THEORY, DESIGN AND APPLICATION OF GRADIENT ADAPTIVE LATTICE FILTERS

BY

M. J. RUTTER

B.Sc., M.K.T., M.I.E.E.E.

Thesis submitted for the degree
of Doctor of Philosophy to the
Faculty of Science,
University of Edinburgh.

1983

Declaration of Originality

This thesis, composed entirely by myself, reports work conducted in the Department of Electrical Engineering at the University of Edinburgh exclusively by myself, with the exception of the section mentioned below.

The work presented in section 6.2 was performed by Mr L R Rollo under the direction of myself and Dr P M Grant.

signed

M J Rutter
7th July 1983

ACKNOWLEDGEMENTS

Few PhD theses have been completed in isolation. Without compromising the Declaration of Originality, the author would like to express his thanks to Dr P M Grant, his supervisor, for much needed advice and encouragement based on a background of personal experience. Dr C F N Cowan, second supervisor, is warmly thanked for many heated but fruitful technical arguments in various places of refreshment throughout Edinburgh.

There is a debt of gratitude to Dr P F Adams and Mr F Westall of British Telecommunications Research Laboratories for their interest and for supplying the impulse responses of appendix D.

Thanks are also due to many friends and colleagues in the department, in particular Dr P B Denyer, Mr D Renshaw and Mr S G Smith. An erstwhile colleague, Dr A Morgül, now at Bogazici University, Istanbul proved both a friend and a source of inspiration.

Finally, thanks are due to my wife Margit, for her patience and constant support over the last three years.

CONTENTS

LIST OF ABBREVIATIONS

AC	Alternating Current
ADC	Analogue to Digital Converter
AGC	Automatic Gain Control
ALE	Adaptive Line Enhancer
AR	Autoregressive
ARMA	Autoregressive Moving Average
ASE	Adaptive Spectral Enhancer
BTRL	British Telecommunications Research Laboratories
CCD	Charge Coupled Device
CMOS	Complementary Metal Oxide Silicon
DAC	Digital to Analogue Converter
DC	Direct Current (zero frequency)
DFB	Data Directed Feedback
EPROM	Electrically Programmable Read only Memory
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FIRST	Fast Implementation of Real Time Signal Transforms
FM	Frequency Modulation
I	In phase
IC	Integrated Circuit
IIR	Infinite Impulse Response
ISI	Intersymbol Interface
LED	Light Emitting Diode
LHS	Left Hand Side
LHT	Left Hand Term
LMS	Least Mean Square
LPC	Linear Predictive Coding
LSB	Least Significant Bit
LSI	Large Scale Integration

MA	Moving Average
MDAC	Multiplying Digital to Analogue Converter
MSB	Most Significant Bit
MSE	Mean Square Error
MTI	Moving Target Indicator
NMOS	N-Channel Metal Oxide Silicon
PARCOR	PARtial CORrelation
PE	Prediction Error
PN	Pseudo Noise
PORL	Post Office Research Laboratories (now BTRL)
PRBS	Pseudo Random Bit Sequence
PSTN	Public Switched Telephone Network
Q	Quadrature
RADAR	Radio Detection and Ranging
RAM	Random Access Memory
RELP	Residual Excited Linear Prediction
RHS	Right Hand Side
rms	Root Mean Square
ROC	Rate of Convergence
SAW	Surface Acoustic Wave
SONAR	Sound Navigation and Ranging
TTL	Transistor Transistor Logic

LIST OF PRINCIPAL SYMBOLS

A(t)	Channel input signal (normally white)
C	$1/\mu$ (cf μ)
D(N)	K(N).Q(N) (cf section 2.3.3)
E<.>	Expectation operator
F(z)	A general polynomial in z
G(n,t),$\underline{G}$	nth tap of a combiner structure, value at time t
H(i)	ith element of a system impulse response
$\underline{I}$	Identity matrix
$K_f(n,t)$	Parcor coefficient. 'f' denotes forward prediction
M	The order of a filter
N	Ditto
P(i)	ith element of cross-covariance vector $\underline{P}$
Q(n)	Mean power output of nth order PE filter
R(i)	ith lag of autocovariance function
$\underline{R}$	Covariance matrix (square)
T	Used to denote matrix transposition
$\underline{V}$	Square matrix of eigenvectors
a(n,i)	ith element of impulse response $\underline{a}$ of nth order forward PE filter
b(n,t)	Backward PE signal value from nth order filter at time t
c(i)	Channel impulse response
d(t)	Desired, conditioning or training signal for adaptive filter
e(n,t)	error signal from n-stage filter at time t
f(n,t)	lattice forward PE signals
g(i)	Coefficients of AR filter
h(i)	ith term of impulse response $\underline{h}$ of an equaliser
i, j, k	Integer variables used in summation
n	Label for a tap or signal number
(opt)	Denotes the optimum value of a variable
r(i)	Backward PE coefficients.
s(n,t),$\underline{s}$	Signal input to a filtering structure
t	Time sample number
u(n,i)	The difference between two sets of parameters (cf section 2.3.7)

$w(i)$	Angular frequency of ith sinusoid
$y(t)$	Filter output
z	$\exp(sT)$, the symbol used in z-transform notation
α	Constant used in iterations
β	Ditto
∂	Partial differential
$\lambda\ (i)$	ith eigenvalue
$\underline{\Lambda}$	Diagonal matrix of eigenvalues
$\mu(n,t)$	Stepsize for iterative algorithms
σ^2	Variance
$\underline{0}$	Zero vector [0 0 0 ... 0]
$\underline{1}$	Unit vector [1 0 0 ... 0]
*	Hermitian transpose or complex conjugation

In the text, arguments will be added or dropped, when unambiguous to do so, to maximise clarity and readability. All vectors defined to be horizontal.

Chapter One

INTRODUCTION

1.1 Channel Distortion

When a signal passes through a communications channel, it normally suffers some degree of distortion [1]. Linear distortion occurs when the filtering effect of the channel delays and attenuates some frequencies more than others. The filtering effect of a twisted pair of wires, for example, results from their distributed parallel capacitance and series inductance.

An important degrading mechanism in data communication is the delay or phase distortion introduced in the channel. Data symbols are sent consecutively down the channel and received consecutively at the channel output. If part of a data symbol is delayed with respect to the rest of that symbol, it will arrive in the time slot reserved for another symbol and cause distortion of the data. The distortion caused by one symbol disturbing another is usually called intersymbol interference (ISI) [1].

Communications channels also add noise to the transmission, to varying extents [2]. This has the effect of altering the value of a received symbol in some unpredictable way. Although noise may be accurately described in terms of its power spectral density, it is not predictable and may not easily be removed from the signal.

Unless steps are taken to minimise the effect of noise and ISI, they can alter a signal to such an extent that the data is corrupted and incorrect decisions will be made in the detector.

1.2 Equalisation

Equalisation [1] is based on the deployment of an additional filter to reverse or compensate for the distortion introduced by the channel. If, for example, one filter (the channel) delays the high frequencies by a certain amount, then another filter (the equaliser)

may be placed in series which delays low frequencies by a similar amount. All frequencies will then take equal lengths of time to pass through the overall system.

The above, highly simplified account explains the process of equalisation. Equalisers are used to minimise the effect of channel distortion or the ISI of the signal. Figure 1.1 shows the operation of such an equaliser in the power spectral domain. The input to the channel is assumed to be spectrally white. The channel then shapes the spectrum, causing distortion and ISI. The frequency response of the equaliser is the inverse of that of the channel, making the signal white once more. Equalisation is a special case of whitening, where both phase and spectral density are corrected.

Figure 1.2 shows the same process in the time domain. A perfect channel would have a unit impulse response, which would produce no distortion when convolved with the input signal. For a distorted channel, a unit impulse applied to the input is convolved (stretched and distorted) with the channel's own impulse response. The equaliser's impulse response convolves with that of the channel output signal to produce an almost undistorted system output, one that matches as closely as possible the single unit impulse which was originally input to the system. The word 'almost' is important here, because equalisation is usually an approximate process whose accuracy depends on the system specification.

1.3 Adaptive Filters

In many cases the characteristics of the channel are known in advance. Examples include magnetic tape playback systems, record reproduction systems and fibre optic detectors [[27] appendix F]. In these cases the equalising filter may be designed in advance and installed using fixed components. Often, however, this is not the case.

There are applications where channel characteristics are not known in advance and may even vary unpredictably during use. Examples include radio and telephone channels [37]. Even when using

the public switched telephone network between two known points, a different line can be allocated each time the connection is made. Once the connection has been made, the line changes its characteristics very slowly indeed. Telephone data modems thus need an equaliser which can adjust its coefficients according to channel type and, ideally, track slow changes in channel characteristics. One solution is the adaptive filter.

In HF radio channels, multipath propagation introduces fades in the received signal [45]. This is a more demanding application for equalisation, as the fades vary the channel characteristics at a fast rate, demanding faster equaliser tracking than in data modems.

Adaptive filters often use two input signals [3]. The s(t) or signal input is normally the distorted signal. The desired or training signal, d(t), is either supplied separately or derived from the filter output. The filter is arranged to adjust its coefficients to force the output to approximate the training signal as closely as possible. The training sequence may be stored in a memory and used to train the modem before data transmission. Alternatively the receiver's detector output may be assumed to be correct and used as a training signal for the equaliser. This only works for cases where there is a modest amount of distortion and the output is valid or correct for most of the decision instants, as incorrect decisions will not drive the filter towards convergence. The technique is often referred to as data-directed feedback (DFB) [6].

Adaptive filters may be of open-loop or closed-loop type. In the open-loop type (figure 1.3) the processor collects data on the statistics of the s(t) and d(t) signals and their cross-covariance. This information is then used to calculate and install the equaliser coefficients. In the closed-loop type (figure 1.4) the output of the filter is subtracted from the training signal to form an error signal. An iterative or learning algorithm then operates to minimise the power in the error signal by optimising the filter coefficients. Closed-loop filters normally have simpler algorithms than open-loop types, although they normally take longer to calculate the filter coefficients to a given precision.

The operation of a filter which learns is a topic very close to Artificial Intelligence. In 1942 Norbert Wiener [4] derived a formula for calculating the optimum coefficient values of the filter. In 1960 Papert [71] proposed an iterative algorithm for a machine to calculate the values itself. In 1961 Gabor [72] constructed such a machine capable of adjusting its own coefficients using a similar algorithm. The machine's use of a magnetic tape memory and servo-controlled parameter adjustments precluded its use for real-time modem equalisation, however. In 1967 Widrow [5] developed the least mean square (LMS) recursive algorithm suitable for use in closed-loop adaptive filters. Lucky [6] of IBM was one of the first to use an algorithm successfully in modem equalisation. Lucky's algorithm was the zero-forcing algorithm, a simplification of the LMS.

1.4 Adaptive Lattice Equalisers

Most adaptive filters to date have used a finite impulse response (FIR) transversal filter as the programmable filter [3] (figure 1.5). This design approach ensures a stable solution, but it suffers from an uncertain rate of convergence (ROC). This uncertainty was found to stem from its dependence on input signal power and on interaction between converging tapweight values. The tapweight interaction is due to correlations between the successive samples of s(t) which are weighted in the filter. This is the very effect that the equaliser is attempting to minimise. This deficiency was highlighted in 1972 by Ungerboeck [7], who related input power and channel characteristics to the rate of convergence of the equaliser and accuracy once converged.

Robert Chang [8] proposed the use of an orthogonalising network to minimise tapweight interaction, producing the general equaliser structure of figure 1.6. He was certainly not the first to mention this technique, however. In 1960 Goodall [9] suggested what he called an 'encoding network' as part of a model for animal intelligence. In 1961, however, Lubbock [10] proposed the use of prediction-error (PE) filtering to orthogonalise the signals for an adaptive filter. This is the first record of lattice mathematics

being used in adaptive filtering. Ironically, in the discussion following the paper, Gabor, the inventor of the first working adaptive filter, dismissed Lubbock's work as unnecessarily complicated.

When an idea originates, it often does so as a result of many seemingly unrelated contributions. Levinson [[4] appendix A] could arguably be regarded as the father of the lattice, both for his work in making Wiener's conclusions accessible to non-mathematicians and for his own work on prediction-error filtering. The modelling of a spectrum using PE filtering was attractive to those working on speech bandwidth compression and this gave rise to the topic of linear predictive encoding (LPC)[11]. When Itakura [12] invented the lattice, then, it was for linear predictive work and not equalisation. Makhoul [11] and Mead [13] then independently proposed similar gradient algorithms, based on the LMS [5], to make the lattice iteratively adaptive.

Once the convergence limitations of the LMS algorithm had been analysed, and Chang had proposed the use of an orthogonalising network, it became clear that the properties of the lattice had applications in equalisation. Griffiths then solved the two problems necessary to create a lattice adaptive equaliser (figure 1.7). In 1977 he reported the stepsize recursion needed to make the structure track and converge reliably [14]. The following year he reported the addition of sidetaps to turn his orthogonalising structure into an adaptive equaliser with reliable convergence properties [15].

The term 'lattice filter' is confusing, since both orthogonalising PE structure and equaliser with sidetaps are adaptive filters. For this reason the ambiguous term will be avoided. The adaptive prediction-error filter, often used in LPC, will be referred to as a 'lattice structure'. The addition of sidetaps to form a more conventional adaptive filter model will be referred to as a 'lattice equaliser'. The term 'lattice equaliser' will thus be used even in conjunction with applications which do not involve equalisation.

The development of algorithms has always been in advance of the

technology needed to economically employ them. An example is the exact least squares lattice, reported by Morf [16]. Using a technique originally proposed by Makhoul [11], the algorithm uses on open-loop algorithm to converge extremely rapidly. A problem inherent in the exact lattice is its requirement for a precise division to calculate the filter coefficients. The approximate divisions necessary for gradient stepsize recursions are certainly possible in digital hardware, but precise divisions are currently very awkward. For this reason, the exact algorithms have merely been explained. When algorithms have been discussed in depth, they have been of gradient, closed-loop type. In chapter three an even faster lattice technique is studied and a computer simulation is presented. However, this too requires exact divisions and is mentioned only for completeness.

When the wide selection of adaptive algorithms is viewed, it is most useful to view them as graded in order of complexity. At one extreme is the simple, cheap but inefficient zero-forcing algorithm [6]. At the other are the rapid matrix inversion techniques, which are extremely difficult to implement in dedicated hardware [17]. The most popular compromise to date has been the LMS transversal algorithm [3]. In view of the simplicity, popularity and low cost of the latter, any competitor must be studied in comparison with it. This has indeed been done in this thesis, and the gradient lattice has been shown to provide a more reliable rate of convergence at the expense of a more complicated hardware construction.

1.5 Thesis Layout

This thesis reports the application of the gradient lattice equaliser in place of the transversal filter in conventional gradient search adaptive filter designs. This chapter was written as an introduction to adaptive filtering in the context of adaptive equalisation. Chapter two gives a more mathematical description of channel equalisation. It tackles the problem of how to optimally minimise the distorting effect of a channel, and lays a foundation of non-adaptive techniques to which later chapters refer.

Chapter three extends the discussion to adaptive algorithms.

While all types are discussed, the emphasis is placed on gradient algorithms, of which the adaptive linear combiner algorithms are the most general form. It ends with a discussion of adaptive lattice structures and equalisers. Chapter four examines the modem channels likely to be encountered in practice in this country. In doing so it quantifies the channel parameters discussed in the previous and following chapters, and offers a guide to future equaliser specifications.

Chapter five presents the results of computer simulation experiments on gradient lattice and transversal equalisers. These confirm the theoretical predictions of chapter three and explore new solutions to the problems uncovered.

The sixth and seventh chapters both discuss hardware realisations of the lattice. The former describes the use of discrete components in an analogue lattice structure and a digital lattice equaliser. The latter reports a custom IC design for a lattice structure.

Chapter eight reports other applications of lattice structures and equalisers in signal processing. This permits a better perspective on the usefulness of the algorithm. The ninth chapter presents the summary and conclusions of the thesis.

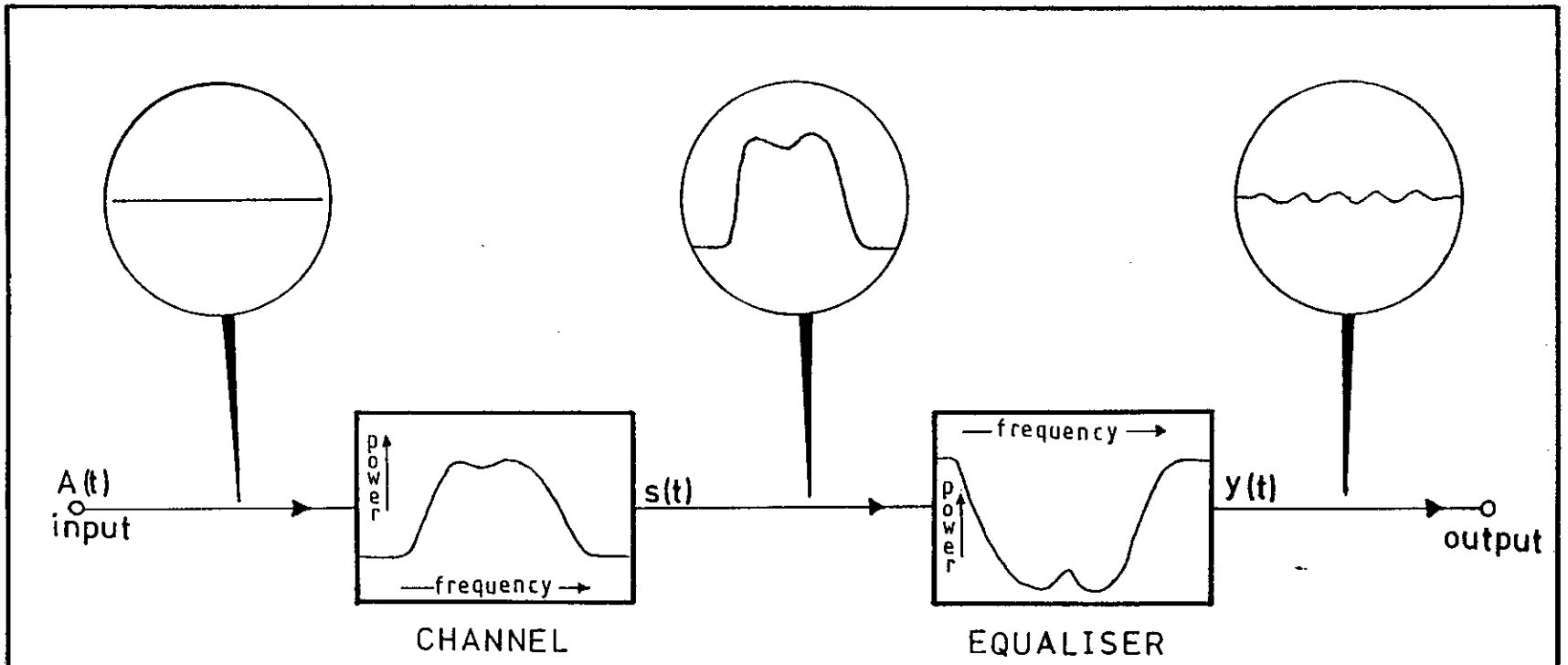


Fig 1.1 Channel distortion and equalisation in the power spectral domain

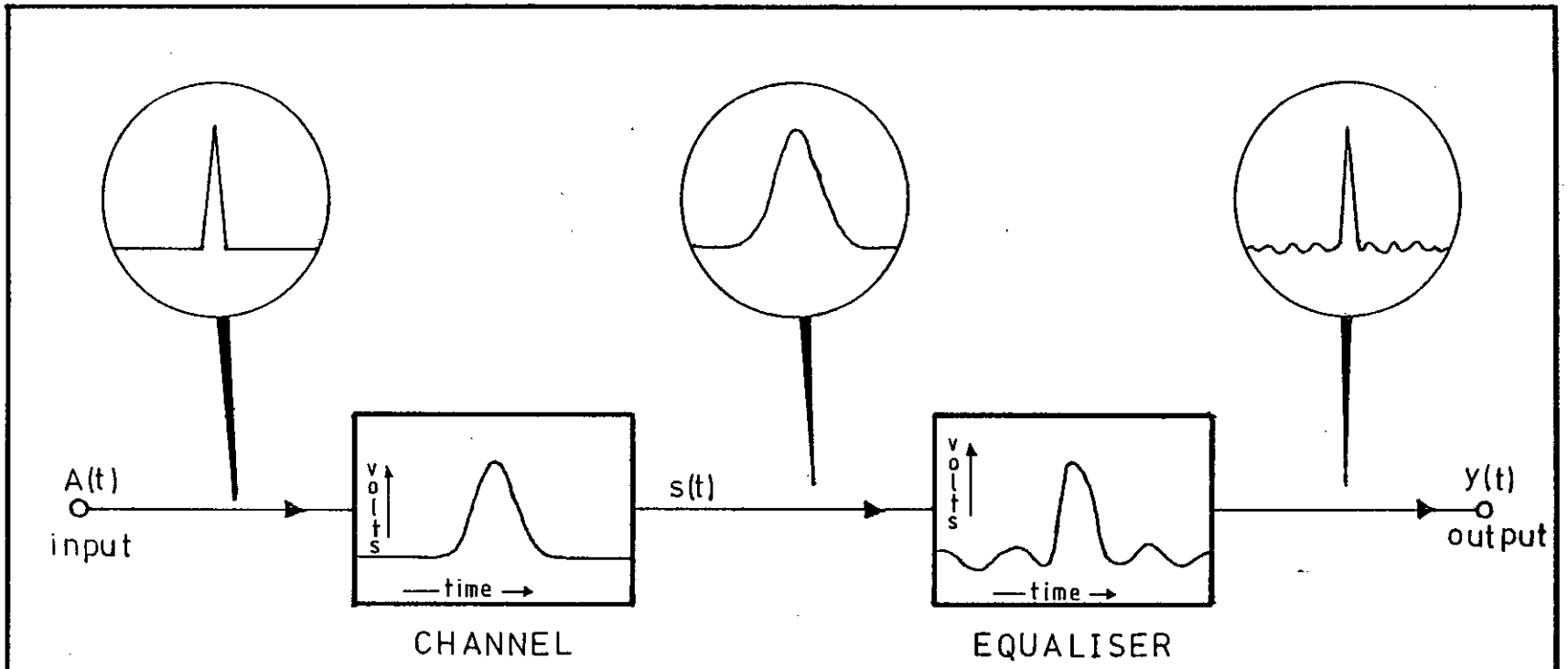


Fig 1.2 Channel distortion and equalisation in the time domain

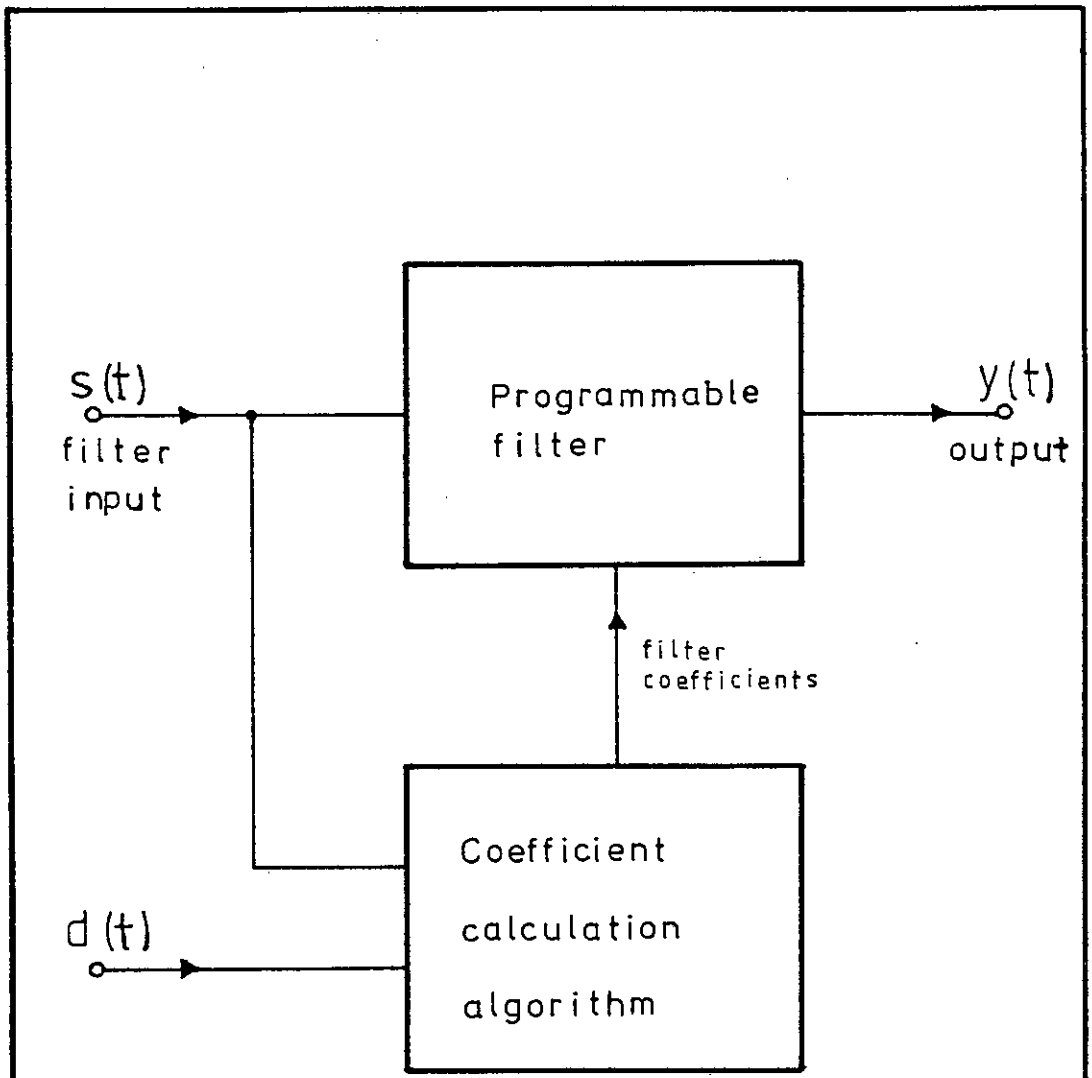


Fig 1.3 An open loop adaptive filter

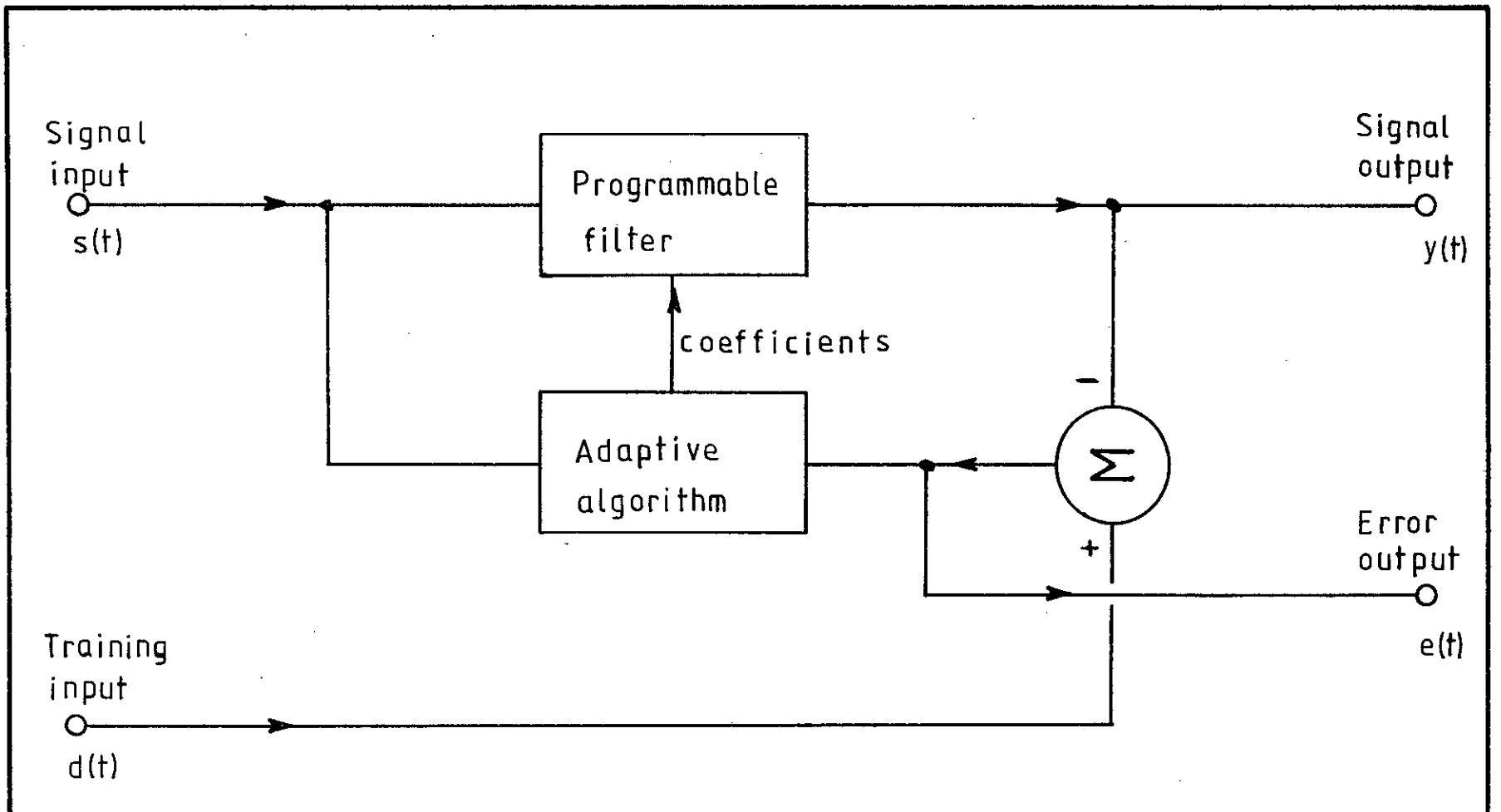


Fig 1.4 A closed loop adaptive filter

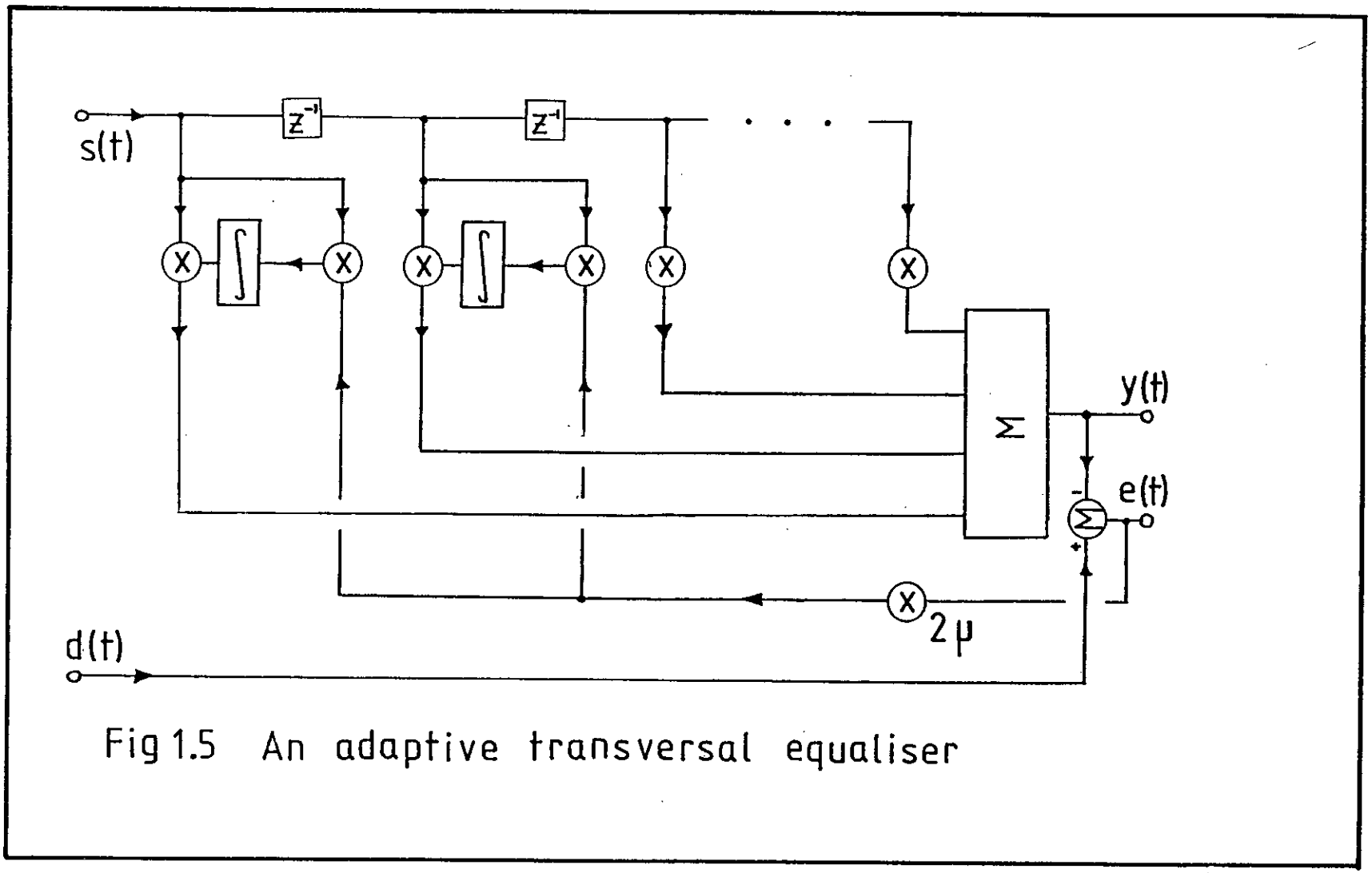


Fig 1.5 An adaptive transversal equaliser

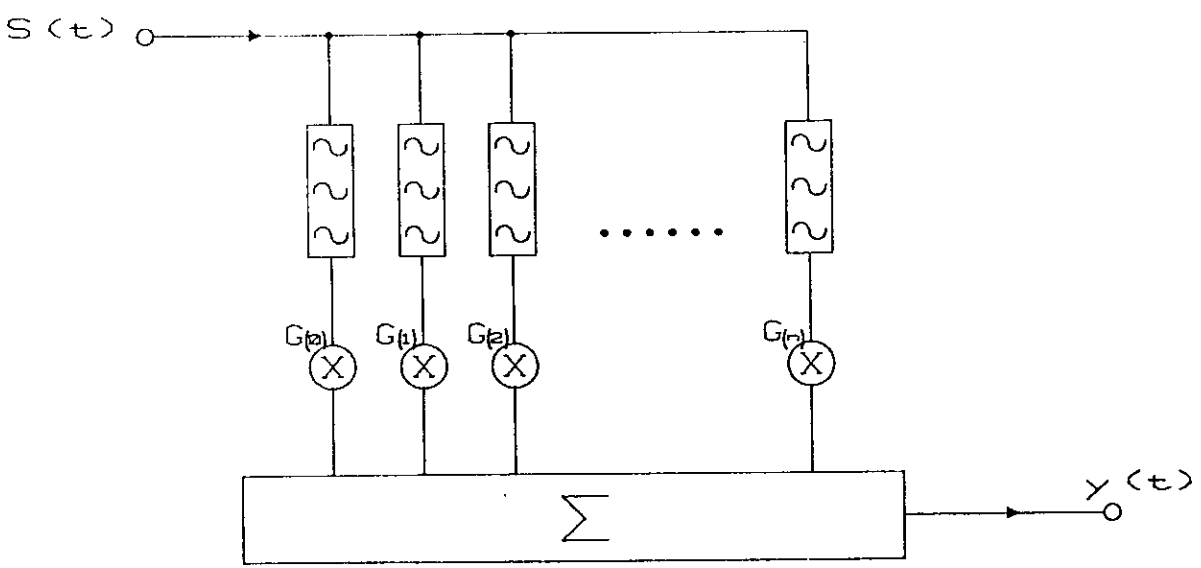


Fig 1.6 The Chang general equaliser

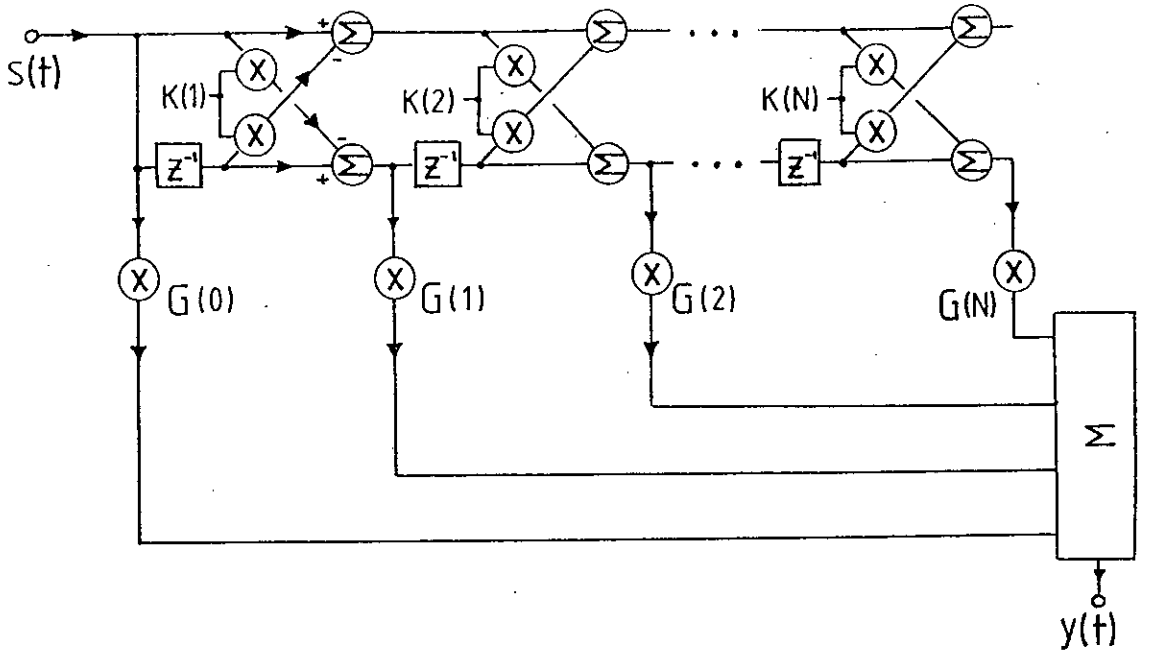


Fig 1.7 The lattice equaliser

Chapter Two

OVERVIEW OF ESTIMATION THEORY

This chapter is an introduction to the mathematics of equalisers. Before continuing in later chapters to discuss filters which optimally vary their coefficients, it is worthwile to examine the solutions towards which these filters should ideally converge.

