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 a désirer, surtout si les pages originales ont été dactylogra phiées 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.

A Multichannel, Demand-Assignment Echo Canceller for Point-to-Multipoint Subscriber Radio Systems

Thanh-Son Nguyen

A Thesis

in

The Department

of

Electrical Engineering

Presented in Partial Fulfilment of the Requirements
for the Degree of Master of Engineering at
Concordia University
Montreal, Quebec, Canada

April, 1991

ISBN 0-315-68746-0

Canada

ABSTRACT

A Multichannel, Demand-Assignment Echo Canceller for Point-to-Multipoint Subscriber Radio Systems

Thanh-Son Nguyen

Echo is a disturbing, but unavoidable phenomenon in any telephony network. Echo control devices are normally introduced at the subscriber interface points to reduce the round-trip delay. This arrangement requires M pairs of echo control devices where M, the number of subscribers, is much larger than N, the number of available trunks in Point-to-Multipoint (P-MP), Demand-Assignment Subscriber Radio systems. Also, due to dynamic trunk allocation of Demand-Assignment schemes, echo control devices need to adapt to different connected subscribers. Pursuing these objectives, various echo cancellation algorithms and structures are studied in this thesis.

The statistical approach based on the Block Least Mean Squares (BLMS), and the statistical behavior of trunks, is proposed. This approach is best suited to the dynamic trunk allocation of a P-MP system. Echo cancellation process is performed only on active trunks. The Multichannel, Demand-Assignment Echo Canceller (MCDAEC) is implemented, and its performances are evaluated.

ACKNOWLEDGEMENT

The author wishes to thank Dr. Tho Le-Ngoc for his guidance throughout the course of this research. His excellent characters, and his confidence in my fulfilling the project are grateful.

Special thanks also to the National Research Council for support in part this research work under the IRAP program. Thanks to my colleagues at SR Telecom Inc. for helping me at some critical moments, especially Mr. Djordje Konforti, who is now with BNR, for his Hardware design.

Lastly, I like to say thanks to my wife and my little boy, who share with me wonderful, sometimes frustrated moments. Thanks to my parents, brothers, and sisters for their supports in my early days.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Point-to-Multipoint (P-MP) Subscriber Radio systems have been widely used for services in rural, and suburban areas where conventional telephony applications may be costly, and inefficient in terms of maintenance. All P-MP systems share a common feature in having a Central station (C/S), and a number of Outstations (O/S).

Fig.1 shows a typical configuration of a P-MP subscriber radio system. An efficient Random-Request, Demand-Assignment, Time-Division Multiple-Access (RR-DA-TDMA) protocol [1] is used to provide services to M subscribers with N channels (or trunks) where M >> N on a demand basis. When a burst randomly requested by an O/S is received, the C/S will acknowledge the requesting station, and assign it an available traffic trunk. The traffic trunk can be data, PCM, or ADPCM. Due to the nature of grouping voice samples in a TDMA scheme, transmission delay is introduced in the system. The transmission delay also increases with the number of inserted repeaters. This delay may increase the annoying effect of echo.

Echo is generated at any point where the speech signal encounters a mismatch in impedance. Essentially, all significant echoes are produced at the hybrid which acts as a two-wire to four-wire interface in a subscriber radio system.

The deleterious effects of such echoes depend upon their

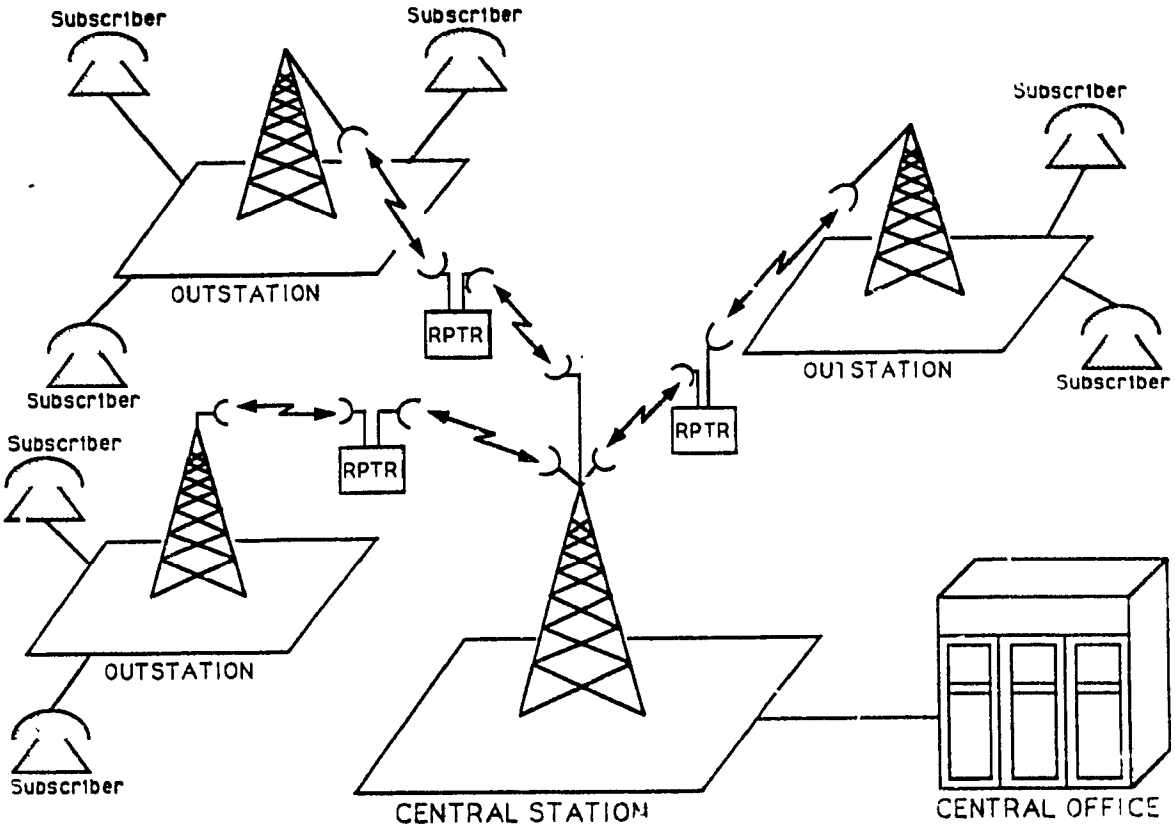


Figure 1. A typical Point-to-Multipoint system

loudness, signal distortion, and system delay. Generally speaking, the more echo is delayed, the more it is untolerable. The annoying effect of echo in the extreme case where the total round trip delay may exceed 30ms motivates a need for finding means to combat this problem.

Echo control devices are normally introduced at the subscriber interface points to reduce the echo path delay. This arrangement requires M pairs of echo control devices where M, the number of subscribers, is much larger than N, the number of available trunks in a DA, P-MP system. In a DA system, traffic trunks will be assigned upon receiving

requests by subscribers. Therefore, for economic reasons, it is more desirable to have a *Group Echo Control Device* operating on a trunk basis. This *Group Echo Control Device* is equivalent to a maximum of N pairs of trunk Echo Control devices. Also, due to the dynamic trunk allocation of DA schemes, echo control devices need to adapt to different connected subscribers. This is the motivation of the research on a *Statistical Multichannel, Demand-Assignment Echo Canceller* reported in this thesis.

The main objectives of this research are to study algorithms and structures of *Statistical Multichannel, Demand-Assignment Echo Canceller* suitable for P-MP systems. Pursuing these objectives, this research covers the following points :

(i) Performance comparison of various Echo Cancellations in terms of complexity, convergence speed.

(ii) Introduction of a statistical approach to design the Multichannel, Demand-Assignment Echo Canceller.

(iii) Implementation and performance evaluation of the introduced Multichannel, Demand-Assignment Echo Canceller.

The thesis consists of five chapters. After this introduction, chapter 2 presents a review of Echo control methods and their performance. Via Net Loss (VNL), a technique which inserts loss in the transmission path, was first used to cancel the echo. The inefficiency of this method when applied in long-circuit network lead to the invention of echo

suppressor. Description of the echo suppressor will identify the lack of performance of this technique in double-talk situation. This leads to the concept of echo cancellation. Both the Least Mean-Squared (LMS), and Least Squares (LS) are discussed.

Chapter 3 describes the configuration of a P-MP subscriber radio system with an emphasis on the echo cancellation problem. Particular design parameters such as location of echo cancellers, echo path delay will be identified. The selection of suitable Echo Cancellation algorithms, and DSPs is discussed. Computer simulations based on deterministic approach are carried out to investigate the convergence speed of the selected echo cancellation algorithm. The statistical approach is proposed to increase the number of trunks per DSP. This approach is based on the statistical behavior of trunks that the probability of trunks activated at the same time is negligible.

Chapter 4 describes the implementation of the proposed Multichannel, Demand-Assignment Echo Canceller based on the statistical approach. Hardware and Software designs will be described along with test results.

Finally, chapter 5 presents the conclusion.

CHAPTER 2

ECHO CONTROL TECHNIQUES & ALGORITHMS

Known echo control devices such as Via Net Loss, Echo Suppressor find their applications in many circuits. With the evolution of VLSI and DSP technologies echo canceller, the most recent echo control device, is widely used due to its high performance in echo cancellation, and double-talking situation. Following in this chapter we present a performance comparison of echo control devices, and also an analysis of echo cancellation algorithms.

2.1 Echo Control Methods

With rare exceptions, all conversations take place in the presence of echo. Echo is a phenomenon in which one's speech reflected to him/her after some delay time due to impedance mismatch at the termination end. The effect of echo becomes more pronounced as the time delay between the speech and echo increases. Fig.2 shows a typical connection via a network where echoes are generated by the impedance mismatch, and imperfect isolation at the hybrids B, and A. To combat the effect of echo many approaches have been employed. In the following sub-sections we examine Echo Control methods such as Via Net Loss (VNL), Echo Suppressor, and Echo Cancellation.

2.1.1 Via Net Loss (VNL):

Introduced in the early days of Echo Controlling, this technique inserts loss at both ends of the transmission paths to attenuate echo. The VNL has found its application in short circuit networks where delay is short, and the insertion loss is acceptable. In long circuit, however, the VNL degrades the system's performance because high insertion loss may block speech signals from one subscriber to another [2].

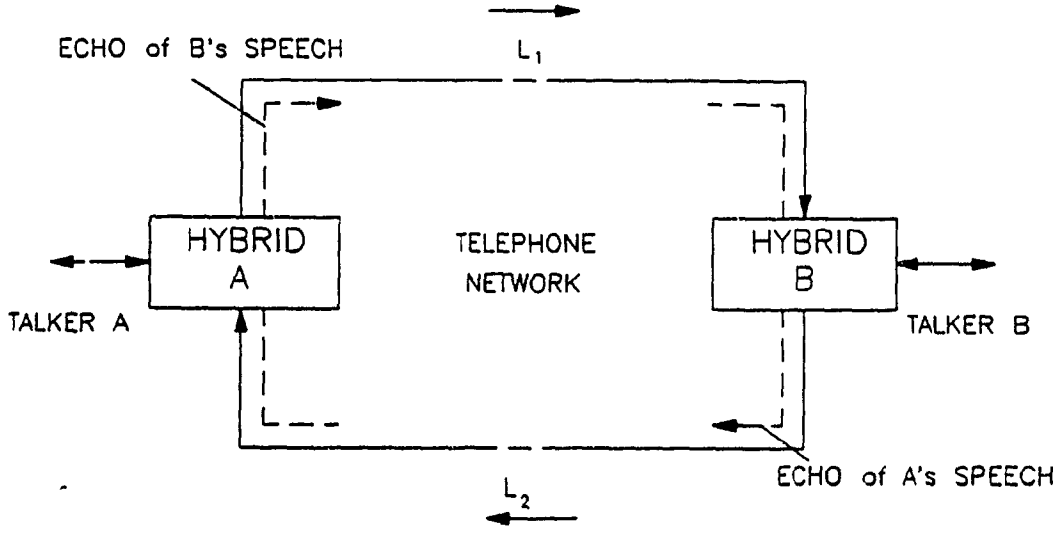


Figure 2. A typical Telephony network with echo

2.1.2 Echo Suppressor :

The drawback of the VNL leads to the invention of Echo Suppressor. It was noted that most conversations consist of single talking, i.e one person speaks at a time. Under this

condition, echoes which return to the talker, are not mixed with desired signal, and therefore can be removed from the circuit by introducing a loss in the current transmission path. This echo control method is called Echo Suppressor. As shown in Fig.3 the switch is activated upon detecting the A's signal. This switch will pass speaker A's signal, and blocks all returning signal from speaker B. Under this scheme it is apparent that Echo Suppressor may interfere with normal conversations when both subscribers speak at the same time, i.e double-talking. When the device is fully effective against echoes, it modifies the transmission system so that it is no longer a full-duplex link capable of carrying information in both directions simultaneously. Instead, it approaches a half-duplex system which can be used in either one direction or the other, but not in both at once. An operating Echo Suppressor must accomplish one major goal : removal of perceptible echo with minimum degradation of desired speech. An obvious problem with Echo Suppressor is that switch detection may not be fast enough when talking direction changes all the time. Modern Echo Suppressor is designed to improve this performance by introducing hang-over time. Such an Echo Suppressor is described by M.M.Sondhi and D.A.Berkley [2].

In a single-talk situation, signal at point A will cease following the transmission delay when speaker A stops talking. Because of the end delay, echo may be produced even after

speaker A stops talking. To prevent this, the switch as shown

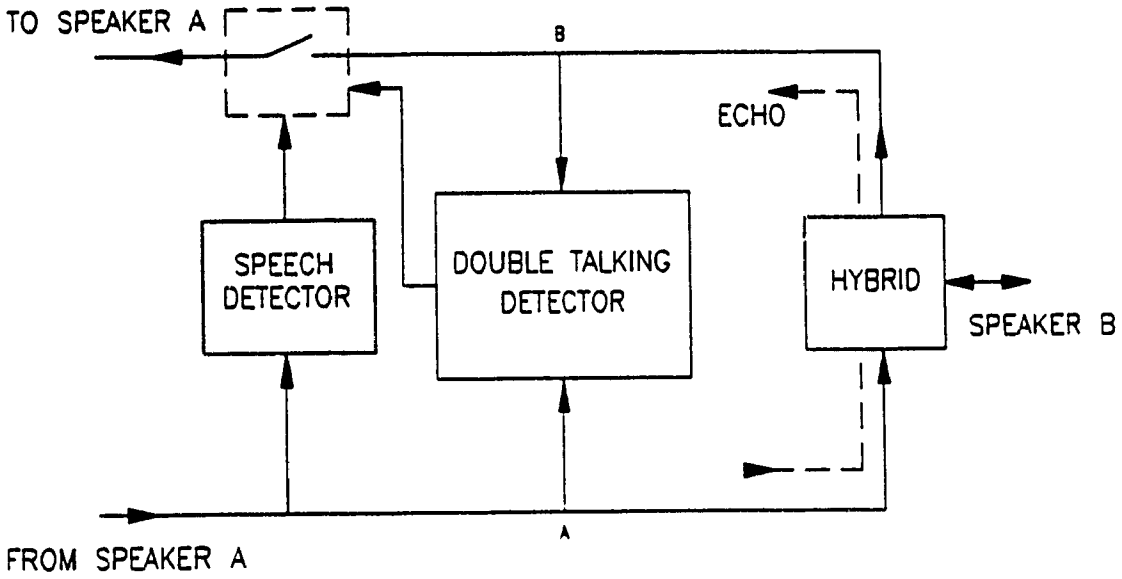


Figure 3. Logical Structure of an Echo Suppressor

in Fig.3 will not be released until a certain time called hang-over time. This hang-over time must exceed the end delay. By introducing this hang-over time the switch detection can be relaxed. However, a hang-over time in excess of a few milliseconds may prevent transmission of B's speech if speaker B begins talking shortly after A is heard to stop. An Echo Suppressor is not satisfatory for the following reasons :

- It may clip, and chop speech or data signals.
- It does not provide good echo control during double-talk.
- It requires switch control during signaling, and in tandem connection.

• It may effect touch-tone signaling when dial tone sent from the incoming switch system is being received.

2.1.3 Echo Cancellation :

Emerged in the sixties, Echo Cancellation soon became recognized as the best technique of Echo control. As shown in Fig.4 it operates on the principle of adaptive modeling of the echo signal, and subsequently cancels the estimated echo in the received signal, thus leaving the speech paths unaffected.

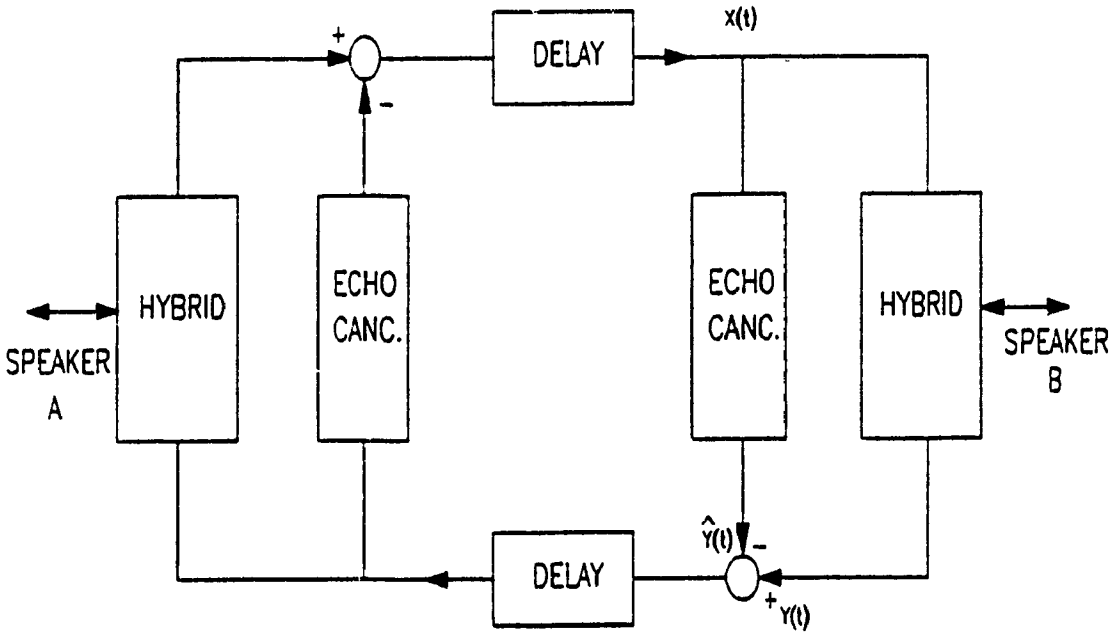


Figure 4. Principle of Echo Cancellation

While the concept of Echo Cancellation may seem simple "To remove the echo, subtract it" it involves some difficulties as follows :

(i) Adaptive Modelling

Echo path is unknown, varied and must be identified and modelled. Using adaptive technique as shown in Fig.4 an estimated model of the impulse response of the actual echo path is generated. Normally, it is assumed that the echo path is linear, and time-invariant [3]. Under these assumptions, the return signal y(t) to the telephone A in Fig.4 can be expressed as :

$$y(t) = h(t) * x(t) + v(t) \qquad (2.1)$$

where :

x(t) is the far-end speech from telephone A.

v(t) is the near-end speech from telephone B.

h(t) is the impulse response of the echo path.

The function of the Echo Canceller box in Fig.4 is to estimate h(t) by $\hat{h}(t)$. The ideal case is when $h(t) = \hat{h}(t)$, i.e echo path response is perfectly estimated.

(ii) Double-talk detection

A major improvement compared with Echo Suppressor is the performance of Echo Canceller in presence of double-talk. Instead of blocking the far-end speech from returning to the talker, the Echo Canceller is transparent, i.e returning signal consists of both echo and near-end signals.

Different Echo Cancellation algorithms have been derived, and will be discussed in detail in section 2.3. We now examine the performance criteria of Echo Canceller.

2.2 Performance Criteria of an Echo Canceller :

The CCITT G.165 Recommendation [4] specifies the required performance, and measured parameters of an Echo Canceller. Using the model of an Echo Canceller in Fig.5 we define the following parameters :

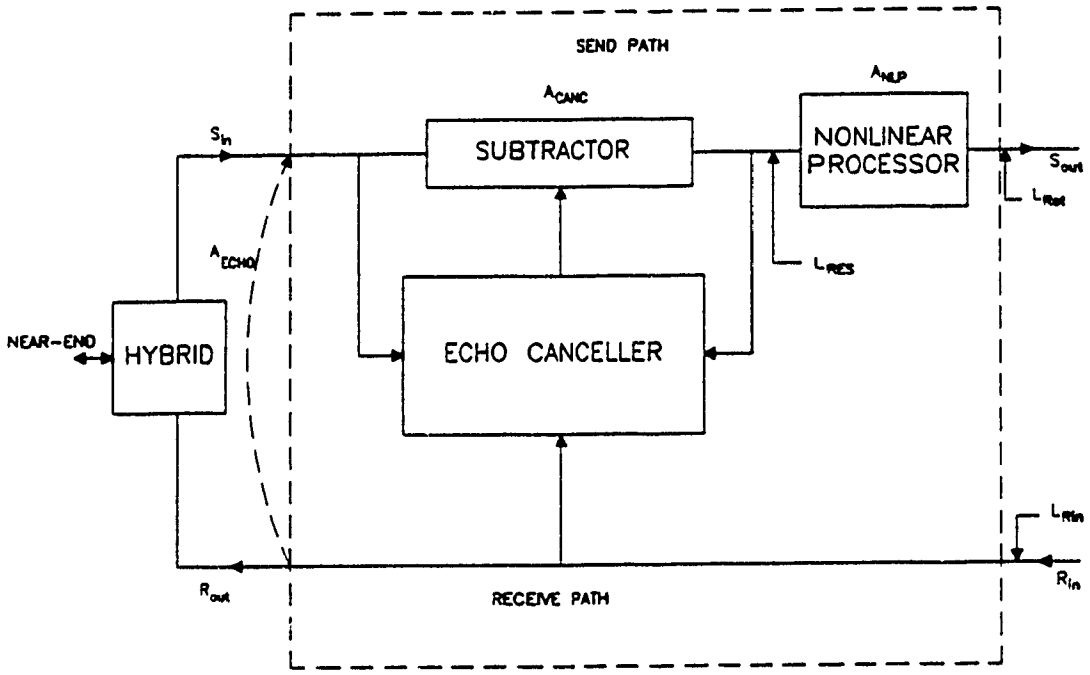


Figure 5. The Echo Canceller Model

(a) Echo Loss (A_{echo}) :

This is the attenuation of a signal from the receive-out port (Rout), to the send-in port (Sin) of an Echo Canceller, due to transmission and hybrid loss, i.e. the loss in the echo path.

(b) Cancellation (A_{canc}) :

This is the attenuation of the echo signal as it passes

through the send path of an Echo Canceller. This attenuation excludes the attenuation due to nonlinear processor.

(c) Residual Echo Level (L_{Res}) :

This is the level of the echo signal at the send-out port of an operating Echo Canceller after cancellation of the circuit signal.

It is related to L_{Rin} by :

$$L_{Res} = L_{Rin} - A_{echo} - A_{canc} \qquad (2.2)$$

Any nonlinear processor is not included.

(d) Nonlinear processor (NLP) :

It is a device having a defined suppression threshold level and in which :

- signals with a detected level as being below the threshold are suppressed.
- signals with a detected level as being above the threshold are passed although the signal may be distorted.

(e) Nonlinear processor loss (A_{nlp}) :

Additional attenuation of residual echo level by a nonlinear processor placed in the send path of an Echo Canceller.

(f) Returned Echo level (L_{Ret}) :

The level of the signal at the send-out port of an operating Echo Canceller which will be returned to the

talker. L_{Res} is related to L_{Rin} by

$$L_{Ret} = L_{Rin} - (A_{echo} + A_{canc} + A_{nlp}) \quad (2.3)$$

(g) Convergence time:

For a defined echo path, the interval between the instant a defined test signal is applied to the receive-in port of an Echo Canceller with the estimated echo path impulse response initially set to zero, and the instant the returned echo level at the send- out port reaches a defined level.

As explained in section 2.1.3, an Echo Canceller must be able to synthesize a replica of the echo path impulse response. This impulse response typically spans a time interval of the order of 2ms to 5ms. Following are the fundamental requirements of an Echo Canceller :

(i) Rapid convergence : the echo paths change as the Echo Canceller is used in successive connections. Under these conditions the requirement is that the Echo Canceller should converge within 500ms.

(ii) Subjective low returned echo level during single talk : residual echo level should be small regardless of the level of the receive speech, and the characteristics of the echo path. Typical Echo Canceller for voice cancellation has a residual echo level of -30dB (including the hybrid loss of 6dB).

(iii) Low divergence during double-talk : when both

subscribers speak at the same time, the Echo Canceller may interpret the transmit signal as a new echo path, and try to adapt to it. It is desirable that very little divergence occurs while this is taking place.

2.3 Echo Cancellation Algorithms :

As discussed earlier, the principle of Echo Cancellation is to generate a replica of echo, and to remove it from the received signal. To obtain this most algorithms employ what is called the adaptive estimation process. An initial guess of tap weights are made, and the residual error is calculated based on these values. In the next iteration, the tap weights are adjusted in such a way that as k (number of iterations) approaches infinity the residual error becomes negligible. At this stage, we say the algorithm converges. Two basic processes can be summarized in adaptive filter :

- The *echo cancelling process* which involves obtaining a replica of the desired response, and generating an estimation error by comparing the estimate with the actual value of the desired response.

- The *coefficient updating process* which involves the automatic adjustment of the filter coefficients in accordance with some algorithms. Algorithms for carrying out these two processes are : the Least Mean Squares (LMS), and the Least Squares (LS). The LMS minimizes mean-squared error, whereas

the LS approach is to minimize the sum of mean-squared errors. Both approaches have different convergence characteristics. Discussion of these algorithms focuses on the following points

- Derivation of the algorithm.
- Discussion of the convergence behavior.

