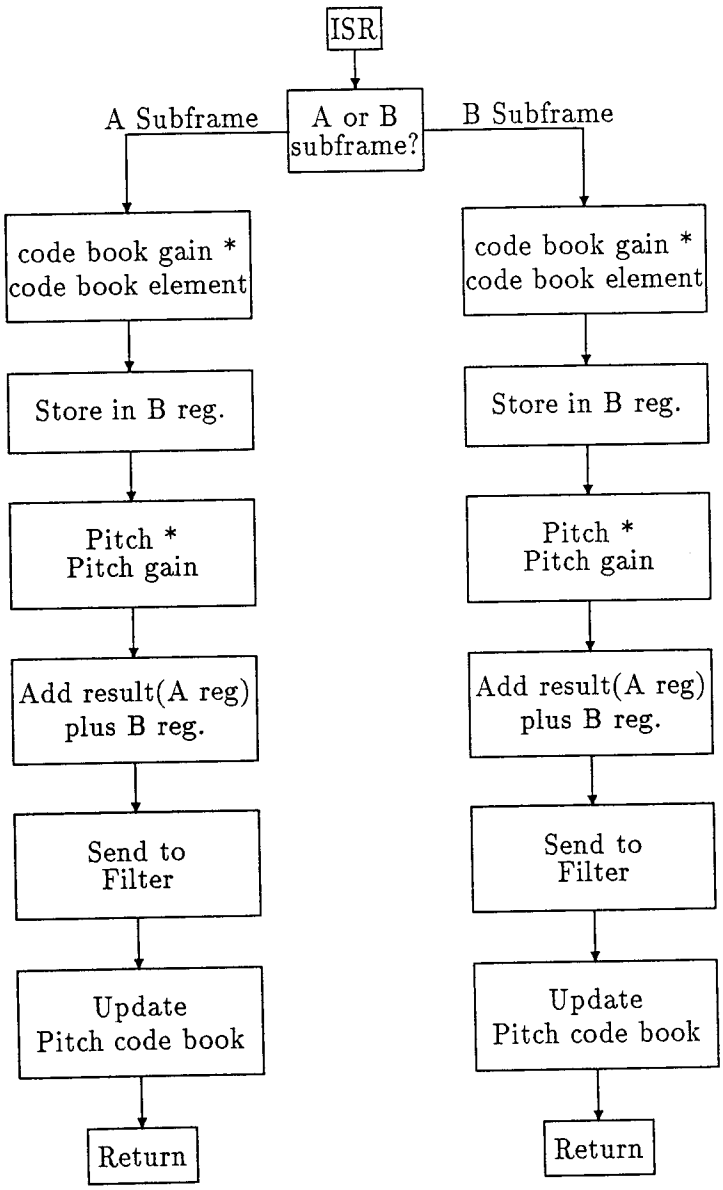


Figure 4.9: Flowchart of CELP synthesis Interrupt Service Routine(ISR). The ISR retrieves data from the A or B subframe locations depending upon the A/B flag which is checked at the beginning of the routine.

algorithm has 148 memory locations. As a result, even without needing to store any other parameters, the CELP synthesis program does not fit into a TSP50C10. The new TSP50C10 will have 240 RAM locations of which 226 are used by the CELP synthesis program.

Chapter 5

Results

Due to the time and memory constraints, the CELP algorithm does not run on the current TSP50C10. Therefore, a simulation of the TSP50C10 was performed. The following three questions were posed.

1. What quality speech will be possible on the new TSP50C10.
2. How can we improve the TSP50C10 results while still using the integer arithmetic?
3. How can we reduce the bit rate of the TSP50C10 and maintain the speech quality.

To gain insight into how to improve speech synthesis using the architecture of the TSP50C10, an analysis was performed whereby the RAM width and the size of the multiplier were varied.