Section one introduces moving average (MA), autoregressive (AR) and autoregressive moving average (ARMA) process models. It discusses the errors and limitations involved in using an MA function to approximate an ARMA type. Section two discusses the coefficients of the equaliser, of which the Wiener solution is the most general and the most important. The matched and inverse filter solutions are described, and it is pointed out that these too may be precise Wiener solutions under certain circumstances.

Section three uses the mathematics of section two to introduce the concepts of linear prediction. Prediction-error (PE) filtering is for some a difficult concept bacause there is no obvious direct use for the output. Nevertheless, the mathematics involved in the calculation of the coefficients lead into the FIR lattice structure and AR methods of spectral estimation. The chapter summary is section four.

This chapter will provide a mathematical grounding on non-adaptive signal processing which will subsequently be referred to throughout the thesis.

2.1 The Channel

A perfect non-distorting channel may delay all parts of a signal by a fixed amount. It may not, however, attenuate. Thus if a pattern of binary ones and zeroes is input to a non-distorting channel as a pattern of ones and minus ones, the polarity and amplitude of the pattern will be unaffected at the output.

A channel may be multi-dimensional, i.e. it may be able to carry more than one independent signal simultaneously. In this case an additional requirement is that the signals on the output are equally independent. Two-dimensional channels are common in data communications and are discussed in detail throughout this thesis. The discussion in this chapter will however be restricted to one-dimensional models for simplicity's sake. Similarly, although multi-level coding is in common use, this description will be limited to two-level coding, denoting one bit of information per unit time. The descriptions will thus be rendered more understandable, with little loss of generality.

Since digital communications usually take place using clocked instants of discrete time, the impulse response of the channel may be given generally as:

$$F_c(z) = \sum_{i=0}^{N} c(i).z^{-i} \qquad (2.1)$$

It is true that many filtering structures have impulse responses of infinite length, but one may pragmatically assume that only N samples are significant. One may then assume the rest to be zero with minimal loss of precision. Should a signal A(t) then be applied to the channel, it will be dispersed over N different time slots. Conversely, one time slot of the channel s(t) will comprise N consecutive input samples in a weighted average.

$$s(t) = \sum_{i=0}^{N} A(t-i).c(i) \qquad (2.2)$$

This causes undertainty as to which output sample s(t) corresponds to which input sample A(t-i). If no equalisation is to be used, the sample corresponding to the maximum of c(i) is usually taken as that corresponding to the channel delay, linking s(t) and A(t-i). Nevertheless, the output sample will be contaminated with other input samples. The contamination is the intersymbol interference (ISI) referred to in chapter one. In some cases the level of contamination may lead to incorrect interpretation, or detection, of the channel output. The introduction of such detection errors is referred to as data corruption.

There is a strong parallel between phased array reception [5] and data communications. Whereas in data communication the unwanted signals are from different time slots, those in phased antennas are at different angles to that of the wanted transmission, which is referred to as the main lobe. Sample contamination by signals from different time slots will be referred to as 'off-lobe' data in accordance with this analogy.

Most physical channels also contribute some level of noise to the signal. In most mathematical treatments this contribution to s(t) is assumed to be white and Gaussian. Being Gaussian it may assume any value in the range $\pm \infty$.

Thus, as we assume this model for the work in the thesis, the value of s(t) is no longer certain. Although a Gaussian variable may assume any value, its value is most probably close to its mean. In data communications, channels are designed so as to have a low noise level, corrupting a minimum number of data elements.

The channel distorting mechanism may be modelled using moving average processes, autoregressive types, or a combination of the two. The MA transfer function was given in equations 2.1 and 2.2.

The autoregressive model involves forming the output from the current input and previous output samples (fig 2.1). Thus:

$$s(t) = A(t) - \sum_{i=1}^{N} s(t-i)g(i) \qquad (2.3)$$

or

$$A(t) = \sum_{i=0}^{N} s(t-i)g(i) \qquad (2.4)$$

Where g(0 ... N) are the filter coefficients, and g(0) is defined as unity. Comparing equations, 2.2 and 2.4 shows that the AR function is the inverse of the MA. Indeed, were the coefficients to be equal, an AR filter in series with an MA type would precisely cancel one anothers' effects (cf section 2.3.7).

ARMA filters use a combination of two transfer functions:

$$s(t) = \sum_{i=0}^{N} A(t-i).c(N-i) - \sum_{i=1}^{M} s(t-i).g(M-i) \qquad (2.5)$$

The characteristics of the AR and MA components are best shown using the z-transform:

$$s(z) = A(z) \left[\sum_{i=0}^{N} c(i)\, z^{-i} \right] / \left[\sum_{i=0}^{M} g(i).z^{-i} \right] \qquad (2.6)$$

Complex values of z may be found which cause the AR or MA parts of the function to become zero. Should the AR part become zero, the ARMA function will become infinite. Should the MA part become zero, then so will the ARMA function. This leads to alternative names of all-pole denoting an AR function and all-zero denoting an MA type.

According to the Wold decomposition theorem [18] any transfer function, AR, MA or ARMA may be represented by an AR function of infinite order. By symmetry, an MA transfer function must also be able to represent any transfer function, since it is the inverse of an AR type and has one more degree of freedom. Letting a function be so represented:

$$s(t) = \sum_{i=0}^{N.} A(t-i)c(i) + \sum_{i=N+1}^{\infty} A(t-i)c(i) \qquad (2.7)$$

Defining A(t) to be white and of unit power, and taking expectation of power:

$$E < s^2(t) > = E < \sum_{i=0}^{N} c^2(i) > + E < \sum_{i=N+1}^{\infty} c^2(i) > \qquad (2.8)$$

Where E<.> is the expectation operator. All expectation terms in equation 2.8 are finite and positive. If the first term on the RHS is taken as an estimate for the LHS, the error would be equal to the second term on the RHS. Increasing the value of N, the order of the transfer function, will reduce the error. Thus for a particular level of maximum tolerated error, a particular value of N may be found which just satisfies it. I.e. any function may be approximated to any arbitrary mean-square error specification by an MA process of the appropriate finite order.

The MA filter function is often called the finite impulse response (FIR) filter, because its impulse response is as long as its order. Apart from the approximation error already discussed, it has limitations stemming from its mathematical model: The transfer function will always contain as many zeroes as the order of the filter. The function has no poles, but poles may be approximated by suitably arranging the zeroes. The fit to any required transfer function or frequency response will furthermore be limited by the number of degrees of freedom of the MA transfer function, i.e. its order.

2.2 Equalisation

Equalisation may be defined as the reversal of or compensation for the dispersive effect of a channel, by using a second filter between channel and detector. The object of the equaliser is to adopt a frequency response which is the opposite of that of the channel and whiten the signal once more. The equaliser must also compensate for the phase distortion introduced by the channel. The result, shown at the end of the signal path in figure 1.1, is a signal which is approximately white and ready for detection.

Conversely, equalisation may be examined in the time domain. The impulse response of the equaliser is taken as the moving average (MA) function:

$$F_h(z) = \sum_{i=0}^{M} h(i)\, z^{-i} \qquad (2.10)$$

As with the channel, the discrete time nature of the impulse response makes no assumptions about the type of filter in use, and one may be just as pragmatic in selecting the order M.

The channel and equalising filter impulse responses convolve

together to give the system impulse response. Taking the system output as y(t):

$$y(t) = \sum_{i=0}^{N} \sum_{j=0}^{M} A(t-i-j).c(i).h(j) \qquad (2.11)$$

or

$$y(t) = \sum_{k=0}^{\infty} A(t-k). \sum_{i=0}^{\infty} c(i)h(k-i) \qquad (2.12)$$

Where hitherto undefined (out-of-range) channel and equaliser impulse response terms are defined as zero. The next step is to define a value of k, the system delay, which relates the nominal input of the channel to the nominal output. Equalisation is usually an approximate process and the closeness of the approximation varies according to the value of the system delay selected [1].

Having selected a value of k, the impulse response of the equaliser is optimised so as to maximise the system impulse response term corresponding to a delay of k, the main lobe, and also to minimise the other off-lobe terms. Chapter three will discuss the criteria by which the maximisation and minimisation may be done.

Figure 1.2 shows the process of equalisation in the time domain. The impulse at the system input may be interpreted as a unit impulse applied to the system, or one bit of information. The impulse response of the channel is convolved with this to give a signal corresponding to the channel impulse response. This is convolved with the impulse response of the equaliser to give the system impulse response. This is shown to be a close approximation to a unit impulse.

The impulse response of the equaliser depends on the impulse response of the channel to be equalised. There are a number of ways of specifying equaliser performance. The three discussed next are the Wiener filter, the inverse filter and the matched filter.

2.2.1 The Wiener Filter

The Wiener filter [4] is the optimum form of FIR equaliser for minimum error power. Since much of the mathematics in this thesis is

based on the Wiener equation, a general approach will be taken.

For any filter one may define an ideal or desired signal d(t) which should be output when the filter is fed input signal s(t). This ideal is seldom realisable in practice. Let the actual output be y(t) and the actual amount by which these differ be e(t). Thus:

$$e(t) = d(t) - y(t) \tag{2.14}$$

The LMS criterion for designing such filters is that the mean of the square of the error, i.e. its mean power, is at a minimum. In the general case the output of the filter comprises a weighted combination of input signals. Thus for a FIR filter:

$$y(t) = \sum_{i=1}^{N} G(i).s(i,t) \tag{2.15}$$

Where G(i) is the weighting and 1 ... N refers to the input signals. The Wiener solution depends on two matrices, $\underline{R}$ and $\underline{P}$. $\underline{R}$ is defined as:

$$\underline{R} = E < \underline{s}^T \underline{s} > \tag{2.16}$$

Where $\underline{s}$ is the vector [s(1,t), s(2,t) ..... s(N,t)]. $\underline{P}$ is defined as:

$$\underline{P} = E < \underline{s} .d(t) > \tag{2.17}$$

I.e. $\underline{P}$ is the cross-covariance between input and desired signal. The Wiener solution is:

$$\underline{G} = \underline{P}\underline{R}^{-1} \tag{2.18}$$

This is often referred to as the Wiener-Hopf equation [5].

For a one-dimensional equaliser, the input signals may be regarded as delayed versions of one another, i.e.:

$$s(i,t) = s(1, i-t) \tag{2.19}$$

This will make $\underline{R}$ the autocovariance matrix. Where R(i) is the ith lag of the input signal autocovariance function:

$$\underline{R} \equiv \begin{vmatrix} R(0) & R(1) & \dots & R(N-1) \\ R(-1) & R(0) & \dots & R(N-2) \\ \dots & \dots & \dots & \dots \\ R(1-N) & R(2-N) & \dots & R(0) \end{vmatrix} \qquad (2.20)$$

$\underline{P}$ will be the cross-covariance function between filter input and desired output:

$$P(i) = E < d(t).s(t-i) > \qquad (2.21)$$

$$\underline{P} = [\, P(k)\ P(k+1)\ \dots\dots\ P(k+N-1)\,] \qquad (2.22)$$

Note that $\underline{P}$ is defined as a truncation of the cross-covariance function, a function infinite length. The position of the truncation, i.e. the value of k, is left to the designer to specify. Thus the Wiener-Hopf equation may be re-expressed in summation notation as:

$$P(j) = \sum_{i=0}^{N-1} h(i).R(i-j) \qquad (2.23)$$

Where h(1 ... N) are the coefficients of the MA equaliser and P(j) is a member of $\underline{P}$. As opposed to the inverse and matched solutions, the Wiener coefficfient values are affected by channel noise. The addition of white noise will alter the value of R(0) and thus the values of h(i). Under low-noise conditions the Wiener solution tends towards the inverse filter, whereas under high-noise conditions the matched solution is approached [19].

2.2.2 The Inverse Filter

The inverse filter makes use of the inverse relationship between AR and MA functions. If the channel may be shown to have an AR transfer function, or one which may be modelled by an AR function, then the optimum equaliser is an MA filter with identical

coefficients. Then, as shown in the description of section 2.1, the channel will be exactly equalised, with no error whatsoever.

Conversely, if the channel can be modelled as an MA function, it will be exactly equalised by an equaliser which can be modelled as an AR function with the same coefficients. It is not, however, always possible to construct stable AR filters.

The AR function has an infinite impulse response (IIR). Thus it may only approximately be modelled by an MA process, which has a finite impulse response. One simple way of calculating the equaliser coefficient is to calculate the first N terms of the impulse response of the IIR filter. The MA FIR filter is then given an impulse response which is a truncated version of the IIR. This will not, however produce the optimum coefficients for the FIR filter, although the result will usually be quite close.

2.2.3 The Matched Filter

When white noise is added to a channel, the only term of the autocovariance function affected is R(0), the power estimate. None of the terms of the cross-covariance function of the Wiener equation are affected. Thus, as the variance, σ of the noise is increased, the terms on the major diagonal of the autocovariance matrix, R, are increased by σ. With high noise levels the effect of off-diagonal terms becomes insignificant. Thus, where $\underline{I}$ is the unit matrix:

$$\text{as } \sigma \longrightarrow \infty, \tag{2.25}$$

$$\underline{R} \longrightarrow \sigma \underline{I}, \tag{2.26}$$

$$\text{so } \underline{h} = \underline{P}\,\underline{R}^{-1} \longrightarrow \underline{I}\underline{P}/\sigma \tag{2.27}$$

$$\text{i.e. } \underline{h} \longrightarrow \underline{P}/\sigma \tag{2.28}$$

Thus the terms of the Wiener equaliser tend towards a value proportional to the terms of the impulse response of the channel

[20]. Indeed, the terms of the channel impulse response become a close approximation to those of the Wiener filter.

The operation of the filter may most readily be understood by noting that the system impulse response given by equation 2.12 becomes proportional to the autocovariance function of the channel's transfer function. No lag of the autocovariance function may be greater in magnitude than the zeroeth lag. This ensures that the system input signal has a maximum contribution to the output after a delay equal to the length of the equaliser.

2.3 Linear Prediction

In linear prediction, the principles of estimation theory are applied to extrapolation. Given the values of the last N signal elements s(t-1) ... s(t-N), one must optimally predict the signal value over the next few samples, s(t) ... s(t + n). Normally, and in this thesis, the prediction is confined to the next sample only, s(t). This is calculated as accurately as possible using a weighted combination of the previous N samples:

$$y(t) = - \sum_{i=1}^{N} a(i).s(t-i) \qquad (2.29)$$

Where a(1 ... N) are the predictor weights and y(t) is the predicted value. This is referred to as forward prediction.

History tells us that the less recent an incident is, the less likely it is that knowledge of it is available. This introduces the concept of the backward prediction filter, which calculates the probable value of a single element based on a knowledge of those following it; using weights r(1 ... N):

$$y(t) = - \sum_{i=1}^{N} r(i).s(t+i) \qquad (2.30)$$

Clearly, in order to optimally predict the element, the weights a(i) and r(i) must be optimised according to the stationary or quasi-stationary statistics of the input signal. Indeed, the importance of the field of linear prediction to the field of signal processing lies

in the relationship of the predictor weights to the aperiodic autocovariance function of the signal.

2.3.1 The Prediction-Error Filter

In the prediction-error (PE) filter, the signal estimates obtained from equations 2.29 and 2.30 are subtracted from s(t), the signal sample itself. This gives an output of the prediction error, the error incurred in making the estimate. Figures 2.2 and 2.3 show the signal flow path for the forward and backward PE filters respectively.

The transfer function of the filter is of direct interest. For the forward PE filter it is:

$$e(t) = s(t) + \sum_{i=1}^{N} a(i)\, s(t-i) \qquad (2.31)$$

$$= \sum_{i=0}^{N} a(i)\, s(t-i) \qquad (2.32)$$

Defining a(0) as unity. Similarly for the backward predictor, with r(0) defined as unity:

$$e(t) = \sum_{i=0}^{N} r(i).s(i+t) \qquad (2.33)$$

It may seem unusual to deliberately create a signal composed entirely of errors. In fact it is this error signal which is used in the transformations of the lattice equaliser. The filter weights are adjusted to minimise this error signal, e(t), according to some criterion. The least mean square (LMS) criterion is commonly used, in which the parameter to be minimised is the mean prediction-error power, Q.

2.3.2 The Normal Equations

The important characteristics of the prediction-error filter stem from the orthogonality properties inherent in them. They are expressed in a set of equations known as the prediction-error equations, normal equations or Yule-Walker equations [21], which will

now be derived. From equation 2.14:

$$e(t) = d(t) - \underline{sh}^T \tag{2.35}$$

$$\text{so} \quad |e(t)|^2 = d^2(t) - d(t)(\underline{hs}^T - \underline{sh}^T) - \underline{hs}^T\underline{sh}^T \tag{2.36}$$

Using the Wiener-Hopf equation (2.18), one may derive the minimum error power, Q. From 2.27:

$$Q = E < d^2(t) - d(t)(\underline{hs}^T - \underline{sh}^T) - \underline{hs}^T d(t) > \tag{2.37}$$

$$= E < d^2(t) > \underline{Ph}^T \tag{2.38}$$

The above equation gives the minimum error of any Wiener filter. To obtain the power out of a PE filter, one substitutes the fact that d(t) = s(t), making $\underline{P}$ the autocovariance function:

$$Q = R(0) - \sum_{i=1}^{N} h(i)\, R(i) \tag{2.39}$$

and substituting -a(i) for h(i), and recalling that a(0) is equal to unity:

$$Q = \sum_{i=0}^{N} a(i)\, R(i) \tag{2.40}$$

Meanwhile the Wiener-Hopf equation is interpreted in a further way:

$$E < \underline{s}^T\underline{sh}^T > = E < \underline{s}^T.d(t) > \tag{2.41}$$

and for a PE filter, for j = 1 to N:

$$E < \sum_{i=1}^{N} s(t+j).s(t-i).h(i) > = E < s(t).s(t+j) > \tag{2.42}$$

So substituting a for h as the PE filter coefficients, and using the fact that a(0) = 1:

$$0 = \sum_{i=0}^{N} a(i).R(j-i) \tag{2.43}$$

Combining equations 2.40 and 2.43:

$$\sum_{i=0}^{N} a(i)R(j-i) = \begin{cases} Q & | \; j = 0 \\ 0 & | \; j = 1 \ldots N \end{cases} \qquad (2.44)$$

These, then are the normal equations which may be solved in order to produce values for $\underline{a}$, [a(0) ... a(n)] and Q. Defining $\underline{1}$ as the vector [1 0 0 ... 0], equation 2.44 may be expressed in matrix form as:

$$\underline{Ra}^T = Q.\underline{1}^T \qquad (2.45)$$

2.3.3 The Levinson-Durbin Recursion

In the last section a link was established between the terms of the aperiodic autocorrelation function, the coefficients of the prediction-error filter and the filter output power. In this section the PARCOR coefficient is introduced and a computationally efficient way is derived for calculating one set of values given one of the other sets. From equation 2.44, defining a(i) = 0 outside the range $0 \leq i \leq N$:

$$\sum_{i=0}^{N+1} a(N,i).R(j-i) = \begin{cases} Q(N) & | \; j = 0 \\ 0 & | \; j = 1 \ldots N \\ D(N) & | \; j = N+1 \end{cases} \qquad (2.47)$$

Note that the argument N has been added to denote the order of the filter. D(N) refers to the extra result obtained when j = N + 1. Owing to the Hermitian nature of the autocovariance function, one may derive a similar expression for the backward PE coefficients:

$$\sum_{i=0}^{N+1} r(N,N+1-i)R(j-i) = \begin{cases} D^*(N) & | \; j = 0 \\ 0 & | \; j = 1 \ldots N \\ Q(N) & | \; j = N+1 \end{cases} \qquad (2.48)$$

From this it becomes evident that for stationary signals, the terms of r are the complex conjugates of a, arranged in reverse order. Let equation 2.48 be multiplied by a term K(n + 1) and subtracted from equation 2.47:

$$\sum_{i=0}^{N+1} R(j-i).(a(N,i) - K(N+1).r(N,i)) \begin{cases} = Q(N) + K(N+1).D^*(N) & \mid j = 0 \\ = 0 & \mid j = 1 \ldots N \\ = D(N) + K(N+1).Q(N) & \mid j = N \end{cases} \quad (2.49)$$

However, there is another set of normal equations governing an autocovariance matrix of this order. This is the set of equations for an order one greater:

$$\sum_{i=0}^{N+1} R(j-i).a(N+1,i) \begin{cases} = Q(N+1) & \mid j = 0 \\ = 0 & \mid j = 1 \ldots N+1 \end{cases} \quad (2.50)$$

Combining equation sets 2.49 and 2.50 to eliminate autocovariance terms gives a number of important results for different values of j:

$$D(N) = K(N+1).Q(N) \quad \mid j = 0 \quad (2.51)$$

$$Q(N+1) = Q(N) - K(N+1).D^*(N) \quad \mid j = N + 1 \quad (2.52)$$

$$a(N+1,i) = a(N,i) - K(N+1).r(N,N+1-i) \mid 0 < j < N+1 \quad (2.53)$$

$$\text{ie, } a(N+1,i) = a(N,i) - K(N+1).a^*(N,N+1-i) \quad (2.54)$$

It is stressed that the conditions j = 0 ... N + 1 all exist simultaneously and thus the above equations are mutually consistent. Eliminating D(N) from equations 2.51 and 2.52:

$$Q(N+1) = Q(N).(1-|K(N+1)|^2) \quad (2.55)$$

Combining equations 2.47 and 2.51 under condition j = N + 1, to

eliminate D(N), gives the last important result in the Levinson-Durbin recursion:

$$\sum_{i=0}^{N+1} a(N,i).R(N+1-i) = K(N+1).Q(N) \qquad (2.56)$$

Equations 2.54, 2.55 and 2.56, then, comprise the full Levinson-Durbin recursion. Working from a filter of zero order, the values of the coefficients R(0 ... N) or K(1 ... N) may be substituted into the equations to derive all terms (a, R, K and Q) for each ascending order in turn. Table 2A summarises how this is done.

2.3.4 The FIR Lattice

So far the discussion has dealt with filters of single specific orders, constructed using multiply, add and delay elements. This section shows how the same elements may be used to produce a structure which simultaneously gives the outputs of PE filters of many orders. The normal equations are shown to give this structure orthogonal properties. The important applications of these properties will be described in section 3.6.

Combining equations 2.32 and 2.54, and adding to the prediction error an argument denoting filter order, gives:

$$e(N,t) = \sum_{i=0}^{N-1} a(N-1,i).s(t-i) - K(N).\sum_{i=1}^{N} a^{*}(N-1,N-i)s(t-i) \qquad (2.58)$$

Adding a subscript to denote forward or backward errors, and applying equation 2.32 and its corresponding version for backward errors gives:

$$e_f(N,t) = e_f(N-1,t) - K(N).e_b(N-1,t-1) \qquad (2.59)$$

and similarly:

$$e_b(N,t) = e_b(N-1,t-1) - K^{*}(N).e_f(N-1,t) \qquad (2.60)$$

This is a method of forming the higher order forward and backward errors, given the lower ones. The recursion is started using the

input signal:

$$e_f(0,t) = e_b(0,t) = s(t) \tag{2.61}$$

This leads to the lattice equations. If one replaces 'e' with 'f' to denote the forward errors and 'b' to denote the backward ones, one sets:

$$f(0,t) = b(0,t) = s(t) \tag{2.62}$$

$$f(n,t) = f(n-1,t) - K(n),b(n-1,t-1) \tag{2.63}$$

$$b(n,t) = b(n-1,t-1) - K^*(n).f(n-1,t) \tag{2.64}$$

Figure 2.4 is the signal flow diagram of a very efficient structure which performs this calculation, giving all forward and backward error signals. This structure, owing to its shape, is called a lattice. Throughout this thesis, unless otherwise specified, all references to FIR lattice structures will refer to this particular kind of PE filter.

The orthogonality conditions of the lattice are an important property. These are derived from the normal equations 2.47 and 2.48, by substituting equations 2.16 and 2.29.

$$\begin{aligned} E < f(N,t).s(t-i) > &= Q(N) \quad | \; i = 0 \\ &= 0 \quad\;\; | \; i = 1 \ldots N \end{aligned} \tag{2.65}$$

$$\begin{aligned} E < b(N,t).s(t-i) > &= Q(N) \quad | \; i = N \\ &= 0 \quad\;\; | \; i = 0 \ldots N-1 \end{aligned} \tag{2.66}$$

This means, with reference to the equivalent transversal PE model (figure 2.5), that error signals are orthogonal to all component input signals except d(t), the signal on the zeroeth order tap. These signals are given by equations 2.29 and 2.30. The next step is that being orthogonal to a number of signals, an error signal is also

orthogonal to linear combinations of those signals. This gives the rest of the orthogonality conditions. for N > M:

$$E < f(N,t-j).f(M,t) > = 0 \quad \begin{array}{l|l} & 1 \leq j \leq N-M \\ Q(N) & j = 0 \end{array} \tag{2.67}$$

$$E < b(N,t).b(M,t-j) > = 0 \quad \begin{array}{l|l} & 0 \leq j \leq N-M-1 \\ Q(N) & j = N-M \end{array} \tag{2.68}$$

Using equation 2.56:

$$E < f(N,t).b(M,t-j) > = 0 \quad \begin{array}{l|l} & 1 \leq j \leq N-M \\ = K(M),Q(N) & j = 0 \end{array} \tag{2.69}$$

$$E < b(N,t).f(M,t-j) > = 0 \quad \begin{array}{l|l} & 0 \leq j \leq N-M-1 \\ = K^*(M),Q(N) & j = M-N \end{array} \tag{2.70}$$

Another important result, used in block processing algorithms, stems from the fact that f and b outputs from successive stages are orthogonal:

$$E < f(n+1,t).b(n,t-1) > = 0 \tag{2.71}$$

From equations 2.63 and 2.68:

$$E < (f(n,t) - K(n).b(n,t-1)).b(n,t-1) > = 0 \tag{2.72}$$

i.e. $$K(N) = E < b(n,t-1).f(n,t) > / E < b^2(n,t) > \tag{2.73}$$

So the normalised cross-correlation between the inputs to the taps of any one stage is equal to the value of the tap or PARCOR coefficient. This leads to a perspective of the lattice stage as a one-tap linear combiner with K as the optimum tap value. The optimum tap value, from the Wiener-Hopf equation, is the covariance of the two

signals divided by the power of the tap input signal. Since for stationary signals the two tap input powers are identical, the PARCOR coefficient value is indeed the cross-correlation between them. Since in general the covariance of A with B is the complex conjugate of the covariance of B with A, the two PARCOR coefficients will be complex conjugates of one another, as is shown by equations 2.63 and 2.64. However, for non-stationary signals the input powers will not be identical and the forward and backward PARCOR coefficients will assume different absolute values.

Lattice algorithms in which two PARCOR coefficients of a stage are permitted to have separate values are referred to as two-valued structures. Those in which they are constrained to be complex conjugates are called one-value types.

Messerschmitt [22] pointed out that were the delays in the lattice structure to be replaced by all-pass filters, it would still exhibit its characteristic orthogonal properties. This may be understood by regarding an all-pass filter as a delay element whose delay varies with frequency. Since a fixed delay is a special case of an all-pass filter, Messerschmitt's form is a more general case of a lattice.

2.3.5 Alternative Lattice Elements

Makhoul [23] reported a number of different structures which could be used as lattice elements. These offer varying degrees of trade-off between coefficient calculation and simplicity of hardware. The general structures are shown in figure 2.6 and may be used as one, two, three or four multiplier lattices. In all cases the multiplier coefficients or amplifier gains are set so that the orthogonality conditions are fulfilled.

In the four multiplier version, figure 2.6(a), one may define an arbitrary gain factor α, which is proportional to the voltage gain of the stage. Thus if amplifier A is set to α and B to $\alpha.K(n)$, the lattice orthogonality conditions are fulfilled. If α is set to unity, the structure becomes the familiar two multiplier lattice of

figure 2.4. It is often useful to set α to a value which gives equal input and output powers. This often saves using a separate calculation to measure the power. It is also clear from the diagram that changing over signs and values of the gains of A and B will make the outputs appear on opposite terminals.

In the three multiplier version, figure 2.6(b), one may also use a gain factor of α to vary the gain. Thus C may be given a gain of α and D, a gain of $(K(n).\alpha/(1+K(n))$. Setting D to $(\alpha/(1+K(n))$ will interchange outputs. Setting α to unity will give the economy of a one multiplier lattice. Again, a value of α may be found which normalises the power out of the three multiplier lattice. Alternative forms of normalised three and four multiplier lattices are one and two multiplier types with normalising amplifiers in the output signal paths.

While normalised and one multiplier lattices are undoubtedly useful, two multiplier lattices have been found to be most economical in hardware for gradient algorithms [24]. The rest of this thesis will thus concentrate on two multiplier lattice elements.

2.3.6 Gramm-Schmitt Orthogonalisation

Gramm-Schmitt orthogonalisation [25, 26], is the production of an orthogonal set of vectors, given a non-orthogonal set. It has relevance to a lattice structure in that it produces orthogonal signals given non-orthogonal input samples.

The first vector given is taken as the first vector of the output set, just as the lattice starts with $b(0,t) = s(t)$. The second vector is then compared with the first to find that component which is common to it. This is similar to setting $K(1)$ equal to the cross-correlation between $s(t)$ and $s(t-1)$. The common component is then subtracted, leaving only a vector orthogonal to the first. This is taken as the second vector of the output set. Expressed in lattice mathematics:

$$b(1,t) = s(t-1) - K(1).b(0,t) \qquad (2.75)$$

Gradually, by adding new vectors (further delayed versions of the input signal) and subtracting the components correlated with other vectors (signals) in the output set, a complete set is built up.

The analogy between the lattice structure and Gramm-Schmitt orthogonalisation is not a complete one. Although the results are identical, the method by which the Gramm-Schmitt set is built up involves operations on the input signal or vector set; the lattice performs its operations on existing output signals. Nevertheless, the result in both cases is an orthogonal output set whose orientation depends entirely on the order and orientation of the input set.

2.3.7 Whitening

It was said at the beginning of section 2.3.3 that the coefficients of the PE filter have a special relationship to the autocovariance function of the input signal. This implies, via Wiener-Khinchine theorem [20], that the coefficients are related to the spectrum of the input. This section will show how the output of a PE filter is whiter, i.e. spectrally flatter than its input. It is also shown how to derive the spectrum of the input from the coefficients of the filter.

Let there be a stable AR process fed with a signal which is spectrally white A(t), i.e. the input samples are uncorrelated. Let the output of this AR process, s(t) feed a forward PE filter of the same order, N. Let the coefficients of the AR filter be g(n,i), the arguments of the variable to mean the same as for the PE filter coefficients. Let the PE filter coefficients be a(n,i) and u(n,i) is defined as (g(n,i) - a(n,i)). Thus the PE filter output, f(N,t), is:

$$f(N,t) = A(t) + \sum_{i=1}^{N} u(N,i).s(t-i) \qquad (2.76)$$

Using the definition that A(t) is uncorrelated with previous samples of itself:

$$E < f^2(N,t) > = E < A^2(t) > + E <(\sum_{i=1}^{N} u(N,i).s(t-i))^2 > \quad (2.77)$$

Substituting Q(N) for the expectation of PE power:

$$\frac{d(Q(N))}{du} = E < \sum_{i=1}^{N} \sum_{j=1}^{N} u(N,i).u(N,j).s(t-i).s(t-j) > \quad (2.78)$$

Setting the differential to zero, one finds that a general solution for minimum PE output power is that u(N,i) is zero for all i. Thus the coefficients of the PE filter are equal to those of the AR filter. From equation 2.76, since the input to the AR filter was white, the output from the PE filter must be also. Furthermore, since the stability conditions of the AR filter force its poles to lie within the unit circle of the z-plane, then so must the PE filter's zeroes. Thus the PE filter is a minimum phase filter, having the maximum of energy in the earlier parts of the impulse response.