Signal processing model for both algorithms is shown in Fig.6. In this figure :

a is the transversal filter coefficient vector.

h is the echo path response coefficient vector.

x is the far-end signal vector.

y is the desired response vector.

$\hat{\mathbf{y}}$ is the estimate of the desired response vector.

e is the residual error vector.

N is the length of the transversal vector.

Two assumptions are made in Echo Canceller :

- the echo path is linear.
- the echo path is time-invariant.

Because of these assumptions, the desired response can be expressed as a linear combination of filter coefficients and time delay. We begin the discussion with the LMS algorithm.

2.3.1 Least Mean Square (LMS) Algorithm

Based on the steepest descent method, the Least Mean Square (which is also called stochastic gradient) is derived

to minimize mean-squared error by updating weight vector in accordance with incoming data. Because it is simple (no matrix computation is required) it is implemented in most of the existing adaptive filters.

2.3.1.1 Derivation of the LMS algorithm

As shown in Fig.6 :

$$e(k) = y(k) - \hat{y}(k) \tag{2.4}$$

where the estimated signal $\hat{y}(k)$ can be expressed as a convolution of input sequence $x(k)$, and the transversal filter coefficients :

$$\hat{y}(k) = \sum_{i=0}^{N-1} a(i)\, x(k-i) \tag{2.5}$$

Expressing as a vector inner product the above relation can be rewritten as :

$$\hat{y}(k) = \mathbf{a}^T(k)\mathbf{x}(k) \tag{2.6}$$

where :

$$\mathbf{a}^T(k) = (a_0(k), a_1(k), \ldots, a_{N-1}(k)) \tag{2.7}$$

$$\mathbf{x}^T(k) = (x(k), x(k-1), \ldots, x(k-N+1)) \tag{2.8}$$

Substituting equation (2.6) in (2.4) yields :

$$e(k) = y(k) - \mathbf{a}^T(k)\, \mathbf{x}(k)$$

The squared err r is :

$$e^2(k) = (y(k) - \mathbf{a}^T(k)\,\mathbf{x}(k))^2$$

$$= y^2(k) + (\mathbf{a}^T(k)\mathbf{x}(k))^2 - 2y(k)\mathbf{a}^T\mathbf{x}(k) \qquad (2.9)$$

Assuming $\mathbf{x}(k)$ and $y(k)$ are wide-sense stationary we can write

$$e^2(k) = y^2(k) + (\mathbf{a}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{a}(k)) - 2\,\mathbf{a}^T(k)\phi_{xy}(k)$$

$$= y^2(k) + \mathbf{a}^T(k)\,\Phi_{xx}(k)\,\mathbf{a}(k) - 2\,\mathbf{a}^T(k)\phi_{xy}(k) \qquad (2.10)$$

where :

$\Phi_{xx}(k) = (\mathbf{x}(k)\ \mathbf{x}^T(k))$ is the (N x N) autocorrelation matrix.

$\phi_{xy}(k) = (\mathbf{x}(k)\ y(k))$ is the (N x 1) cross correlation matrix.

The mean-squared error (MSE) is :

$$\epsilon(k) = E\,[e^2(k)]$$

$$= E\,[y^2(k)] + \mathbf{a}^T(k)\,\Phi_{xx}(k)\,\mathbf{a}\,(k) - 2\,\mathbf{a}(k)\,\phi_{xy}(k) \qquad (2.11)$$

The MSE function has a quadratic form in the estimated filter coefficients $\mathbf{a}$, and the minimum can be obtained by setting the gradient ∇ to zero.

$$\nabla(k) = \partial\epsilon(k)/\partial\mathbf{a}(k) = [\partial\epsilon/\partial a_0(k),\ \partial\epsilon/\partial a_1(k),\ldots,\ \partial\epsilon/\partial a_{N-1}(k)]$$

$$= 2\,\Phi_{xx}(k)\,\mathbf{a}(k) - 2\,\phi_{xy}(k) \qquad (2.12)$$

The optimum tap weight vector which minimizes the MSE is the solution to a set of N simultaneously linear equations called Wiener equation :

$$\Phi_{xx}(k)\ \mathbf{a}_{opt}(k) = \phi_{xy}(k) \qquad (2.13)$$

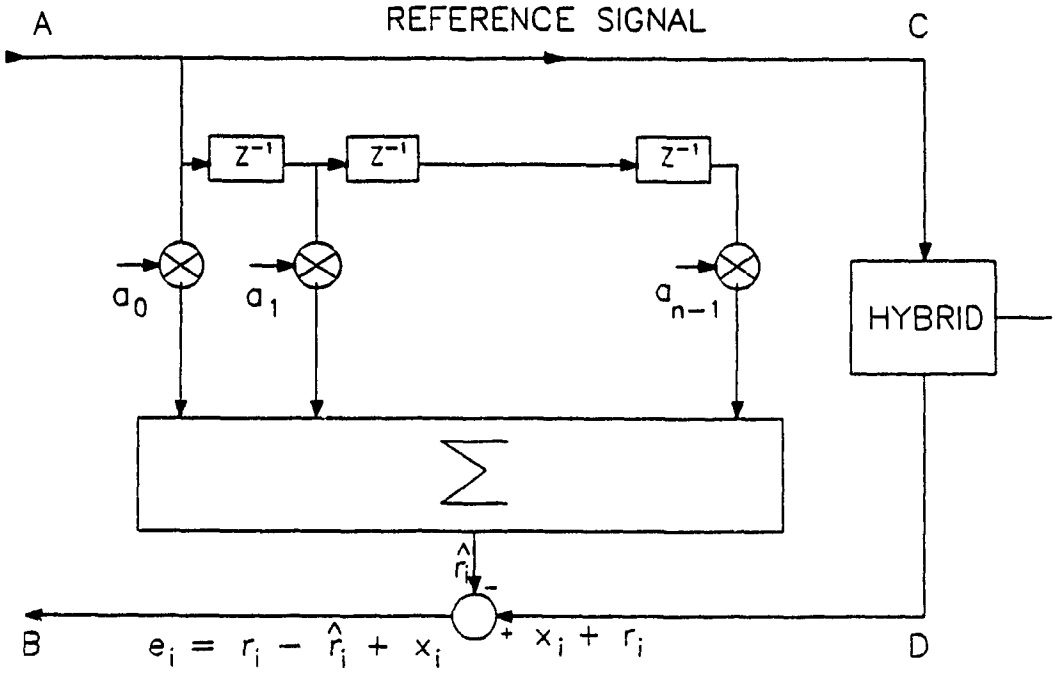


Figure 6. Signal Processing Model of Echo Canceller

The Wiener equation (also called a normal equation) can be solved by using either Gaussian or Levinsion recursion where a fixed number of calculations is required [5]. As shown in equation (2.13) the computation of the optimum tap-weight vector $\mathbf{a}_{opt}(k)$ requires knowledge of two quantities :

(i) The autocorrelation of the input signal.

(ii) The cross correlation of the input signal, and the desired response.

Solution for obtaining the optimum tap-weight vector using this method is computationally difficult, especially when the number of taps is large. Adaptive technique based on steepest-descent method can also be employed to solve this equation. The steepest descent algorithm obtain the optimum tap-weight vector by using the following steps :

(i) Make an initial guess of the tap-weight vector $\mathbf{a}(0)$.

(ii) Compute the *gradient vector* ∇ as expressed in equation (2.12).

(iii) Update the tap-weight vector by changing the present value of the tap-weight vector in negative direction of the *gradient vector*.

The steepest descent method is represented by :

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \mu \nabla(k+1) \qquad (2.14)$$

(iv) Go back to step (ii), and repeat the process.

The vector $\mathbf{a}(k+1)$ and $\nabla(k+1)$ represent the estimated coefficients of the optimum tap-weight vector, and the gradient at sample time k+1, respectively. The gradient $\nabla(k)$ as represented in (2.12) can be expressed in different form as

$$\begin{aligned}\nabla(k) &= \delta E\,[(y(k) - \mathbf{a}(k)\,\mathbf{x}(k))^2] \,/\, \delta \mathbf{a} \\ &= \delta E\,[y(k)^2 + \mathbf{a}^T(k)\,\mathbf{x}(k)\,\mathbf{x}^T(k)\,\mathbf{a}(k) - 2y(k)\mathbf{a}^T(k)\mathbf{x}(k)] \,/\, \delta \mathbf{a} \\ &= -2E\,[\,\mathbf{x}(k)\,(y(k) - \mathbf{a}^T(k)\,\mathbf{x}(k))] \qquad (2.15)\end{aligned}$$

The LMS recursive equations are :

$$e(k+1) = y(k+1) - \mathbf{a}^T(k)\,\mathbf{x}(k+1) \qquad (2.16)$$

$$\mathbf{a}(k+1) = \mathbf{a}(k) + 2\mu\, \mathbf{x}(k+1)\, e(k+1) \qquad (2.17)$$

2.3.1.2 Convergence

The LMS algorithm requires that the estimated impulse response $\mathbf{a}(k)$ reaches optimum $\mathbf{a}_{opt}$ after a number of iterative processes.

Substituting $e(k+1) = y(k+1) - \mathbf{a}^T(k)\, \mathbf{x}(k+1)$ into (2.17) :

$$\mathbf{a}(k+1) = \mathbf{a}(k) + 2\mu\, \mathbf{x}(k+1)\, (y(k+1) - \mathbf{a}^T(k)\, \mathbf{x}(k+1)) \qquad (2.18)$$

Taking expected values of both sides of (2.18) gives :

$$E[\mathbf{a}(k+1)] = E[\mathbf{a}(k) + 2\mu\, \mathbf{x}(k+1)\{y(k+1) - \mathbf{a}^T(k)\mathbf{x}(k+1)\}] \qquad (2.19)$$

$$= E[\mathbf{a}(k) + 2\mu\, \mathbf{x}(k+1)y(k+1) - 2\mu\, \mathbf{x}(k+1)\, \mathbf{a}^T(k)\, \mathbf{x}(k+1)]$$

$$= E[\mathbf{a}(k)]\, [(I - 2\mu\mathbf{x}(k+1)\mathbf{x}^T(k+1)] + 2\mu E[\mathbf{x}(k+1)y(k+1)]$$

$$= (I - 2\mu \mathbf{\Phi}_{xx}(k))\, E[\mathbf{a}(k)] + 2\mu \boldsymbol{\phi}_{xy}(k) \qquad (2.20)$$

Assuming the input signal, and the desired response are uncorrelated :

$$\boldsymbol{\phi}_{xy}(k) \approx 0$$

Equation (2.20) is simplified to :

$$E[\mathbf{a}(k+1)] = E[\mathbf{a}(k)]\, (I - 2\, \mu\, \mathbf{\Phi}_{xx}(k))$$

Defining the weight vector error at time k as :

$$\tilde{\mathbf{a}}(k) = \mathbf{a}(k) - \mathbf{a}_{opt} \qquad (2.21)$$

where :

$\mathbf{a}(k)$ is the estimated tap-weight value at time k.

a_{opt} is the optimum tap-weight value.

So :

$$E[\tilde{a}(k+1)] = E[\tilde{a}(k)]\ (I - 2\ \mu\ \Phi_{xx}(k)) \qquad (2.22)$$

The autocorrelation can be rewritten as :

$$\Phi\ (k) = Q\ \Lambda\ Q^{T} \qquad (2.23)$$

where :

Q is the unitary matrix which has a property that its transpose is equal to its inverse, and whose columns are eigenvectors of the autocorrelation Φ (k).

Λ is the diagonal matrix whose diagonal elements are eigenvalues of the autocorrelation Φ (k). These eigenvalues are all positive, and real.

Substituting (2.23) in (2.22) :

$$E[\tilde{a}(k+1)] = E[\tilde{a}(k)](I - 2\mu\ Q\ \Lambda\ Q^{T}) \qquad (2.24)$$

To simplify the notation let ignore the expectation operator.

Multiply both sides of (2.24) by Q :

$$Q^{T}\ \tilde{a}(k+1) = Q^{T}(I - 2\mu\ Q\ \Lambda\ Q^{T})\ \tilde{a}(k) \qquad (2.25)$$

Using property of the unitary matrix :

$$Q^{T}\tilde{a}(k+1) = (I - 2\mu\ \Lambda\ )Q^{T}\tilde{a}(k) \qquad (2.26)$$

Define :

$$v(k) = Q^{T}\tilde{a}(k)$$

Equation (2.26) can be rewritten as :

$$v(k+1) = (I - 2\mu \lambda_k)\ v(k) \qquad (2.27)$$

By assuming an initial value of tap-weight vector as zero, and deducting (2.27) we obtain the general expression for v at time k is :

$$v_n(k+1) = (1 - 2\mu \lambda_n)^k\ v_n(k) \qquad (2.28)$$

This has the form of a geometric series with a geometric ratio of $(1-2\mu\lambda_n)$.

For stability or **convergence** of the algorithm it is required that $-1 < 1 - 2\mu\lambda_n) < 1$, or

$$0 < \mu < 1/\lambda_{max} \qquad (2.29)$$

where λ_{max} is the largest eigenvalue of the autocorrelation

2.3.2 Least Squares (LS) algorithm :

LS algorithm minimizes sum of squared errors. Both LS's derivation, and convergence behavior are given.

2.3.2.1 Derivation of LS algorithm :

In Fig.5 the residual error at sampling time k is expressed as

$$e(k) = y(k) - \sum_{i=0}^{N-1} a(i)x(k-i-1) \qquad (2.30)$$

where a(i)'s are unknown parameters of the model, and y(k) is

the desired response. In a shorter form the above equation can be rewritten as :

$$e(k) = y(k) - \hat{y}(k) \tag{2.31}$$

with :

$$\hat{y}(k) = \sum_{i=0}^{N-1} a(i)x(k-i-1)$$

The LS method minimizes sum of squared errors. In other words, we choose the tap weights a(k)'s such that the sum of squared errors is minimum.

$$\sum_{i=i1}^{i2} |e(i)|^2 \rightarrow \epsilon \tag{2.32}$$

In equation (2.32) the i1, and i2 define the index limits at which the error minimization occurs. The values of i1 & i2 depend on the type of windowing methods. There exists four different methods of windowing [5].

(a) Covariance Method: which makes no assumption about the data outside the interval (i,M) with i1 = N, i2 = M, where N is the number of tap weights, and M is any arbitrary number such that M ≥ N. The input data can be arranged in the matrix form as :

$$\begin{matrix} x(N-1) & x(N) & \ldots\ldots & x(M) \\ x(N-2) & x(N-1) & \ldots. & x(M-1) \\ .. & .. & .... & .. \\ x(1) & x(2) & .... & .. \\ x(0) & x(1) & & x(M-N+1) \end{matrix}$$

(b) Autocorrelation Method which makes the assumption that data prior to time i = 1, and the data after i = M are zero. By using i1 = 1, and i2 = M+N-1 the input data can be taken on the form :

x(1)	x(2)	...	x(N)	...	x(M)	.... 0	0
0	x(1)	...	x(N-1)	..x(M-1)	..x(M)	.....	0
..	0	...	..	..	..		0
0	0		x(1)	x(M-N+1)	x(M-N)		x(M)

(c) Prewindowing Method which makes the assumption that the input data prior to i = 1 are zero but makes no assumption about the data after i = M. The matrix of input data takes on the form :

x(1)	x(2)	.....	x(N)	x(N+1)	.....	x(M)
0	x(1)	.....	x(N-1)	x(N)	.....	x(M-1)
0	0	.....	..	..		..
..	..	.....	..	..		..
0	0		x(1)	x(2)		x(M-N+1)

(d) Postwindowing Method which makes no assumption about the data prior to time i = 1, but assumes that the data after i = M are zero in the following form :

x(N)	x(N+1)	....	x(M)	0		0
x(N-1)	x(N)	....	x(M-1)	x(M)		0
...	..		...	..		0
x(1)	x(2)		x(M-N+1)	x(M-N)	...	x(M)

The derivation of the LS equation is carried out using the covariance windowing method. Other methods are given in [5]. As previously mentioned, the problem we have to solve is to determine a set of tap weights a(k)'s for which the sum of squared errors is minimum.

The residual error is :

$$e(k) = y(k) - \mathbf{a}^T(k)\, \mathbf{x}(k) \tag{2.33}$$

By expressing residual error e(k), and the desired response y(k) as elements of (M-N+1) by 1 vectors as follows :

$$\epsilon^T = [e(N),\ e(N+1),\ldots,\ e(M)] \quad , \text{ where } M \geq N \tag{2.34}$$

$$\mathbf{b}^T = [y(N),\ y(N+1),\ldots,\ y(M)] \tag{2.35}$$

then equation (2.33) can be rewritten as :

$$\epsilon^T = \mathbf{b}^T \cdot \mathbf{a}^T\{x(N),\ x(N+1),\ldots,\ x(M)\}$$

$$\epsilon^T = \mathbf{b}^T - \mathbf{a}^T\mathbf{A} \tag{2.36}$$

where :

$$\mathbf{A}^T = [x(N),\ x(N+1),\ldots,\ x(M)]$$

Applying the Hermitian property of equation (2.36)

$$\epsilon = \mathbf{b} - \mathbf{a}\,\mathbf{A} \tag{2.37}$$

Expressing equation (2.32) in the other form using matrix representation of residual error e(k) :

$$\sum_{i=N}^{M} |e(k)|^2 = \epsilon^T \epsilon \tag{2.38}$$

Substituting equations (2.36), and (2.37) in (2.38) yields :

$$\sum_{i=N}^{M} |e(k)|^2 = (\mathbf{b}^T - \mathbf{a}^T\mathbf{A}^T)(\mathbf{b} - \mathbf{A}\,\mathbf{a}) \qquad (2.39)$$

$$= \mathbf{b}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\,\mathbf{a} - \mathbf{a}^T\mathbf{A}^T\mathbf{b} + \mathbf{a}^T\mathbf{A}^T\mathbf{A}\,\mathbf{a}$$

Differentiating $\Sigma\,|e(k)|^2$ with respect to $\mathbf{a}$:

$$\delta\xi/\delta\mathbf{a} = -2\mathbf{A}^T\mathbf{b} + 2\mathbf{A}^T\mathbf{A}\,\mathbf{a} \qquad (2.40)$$

$$= -2\mathbf{A}^T\{\mathbf{b} + \mathbf{A}\,\mathbf{a}\}$$

$$= -2\mathbf{A}^T\epsilon \qquad (2.41)$$

As before, denoting $\mathbf{a}_{opt}$ as the optimum tap weight vector for which the sum of error squares is minimum, i.e $\delta\xi/\delta\mathbf{a} = 0$. It follows :

$$\mathbf{A}^T\mathbf{A}\,\mathbf{a}_{opt} = \mathbf{A}^T\mathbf{b} \qquad (2.42)$$

Recall that $\mathbf{A}$ and $\mathbf{b}$ represent the inputs of the transversal filter, and the desired response, respectively. From the definitions of cross-correlation and autocorrelation equation (2.42) can be expressed in the following form :

$$\Phi_{xx}\,\mathbf{a}_{opt} = \Phi_{xy} \qquad (2.43)$$

Equation (2.43), which represents the LS estimate, is analogous to the Wiener equation as in the LMS algorithm. In many applications, we have to reconstruct (2.40), and to resolve it as new data becomes available. This approach is computionally expensive since matrix calculation is involved. For practical reasons, a Recursive Least Squares (RLS) algorithm is often used. In RLS the tap weight at sample time

k is recursively estimated based on previous tap value at time (k-1), and the new data x(k) and y(k).

A weighting factor is usually introduced into the sum of squared errors of equation (2.38). This factor is to fade out the past data, and one form of weighting used is the exponential function defined by :

$$w(k,i) = \lambda^{k-i} \quad , \qquad i = 1, 2, \ldots, k$$

where $0 < \lambda < 1$.

Equation (2.38) is rewritten with the introduction of the weighting factor as :

$\Sigma\, w(k,i)|e(k)^2|$

Correspondingly, we define :

$$\Phi_{xx}(k) = \sum_{i=1}^{k} \lambda^{k-i}\, \mathbf{x}(i)\, \mathbf{x}^T(i) \tag{2.44}$$

$$\phi_{xy}(k) = \sum_{i=1}^{k} \lambda^{k-i}\, \mathbf{x}(i)\, y(i) \tag{2.45}$$

Taking the term i = k out of the summation, equation (2.44) can be rewritten as :

$$\Phi_{xx}(k) = \lambda\Big[\sum_{i=1}^{k-i} \lambda^{k-1-i}\, \mathbf{x}(i)\mathbf{x}^T(i)\Big] + \mathbf{x}(k)\mathbf{x}^T(k) \tag{2.46}$$

By definition in equation (2.46) the first term on the right side is equal to $\Phi_{xx}(k-1)$.

So :

$$\Phi_{xx}(k) = \lambda\Phi_{xx}(k-1) + \mathbf{x}(k)\mathbf{x}^T(k) \tag{2.47}$$

Similarly :

$$\phi_{xy}(k) = \lambda\phi_{xy}(k-1) + \mathbf{x}(k)y(k) \qquad (2.48)$$

Using the LS estimate in equation (2.43) we can obtain a recursive formula for the tap-weight vector.

$$\mathbf{a}(k) = \Phi_{xx}^{-1}(k)\phi_{xy}(k)$$

To compute the inverse of the autocorrelation we have to use the matrix inversion formula as defined by :

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^{T}$$

Relating **A**, **B**, **C**, **D** with th terms in equation (2.47)

$$\mathbf{A} = \Phi_{xx}(k)$$

$$\mathbf{B}^{-1} = \lambda\Phi_{xx}(k-1)$$

$$\mathbf{C} = \mathbf{x}(k) \qquad \mathbf{D} = 1$$

By substituting the above relations in the matrix inversion lemma we obtain the inverse of the autocorrelation as :

$$\Phi_{xx}^{-1}(k) = \lambda^{-1}\Phi_{xx}^{-1}(k-1) - \lambda^{-1}\Gamma(k)\mathbf{x}(k)\Phi_{xx}^{-1}(k-1) \qquad (2.49)$$

where :

$$\Gamma(k) = \{\lambda^{-1}\Phi_{xx}^{-1}(k-1)\mathbf{x}(k)\}/\{1+\lambda^{-1}\mathbf{x}^{T}(k)\Phi_{xx}^{-1}(k-1)\mathbf{x}(k)\} \qquad (2.50)$$

Substituting (2.49) in the LS estimate, and using (2.48) we obtain :

$$\mathbf{a}(k) = \Phi_{xx}^{-1}(k)\phi_{xy}(k)$$

$$= \lambda\Phi_{xx}^{-1}(k)\phi_{xy}(k-1) + \Phi_{xx}^{-1}(k)\mathbf{x}(k)y(k) \qquad (2.51)$$

After manipulating by using the definition of $\Gamma(k)$ as in equation (2.50) we can express the tap-weight updation in a similar form as the LMS algorithm :

$$\mathbf{a}(k) = \mathbf{a}(k-1) + \Gamma(k)\alpha(k) \tag{2.52}$$

where :

$$\alpha(k) = y(k) - \mathbf{a}^T(k-1)\mathbf{x}(k) \tag{2.53}$$

Equations (2.50), (2.52), and (2.53) constitute the RLS algorithm.

2.3.2.3 Convergence

Three aspects of convergence have been extensively discussed in [5] :

(a) Convergence of the estimate $\mathbf{a}(k)$ in the mean.

(b) Convergence of the estimate $\mathbf{a}(k)$ in the mean square.

(c) Convergence of the average mean-squared value of the "a priori estimation error".

In the following analysis we discuss the third aspect of convergence based on the RLS algorithm, and then make some comparision with that of the LMS algorithm (this aspect is selected for discussion because it is analogous to the LMS algorithm).

Two errors : the "a priori estimation error" and the "a posteriori estimation error" are defined as :

$$\alpha(k) = y(k) - \mathbf{a}^T(k-1)\,\mathbf{x}(k) \tag{2.54}$$

$$e(k) = y(k) - \mathbf{a}^T(k)\,\mathbf{x}(k) \tag{2.55}$$

Initial conditions of RLS require that :

$$\Gamma(0) = \zeta^{-1}\mathbf{I} \tag{2.56}$$

$$\mathbf{a}(0) = 0 \tag{2.57}$$

where ζ is a small positive constant. For comparision between the convergence charateristics of the RLS and that of the LMS we discuss the "a priori estimation error" mean-squared value since its learning curve has the same shape as that of the LMS algorithm.

Substituting y(k) from equation (2.55) to equation (2.54) :

$$\begin{aligned}\alpha(k) &= e(k) + \mathbf{a}^T(k)\,\mathbf{x}(k) - \mathbf{a}^T(k-1)\,\mathbf{x}(k)\\ &= e(k) - [\mathbf{a}^T(k) - \mathbf{a}^T(k-1)]\,\mathbf{x}(k)\\ &= e(k) - \tilde{\mathbf{a}}^T(k)\,\mathbf{x}(k)\end{aligned} \tag{2.58}$$

where $\tilde{\mathbf{a}}(k) = \mathbf{a}(k) - \mathbf{a}(k-1)$ is the weight error vector at time k.