5.1 The Simulation

The simulation was performed on the SUN workstations in C. The filter simulation is the same as that used at Texas Instruments. The simulation of

Algorithm	Parameter	Bits Allocated
NSA 4.8 kbps	10 LSP's per frame	34 bits
	Pitch Index	7 per subframe
	Pitch Gain	5 per subframe
	Stoch. Code Book Index	9 per subframe
	Stoch. Code Book Gain	5 per subframe
TSP50C10 7.47 kbps	10 PARCOR coeff. per frame	100 bits
	Pitch Index	7 per subframe
	Pitch Gain	5 per subframe
	Stoch. Code Book Index	9 per subframe
	Stoch. Code Book Gain	5 per subframe

Figure 5.1: Bit allocation for NSA implementation vs. TSP50C10 version.

the assembly language instructions mimicked the TSP50C10 ALU operation. In other words, since the ALU is 14-bits wide, the numbers were stored in 14-bit format and thus the precision of the numbers in the simulation realistically reflected the precision in the TSP50C10. Similarly the pitch code book values were scaled down to the width of the RAM and the K-parameters were stored as 12-bit integers.

5.2 Using the New Generation Architecture

The major difference between the TSP50C10 implementation and the NSA version is the integer arithmetic and the associated precision. The TSP50C10 synthesized speech is slightly degraded as compared to the floating point synthesized speech. Using the floating point NSA algorithm on the `dam2.spd` speech file, the segmented SNR is 7.05 dB, whereas the integer version has a segmented SNR of 2.96 dB. The NSA algorithm uses 4.8 kbps while the TSP50C10 uses 7.47 kbps. The bit allocation for the two techniques is given in figure 5.1.

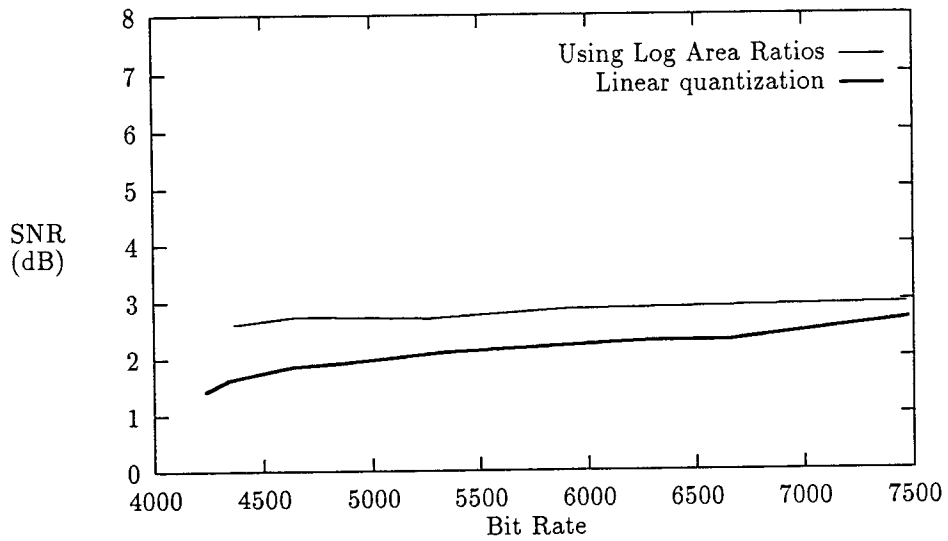


Figure 5.2: Log area ratios signifcantly improve the speech quality at low bit rates.

The simulation analyzes the trade-off between bit rate and hardare precision. At lower bit rates, log area ratios (LAR's) were used to obtain better quantization of the PARCOR coefficients. The log area ratios make a significant difference in the speech quality at low bit rates as shown in 5.2.

The segmented SNR's for various runs of the simulation are tabulated and graphed in the next section.

Sim. No.	Description	Bit Rate	SNR(dB)
–	Full 32-bit floating point using LSP's	4.8 kbps	7.05
Sim. 1	Full 32-bit floating point using PARCOR	7.47 kbps	7.37
Sim. 2	*Standard* TSP50C10 Simulation 148 byte-wide pitch code book 14-bit intermediate calculations 14-bit excitation 12-bit uniform bit allocation	7.47kbps	2.96

Figure 5.3: Simulations performed to learn bits-to-SNR trade-off.