The foregoing proof has explained how a PE filter perfectly whitens those signals generated by AR processes of the same order or lower. Since the power spectrum of any process, AR, MA or ARMA may be approximated to increasing degrees by AR processes of increasing orders [18], a PE filter will maximally flatten the spectrum of any signal fed into it, no matter what the generating process was [21].

Figure 2.7 shows the effect on the spectrum of an MA process, of passing it through a PE filter. The white generating signal was filtered by convolving it with the three point impulse response R11 (appendix D), which is shown in figure 2.8. The zeroeth order spectrum is that of the PE filter input, showing a maximum at DC and a minimum at the Nyquist frequency. The figure shows the spectra of the outputs of successive filter stages, becoming flatter as the order rises.

Since the generating process is MA, it can never be completely whitened by a PE filter of finite order. Nevertheless, the spectrum

of the output of the seventh order filter appears flat to the human eye. The family of spectra illustrates the concept of pole fitting. According to this viewpoint, the PE filter 'sees' the input signal as a number of poles (an AR filter is indeed an all-pole filter) and locates zeroes in its own passband so as to cancel them out. Thus the maximum at DC in the input signal has been removed by a high-pass filter in the first-order spectrum, leaving a maximum in the middle of the spectrum. This maximum is in turn cancelled by a mid-band zero in the third-order spectrum.

Figure 2.9 shows the system impulse response of the seventh order PE filter, graphically demonstrating the difference between whitening and equalisation. Far from regaining a unit impulse, the system impulse response has been expanded to give an exponentially decaying ringing impulse, centred on the Nyquist frequency. Both of these aspects are consistent with whiteness: The channel carries information at frequencies up to but not including Nyquist. The exponential decay of the all-pass system impulse response gives maximum spread of power spectral density evenly over all frequencies [73].

2.3.7.1 Sinusoidal Inputs

The behaviour of the PE filter when fed deterministic inputs is the foundation of an important branch of spectral estimation. The mathematical description starts rather obliquely:

Suppose that there is an input signal for which Q(N) is zero, but Q(N-1) is not. From equation 2.55 the absolute value of K(N) must be unity. Using the model of an AR generating process, this means, from recursive equation 2.54, that coefficient a(N) of PE and AR filters also has an absolute value of unity. This will put the AR filter on the exact threshold of instability, since R(N) of its output sequence will equal R(0) in magnitude. Furthermore, since Q(N) is zero, this means that the input to the AR filter is zero. Since Q(N-1) is non-zero, the only inference is that the AR filter model is in oscillation and the input to the PE filter is one or a number of sines.

The Paley-Wiener criterion [28] may be applied to the PE filter, i.e. a transfer function may be identically zero at a spot frequency but not over a band of frequencies. Since Q(N) is zero, this implies that the input could not have been a broadband signal, but a collection of sines. Moreover, since Q(N) is zero, this must be the unique LMS solution for the filter coefficients.

A notch at a certain frequency is represented in the z-plane by a conjugate pair of zeroes on the unit circle. This gives a transfer function of:

$$F_w(z) = \prod_{i=1}^{N/2} (1 - 2 \text{ Cos } (w(i).t)z^{-i} + z^{-2}) \qquad (2.79)$$

Where w(i) is the angular frequency of the notch. This confirms that the highest order coefficient of the filter is +1 and thus K(N) is always -1. Thus the input frequencies may be determined by factorising the filter coefficients and solving for w(i).

Figure 2.10 shows the periodograms (power spectra) of the coefficients of a set of PE filters, fed with a combination of five sinusoids. The inverse of the spectral response is shown, so the notches appear as peaks, reflecting the spectral density of the input. As in figure 2.7, the figure shows the attempts by filters of increasing order to whiten the output by fitting zeroes into the transfer function. This becomes totally successful when the order of the filter is twice the number of input sinusoids. Since Q(N) is then zero, it is impossible to calculate K(N+1) using equation 2.56 and the eleventh order filter does not exist. One may infer that a(N, 11 .... ∞) and K(N+1 .... ∞) are zero for higher order filters by adding gradually reducing amounts of noise to the input and calculating the filter coefficients.

Should a little white noise be added to the filter input along with the sines, then Q(N) will no longer be zero. The spectrum of the transfer function, however, will alter very little, depending on the amplitude of the noise. In this case spectral whitening will not require the complete notching out of portions of the spectrum, but

merely the reduction of the amplitudes of the sines to the level of the output noise. The zeros will thus no longer lie on the unit circle, but slightly off it, by an amount depending on the amplitude of the noise. Factorising the transfer function of the PE filter will thus give an indication of the amplitude and frequency of the sinusoids.

The assumption that the input signal consists of a number of sinusoids against a background of white noise is the initial premise of Pisarenko harmonic decomposition [29]. In this technique the transfer function of the PE filter is indeed factorised to give the amplitude of the noise and the amplitudes and frequencies of the sines.

2.3.7.2 Autoregressive Spectral Estimation

The technique known as AR spectral estimation [30] is a general one for transforming PE filter coefficients to estimate the input spectrum. The input signal is assumed to have originated from an AR process with the coefficients of the calculated PE filter. Thus the spectral response of the PE transfer function is the inverse of that of the generating AR process. The Fourier transform of the PE filter coefficients is thus calculated. The square of the modulus then gives the power spectrum of the PE filter. This is then inverted, as was done in figure 2.10, to give the power spectrum of the AR process. Both figures 2.7 and 2.10 were generated by calculating the autocovariance function of the input signal and then applying the Levinson-Durbin recursion to get the PE filter coefficients.

The AR spectral estimation technique is often more useful than that of straight Fourier transformation of the input signal. Fourier transformation suffers from the windowing problems of MA estimation techniques [29], and is most accurate in portraying the zeroes of a signal. The AR spectral estimation technique accurately portrays poles, i.e. spectral peaks, at the expense of detail in the troughs. In most applications it is more important to detect regions of greater spectral density than those of low density.

2.4 Summary

This chapter was not intended as a general introduction to digital filtering. Instead, it has covered the time domain properties of MA filters. The discrete time approach necessary in digital filtering has greatly simplified the mathematics.

The importance of the Wiener filter has been highlighted, both in general filtering and in equalisation. The calculation of the filter coefficients has been described. The calculation depends on the concepts of correlation and its antithesis, orthogonality.

Orthogonalisation, the creation of orthogonal sets of vectors or signals, has been introduced. The next chapter will demonstrate the importance of the orthogonalising effect of the PE filter in adaptive filtering. Orthogonality between successive signal samples is referred to as whiteness - a reference to the visual spectrum. The technique of calculating a filter to whiten a signal has been described and it has been shown how this step may help to estimate the spectrum of the input signal.

a) $a(N,0) = 1.$

$a(N,i) = 0, \quad \text{if } i \leq 0 \text{ or } i \geq N.$

$a(N,i) = a(N-1,i) - K(N).a^*(N-1,N-i).$

$a(N,N) = - K(N).$

b) $Q(0) = R(0)$

$Q(N) = Q(N-1).(1 - |K(N)|^2).$

c) $\sum_{i=0}^{N} R(N-i).a(N-1,i) = Q(N-1).K(N).$

If R(0...N) are known;

1) use (b) to get Q(0)
2) use (c) to get K(1)
3) use (a) to get a(1,0&1)
4) use (b) to get Q(1) etc

If K(1...N) and Q(N) are known;

1) use (b) repeatedly to get Q(0...N) and R(0)
2) use (a) repeatedly to get a(0...N,0...N)
3) use (c) repeatedly to get R(0...N)

Table 2A The Key Equations of the Levinson-Durbin Recursion

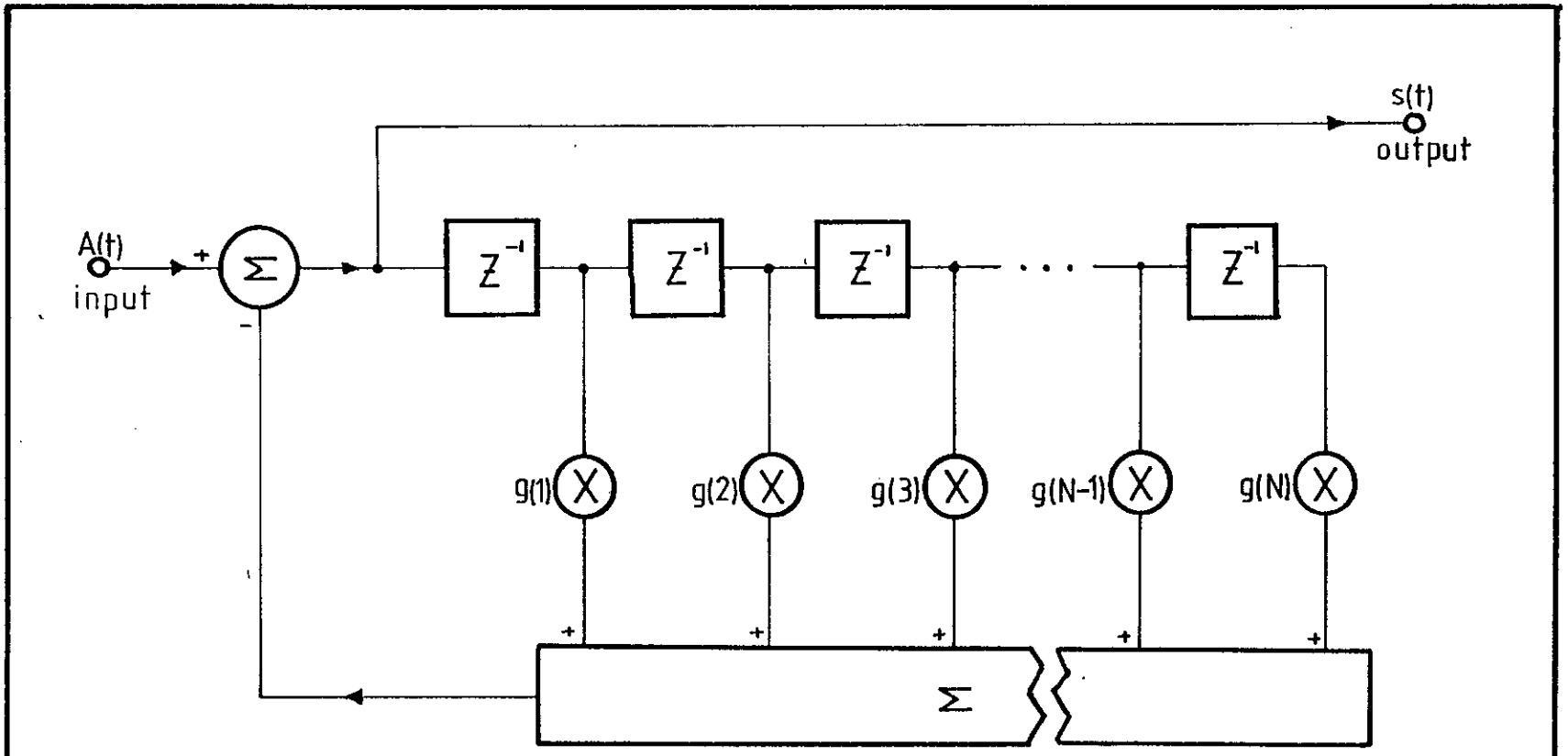


Fig 2.1 The autoregressive filter model

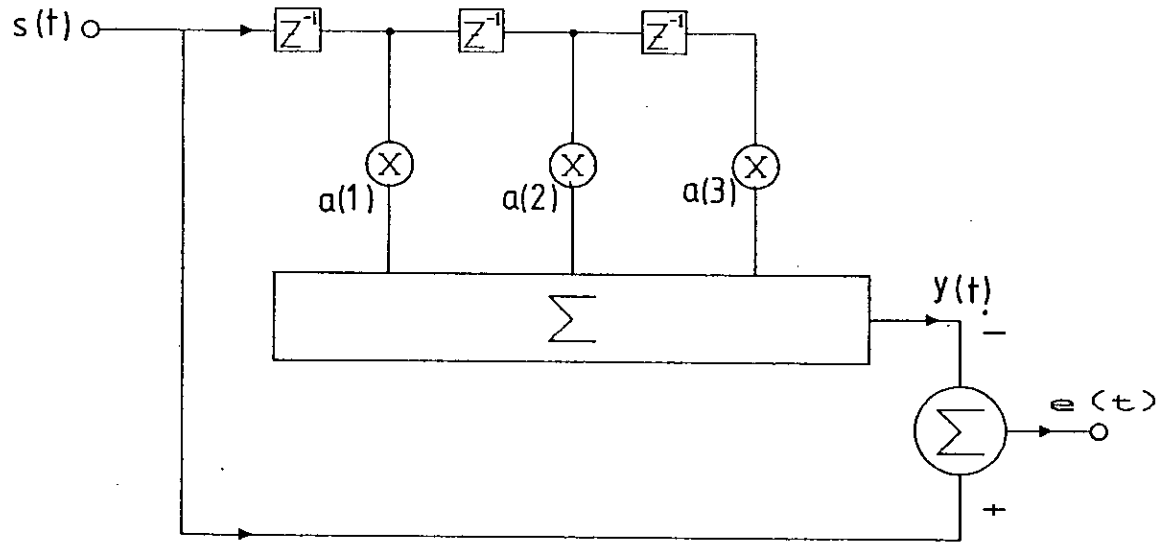


Fig 2.2 A third-order forward prediction-error filter

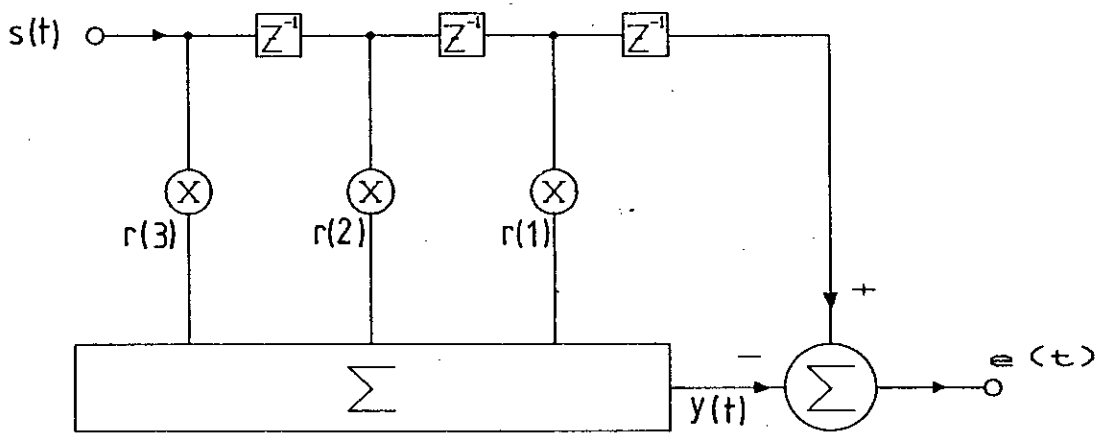


Fig 2.3 A third-order backward prediction-error filter

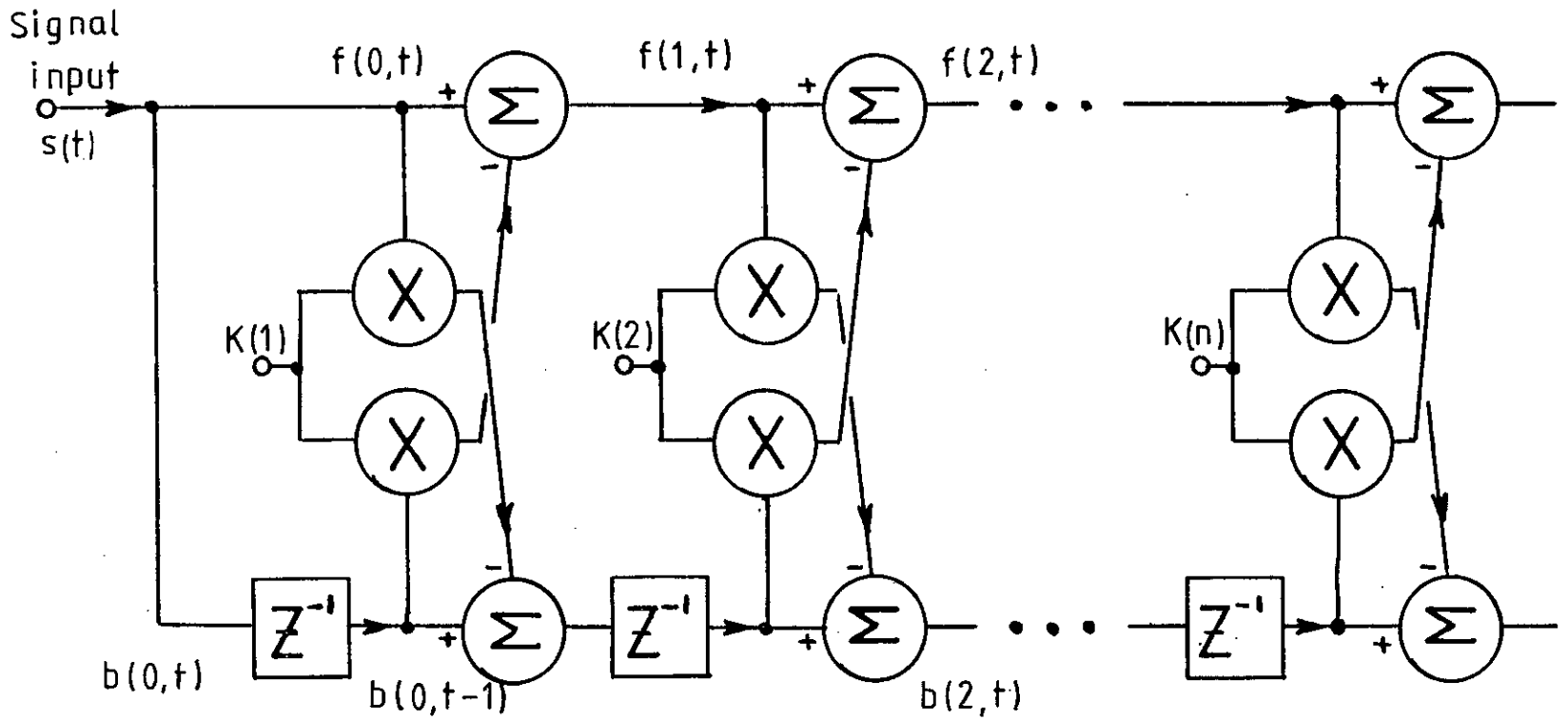


Fig 2.4 The lattice structure

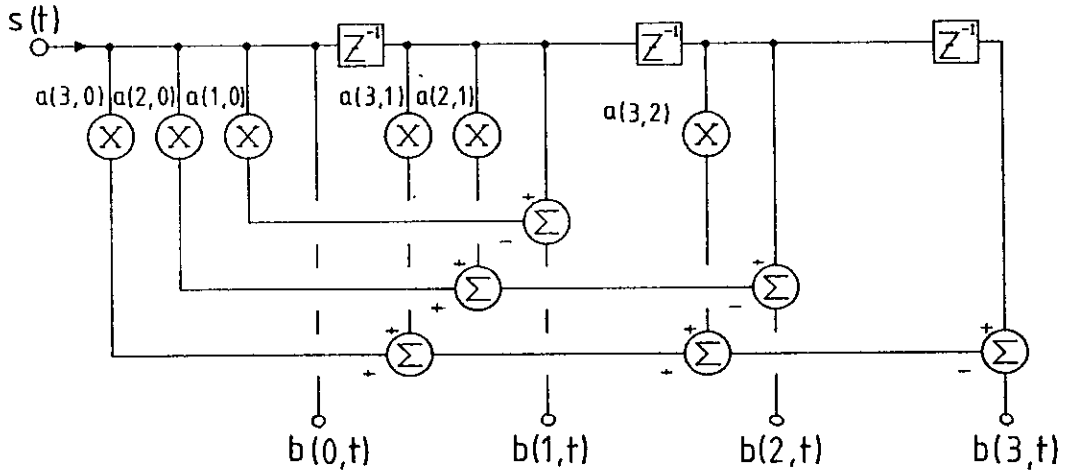


Fig 2.5 The transversal equivalent of a lattice structure

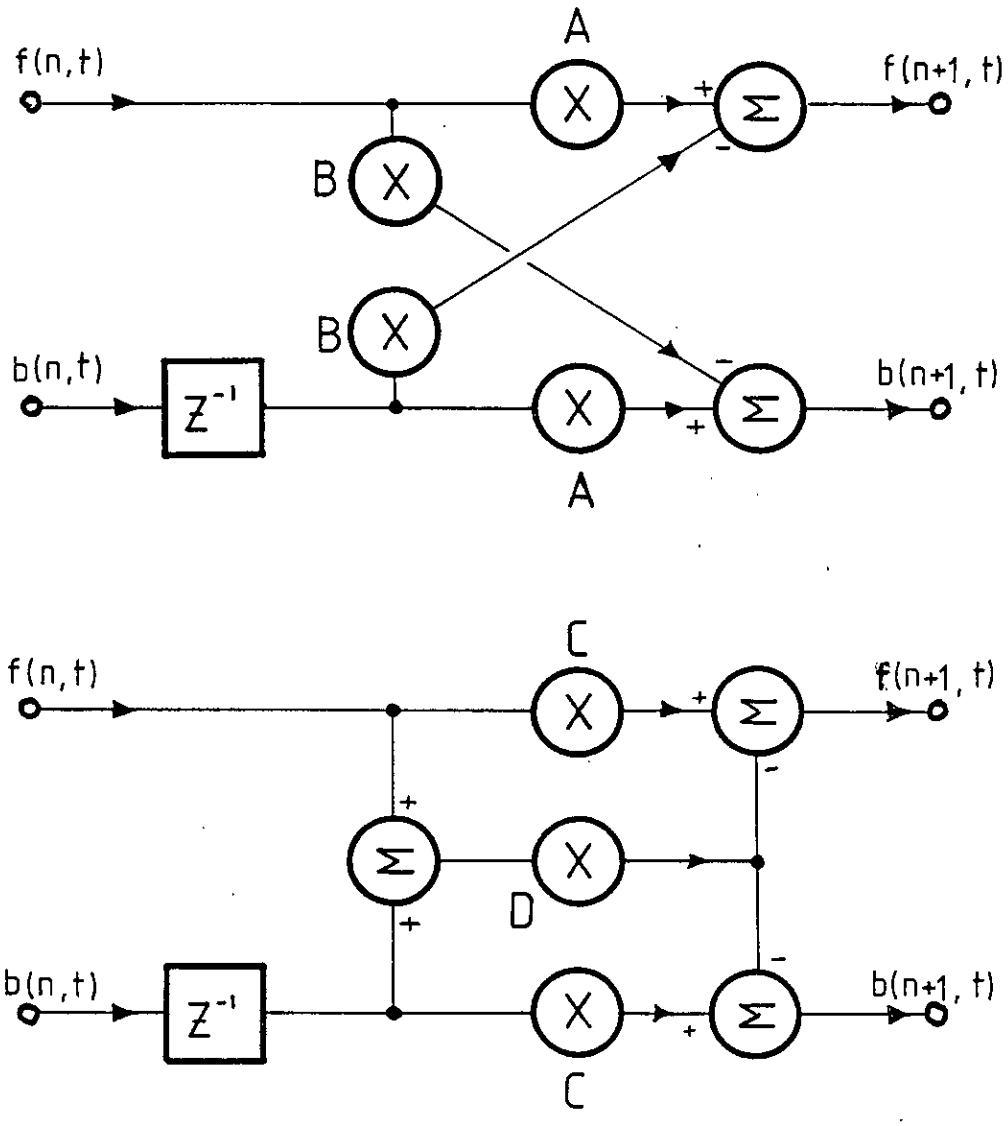


Fig 2.6 Four multiplier and three multiplier lattice elements

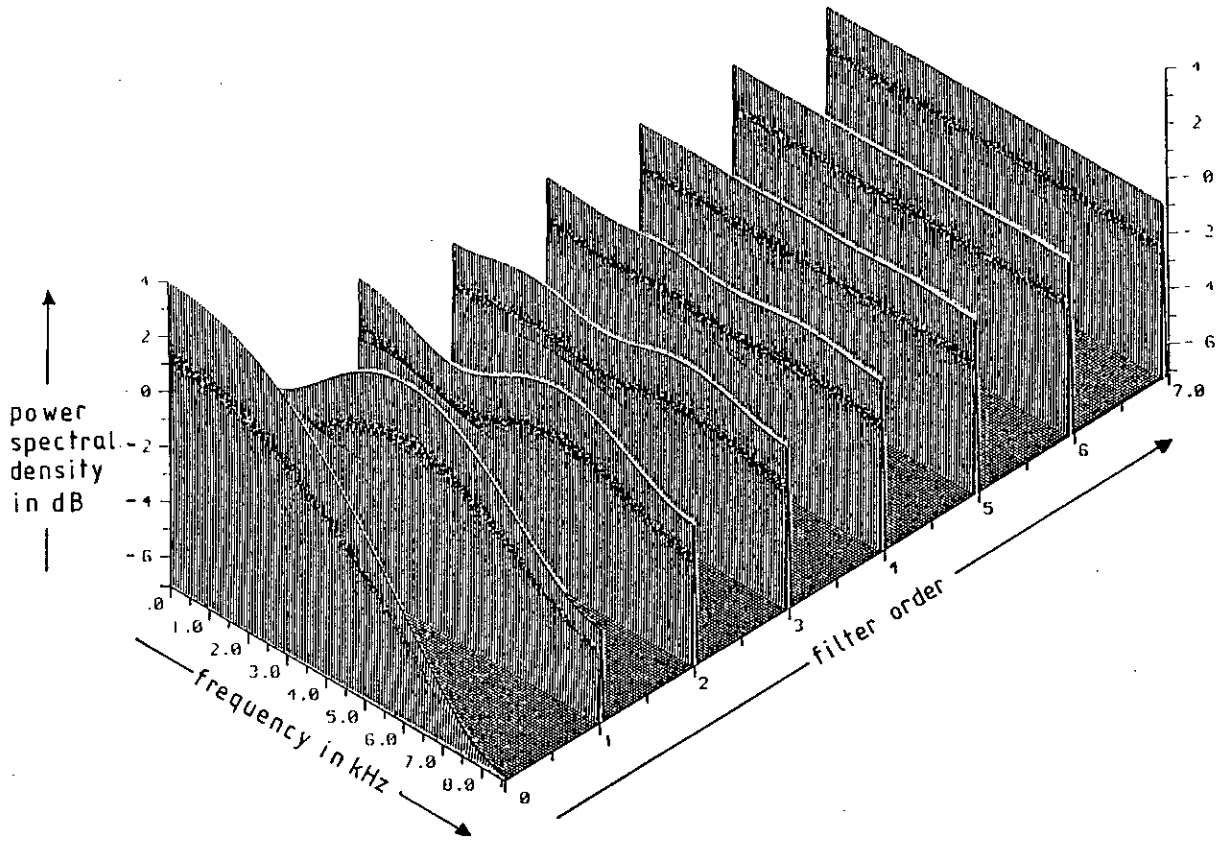


Fig 2.7 The power spectra of the outputs of various orders of PE filter, showing the progressive whitening of a signal

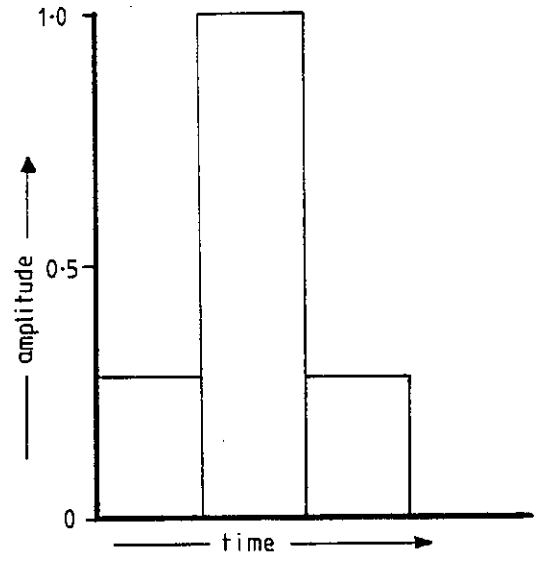


Fig 2.8 The R11 channel impulse response

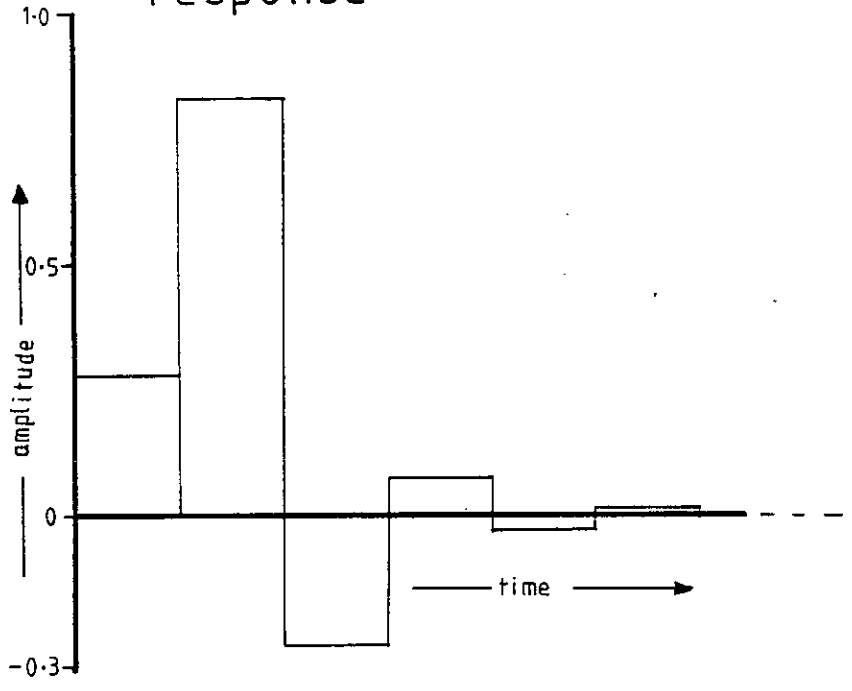


Fig 2·9 The system impulse response of an R11 channel after whitening

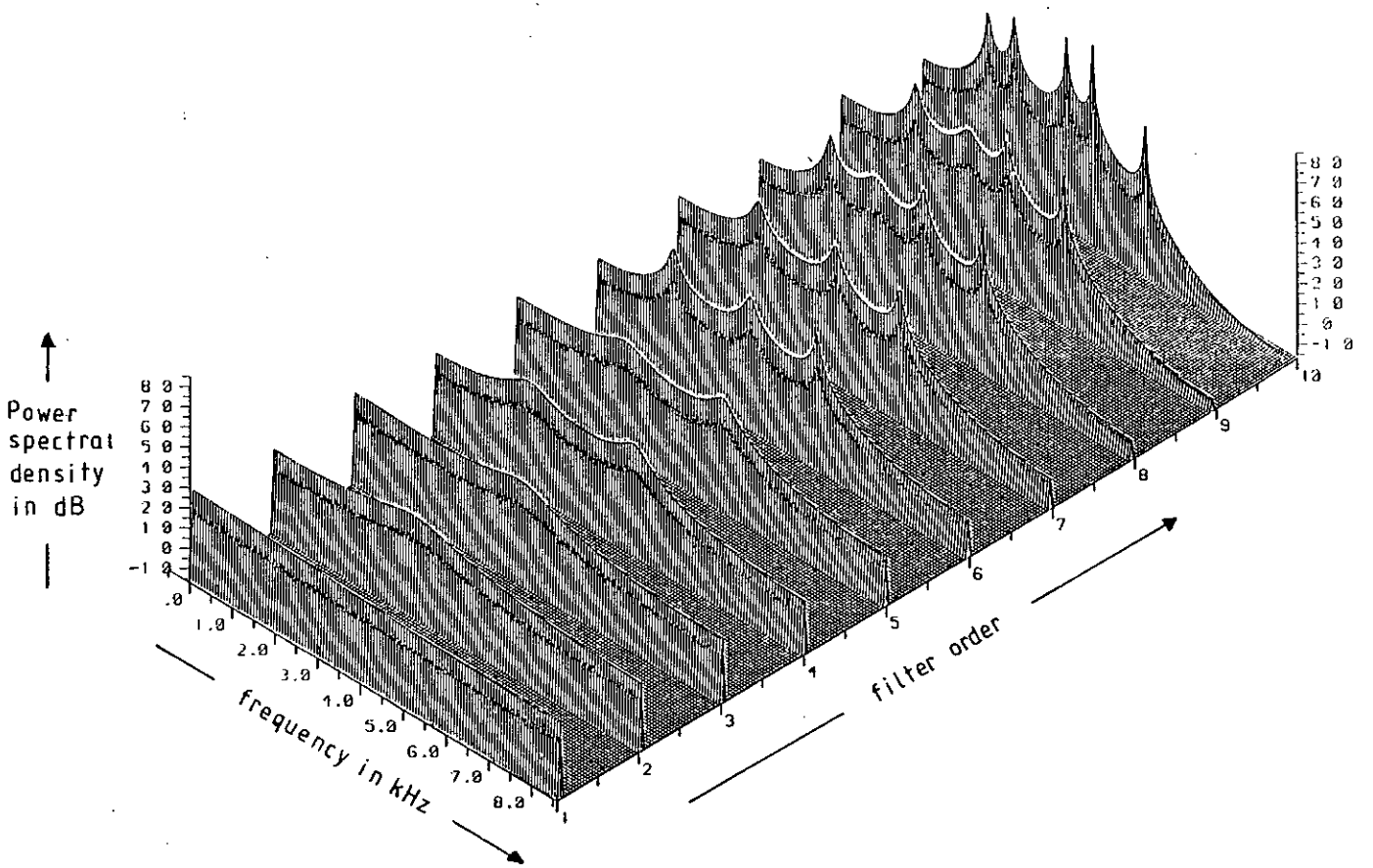


Fig 2.10 The inverse of the frequency response of PE filters of increasing order. The input signal is a collection of five sinusoids.

Chapter Three

ADAPTIVE FILTERING STRUCTURES

This chapter extends the work of the previous chapter to structures and algorithms which can adjust their own coefficients. Many distant relations of the lattice equaliser are mentioned, such as ARMA models and frequency domain filters. The main discussion, however, is directed towards the last sections which describe the lattice equaliser itself.

Section one describes the two main models for an adaptive filter, open-loop and closed-loop. Section two explains the linear combiner, a general structure from which most closed-loop models are derived. Its characteristics are described in detail, because they strongly influence the properties of the lattice and transversal adaptive filter.

Section three discusses the adaptive transversal filter. Its shortcomings are related to the Toeplitz nature of the input signal's autocovariance matrix. Hence the slowing effect of the matrix eigenvalue ratio on the rate of convergence (ROC) of the filter is related to the frequency response of the channel and the resulting intersymbol interference (ISI). Section four relates the transversal filter to the rest of the Chang family of filters or equalisers. It also describes other family members, notably the lattice equaliser whose special features avoid some of the problems experienced in transveral filters.

Adaptive prediction-error filters themselves may only rarely be used as adaptive equalisers as they whiten rather than equalise [55]. The most useful form is the lattice structure. In section five the adaptive lattice is described as a network of one-tap adaptive transversal algorithms. The mathematical equivalence of lattice and transversal PE structures is also shown.