We are interested in obtaining the average of the mean-squared priori estimation error. Assuming that the measurement error e(k) has zero mean, and a variance of σ^2 the following steps are taken :

(i) Computing the priori estimation mean-squared error

$$E\,|\alpha^2(k)| = E[e^2(k)] - 2E[e(k)\,\tilde{\mathbf{a}}^T(k-1)\,\mathbf{x}(k)] + E[\mathbf{x}^T(k)\tilde{\mathbf{a}}(k-1)\tilde{\mathbf{a}}^T(k-1)\mathbf{x}(k)]$$

$$= E[e^2(k)] + E[\mathbf{x}^T(k)\ \tilde{\mathbf{a}}(k-1)\ \tilde{\mathbf{a}}^T(k-1)\ \mathbf{x}(k)]$$

$$= \sigma^2 + \mathbf{x}^T(k)\ E[\tilde{\mathbf{a}}(k-1)\ \tilde{\mathbf{a}}^T(k-1)]\ \mathbf{x}(k) \qquad (2.59)$$

(ii) Averaging the mean-squared error

Following steps shown in [5] the average mean-squared error produced by the RLS algorithm is :

$$E\ [\ E\ |\alpha^2(k)|\ ]\ \approx\ \sigma^2 + N\ \sigma^2 / n \qquad (2.60)$$

Some observations are made :

(a) As shown in equation (2.60) the RLS algorithm is independent of the charateristics of the input signal. Convergence rate is therefore insensitive to the input signal. The LMS algorithm convergence behavior is dependent on the eigenvalues of the input signal.

(b) The RLS algorithm convergence rate is of the order of 2N, where N is the number of the filter taps.

2.3.3 Modified algorithms

In addition to the LMS and LS algorithms, there exists also some modified algorithms whi h are developed to reduce complexity of either the LMS or the LS algorithm. In the LMS category we have the Sign algorithm (SA), and the Block Least Mean Square (BLMS). Similarly, the Fast Recursive Least Square (FRLS), and the Fast Transversal Filters (FTF) belong to the

LS class [5].

2.3.3.1 Sign Algorithm (SA)

A modified version of the LMS algorithm, the SA reduces number of computations by replacing the correlation $e(k+1)\mathbf{x}(k+1)$ of equation (2.14) by the sign of the error $e(k)$

$$\mathbf{a}(k+1) = \mathbf{a}(k) + 2\mu \, \text{sign}(e(k+1)) \, \mathbf{x}(k+1) \qquad (2.61)$$

This replacement results in a degradation of speed, and high instability [5].

2.3.3.2 Block Least Mean Square (BLMS)

Derived from the LMS algorithm, this BLMS approach updates a block of M coefficients per iteration. Equation (2.17) is modified as :

$$a(k+M) = a(k) + 2\mu \sum_{n=0}^{M-1} e(k-n)x(k-n-1) \qquad (2.62)$$

When M = 1 the BLMS becomes the LMS algorithm. The condition for convergence of the BLMS is the same as that of the LMS :

$$0 < \mu_B < 1/\lambda_{max}$$

2.3.3.3 Fast Recursive Least Square (FRLS) Algorithm

This modification of the RLS algorithm uses the shifting property of the $\Gamma(k)$ of equation (2.52) to avoid computing this matrix. This is done without any storage of the M x M matrix as required by the conventional form of the RLS algorithm.

2.3.3.4 Fast Transversal Filters (FTF) Algorithm

Developed by Cioffi and Kailath in 1984 [5] this FTF algorithm employs four transversal filters :

- one filter defines the desired response of the adaptive filter.
- the other three filters perform convergence process.

The FTF algorithm uses simple equations of the LMS, and yet its speed is comparable with that of the RLS.

CHAPTER 3

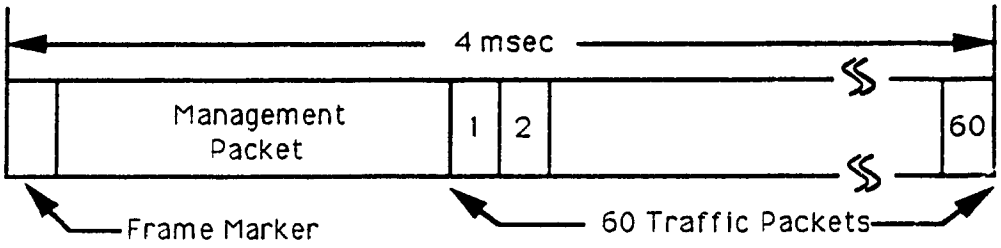
ECHO CANCELLATION IN DA, P-MP SYSTEMS

This chapter first presents the concept of echo cancellation suitable to Point-to-Multipoint, Demand-Assignment Time Division Multiple Access (DA-TDMA) radio systems, and its characteristics. It then discusses the criteria and performance evaluation to select the echo cancellation algorithm, and appropriate DSP suitable to implementation of a Multichannel, Demand-Assignment Echo Canceller (MCDAEC). A *statistical approach*, which is based on the behavior of trunks, is introduced to increase the capacity of trunks per DSP.

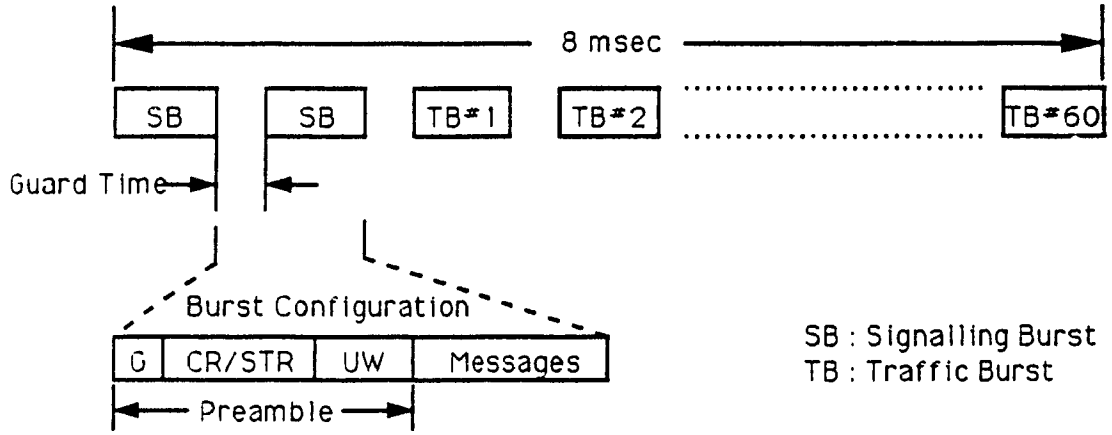
3.1 System Description

Previously we mentioned that all P-MP Subscriber Radio Systems share a common feature in having a Central Station (C/S), and a number of Outstations (O/S) communicated via microwave links. A typical P-MP subscriber radio system is shown in Fig.1. The Central station normally located in the Central office is equipped with an omnidirectional antenna. The Outstations (O/S's) may be placed in any location in the exchange serving area, and when a single hoop ranges are exceeded, repeaters are used. In this study we assume a typical system which can have up to 10 repeaters for a range

of 720 km. With N trunks, the system provides services to M subscribers on a Demand-Assignment basis, where M >> N. The transmission in the downward direction (from C/S to O/S) is in a Time Division Multiplexing mode with a 4ms frame. The transmission in the upward direction (from O/S to C/S) is a Time Division Multiple Access with a 8ms frame as shown in Fig.7.



Downward Frame Structure



Upward Frame Structure

Figure 7. A Time Division Multiplex Structure

Due to the nature of grouping samples in TDMA transmission delay is introduced in the system which may increase annoying effect of echo. Echo is present when there

is an impedance mismatch in O/S 2-wire loop cards (or C/S two wire line cards). There are two parameters which makes echo annoying :

- Level of echo due to mismatch at subscriber interface, and network terminal points.
- System round-trip delay.

Echo Cancellers can be introduced to each subscriber interface loopcards in the Outstation. In this case, the number of echo canceller will be M, the number of the subscribers, much larger than N, the number of available trunks in the system. This unnecessarily increase the system cost. A more economical approach is to implement Echo Cancellers in the Central station. In this case the maximum required number of echo canceller pairs is N. Fig.8 shows the model of a proposed Multichannel, Demand-Assignment Echo Canceller (MCDAEC).

When both Echo Cancellers are in the C/S, the echo paths include delay due to distance from echo canceller to the source of echo, propagation and frame delays. As shown in Fig.8, the echo path due to the mismatch impedance in the loopcard of the Outstation will be :

$$\tau_{EC} = \tau_B + \tau_P + \tau_R + \tau_I$$

where :

- τ_B is the sum of the upward TDMA frame, and the downward TDM frame, e.g for a system with an upward TDMA frame of 8ms

and a downward TDM frame of 4ms, the τ_B is 12ms.

• τ_P is round-trip propagation delay due to distance from the location of the Echo Cancellers to subscribers, e.g for a range of 720km :

$$\tau_P = 2 \times (720/300)$$

• τ_R is the total processing delay of the repeaters in tandem e.g for a link with 10 repeaters and individual repeater delay of 30 μs, τ_R is 0.3ms.

• τ_I is the impulse delay due to the spanning time of the echo. This delay is typically of the order of 2ms to 5ms. Measured result on a typical subscriber radio system shows a delay of about 2.9ms.

For values given in the above examples, the echo delay for this path is :

$$\tau_{EC} = (12ms) + (2 \times (720/300)) + (0.3ms) + (2.9ms)$$

$$= 20.0ms$$

This echo is handled by the Echo Canceller for direction from A to B.

• For echo due to mismatch impedance in Line Card at the Central station, its echo path delay is about 5ms. This echo is handled by the Echo Canceller for direction from C to D.

3.2 Design Requirements

The calculations of echo paths from the model in Fig.8 lead to the design of two Echo Cancellers with different delays. Also, dynamic operation of a trunk (a trunk can be assigned to any subscriber) requires a dynamic structure of Echo Canceller. In this section we discuss the system requirements, performance specifications using the model in Fig.8.

3.2.1 System Requirements

The design aim of the MCDAEC needs an Echo Cancellation algorithm which satifies the following requirements :

(a) Simple implementation : the P-MP system has N full-duplex trunks, hence needs N full-duplex Echo Cancellers required. It is economically desirable to multiplex all these N full-duplex E.C's in as few DSP's as possible. Common sense dictates that a simple algorithm is necessary.

(b) Small number of operations (multiplications & additions) : most consuming time operations in adaptive FIR filters are the multiplications and additions. To increase the real-time performance means number of multiplications and additions per iteration should be small.

(c) Fast tracking speed : trunk status dynamically

changes from one subscriber to another, so it is necessary that the algorithm has the ability to track the changes rapidly.

(d) Small memory utilization : external memory access is more time expensive than internal memory access. We can multiplex more echo canceller pairs per DSP if small external memmory is required, i.e a small memory utilization is desirable since most DSP's have small internal memory of the order of 512 words.

In previous chapter, we outlined various Echo Cancellation algorithms. The usefulness of these Echo Control techniques depends on each application. For application in Demand-Assignment, Point-to-Multipoint Radio System the areas of complexity in Echo Cancellation are :

- The speed of adaption, and the accuracy of the cancellation after adaption are two important measures of performance of an Echo Cancellation algorithm. A trade-off of the two measures are required : as the speed of adaption is increased (small number of taps used in FIR) the accuracy of the replica echo becomes poorer.

- The Demand-Assignment operation of the MCDAEC requires a "dynamic" structure of Echo Canceller. Since a trunk is not dedicated to any particular subscriber, the adaption must be dynamically changed on each new assignment, and possibly from one frame to another. The Echo Cancellation, therefore, has to be able to adapt to a new echo path with fast speed.

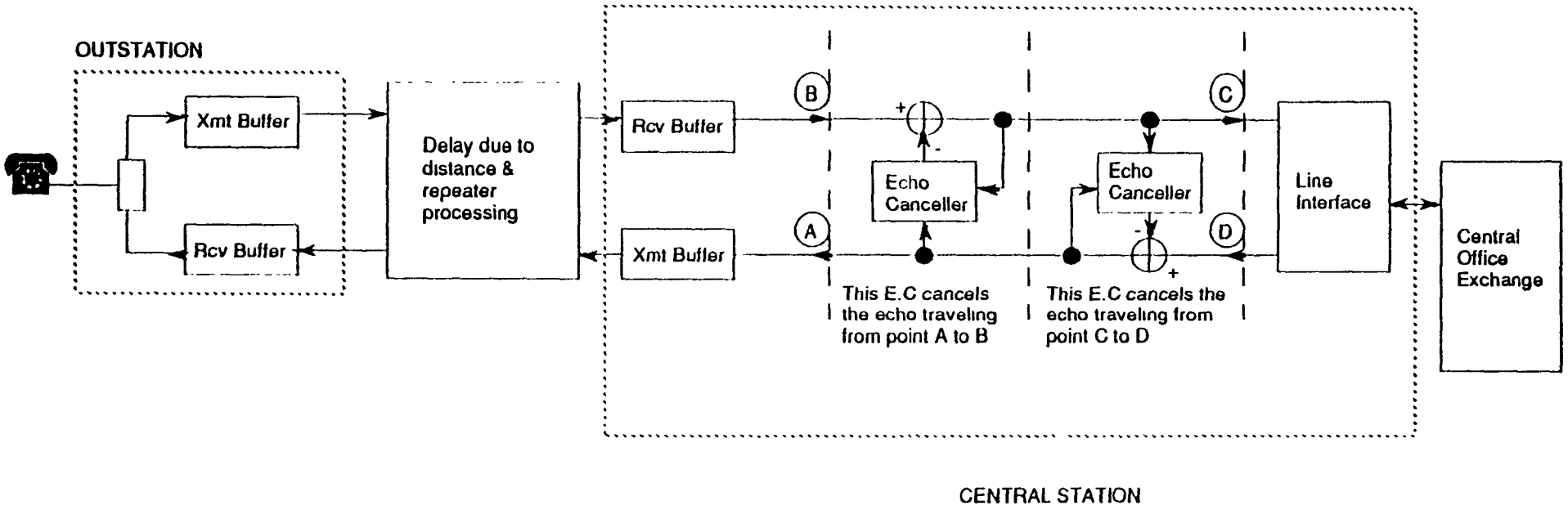


Figure 8. A Simpified model to illustrate the location of the MCDAEC in the Central station

3.2.2 Performance Requirements

The performance specifications for the design of the MCDAEC can be divided into two groups : the generic specifications, and the specific specifications.

- The generic specifications are based on the G.165 CCITT Recommendations [4]. These specifications are to provide the basic requirements for the design of an E.C with acceptable performance as follows :

  (a) Echo Return Loss Enhancement (ERLE) : should be smaller than or equal to -24dB with reference to input signal.

  (b) Convergence Speed : the ERLE should be $\leq$ -24 dB within 500ms with initial coefficients set to 0.

  (c) Real-time : PCM sampling time is 8 Khz, i.e the frame is 125 μs.

- The specific specifications are the requirements for the design of the MCDAEC. These specifications are listed in the following :

  (d) Echo Path Delay : two Echo Cancellers are require for each full-duplex trunk. One has echo delay of 20ms, and the other has echo delay of 5ms. Details on the design will be described in section 3.3.1.

  (e) A small number of DSPs : as discussed earlier, a P-MP system with N trunks needs N pairs of echo cancellers. For cost-effective reason, a minimization of the number

of DSPs could be significant, especially when N is large. If the number of DSPs is Q, then Q should be as small as possible compared to N.

Both the generic and specific specifications represent a challenge in searching for an E.C algorithm, and a powerful DSP. The generic specifications suggest an E.C algorithm which should have fast convergence speed, and small ERLE. The least squares E.C algorithms are known for their fast convergence speed. Yet, these algorithms are expensive to implement because they require a large number of operations per iteration. These are the trade-off performance that have to be made in selecting an E.C algorithm. In the following section, we revisit both the Least Mean Squared (LMS) algorithm, and the Least Squares algorithm with focus on the following :

- Implementation complexity (number of operations per iteration).
- Tracking speed.

3.3 Performance Evaluation of Echo Cancellation

The performance specifications for the design of the MCDAEC suggest for a selection criteria of echo cancellation algorithms. Echo cancellation algorithms should have fast convergence and tracking speeds. Since trunk is not dedicated to any particular subscriber, the adaption changes dynamically from on each new assignment. The echo canceller, therefore,

has to be able to adapt to new changes in the echo path with sufficiently fast speed. Furthermore, simple structure of echo cancellation algorithm is desirable due to the objective of minimizing the number of DSPs used in the MCDAEC. Simplicity means a small number of operations (multiplication & additions) per iteration, a small utilization of memory.

Echo cancellation algorithms are available in two basic categories : Least Mean Squared (LMS), and Least Squares (LS). In each category there also exists a number of modified algorithms such as Block Least Mean Square (BLMS) in the LMS category, or Recursive Least Square in the LS category.

3.3.1 Least Mean Square (LMS) Algorithm :

3.3.1.1 General Algorithm

The LMS is the most commonly implemented algorithm in many Echo Cancellers. The criterion function is taken to be the expected valued of the squared error, and the taps are adapted according to the stochastic steepest descent algorithm [6]. As previously discussed in chapter 2 the steepest-descent equations are

$$e(k) = y(k) - x(k) * h(k) \qquad (3.1)$$

$$a\ (k+1) = a\ (k) + 2\mu \times e(k+1)\ x\ x(k+1) \qquad (3.2)$$

The LMS algorithm can be formulated in the following steps :

(i) Initializing : filter coefficients are set to zero,

and initial parameters are declared.

(ii) Inputting : new samples are read.

(iii) Echo Estimating : the echo replica is estimated, and residual error is computed using Eq. (2.16). The residual error is compared with optimum threshold value. If the optimum residual error has been achieved, go to step (v).

(iv) Coefficient Updating : coefficients are updated using the current far-end sample and residual error as shown in equation (2.14).

(v) Go to step (i).

3.3.1.2 Complexity : the steepest-descent equations require only O(2N) multiplications, and O(2N) additions per iteration for an N-tap filter, where O(J) stands for "on the order of J". In addition, a memory size of 2 x N words is required (N most recent far-end samples, and N tap coefficients). With the currently available DSPs having small memory size (on the order of 512 words) this LMS seems to suit just well. Furthermore, for some filters where a large memory is required the number of memory accesses per iteration should be small so that real-time execution performance is not deteriorated due to number of wait states being introduced.

3.3.1.3 Tracking Speed: the value of μ in the above steepest-descent equations plays an important role in determining the convergence speed, stability, and residual

error after convergence. For a large value of μ, the convergence becomes faster, but it results in a larger residual error and is more prone to instability. The convergence speed of the LMS is considerably slow, and is dependent on the characteristics of the far-end signal [7]. To overcome this problem the loop gain μ is allowed to vary inversely with signal power. As the power in the correlation product k)x(k) is proportional to signal variance, the choice of a loop gain μ that is inversely proportional to the variance provides a performance which is invariant to far-end signal power. In the case of the MCDAEC the echo path is dynamically changed since a trunk is not dedicated to any particular subscriber. For a new echo path the Echo Canceller has to take some times to learn the chararteristics of the new echo path, a slow tracking speed may cause an undesirable divergence.

3.3.2 The Block Least Mean Square (BLMS) :

The BLMS is another class of Echo Cancellation in the LMS category. It involves the calculation of a block of finite set of outputs from a block of inputs values [8]. As before we are interested in the following characteristics :

3.3.2.1 Complexity : the BLMS algorithm is almost identical to the LMS one with respect to the implementation simplicity. The BLMS also requires 2N multiplications, and 2N

additions per iteration. Because of its structure (a block of coefficients are updated per iteration) the BLMS requires a larger memory storage by M words, where M is the block size of the BLMS. However, the memory access becomes less in the BLMS which does not need to update coefficients as often as in the case of the LMS. For the MCDAEC application this performance of the BLMS is equivalent to that of the LMS.

3.3.2.2 Tracking Speed: as shown by G.A.Clark, et al. [9] the speed of the BLMS is a function of block size M, and the loop gain μ. Simulation results [10] also show that a BLMS using a block size of 16 and inversely proportional to far-end signal power loop gain performs better than the LMS algorithm under the same conditions.

3.3.3 The Sign Algorithm (SA)

This algorithm also belongs to the LMS category. The same principle as the previous two, but the correlation factor is avoided by using the signs of e(k), and x(k). Hence, the SA is simpler, and less expensive. Because of its unnormalized gain the SA convergence speed is slower. The following observations are obtained when we try to increase the convergence speed :

- If the gain is adjusted to give an acceptable residual error, the convergence speed is very slow.
- If the gain is increased for fast convergence, the residual error may not be acceptable [11].

3.3.4 Least Squares (LS) Algorithm

3.3.4.1 General Algorithm

Instead of minimizing the mean-squared-errors, this approach minimizes a weighted sum of squared errors. A more complete details have been discussed in chapter 2. Similar equations to the LMS have been obtained as follows :

$$e(k) = y(k) - \mathbf{a}^T(k)\mathbf{x}(k) \tag{3.3}$$

$$a_n(k+1) = a_n(k) + \sigma R^{-1}(k)\ x(k)\ e(k) \tag{3.4}$$

$$R(k) = (1 - \sigma)R(k-1) + \sigma x(k) x^T(k) \tag{3.5}$$

The LS algorithm can be formulated as follows :

(i) Initializing : filter coefficients are set to zero, and R^{-1} is initialized.

(ii) Inputting : new samples are read.

(iii) Echo Estimating : echo replica is generated, and residual error is computed, and compared with optimum threshold value. If an optimum residual error has been obtained, go to step (v).

(iv) Coefficient Updating : two steps are involved :

- computing the matrix $\Gamma(k)$.
- updating the coefficients.

(v) Go to step (i).

3.3.4.2 Complexity : the correlation term in the case involves matrix multiplications, and additions as compared to the LMS approach which is simpler because of scalar

correlation. The direct implementation of the above equations is very costly, as the computation of $\Gamma(k)x(k)$ requires more than N3/6 multiplications per iterations, where N is the size of the FIR.

3.3.4.3 Tracking Speed: the normalization with $R^{-1}(k)$ in the LS offers more than the simple power normalization (varying the loop gain inversely with signal power) in LMS : it essentially normalizes the adaption in each eigen vector direction by the signal power. Thus the convergence becomes independent of both signal type, and power.

This algorithm requires a large amount of computations. A number of modified algorithms have been devised to reduce the required number of multiplications and divisions per iteration to be proportional to N.

3.3.5 Fast Recursive Least Square (FRLS)

The FRLS Algorithm is governed by the equation :

$$a_n(k+1) = a_n(k) + K_n(k)\ e(k) \tag{3.6}$$

$$K_n(k) = R^{-1}(k)\ x(k) \tag{3.7}$$

3.3.5.1 Complexity : this algorithm exploits the shifting property to compute the vector $K_n(k)$ recursively [12]. Therefore, it avoids the computation and storage of N by N matrix, i.e the vector x acts like a shift register such that x(k+1) is only a "push down" version of x(k) with a new sample

on top. The number of multiplications, and additions is then reduced to about O(10N).

3.3.4.2 Tracking Speed : as the LS approach this FRLS Algorithm offers a fast start-up convergence, independent of the characteristics of input signal.

Other improved algorithms have been introduced in [5]. Commonly, faster algorithms are obtained by reducing the number of multiplications, divisions, and additions per iteration. For instance, Memory-Tap Algorithm [13] uses table look-up method. The problem with this algorithm in the MCDAEC is that a very large memory is required. Several other authors [14] also attempted to improve real-time performance of E.C by using the fact that an impulse response consists of three parts : Flat delay, Active delay, and Tail delay. The Active part, if correctly detected, can help in using less number of taps in FIR, hence the computation is reduced.

As previously mentioned, the MCADEC requires a simple, but fast Echo Cancellation algorithm. The LS, Fast Recursive Least Square,... offers very fast start-up convergence, but their complexity is not attractive. On the other hand, LMS and BLMS is slow (but simulation results show their convergence speeds conform with the CCITT G.165 recommendation), and simple. The BLMS has an advantage in the reduction of the number of memory accesses per iteration.

Table 1. A Comparison of Echo Cancellation Algorithms

ALGORITHM	BASIC THEORY	No. of oper. per iteration	MEMORY STORAGE	MEMORY ACCESS
Least Mean Squared LMS	$e(k) = y(k) - \Sigma\, h\,(k)x(k-n)$ $a\,(k+1) = a\,(k) + 2\mu e(k)x(k-n)$	2N Mult 2N Add	•Store N most recent far-end samples. •Store N updated coefficients.	Appr. 3N per iteration
Block Mode Update BLMS	$e(k) = y(k) - \Sigma\, h\,(k)x(k-n)$ $a(k+M) = a(k) + 2\mu\Sigma\, e(k-1)x(k-n-1)$	2N Mult 2N Add	•Store M most recent err. est. •Store N most recent far-end samples. •Store N updated coefficients.	Appr. 2N+N/M per iteration
Recursive Least Square	$e(k) = y(k) - \Sigma\, h(k)x(k-n)$ $a\,(k+1) = a\,(k) + \Gamma(k)x(k)$	10N + 4 Mult 12N + 5 Add	•A total storage of 5N + 6 words.	Appr. 8N

3.4 Performance Evaluation of DSPs

One of the most time-consuming computation in E.C algorithm is the convolution, and correlation which involve multiplications and additions. Digital Signal Processors (DSPs) are specialized microcomputers suitable for these real-time digital signal processing applications. The most basic architecture is the integration of fast multiplier/accumulator hardware into the data path. Arithmetic operations are not done on a separate co-processor. They are integrated into basic instructions.

The selection of a powerful DSP is based on the specifications as discussed in section 3.2.2. The key performance of an DSP is its Multiply-Accumulate-Shift speed. Most of DSPs can perform these three operations in a single instruction cycle. The next performance criteria is DSP's internal memory capacity. This criteria is also important for the design of the MCDAEC which would multiplex echo canceller pairs per DSP. The real-time performance will be degraded if extensive external memory accesses are required (external memory access requires more real-time). In addition, the DSP multiprocessing configuration is also taken into consideration in designing the MCDAEC. These criteria of DSP selection are summarized below :

(i) Multiply-Accumulate-Shift.

(ii) Instruction Cycle.

(iii) Memory.

(iv) Multiprocessing configuration.

In addition to the above requirements, the following engineering aspects are also considered :

(v) Cost (unit, development system).

(vi) Engineering support.

The following discussion is based on the MCDAEC with specifications as :

- 64 x 64 taps for a full-duplex Echo Canceller.
- 160 far-end samples to be stored (representing 20ms).