5.3 Tables and Graphs

ALU	RAM width	dam2	f002	animals
14-bit ALU	8 bits	2.80	3.64	3.45
14-bit ALU	10 bits	4.39	6.23	6.52
14-bit ALU	12 bits	5.16	5.85	7.09
16-bit ALU	8 bits	2.74	2.26	4.63
16-bit ALU	10 bits	4.27	6.40	4.66
16-bit ALU	12 bits	4.92	5.70	6.46
16-bit ALU	14 bits	5.05	6.19	6.27

Figure 5.4: The segmented SNR(dB) versus hardware parameters at 4.23 kbps.

From figure 5.7 the segmented SNR is very flat over bit rates between 4.23 and 7.47 kbps for a given RAM width. However, by increasing the RAM width, the segmented SNR jumps significantly. By comparing figures 5.7 and 5.8 the segmented SNR remains essentially unchanged by increasing the ALU by two bits.

Since the speech quality improves with the RAM width, the best configuration is the 16-bit ALU and 14-bit RAM.

The 14-bit ALU and 14-bit RAM configuration is not shown in the graphs

because the SNR for each was extremely low. The hardware performs better with a 14-bit ALU and 8-bit RAM than with a 14-bit ALU and 14-bit RAM! With the 14-bit ALU and 8-bit RAM, products are scaled down to 7-bits to have a magnitude of $\pm 2^{13}$. Using a 14-bit ALU and 14-bit RAM, numbers would be scaled by 13 bits to have a magnitude of $\pm 2^{13}$.

ALU	RAM width	dam2	f002	animals
14-bit ALU	8 bits	2.69	3.67	3.37
14-bit ALU	10 bits	4.55	5.98	6.80
14-bit ALU	12 bits	5.46	5.95	7.68
16-bit ALU	8 bits	2.99	2.31	4.83
16-bit ALU	10 bits	4.41	5.84	7.07
16-bit ALU	12 bits	5.07	5.93	6.88
16-bit ALU	14 bits	5.22	6.36	6.26

Figure 5.5: The segmented SNR(dB) versus hardware parameters at 5.23 kbps.

ALU	RAM width	dam2	f002	animals
14-bit ALU	8 bits	2.96	3.09	3.48
14-bit ALU	10 bits	4.66	6.29	6.69
14-bit ALU	12 bits	5.64	5.31	7.70
16-bit ALU	8 bits	2.92	4.66	4.84
16-bit ALU	10 bits	4.50	6.13	6.79
16-bit ALU	12 bits	5.27	6.05	6.52
16-bit ALU	14 bits	5.29	5.72	6.21

Figure 5.6: The segmented SNR(dB) versus hardware parameters at 7.47 kbps.

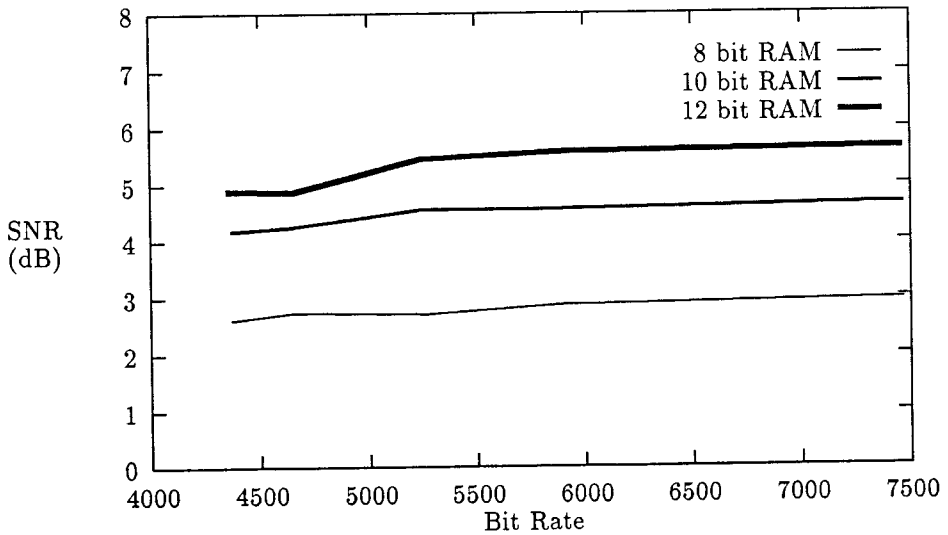


Figure 5.7: The segmented SNR as a function of the bit rate at various RAM widths for a 14-bit ALU(from dam2.spd speech file.)