Section six, although short, may be regarded as the main section of the chapter. It draws on previous sections to explain the

operation of the lattice equaliser. Two forms of equaliser are described, open-loop and closed-loop. The former is described here and a computer simulation is presented which uses the mathematics of section 2.3.3. The rest of the thesis is devoted to the closed-loop version.

3.1 The General Adaptive Filter

There are two main models for the adaptive filters: open-loop and closed-loop. The latter will be described first.

Filters are used to enhance desirable components and to attenuate the undesirable components of a signal. For each input, there is an idealised output which is required from the filter. Often the filter can only produce an approximation to this ideal response. In adaptive filtering the ideal response is input to the filter as a training signal.

The error signal (defined in equation 2.14) is then fed into an algorithm for improving the filter coefficients in the direction of some, hitherto unspecified, optimum. For most algorithms the filter coefficients may be started from an arbitrary setting and will eventually converge to a value close to their optimum.

The filter signal names may sometimes be misleading. Equalisation is an inverse modelling application which compensates for a distorting channel, in inverse modelling the distorted signal is fed to the s(t) port while the training signal feeds to the d(t) port. In direct modelling applications, however, the known training signal is fed into the s(t) filter input port. In most applications there is a finite error signal even after the filter has converged, leading to random fluctuations in filter coefficients. In many cases such as the PE filter, the output signal labeled "error" in figure 1.4 is actually the required output.

Often there is no separate training signal available. In prediction error (PE) filtering, which is central to this thesis, the training signal is obtained by delaying the filter input. In data-

directed feedback (DFB) operation the filter output is detected to estimate the transmitted signal value. The voltage corresponding to the value of the symbol is then fed back to the filter for coefficient optimisation (figure 3.1). Clearly this assumption that the detector output is valid will only work if the decisions are nearly always correct, as incorrect decisions will cause the filter coefficients to diverge from the optimum values. Thus DFB will only work under a limited set of conditions.

Open-loop filters, figure 1.3, do not use the recursive self-correcting algorithm of closed-loop systems. They use only the input data to form an estimate of the filter coefficients. In practice this is computationally more demanding but by removing the recursive feedback faster convergence can be achieved.

In this general description the structure of the filter itself has not so far been discussed. It may be AR, MA or ARMA and open and closed-loop adaptive algorithms may be found for all three. In this thesis, however, the emphasis is on MA adaptive filtering, which is by far the simplest and most popular.

3.2 The Adaptive Linear Combiner

The linear combiner problem may be generalised as follows: Consider a set of input signal samples s(0,t) .... s(N-1,t). Some of the signal components will be correlated with the training signal in some way and may also carry unwanted signals which one may call "noise". The linear combiner (figure 3.2) forms a single weighted combination y(t) of these signal samples. The combination process may be expressed as:

$$y = \underline{s}\underline{G}^T \qquad (3.1)$$

Where the vector $\underline{G}$, G(1 ... N-1), represents the weightings applied to input $\underline{s}$, s(1 ... N-1, t). The process is made adaptive by applying equation 2.14 to derive an error signal. The adaptive algorithm then seeks to minimise the error signal according to some criterion. In many cases the ideal criterion is the peak error, but

the most usual (perhaps because of its simpler mathematics) is the mean error power. The latter is referred to as the least mean square (LMS) criterion.

There are many algorithms which may be used to minimise the error power. Many such as Newton-Raphsen [31] and Box's derivateless method [32] require a knowledge of the error power at more than one tap setting to be retained in memory. A random search method [33] involves randomly perturbing the tap settings by a small amount and ascertaining if it reduces, or increases, the error power. The latter corresponds to the human method of optimisation by experiment. Another technique with strong links with human methods is the LMS gradient technique [53], often attributed to Widrow. This method is so relatively simple to implement and effective that many adaptive filter designs are based on it. It is thus discussed in detail in the next section.

3.2.1 The LMS Algorithm

The principle of the LMS algorithm is simple: Each tapweight setting gives a certain error power. The gradient of the power with the tapweight values is:

$$\Delta = \partial E < e^2 > / \partial \underline{G} \qquad (3.2)$$

Where E <.> is the expectation operator denoting a mean value.

In order to reduce this error it is only necessary to move down the gradient by a small step. In the rest of this discussion the expectation operator is dropped and the instantaneous error power is taken as a noisy but unbiased estimate of the mean square error. Using μ to represent the stepsize, the following recursion shows how the tap weight value G(t) can be updated at the following time instant to G(t + 1):

$$\underline{G}(t+1) = \underline{G}(t) - \mu.(\partial E < e^2 > / \partial \underline{G}(t)) \qquad (3.3)$$

Now from 2.14 and 3.1.

$$e^2(t) = (d - \underline{s}\underline{G}^T)^2 \tag{3.4}$$

$$\partial(e^2)/\partial\underline{G} = -2\underline{s}(d - \underline{s}\underline{G}^T) \tag{3.5}$$

Substituting 2.14 again, and substituting back into 3.3 one gets,

$$\underline{G}(t+1) = \underline{G}(t) + 2\mu\underline{s}e \tag{3.6}$$

The multiplication by 2 will be incorporated in the stepsize from now on. The above simple equation is the LMS adaptive algorithm most often used in closed loop adaptive filters [5].

At the minimum of the error curve the weighting values become optimal and the gradient reduces to zero. The latter implies that this point is reached asymptotically after an infinite number of recursions. At the zero point, equation 3.5 gives:

$$\underline{G}(opt) = d\underline{s}.(\underline{s}^T\underline{s})^{-1} \tag{3.7}$$

Now the expectation of the product $\underline{s}\underline{s}^T$ is the cross-covariance function between the input signals, which we will call $\underline{R}$. Also the expectation of $\underline{s}d$ is $\underline{P}$ the cross-covariance function between the input and the training signal. Thus the equation may be rewritten:

$$\underline{G}(opt) = \underline{P}\underline{R}^{-1} \tag{3.8}$$

This is the Wiener-Hopf equation, equation 2.18 stated in a more general way, [4] which gives the optimum tap values for the above conditions. Since from equation 3.4 the gradient is a multidimensional paraboloid, the recursion will always tend towards a global minimum, the optimum solution, since there are no local minima.

Note that the instantaneous values of the signal have been taken in this discussion and assumed to be representative of the average stationary values. In practice this assumption leads to a noisier

convergence than would be expected (c.f. section 3.2.4), [7], but the optimum is approached very closely.

The LMS algorithm has proved very popular in the adaptive filtering field because of its simplicity and computational economy. In fact equation 3.6 has been successfully used in more applications than are covered by this derivation. It has been shown to track non-stationary signals [34] and successfully converges for filters not having a linear combiner structure. The algorithm has been shown to work for AR adaptive filters [35] and even lattices (c.f. section 4.4.3). Nevertheless not all applications have paraboloid error surfaces, e.g. the AR filter [36], and it is possible that the minimum found will not be a global one.

3.2.1.1 Orthogonality Conditions

The operation of the lattice equaliser depends on the orthogonality condition of the LMS algorithm, which will be derived here. The condition is that after convergence the error power output from the combiner is orthogonal to the input signals. Since at convergence the gradient is zero, then from (3.5);

$$\underline{0} = -E < 2\underline{s} (d - \underline{s}\underline{G}^T) > \tag{3.9}$$

$$= E < \underline{s}.e > \tag{3.10}$$

This shows that the expected value of the product is zero, i.e. that it has zero mean. A second result of the zero gradient condition will be used in section 5.3: Since the rate of change of expected error power with tap-values is zero, this means that the mean error power is zero-sensitive to small pertubations in the tapweights. Section 3.2.4 will show that zero-sensitivity is a first-order effect only and that the second order effects are important also.

3.2.1.2 Convergence Conditions

So far the discussion has shown how the LMS recursion converges to the Weiner-Hopf solution. The conditions under which convergence

can be guaranteed have not yet been explored. For this thesis, which discusses fast-convergence algorithms, the rate of convergence (ROC) is also of importance.

From equations 3.3 and 3.5

$$E < \underline{G}(t+1) > = \underline{G}(t) + \mu \underline{s} \, (d - \underline{s}\underline{G}^T(t)) \tag{3.12}$$

$$= (\underline{I} - \mu \underline{s}^T \underline{s})^{t+1} \, \underline{G}(0) + \mu \sum_{i=0}^{t} (\underline{I} + \mu \underline{s}^T \underline{s})^t \underline{s} d \tag{3.13}$$

$$\text{So} \quad E < G(\infty) > = (\underline{I} - \mu \underline{s}^T \underline{s})^{\infty} \underline{G}(o) + \mu \sum_{i=0}^{\infty} (I + \mu \underline{s}^T \underline{s})^t \underline{s} d \tag{3.14}$$

Now the right-hand term on the right-hand side is identical to the Wiener-Hopf solution, $\underline{P}\underline{R}^{-1}$. If the left-hand term, which includes the arbitrary starting point, can be shown to converge to zero, then convergence will be proven.

The left-hand term may be re-expressed as:

$$LHT = (\underline{I} - \mu \underline{R})^{\infty} \underline{G}(o) \tag{3.15}$$

Letting $\underline{V}$ be the matrix of eigenvectors of $\underline{R}$, and $\underline{\Lambda}$ the corresponding diagonal matrix of eigenvalues:

$$\underline{R} = \underline{V}\underline{\Lambda}\underline{V}^{-1} \tag{3.16}$$

So

$$LHT = (\underline{V}\underline{I}\underline{V}^{-1} - \mu \underline{V}\underline{\Lambda}\underline{V}^{-1})^{\infty} \underline{G}(o)$$

$$= (\underline{V}(\underline{I} - \mu \underline{\Lambda})\underline{V}^{-1})^{\infty} \underline{G}(o) \tag{3.17}$$

This will converge to zero if the magnitude of each term for the sum in the inner bracket is less than unity, i.e.:

$$|1 - \mu\lambda| < 1 \tag{3.18}$$

where λ is any eigenvalue , or:

$$0 < \mu < 2/\lambda(max) \tag{3.19}$$

This introduces the concept of "modes" related to the individual eigenvectors of the cross-covariance matrix. Since the eigenvectors are orthogonal, the modes are mutually independent. From 3.15 a mode will converge fastest if, subject to 3.19,

$$\mu = 1/\lambda(i) \tag{3.20}$$

Where $\lambda(i)$ is the eigenvalue corresponding to that mode. In fact μ has normally to be a constant for all modes. Selecting the maximum possible stepsize permitted by equation 3.19, the inner term of equations 3.17 converges as:

$$(1 - \lambda(i)/\lambda\ (max))^{n} \rightarrow 0 \text{ as } n \rightarrow \infty \tag{3.21}$$

i.e. the error power decays exponentially with the time constants of the various decaying modes selected by their respective eigenvalues [34].

A glance at figure 3.3 will show how this convergence operates. The largest eigenvector initially contributes most to the error power, but it converges most rapidly leaving the second largest. Eventually the most significant contribution comes from the smallest eigenvector, which converges most slowly. In other words the rate of convergence of each mode is controlled by the magnitude of the eigenvalue, which sets the loop gain for that particular mode. This explains why the rate of convergence in the latter stages of convergence is governed by the eigenvalue ratio (EVR), the ratio of maximum and minimum eigenvalue. Substituting $\lambda(min)$ into equation 3.21 does indeed produce an EVR term which dictates the ROC of the slowest mode. Thus $\lambda(max)$ sets μ (max) for maximum initial convergence but $\lambda(min)$ ultimately controls convergence rate for high iteration numbers. The overall ROC is thus governed by eigenvalue ratio.

It is very difficult to estimate the maximum eigenvalue in practice, so the signal power R(0) is often used to estimate the optimum stepsize. Thus the rate of convergence of equation 3.15 will

depend on the ratio of the minimum eigenvalue to the power estimate, i.e. the normalised value of the minimum eigenvalue. A glance at figure 3.3 will show that over the main region of convergence (i.e. until $\lambda(\min)$ becomes dominant), $\lambda(\min)$ gives too pessimistic a picture. It has been found while running simulations that $\sqrt{\lambda(\min)}$ provides a better approximation for estimating the overall rate of convergence.

3.2.2 The Distributed Linear Combiner

The linear combiner algorithm discussed in the previous section updated all taps simultaneously using a global algorithm. For this reason it will be referred to in future as the global linear combiner to differentiate it from the distributed combiner algorithms. The distributed combiner updates its taps using many independent processor loops which are each independantly adjusted for optimum convergence. In this way it avoids the limitations on stepsize caused by eigenvalue spread. The penalty for this reliable rate of convergence is a greater inaccuracy in the tap values and a limitation on the categories of signal which can be accommodated.

Let there be a signal, A(t), which is formed from a linear combination of orthogonal signals s(1 ... M, t). If the component signals, s(1 ... M,t), are mutually orthogonal, their cross-correlation matrix is diagonal. However, suppose that only some of the signals, s(1 ... N, t) are available for recombination. How best may they be optimally combined?

Since the input covariance matrix is diagonal, the inverse of the matrix will also be diagonal, with the terms being the individual inverses of the terms of the cross-covariance matrix. Thus the optimum value of the tapweights derived from the Wiener-Hopf equation will be dependent solely on the power estimates of the orthogonal input signals and their individual cross-covariances with the training signal. Thus the calculation of the optimum tap value for one signal is independent of the values of the other signals and may be done using individual adaptive algorithm loops.

Since the input signals are mutually uncorrelated, adding or subtracting some of the input signals from the training signal will not affect its cross-covariance with the rest of the input signals. The tap-values will remain totally independent of one another.

The structure of the signal paths of the global and distributed combiners are identical. The differencies are in the recursive algorithm loops. Both algorithms presented here use the same tap updating technique, derived from the LMS algorithm:

$$G(n, t+1) = G(n,t) + \mu(n,t).s(n,t).e(n+1,t) \qquad (3.23)$$

The variable stepsize implied in the equation will be discussed in section 3.2.3. There are two methods of obtaining the error signals for the recursion in distributed combiner. These are the order-independent, figure 3.4(a)

$$e(n+1, t) = d(t) - G(n,t).s(n,t) \qquad (3.24)$$

and the order-dependent, figure 3.4(b)

$$e(0,t) = d(t) \qquad (3.25)$$

$$e(n+1,t) = e(n,t) - G(n,t).s(n,t) \qquad (3.26)$$

These two methods give rise to distinct algorithms for the order dependent and independent distributed combiners.

Adding or subtracting to the training signal components which are orthogonal to an input signal will not affect the covariance of that input signal with the error signal. For orthogonal input signals the cross-covariance matrices $\underline{R}$ and $\underline{P}$ of the global and distributed combiners will have the same mean values and thus the same eigenvalues. Thus according to equation 3.17 they will both have the same ROC for a given stepsize.

Experimental work presented later in section 5.3.2 implies that the global combiner produces a better result when input signals are

non-orthogonal but that the distributed combiner tracks non-stationary signals faster. This interesting observation is noted but is not discussed further in this thesis. It may form the basis of future studies.

3.2.3 Optimum Stepsize

Equation 3.17 implies that ROC is dependent upon the stepsize and the various eigenvalues of the cross-covariance matrix. However, as the distributed combiner is used with orthogonal input signals, it has a diagonal input cross-covariance matrix where each of the items on the diagonal are power estimates of the respective input signals which are equal in value to the eigenvalues. Thus the convergence of each tap is dependent solely on its input power and the stepsize used. Setting the tap stepsize individually according to the power estimates, equation 3.20, will ensure that all taps converge at the same fast rate. Section 3.2.4 will show that it is not always desirable to select the maximum ROC. Nevertheless if the stepsize is made inversely proportional to the power estimates, all taps will converge at the same relative rate.

In the rare case where the mean power contents of the input signals are known in advance, the stepsize may be fixed. If not, the stepsize-eigenvalue product must be optimised. This may be done either by applying an AGC to the input signal and using a constant stepsize or by varying the stepsize according to the input power estimates.

Griffiths [14] proposed an alpha filter for achieving the second approach. This measures the mean input power by squaring each sample and averaging the result. The alpha filter uses exponentially weighted averaging in a simple algorithm which is suited to recursive estimation in stationary environments.

$$\lambda(n,t+1) = \lambda(n,t).(1-\alpha) + \alpha.s^2(n,t) \qquad (3.27)$$

$$\mu(n,t+1) = \beta/\lambda(n,t+1) \qquad (3.28)$$

The constant α has a small value $1 < \alpha < 0$, which controls the decay time of the memory which averages over approximately $1/\alpha$ samples. The constant β sets the value of μ for a certain pre-determined, ROC. A mathematically identical version used in the simulations of chapter 5 is:

$$C(n,t+1) = C(n,t).(1-\alpha) + \alpha.\beta.s^2(n,t) \quad (3.29)$$

$$\mu(n,t+1) = 1/C(n,t) \quad (3.30)$$

$$\text{where } C(n,t) = \beta\lambda(n,t) \quad (3.31)$$

Satorius [38] made β equal to the reciprocal of α, enabling their product to be dropped from the right-hand term of equation 3.29. Should a step-function occur in the input power (e.g. at switch-on), then the algorithm will track it exponentially, with a time-constant proportional to $1/(1 - \alpha)$.

This stepsize recursion may equally well be applied to the individual taps of global combiner applications, where it will ensure a reasonably high and stable ROC. Should the input signals be orthogonal, the stepsize recursion may be used with the individual taps of either global or distributed combiner algorithms. Having optimally set the stepsize per tap, global and distributed algorithms will have the same ROC.

3.2.4 Algorithm Noise in the Distributed Combiner

Given the orthogonal conditions necessary for successful use of the distributed linear combiner algorithm , ROC is just the same as for a global algorithm. This is not the case for algorithm noise, in which the distributed algorithm's performance is decidedly inferior. To show this it is necessary to introduce the equation for converged algorithm noise [7].

$$E\langle e^2\rangle = 2.e^2(opt) / (2 - \mu.R(0).N) \quad (3.33)$$

This gives the error power output from an adaptive processor in

terms of the theoretical optimum (calculated according to equation 3.8), R(0) the input power to the taps, N the number of taps updated by the algorithm, and μ the stepsize. The additional error power is caused by random fluctuations in the tap values caused by the update algorithm itself, and is called algorithm noise.

For the distributed algorithm equation 3.33 has serious consequences. There will certainly be tap signals which have a low correlation with the training signal used by the algorithm. In these cases the optimum error power is only slightly less than the power of the training signal used. Since the excess error is proportional to the optimum error, a high optimum error means a high algorithm noise.

Equation 3.33 was used to calculate the algorithm noise of the global and distributed combiners in two examples. In both, eight taps were fed with orthogonal signals which ideally would give an error power 30dB below the training signal power level. In one case this optimum would be given by an equal combination of all signals (rare in channel equalisation) and in the other the only signal contributing would be that on the fifth tap (representing a non-distorting channel). In both cases a stepsize equal to the reciprocal of C, where C is eight times the tap input power, was selected. This stepsize has been reported to be the optimum for the global algorithm [7].

The results presented in table 3A showed the global algorithm to be superior: The total algorithm noise in the global algorithm was equal to the optimum error, in both examples. After convergence the total error power was calculated at -27 db. The order independent algorithm in both cases produced an error power of -3.3 dB. The error was greater than that of the global algorithm for the reasons mentioned above.

The order dependent algorithm was the only one in which the order of the taps affected the algorithm noise. When all taps carried equally correlated signals, the error power was at -4.8 dB. When the fifth tap carried the well-correlated signal, the error power was -5.56 dB. The best result was obtained when the correlated signal was

on the first tap, giving an error of -27.8dB, 0.8 dB better than for the global algorithm. If the correlated signal was switched to the second tap, however, the error signal immediately deteriorated to -9.7dB. This shows that the best results are obtained when the earlier taps carry signals highly correlated with the training signal.

In conclusion, the algorithm noise of the distributed linear combiner is in general worse than that of the global combiner. Of the two distributed algorithms, the order-dependent one produces superior results. The order-dependent algorithm produces best results if the most correlated tap signal can be arranged to be on earlier taps. In certain untypical cases the order-dependent distributed algorithm can even produce a lower algorithm noise than the global algorithm.

3.3 Properties of the Transveral Adaptive Filter

The output of a one-dimensional channel carrying a signal may be split up into many signals which are linearly recombined to form a filtered signal. The simplest method is to use a shift register to form a transversal filter. All filtering, analogue or digital, results in a weighted combination of time-delayed versions of the input being presented at the output. The autocovariance function of the impulse response of the channel will set a level of intersymbol interference (ISI). This in turn will cause non-orthogonality in the tap signals, so a global algorithm is required.

The taps of the combiner may be fed from more than one shift register. Furthermore, more combiners may generate different signals from the same inputs. In SONAR applications many transducers feed separate shift registers, all of which feed the taps of a single global combiner [39]. In two-dimensional filtering, used with complex channels, the same principle is applied to two shift registers, one fed from the I-Channel and the other from the Q.

With a shift register the cross-covariance matrix $\underline{R}$ becomes the autocovariance matrix of the channel, a Toeplitz matrix. Variations

in the signal power estimates on the main diagonal caused variations between the eigenvalues in the case of the distributed combiner, but here the powers of the tap signals are all the same. Instead, eigenvalue variations are caused by the intersymbol interference causing non-zero off-diagonal terms in the matrix. Were there no ISI, the EVR would be unity. Since the power supplied to each tap is the same, the algorithm uses a global stepsize also.

3.3.1 The Eigenvalues of a Toeplitz Matrix

The ROC of the most popular form of adaptive filter is dependent on the eigenvalues of the channel autocovariance matrix. It is therefore disappointing that the literature on the eigenvalues of Toeplitz matrices is largely inaccessable to the engineer. The most readable and informative work seems to be by Widom [40].

The average of the eigenvalues is equal to the channel power estimate. Thus the power estimate is bounded by the maximum and minimum eigenvalues. Should the eigenvalue ratio be unity, then all will be equal to the power estimate. There will be as many eigenvectors as the rank of matrix, i.e. the number of taps in the filter. Eigenvalues may coincide, however.

The eigenvalues all have values corresponding to points on the power spectrum of the signal. i.e. they are bounded by the maximum and minimum values of the power spectrum and may never be negative. They appear (at the current level of mathematical understanding) to be unpredictably, almost stochastically distributed over the spectrum. Nevertheless, EVR rises with rank in a non-linear manner and for a matrix of infinite rank, the spectrum will be completely and continuously covered with eigenvalues, albeit with uneven densities.

The existence of an eigenvalue of zero will mean that the autocovariance matrix is, strictly speaking, non-invertable. In fact, this leads to a family of solutions to the Wiener-Hopf equation, any one of which will give an optimally low error power.

Figure 3.3 shows that as the adaptive recursion advances the major contribution to the error signal comes from an eigenvalue close to a specific value. This value reduces as the number of iterations rise. If a finite number of iterations is allowed for convergence, then at the end of that time the major source of misadjustment and error will come from an eigenvalue of a specific value, the value depending on the number of iterations. The problem posed by a zero eigenvalue, then, lies not in its own value, but the implication that the channel eigenvalues range down to zero. It is therefore probable that after the number of iterations needed for convergence there also exists an eigenvalue whose value is close to that for maximum disruption at that iteration number.

3.4 The Chang Equaliser Family

Section 3.3 explained that a one-dimensional signal must be delayed before filtering using a combiner structure, and that all filters delayed a signal in some way. This leads to the concept of the Chang general equaliser [8], figure 1.6. Here, the input signal feeds filters of arbitary but differing characteristics, which in turn feed the taps of the combiner. The combiner then sums the signals in an optimal manner.

The transveral filter is only one member of the Chang family. However, a good reason would be needed before abandoning its low cost and simplicity in favour of a more complicated version. The most usual reason is that the signal samples which are provided by the simple structure are correlated with each other i.e. they are not orthogonal. This causes a wide spread in the eigenvalues of the input signal's cross-covariance matrix, which slows the rate of covergence of the adaptive algorithm.

Many ways of producing orthogonal signals have been suggested. They may be approximate, such as Fourier Transform [41] or Walsh Transforms [42]. Alternatively they may be exact such as eigenvector decomposition [8] or prediction-error filters. Exact orthogonalisation techniques may not use transforms with fixed coefficients, but require a knowledge of the input signal's

autocovariance function. When orthogonal signals are supplied to a linear combiner, its cross-covariance matrix becomes diagonal and the adaptive stepsize algorithm (cf section 3.2.3) may be applied to ensure a reliable ROC.

If it is certain that orthogonalisation has been exact, then either combiner algorithm may be used. If orthogonalisation has been approximate, then the global combiner will produce the best results. Distributed algorithms may certainly be used in the latter case [43], but they will give a sub-optimal solution with greater converged error.

3.5 Adaptive Prediction-Error Filter Algorithms

3.5.1 The Transversal Structure

The signal flow diagram for the transversal PE structures are figure 2.2 for the forward predictor and figure 2.3 for the backward predictor. There are three main methods of obtaining coefficient values for the filter: Invert the autocovariance matrix, transform the lattice PARCOR coefficients or use the LMS algorithm directly.

The autocovariance function of an input signal may be estimated using correlators. These will give a short-term estimate of the stationary characteristics of the input signal, whose accuracy rises with the length of measurement time. The values of the function are then substituted into the Toeplitz autocovariance matrix, which is then inverted to satisfy the Wiener-Hopf equation (3.9). From equation 2.45

$$\underline{R}\underline{a}^T = Q(N).\underline{1}^T \qquad (3.34)$$

Where $\underline{a}^T$ are the PE filter coefficients including training signal, Q(N) is the power output of the filters and $\underline{1}$ is the matrix [1,0,0...]. Thus:

$$\underline{a} = Q(N).\underline{1}.\underline{R}^{-1} \qquad (3.35)$$

I.e. the coefficients of the PE filter are proportional to the first column of the inverted matrix. Since the first term of $\underline{a}$ is always unity, the terms of the column are normalised to make this so.

The second method is to transform the lattice PARCOR coefficients using equation 2.54. The lattice coefficients themselves may be derived adaptively or from the autocovariance function, using the recursion of table 2A. The latter is a particularly attractive solution, since the transversal PE filter coefficients are produced as a by-product of the Levinson-Durbin recursion. Indeed, it is more computationally efficient to use the recursion on the terms of an estimated autocovariance function than to invert the autocovariance matrix by a general method.

The third method is to map the PE filter onto an adaptive transversal structure and use the global LMS recursion. Thus the input signal which is summed into the output with unity gain is regarded as the training signal, and the output of the filter is the error used in the tap update recursion. There are two reservations about this method: As a transversal filter its ROC will be governed by the stationary statistics of the input signal, as shown in section 3.3. Further, since the filter output power is rarely small (cf 2.3.7), the large error signal will lead to a high algorithm noise level.

3.5.2 The Lattice Structure

The coefficients of the lattice may be calculated in an open-loop or closed-loop manner. The simplest and cheapest open-loop method is the covariance method [11]. This uses the estimated autocovariance function of a block of data to calculate the PARCOR coefficients as in Table 2A.

Another open-loop block processing method is to run the data through each stage of the lattice in turn. This makes use of the fact that the ideal value of the PARCOR coefficient is the normalised

covariance of the two input signals. Thus for the data arriving at a stage:

$$K_f(n) = \left(\sum_{i=o}^{m-n} f(n-1,t).b(n-1,t-1) \right) / \left(\sum_{i=o}^{m} f^2(n-1,t) \right) \quad (3.36)$$

$$K_b(n) = \left(\sum_{i=o}^{m-n} f(n-1,t).b(n-1,t-1) \right) / \left(\sum_{i=o}^{m-n} b^2(n-1,t-1) \right) \quad (3.37)$$

Where f and b are the forward and backward signals, M is the number of signal samples available in the data block and $K_f(n)$, $K_b(n)$ refer to the forward or backward PARCOR coefficients. The two PARCOR values may then be used separately in a two-valued PARCOR lattice, or in some combined way in a one-valued PARCOR lattice to obtain the f and b value for the next stage.

A problem presented by this method is that short-term fluctuations in the statistics of a sample make it a noisy inaccurate estimate of the statistics of a population. Thus the estimated value of a PARCOR coefficient may exceed its permitted bounds of ± 1. In LPC this will lead to instability of the recursive regenerating filter (c.f. section 8.1.1). The solution used by Makhoul [11] is to use a one-valued PARCOR lattice, where $K_f(n) = K_b(n)$. The forward and backward PARCOR coefficients are combined in a manner expressed generally as:

$$K(n) = S.(0.5(|K_f(n)|^r + |K_b(n)|^r))^{1/r} \quad (3.38)$$

Where K(n) is the combined value, S is the sign of one of the estimated values (the two signs will be identical) and r is some index. Burg [17] showed that the harmonic mean, r = -1 does satisfy an LMS criterion. Makhoul [11] showed that $r < 0$ guarantees that the average will be in bounds as long as both estimates are. Itakura [12] favoured the geometric mean $r \to 0$. While the latter does not satisfy any LMS criterion, it does ensure that the average will be unconditionally within bounds. The geometric mean method is

identical to the autocorrelation formula for stationary signals [20].

The closed-loop adaptive gradient algorithm uses the LMS recursion again. If the forward channel of the lattice is regarded as a one-tap linear combiner, where the output is the error signal, the Widrow recursion becomes:

$$K_f(n+1,t+1) = K_f(n+1,t) + \mu(n,t).b(n,t-1).f(n+1,t) \qquad (3.39)$$

$$K_b(n+1,t+1) = K_b(n+1,t) + \mu(n,t).f(n,t).b(n+1,t) \qquad (3.40)$$

Makhoul [11] and Mead [13] averaged the two to reduce algorithm noise:

$$K(n+1,t+1) = K(n+1,t) + \mu(n,t)\ (b(n,t-1).f(n+1,t) + f(n,t).b(n+1,t)) \qquad (3.41)$$

Equation 3.39 would be suitable for a two-valued PARCOR lattice, tracking a non-stationary channel with optimum speed. Equation 3.41 is more suitable for equalisation, where non-stationarity need not be tracked as rapidly and coefficient accuracy is of greater importance.

Makhoul [23] proposed a formula for recursively estimating lattice PARCOR coefficients which does not map onto the LMS recursion. Instead it uses alpha filters, similar to equation 3.22, to calculate both covariance and normalising power estimates:

$$P(n,t+1) = P(n,t).(1 - \alpha) + \alpha.f(n,t).b(n,t-1) \qquad (3.42)$$

$$Q(n,t+1) = Q(n,t).(1 - \alpha) + \alpha.f^2(n,t) \qquad (3.43)$$

$$K(n+1,t+1) = P(n,t+1)/Q(n,t+1) \qquad (3.44)$$

It may be readily seen that this is similar to the non-recursive method of equation 3.37. In common with the technique of equation 3.37, the power estimate, Q, may be derived from the f-channel signal, the b-channel signal or a combination of the two. Used with

a varying averaging coefficient, α, this recursion formula is used in the exact least square lattice of [47].

The equations of 3.42 - 3.44, although they are in recursive form, are actually open-loop and therefore non-gradient types. It will be seen that there is no feedback from the output of the stage to correct any possible errors in coefficient calculation. For this reason also, the accuracy of the PARCOR error power is independent of the output signal power.

3.5.2.1 The Exact Least Squares Lattice

The exact least squares lattice algorithm is an open-loop type. Since the lattice equaliser structure may be regarded as a set of interconnected Wiener filters, the algorithm calculates the coefficient values for their individual taps. Since each filter involves only one tap, the value is a cross-covariance divided by the tap input power.

The recursion starts by estimating the signal input power, using a recusion similar to that of equations 3.42 - 3.44 to maintain an exponentially weighted estimate. The mean power at each point in the circuit is then calculated recursively using a formula similar to that of equation 2.55. The forward and backward powers are normally computed separately, as it is not assumed that the signal is stationary. The signals passing through the structure are then used to give instantaneous cross-covariance estimates, which are exponentially averaged to give mean values. Finally the filter coefficient values are calculated in a manner similar to that of equations3.42 - 3.44 dividing the covariance by the power to give a Wiener coefficient value.

The advantage of the technique is that it permits faster convergence and tracking with a far lower algorithm noise. Satorius [47] has reported that this so-called "exact" algorithm converges 2-3 times faster than a conventional gradient type.

Unfortunately there is a need for a high calculation accuracy in

"exact" algorithms. The division needed in gradient algorithms is merely to calculate the optimum rate of convergence; a moderate error will have little impact on rate of convergence of algorithm noise. The division in the "exact" algorithm, however, determines the coefficient values and must be done to a far higher precision. It is this precise division which restricts the algorithm's usefulness at the current state of technology. For this reason the "exact" algorithm has been relegated to the ranks of ideas whose time has not yet come. This thesis will meanwhile concentrate on gradient techniques.

3.5.3 Hybrid Forms

The lattice PE filter is an ideal orthogonalising network to ensure the fast convergence of the taps of a linear combiner. Unfortunately it is only suitable for one- or two-dimensional channels. With multi-dimensional channels of high orders the number of computations rises dramatically. For this reason it is not directly applicable to spatial arrays, such as in RADAR, SONAR or beam-steering. The traditional solution in the latter applications has been to tolerate the eigenvalue disparity and resulting slowed ROC of using a combiner directly, or to use other orthogonalisation techniques [33]. An alternative lies in orthogonalisation using a network of PE filters in transversal form, as in figure 2.5. These will certainly orthogonalise array signals, but their own ROCs will be limited by the input signal statistics.

A better version is that of figure 3.5. It, too is fed from a delay line, making it suitable for array implementation. However, instead of the one global algorithm usually used with delay lines, a number of one-tap distributed algorithms are used instead. In each algorithm a multiplier feeds a signal to the negative input of a subtraction element. The output of that subtraction element is the error signal for updating the coefficient of the multiplier using the LMS recursion. A normalised ROC for each coefficient is provided by operating the stepsize update recursion (equation 3.27) on the input to the multiplier.

The structure is more economical than it appears on paper. If a delay line with accurate delays or an evenly-spaced array are used, only one recusion per coefficient is needed. Thus only one recusion will update all multiplier coefficients marked "K(1)". If the delays are inaccurate or the array elements irregularly spaced, then each multiplier must have its own update recursion in order to maintain orthogonality.

The principle behind the structure lies in equation 2.54, which shows that higher-order PE filters are formed by combining delayed and time-reversed versions of lower-order ones. This method may be used to build up orthogonalising structures of any order. Being essentially a transversal structure, it is suitable for reliably converging orthogonalising structures using delay elements not normally useful for lattices, such as CCD or SAW delay lines

3.6 The Lattice Equaliser

The lattice equaliser, figure 1.7, is simply another Chang equalising structure. Here, the lattice structure acts as the orthogonalising network, feeding a conventional linear combiner. The orthogonality condition stems from the fact that since the error output of a converged linear combiner is orthogonal to the input signal (cf section 3.2.1.1) and since lower order error signals are formed of linear combinations of the input signal (figure 2.5), then each error output signal is orthogonal to lower order error output signals. Thus the error signals are mutually orthogonal. The tapweight update recursion normally [38] uses the order-dependent distributed combiner algorithm of equation 3.26, and figure 3.4, although the global combiner structure will also work.