In the remaining of this section, we will review the architectures of the MSM6992 (Oki Electric), the Nippon Electric μPD7720, μPD77230, the AT & T WEDSP16, WEDSP32, the Texas Instrument series TMS32020, TMS320C25, TMS320C30, and the Motorola DSP56000/01 with focus on each DSP's Multiply-Acumulate-Shift benchmark, Instruction Cycle, on chip memory, and the feasibility of supplementing this memory with external RAM, and DSP's peripheral using the Host architecture concept.

3.4.1 The Oki Electric Family

One member of this family, MSM6992, is described here. This DSP consists of four blocks : Arithmetic Block (AB), Data Memory Block (DMB), I/O Block (IOB), and the Control Block (CB). As a high speed floating point DSP the MSM6992

features :

- A ROM of 1024 words x 32 bits within the chip, and can be expanded to a maximum of 64K words x 32 bits by an external device.
- A data RAM of 256 words x 32 bits within the chip, and an external memory of a maximum of 64K words x 32 bits can be provided outside the chip.
- A fast instruction cycle of 100ns.
- A CMOS technology, 400mW.
- Multiprocessor interface.

The Data Memory Block (DB) of the MSM6992 is partitioned into two parts : RAMX and RAMY, each has 128 words x 22 bits RAMX, RAMY can be used independently, or as a single page of memory. In the latter case it is called RAMXY. The Multiply-Accumulate is performed by a floating-point multiplier FMPY. With this arrangement of data it is very effective for Echo Canceller application where coefficients, and far-end signal can be stored in RAMX, and RAMY, and the convolution can be carried using a single instruction. The MCDAEC utilizes a large memory since a number of trunks shares the same DSP. External memory can be used at the expense of increasing the processing time. This DSP employs an effective Host design concept. The following modes are provided as the interface modes :

(i) Slave mode : in this mode the Host processor accesses the I/O register of the DSP.

(ii) Master mode : in this mode the DSP itself, when instructed, accesses the external data memory, and I/O port.

(iii) DMA mode : this mode performs the interface with the external 8-bit or 16-bit DMA controller [16].

The Slave mode would be very useful in the design of MCDAEC where of number of DSP's can be controlled by one Host processor which communicates with the main CPU of the TDMA controller of the Central station. Although the I/O interface mode of the MSM6992 is suitable in the design of the MCDAEC, its limited on-chip memory, and little engineering support make it unattractive.

3.4.2 The Nippon Electric Family

As a second generation DSP the NEC μPD7720 introduced in 1980 has a microcode-like instruction set and a parallel architecture which enable a single instruction to load the two multiplier inputs, accumulate the multiplied product, modify both RAM/ROM pointers, and execute a return from subroutine. The NEC μPD7720 has on chip :

- a 512 x 23-bit program ROM.
- a 510 x 13-bit data coefficient ROM.
- a 128 x 16-bit data RAM.
- a 250 nsec 16 x 16-bit parallel multiplier which gives a 32-bit result.

Although the NEC μPD7720 has several characteristics in common with the MSM6992, its low speed is a major problem in MCDAEC application where the real-time requirement is crucial. NEC μPD77230, a third generation, offers a high speed (150ns) and a larger memory storage : 1K x 32-bit RAM which is organized as two separately addressable 512 x 32-bit blocks. The Multiply- Accumulate operation can be accomplished by arranging coefficients and far-end samples in two blocks. However, a significant amount of overhead is required in manipulating pointers when performing the FIR computation [17].

3.4.3 The AT & T Family

The WEDSP16, a member of the AT&T DSP, is a general purpose, programmable, 16-bit fixed point DSP. It features :

- a 2K x 16-bit ROM, which can be expanded off-chip to 64K x 16 bits with no loss of speed.
- a 512 x 16-bit on-chip RAM.
- a 15 x 16-bit on-chip Cache which can be repeated up to 127 times. The advantage of the Cache is that it allows low programming overhead for looping (1 cycle of overhead for up to 127 repeats).
- a Serial I/O port, and a Parallel I/O port.

The Multiply-Accumulate instruction has three stages in the pipeline : data fetch, multiply, and accumulate. A typical

example to illustrate this instruction as follows [18] :

Inst (1) y = *r1++ x = *pt++

Inst (2) p = x*y

Instr (3) a0 = a0+p

In instruction 1, the data in the RAM location indexed by register r1 is moved to the y register, and r1 is incremented by one. Similarly, the data in the ROM location pointed by register pt is moved to x register, and pt is then incremented by one memory location. The product of contents of x and y registers is stored in p, and added to accumulator a0. The DSP16 instruction syntax follows the C programming language. Its 55ns per instruction, and a structured architecture provides users with an efficient design. However, as will be discussed later, the WEDSP16 is not well-supported and cost-effective compared to TI TMS320C25, and Motorola DSP56000/01.

3.4.4 The Texas Instrument TMS320 Family

The TMS320 Family of processors includes the first generation such as TMS32010, TMS320C15, TMS320C17; second generation TMS32020, TMS320C25, and third generation TMS320C30 which is a floating-point DSP, and has the highest speed (33 MIPS) in the family. We will discuss some distinct architectures of the DSP related to the MCDAEC implementation.

As noted in [19], the high performance of the TMS320

series is accomplished using the following concepts :

- Hardvard architecture.
- Extensive pipelining.
- Dedicated Hardware multiplier.
- Special DSP instruction such as MACD which performs multiply-accumulate-data shifting in the same instruction.
- Fast instruction cycle.

The first generation of TMS320 Family is considerably slow for MCDAEC application, so the discussion is omitted here. The second generation includes TMS32020 which features

- 200ns instruction cycle.
- 544 words of on-chip RAM which is partitioned into blocks B0, B1, B2.
- Double-buffered serial port.
- NMOS technology.

As discussed in [10] the TMS32020 was used to implement a half-duplex Echo Canceller of 128-taps. Its architecture allows an efficient convolution which is the most time-consuming task in Echo Canceller. Far-end samples, and FIR coefficients can be arranged in block B1, B0 and the convolution can be accomplished in a single instruction. It is relatively slow for MCDAEC application where a number of trunks shares the same DSP. Furthermore, TMS32020 is fabricated in NMOS which requires a high power consumption. A better performance of TMS32020 is TMS320C25 which offers a

higher speed of 100ns, and is fabricated in CMOS. Sharing the same architecture with the TMS32020, TMS320C25 differs in :

- Eight auxiliary registers with a dedicated arithmetic unit.
- Eight level hardware stack.
- Accumulator carry bit, and related instructions.

Externally, the program, and data memory spaces are multiplexed over the same bus so as to maximize the address range for both spaces while minimizing the pin count of the device (TMS320C25 is the same size as TMS32020, 68-pin PGA package). Internally, the TMS320C25 architecture maximizes processing power by maintaining two separate bus structures, program and data for full-speed execution. Program execution takes the form of a three level instruction pipeline : Fetch, Decode, and Execute. For MCDAEC application the following drawbacks are noted :

- Both data, and coefficients have to be resided in on-chip RAM in order to use the MACD with no loss of execution time. Because of its multiplexed arrangement the MCDAEC requires a large memory storage of data, and coefficients. External memory is necessary for storing coefficients. Overhead time arises in this case when data, and coefficients are block-moved from external memory, to on-chip memory before executing the MACD instruction.
- A single serial-port is not effective in this

application where two multiplexed signals (T_x from C/S - O/S and R_x from O/S to C/S) are the inputs of the DSP.

- Host design concept of the MCDAEC may not be used in TMS320C25 because of its structure in which no Host port is provided.

Third generation , TMS320C30, provides a superior performance in speed, interfacing ports, but is very expensive, and not yet available at the time of this project.

3.4.5 The Motorola DSP Family

In general, the Motorola DSP56000/01 has an equivalent performance to the TMS320 Family. DSP56000/01 is a high-performance, user-programmable DSP implemented in high-density, low-power CMOS technology. It features [20] :

- 100ns instruction cycle.
- High precision (floating-point, and 24-bit fixed point arithmetic).
- 512 words x 24-bit data RAM which is partitioned into XRAM, and YRAM.
- Bootstrap mode (DSP56001).

One of the attractive features of the DSP56001 is its bootstrap mode. An Echo Canceller program can be stored in low-cost memory, and downloaded into each DSP during system initialization. The DSP56000/01 interface facility is powerful with Host processor interface, Serial Communication interface,

and Synchronous Serial interface. The Host interface is a byte-wide, full-duplex parellel port that can be connected directly to the data bus of a host processor. Information can be exchanged between the Host, and DSPs using this port. The Serial Communication interface (SCI), and Synchronous serial interface (SSI) provide full-duplex serial communications. Multiplexed signals T_x, and R_x can be input using these two using these two serial ports. In addition, the DSP56000/01 provide three modes of power : full, half, and shut-down power consumption. This feature is useful in MCDAEC where upon knowing status of subscriber trunks sharing the same DSP the power can be switched to a standby power mode.

The MSM6992 in the Oki Electric Family has several powerful features which are suitable for implementing the MCDAEC. However, cost and engineering support for the MSM6992 are the disadvantages of this DSP at the time of this project. The WEDSP16 shares the same disadvantages. The remaining two candidate DSPs are : TMS320C25, and DSP56000/01. Disadvantages of the TMS320C25 have already been given in section 3.4.5. DSP56000/01 host structure is the most suitable DSP for implementing the MCDAEC which employs multiprocessing configuration.

3.5 A New Approach : The Statistical BLMS

In two previous sections we compared different Echo Cancellation algorithms, and reviewed currently available DSPs. In this section, we will approach as follows :

- Implement the BLMS algorithm, and perform simulation tests to investigate the convergence speed, actual execution time.
- Propose a new approach based on the statistical behavior of trunks.

3.5.1 Simulation of the BLMS

An implementation of a single half-duplex Echo Canceller is carried out to serve these purpcses :

- Investigating convergence speed of the BLMS.
- Investigating actual execution time.

The model of a half-duplex Echo Canceller as shown in Fig.9 has the following specifications :

- Echo path delay of 20ms which includes a bulk delay of 12ms, and 8ms due to echo impulse response, propagation delay, and repeater processing delay as represented by Blocks 1, and 2.
- White-noise as input test signal.
- The Echo path delay is simulated using a first-order low-pass filter with transfer function shown in Fig.10.

• BLMS algorithm of size M = 16, and DSP56001.

Two half-duplex echo canceller modules (128 taps, and 64 taps) were simulated. With a sampling rate of 8kHz, a 20ms Echo Canceller would require 160 taps. Since the initial frame delay of 12ms is previously known, we can bypass the Echo Canceller during this interval, so minimum number of taps can be used to maximize the execution time. It is estimated that 64-tap Echo Canceller performs as well as a 128-tap Echo Canceller. This expectation is confirmed in Fig.11. In this Figure, the Echo Return Loss Enhancement (ERLE), which characterizes a performance of an Echo Canceller, is defined as :

$$ERLE = -10\log \{ E^2(error)/E^2(echo) \} \qquad (3.8)$$

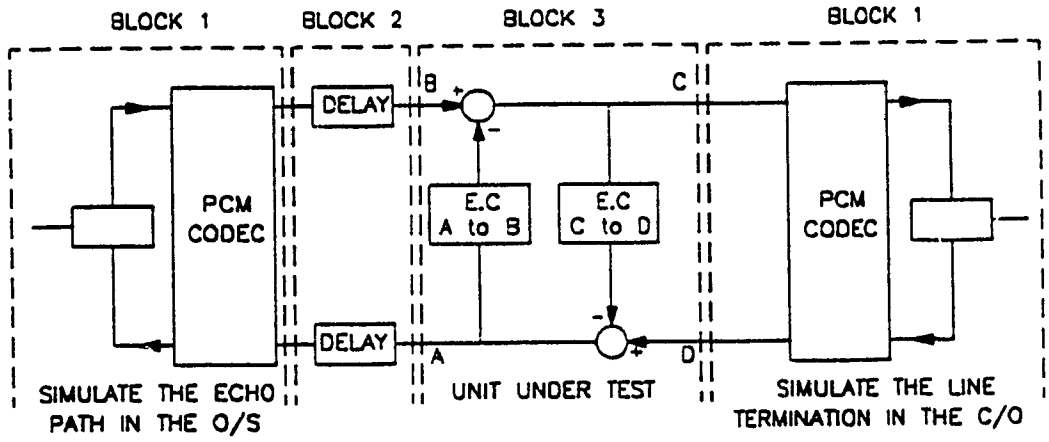


Figure 9. A Model of The Echo Canceller Simulator

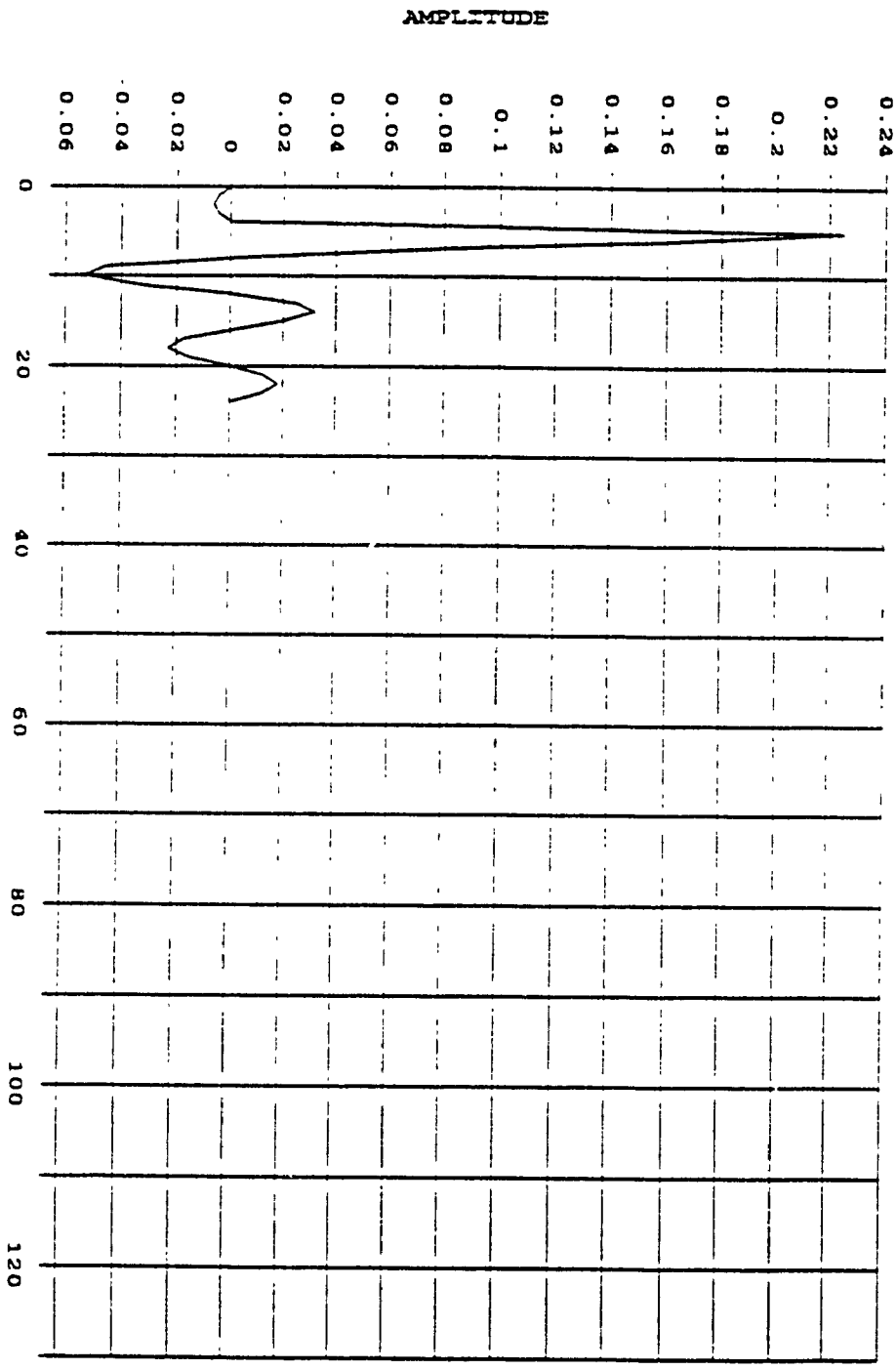


Figure 10. The Echo Path Impulse Response

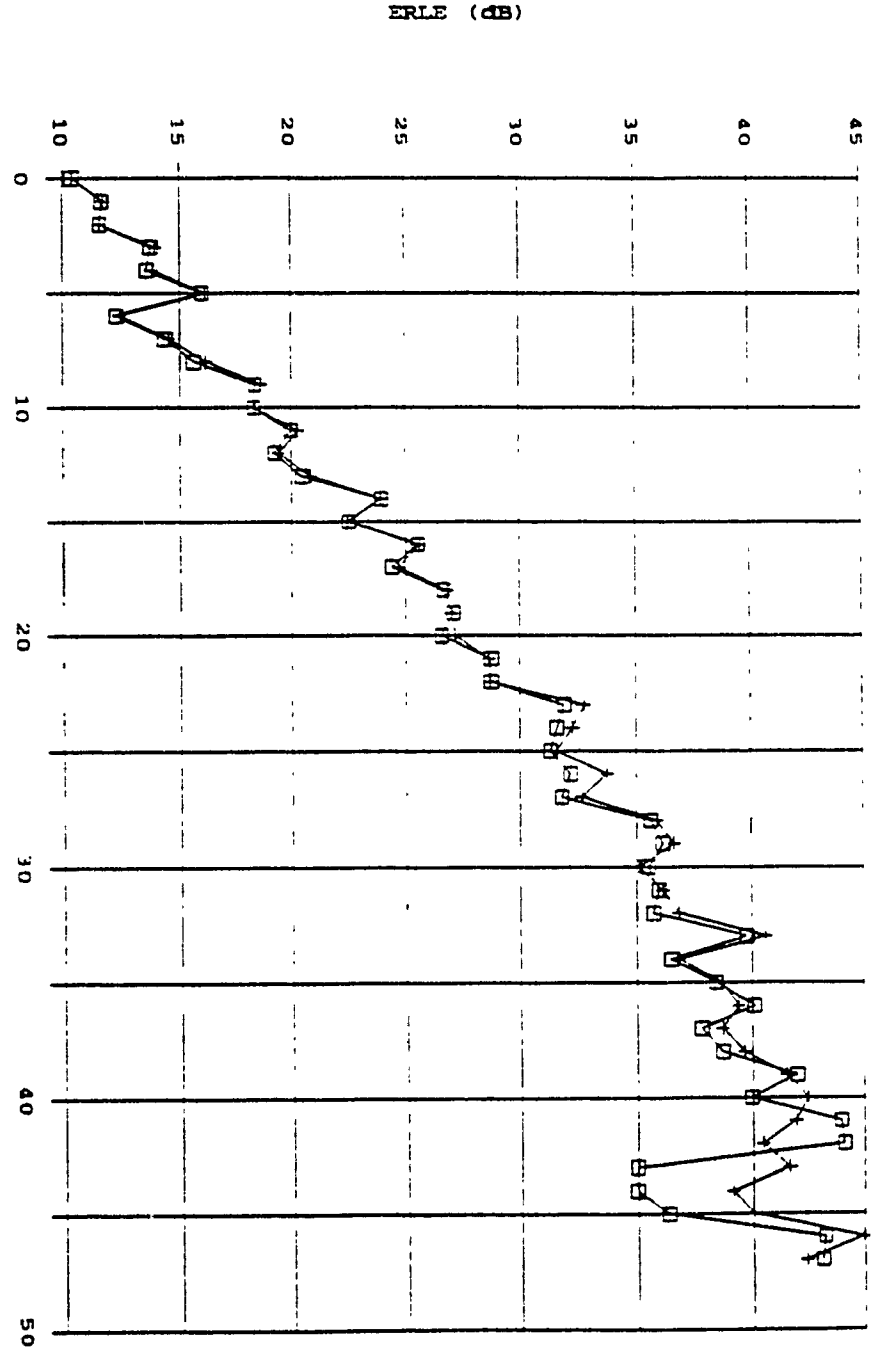


Figure 11. The Echo Return Loss Enhancement (ERLE)

Every point on the graph represents average of 80 samples. As shown, similar results were obtained for both modules. Convergence speed is defined as the time taken for an Echo Canceller to converges to -27 dB, which is about 250ms from the simulation.

Also, the coefficients as estimated by the Echo Cancellers are compared with the echo path coefficients. Excellent results were achieved.

Table 2 summarizes the modules used in Echo Cancellers, and the amount of time taken for each module.

A full-duplex Echo Canceller for a single channel was also developed based on results from the previous simulation of a half-duplex Echo Canceller. Additional tasks are performed :

• Active direction detection : during double talk (when both subscriber speak at the same time) the Echo Cancellation process is frozen to avoid divergence. Therefore, at any time, only one of the two Echo Cancellers (a full-duplex Echo Canceller consists of two half-duplex Echo Cancellers) can be active. A detection of currently active direction can improve excution time by switching on the corresponding Echo Canceller.

• A bulk delay : this is to take into account the initial 12ms delay (in direction C/S to O/S only).

A full-duplex Echo Canceller takes 840 cycles.

Table 2. A Half-Duplex Echo Canceller Real-Time Analysis

STEP	MODULE	CPU CYCLE
1	Cycle Start	32
2	Echo Estimation	106
3	Residual Output Suppression	15
4	Linear to μ (or A) law	27
5	Power Estimation	32
6	Output Normalization	30
7	Near-End speech detection	48
8	Coefficient Increment Update	99
9	Coefficient Update	31
10	Cycle end	30
	TOTAL	450

Based on these results the number of full-duplex E.Cs per DSP can be calculated :

$$N\_ec = Tcyc / cyc$$

where :

- N_ec is the number of E.Cs per DSP.
- Tcyc is the total cycles per iterations, Tcyc = 125μs/97.5ns = 1280.
- cyc is the cycles required for a full-duplex E.C per iteration, cyc = 840.

The estimated number of E.Cs per DSP is 2 if the BLMS algorithm is implemented using the DSP56001 (the real-time overhead due to I/O operations is not included in the estimation). A P-MP subscriber system with N full-duplex trunks would need N/2 DSPs.

3.5.2 The Statistical Approach

We examine a scenario which involves the speaker A with an echo canceller to cancel echo reflected to A. When the speaker A is silent, the E.C is idle. When the speaker A starts speaking at time t_0, the E.C is activated and performs echo cancellation with a set of coefficients. While the speaker A continues talking, the E.C performs echo cancelling and updating its coefficients. At time $t_1 > t_0$ the echo return loss is $\leq$ -24 dB. At this time we say the echo canceller has converged. If the speaker A continues speaking, the E.C can perform only echo cancelling with the set of converged coefficients.

Based on the above scenario, and preliminary analysis of the real-time performance of the echo canceller, the following observations are made :

(i) The convergence rate of the Echo Canceller algorithm is well within the 500ms as recommended by the CCITT G.165 (for white noise as done in this case). In the above scenario ($t_1 - t_0$) should be $\leq$ 500ms.

(ii) Once the Echo Canceller converges, its coefficients do not need to be frequently updated during the conversation, therefore the Echo Canceller can operate with a set of converged coefficients.

(iii) Statistically speaking, in a DA-TDMA subscriber radio system the probability that all N full-duplex voice trunks are required at the same time, is very small [21]. When trunks handled by the same DSP become active at the same time the DSP will serve trunks on a first come, first serve basis. In this case, if a trunk becomes active when the DSP has been already at the full load, the trunk will not be served by the DSP. This implies that echo on this trunk will not be cancelled until the DSP becomes available within a maximum time of 500ms. The maximum time of 500ms is based on the fact that the estimated coefficients of the echo canceller converge within 500ms. After this time, the coefficients need not be frequently updated, and the DSP is now available to serve new trunks.

Therefore, we propose the following statistical approach to improve the throughput, and to minimize the number of required DSPs :

(a) At any given time a trunk can be in any one of the four states : IDLE, TRANSIENT, STEADY, and WAIT. These states are defined as follows :

- IDLE : trunk is not active (i.e it carries no traffic),

and it will remain in this state until a request for voice transmission is made. While in this state, system counters (Nt : number of trunks in TRANSIENT state; Ns : number of trunks in STEADY state) are unchanged.

- TRANSIENT : a trunk starts carrying voice traffic, and its ERLE > -24dB. In this state, the following tasks are performed :
  - Cancelling echo.
  - Updating coefficients.
- STEADY : a trunk is currently carrying voice traffic, and its ERLE ≤ -24 dB. In this state only echo cancelling is performed.
- WAIT : a trunk is active, but the DSP is completely busy. It waits for service from DSP.

Table 3 gives a summary of trunk status in the system at any time. From simulation results, the number of full-duplex trunks that can be handled by each DSP is three. As an example, in case 4 (Table 3) the present status indicates that there is a trunk in STEADY state (S) mode. In the next sampling time three more trunks become active at the same time. Echo Cancellation will be performed on the two trunks which becomes active first. These two trunks have initial state as TRANSIENT (T), and the last one will be in WAIT (W) state. During the next 500ms the first two trunks should reach STEADY state. At this moment, the total number of trunks in STEADY state are three, and there is one trunk in WAIT state.

The trunk in WAIT state is served after a waiting time of 500ms, and moves in TRANSIENT state. In another case, we have one TRANSIENT trunk, and three trunks become active at the same time. To handle this case, a total waiting time of 1000ms is taken before all trunks are served. Because waiting time of more than 500ms is undesireable, a more realistic design (taking into account the Input/Output time, and overhead required) is to have each DSP handling a maximum number of three trunks using the statistical approach. To illustrate the improvement of this approach over the deterministic approach we consider the case that there are two trunks in TRANSIENT state. During the next sampling time, while these two trunks are still in TRANSIENT state, a third trunk becomes active. In deterministic approach there is no STEADY state, and therefore all trunks are processed without detecting their Echo Return Loss Enhancement, i.e number of operations (echo cancelling & coefficient updating) are always performed. Consequently, DSP cannot handle more than 2 trunks in TRANSIENT state. In the statistical approach coefficient updating is halted when the trunk has reached a STEADY state. The third trunk waits for a maximum time of 500ms (time for one of the two trunks to reach STEADY state), and then will be served. For a number of three trunks to be handled by each DSP, the proposed approach may introduce a maximum waiting time of 500ms (as in previous example), but gives a 30% increase in number of trunks per DSP over the conventional approach.