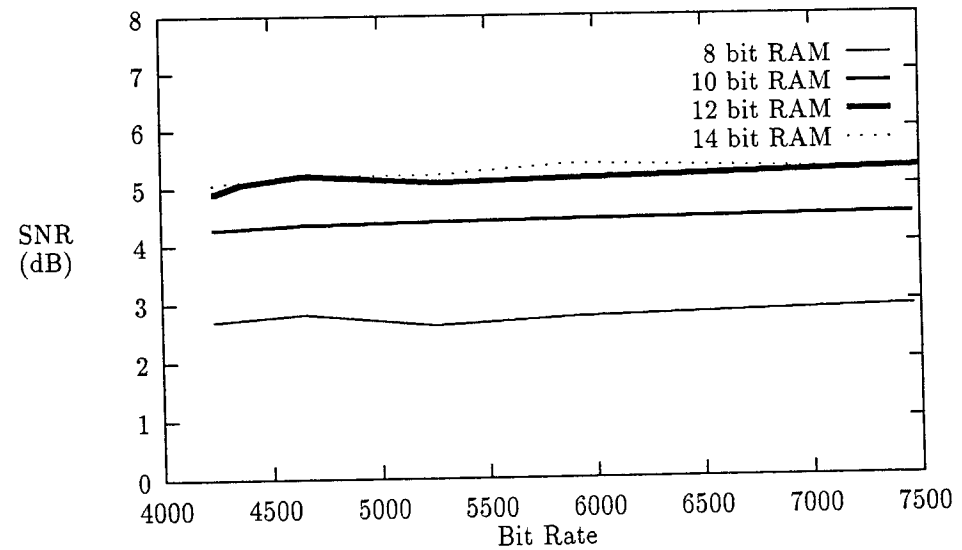


Figure 5.8: The segmented SNR as a function of the bit rate at various RAM widths for a 16-bit ALU(from dam2.spd speech file.)

5.4 Hardware Suggestions

5.4.1 The Filter Order

When the graphs from the previous section were presented at Texas Instruments, it was suggested that the RAM width should be increased for the next generation TSP50C10. They agreed; however, there is not enough circuit area on the chip to include a wider RAM.

From the previous section, it is seen that the segmented SNR is flat down to 4.23 kbps. To achieve lower bit rates, the high order LPC-10 coefficients were set to zero which had negligible effect on the speech quality. Hence higher quality speech would be possible with the new TSP50C10 if the LPC-10 filter were changed to a lower order filter (e.g., LPC-7) while increasing the RAM width.

Apart from producing higher quality speech, lowering the order of the filter would have three beneficial effects on the program. Firstly, fewer instruction cycles would be stolen from the microprocessor portion of the chip by the synthesizer. Secondly, the bit rate would go down because fewer filter parameters would be needed. Thirdly, the program execution time would speed up in a critical area,(i.e., the K-parameter update).

5.4.2 LSP Parameterization

The TSP50C10 uses a PARCOR lattice filter for the vocal tract model. Due to the better properties of the LSP parameters, it is suggested that LSP filter parameterization is used instead of PARCOR coefficients.

5.5 Conclusion

CELP synthesis is not feasible on the current TSP50C10 but will be on the next generation TSP50C10. If log area ratios are used to quantize the PARCOR coeffients, remarkably good speech quality is obtainable using the integer arithmetic at bit rates down to 4.23 kbps. The speech quality improves only slightly as the bit rate is increased above 7.47 kbps. To improve the speech the width of the internal RAM must be increased. Interestingly, changing from a 14-bit ALU to a 16-bit ALU makes very little difference to the speech quality. In terms of trading circuit area on the VLSI chip, CELP would be better served by reducing the LPC-10 filter to LPC-6 or LPC-7 and increasing the RAM width.

By recognizing that CELP does not need any "speech editing" to optimize the speech parameters depending upon the speaker, CELP is a viable speech synthesis technique for the next generation TSP50C10.