The values of the equaliser coefficients may be calculated open-loop from estimates of the channel autocovariance function and its cross-covariance with the training signal, perhaps using the hardware of figure 3.6. The autocovariance function is first used in the Levinson-Durbin recursion of table 2A to give the lattice PARCOR coefficient values K(n), the Prediction-Error power Q(n), and the impulse response of each stage of the lattice a(i,n). Since the

sidetap signals are orthogonal, the tap coefficient values may be calculated using equation 3.8, which may be re-expressed here as:

$$G(0) = P(0)/Q(0) \tag{3.46}$$

$$G(n) = \left(\sum_{i=0}^{n} P(i).a(i,n) \right)/Q(n) \tag{3.47}$$

Where P(i) is the ith term of the cross-covariance function. Now the impulse response of one tap's output is the impulse response of its lattice stage, times the tap value. The impulse response of the whole equaliser, then, is the sum of all of these:

$$h(i) = \sum_{n=0}^{N} G(n).a(i,n) \tag{3.48}$$

Where h(i) is the ith term of the impulse response. The method of solving the Wiener-Hopf equation recursively using the equations of table 2A and 3.46 - 3.48 is known as the Levinson-Trench algorithm and was first proposed for channel equalisation by Butler [17].

It will be noted that the impulse response of a lattice stage has a length one greater than its order. This leads to a method of calculating the tapweight values of a transversal filter of optimum length [49], using the following recursion:

$$e^2(0, opt) = E < d^2(t) > \tag{3.49}$$

$$e^2(n+1,opt) = e^2(n,opt) - Q(n).G^2(n) \tag{3.50}$$

Where the optimum Wiener error of a transveral filter of length n is $e^2(n+1, opt)$. Used with the Levinson-Trench algorithm, equation 3.50 allows the recursion to be stopped when the error signal has dropped below a pre-defined threshold. This allows the number of taps, and their truncation error contribution, to be minimised.

3.6.1 Open-Loop Equaliser Simulation Results

The Levinson-Trench algorithm and its error calculation recursion were simulated to test their suitability in equaliser construction.

A real channel of eigenvalue ratio 11 with no additive noise was selected for the trial. This is channel R11 in appendix D.

The graph of figure 3.7 shows the results of the simulation. It is a graph of error power versus iteration number. The solid line shows, for comparison purposes, the convergence of an 11-tap adaptive LMS transversal filter starting from zero tapweight values using an optimum stepsize calculated according to equation 5.6. Its signal is a 2^{24}-1 length PN code.

The dotted line shows the convergence of the optimum length equaliser using 11 taps and 63 bit PN sequence, suggested by Yücel [48]. This cyclic equalisation technique eliminates the effects of short-term statistical fluctuations, ensuring an exact estimate every 63 iterations. Thus convergence to almost -60 dB may be said to have taken only 63 iterations.

The hatched line shows how the length-optimising recursion would work. After the 63 iterations have collected accurate estimates of line statistics, data collection is stopped. The threshold for equaliser error was set at -50 dB (an unusually low power for equaliser operation). The Levinson-Trench algorithm was then operated along with equation 3.50 to find the minimum equaliser length necessary to obtain this. The optimum length was found to be 10 taps, giving an error of -51 dB.

This single simulation shows that the technique works under rather ideal circumstances. Further topics for more detailed investigation include the effect of noise on the system and the necessary precision of the processor performing the algorithm.

3.7 Summary

This chapter has shown how the lattice equaliser can be considered simply as one member of a whole family of adaptive structures. The main emphasis has been placed on a thorough analysis of the properties of adaptive linear combiners. The LMS algorithm which is used to update combiner coefficient values in most gradient

adaptive applications has been explained, along with its advantages and limitations. The distributed combiner has been shown to use a version of the same LMS gradient algorithm and to have further advantages and limitations, the main limitation being increased algorithm noise.

The properties of the transversal adaptive filter, the simplest and most popular adaptive filter form, have been explained. Its limitations, such as dependancy of ROC on channel frequency response, and thus on the ISI which an equaliser is to remove, have been shown. The transversal filter is presented as a yardstick against which to compare lattice equaliser performance.

Lastly the lattice equaliser, has been presented as one of the Chang family of equalisers to which the transversal filter also belongs. The orthogonalising properties of the lattice have been shown to overcome some of the problems inherent in the transversal structure. The equivalence of lattice and transversal structures has been established, and an example has shown that the former has many other applications in addition to gradient adaptive filter operation.

Algorithm Type	All Signals Equally Correlated	One Highly Correlated Signal on: tap 1	tap 2	tap 5
Global	- 27	- 27	- 27	- 27
Order Dependent	- 4.8	- 27.8	- 9.7	- 5.56
Order Independent	- 3.3	- 3.3	- 3.3	- 3.3

Figures given in dB

Table 3A Error Signal Attenuation for Various Linear Combiner Models

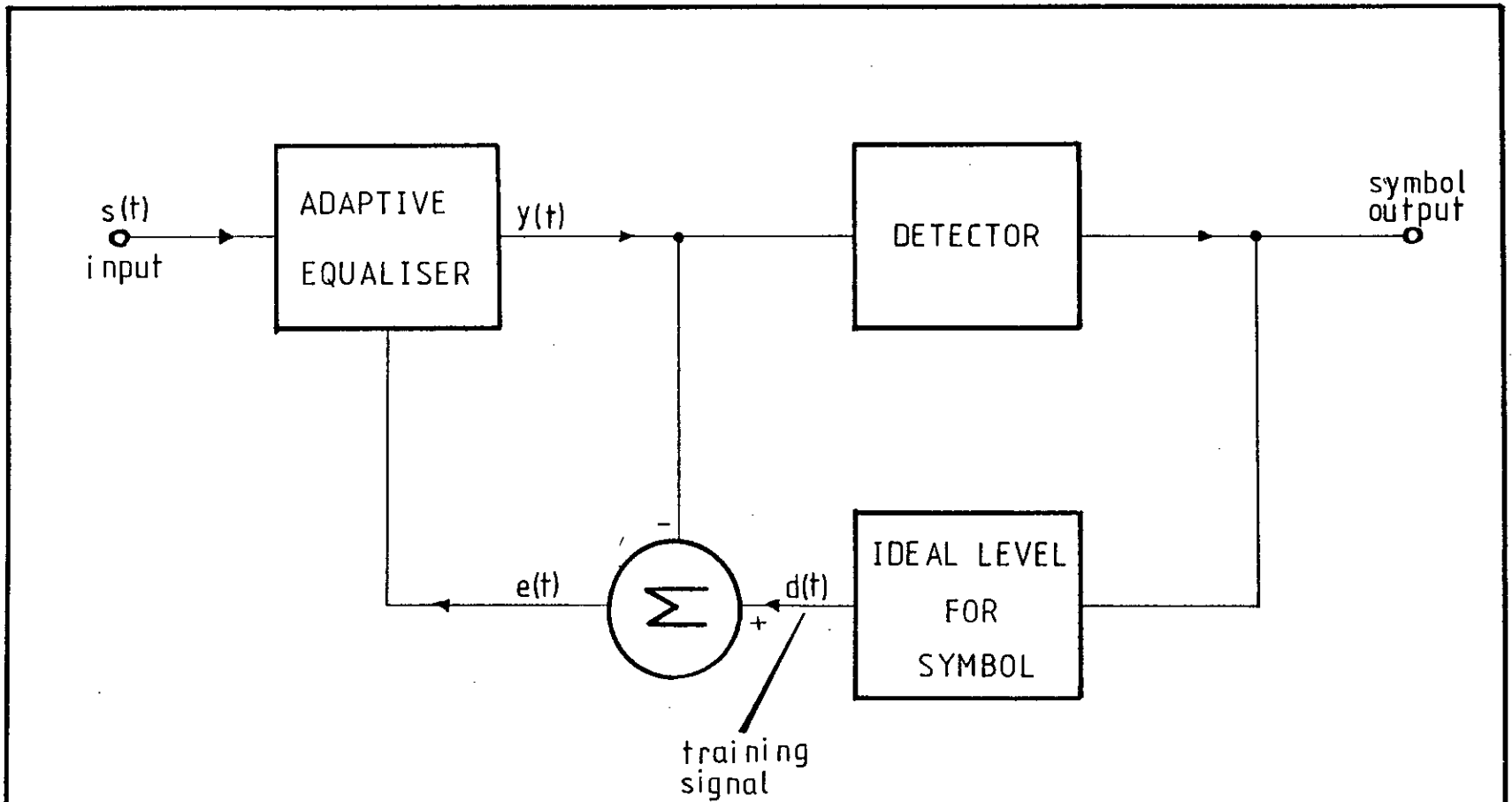


Fig 3.1 The generation of a training signal using data directed feedback

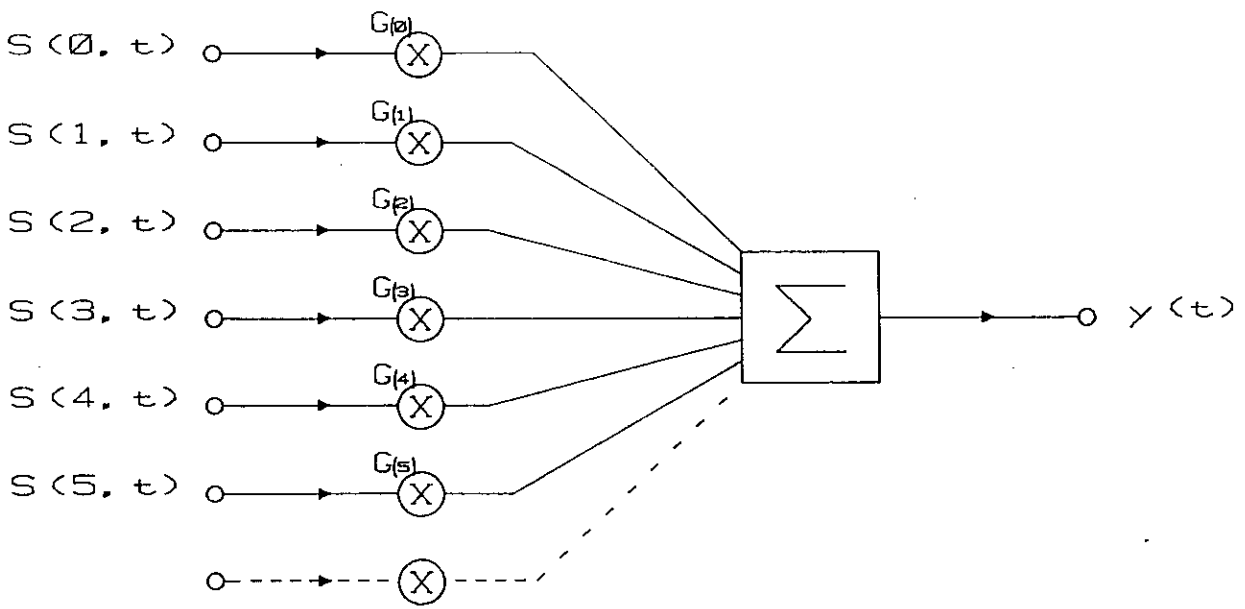


Fig 3.2 A linear combiner

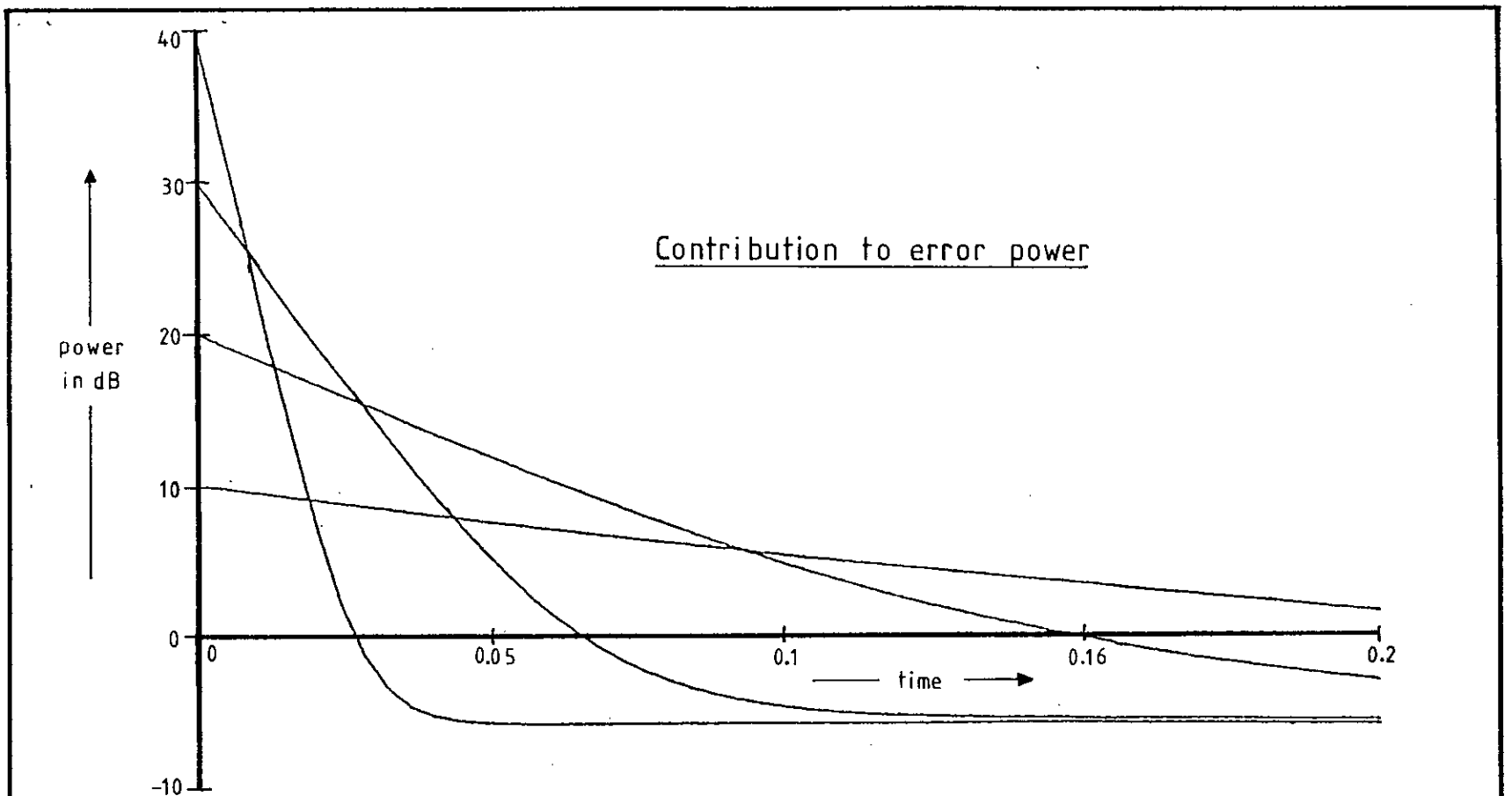


Fig 3.3 Relationship between rate of convergence and eigenvalue. The higher eigenvalues converge more rapidly than the lower ones.

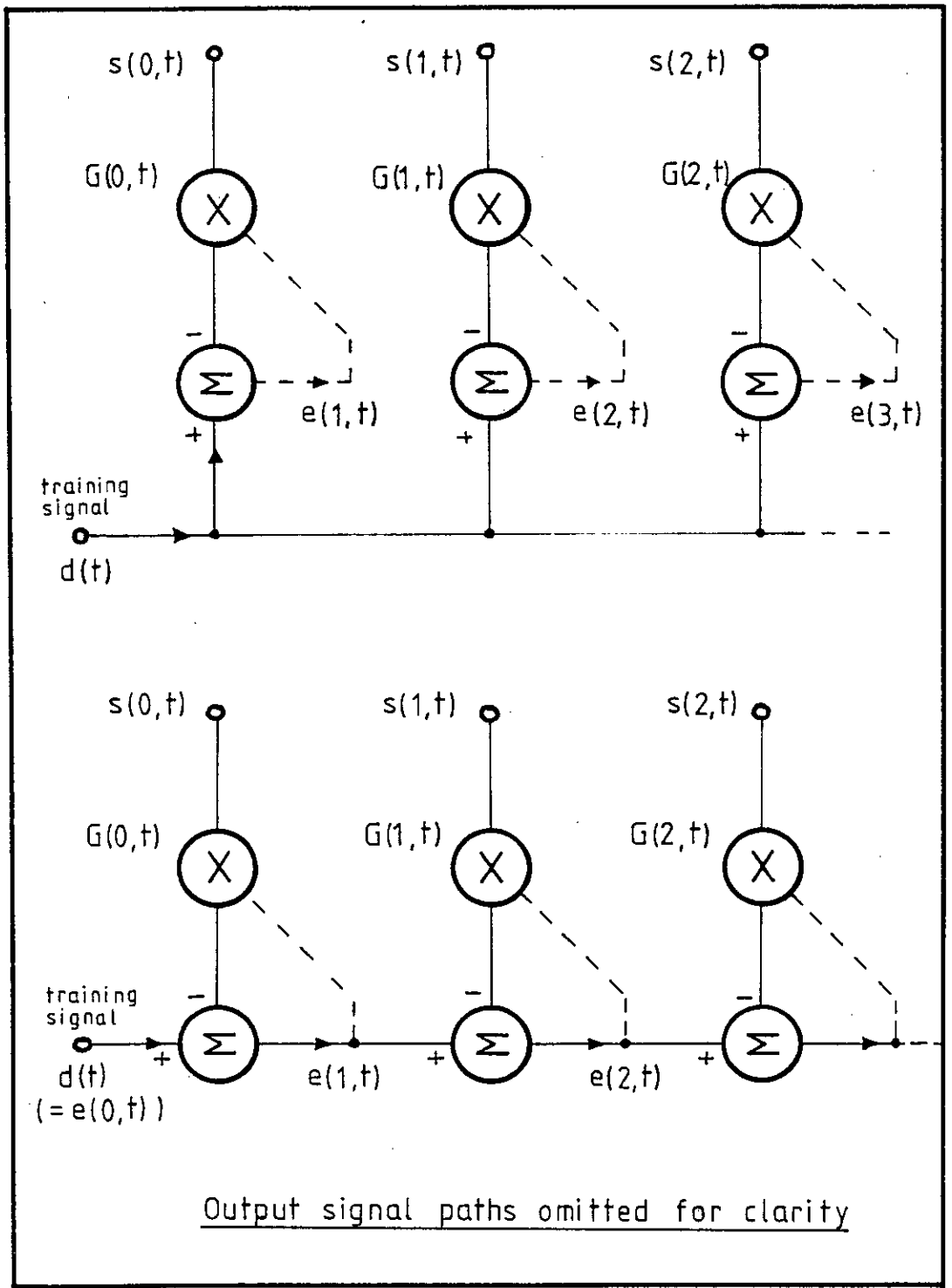


Fig 3.4 Order independent (upper) and order dependent (lower) error signal derivations

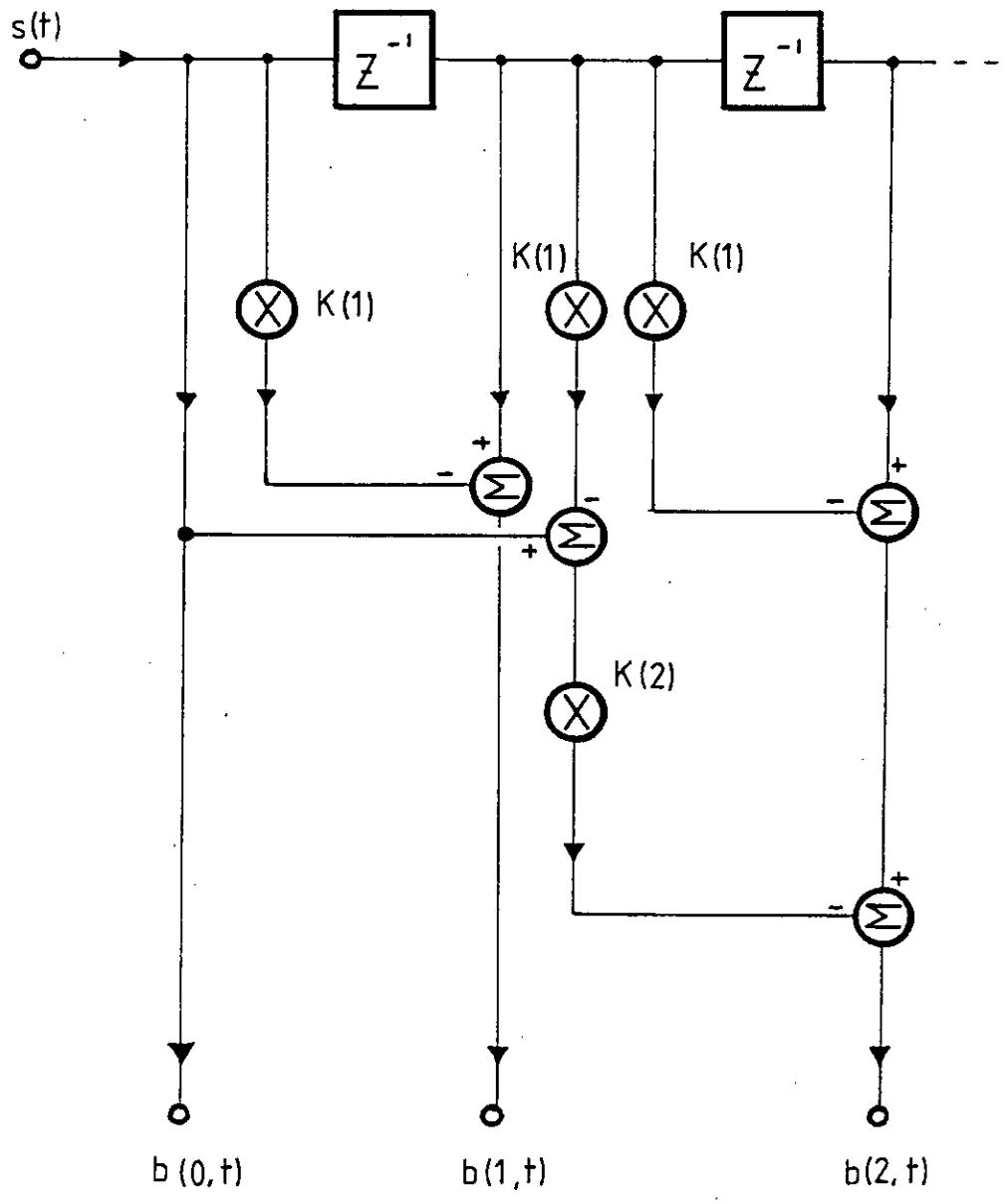


Fig 3.5 A hybrid lattice

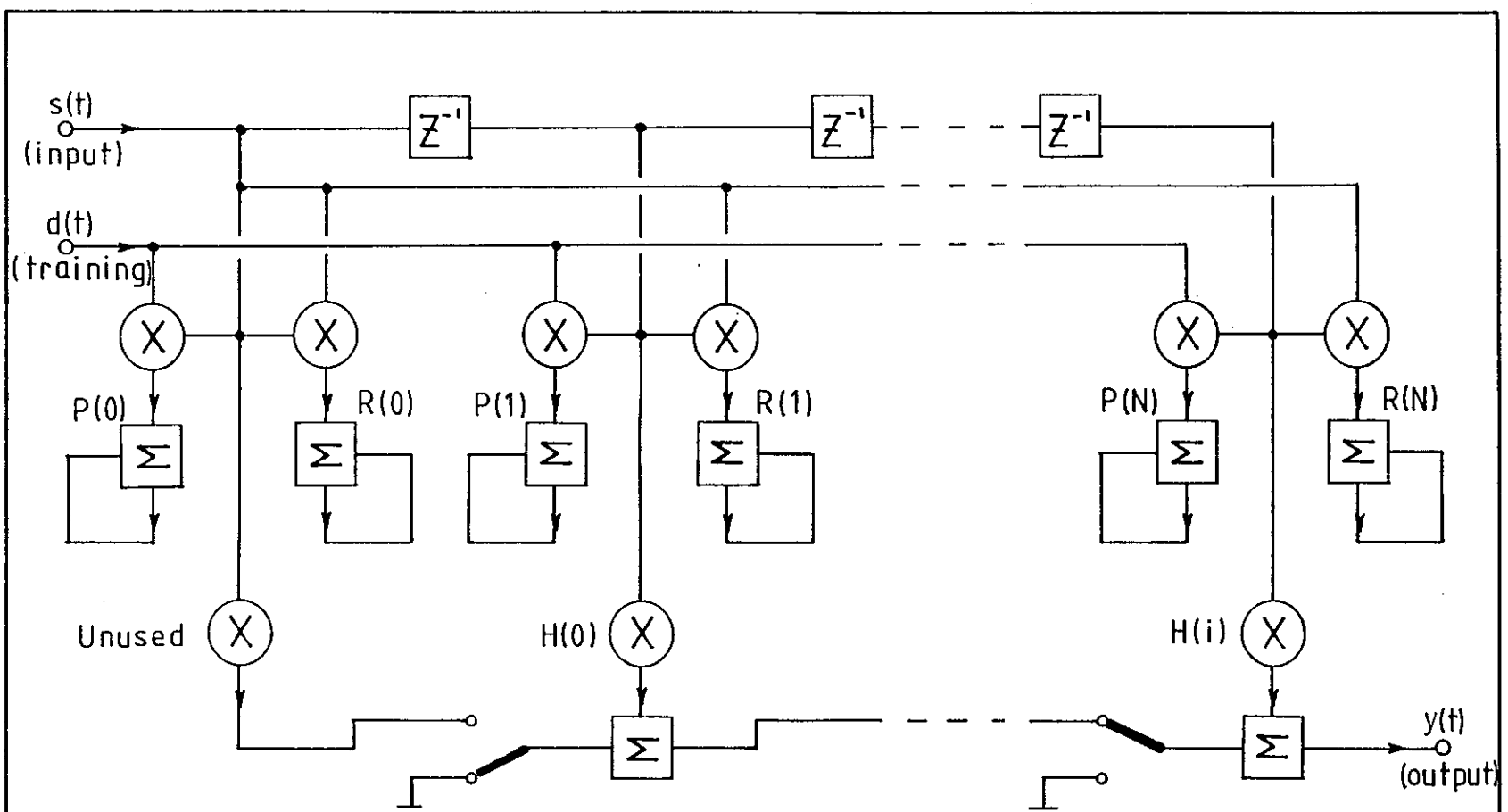
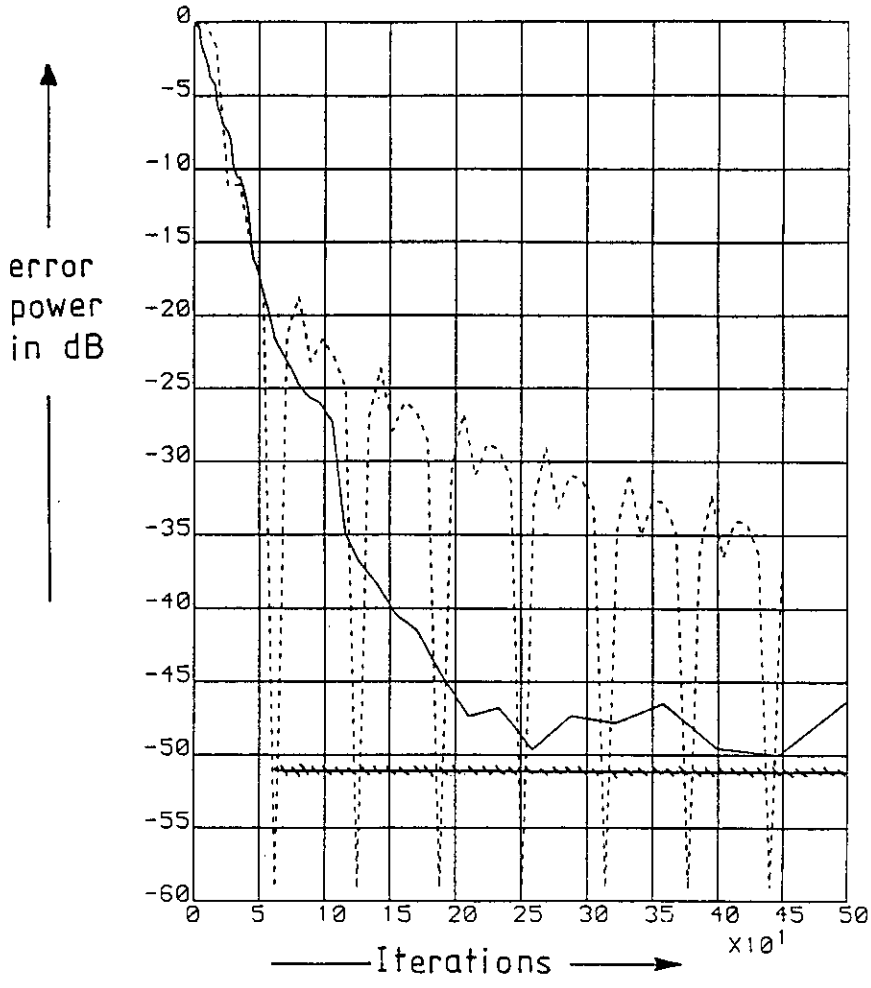


Fig 3.6 Equaliser hardware (processor not shown)



Channel: R11 (cf appendix D)
Solid line: Transversal equaliser
Dashed line: Open-loop equaliser
Hatched line: Open-loop equaliser frozen at 63 iterations with 10 taps

Fig 3.7 Open-loop equaliser simulation results

AN INVESTIGATION OF OPTIMUM EQUALISER DESIGN FOR DATA CHANNELS

A thesis on equalisation would not be complete without a discussion of the line impairments which make equalisation necessary. This chapter starts from the physical devices whose imperfections cause intersymbol interference in data transmission. The nature of the impairments on the data signals themselves are next discussed, also the way in which a poor channel may corrupt data.

Section one discusses the physical imperfections which prevent a line from having a unit impulse response. Section two describes a survey done in the British Isles on the numerical parameters of the impairments likely to be encountered. Section three describes mathematically the concepts of the eye diagram and the error power, and relates them to the system impulse response. Section four describes the multidrop environment, an application in which rate of equaliser convergence is critical. Section five is a description of the synthetic raised-cosine channels, used later in simulations. Section six presents an analysis of the parameters of certain typical modem channels while section seven is a summary of the chapter.

4.1 The Nature of Telephone Channel Impairments

At first glance a telephone channel appears to be simply a twisted pair of wires, with very little signal processing action. It came as a surprise to early pioneers that a telephone channel does filter a signal, sometimes to the extent of making it unintelligible [37]. This is the result of the sum of the effects of the many small distortions encountered along the long signal path.

The line itself has a filtering action, delaying and attenuating certain frequency components. The distributed parallel capacitance and series inductance of an ordinary uncompensated cable acts as a low-pass filter, causing the attenuation of high frequencies. This is normally tolerated over local lines, but trunk lines have loading coils added to reduce the effect. The result of the coils is to

sacrifice phase-frequency characteristics for a flat amplitude-frequency response. The human ear is more sensitive to amplitude than phase distortion, but the latter causes ISI in modem traffic. Other discontinuities, such as feeding coils, blocking capacitors, changes in cable impedance and hybrids also contribute to the filtering effect.

At certain points on the line, major discontinuities may cause echo. Strictly speaking, echo is also a filtering effect, since all linear filtering is a process of delaying and adding. Nevertheless, some major discontinuities, typically hybrids, cause a signal to echo back and forth along a line, increasing the ISI.

The intersymbol interference caused by the filtering effect may be compounded by phase or amplitude discontinuities. Sometimes called phase or amplitude "hits", these are sudden changes in line characteristics caused by components being suddenly switched into or out of the circuit. The phenomenon usually causes a sudden increase in ISI as the line alters the characteristics to which the equaliser was originally tuned. Many equalisers operate in tracking mode after initial convergence and can adjust to such line changes.

Frequency offset is introducted when a channel is frequency-division multiplexed to another frequency band and then restored later to its original frequency. Inaccuracies in the heterodyne oscillator frequency will then cause a slight frequency shift to be introduced in the channel. In a modem this effect is normally compensated by its carrier-locking circuitry. In the British Isles, the shift is rarely greater than 0.5 Hz [51].

Non-linear distortion of the amplitude of the signal is introduced by non-linear circuit elements along the signal path. These are sometimes semiconducting contacts in mechanical switching assemblies, caused by dirt or oxidation. Another cause is non-linearity in amplifying elements.

Additive noise is introduced through a number of effects. Cross-talk between channels will superimpose an unwanted signal on the

wanted one. Thermal effects in amplifiers and other components will introduce Gaussian noise. Switching transients and impulses from electric storms may be coupled into the line to cause impulsive noise. Since impulsive noise bursts almost invariably cause a detection error in data transmission, it is not normal to adjust equaliser coefficients to allow for them. The Gaussian noise level, however, affects the optimum values of the equaliser coefficients (cf section 2.2.3).

4.2 The BTRL Survey

In 1967 the then Post Office Research Centre (PORC) conducted a survey into the properties of the line in the UK national public switched telephone network (PSTN). A representative cross-section of 536 lines were selected and their transmission characteristics were measured in both directions [51]. The measurements were of amplitude-frequency characteristics, amplitude-group delay and of any major echoes on the line. Figure 4.1 shows a typical test result, the amplitude and group delay of a line. The result of the survey led to an understanding of the typical and extreme conditions likely to be encountered over dial-up lines in Great Britain. The data was then analysed with a view to constructing a line simulator for laboratory use [50].

It was broadly noted that the unloaded local cable used in the connection from exchange to subscriber mainly produced pure amplitude-frequency distortion. The hybrid transformer at the exchange, which converted the two-wire local connection into a four-wire trunk connection, was often imperfectly matched, leading to ripples in the amplitude-frequency characteristic and in severe cases to echoes. The trunk cable between exchanges is fitted with loading coils, distributed along its length. This has the effect of improving the frequency-amplitude characteristic of the line at the expense of the group delay properties. Minor faults and mismatches in the loading coils also caused transmission impairments. Line-noise, non-linearity and frequency offset were also measured.

After the data was compiled, it was analysed statistically [2] so

that the probability of a certain parameter value was known. These were mean values of parameters taken from all line data. Since the cross-covariance between parameter values was not measured it was not possible to reconstruct the response of a typical line. Figure 4.2 illustrates this point; the smooth curves of parameter mean values bear no information on the irregularities of the graph of figure 4.1.

The next step was to build a simulator which would distort an input signal in the same way as a particular type of line [50]. For this work the means of the parameter values were used. Thus in simulating a typical line consisting of 15 km of local line in series with 50 km of trunk line, the frequency response, group delay and noise would be the mean values for lines of those dimensions and not the uneven curves of figure 4.1. The simulator also had facilities for adding echo, non-linearity and other typical line impairments.

Later a computer model was produced so that line impairments could be used in computer simulations. Again, line models were produced which fitted the analysis of the data, rather than resembling the data itself. For various modem specifications, it was also possible to add the effect of the transmitter and receiver filters. This gave the characteristics of the analogue part of a data transmission channel from modulator output to demodulator input, excluding the effect of equaliser circuits.

The passband of the PSTN does not extend to DC, so it is normal to modulate data onto a carrier centred approximately in the middle of the line passband. The orthogonality of sine and cosine functions enables two independant data channels to be used, I and Q, to double the channel throughput. Transmission impairments then cause data contamination, not merely with ISI from adjacent bits in the same channel, but also with data from the second channel.

Instead of modelling the channel from modulator output to demodulator input it is also possible to model the channel at baseband, i.e. from modulator input to demodulator output (excluding the effect of any equaliser circuits). This model in the time domain is known as an equivalent baseband impulse response (figure 4.3).