Table 3. A System Real-Time Analysis

Present Status			Coming Trunk	Next Status			Comment
T	S	W	Number	T	S	W	Waiting time of the coming trunks (ms)
0	0	0	0	0	0	0	
0	1	0	1	1	1	0	
0	1	0	2	2	1	0	
0	1	0	3	2	1	1	500
0	2	0	0	0	2	0	
0	2	0	1	1	2	0	
0	2	0	2	1	2	1	500
0	3	0	0	0	3	0	
0	3	0	1	1	3	0	
0	4	0	0	0	4	0	
1	0	0	0	1	0	0	
1	0	0	1	2	0	0	
1	0	0	2	2	0	1	500
1	0	0	3	2	0	2	1000
2	0	0	0	2	0	0	
2	0	0	1	2	0	1	500
2	0	0	2	2	0	2	1000
2	0	1	1	2	0	2	1000

Chapter 4

STRUCTURE OF THE MULTICHANNEL ECHO CANCELLER

This chapter describes the hardware and software structures of the MCDAEC. The hardware configuration consists of a TDMA Controller CPU, a Controller, a dual-port RAM, and a number of DSPs. Similarly, software structure is divided into four phases : input/output (I/O), status detection, echo cancellation, and coefficient updating. During each frame of 125 μs the states of the trunks are detected, and updated.

Finally, the test results are performed to evaluate the performance of the MCDAEC.

4.1 Hardware Configuration

The MCDAEC is located in the C/S to perform Echo Cancellation on trunk basis. A simplified block diagram of the MCDAEC is shown in Fig.12.

As shown in this figure four main functional units are identified : the TDMA Controller CPU, the dual-port RAM, the Controller of the E.C card, and the DSPs.

Part of the CPU's function is to maintain the Echo Canceller card. During each frame of 8ms trunk status (IDLE or ACTIVE) is passed to the Controller via Dual-port RAM. The dual-port RAM is the main interface between the CPU and the

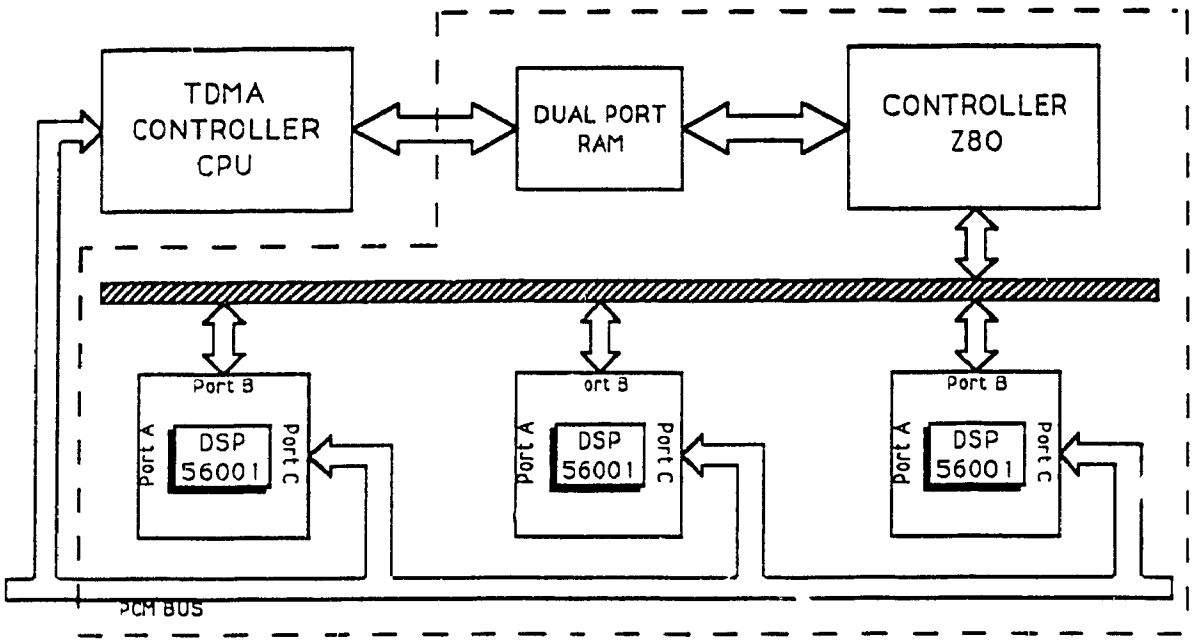


Figure 12. A simplified MCDAEC Block Diagram

Controller. It is used as a communication channel through which information is transferred from the CPU to the Controller for controlling the Echo Canceller card, or from the Controller to the CPU the information on trunk status for maintenance purposes.

The Controller, which is realized by a low cost Z80 processor, functions as a Master while the DSPs are the Slaves. The Controller communicates with the DSPs by means of Host port (port B). Maintenance information such as trunk status (IDLE, or ACTIVE), Watchdog (SET, CLEARED),.., are passed from the Controller via fast interrupts using the appropriate vectors which will be described later.

The MCDAEC is realized by 20 identically configured DSP56001s. Each DSP performs echo cancellation process for three full-duplex trunks, and is directly interfaced with the

Controller through the Host port (port B), with the PCM streams (from C/S) through port C in which Synchronous Serial Interface (SSI), and Serial Communication Interface (SCI) will be used.

4.1.1 Maintenance CPU

As previously described the Maintenance Function is implemented in part of the TDMA Controller.

4.1.1.1 Interfacing with the Controller

The CPU card interfaces with the Controller through a dual-port RAM. The memory map is shown in Table 4 below.

For address range B000 - B076 the Maintenance CPU will update trunk information to the Controller using the commands in table 5. This information is received by the Controller, and will be passed to the corresponding DSPs. In a similar fashion, E.C status detected by the Controller is transferred to the Maintenance CPU for display purposes. The E.C Controller (B200) can be in both Read and Write modes. In the Read mode the Maintenance CPU reads the E.C Status (B200) information about the status of the Controller (NORMAL or FATAL ERROR). In the Write mode the Maintenance CPU resets, or bypasses the whole E.C card by writing to the E.C Controller (B200).

Table 4. Dual-port RAM Memory Map

CPU	Z80	DESCRIPTION	HW
B000	A000	CPU Command for trunk #00	00
B002	A001	CPU Command for trunk #01	01
B004	A002	CPU Command for trunk #02	02
....	....	.....	..
B076	A03B	CPU Command for trunk #59	59
B0A0	A050	E.C. Status for trunk #00	00
B0A2	A051	E.C. Status for trunk #01	01
B0A4	A052	E.C. Status for trunk #02	02
....	....	.....	..
B116	A08B	NOT USED	
....	....	.....	
B1FE		NOT USED	
B200		E.C. Controller	

The CPU Commands are summarized in the following table.

Table 5. CPU Commands

CODE	DESCRIPTION
00	Mailbox Empty
01	Trunk Idle
02	Trunk Active
03	E.C. Disabled
04	E.C. Enabled
05	G.165 Test ON
06	G.165 Test OFF

Similarly, the E.C Status for trunk #00 to #59 are summarized in table 6.

Table 6. Echo Canceller Status

CODE	DESCRIPTION
00	Mailbox Empty & Trunk is O.K.
81	Trunk is inoperable & E.C. Bypassed.
82	Trunk bypassed at CPU request
83	E.C. bypassed : DSP RAM (or DSP) failed
84	E.C. bypassed : I/O error

4.1.1.2 Interfacing with the DSPs

The PCM bus structure of a typical P-MP consists of even and odd buses. Each PCM bus holds an equal number of voice trunks. The even PCM bus coming from the CPU Mux card is inputted to the group of 10 DSPs handling E.C for even trunks. Similarly, another group of 10 DSPs handling E.C for odd trunks is connected to the odd PCM stream. The generalized even/odd PCM structure is shown in Fig.13.

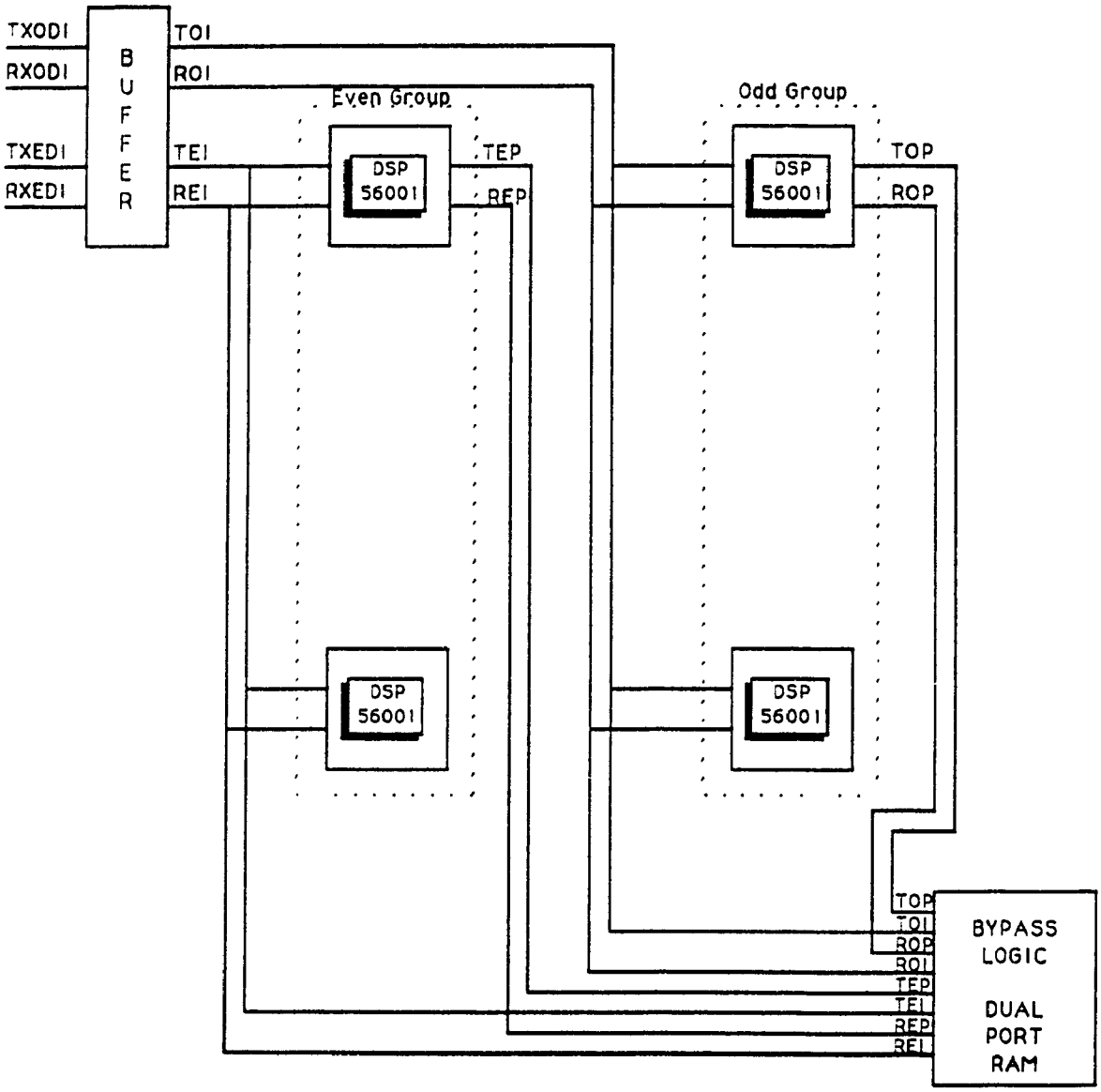


Figure 13. A Generalized Even/Odd PCM Structure

Four maintenance functions are performed by the Maintenance CPU as shown in Fig.14.

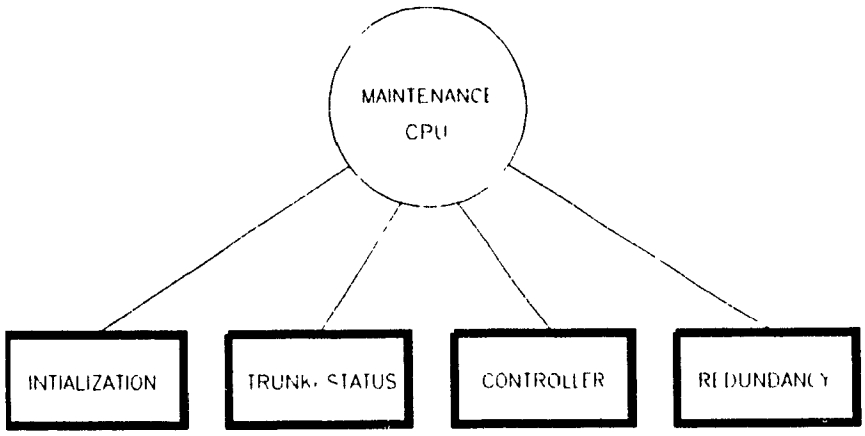


Figure 14. Maintenance Functional Block Diagram

4.1.1.3 Initialization

After the power-up reset the Maintenance CPU activates the Controller reset for downloading process (to be described later). As a first step of the initialization process a RAM test is performed (from B000 - B076). Results of this RAM test determine the status of the Echo Canceller card as follows :

- Not Equipped : if none of write/read attempt is successful, and 00's have always been read.
- Defective : if at least one write/read attemp is failed (data read is not the same as data written, and data read is different than 00).
- Normal : all writes/reads are matched.

A *no action* code is initiated by the Maintenance CPU

indicating a successful initialization; otherwise a *bypass* code is activated which will bypass the whole E.C card.

4.1.1.4 Trunk status updating

One of the distinct characteristics of the proposed Statistical Approach is that a trunk has four states : IDLE, TRANSIENT, STEADY, and WAIT. As explained later, a trunk in IDLE state indates that no request is made, and the DSP can use its time to process other trunks sharing the same processor. In addition, by knowing the trunk status (IDLE, ACTIVE) appropriate actions can be taken such as to keep the DSP in power-down saving mode (when all three trunks sharing the same unit are IDLE). Therefore, it is important that status of trunks has to be dynamically updated.

To provide users the means for maintenance options such as Bypass the whole E.C card (or individual lines), G.165 Test Verification are implemented. In summary, the Maintenance CPU updates the Echo Canceller Mailbox (ECMB) the following information

- Assigned or Idle.
- E.C. disabled or enabled.
- Normal or G.165 test mode

The E.C. is bypassed for trunks which are assigned to lines other than 2-wire voice ones. The whole E.C. card or individual lines (up to 2048 lines) can be selected to be

bypassed. A G.165 test mode is selected on a trunk basis. In this mode, only the trunk selected for G.165 test is echo cancellation enabled, while the rest are echo cancellation disabled.

4.1.1.5 Controller Monitoring

The Controller functions as a Master to all 20 DSPs. It is necessary that the operationality of the Controller is monitored. A malfunction of the Controller seriously harms the performance of the E.C card (in fact nothing will work). Therefore, when this condition occurs it is essential that the whole E.C card will be bypassed.

The Maintenance CPU monitors the Controller by scanning E.C status register (address $B200, I/O) every 500 ms. This is to detect a malfunction caused by the DSP Controller. Should this occur the E.C card will be bypassed so that conversations are still possible (with echoes), rather than not at all. A fatal error arises when the DSP Controller fails to operate properly (bit 7 of $B200 is set with a recognition time of 50ms). When this condition is detected, the CPU will initiate a reset (by writing $01 into $B200), and then bypass the E.C card (by writing $00 into $B200). The CPU will then scan the status register in the next 500 ms. If the error still exists the same process (resetting & bypassing E.C card). No further action is taken if the error still appears after the 2nd reset

attempt. Should the fatal error disappears the resetting will be performed by Watchdog timer. The CPU resumes its detection routine.

4.1.1.6 Redundancy

This option enhances the system reliability. A second Echo Canceller card is used in a stand-by mode. This feature includes :

- Conditions for a switch over : a malfunction in DSP Controller, or more than 5 DSPs are inoperable.
- Information transfer (from main CPU to backup CPU) disable/enable E.C. (the whole card, or individual lines).

4.1.2 The Controller

The Controller is realized using a Z80 CMOS processor as shown in Fig.12. The Controller communicates with both the Maintenance CPU and the DSPs.

4.1.2.1 Interfacing with the TDMA Controller CPU

As discussed in the previous section, the Controller receives updated trunk status information, and passes this to the corresponding DSPs. In the other direction, the Controller

alerts the TDMA Controller CPU the functional status of the DSPs for maintenance purposes. All of these processes are done via the dual-port RAM which functions as an Echo Canceller Mailbox (ECMB).

4.1.2.2 Interfacing with the DSPs

The Controller is interfaced with the DSPs through the host port B.

- All DSP units are connected as I/O mapped peripherals communicating with DSP Controller via DSP host parallel I/O port.
- DSP and Z80 real time codes are stored in 32K x 8 EPROM which is external DSP Controller memory. The DSP code is downloaded via DSP Controller to each DSP unit using the parallel host port.
- Dynamic data storage for DSP Controller software is provided using 8K x 8 external RAM.
- Dual port RAM mapped as Z80 external memory is used as E.C. Mail Box for C/S CPU - DSP Controller communication. 4ms frame sync pulses should be used as DSP Controller maskable interrupts in order to synchronize E.C. Mail Box read and write operations.
- DSP Controller watchdog timing is accomplished using external counters clocked by 8kHz clock. The condition when watchdog timer activates DSP Controller reset should

be latched and C/S CPU notified. The latch can be cleared only by DSP Controller during normal operation. DSP Controller can as well be reset by C/S CPU.

- Memory mapped control registers provide for LED control, DSP reset control, watchdog reset and DSP host port read/write.
- Memory mapped read buffer carries input synchronization flags for E.C. Mail Box operation, test strobes for G.165 testing, system on-line and test mode flags .

4.1.2.2.2 DSP Controller Maintenance

After the power-up reset (intiated by TDMA Controller CPU) the Controller should enter "initialization" routine where mode 1 should be defined. The TCLEN signal is checked, and the OFLIN (off line) routine will be processed if TCLEN signal is detected inactive. This OFLIN will shut down all DSPs using COM-8 command. In this mode the power consumed by DSPs are 0. The DSPs will remain in this state until the TCLEN becomes active. Information exchanges between the Controller and DSPs are done via fast interrupts using the following commands :

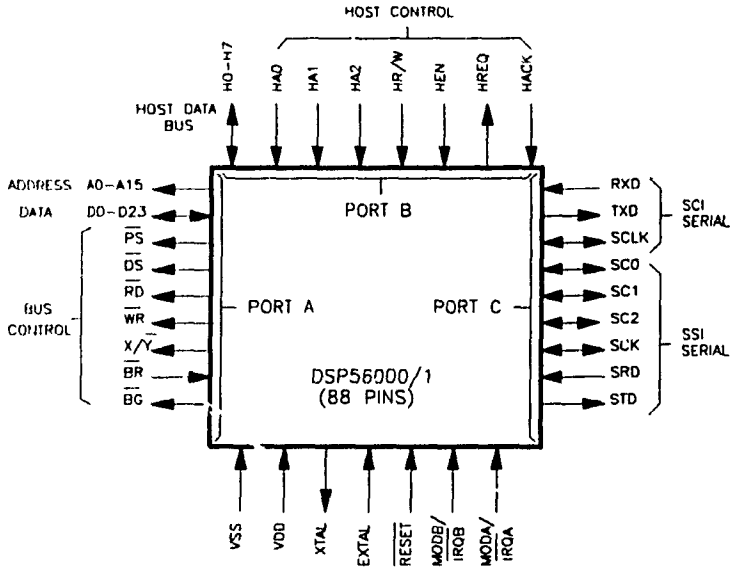
COMMAND	DESCRIPTION	VECTOR
COM0	reserved	0024
COM1	TRUNK1 Command word	0026
COM2	TRUNK2 Command word	0028
COM3	TRUNK3 Command word	002A
COM4	IRQB Disable	002C
COM5	IRQB Enable	002E
COM6	WDF Set	0030
COM7	WDF cleared	0032
COM8	Power down STOP mode	0034
COM9	Get WDF	0036

4.1.3 The DSPs

• Each DSP unit consists of one DSP56001 IC with local 8K x 24 RAM, and handles Echo Cancellation for three trunks. The DSP56001 is an 88-pin integrated circuit. Its input and output signals are organized into seven functional groups which are shown in Fig.15.

A block diagram of the DSP56001 architecture is shown in Fig.16.

The core of the processor is organized as three separate execution units : the Data ALU, the Address Arithmetic Unit, and the Program Controller. These execution units operate in parallel, providing the resources needed to execute most



Port A : Address, and Data buses.

Port B : Host Interface.

Port C : Serial Communications Interface (SCI), and Synchronous Serial Interface (SSI).

Figure 15. DSP56001 Functional Signal Group

instructions in a single 97.5 nsec instruction cycle (20.5 MHz clock). In addition, each execution unit calculates results in a single instruction without pipelining. The DSP56001 provides a large set of on-chip memory, and I/O peripheral resources to support the core processor.

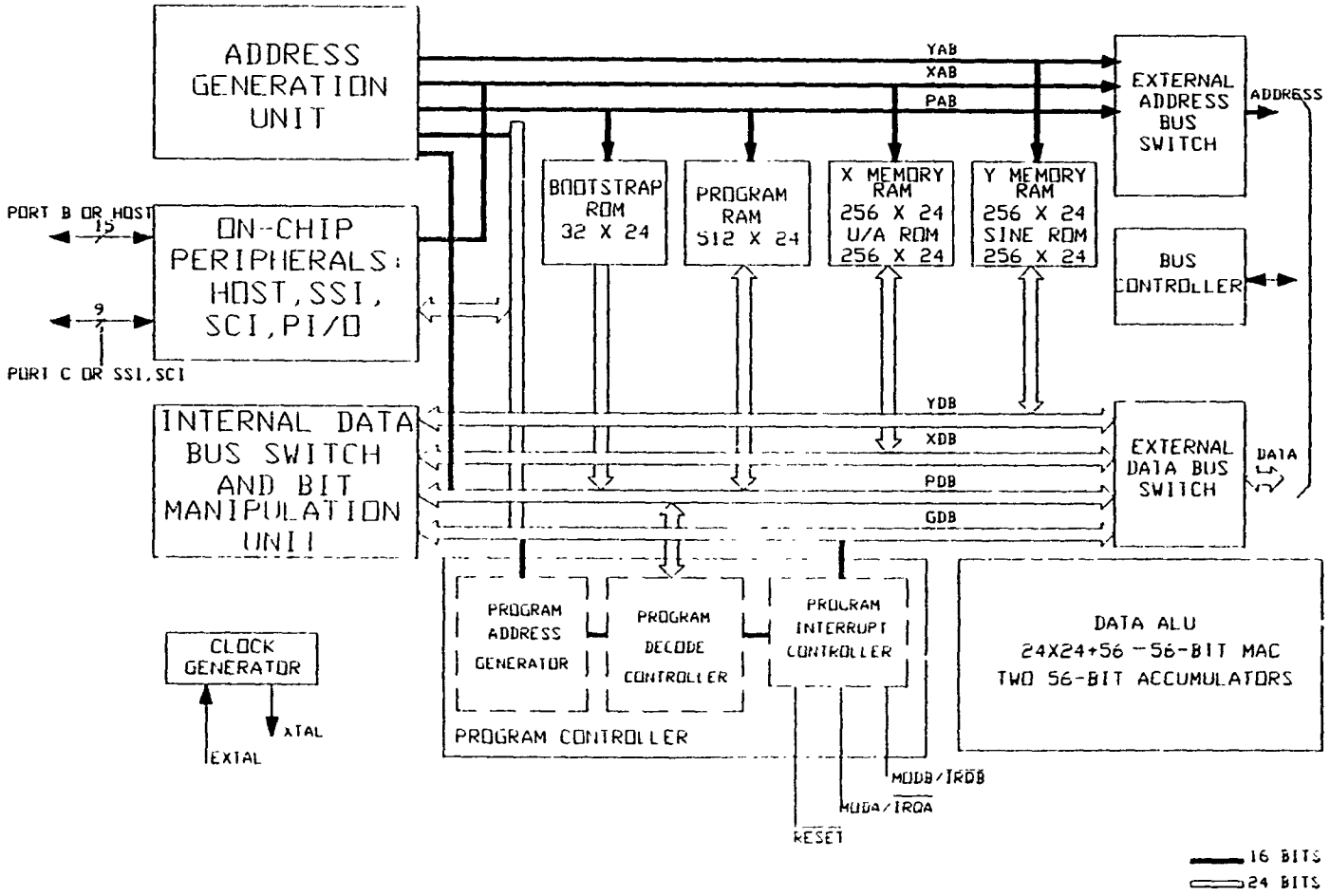


Figure 16. DSP56001 Architecture Block Diagram

The DSP56001 contains three on-chip RAMs :

- a 256 x 24 bit X data RAM.
- a 256 x 24 bit Y data RAM.
- a 512 x 24 bit program RAM.

In addition, a special 32 x 24 bit Bootstrap ROM is factory programmed with a bootstrap loader program that performs the initial loading of the program RAM.

4.2 Software Configuration

There are 20 identical units to handle all 60 trunks. The statistical multichannel echo cancellation process approach is divided into 4 main phases as shown in Fig.17.

Phase 0 : Input / Output (I/O)

During each PCM frame of 125 μs inputs/outputs of 3 full-duplex trunks are performed.

Phase 1 : Status Detection

Status of each trunk (double/single/silence & transient/steady) is detected. The execution of the next two phases is performed based upon results from this phase.