Bibliography

[Atal1] Atal, Bishnu S., 'Predictive Coding of Speech at Low Bit Rates', *IEEE Transactions on Communications*, Vol. Com-30, No.4, April 1982.

[Atal2] Atal, B.S., 'High Quality Speech at Low Bit Rates: Multi-pulse and Stochastically Excited Linear Predictive Coders', *International Conference on Acoustics, Speech, and Signal Processing*, pp.1-4, April 1986.

[Campbell] Campbell, Joseph P. Jr., Vanoy C. Welch and Thomas E. Tremain, 'The New 4800 bps Voice Coding Standard', *Military & Government Speech Tech 1989*, Arlington, Virginia, November 14, 1989.

[Campbell2] Campbell, Joseph P. Jr., Thomas E. Tremain, Vanoy C. Welch, 'The DOD 4.8 KBPS Standard', *Kluwer Academic Publishers*, 1990.

[Fenichel] Fenichel, Robert M., 'Proposed Federal Standard 1016 Telecommunications: Analog to Digital Conversion of Radio Voice by 4,800 bit/second Code Excited Linear Prediction(CELP).' November 13, 1989.

[Flanagan] Flanagan, James L., Manfred R. Schroeder, Bishnu S. Atal, Ronald E. Crochiere, Nuggegally S. Jayant, and Jose M. Tribolet, 'Speech Coding', *IEEE Transactions on Communications*, Vol. Com-27, No.4, April 1979.

[Kab&Ram] Kabal, Peter, and Ravi Prakash Ramachandran, 'The Computation of Line Spectral Frequencies Using Chebyshev Polynomials', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. Assp-34, No. 6, pp. 1419-1425, Dec. 1986.

[Makhoul] Makhoul, J., 'Linear Prediction: A Tutorial Review', *Proc. IEEE*, Vol. 63, pp. 561-580, 1975.

[Mar&Gra] Markel, J.D. and A. H. Gray Jr., 'Linear Prediction of Speech', Springer-Verlag, New York, 1976.

[Papamic] Papamichalis, Panos E.,'Practical Approaches to Speech Coding', Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[Rab&Sch] Rabiner, L.R., and R.W. Schafer, 'Digital Processing of Speech Signals', Prentice Hall, Englewood Cliffs, New Jersey, 1978.

[Rose&Bar] Rose, R.C., and T. P. Barnwell III, 'Quality Comparison of low Complexity 4800 bps Self Excited Vocoders', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 1637-1640, 1987.

[Schroeder] Schroeder, Manfred R., and Bishnu S. Atal, 'Stochastic Coding of Speech at Very Low Bit Rates: The Importance of Speech Perception', *Speech Communication 4(1985)*, Elsevier Science Publishers B.V., North-Holland.

[Schroeder] Schroeder, Manfred R., and Bishnu S. Atal, 'Code Excited Linear Prediction(CELP) High Quality Speech at Very Low Bit Rates.' *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 937-940, March 1985.

[Soo&Juang] Soong, F. K., and B. W. Juang, 'Line Spectrum Pair (LSP) and Speech Data Compression', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, San Diego, CA, Mar. 1984. pp. 1.10.1-1.10.4.

[Sug&Itak] Sugamura, Noboru, and Fumitada Itakura, 'From LPC to LSP', *Speech Communication*, Vol. 5, No. 2, June 1986, pp. 199-215.

[T.I.] Texas Instruments, 'TSP50C10 Speech Synthesizer Design Manual', Texas Instruments, Linear Products, March 1989.

[Tra&Ata] Trancoso, Isabel M., and Bishnu S. Atal, 'Efficient Search Procedures for Selecting the Optimum Innovation in Stochastic Coders', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, No.3, pp. 385-396, March 1990.

[Tremain] Tremain, Thomas E., Joseph P. Campbell Jr., and Vanoy C. Welch, 'A 4.8 kbps Code Excited Linear Predictive Coder', U.S. Department of Defense, R5 Fort Meade, Maryland, U.S.A.

[Tretter] Tretter, Steven A., 'Introduction to Discrete-Time Signal Processing', John Wiley & Sons, New York, 1976.