When in other chapters the channel impulse response is discussed, it refers in data modems to this equivalent baseband impulse response. The simulations used in chapter five and the channels analysed in this chapter use equivalent baseband impulse responses supplied by British Telecommunications Research Laboratories (BTRL, formerly PORC).

4.3 The Effect of Channel Impairments on Data Equalisation

Of the channel impairments discussed in section 4.1, only some affect data equalisation. Frequency shift, for example, is relevant to modem design, but not to equalisation. The disruptive effect of impulsive noise was discussed, but this does not affect equaliser setting either. The effects of filtering and noise are discussed here, first in terms of eye diagram, then in terms of the equaliser error power.

4.3.1 The Eye Diagram

The concept of the eye diagram arises from the oscilloscope trace of the data signal before detection (figure 4.4). The oscilloscope will synchronise the signal, a near-random sequence of ones and zeroes, until the symbols of successive sweeps lie on top of one another. If two adjacent time-slots are displayed they will appear as two adjacent openings, bounded on the top by 'one' symbols, say, and on the bottom by zeroes. The area in the middle will normally be left open and the area between the openings will be bounded by the cross-overs between adjacent ones and zeroes. The effect is similar to that of a pair of eyes.

The filtering effect of a channel adds intersymbol interference (ISI) which equalisation can minimise but seldom remove. This has the effect of broadening the outline of the eye and making it indistinct (Figure 4.4b). The ISI has caused small amounts to be added or subtracted from the binary signal to introduce uncertainty.

A binary signal is normally detected by thresholding. Thus no errors will occur as long as the uncertainty due to ISI does not move

the value of the signal to the wrong side of the threshold. If no errors ocur due to ISI, the eye is said to be open. Should the eye be closed, there will be no area around the threshold into which the signal does not enter, and hence detection cannot be made without introducing errors.

The region in which a signal may lie, a one for example, is bounded on its upper and lower sides by the sum of all ISI contributions, should they all act in unison. This enables the openness of the eye to be calculated from the system impulse response and expressed as a percentage of the fully open eye. Let H(0 ... N) be the system impulse response, i.e. the impulse response from modulator input to demodulator output (this time including the effect of equalisers). For bipolar quadrature binary data the data output to the detector will come from the real part of the main lobe, Re(H(m)), say. All other contribution will be intersymbol interference. The interference will come from the same channel as the main lobe via the real part of the system impulse response and from the channel in quadrature with the main lobe via the imaginary part of the system impulse response. I.e for quadrature phase-shift keying with bipolar binary signals on either channel, the maximum value of ISI is:

$$\text{ISI (max)} = \sum_{i=0}^{N} |\text{Im}(H(i)| + \sum_{i=0}^{N} |\text{Re}(H(i)| - |\text{Re}(H(m))| \quad (4.1)$$

$$\text{Openness} = (|\text{Re}(H(m))| - \text{ISI (max)}).\ 100\% \ /\ |\text{Re}(H(m))| \quad (4.2)$$

In table 4A, eye openness is expressed as a fraction and not as a percentage. Thus it is the minimum value of a binary one expressed as a fraction of the nominal value of a binary one.

The discrete, random addition and subtraction of ISI has an effect reminiscent of, but not the same as, a binomial distribution. From the central limit theorem [37], the probability distribution of the signal value within the possible limits will approximate a Gaussian distribution. The most probable value will be the mean, the value the signal would have had were there no ISI.

4.3.2 Error Power

The error power was defined in section 2.2.1 as the mean square value of the difference between the theoretical (or desired) value of the signal and its actual value. The error itself comprises the ISI, as defined in section 4.3.1, plus the effect of the additive noise of the channel.

The off-lobe data symbols comprising the ISI are effectively uncorrelated with the main lobe. Furthermore, the noise entering the equaliser from the channel may be regarded as white and Gaussian [37]. This gives the following formula for the error power, Q, of the output of an equaliser for QPSK modulation with bipolar binary signals on either channel.

$$Q = A^2 \left(\sum_i |H(i)|^2 - Re^2(H(m))\right) + \sigma^2 . \sum_i |h(i)|^2 \qquad (4.3)$$

Where A^2 is the channel input power, m denotes the main lobe, H(i) denotes the system impulse response, σ^2 is the variance of the additive noise and h(i) are the equaliser impulse response. The summations are over all existing values of the variables.

It was stated in section 4.3.1 that the distribution of ISI approximates to truncated Gaussian. Although the central limit theorem does not give information about the behaviour of the function near the edge of the distribution, the probability of the edge being encountered is very low. A 16-tap equaliser operating on a channel whose impulse response has the effective length of 16 samples will generate a system impulse response 31 samples long. When the imaginary part of the impulse response is taken into account, this gives a probability of 2^{61} that the edge of the distribution is reached. One may thus assume with some confidence that the error distribution is true Gaussian, and little calculation error will be caused by the approximation.

Figure 4.5 shows a graph of the probability of an error occuring in the detection of a bipolar binary signal plotted against the signal to noise ratio in dB [37]. It shows the threshold effect

where an increase of 2.3 in noise power increases the error probability from 10^{-8} to 10^{-4}. On most PSTN lines the additive noise level is below 30 dB. The error power tolerated in a modem is heavily dependant upon modulation type but is generally expected to be below 20 dB [52]. The contribution due to ISI will be discussed in section 5.1.

4.4 The Multidrop Environment

There are many communications applications where a single cable is used to communicate with many modems. A typical example may be a railway signalling application where signalling commands are issued from a signal box to devices along the track and data from the devices is relayed to the box.

The normal way of achieving this is to use a multidrop system. The traffic is controlled from the master station, its modem transmitter sending to the receivers of all other modems. When a message is required from an outlying device, the master commands the outlying slave to send. The slave then, and only then, connects its transmitter to the return line and commences sending. The only receiver connected to the return line is that of the master station.

Equalisation presents a problem for the master station receiver, since it must rapidly adjust to one of many different transmitters and channel responses. The problem may be tackled in one of two ways. The first is to remember each equaliser setting for each slave transmitter and install that setting when the slave is commanded to transmit. The second is to re-equalise the master receiver after each change in transmitter. The latter technique is favoured as it requires less communication between modem and other data processors within the master station. However, it does present a need for a fast-converging equaliser if data traffic is to be efficient. This is a typical application for an equaliser of reliable ROC, such as a lattice.

4.5 Raised Cosine Channels

Both real and complex test channels are used in this thesis. The complex test channels are equivalent baseband impulse responses (cf table 4A), which are analysed in the next section. The real channels are of synthetic raised cosine type, also used in [38]. The transfer function is of the form:

$$C(z) = a + z^{-1} + a.z^{-2} \qquad (4.5)$$

Where $0 < a < 0.5$

This gives a maximum gain at DC, when the voltage gain is (1 + 2.a). The minimum gain, at Nyquist frequency, has a value of (1 - 2.a). Thus the power gain and bounds on the eigenvalues are given by:

$$\text{Power gain (DC)} = (1 + 2.a)^2 \qquad (4.6)$$

$$R(0) = 1 + 2.a^2 \qquad (4.7)$$

$$\text{Power gain (Nyquist)} = (1 - 2.a)^2 \qquad (4.8)$$

As the order of equaliser and autocovariance matrix rise, the eigenvalues tend asymptotically towards their extreme values [40].

4.6 An Analysis of Modem Channels

This section summarises the characteristics of some modem channels likely to be encountered in Great Britain. The summary is based on an analysis of the the equivalent baseband impulse responses supplied by BTRL and included in appendix D.

Table 4A presents the results of the analysis in tabular form. Nine impulse responses were used in the calculation. For each impulse response, calculations were done for equalisers having 8, 11, 16, 22 and 32 taps. For each set of taps four measurements were

taken. The autocovariance matrix of the channel was calculated and its eigenvalues obtained using a standard library package. The first measurement was the ratio of maximum to minimum eigenvalue, expressed as a fraction. The second was the ratio of the minimum eigenvalue to the mean power of the channel. These first two measurements were to assess the slowing effect of the eigenvalue disparity on the rate of convergence of a conventional transversal equaliser. The optimum Wiener filter was then calculated and convolved with the channel impulse response to give a system impulse response. Then equation 4.3 was applied to obtain the equalised error power and equation 4.2 gave the openness of the eye after equalisation.

Four of the channels analysed were for V27 modems on typical British lines. These were E10C4, E15, E8C3 and E5. The codes refer to the type of line thus E10C4 means ten miles of local line in tandem with one hundred (= 4 x 25) miles of trunk line. The channel marked E5R is a channel of low distortion formed by taking only the real part of the E5 impulse response. Channel E8C3A is an E8C3 line on which a V29 modem is operating. Those channels marked R11, R21 and R81 are synthetic real channels, described in section 4.5.

Chapter two describes how a truncation of an infinitely long impulse response may be taken, provided one accepts the slight error involved. There is good reason to believe that the 16-point truncation used was inadequate in the case of the E8C3A, V29 channel. The latter impulse response had its power dispersed over the truncation, so that a large amount must also have fallen outside the truncation taken. It is therefore better to regard the channel as simply a difficult impulse response and not one representative of British lines.

4.6.1 Results

The first hypothesis to be tested was that of section 3.3.1, that eigenvalue ratio rises with the order of the autocovariance matrix, eventually tending towards an asymptote, which may be regarded as the ratio for a matrix of infinite order. Figure 4.6 is a graph of eigenvalue ratio against order for the four V27 channels. It shows

that eigenvalue ratio always rises with order and that the differential always decreases with order. The orders considered are not high enough to show the asymptotic flattening expected with high orders. The graph also shows that a V27 modem on British telephone lines using equalisers of less than 32 taps will not encounter an eigenvalue ratio of more than 23 or so.

The second hypothesis investigated involved the pole-zero modelling of channel impulse responses. Those channels with badly distorted power spectral densities will have high eigenvalue ratios, according to section 3.3.1. The fluctuations in their frequency responses are due to poles and zeroes near to the unit circle, which will require similarly placed zeroes and poles in the equaliser response to cancel them out. Poles near the unit circle create long impulse responses which are badly distorted by truncation. There should therefore be a correlation between equalised error and eigenvalue ratio.

Figure 4.7 is a scatter diagram for various channels, plotting eigenvalue ratio at order 32 against equalised error for a 16-tap equaliser. The high order eigenvalue was chosen in an attempt to approach the asymptotic value for the channel. A 16-tap equaliser was selected because this is a typical length for V27 channels. The result is a scatter diagram so dispersed as to show no trend at all. Clearly the link between eigenvalue ratio and equalised error is tenuous or non-existent.

The third hypothesis investigated is that equalised error is inversely proportional to the number of taps in the equaliser. Certainly equalised error is a non-positive falling function of equaliser taps. An equaliser of order (N+1) can have the same tap values as that of order (N), with the extra tap set to zero, it will then have an identical equalised error.

Figure 4.8 is a graph of equalised error in dB, plotted against equaliser order. The graph shows that increasing the order reduces the error. Were the lines straight, the inverse proportionality could be said to exist. The lines are not straight but largely so.

There are both positive and negative minor changes in slope. The coefficients of proportionality vary widely according to channel type. In general one may say that the lines with the longest lengths of trunk cable require equalisers of higher order, and that a 16-tap equaliser guarantees an equalised error of under 20dB for V27 channels.

The fourth hypothesis is that there is a strong correlation between eye openness and equalised error power. Certainly, if viewed rigorously, there will be functions which give widely differing values for equations 4.2 and 4.3. Nevertheless, it is intuitive to assume that a small error corresponds to an open eye.

Figure 4.9 is a graph of eye openness against equalised error in dB. It is a scatter diagram for all the points of table 4A which fall within the bounds of the graph. One may conclude that the relationship is statistical, i.e. it is a correlation and not a function. Nevertheless, the correlation is a strong one and for a given eye openness it is possible to predict the error to within 10 dB. As would be expected, the error power is proportional to the square of the amount by which the eye is closed.

The E8C3A impulse response is interesting in that it produces negative eye openings for equalisers of orders up to 32. This means that even in the absence of noise there will be data sequences which will cause detection errors. Figure 4.10 is an extension of the graph of eye openness against equalised error, including the negative points of the E8C3A impulse response. The graph shows a continuation of the trend, and to the same degree of accuracy of 10 dB.

4.7 Summary

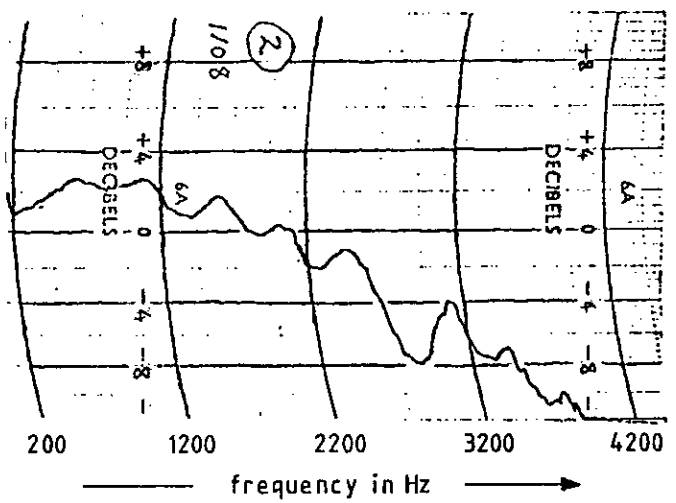
This chapter has explored the environment in which an equaliser must operate. It has assigned numerical values to the mean-square errors and eigenvalue ratios likely to be encountered. The approximate concept of the eye diagram has been treated mathematically and an expression derived for calculating eye openness and mean square error, given the system impulse response. The

properties of typical channels have been examined and it has been determined that there is no significant correlation between equalised error and channel eigenvalue ratio, also that equalised error is approximately inversely proportional to equaliser order.

TAPS	EIGENVALUE RATIO	1/EV(MIN)	MSE IN dB	EYE OPENNESS
E10C4				
8	6.658609	3.403571	-12.35137	0.1046817
11	7.677429	3.792810	-18.00791	0.5067727
16	8.850311	4.254626	-22.38550	0.7154701
22	9.726448	4.608709	-28.31456	0.8554990
32	10.48987	4.920909	-39.31444	0.9590586
R11				
8	10.21256	5.063713	-41.18054	0.9851232
11	11.17145	5.439942	-59.12964	0.9978568
16	11.84226	5.698766	-80.99513	0.9998688
22	12.15660	5.818564	-123.8525	0.9999962
32	12.35414	5.893235	-166.8401	1.000000
R21				
8	19.05673	8.750297	-33.10348	0.9574884
11	21.87949	9.846131	-47.19351	0.9907299
16	23.94901	10.63611	-66.05511	0.9990363
22	24.94018	11.00978	-99.70486	0.9999441
32	25.56765	11.24436	-140.5470	0.9999994
E15				
8	14.27068	5.993071	-25.49586	0.7860736
11	16.87962	6.793363	-30.71485	0.8804003
16	19.39083	7.560605	-39.28054	0.9602217
22	20.91163	8.024557	-52.42568	0.9911799
32	22.03434	8.365689	-71.91224	0.9990726
E8C3A				
8	11.06897	5.187501	-8.677869	-0.8171989
11	15.37366	6.898864	-10.83935	-0.5811752
16	24.97954	10.79023	-12.60978	-0.4115031
22	39.32095	16.62610	-14.12917	-0.2935160
32	66.91224	27.85939	-16.30831	-0.1190169
E8C3				
8	4.454976	2.530446	-19.62250	0.6551821
11	4.909588	2.711337	-25.62937	0.8084315
16	5.352496	2.890815	-31.07352	0.9039448
22	5.640431	3.010756	-39.38834	0.9623141
32	5.868379	3.107154	-54.20237	0.9934513
E5				
8	2.485475	1.691275	-41.83129	0.9738081
11	2.612868	1.744576	-48.26092	0.9892463
16	2.716289	1.787455	-65.28307	0.9986141
22	2.771251	1.810069	-94.40558	0.9998776
32	2.809024	1.825519	-128.7572	0.9999974
E5R				
8	1.025646	1.009855	-73.77883	0.9996178
11	1.026830	1.010029	-93.55060	0.9999183
16	1.027663	1.010140	-159.9963	0.9999725
22	1.028061	1.010190	-172.0057	1.000000
32	1.028317	1.010221	-213.1534	1.000000
R81				
8	45.07692	19.33276	-24.41294	0.8646705
11	57.91903	24.29729	-34.37289	0.9535904
16	68.61058	28.36772	-48.44526	0.9912261
22	74.07352	30.42401	-66.65412	0.9989016
32	77.61807	31.74789	-103.5993	0.9999656

Table 4A Statistics calculated from BTRL equivalent baseband impulse responses.

frequency	attenuation	group delay
(Hz)	(dB)	(μs)
200	15	1450
300	14	700
400	14	570
500	14	280
675	13	180
900	14	200
1100	14	260
1300	16	-160
1500	16	290
1700	16	0
1900	18	10
2100	18	400
2400	19	850
2700	24	-150
3000	23	340
3500	30	320



Route from Kings Langley to Wordsworth exchange, both in North West London

Fig 4.1 The frequency response of a PSTN telephone channel.

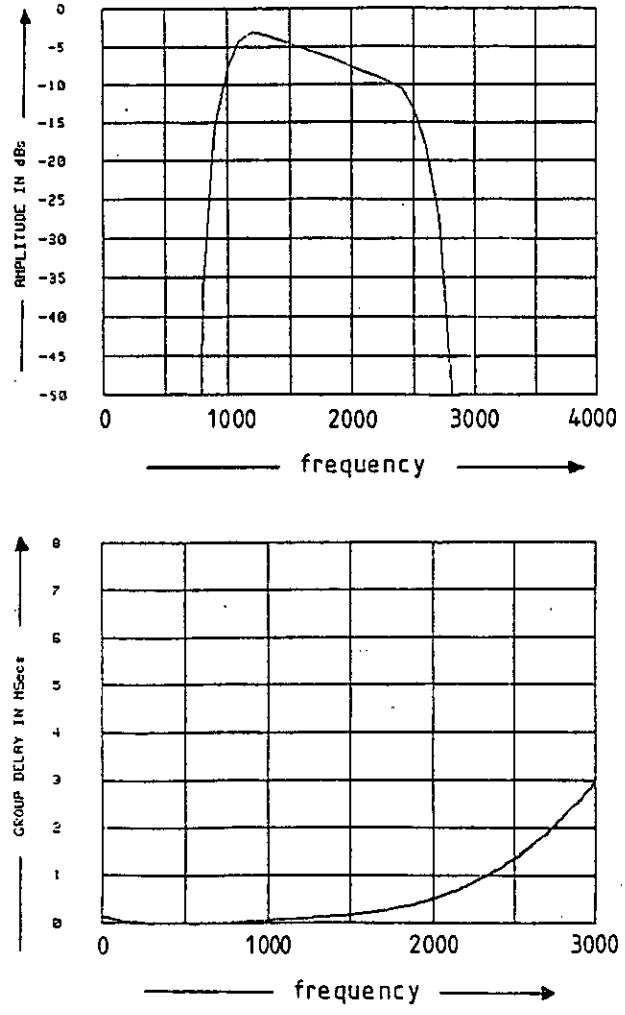


Fig 4.2 The characteristics of an average telephone channel (E8C3) including the effect of V27 line filters

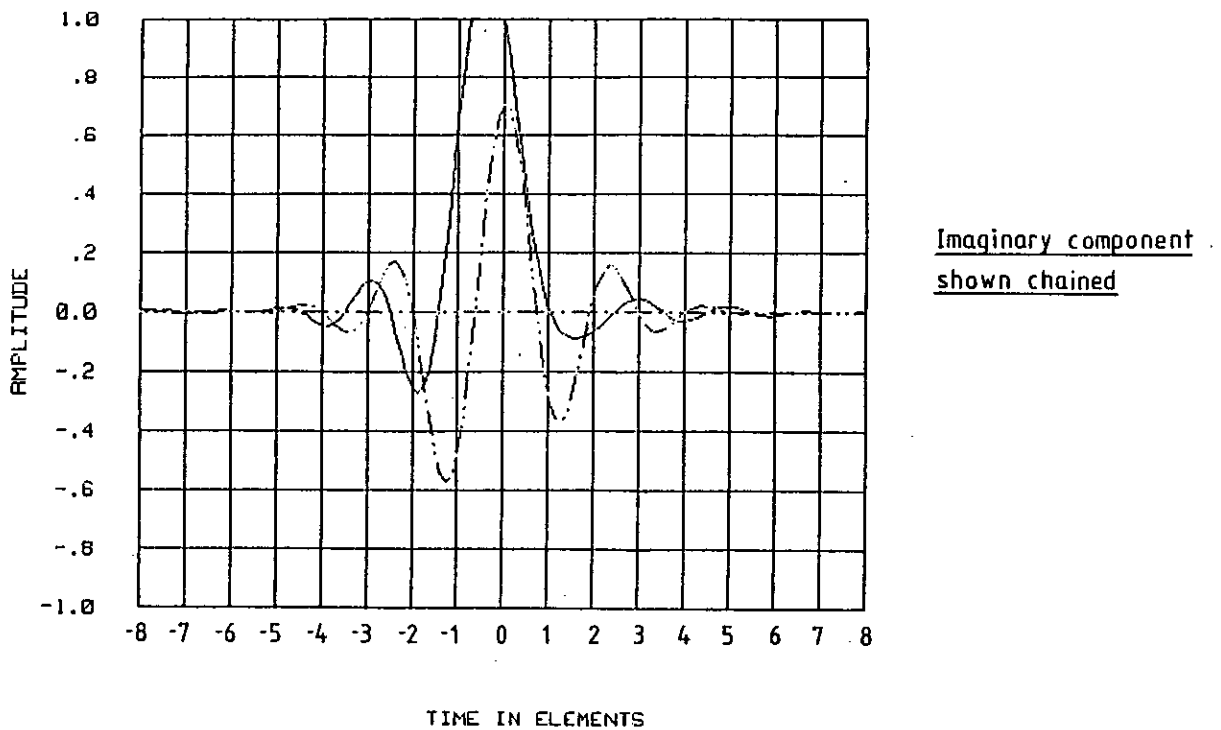


Fig 4.3 The equivalent baseband impulse response of an average V27 modem channel (E8C3).

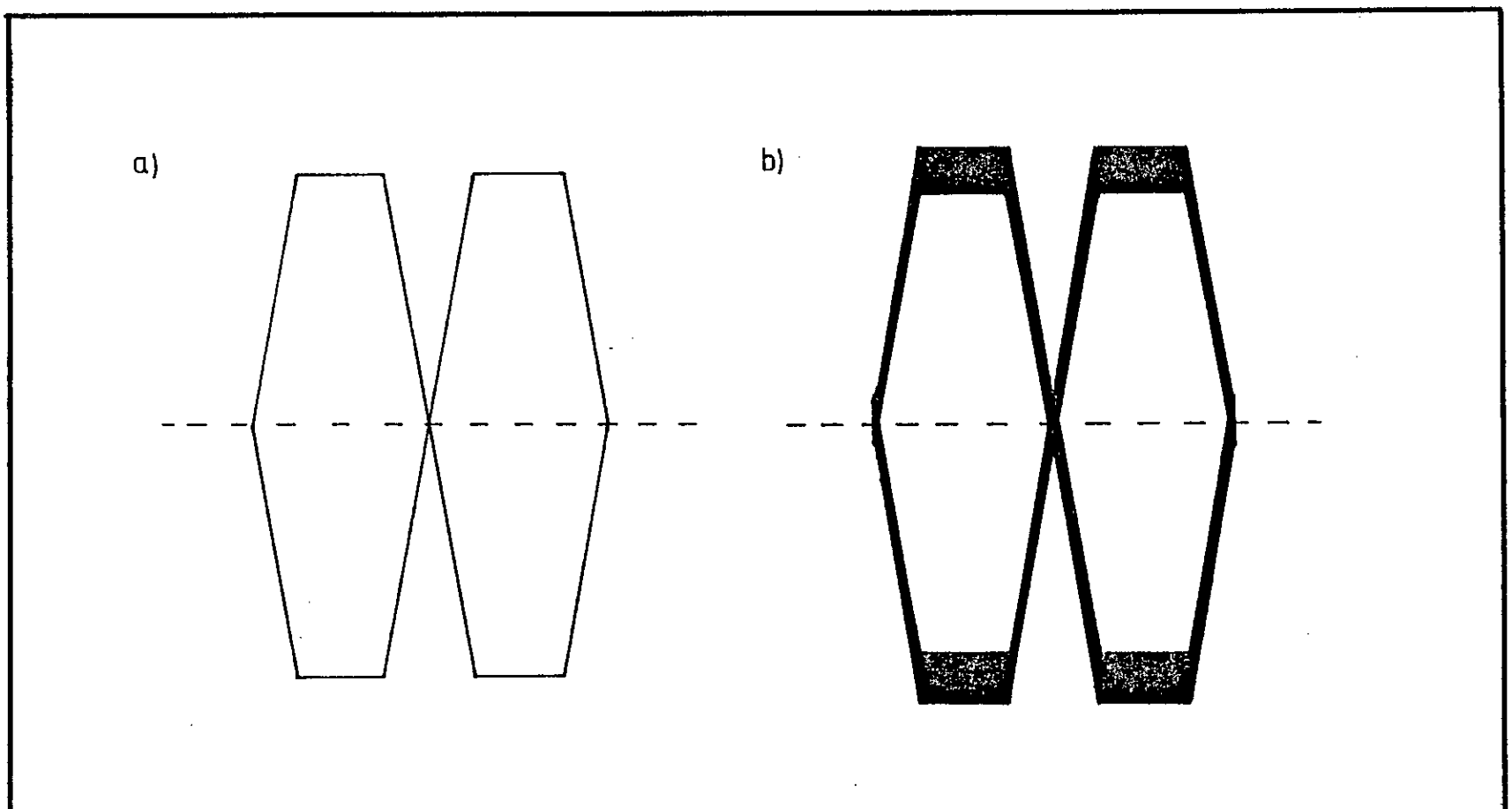


Fig 4.4 a) A perfect eye diagram.
b) An eye diagram showing ISI.

From [37]

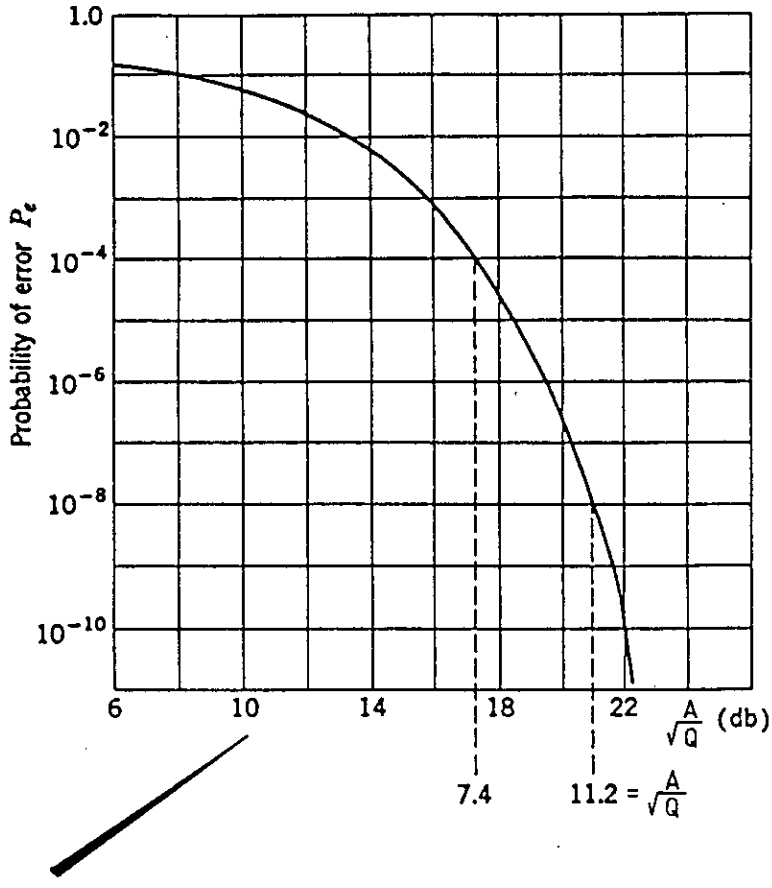


Fig 4.5 Probability of error for binary detection in Gaussian noise

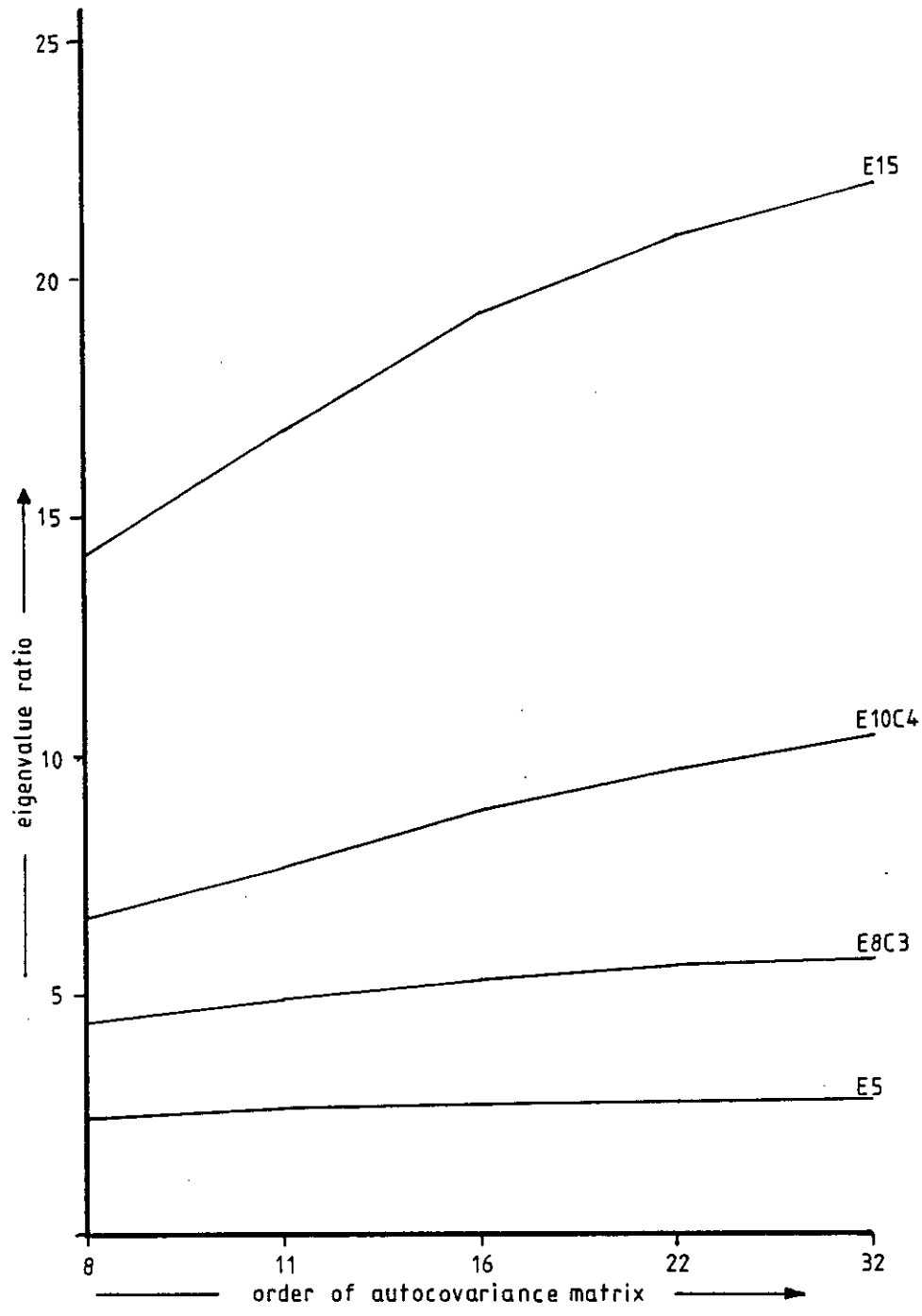


Fig 4.6 A graph of eigenvalue ratio against matrix order for various impulse responses.

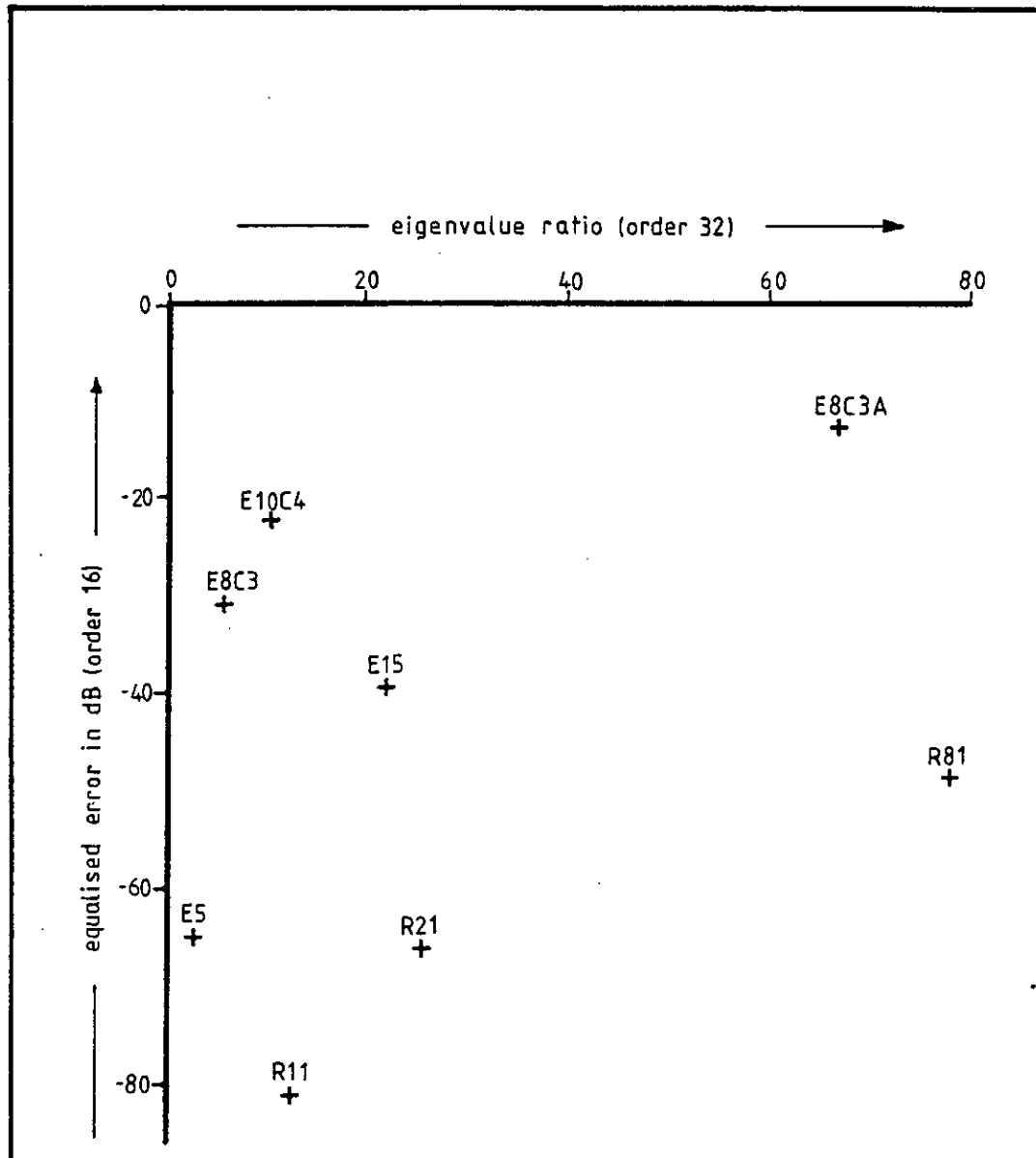


Fig 4.7 A scatter diagram of eigenvalue ratio versus equalised error.