Phase 2 : Echo Cancellation

A replica of echo is generated using a 64 tap finite impulse response filter (FIR). In this phase the residual error is also computed.

Phase 3 : Coefficients Updation

Coefficients updation is executed based upon results from phase 1. This phase is processed only if a trunk is single, or in transient state (residual error is greater than 24dB).

In the following the state machine of the MCDAEC will be described. The state machine consists of four states : IDLE, WAIT, TRANSIENT, and STEADY state as shown in Fig.18.

- IDLE : a trunk is not requested (REQ = 0). In this

state system counters indicating the number of trunks in TRANSIENT state (Nt), and the number of trunks in STEADY state (Ns) are unchanged.

- TRANSIENT : a request for voice transmission is made (REQ = 1), and the Echo Return Loss Enhancement (ERLE) is greater than -24 dB. In this state the following tasks are performed :
    - Echo Cancellation.
    - Coefficient Updating.
- STEADY : a voice traffic trunk is assigned (REQ = 1), and the ERLE $\leq$ -24 dB. In this state, only echo cancellation is performed.
- WAIT : a trunk is requested for voice transmission (REQ is 1), but the DSP is currently busy in serving other trunks. The trunk remains in this state until the DSP is free to serve it. The maximum waiting time is 500 ms.

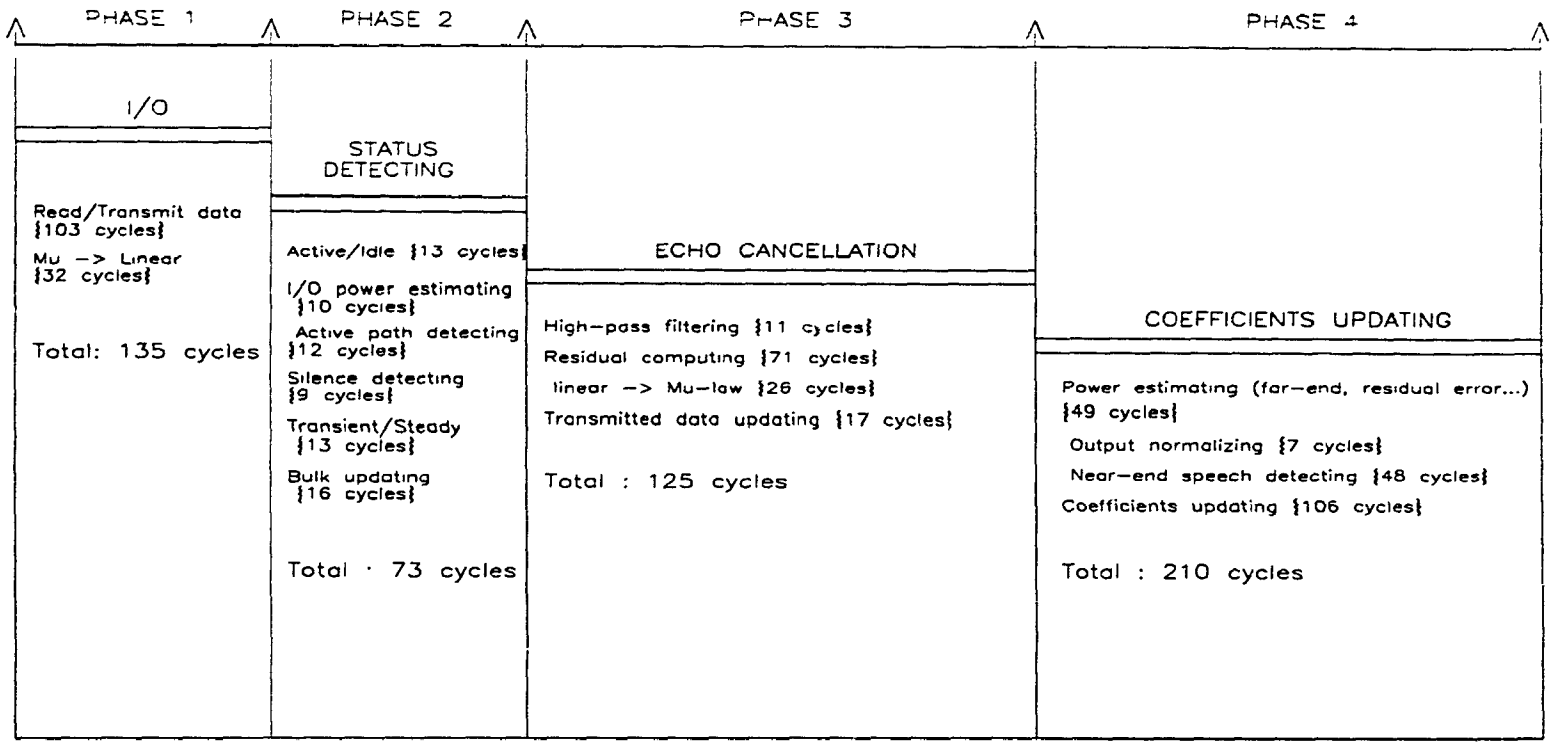


Figure 17. The MCDAEC Process

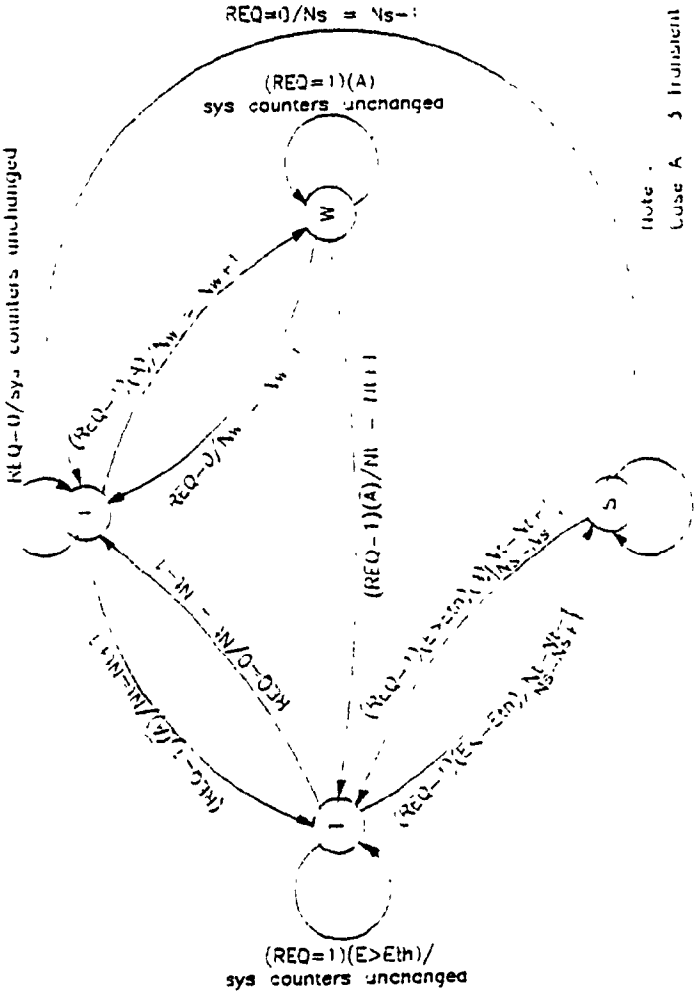


Figure 18. State Diagram for each channel

The operation of the MCDAEC in four phases (Fig.17) is as follows :

Step 1 : Input/Output

For each frame of 125 μs three pairs of I/O need to be performed. Real time I/O activity scheme is shown in Fig.19.

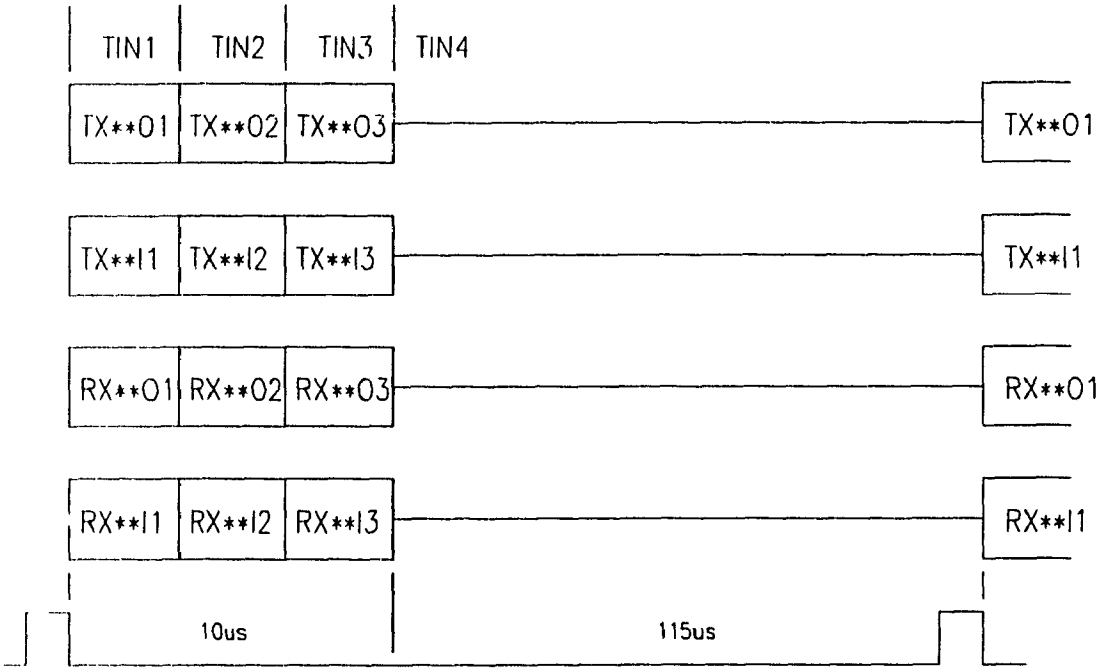


Figure 19. Input/Output of the MCDAEC

IRQB is the external interrupt whose negative edge starts the Echo Canceller real-time routine. Within each TIN time interval the following actions are performed :

TIN1 : 0 to 3.29 μs	load STX with T_x**02
	load TX with R_x**02
TIN2 : 3.29+ μs to 6.58 μs	load STX with T_x**03
	load TX with R_x**03
	read T_x**I1 from SRX
	read R_x**I1 from RX
TIN3 : 6.58+ μs to 9.87 μs	read T_x**I2 from SRX
	read R_x**I2 from RX
TIN4 : 9.87+ μs to 10 μs	read T_x**I3 from SRX
	read R_x**I3 from RX

During the first 10 μs, listed I/O actions will be performed with T_x**I bytes converted (LSB becomes MSB) using the SC conversion table. Once the I/O routine has been completed, Echo Cancellation is then processed for each trunk at a time.

Step 2 : μ/A Law Converting

For each trunk a pair of signals T_x (from C/S to O/S), and R_x (from O/S to C/S) in coded PCM (μ or A law) are converted into linear using a table lookup already implemented in DSP56001's ROM. The table is only for positive signals, therefore a sign detection needs to be processed, and then converted, accordingly.

Step 3 : Trunk Status Detection

During each frame status of each trunk is monitored. The

fol..wing table summarizes the state transition, and the actions taken between states.

Table 7 : State transition

Current State	Previous State	Actions taken
IDLE	IDLE	• skip E.C process
IDLE	TRANSIENT	• skip E.C process • update counter Nt = Nt - 1
TRANSIENT	IDLE	• perform E.C process. • update counter Nt = Nt + 1
TRANSIENT	TRANSIENT	• perform E.C process.

Step 4 : Wait state Detecting

Real-time analysis shows that only two trunks in transient state (ERLE > -24dB) can be processed during each frame. A third trunk in transient mode is forced into WAIT state. The maximum waiting time is 500 ms. Trunks are performed on a first come, first served basis.

Step 5 : Input Power Estimating

Average powers of Tx and Rx are updated. The estimated computation has been used as follows :

$$\bar{T}x(i) = (1 - \alpha) \times \bar{T}x(i-1) + \alpha \times ABS(Tx(i)) \quad (4.1)$$

$$\bar{R}x(i) = (1 - \alpha) \times \bar{R}x(i-1) + \alpha \times ABS(Rx(i))\} \quad (4.2)$$

where $\alpha = 2^{-5}$

Step 6 : Silence Detecting

Echo cancelling performed on a silent trunk could introduce some unwanted, additional noise in the system. Both subscribers (Tx , and Rx) power levels are compared with a silence threshold of -48 dB. The rest of E.C routine will be skipped if both subscribers are silent.

Step 7 : Active Path Detecting

Both subscribers' power levels are compared with each other to determine which of the two is current talker. Echo canceller which corresponds to the currently talking subscriber will be switched on, while the other remains unactive.

Step 8 : Bulk Delay Detecting

This is only applicable to the Echo canceller from C/S to O/S direction. A 12ms initial delay (for a typical P-MP subscriber system) is introduced when this side is active. Input signal Tx will be buffered until a 12ms delay has been finished.

Step 9 : High-pass Filtering

The received near-end signal SDC(n) is passed through a HPF with a cutoff frequency of 160 Hz. The reason for implementing a HPF is to remove DC offset which is generated by codec. An approximation of HPF is used to save computation-

time :

$$SO(n) = (1-2^{-3}) \times \{ SO(n-1) + [SDC(n) - SDC(n-1)]/2 \} \quad (4.3)$$

where :

SO(n), SO(n-1) : HPF near-end at time n, n-1.

SDC(n), SDC(n-1) : linear near-end at time n, n-1.

Step 10 : Echo Estimating

A 64-tap FIR is used to estimate response of echo path. Far-end samples y(n)'s are convolved with the FIR coefficients to generate a replica of echo.

$$\hat{r}(n) = \sum_{m=0}^{m=63} \{a(n) \times y(n+m)\} \quad (4.4)$$

The residual echo is calculated as :

$$e(n) = SO(n) - \hat{r}(n) \quad (4.5)$$

Step 11 : Residual Error Suppressing

To further improve the performance a suppressor is introduced in the design. The ERLE is calculated as :

$$ERLE = L_e(n-1)/L_y(n-1) \quad (4.6)$$

where :

$L_e(n-1)$: estimated residual error power at time n-1.

$L_y(n-1)$: estimated far-end power at time n-1.

Residual error e(n) will be clipped to 0 if ERLE $\leq$ -24dB.

Step 12 : Power Estimating

Approximation is used to compute powers of far-end, residual error because exact calculation is very time-expensive.

$$L_y(n) = L_y(n-1) + 2^{-7} \times \{ABS(y(n)) + cutoff - L_y(n-1)\} \quad (4.7)$$

$$L_e(n) = L_e(n-1) + 2^{-7} \times \{ABS(e(n)) - L_e(n-1)\} \quad (4.8)$$

where :

y(n) is the far-end sample at time n.

e(n) is the residual error at time n.

Step 13 : Output Level Normalization

The SBLMS algorithm requires a storage of 16 most recent outputs. These outputs are normalized by $L_y(n)$.

$$U_n(n) = e(n)/L_y(n) \quad (4.9)$$

The normalized outputs will be used to update coefficients. A multiplication of 8192 is required to represent these normalized outputs in the same format with coefficients.

$$U_n(n) = U_n(n) \times 8192 \quad (4.10)$$

Step 14 : Near-End Speech Detection

Coefficients updating is processed only in the absence of near-end speech. Near-end speech is detected by comparing near-end signal Lx (n) with far-end signals within the 8ms time frame.

Near-end is detected when the following condition meets

$L_x(n) > \frac{1}{2} \max (L_y(n), L_y(n-1),\ldots,L_y(n-63))$

Furthermore, coefficients updating is bypassed if the far-end talker is silent for a long time. This is to prevent introducing additional noises in the system.

Step 15 : Coefficients Updation

A block size of 16 is used in the SBLMS algorithm. Coefficients are divided into 16 groups as follows :

group 0 : a_0, a_{16}, a_{32}, a_{48}

group 1 : a_0, a_{17}, a_{33}, a_{49}

group 15: a_0 , a_{31}, a_{47}, a_{63}

During each PCM frame (125 μs) coefficients in each group (from 0, 1, to 15) are updated.

$$a_H(n+1) = a_H(n) + (2^{-10}/L_y(n)) \times \{\sum_{m=0}^{m=15} e(n+m) \times y(n+m+H)\} \quad (4.11)$$

Rearranging and substituting $U_n(n) = \{e(n)/L_y(n)\} \times 8192$

$$a_H(n+1) = a_H(n) + (2^{-10}/L_y(n)) \times 8192 \times \{\sum_{m=0}^{m=15} un(n+m)y(n+m+H)\} \quad (4.1$$

where H is the block size counter (from 0, 1, 2,.., 15)

4.3 Performance Tests

The MCDAEC is implemented using the proposed approach described in section 3.5.2. Also as discussed in section 3.5.1 each DSP56001 can handle up to three trunks. Therefore, a typical P-MP subscriber radio system with 60 trunks needs 20 DSP56001s.

The following subjective and objective tests are carried out to evaluate the *echo cancellation performance* consisting of real-time performance, and convergence speed.

The test set-up consists of a typical P-MP subscriber radio system, and test equipments as shown in Fig.20.

- A Central station (C/S) having 60 trunks.
- An Outstation (O/S).
- A Transmission Measurement Set (TMS) to generate white noise, and measure echo return loss.

Both subjective and objective tests are included as follows :

4.3.1 Subjective Tests

The following tests are performed to subjectively evaluate the echo cancellation performance. They consist of :

- Simulated additional delays into the SR500 system to increase the Echo Path delay so that echo becomes easily

distinguished.

• Two versions of MCDAEC were subjectively tested : one with Echo Canceller *enabled*, the other with Echo Canceller *disabled*. We tested both versions, and observed that the version with Echo Canceller *enabled* improves the system's performance by effectively removing the echo.

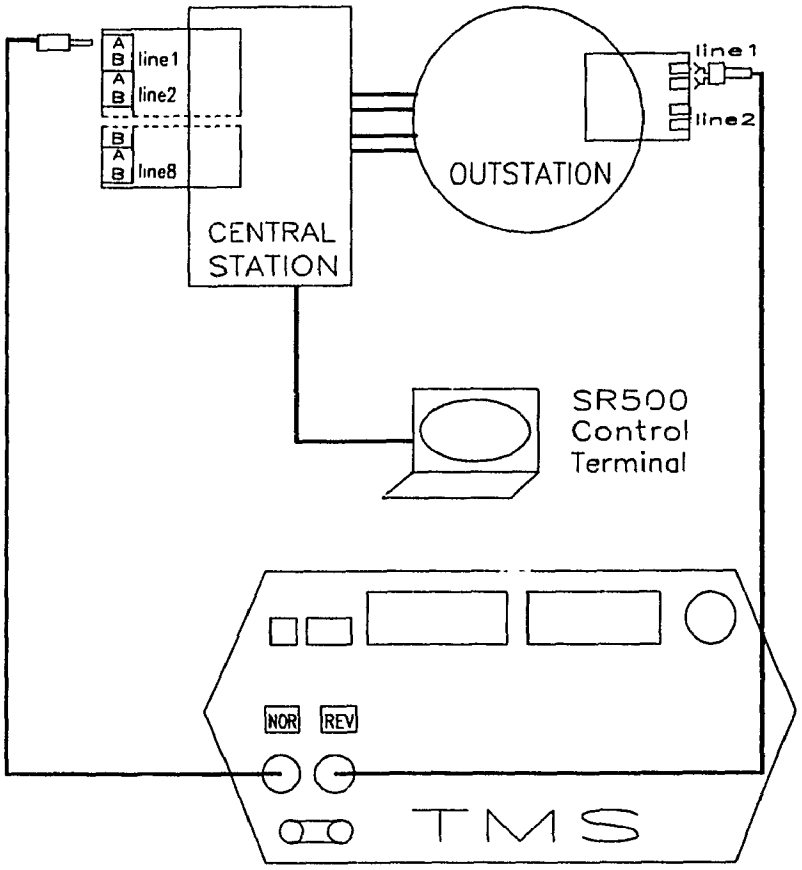


Figure 20. A Test Set-Up

• Adaptive characteristic test was also performed. The MCDAEC should be able to detect to changes in the echo path, and to adapt to it. Results were plotted, and shown in Fig.21

In this figure echo canceller coefficients of one typical trunk (in this test trunk #3 was randomly selected) were recorded and plotted for two cases :

- During the first test no additional delay is introduced in the system. Fig.21 shows the peak coefficient at around 0.7ms.
- During the second test an additional delay of 3ms is introduced in the system. Fig.21 shows the peak coefficient has been shifted by 3ms as expected.

4.3.2 Objective Tests

The objective tests are carried out using the TMS to inject white noise in the system, and then measure the echo return loss. The tests are performed based on CCITT G.165 Recommendation. The test cases are as follows :

- Testcase #1 : Steady state and Returned echo level

  With the coefficients set to 0, and the nonlinear processor (NLP) disabled, the echo returned level is measured for different input signal varying from -30dBm to -10dBm. The results were plotted and compared with the CCITT G.165 recommended performance. As shown in Fig.22

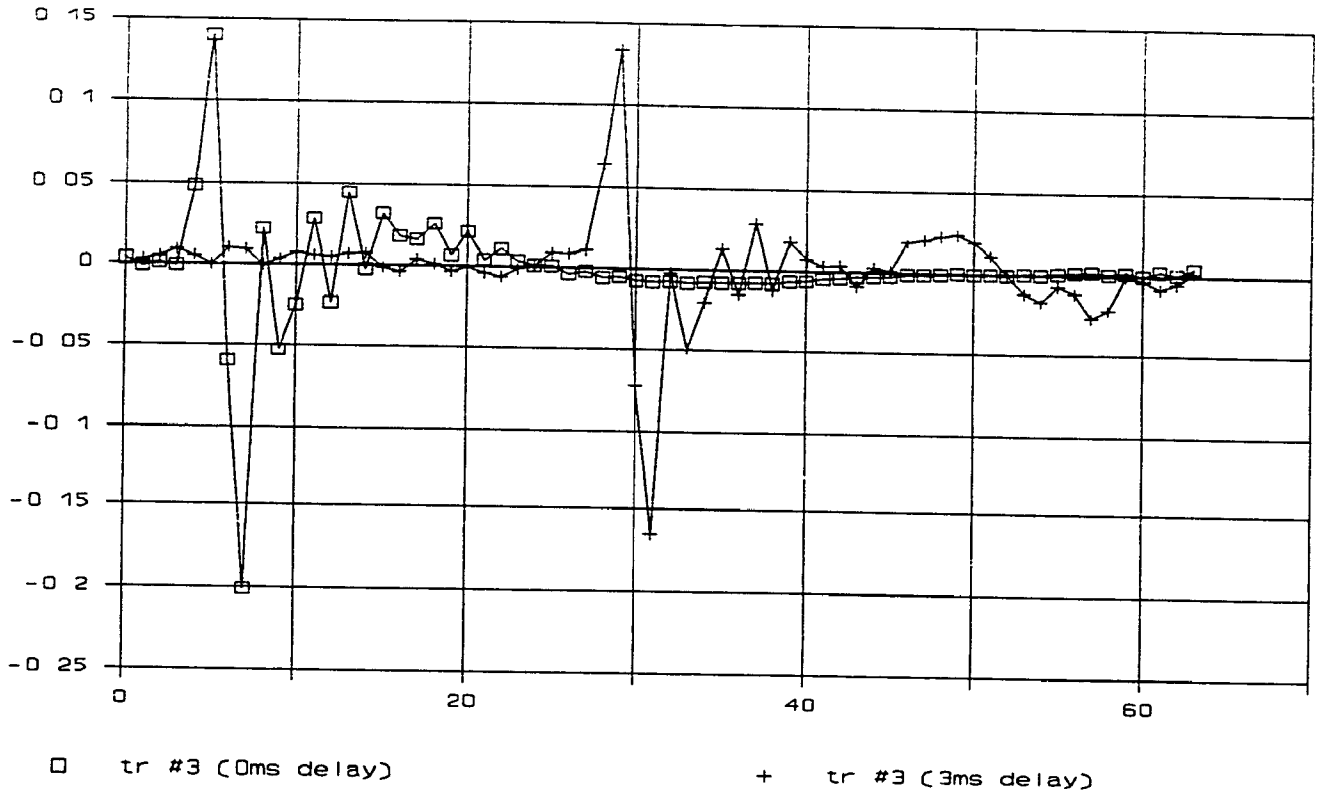


Figure 21. Adaptive Characteristic Test of the MCDAEC

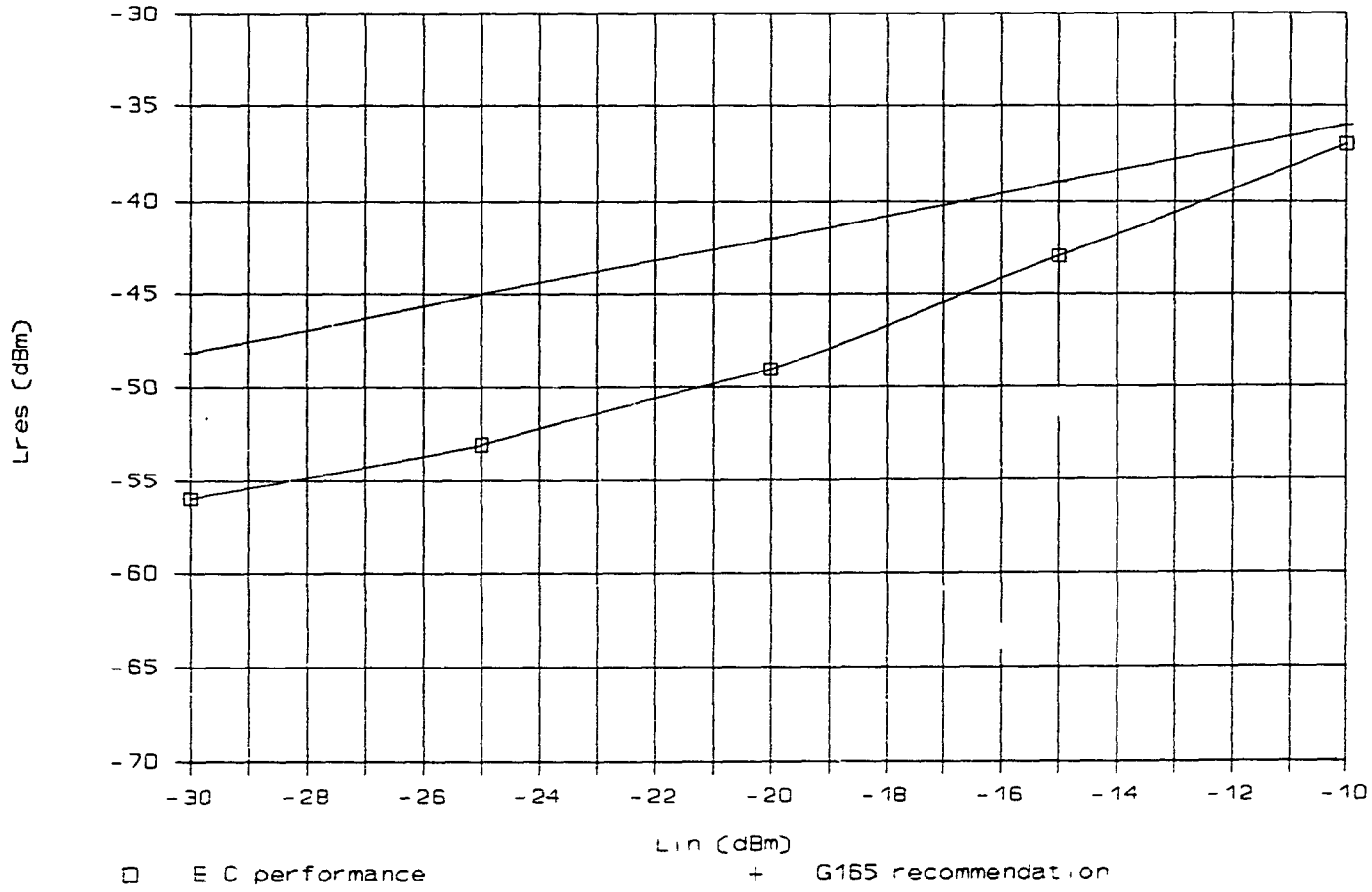


Figure 22. Steady State Residual Error

the echo canceller exceeds the G165 recommended performance for all values of the input signal. Fig.22 also shows that the smaller the input signal is, the better the echo canceller performs.