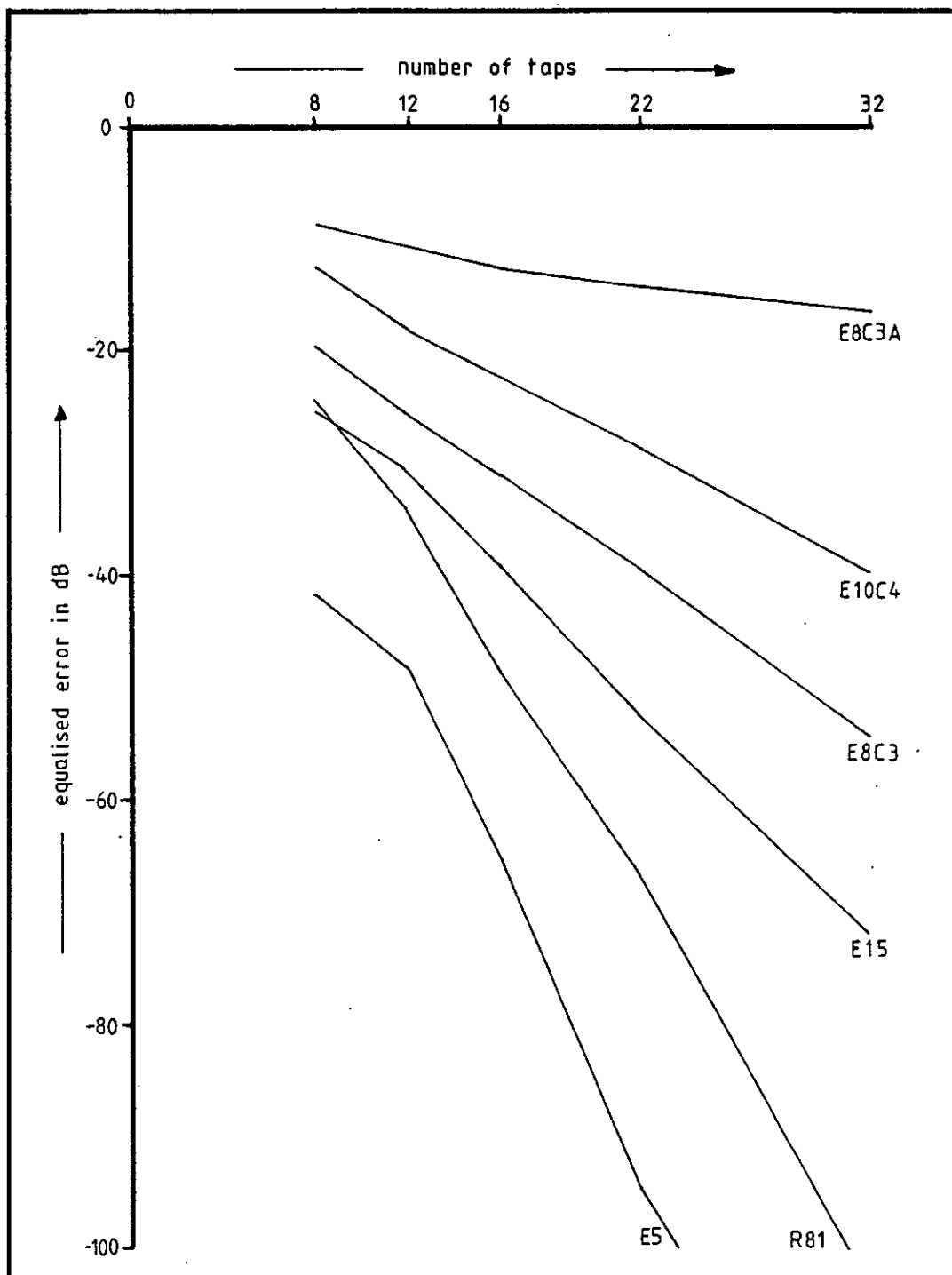


Fig 4.8 A graph of equaliser order versus equalised error power

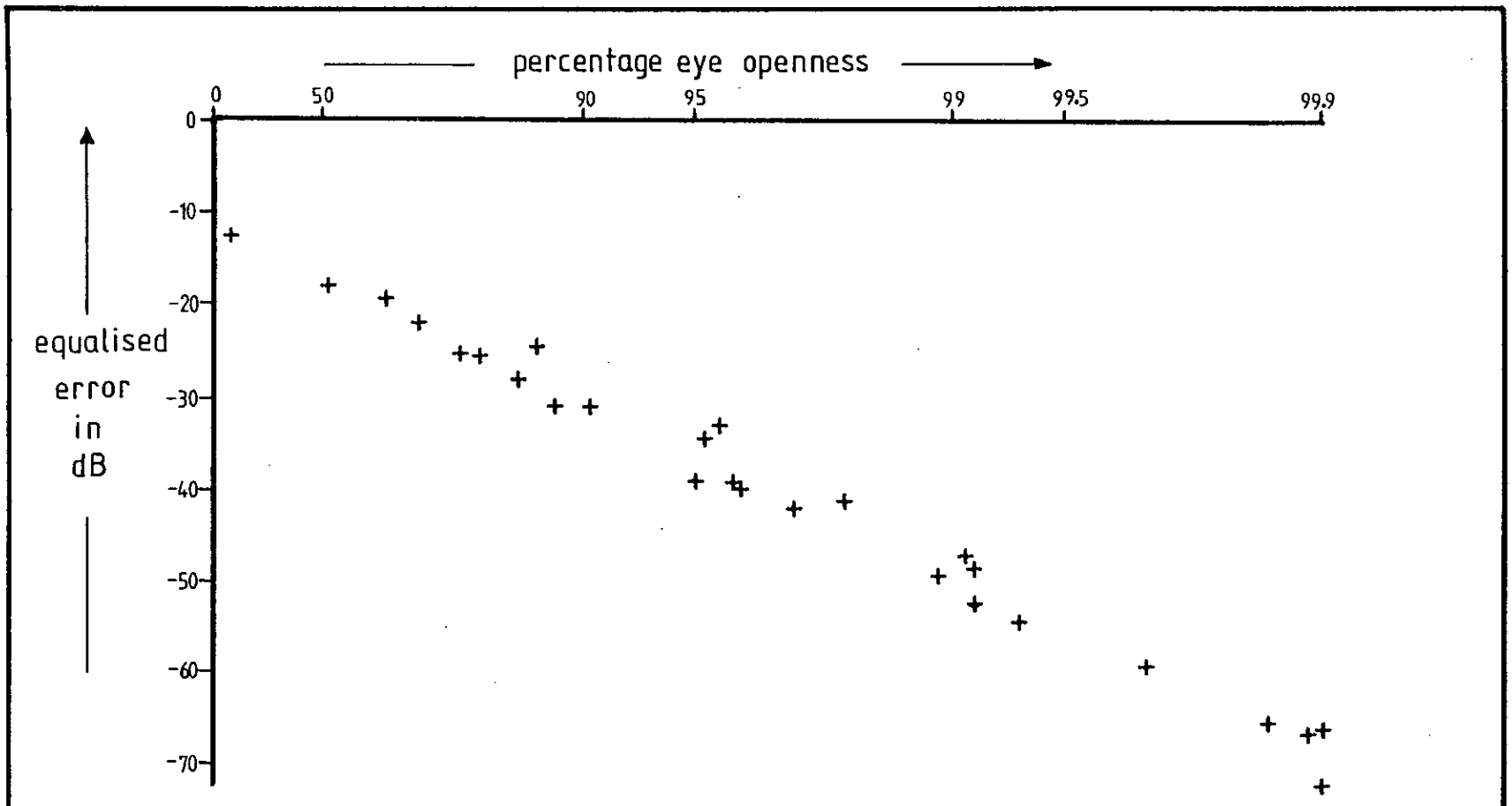


Fig 4.9 A scatter diagram of eye openness against equalised error power

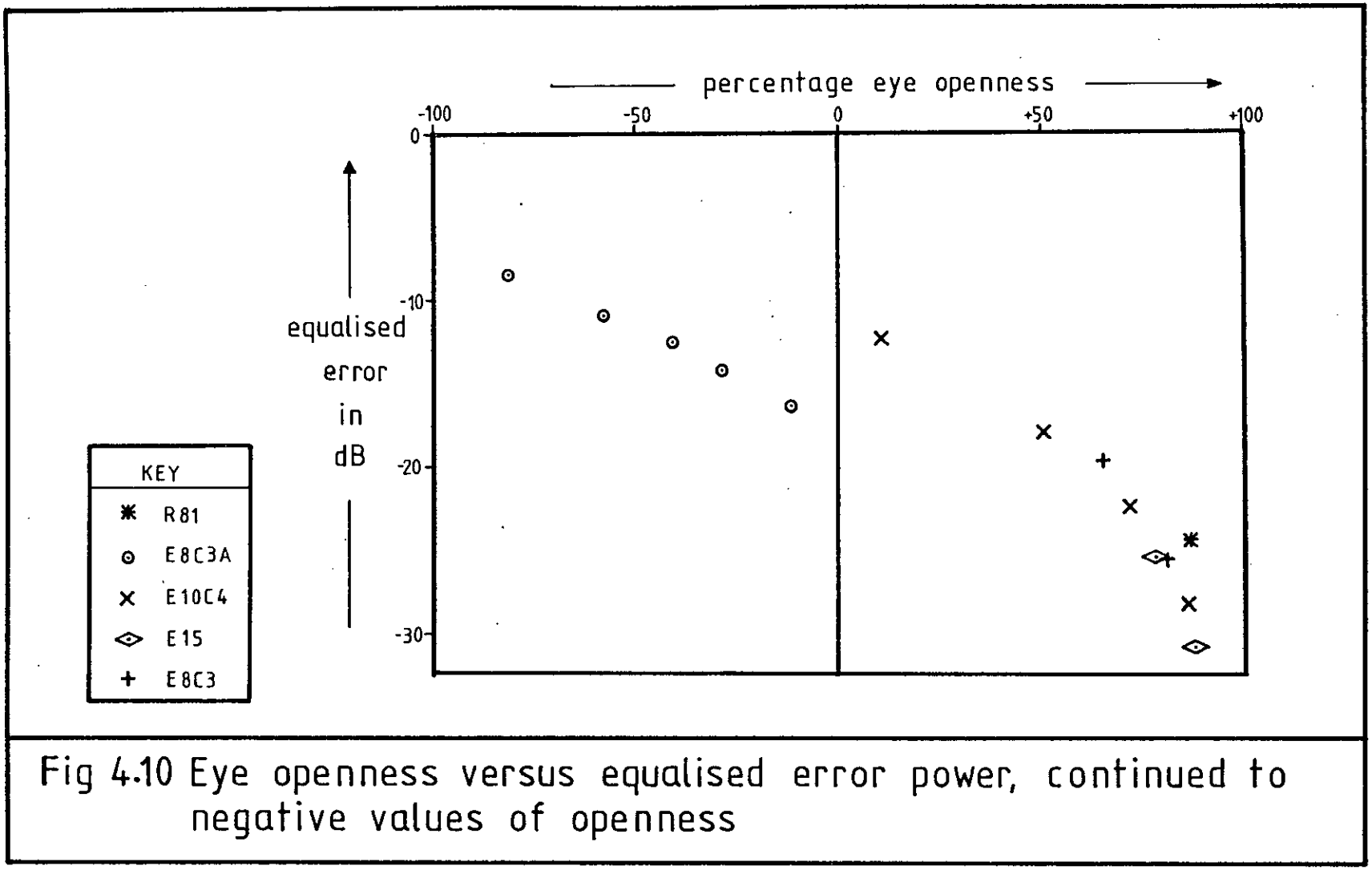


Fig 4.10 Eye openness versus equalised error power, continued to negative values of openness

Chapter Five

SIMULATIONS OF EQUALISER PERFORMANCE

Section one of this chapter discusses sources of error in gradient adaptive filter structures. The discussion covers both those errors which progressively reduce during the convergence of the filter and those which remain after the filter has converged. This is explained using the widely applied adaptive transversal filter design as an example. Section two applies this discussion to the adaptive lattice structure and equaliser designs. It shows that while the addition of a lattice can accelerate the convergence of an equaliser on certain channels, it results in a penalty of increased converged error. In some cases, the converged error may be increased to such an extent that it is unsuitable to apply lattice filter designs to these applications. Section three explores various specific lattice filter designs, some of which provide fast convergence with a low converged error. Table 5B gives the standard conditions under which these simulations were performed. The conclusions which may be drawn from the simulations included in this chapter are summarised in section four.

5.1 Sources of Error in Gradient Algorithms

The error power E $\langle e^2 \rangle$ of an adaptive filter is the mean square value of the error signal (cf section 2.2). There are four main sources of error in gradient adaptive filters: Channel noise is the addition to the input of a broadband unwanted signal, which is uncorrelated with the training signal. Polynomial degree refers to the length or order of the filter and the resulting limitation on its ability to equalise the signal. Algorithm noise is a small random fluctuation in the coefficient values of the converged filter, caused by the operation of the gradient algorithm. Lastly there is non-convergence, which simply refers to a converging filter which has not yet minimised its output error power. The four error sources are not totally independant of one another and interact, usually to the detriment of the filter output.

5.1.1 Channel Noise

Channel noise is a broadband signal, which is usually introduced into the channel by its physical construction, e.g. thermal noise in amplifiers or switching transients. This noise signal sets a bound on the received signal to noise ratio (SNR), which also restricts the equalised SNR. In a few cases it may be possible to obtain a separate correlated sample of the noise signal alone, without the transmitted data, which would permit the application of noise cancellation techniques [3] for SNR enhancement prior to equalisation.

5.1.2 Polynomial Degree

When equalising an ARMA channel response with an MA equaliser, equalisation is only approximate (cf section 4.3.2). The degree of intersymbol interference (ISI) left on the equalised signal then contributes to the error signal. In this case the error power is due to off-lobe terms in the system impulse response (cf section 2.1).

Channel noise also affects the error due to polynomial degree, worsening the ISI on the equaliser output. This may best be explained with reference to the autocovariance matrix of the input to a transversal filter. In the Wiener-Hopf equation (equation 2.18) the cross-covariance of the input signal with the training signal is unaffected by noise, as are all off-diagonal terms of the input autocovariance matrix. The diagonal terms, however, R(0) (the channel power estimate) is raised by the power of the noise in the channel. Thus:

$$R_N(0) = R(0) + \sigma^2 \qquad (5.1)$$

Where $R_N(0)$ is the actual channel power estimate, R(0) is the value had the noise not been present and σ^2 is the variance of the noise. The resulting alteration to the values of the diagonal terms in the autocovariance matrix will in turn alter the values of the Wiener equaliser coefficients. The equaliser coefficients will then no longer have those values which would produce minimum ISI in a

noise-free channel. The result is a deterioration in error power over and above that caused by the additive noise, caused by the increased ISI. Thus in a channel for which a certain equaliser could have produced an error power of -50 dB, but which has additive noise 30 dB below the signal, the equalised error will be higher than the expected -29.9 dB. It will be closer to a value of -27 dB.

Under high noise conditions the significance of the off-diagonal terms in the autocovariance matrix progressively reduces with increasing noise. Eventually only a diagonal matrix remains, whose diagonal terms are equal to the noise power estimate:

$$\underline{R} \rightarrow \sigma^2 \underline{I} \quad \text{as} \quad \sigma^2 \rightarrow \infty$$

$$\underline{R}^{-1}\underline{P} \rightarrow \underline{P}/\sigma^2 \qquad (5.2)$$

This explains how under high noise conditions the coefficients of the equaliser converge towards the matched filter for the channel impulse response (cf section 2.2.3).

5.1.3 Algorithm Noise

After an equaliser's coefficients have converged towards the optimum values, the recursive convergence process will continue to randomly disturb the coefficient values. This arises from the short-term estimates of $\underline{R}$, $\underline{P}$ and thus the gradient, fluctuating around their stationary values. In the Widrow LMS algorithm, equation 3.6, the update term is equal to the stepsize times the product of the input signal vector and the instantaneous error value. Since both of these signals are uncorrelated after convergence (equation 3.10) the update term will have zero mean. This will leave the mean values of the equaliser taps unaltered. The variance of the update term, however, will provide a non-zero disturbance which is dependent on the variances of the two signals and the selected stepsize. The effect on the coefficients of the filter is a small random deviation from their optimal value. Viewed instantaneously, this small misadjustment raises the error power above its optimum value. Viewed in the long term, this addition to error power by small, randomly

fluctuating misadjustments is called algorithm noise.

Ungerboeck [7] noted that it was impractical to calculate the highest eigenvalue of the channel and use this to set the maximum value of the stepsize to provide the most rapid convergence. He noted instead that the equaliser input power R(0) was an easier quantity to measure, e.g. with the alpha filter of equation 3.27. Now for a channel of unity eigenvalue ratio all eigenvalues are equal to R(0) and for spreads of eigenvalue ratios R(0) is the average of the eigenvalues [40]:

$$R(0) = \left(\sum_{i=1}^{N} \lambda(i) \right)/N \qquad (5.3)$$

Where N is the number of taps and $\lambda(1...N)$ are the eigenvalues of the channel autocovariance matrix. He suggested a formula for approximately evaluating algorithm noise:

$$E\langle e^2 \rangle = \left(\sum_{t=1}^{N} A.B^t \right) + 2.e^2(opt)/(2 - \mu.N.R(0)) \qquad (5.4)$$

Where e(opt) is the rms value of the optimum error power for the equalised channel, and $E\langle e^2 \rangle$ is the actual error power when the algorithm noise is taken into account. The terms A and B are related to the stepsize and the channel autocovariance matrix, as defined in [7]. Since $B < 1$, the term becomes insignificant as the iteration number, t, rises to high values. B is also dependent upon the stepsize, implying that larger stepsizes lead to a faster convergence but a greater algorithm noise.

For a two-dimensional (I + Q) equaliser, N may be regarded as the number of degrees of freedom, i.e. twice the number of taps. R(0) is taken to refer to the power per channel. Thus a two-dimensional equaliser with the same algorithm noise per channel as a one-dimensional type would in general need half the stepsize. It would then converge at half the rate of the one-dimensional equation.

5.1.4 Non-convergence

Non-convergence is simply a deviation in the filter weights from those which provide e(opt), the minimum error. It may be due to algorithm noise, but is usually taken to refer to the coefficients of a filter which is converging but has not yet converged from some arbitrary tap setting towards the optimum coefficient values. In the simulations presented in this chapter, and indeed the simulations presented in most of the published literature, the standard initial tap setting is normally taken as the zero vector. In hardware realisations two further choices are the totally random setting present when the device is turned on, or a single non-zero tap setting in the middle of the filter. The latter setting is very close to the optimum for well-conditioned channels and often allows decision directed feedback (DFB) operation to be started immediately [1].

A detailed discussion of the rate of convergence is outside the scope of this thesis, but the following comments may be made: The convergence of the filter is asymptotic, so although the filter will never reach convergence, the contribution of non-convergence to the error power will rapidly become insignificant. According to Widrow's model (equation 3.17) the rate of convergence is dependent upon the stepsize and the eigenvalues.

Ungerboeck [7] noted that the optimum stepsize for the fastest convergence was:

$$\mu(\text{opt}) = [E\langle e^2\rangle - e^2(\text{opt})]/(N.R(0).E\langle e^2\rangle) \qquad (5.5)$$

Where $E\langle e^2\rangle$ is the error arising from the current setting of the taps and N is the number of taps in the algorithm. R(0) refers to the input to the tap in question. At the start of convergence, when the equaliser error greatly exceeds e(opt), the equation is approximately:

$$\mu(\text{opt}) = 1/(N.R(0)) = 1/\left(\sum_{i=1}^{N} \lambda(i)\right) \qquad (5.6)$$

This will hereafter be referred to as the 'optimum stepsize'. It will lead to a very rapid convergence to within 3 dB of e(opt). Ungerboeck suggested that the difference in the rates of convergence and converged error were so small in most cases that there was no need to use the full formula of equation 5.5 for transversal equalisers, but that equation 5.6 was sufficiently accurate.

After convergence equation 5.4 becomes:

$$E < e^2 > \; = \; 2.e^2(opt) \; / \; (2 - \mu.N.R(0)) \qquad (5.7)$$

Substituting the value of equation 5.6, one gets:

$$E < e^2 > \; = 2.e^2(opt) \qquad (5.8)$$

I.e. the recursion will eventually converge to give an algorithm noise equal to the optimum error power, providing a converged error power 3 dB greater than optimum.

The optimum stepsize for the algorithm is set by R(0) and the minimum eigenvalue governs the convergence of the slowest eigenvector. It can therefore be deduced that the rate of convergence is proportional to the ratio $\lambda(min)/R(0)$. Commenting on the converging term of equation 5.4, Ungerboeck [7] stated that a channel of unity eigenvalue ratio and optimum stepsize would gradually reduce its error power according to:

$$E < e^2 \; (t+1) > \; = E < e^2 \; (t) > \; . \; (1 - 1/t) \qquad (5.9)$$

Where t is the iteration number. Thus the error power reduces with increasing iterations. Alternatively, with reference to equation 5.4:

$$\Sigma AB^t \; \equiv \exp \; (- \; t/N) \qquad (5.10)$$

This assumes that the optimum stepsize has been selected. Should the eigenvalue ratio be greater than one, the eigenvalues will assume values which lie on the power spectrum of the input signal [40].

While the ROC depends on all of the eigenvalues, the reduction in the rate of convergence was noted in simulation work to be approximately proportional to the square root of the normalised value of the minimum eigenvalue (cf section 3.2.1.2). Figure 5.1 shows the convergence curves of two 8-tap transversal filters converging on channels (R11 and R81) of normalised minimum eigenvalues 3.4 and 19.3, illustrating this point. Thus, assuming that the optimum stepsize has been adopted, equation 5.4 may be re-expressed as:

$$E < e^2 > = \exp\ (t/(N.\sqrt{\lambda(\min)/R(0)}) \ + \ 2e^2(\text{opt}) \qquad (5.11)$$

While for other stepsizes the approximate equation becomes:

$$E < e^2 > = \exp(t.\mu/(N^2\sqrt{\lambda(\min)/R(0)}) \ + 2.e^2(\text{opt})/(2 - \mu.N.R(0)) \qquad (5.12)$$

5.1.4.1 The Effect of Channel Noise on Rate of Convergence

It has been informally reported by many transversal adaptive filter researchers that the addition of a little white noise to the channel accelerates convergence. The reason for this lies in the effect it has on the eigenvalues of the channel.

The characteristic equation for an autocovariance matrix is:

$$\begin{vmatrix} R(0)-\lambda & R(1) & \ldots & R(N-1) \\ R(1) & R(0)-\lambda & \ldots & R(N-2) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ R(N-1) & R(N-2) & \ldots & R(0)-\lambda \end{vmatrix} = 0 \qquad (5.13)$$

Now let the term R(0) refer to the power estimate of the noise-free channel, $\lambda(a)$ refer to an eigenvalue of the channel and $\lambda(n)$ refer to the corresponding eigenvalue of a noisy channel. Then:

$$R(0) - \lambda(a) = \sigma^2 + R(0) - \lambda(n) \qquad (5.14)$$

$$\lambda(n) = \sigma^2 + \lambda(a) \qquad (5.15)$$

Thus the noise raises all eigenvalues, reducing the eigenvalue ratio, raising the value of the minimum eigenvalue and speeding up convergence.

This may be explained using figure 5.2. This histogram may be regarded as the power Fourier transform of the channel's transfer function. Since the eigenvalues assume values which lie on the channel power spectrum [40], and they determine the loop gain of the different modes of convergence of the algorithm [53], one may say that the lowest eigenvalues are bounded by the minimum values of the power spectrum. The addition of white noise raises the whole spectrum, increasing the individual eigenvalues. In particular, the smallest eigenvalues are increased, reducing the overall eigenvalue ratio and accelerating convergence.

5.2 Errors and the Lattice

5.2.1 The Convergence of the Lattice Structure

The most comprehensive work on the convergence properties of the gradient adaptive lattice structure was reported by Honig [46]. In a preliminary paper he pointed out that the modes of convergence were unidirectionally coupled to some extent. Thus, as the lattice starts up with all PARCOR coefficients equal to zero, each will initially converge towards one of the lags of the autocovariance function. I.e. since K(1...N) = 0, then from equation 2.62 - 2.64:

$$f(0 \ldots N,t) = f(0,t) = s(t) \qquad (5.16)$$

$$b(0 \ldots N,t) = b(0,t-N-1) = s(t-N-1) \qquad (5.17)$$

so from equation 3.36 :

$$E\langle K(n,t) \rangle = E\langle s(t).s(t-1-N) \rangle / E\langle s^2(t) \rangle \qquad (5.18)$$

Equation 5.18 only refers to the initial or 'target' values

towards which the PARCOR coefficients initially aim.

As convergence proceeds, the lower order PARCOR coefficients tend towards their true values, altering the target values of the higher order coefficients. Thus at initial start-up the lower order stages slow the convergence of the higher order stages by altering the target values towards which they are converging. This has a less serious effect than may first appear, as it is a property of non-deterministic autocovariance functions and their associated PARCOR coefficients that they reduce in value with increasing order (cf section 2.3.7). Thus higher order coefficients will not need to change their values over such a wide range, permitting the target values of the PARCOR coefficients to be tracked more easily.

Once converged, the values of the individual PARCOR coefficients are to some extent decoupled from one another. For example, the lower order stages are unaffected by the behaviour of the higher order stages, since there is no signal path back through the filter. Further, the optimum values of the lower order PARCOR coefficients are independent of changes in the higher order autocovariance lags or the optimum values of the higher order PARCOR coefficients (cf section 3.2.1.1).

Decoupling in the forward direction exists as a first-order effect of the LMS criterion (cf section 3.2.1.1). The signals feeding the nth stage are insensitive to small changes in the values of K(1 ... n-1) once the latter have reached their optimum values. Thus small changes in lower order K values will minimally disturb those of higher orders.

Figure 5.3 shows the convergence of the PARCOR coefficients of a lattice onto a stationary input signal. The signal is a combination of five sinusoids plus a little white noise, and thus untypical of normal non-deterministic functions. The graph shows a rapid convergence of the first, and a less rapid convergence of the fourth PARCOR coefficient towards their optimum values. The eighth and twelfth coefficients have clearly started to converge towards one pair of values and then changed course as earlier coefficients have

deflected the target values towards which they are converging. Indeed, the twelfth coefficient takes a significantly positive value for the first seventy iterations, before going negative towards its true final value.

In the published literature on gradient lattices [15, 24, 38], the assumption is made that the convergence speeds should be the same for all stages. This allows all the lattice coefficients to track changes in the input signal at equal rates.

5.2.2 The Convergence of the Lattice Equaliser

The behaviour during convergence of multistage adaptive filters with chained distributed algorithms, such as the lattice equaliser, is a subject which has been hitherto inadequately covered. Even Honig [46] limited his mathematical description to two-stage lattice structures. Satorius [38] implied that a good solution is to give both PARCOR coefficients and equaliser sidetaps the same rate of convergence. This was confirmed in the following simulation, which used the structure of figure 1.7 and the algorithm of table 5A.

Table 5A is the full algorithm which is used directly or in slightly modified form for all the following simulations in this chapter. It is essentially the same as that used by Satorius [38], with two exceptions. Satorius' simulation was purely real with a synthetic channel response. Satorius also used an extra computation for the power estimate for the sidetaps:

$$C_G(n,t+1) = C_G(n,t) \cdot (1-\alpha) + |b(n,t-1)|^2 \qquad (5.19)$$

$$\mu_G(n,t+1) = 1/C_G(n,t+1 \qquad (5.20)$$

This was a curious step, since the purpose of the power estimate recursion was merely to estimate the power at the input of a particular lattice stage and thus set the optimum stepsize. Since the power in forward and backward channels is identical [21], the extra recursion to calculate the sidetap stepsize will produce a noisier estimate with half the mean value.

In the simulations a value of 0.07 was selected for α. Three convergence curves appear on figure 5.4. The dotted line corresponds to the algorithm of table 5A in which the sidetap stepsize is twice that of the lattice PARCOR coefficients. The other two curves correspond to stepsize ratios of 1.414 and 2.828. A number of simulations were performed, with β reduced progressively by a factor of 0.707 for each successive run. For each iteration number the minimum corresponding error power for any simulation was recorded and entered on the graph. The graph is thus a composite one which eliminates the effect of the stepsize by showing the best result out of a number of different stepsize trials. These complex simulations used the complex channel impulse response E8C3 (cf table 5B).

The result shows that the dotted line, representing a sidetap stepsize ratio of 2, converges faster. Further simulations on other channels confirmed this result. The factor of 2 stems from the fact that the PARCOR update recursion has two correlation terms, whereas the sidetap update has only one. The factor of two thus ensures that all of the distributed algorithms converge at the same rate.

5.2.3 Algorithm Noise in the Lattice Structure

The problem of algorithm noise in the lattice structure may lead to unsuitably high values of converged error in equalisation applications. The power on the output of the nth stage of a lattice which has a fixed, correct value for its PARCOR coefficient is, [21]:

$$Q(n) = (1 - |K(n)|^2)\, Q(n-1) \qquad (5.22)$$

Where Q(n-1) is the power on the input to the stage. Applying this to equation 5.7 gives a value for the output power of a stage using a recursive gradient algorithm with only one correlation term:

$$Q(n) = 2.(1 - |K(n)|^2).Q(n-1)/(2 - \mu.Q(n-1)) \qquad (5.23)$$

The addition of a second correlation term to form the PARCOR recursion of equation 3.41 will double the rate of convergence but

increase the noise by 2, since the addition is non-coherent. This leads to the equation:

$$Q(n) = (1 - |K(n)|^2).Q(n-1)/(1 - 0.707.\mu.Q(n-1)) \quad (5.24)$$

Since the divisor in equation 5.24 is always less than unity, this leads to an increase in output power due to algorithm noise. The noise will increase as μ increases and decrease as K deviates from zero towards its bounds of $\pm$ 1. Since for stochastic signals the value of the PARCOR coefficient rarely exceeds $\pm$ 0.5 and the value of μ must be high for fast convergence, this makes the algorithm noise very high. The process is analogous to the increase in noise when using a distributed, rather than a global linear combiner (cf section 3.2). The noise, and indeed the whole output signal, tends toward zero as the optimum magnitude of K tends to unity. However, this condition is rarely encountered in equalisation (cf section 2.3.7).

The algorithm noise of the lattice stage will add to that supplied from the previous stages and grow as it propagates down the lattice. Since the PARCOR coefficient is arrived at by a cross-correlation, the increase in uncorrelated noise will move it from its optimum value towards zero. Moreover, later stages will perceive a noisier channel, as will lattice and combiner stages fed by them, and this will in turn affect the target values towards which later stages converge.

In the first approximation, the algorithm noise generated by the lattice stage will be uncorrelated with the noise in the other channel, or with the signal in either channel [46]. Since the noise is not coherent, it will be increased rather than decreased by the next stage:

$$\sigma^2(n) = \sigma^2(n-1).\ (1 + |K(n)|^2) \quad (5.25)$$

Thus although the total signal power is being decreased according to equation 5.22, the noise power is increased in approximately the same ratio. The above equation will not apply to noise from earlier

stages, as this will have been coloured by passing through a lattice stage. Nevertheless, the same principle holds, that the lattice structure amplifies its own algorithm noise while attenuating the input signal.

The balance of rate of convergence with algorithm noise is a special problem in gradient lattices and other distributed gradient structures. From equation 5.23 there is a point of instability when the stepsize is equal to (2/Q(n-1)). The stepsize giving most rapid convergence is half this value [7], giving an algorithm noise power equal to the optimum error. Now in global algorithms, such as that used with the adaptive transversal filter, the optimum error is often many tens of decibels below the filter output level, and the extra 3 dB will make little difference to the filtered output signal. In a distributed structure, however, the signal called 'error' is actually the input to the next distributed stage, and a signal to noise ratio of unity is usually undesirable. Thus the stepsize which gives the highest ROC in a gradient lattice gives an unacceptably high algorithm noise and the system designer must pay attention to the noise/ROC tradeoff.

Figure 5.5 is a graph comparing the theoretically predicted noise power output of a lattice structure with an experimental simulation result. The solid line is the simulation result and the dotted line the result predicted according to equation 5.24. A real PN code of unity power was fed into a gradient adaptive lattice structure and the noise power was measured after 8, 11, 16, 22 and 32 stages. Stepsizes of 0.03125 and 0.000970 were used. The result shows that for low stepsizes the theory holds to within 0.5 dB, but there is a 2-3dB underestimate of the noise for the high stepsize. The reason for this discrepancy may lie in second order correlation effects which become significant at high noise levels. However, these large stepsizes are unlikely to be used in a practical equaliser design.

5.2.4 Algorithm Noise in Lattice Equalisers

The discussion of the poor noise properties of the distributed linear combiner in section 3.2.4 and that of the lattice in the

previous section would suggest that algorithm noise is a serious problem for the gradient adaptive lattice. This is indeed the case. Figure 5.6 shows a graph of converged error plotted against stepsize for transversal and lattice gradient adaptive equalisers. In this set of simulations the lattice did not use a stepsize inversely proportional to the stage power estimate, as in the equation of table 5A. Instead, a global fixed stepsize was used, as is more usual in transversal algorithms. This removed the fast-convergence property of the lattice and gave it a ROC comparable with that of the transversal filter. The graph clearly shows that for low stepsizes the error powers of both algorithms tend towards the optimum value for the channel. The lattice, however is clearly inferior when large stepsizes are used. The very reason for using the lattice algorithm, which is more computationally expensive than the transversal, is that it converges more rapidly on certain channels. Since rapid convergence implies large stepsizes, its poor noise performance is a definite disadvantage.