- Testcase #2 : Convergence

This test verifies that the echo canceller converges to at least 27dB within 500ms. The test conditions are as follows

- coefficients set to 0.
- NLP is enabled.
- Input signal Lin = -15dBm.
- Noise signal Ln = -10dBm.

The echo cancellation is enabled for 500ms. At the end of this interval, the echo return loss (ERL) is recorded.

Test result : ERL = 45dB after 500ms.

- Testcase #3 : Double Talk

This test verifies that the echo canceller does not diverge in the presence of double talk, i.e the residual error should not increase more than 10dB over the steady state level in testcase #1. The test conditions are as follows :

- Echo canceller fully converged.
- Input signal Lin = -10dBm.
- Noise signal Ln = -10dBm.
- NLP is disabled.
- ERL = 18dB.

Test result : The residual error Lres = -37dBm.

The steady state residual error as obtained in testcase #1 was about -37.5dBm (Fig.22). Residual error increased by about 0.5dB which is within the acceptable range of 10dB.

Chapter 5

CONCLUSION

In this thesis, a major work was done in investigating echo cancellation algorithms, and structures of digital signal processors (DSPs). Echo cancellation can be classed in two categories : Least-Mean Squared (LMS), and Least Squares (LS).

Conventional, well-known algorithm LMS has the minimum number of computation per iteration, but a relatively slow speed. In addition, its performance depends on the characteristics of input signal : a highly correlated signal such as voice converges slower white-noise. To solve this the gain, μ, has been allowed to vary inversely to the input signal power. Modified algorithms have also been developed such as the Block Least Mean Square (BLMS) in which a group of coefficients are updated per iteration. This approach is particularly useful in reducing the access of external memory. A second category of Echo Cancellation algorithm, the Least Squares (LS), has fast initial convergence speed but a complicated implementation.

Requirements of the MCADEC : as many channel per DSP as possible with a real-time constraint of 125 μs, the BLMS was chosen to implement using DSP56001.

The MCDAEC requires a multiprocessor configuration with one Host as the low cost Z80 and 20 DSP's as Slaves. The DSP56001 with its powerful instruction set, and multiprocessor interface features is the selected DSP in this application.

Using a statistical approach, status of trunks are dynamically updated and given to the corresponding DSP. Fast interrupts were used for this purpose because of their overhead- free structure. In addition, for maintaince purposes, the operational status of the Host is monitored by a watchdog timer. Infinite loop or software undefined conditions cause a wrong watchdog timer cycle which would be reset by the main CPU of the system, and appropriate actions were taken. Experimental results showed that the convergence rate of the MCADEC is within the 500ms as required by the CCITT G.165. Echo Return Loss Enhancement (ERLE) exceeded the -27dB as required.

REFERENCES

[1]. Tho Le-Ngoc, *A Random-Requested Demand-Assigned Multiple Access Protocol for Point-to-Multipoint Radio Systems*, ICC'85, June 23-26, Chicago, Illinois.

[2]. M.M.Sondhi and D.A.Berkley, *Silencing Echoes on the Telephone Network*, Proc. IEEE, Vol.68, No.8, Aug. 1980.

[3]. D.L.Duttweiler, *A Twelve-Channel Digital Echo Canceller*, IEEE Trans. on Communications, Vol.COM-26, No.6, May 1978.

[4]. CCITT Recommendation, *Echo Cancellers*, Red Book, Vol.III, Rec. G.165.

[5]. Simon Haykins, *Signal Processing Algorithms*,

[6]. B.Widrow and S.D.Stearn, *Adaptive Signal Processing*, 1985.

[7]. C.W.K.Gritton and D.W.Lin, *Echo Cancellation Algorithms*, IEEE ASSP Magazine, April 1984.

[8]. M.J.Gingell & B.G.Hay, *A Block Mode Update Echo Canceller using Custom LSI*, IEEE 1983.

[9]. G.A.Clark et al., *Block Implementation of Adaptive Digital Filter*, IEEE ASSP, Vol. ASSP-29, No.3, June 1981.

[10]. *Digital Signal Processing Applications using the TMS320 Family*.

[11]. Verhoeckx et al., *Digital echo cancellation for baseband transmission*, IEEE ASSP-27, 1979.

[12]. D.D.Falconer and L.Ljung, *Application of Fast Kalman Estimation to Adaptive Equalization*, IEEE Trans. on

Communication, Vol.COM-26, Oct. 1978.

[13]. Xixian Chen and Guangxin Yue, *A Microprocessor-Based Memory-Tap Echo Canceller*, IASSP 86, Tokyo.

[14]. S.Kawamura and M.Hatori, *A Tap Selection Algorithm For Adaptive Filters*, ICASSP 86, Tokyo.

[15]. R.Montagna and L.Nebbia, *A Fast Adaptive Echo Canceler with delay estimation for time variant telephone circuits*, IEEE Globecom '84 Conf. Rec., Vol.3, Nov 1984.

[16]. Yoshikazu Mon et al., *Architecture of High-Speed 32-bit Floating-point Digital Signal Processor*, IASSP 86, Tokyo.

[17]. Bill Eichen, *NEC's A μPD77230 Digital Signal Processor*, IEEE Micro, Dec. 1986.

[18]. Frank Ferro and Kreg Ulery, *The Architecture and Programming of the WEDSP 16 Digital Signal Processor*, Mini Micro 1987.

[19]. K.S.Lin et al., *The TMS320 Family of Digital Signal Processors*, Proc. IEEE, Vol.75, No.9, Sept. 1987.

[20]. Kevin L.Kloker, *The Motorola DSP56000 Digital Signal Processor*, IEEE Micro, Dec. 1986.

[21]. T.Le-Ngoc, T.S.Nguyen, and D.Konforti, *A Multichannel Demand-Assignment for Point-to-Multipoint Subscriber Radio Systems*, Canadian Conference on Electrical and Computer Engineering, 1989.

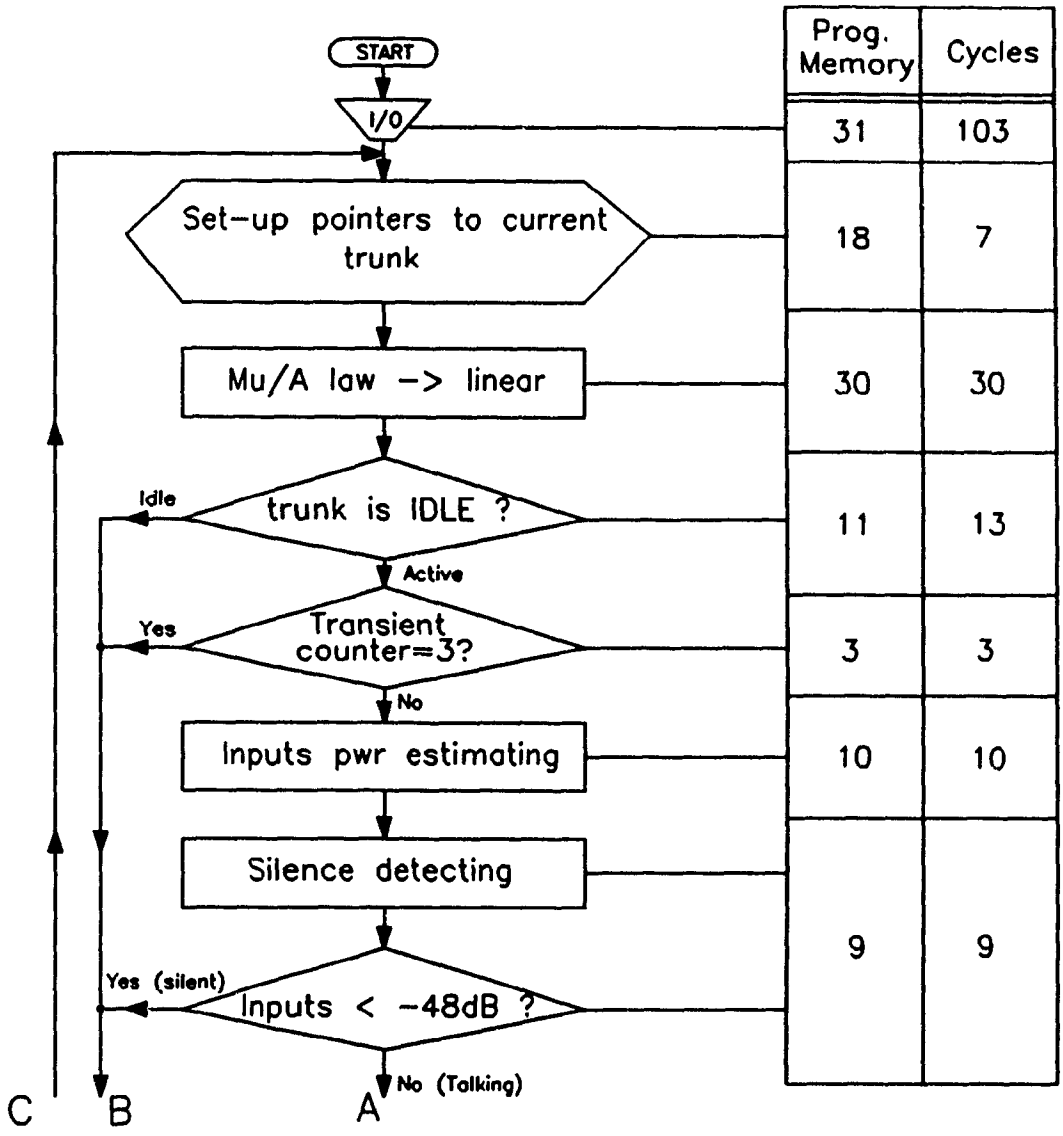
APPENDIX A.

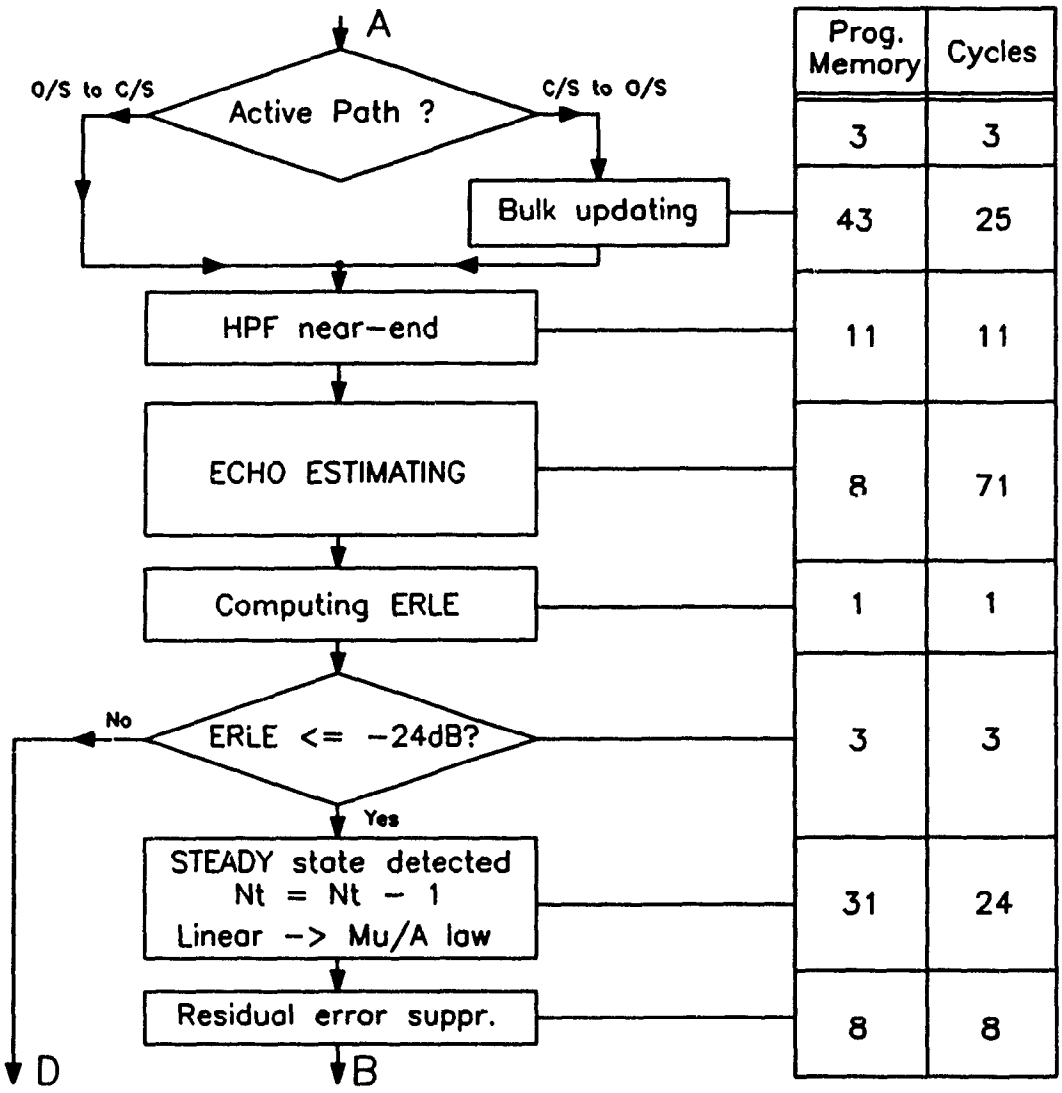
A MEMORY MAP FOR ECHO CANCELLER

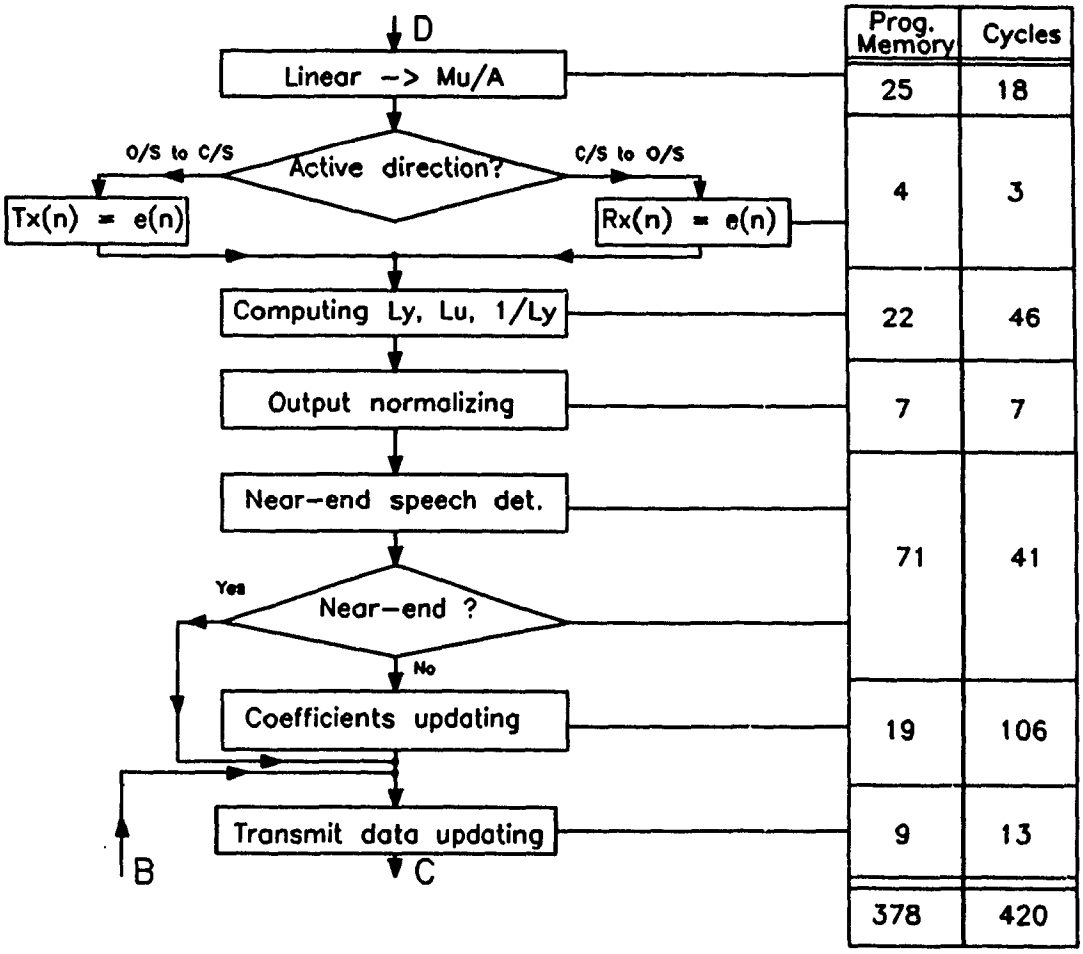
ABBRV.	DESCRIPTION
RX0R	μ (or A) law Rx (i)
RX0L	Linear Rx (i)
TX0R	μ (or A) law Tx (i)
TX0L	Linear Tx (i)
STAT	CPU command
ABSTXF	Short-tau estimated power of Tx (i)
ABSRXF	Short-tau estimated power of Rx (i)
ACTDIR	currently active direction (C/S -> O/S = 1)
BULKDEL	Bulk delay counter
R6	Bulk delay pointer
R4	Reference sample buffer pointer
S1DC	Linear Near-end sample
S0	Near-end sample after HPF
ABSS0	Absolute HPF Near-end sample
ABSOUT	Long-tau residual error
1/Ly	Inverse of Far-end power
HCNTR	Hang-over counter
Ly	Long-tau estimated power of Far-end sample
R2	Residual error buffer pointer
ABSS0F	Short-tau estimated Near-end power
ABSY0F	Short-tau estimated Far-end power
H	Block Least Mean Square counter
M0	Local Maxima (i)
M1	Local Maxima (i-1)
M2	Local Maxima (i-2)
M3	Local Maxima (i-3)

APPENDIX B.

A FLOWCHART OF THE MCDAEC




A
O/S to C/S
Active Path ?
C/S to O/S
Bulk updating
HPF near-end
ECHO ESTIMATING
Computing ERLE
No
ERLE <= -24dB?
Yes
STEADY state detected
Nt = Nt - 1
Linear -> Mu/A law
Residual error suppr.
D
B
Prog. Memory
Cycles
3
3
43
25
11
11
8
71
1
1
3
3
31
24
8
8


D
Linear –> Mu/A
O/S to C/S
Active direction?
C/S to O/S
Tx(n) = e(n)
Rx(n) = e(n)
Computing Ly, Lu, 1/Ly
Output normalizing
Near–end speech det.
Yes
Near–end ?
No
Coefficients updating
Transmit data updating
B
C
Prog. Memory
Cycles
25
18
4
3
22
46
7
7
71
41
19
106
9
13
378
420

APPENDIX C.

A PROGRAM LISTING OF THE MCDAEC

Version : ECMU_NZ.V2, "Mu-law, zero suppression"

Echo Canceller

```
2                       Page   132,,1,1
3                 ;*****************************************************************************
4                 ;*          A Multichannel, Demand-Assignment Echo Canceller          *
5                 ;*                    for 3 full-duplex channels                     *
6                 ;*************************** *************************************************
7                 ;
8                 ;  Algorithm constants
9                 ;
10      00000010    bl_size   equ    16                  ;BLMS block size = 16
11      00008004    cutoff    equ    $008004               ;cutoff threshold (-48 dB)
12      00000060    delay     equ    96                  ;size of bulk delay buffer
13      00000258    hangt     equ    75000/125             ;hang-over counter = 75ms
14      0.031250    htau      equ    .03125              ;2**(-5)
15      00000040    ntaps     equ    64                  ;filter length = 64
16      00008004    silence   equ    $008004               ;silence threshold (-48 dB)
17      0008137F    thre      equ    $08137f               ;suppression threshold (-24 dB)
18      0.003906    shift_8   equ    0.0039063             ;demultiplex parameters
19      00000080    shift_16  equ    $80
20      000000FF    maskoff   equ    $ff
21      0000FF00    mask_2    equ    $00ff00
22      000000FF    mask_3    equ    $ff
23      00003600    scitab    equ    $3600               ;SCI conversion table
24                ;-------------------------
25                ;  Equates for I/O programming
26                ;-------------------------
27      0000FFFE      m_bcr     equ    $fffe               ;port A Bus Control register
28      0000FFE0      m_pbc     equ    $ffe0               ;port B control register
29      0000FFE1      m_pcc     equ    $ffe1               ;port C control register
30      0000FFE3      m_pcddr   equ    $ffe3
31                ;-------------------------
32                ;  Equates for Host Interface
33                ;-------------------------
34                ;  Register addresses
35
36      0000FFEB      m_hrx     equ    $ffeb               ;Host received data register
37      0000FFEB      m_trx     equ    $ffeb               ;Host transmit data register
38      0000FFE8      m_hcr     equ    $ffe8               ;Host control register
39                ;-------------------------------------------
40                ;  Equates for Serial Communications Interface (SCI)
```

```
41                ;-----------------------------------------
42                ; Register addresses
43
44      0000FFF4    m_srxl   equ   $fff4          ;SCI receive data register
45      0000FFF4    m_stxl   equ   $fff4          ;SCI transmit data register
46      0000FFF0    m_scr    equ   $fff0          ;SCI interface control register
47      0000FFF2    m_sccr   equ   $fff2          ;SCI clock control register
48      0000FFF1    m_ssr    equ   $fff1
49                ;-----------------------------------------
50                ; Equates for Synchronous Serial Interface (SSI)
51                ;-----------------------------------------
52                ; Register addresses
53
54      0000FFEF    m_rx     equ   $ffef          ;Serial Receive data register
55      0000FFEF    m_tx     equ   $ffef          ;Serial Transmit data register
56      0000FFEC    m_cra    equ   $ffec          ;SSI Control register A
57      0000FFED    m_crb    equ   $ffed          ;SSI Control register B
58      0000FFEE    m_sr     equ   $ffee          ;SSI Status register
59                ;---------------------------------
60                ; Equates for Exception processing
61                ;---------------------------------
62                ; Register addresses
63      0000FFFF    m_ipr    equ   $ffff          ;Interrupt Priority register
64                ;--------------
65                ; Global data
66                ;--------------
67      00000000    abs_rx   equ   0             ;|Rx(i)|
68      00000001    abs_tx   equ   1             ;|Tx(i)|
69      00000002    absy0f   equ   2
70      00000003    abss0f   equ   3
71      00000004    cutof    equ   4             ;-48 dB
72      00000005    dir_new  equ   5             ;temporary storage of act. dir
73      00000006    minus6   equ   6             ;-6
74      00000011    nt       equ   17            ;number of trunks in TRANSIENT
75      00000008    one      equ   8             ;1
76      00000009    renorm   equ   9             ;4096
77      0000000A    resec    equ   10            ;residual error
78      0000000B    thres    equ   11            ;threshold error
79      0000000C    tmp_Ly   equ   12            ;temporary storage of far-end power
80      0000000D    tmp_iaby equ   13            ;temporary storage of 1/Ly(i)
81      0000000E    tmp_rx   equ   14            ;temporary storage of linear Rx
82      0000000F    tmp_tx   equ   15            ;temporary storage of linear Tx
83      00000010    tmp_s0   equ   16            ;temporary storage of Lx(i)
84      00000007    wdflin   equ   7             ;watchdog flag in
```

```
85    000000D0     wdflot   equ    $d0                    ;watchdog flag out
86                 ;-----------------------
87                 ;  Local channel data
88                 ;-----------------------
89    00000012     ch1_dat  equ    18                     ;data for channel #1 (26 loc.data)
90    0000002C     ch2_dat  equ    44                     ;data for channel #2
91    00000046     ch3_dat  equ    70                     ;data for channel #3
92                 ;-----------------------------
93                 ;  Normalized residual errors
94                 ;-----------------------------
95    X:0060             org    x:96
96    X:0060       res_er3  dsm    bl_size-1              ;          --//-- trunk 3
97    X:0080       coef_r3  dsm    ntaps                  ; coef. of 3rd trunk (C/S -> O/S)
98    X:00C0       coef_t3  dsm    ntaps                  ;          "        (O/S -> C/S)
100                ;-------------------------------------------- ---;
101                ; 3 sets of full-duplex Echo canceller coefficients ;
102                ;-------------------------------------------- ----;
103   X:2000             org    x:$2000
104   X:2000       res_er1  dsm    bl_size-1              ;16 prev. norm. residual trunk #1
105   X:2010       res_er2  dsm    bl_size-1              ;          "        trunk #2
106   X:2040       coef_r1  dsm    ntaps                  ;coef. of 1st trunk (C/S -> O/S)
107   X:2080       coef_t1  dsm    ntaps                  ;          "        (O/S -> C/S)
108   X:20C0       coef_r2  dsm    ntaps                  ;coef. of 2nd trunk (C/S -> O/S)
109   X:2100       coef_t2  dsm    ntaps                  ;          "        (O/S -> C/S)
111
112   Y:0000             org    y:0
113
114   Y:0000       ref_t1   dsm    ntaps+bl_size-1          ;reference samples of 1st trunk
115   Y:0080       ref_t2   dsm    ntaps+bl_size-1          ;          "        2nd
116
117   Y:2140             org    y:$2140
118
119   Y:2180       bulk_1   dsm    delay-1                ;bulk delay of trunk #1 (12ms)
120   Y:2200       bulk_2   dsm    delay-1                ;          --"--    #2 (12ms)
121   Y:2280       bulk_3   dsm    delay-1                ;          --"--    #3 (12ms)
122   Y:2300       ref_t3   dsm    ntaps+bl_size-1
123
```

```
124          ;=============================================================
125            ;       Macro 1 : μ-law to Linear conversion          ;
126            ;       ------------------------------------          ;
127            ;Input : in the 8 most significant bits of a1. The remain- ;
128            ;       bits of a are ignored.                        ;
129            ;Output :in the 56 bit accumulator a. The linear fraction ;
130            ;       is in the 13 most significant bits of a1, and the ;
131            ;       11 least significant bits are zeroes.         ;
132          ;=============================================================
133            mulin    macro
134 m           _shift   equ   $80                  ;shift constant
135 m           _mutable equ   $100                 ;base address of μ-law table
136 m           ;
137 m                    move  a1,b                 ;save input
138 m                    lsl   a       #>_shift,x0  ;shift out sign bit, get shift
139 m                    lsr   a       #_mutable,x1 ;shift in zero, get table address
140 m                    tfr   x1,a    a1,x1        ;swap table base, and offset
141 m                    mac   x1,x0,a              ;shift offset down, add to base
142 m                    move  a,r5
143 m                    move  x:one,y0             ;1
144 m                    lsl   b       x:(r5),a     ;c=sign bit, look-up linear data
145 m                    neg   a       a,b          ;a=neg result, b=pos result
146 m                    tcs   b,a     r7,r5        ;if pos sign, correct result
147 m                    endm
148          ;=============================================================
149            ;       Macro 2 : Linear to μ-law conversion          ;
150            ;       ------------------------------------          ;
151            ;Input : 56 bit accumulator a (13 bits linear fraction are ;
152            ;       in the most significant bits of a1)           ;
153            ;Output :in the 8 most significant bits of a1. The 16 LSB's;
154            ;       of a1 are zero.                               ;
155          ;=============================================================
156            linmu    macro
157 m           _bias    equ   $008400              ;absolute bias = 33
158 m           ;
159 m                    tfr   a,b     a,a          ;save & limit input data
160 m                    abs   a       #_bias,x0    ;form input sign magnitude, bias
161 m                    add   x0,a    #7,r1        ;add bias to magnitude, get chord
162 m                    move  a,a                  ;limit again
163 m                    rep   #7                   ;find chord number by normalizing
164 m                    norm  r1,a                 ;biased magnitude to get step
165 m                    asl   a                    ;isolate step number
166 m                    asl   a       b,b          ;limit input again
167 m                    neg   b       r1,a2        ;invert sign bit, get chord number
```

```
168 m                   asr     a                       ;combine chord and step
169 m                   asr     a
170 m                   asr     a
171 m                   not     a                       ;invert 7 LSB's for μ-law
172 m                   asl     b                       ;get sign bit
173 m                   ror     a       #<$ff,x0            ;combine sign, chord and step
174 m                   and     x0,a                    ;clear 16 LSB's
175 m                   endm
177             ;-----------------------------------------------;
178             ;               Synchronization routine                 ;
179             ;-----------------------------------------------;
180     P:0040          org     p:$40
181             sync
182     P:0040 069680   do      #150,dum
            000042
183     P:0042 000000   nop
184             dum
185             ;  initializing all I/O registers
186     P:0043 08F4A1   movep           #$01e7,x:<<m_pcc        ;port C is SSI & SCI configured
            0001E7
187     P:0045 54F400   move            #$0c004a,a1
            0C004A
188     P:0047 070A0C   movem           a1,p:$0a
189             irqb_w1
190     P:0048 000086   wait
191     P:0049 0C0048   jmp     <irqb_w1
192             ;-----------------------------------------------;
193             ;                       I/O routine                     ;
194             ;-----------------------------------------------;
195             begin
196             ;******* start of TIN1 *******
197     P:004A 44F400   move            #>$80,x0
            000080
198     P:004C 45AE00   move            x:46,x1
199     P:004D 45F4A8   mpy     x0,x1,b  #>$ff,x1
            0000FF
200     P:004F 70F46E   and     x1,b     #scitab,n0
            003600
201     P:0051 21B000   move            b1,r0
202     P:0052 469400   move            x:20,y0             ;save previously corrected Tx(1)
203     P:0053 4DE800   move                        y:(r0+n0),x1
204     P:0054 45F4A8   mpy     x0,x1,b  #>$ff,x1
            0000FF
205     P:0056 20006E   and     x1,b
```

```
206                ;****** start of TIN2 *******
207     P:0057 0AB181 w_1      jclr   #1,x:<<m_ssr,w_1
            000057
208     P:0059 08CD34          movep          b1,x:<<m_stxl          ;transmit Tx(2)
209     P:005A 0AB182 w_2      jclr   #2,x:<<m_ssr,w_2
            00005A
210     P:005C 085034          movep          x:<<m_srxl,r0          ;read Tx(1)
211     P:005D 000000          nop                                   ;LSB's -> MSB's conversion
212     P:005E 5EE800          move                  y:(r0+n0),a
213     P:005F 45F000          move           x:72,x1
            000048
214     P:0061 45F4A8          mpy    x0,x1,b  #>$ff,x1
            0000FF
215     P:0063 56146E          and    x1,b     a,x:20
216     P:0064 21B000          move           b1,r0
217     P:0065 000000          nop
218     P:0066 4DE800          move                  y:(r0+n0),x1
219     P:0067 45F4A8          mpy    x0,x1,b  #>$ff,x1
            0000FF
220     P:0069 20006E          and    x1,b
221                ;******* start of TIN3 ********
222     P:006A 0AB181 w_3      jclr   #1,x:<<m_ssr,w_3
            00006A
223     P:006C 08CD34          movep          b1,x:<<m_stxl          ;transmit Tx(3)
224     P:006D 0AB182 w_4      jclr   #2,x:<<m_ssr,w_4
            00006D
225     P:006F 085034          movep          x:<<m_srxl,r0          ;read Tx(2)
226     P:0070 000000          nop                                   ;LSB's -> MSB's conversion
227     P:0071 5EE800          move                  y:(r0+n0),a
228     P:0072 45F4D8          mpy    x0,y0,b  #>$ff,x1
            0000FF
229     P:0074 562E6E          and    x1,b     a,x:46
230     P:0075 21B000          move           b1,r0
231     P:0076 000000          nop
232     P:0077 4DE800          move                  y:(r0+n0),x1
233     P:0078 45F4A8          mpy    x0,x1,b  #>$ff,x1
            0000FF
234     P:007A 26FF6E          and    x1,b     #maskoff,y0
235                ;******* TIN4 *******
236     P:007B 0AB181 w_5      jclr   #1,x:<<m_ssr,w_5
            00007B
237     P:007D 08CD34          movep          b1,x:<<m_stxl          ;transmit Tx(1)
238     P:007E 0AB182 w_6      jclr   #2,x:<<m_ssr,w_6
            00007E
```

```
239     P:0080 085034       movep       x:<<m_srxl,r0       ;read Tx(3)
240     P:0081 44F400       move        #>128,x0
            000080
241     P:0083 5FE800       move                y:(r0+n0),b
242                                             ;       Rx(1) Rx(2) Rx(3)
243     P:0084 084C2F       movep       x:<<m_rx,a1         ;read SSI = $ x x  x x  x x
244     P:0085 218500       move        a1,x1
245     P:0086 577056       and   y0,a  b,x:72              ;Tx(3)
            000048
246     P:0088 5412A8       mpy   x0,x1,b  a1,x:18          ;Rx(1) = $xx0000
247     P:0089 212C00       move        b0 a1
248     P:008A 212556       and   y0,a  b0,x1
249     P:008B 542CA8       mpy   x0,x1,b  a1,x:44          ;Rx(2) = $xx0000
250     P:008C 212C00       move        b0,a1
251     P:008D 448756       and   y0,a  x:wdflin,x0
252     P:008E 547000       move        a1,x:70             ;Rx(3) = $xx0000
            000046
253
254             ;*** Watchdog timer ***
255
256     P:0090 4C7000       move                x0,y:wdflot
            0000D0

258             ;-------------------------------------------------------
259             ;               ECHO CANCELLING ROUTINE
260             ;r0 -> Rx coefficients      r4 -> reference buffer
261             ;r1 -> Tx coefficients      r5 ->
262             ;r2 ->                      r6 -> bulk delay buffer
263             ;r3 -> residual errors buffer   r7 -> local channel data
264             ;Time taken : 8 cycles
265             ;-------------------------------------------------------
266             process
267     P:0092 54F400       move        #$0c009c,a1         ;{13 cycles}
            0C009C
268     P:0094 07708C       movem       a1,p:$1fd
            0001FD
269     P:0096 60F400       move        #coef_r1,r0
            002040
270     P:0098 61F400       move        #coef_t1,r1
            002080
271     P:009A 371200       move        #ch1_dat,r7         ;local data ptr.adrs of channel #1
272     P:009B 0C00C1       jmp   <ec64                     ;process channel #1
273             tr2
274     P:009C 54F400       move        #$0c00a6,a1         ;{13 cycles}
```

```
          0C00A6
275       P:009E 07708C        movem       a1,p:$1fd
          0001FD
276       P:00A0 60F400        move        #coef_r2,r0
          0020C0
277       P:00A2 61F400        move        #coef_t2,r1
          002100
278       P:00A4 372C00        move        #ch2_dat,r7          ;local data ptr.adrs of channel #2
279       P:00A5 0C00C1        jmp    <ec64                     ;process channel #2
280                 tr3
281       P:00A6 54F400        move        #$0c00ae,a1          ;{11 cycles}
          0C00AE
282       P:00A8 07708C        movem       a1,p:$1fd
          0001FD
283       P:00AA 308000        move        #coef_r3,r0
284       P:00AB 31C000        move        #coef_t3,r1
285       P:00AC 374600        move        #ch3_dat,r7          ;local data ptr.adrs of channel #3
286       P:00AD 0C00C1        jmp    <ec64                     ;process channel #3
287                 ;****** merge Rx(1) & Rx(2) & Rx(3) *******
288       P:00AE 47AC13        clr    a    x:44,y1
289       P:00AF 46F400        move        #shift_8,y0
          008000
290       P:00B1 4792B0        mpy    y0,y1,a  x:18,y1
291       P:00B2 44F400        move        #mask_2,x0
          00FF00
292       P:00B4 44F446        and    x0,a     #>mask_3,x0      ;A1 = $00xx00
          0000FF
293       P:00B6 47F072        or     y1,a     x:70,y1          ;merge Rx(1) & Rx(2) , A1 = $xxxx00
          000046
294       P:00B8 46F400        move        #>shift_16,y0
          000080
295       P:00BA 2000B8        mpy    y0,y1,b
296       P:00BB 20004E        and    x0,b                      ;B1 = $0000xx
297       P:00BC 21A700        move        b1,y1
298       P:00BD 200072        or     y1,a                      ;merge Rx(1) & Rx(2) & Rx(3)
299       P:00BE 08CC2F        movep       a1,x:<<m_tx          ;****SSI transmit*****
300                 irqb_w2
301       P:00BF 000086        wait                             ;next IRQB ?
302       P:00C0 0C00BF        jmp    <irqb_w2
```

```
304                ;-----------------------------------------------------------
305                ;            μ law -> linear conversion Routine
306                ;            ----------------------------------
307                ; Input : 8 MSB bits of a1.
308                ; Output : 13 MSB bits of a1.
309                ; Alter registers : r7 -> local data
310                ; Time taken : 32 cycles
311                ;-----------------------------------------------------------
312                ec64
313                ;{16 cycles}
314                ;  Convert μ law Rx to linear  {16 cycles}
315                ;
316   P:00C1 54DF00          move          x:(r7)+,a1          ;PCM μ law Rx stored in acc.
317                       mulin                             ;μ law -> Linear
331   P:00CE 545F00          move          a1,x:(r7)+
332   P:00CF 540E26          abs    a      a1,x:tmp_rx         ;compute |Rx(n)|
333   P:00D0 560000          move          a,x:abs_rx
334                ;{15 cycles}
335                ;  Convert μ law Tx to linear  {16 cycles}
336                ;
337   P:00D1 54DF00          move          x:(r7)+,a1          ;PCM μ law Tx stored in acc.
338                       mulin                             ;μ law -> Linear
352   P:00DE 540F26          abs    a      a1,x:tmp_tx         ;compute |Tx(n)|
353   P:00DF 560100          move          a,x:abs_tx



355                ;-----------------------------------------------------------
356                ;                Trunk detection routine
357                ;                -----------------------
358                ;Tasks performed :
359                ; - TRANSIENT/STEADY/WAIT -> IDLE : bypass E.C process
360                ; - IDLE -> ACTIVE : trunk is in TRANSIENT state (Nt = Nt+1)
361                ;Alter registers : r5,r7
362                ;Time taken : 12 cycles
363                ;Program words : 9 words
364                ;-----------------------------------------------------------
365                trunk
366                ;{5 cycles}
367   P:00E0 0A5F80          jclr   #0,x:(r7)+,idle             ;IDLE : bypass E.C process
          0001F2
368   P:00E2 579100          move          x:nt,b
369   P:00E3 22F500          move          r7,r5               ;save r7
```

```
370             ;{7 cycles}
371             ;  Current trunk active : updating nt
372             ;
373     P:00E4 0A67A4       jset    #4,x:(r7),wait_st
            0000E9
374     P:00E6 0A6724       bset    #4,x:(r7)               ;IDLE -> TRANSIENT : update Nt
375     P:00E7 200058       add     y0,b
376     P:00E8 571100       move            b,x:nt          ;Nt = Nt + 1
378             ;-------------------------------------------------------------
379             ;               Wait state routine
380             ;               ------------------
381             ;Tasks performed :
382             ; - Check status of the E.C, and make decision on putting the
383             ;   most recently arriving trunk (in WAIT state )
384             ;Alter register : r7
385             ;Time taken : 10 cycles
386             ;Program words : 9 words
387             ;-------------------------------------------------------------
388             wait_st
389             ;{6 cycles}
390     P:00E9 579100       move            x:nt,b          ;get transient counter nt
391     P:00EA 44F400       move            #>3,x0
            000003
392     P:00EC 20004D       cmp     x0,b                    ;3 trunks in TRANSIENT state ?
393     P:00ED 0E10F0       jge     <set_1                  ;Yes
394             set_0
395             ;{4 cycles}
396     P:00EE 0A5F05       bclr    #5,x:(r7)+              ;reset WAIT flag
397     P:00EF 0C00F4       jmp     <pwr_est
398             set_1
399             ;{4 cycles}
400     P:00F0 0A5F25       bset    #5,x:(r7)+              ;set WAIT flag
401     P:00F1 20005C       sub     y0,b                    ;Nt = Nt - 1
402     P:00F2 571100       move            b,x:nt
403     P:00F3 0C01FD       jmp     <waitt                  ;bypass E.C process
```

```
405                ;-----------------------------------------------------------------
406                ;                    Input power estimation
407                ;                    ----------------------
408                ;For each trunk a pair of signals Tx(C/S -> O/S), and Rx (O/S
409                ;-> C/S) are received. The algorithm for estimation is as foll-
410                ;ows :
411                ;
412                ; ~Tx(i) = (1-α)~Tx(i-1) + α|Tx(i-1)|
413                ; ~Rx(i) = (1-α)~Rx(i-1) + α|Rx(i-1)|   , α = 2**(-5)
414                ;
415                ; Input : ~Rx(i-1), ~Tx(i-1), |Rx(i-1)|, |Tx(i-1)|
416                ; Output : ~Rx(i), ~Tx(i)
417                ; Alter registers : r7 -> ~Tx(n), ~Rx(n)
418                ; Time taken : 10 cycles
419                ; Program words : 10 words
420                ;-----------------------------------------------------------------
421                pwr_est
422                ;{5 cycles}
423                ;  Estimate ~Tx(i) {5 cycles}
424                ;
425   P:00F4 57E700        move          x:(r7),b          ;get ~Tx(i-1)
426   P:00F5 240414        sub   b,a     #<$04,x0          ;compute |Tx(i-1)| - ~Tx(i-1)
427   P:00F6 21C700        move          a,y1
428   P:00F7 4780CB        macr  x0,y1,b x:abs_rx,y1       ;~Tx(i-1)+ α{|Tx(i-1)| -~Tx(i-1)}
429   P:00F8 575F00        move          b,x:(r7)+         ;update ~Tx(i)
430                ;{5 cycles}
431                ;  Estimate ~Rx(i) {5 cycles}
432                ;
433   P:00F9 1FA771        tfr   y1,a    x:(r7),b  b,y1    ;get |Rx(i)|, ~Rx(i-1), save ~Tx(i)
434   P:00FA 3F0314        sub   b,a     #3,n7             ;compute |Rx(i-1)| - ~Rx(i-1)
435   P:00FB 21C500        move          a,x1
436   P:00FC 4584AB        macr  x0,x1,b x:cutof,x1        ;~Rx(i-1)+ α{|Rx(i-1) - ~Rx(i-1)}
437   P:00FD 575F71        tfr   y1,a    b,x:(r7)+         ;update ~Rx(i)
```

```
439             ;-----------------------------------------------------------------
440             ;                   Silence detection routine
441             ;                   -------------------------
442             ;In this routine ~Tx(i) and ~Rx(i) are compared with silence
443             ;threshold of -48 dB. Silence is detected when both subscribers
444             ;power level are < -48 dB.
445             ;Alter register : none
446             ;Time taken : 9 cycles
447             ;Program words : 6 words
448             ;-----------------------------------------------------------------
449             silent
450             ;{6 cycles}
451             ;   Compare ~Rx(i) with Silence threshold (-48 dB)
452             ;
453   P:00FE 44F46D       cmp    x1,b    #>delay,x0          ;~Rx(i) < -48dB ?
        000060
454   P:0100 0E9102       jlt    <nxt_test                   ;Silence, test next subscriber
455   P:0101 0C0104       jmp    <act_path                    ;Talking
456             nxt_test
457             ;{3 cycles}
458   P:0102 200065       cmp    x1,a                        ;~Tx(i) < -48dB ?
459   P:0103 0E91FD       jlt    <trans_det                   ;SILENCE detected, bypass E.C
461             ;-----------------------------------------------------------------
462             ;                   Active path detection
463             ;                   ---------------------
464             ;~Rx(n) < ~Tx(n)  : C/S -> O/S
465             ;~Rx(n) => ~Tx(n) : O/S -> C/S
466             ; Alter registers : r7 -> active direction
467             ; Time taken : 13 cycles
468             ; Program words : 13 words
469             ;-----------------------------------------------------------------
470             act_path
471             ;{3 cycles}
472             ;   Compare two signals  {3 cycles}
473   P:0104 47880D       cmp    a,b     x:one,y1             ;~Rx(i) < ~Tx(i) ?
474   P:0105 0E7124       jgt    <hpf_1                      ;dir O/S -> C/S, E.C(Rx) active
475                                                          ;no bulk delay in this direction
476             ;{1 cycle}
477             ;   Active direction C/S -> O/S (ACTDIR = 1)  {1 cycle}
478   P:0106 45E700       move           x:(r7),x1            ;save previous active direction
479             ;{5 cycles}
480             ;   Direction changed ?  {4 cycles}
481   P:0107 475F71       tfr    y1,a    y1,x:(r7)+
482   P:0108 54E765       cmp    x1,a    x:(r7),a1
```

```
483   P:0109 0EA10B        jeq    <upd_bulk                    ;No
484             ;{1 cycle}
485             ;  Yes, reinitialize bulk delay counter  {1 cycle}
486   P:010A 446741        tfr    x0,a     x0,x:(r7)           ;bulk delay = 96
487             ;{6 cycles}
488             ;  Update bulk delay counter {3 cycles for E.C}
489             upd_bulk
490   P:010B 205F03        tst    a        (r7)+               ;bulk delay complete ?
491   P:010C 0EA115        jeq    <hpf                         ;Yes, performing Echo Cancelling
492   P:010D 205774        sub    y1,a     (r7)-               ;bulk counter = bulk counter - 1
493   P:010E 565F00        move            a,x:(r7)+
494   P:010F 000000        nop                                 ;pipeline delay
496             ;-------------------------------------------------------------------
497             ;                  Bulk delay routine
498             ;                  ------------------
499             ;Alter registers : r6 -> bulk delay buffer pointer
500             ;                  r7 -> bulk delay pointer
501             ;Time taken : 6 cycles
502             ;Program words : 5 words
503             ;-------------------------------------------------------------------
504             bulk
505             ;{1 cycle}
506             ;  Retrieve r6  {1 cycle}
507   P:0110 66E700        move            x:(r7),r6
508             ;{5 cycles}
509             ;  Update bulk delay buffer {6 cycles}
510   P:0111 568F00        move            x:tmp_tx,a
511   P:0112 5E5600        move                    a,y:(r6)-
512   P:0113 666700        move            r6,x:(r7)
513   P:0114 0C01FD        jmp    <trans_det                   ;next frame
515             ;-------------------------------------------------------------------
516             ;                High-pass Filtering routine
517             ;                ---------------------------
518             ;Alter registers : r2, r4 -> reference buffer
519             ;                  r6 -> bulk delay buffer
520             ;                  r7 -> bulk delay buffer pointer
521             ;Time taken : 27 cycles
522             ;Program words : 36 words
523             ;-------------------------------------------------------------------
524             hpf                                         ;entry point for C/S -> O/S
525             ;{7 cycles}
526             ;  Updating reference buffer  {7 cycles}
527   P:0115 66DF00        move            x:(r7)+,r6          ;retrieve bulk delay pointer
528   P:0116 77F400        move            #>-1,n7             ;offset -1
```

```
                FFFFFF
529     P:0118 64E700          move          x:(r7),r4          ;retrieve reference buffer pointer
530     P:0119 5EE600          move                  y:(r6),a   ;y(n) = y(n-96)
531     P:011A 5E5400          move                  a,y:(r4)-
532     P:011B 568F00          move          x:tmp_tx,a         ;load most recent samp. into bulk buf.
533                ;{1 cycle}
534                ;  Updating bulk delay buffer  {1 cycle}
535     P:011C 5E5600          move                  a,y:(r6)-
536                ;{3 cycles}
537                ;  Save r4, r6 for next frame  {2 cycles}
538     P:011D 646700          move          r4,x:(r7)
539     P:011E 666F00          move          r6,x:(r7+n7)
540                ;{6 cycles}
541                ;  Updating r4, r7  {6 cycles}
542     P:011F 229200          move          r4,r2
543     P:0120 205F00          move          (r7)+
544     P:0121 205C00          move          (r4)+
545     P:0122 568E00          move          x:tmp_rx,a         ;linear near-end sample sdc(n)
546     P:0123 0C012E          jmp    <hpf_com
547                ;{11 cycles}
548                ;  Computing high-pass filtering near-end sample
552                hpf_1                                     ;entry point for O/S -> C/S
553     P:0124 223013          clr    a      r1,r0           ;r0 -> O/S to C/S coefficients
554     P:0125 566700          move          a,x:(r7)         ;active direction = 0
555     P:0126 568F00          move          x:tmp_tx,a       ;linear near-end sample sdc(n)
556     P:0127 64EF00          move          x:(r7+n7),r4     ;retrieve r4
557     P:0128 578E00          move          x:tmp_rx,b       ;updating reference buffer
558     P:0129 5F5400          move                b,y:(r4)-  ;y(n) = most recent sample
559     P:012A 204F00          move          (r7)+n7
560     P:012B 229200          move          r4,r2
561     P:012C 645F00          move          r4,x:(r7)+       ;save r4
562     P:012D 205C00          move          (r4)+
563                hpf_com
564     P:012E 77F400          move          #>-1,n7          ;offset -1
                FFFFFF
565     P:0130 47DF00          move          x:(r7)+,y1       ;get sdc(n-1)
566     P:0131 000000          nop
567     P:0132 565F00          move          a,x:(r7)+        ;updating S0(n) = SDC(n)
568     P:0133 205A26          abs    a      (r2)+
569     P:0134 566700          move          a,x:(r7)         ;updating ABSS0(n)
```