The converged error of an equalised channel of low distortion may be calculated to give a figure which is not acceptable for most modem applications. Two assumptions are made: That the impulse response of the equaliser is symmetrical about a main centre tap and that the lattice PARCOR coefficients are small, leaving the sidetap coefficients approximately symmetrical. Since the increase in noise of each stage is small, it may be assumed that each stage produces an equal small noise power, σ, which is summed down the lattice. The noise contribution of the main tap is thus:

$$\text{Noise } (N/2) = G^2(N/2).(\sigma^2.(N/2)) \tag{5.26}$$

The noise contribution of any two symmetrically placed taps is:

$$\text{Noise } (N/2 + M) + \text{Noise } (N/2 - M) = G^2(N/2 + M).\sigma^2(N/2 + M) + G^2(N/2 - M).\sigma^2(N/2 - M) \tag{5.27}$$

$$= 2.G^2(N/2 + M).\sigma^2(N/2) \tag{5.28}$$

Thus, summing the noise contribution of all of the taps and assuming that both training signal and tap input have unity power:

$$E \; < d^2(t) > \; = \; \sum_i G^2(i) \; = \; 1 \qquad (5.29)$$

$$\text{Noise (total)} \; = \; N\sigma^2/2 \qquad (5.30)$$

It is assumed that the stepsize for fastest transversal convergence has been used, causing the combiner structure to converge at the ROC of a transversal filter on a well-conditioned line. substituting equation 5.30 into equation 5.24 with the PARCOR coefficients assumed close to zero gives:

$$\text{Noise (total)} \; = \; (1 - 1/(2\sqrt{2}\,N))^{-N/2} - 1 \qquad (5.31)$$

$$\text{as } N \to \infty, \; \text{Noise (total)} \to \; \exp(1/(4\sqrt{2})) - 1 \qquad (5.32)$$

Thus the signal driving the taps of an 8-tap equaliser would have a noise level of -7.0 dB and for an infinite number of taps the result would be -7.1 dB. One further decibel may be subtracted from the noise level to take into account the effect of the noise on the optimum values of the taps (cf section 3.2). Thus the calculated noise level is -8 dB as against simulation results of -10.2 dB. The 2 dB discrepancy is due to second order effects which were not accounted for in the simplified theory. Nevertheless, this mathematical description shows that the noise performance of the lattice equaliser is poor, no matter what length of filter is used. The value of -10.2 dB was found in further simulation results to be largely independent of channel conditioning, since most equaliser PARCOR coefficient values are close to zero in normal modem equalisation applications anyway.

Figure 5.7 illustrates the fact that the lattice noise floor is almost independent of filter length. This is a series of convergence curves for transversal and lattice equalisers on a channel of eigenvalue ratio 8. The algorithm is the same as that used for figure 5.6, in that a global stepsize, that of the transversal equaliser, was used for all stages. Nevertheless, the figure clearly shows that while

transversal equalisers reduce their converged error with increasing length, lattices are limited by the fixed error threshold due to algorithm noise.

5.3 Simulations

This section presents a number of different simulations of various lattice algorithms and equaliser configurations. The conventional algorithms are presented first, showing that algorithm noise is a serious problem. Algorithms are then presented which minimise this problem while retaining the advantage of the lattice orthogonalising structure.

5.3.1 Simulations of the Conventional Lattice Algorithm

Figure 5.8 shows the response of 8-tap gradient lattice and transversal equalisers converging on a real synthetic channel. The standard test conditions are given in table 5B. The transversal adaptive filter uses the optimum stepsize, which gives it fastest convergence. The lattice algorithm used a stepsize of half this in an attempt to reduce algorithm noise.

The result shown is representative of results obtained throughout the simulation work on lattice filters: The transversal filter converges more slowly due to tapweight interactions, but nevertheless reaches a lower converged error. Were the lattice's stepsize to be reduced so as to produce the same converged error, it would converge many times more slowly than the transversal. Moreover, on channels of reduced eigenvalue ratio, the lattice loses even its speed advantage, becoming both slower and noisier than the transversal.

A more encouraging result was obtained by starting off the algorithm with stepsize values close to unity, which is much higher than the steady-state values. This will be referred to as a 'high start' technique throughout the thesis. Figure 5.9 shows that this gives the distributed algorithm an initially rapid convergence (accompanied by high noise). After a while the stepsize has reduced to its steady-state value (cf section 3.2.3) to minimise the

algorithm noise. The result is more promising, showing that the lattice can converge up to three times faster if a degradation in steady-state algorithm noise is permitted, or twice as fast for the same output error power (figure 5.9b).

Unfortunately, these were the best possible conditions for the lattice. The optimum converged error of the channel helped to mask the algorithm noise and the high eigenvalue ratio slowed the transversal down greatly. These results could have been artificially improved by adding noise to the channel to further mask the different converged errors of the filters. Figure 5.9c shows a more representative picture of convergence of 16-tap equalisers onto an ill-conditioned complex channel. Five different steady-state stepsizes have been tried, showing the trade-off between ROC and algorithm noise. The result again shows that the lattice can converge faster, at the expense of increased converged error.

5.3.2 Variations on Lattice Algorithms

In view of the poor noise performance of the lattice and distributed combiner structures, some variations of the algorithm design were attempted. The object was to minimise the algorithm noise and yet retain the reliable convergence properties of the lattice. The high start feature was regarded as desirable and was thus retained as far as possible.

In view of the poor noise performance of the distributed linear combiner, it was replaced by the global combiner algorithm. The lattice stepsize estimator was retained to normalise the rate of convergence of the taps. The result was inferior to the performances of the previous section. The algorithm's poor converged error was probably due to the random fluctuation in tap signal statistics, caused by the algorithm noise of the lattice structure. Having less tapweight interaction, the order-dependent distributed combiner (cf section 3.2.2) tracked these fluctuations and performed better under these conditions.

A separate attempt was made, retaining the distributed combiner

but freezing the lattice PARCOR adaption after 15-30 iterations. It was assumed that as the lattice would be close to convergence, there would be a far smaller degree of correlation between the signals feeding the combiner taps than at the outset of convergence. The idea did not work as well as the algorithms in the previous section because the distributed combiner only tends towards an optimal solution if the tap signals are truly orthogonal. It was found that the lattice had not operated for long enough to ensure this orthogonality condition. Algorithms in which convergence is stopped after a few iterations will be referred to as "timed" algorithms.

The most successful approach was a combination of the two previous ones. The lattice was run for 15-30 iterations in order to improve the eigenvalue ratio of the cross-covariance matrix of the tap signals. A global combiner then used these signals to converge to the extremely low error powers that such a combiner can reach.

Figure 5.10a shows this approach when using a stepsize only half that which leads to fastest convergence in the transversal. Not only does it converge faster than the transversal equaliser (on the badly conditioned 8-tap channel) but it achieves a lower converged error. This is the most promising result to date. Figure 5.10b shows a 16-tap complex simulation on channels of low and high distortion. Again, convergence is at the rate of the faster transversal filter and the converged error is close to the optimum. Figure 5.10c shows a similar test on 22-tap channels, showing that the timed lattice is dependent on channel conditioning, but less so than the transversal algorithm. This demonstrates the fact that tap-signal interactions are minimised but not eliminated. The transversal filter performance is clearly better on the well-conditioned channel.

5.3.3 Other Related Configurations

A further variation of the lattice algorithm was suggested, which used a recursion involving only one correlation [78]. With reference to equation 5.17, the new recursion was:

$$K(n+1, t+1) = K(n+1, t) + 2.\mu(n,t).b^*(n,t).f(N,t) \qquad (5.34)$$

Where f(N,t) is the output of the final stage of the lattice (cf figure 5.11). Being a one-correlation recursion, it converges as rapidly as that of equation 3.41, but on well-conditioned channels its algorithm noise is worse by a factor of 2. On badly conditioned channels, however, the power output from the final stage drops as the PARCOR coefficient values are displaced from zero. This, in theory, should reduce the algorithm noise. In fact, as figure 5.12 shows, results were similar to those of the high-start algorithm of section 5.3.1, even on badly conditioned channels. The algorithm, originally conceived for LPC work, did not prove suitable for this channel equalisation application.

One idea that did help greatly was that of placing a single adaptive lattice stage in front of a conventional transversal adaptive filter, using a recursive stepsize algorithm in order to maintain optimum ROC. The theory behind this configuration, illustrated in figure 5.13, is that the lattice stage whitens the signal (cf section 2.3.7), improving the eigenvalue ratio of the equaliser input signal. Unfortunately, although this does indeed lead to a faster ROC, it has the effect of spreading the equaliser input signal's impulse response. This in turn increases its converged error (cf section 4.3). For this reason the most successful results were obtained using only one lattice stage, whose convergence was stopped after 15-30 iterations to reduce algorithm noise.

Figure 5.14 shows the results of the simulations using this pre-whitener. These are respectively, a real 8-tap simulation, complex 16-tap simulations on well and badly conditioned channels and again with a 22-tap complex simulation. They show that although the whitener does not completely recondition the channel, it nevertheless improves the convergence on ill-conditioned channels. It is a cheap additional circuit which will speed up the convergence of existing modem designs.

5.4 Summary

This chapter has examined the rate of convergence and converged error of the lattice and transversal gradient equaliser algorithms. It has clearly shown how the selection of the stepsize parameter involves consideration of the conflicting requirements of fast convergence and low converged error, i.e. minimisation of algorithm noise.

It may be possible to take advantage of the reliable convergence characteristics of the lattice to optimally vary the stepsize according to equation 5.5. This would certainly improve the performance of the gradient lattice at least to the level of the transversal. However, the increase in complexity of the algorithm would make it less competitive with even more efficient ones, such as the exact least squares lattice [47].

Of those lattice algorithms examined, the most promising was the timed lattice with the global sidetap recursion, described in section 5.3.2. With very little variation in rate of convergence, it easily reaches the low converged error values of the transversal equaliser.

$$K(1 \ldots N, 0) = 0$$
$$b(1 \ldots N, 0) = 0$$
$$G(0 \ldots N, 0) = 0$$

$$f(0,t) = b(0,t) = s(t)$$
$$e(0,t) = d(t)$$
$$y(0,t) = 0$$

$$f(n+1, t) = f(n,t) - K(n+1, t).b(n, t-1)$$
$$b(n+1, t) = b(n,t-1) - K^*(n+1, t).f(n,t)$$
$$e(n+1, t) = e(n,t) - G(n,t).b(n, t-1)$$
$$y(n+1, t) = y(n,t) + G(n,t).b(n, t-1)$$

$$K(n+1, t+1) = K(n+1, t) + \mu(n,t).(f(n+1, t).b^*(n, t-1) + b^*(n+1, t).f(n,t)$$

$$G(n, t+1) = G(n,t) + 2\mu(n,t).e(n+1, t).b^*(n, t-1)$$

$$C(n, t+1) = C(n,t).(1-\alpha) + \alpha.\beta(|f(n,t)|^2 + |b(n, t-1)|^2)$$

$$\mu(n, t+1) = 1/C(n, t+1)$$

Table 5A The algorithm used in lattice simulations

The following sets out the standard set of experimental conditions under which most simulations were performed. Possible deviations of individual simulations from this standard will be given in the test as appropriate.

Test Signal:
2-d PN sequence of length (2^{24} -1) bits and amplitude of ± 1 per channel. (1-d simulations used the real part only)

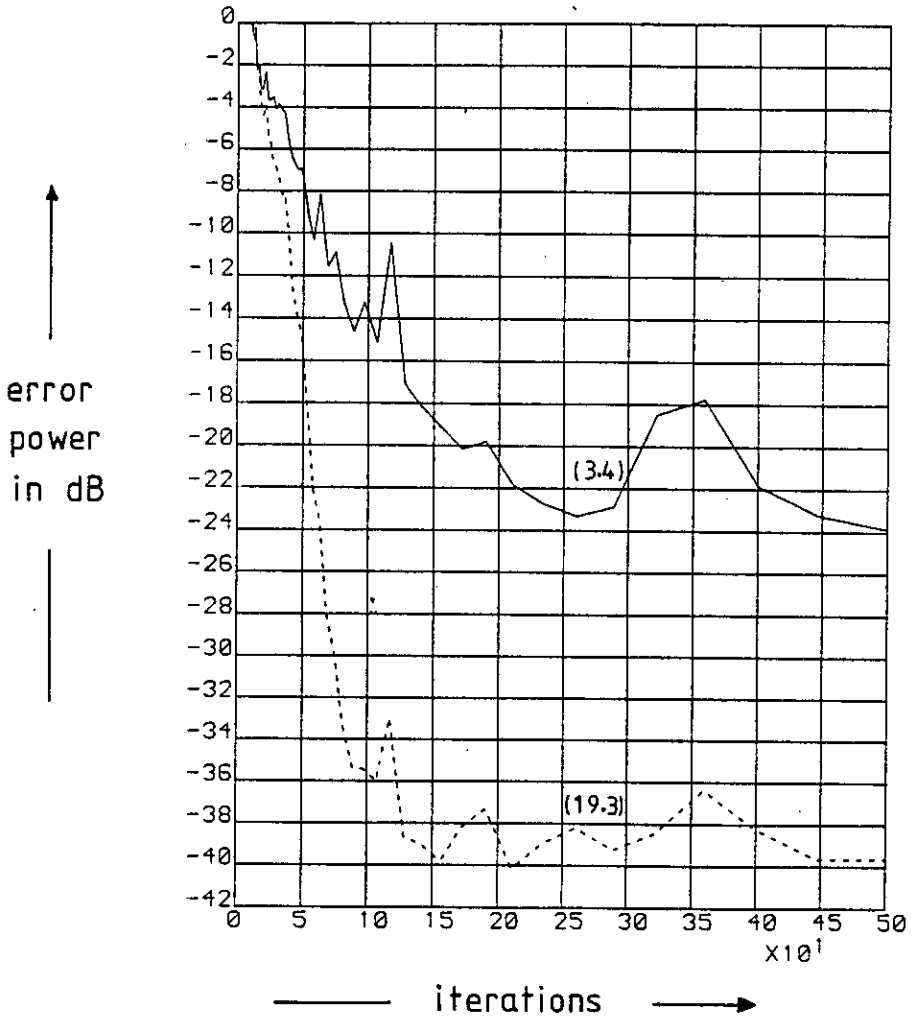
Test channel (cf sections 4.6, 5.3 and appendix D):
8-tap real simulations:
Synthetic channel R81 of EVR $\approx$ 50 and optimum equalised error -24 dB.
16 and 22-tap complex simulations:
Complex BTRL equivalent baseband impulse responses.
E15: EVR 20, optimum error 39/52 dB for 16/22 taps.
E8C3A: EVR 2.7, optimum error 65/94 dB for 16/22 taps.
No simulations used additive channel noise.

Stepsize for combiner structure:
$1/(N \cdot R(0))$, where $R(0)$ is the input power per channel and N is the number of real taps or twice the number of complex taps. This will be referred to as the "optimum stepsize" [7].

Graphs of convergence curves:
Error signal power in dB versus iteration number.
Transversal filter performance shown dotted, lattice shown in solid line.
Error signal power calculated by convolving channel impulse response with equaliser impulse response and calculating off-lobe power (cf section 4.3.2).

Initial tap settings: zero.

Table 5B The experimental conditions under which computer simulations were performed



8 tap filter

normalised minimum eigenvalues of channels: 3.4 (upper)
19.3 (lower)

Fig 5.1 The convergence properties of an LMS adaptive transversal filter

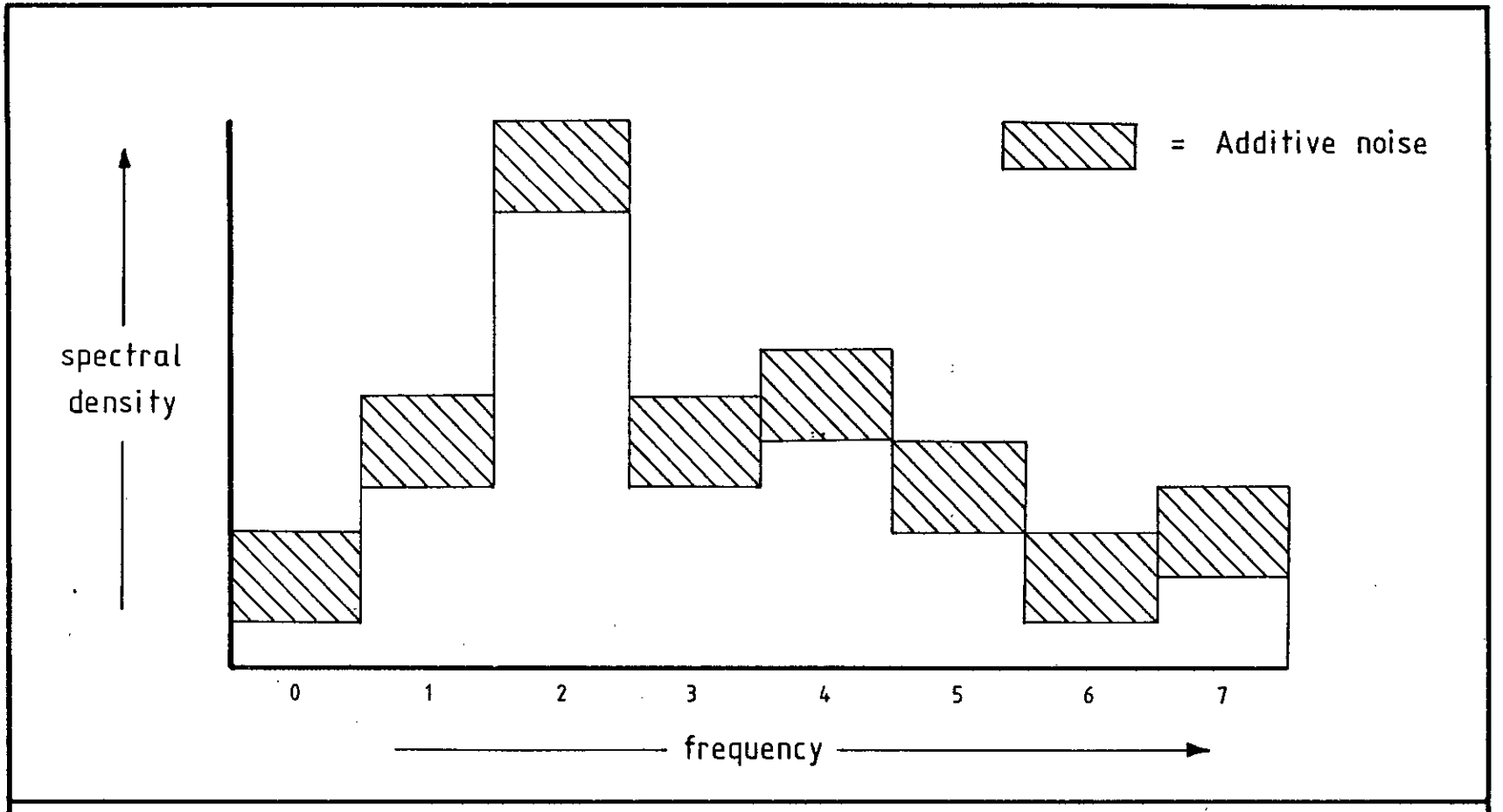


Fig 5.2 Schematic histogram of the power spectrum of a signal before and after the addition of noise

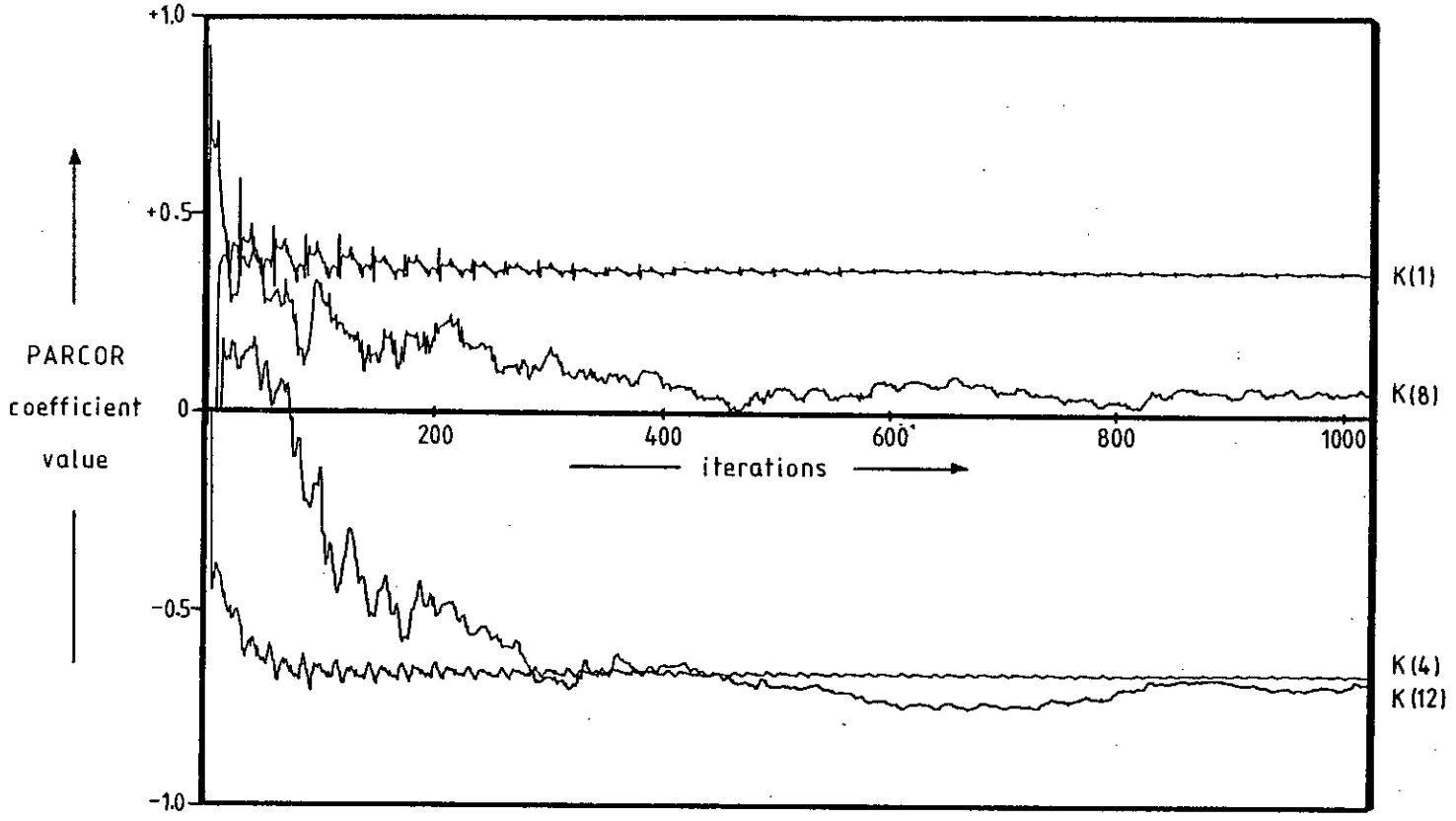
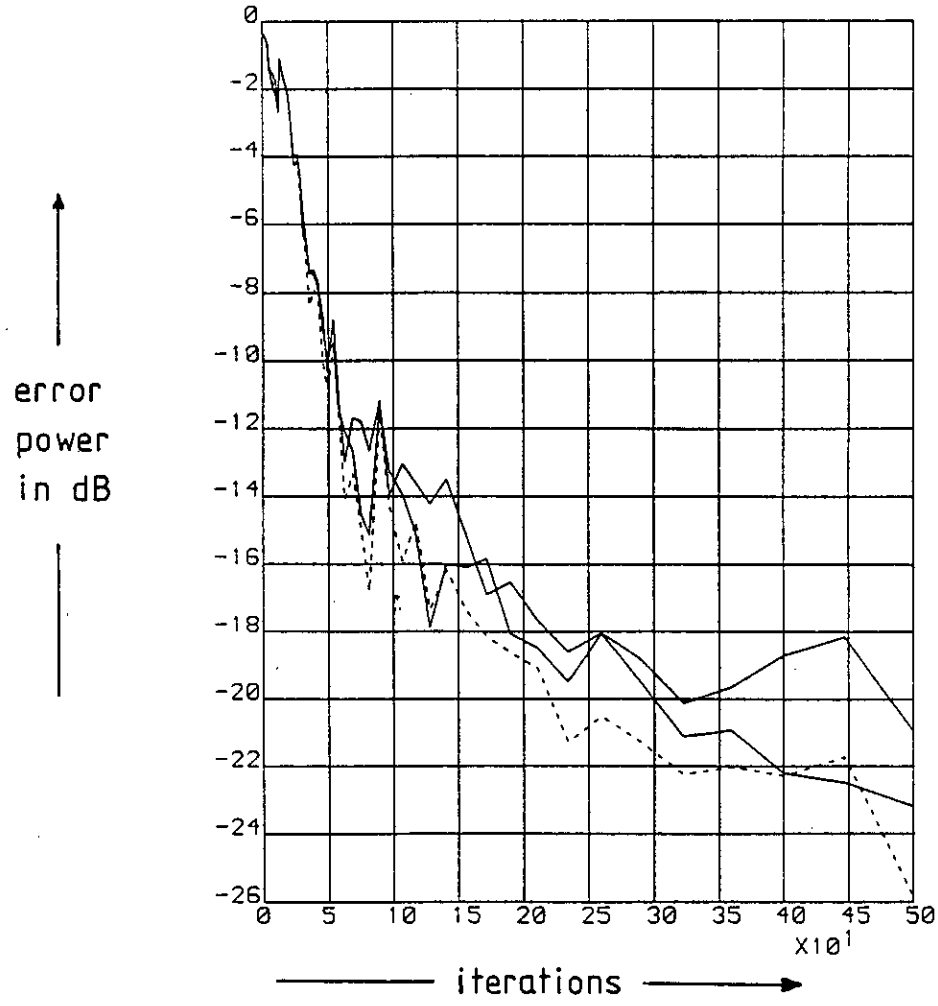
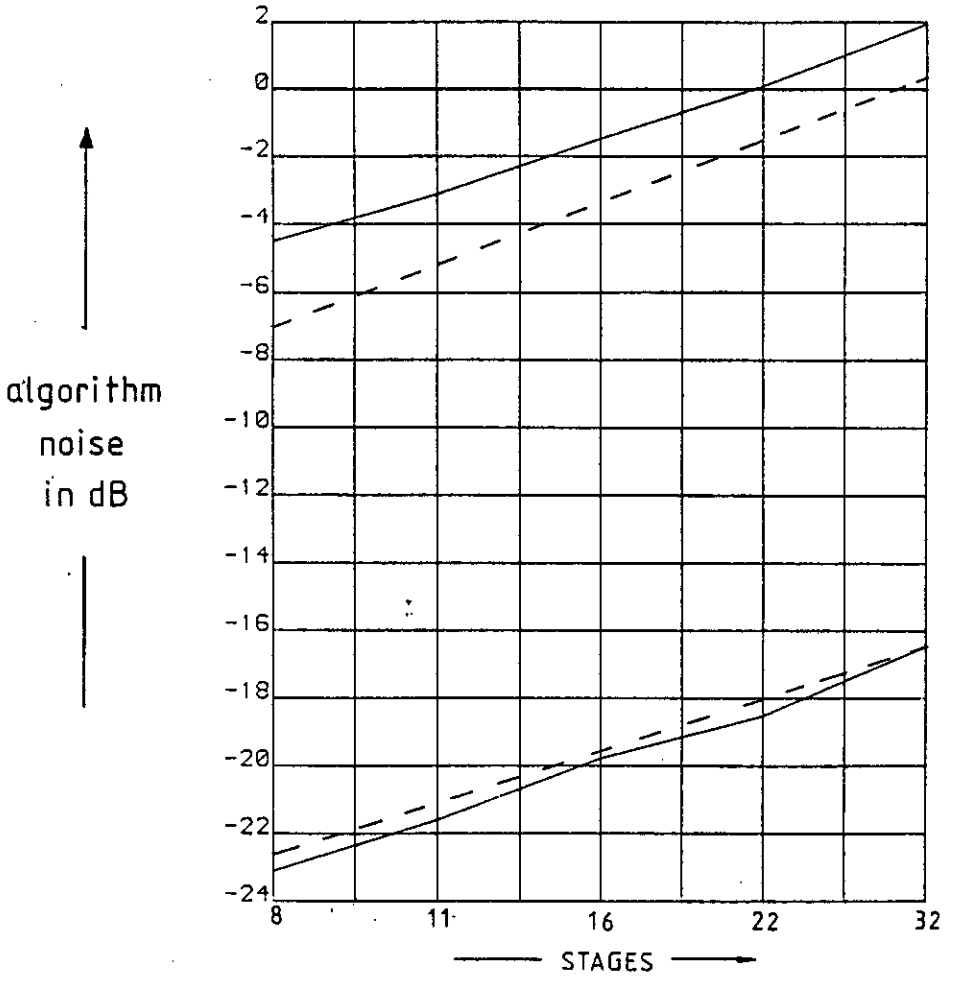


Fig 5.3 The convergence of the coefficients of a lattice structure onto a stationary signal (reducing stepsize employed)



lattice:combiner stepsize ratios: 1.414 solid
2.0 dotted
2.828 solid

Fig 5.4 The convergence properties of a gradient adaptive lattice equaliser with different ratios of stepsize



solid line: simulation result
dashed line: theoretical prediction
upper pair: stepsize 0.03125 / R(0)
lower pair: stepsize 0.00097 / R(0)

Fig 5.5 Algorithm noise generated by various lengths of lattice

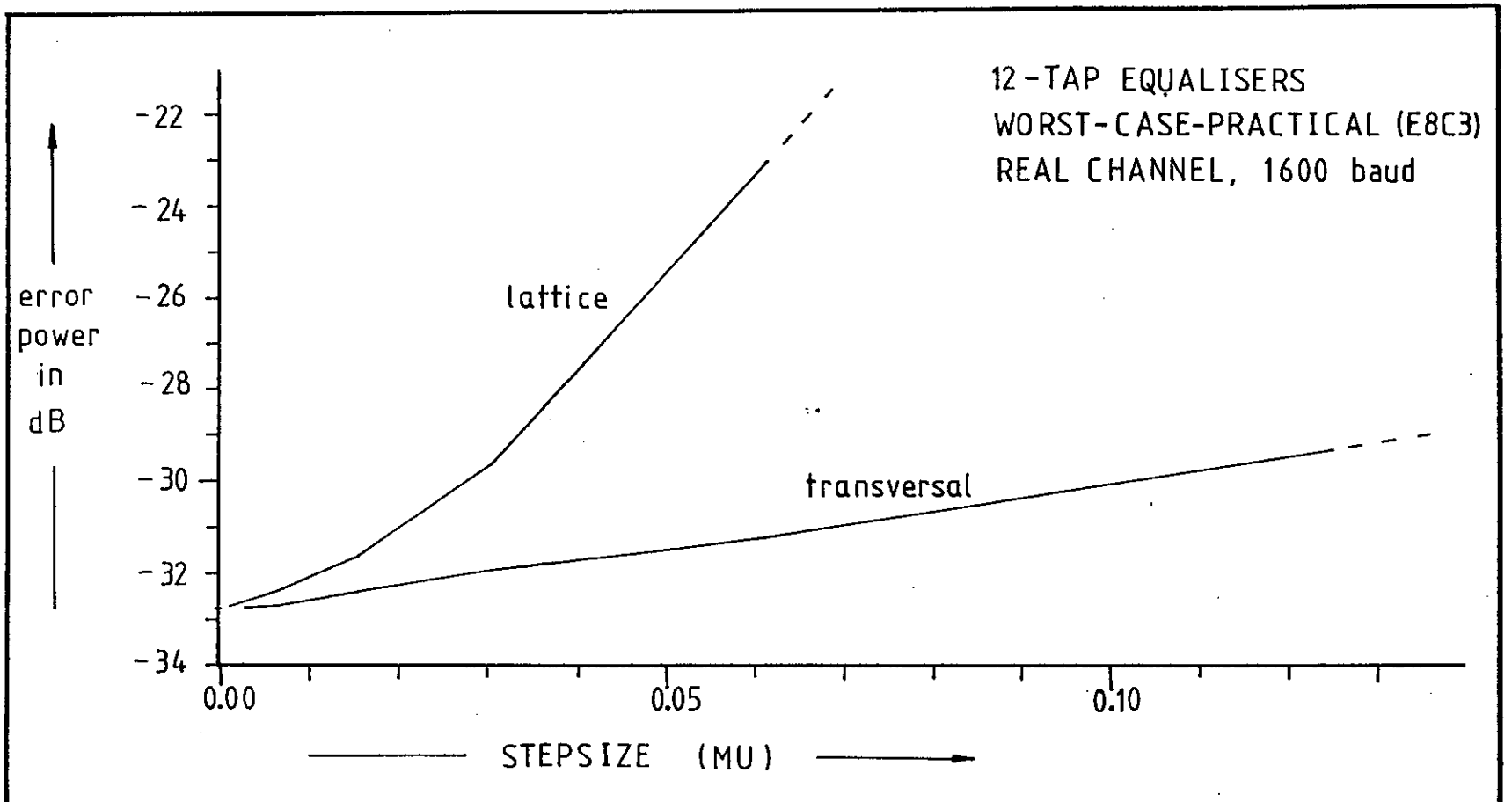
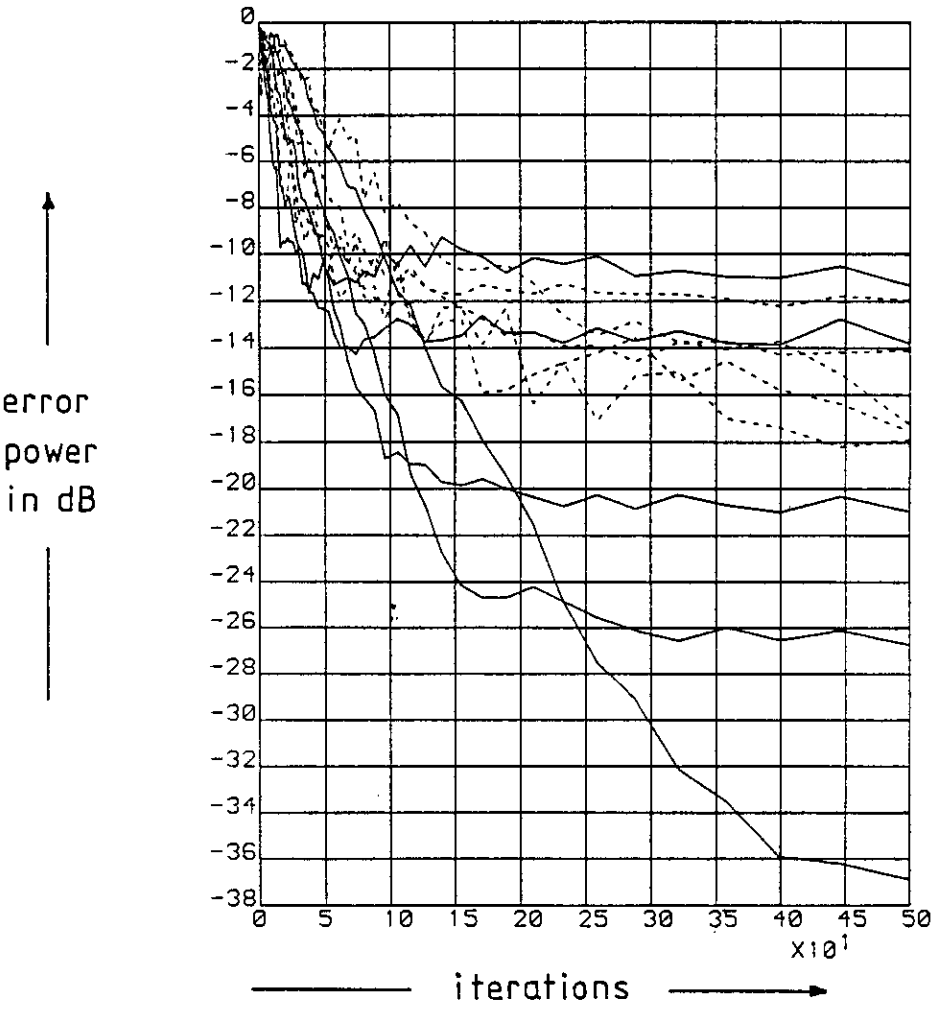
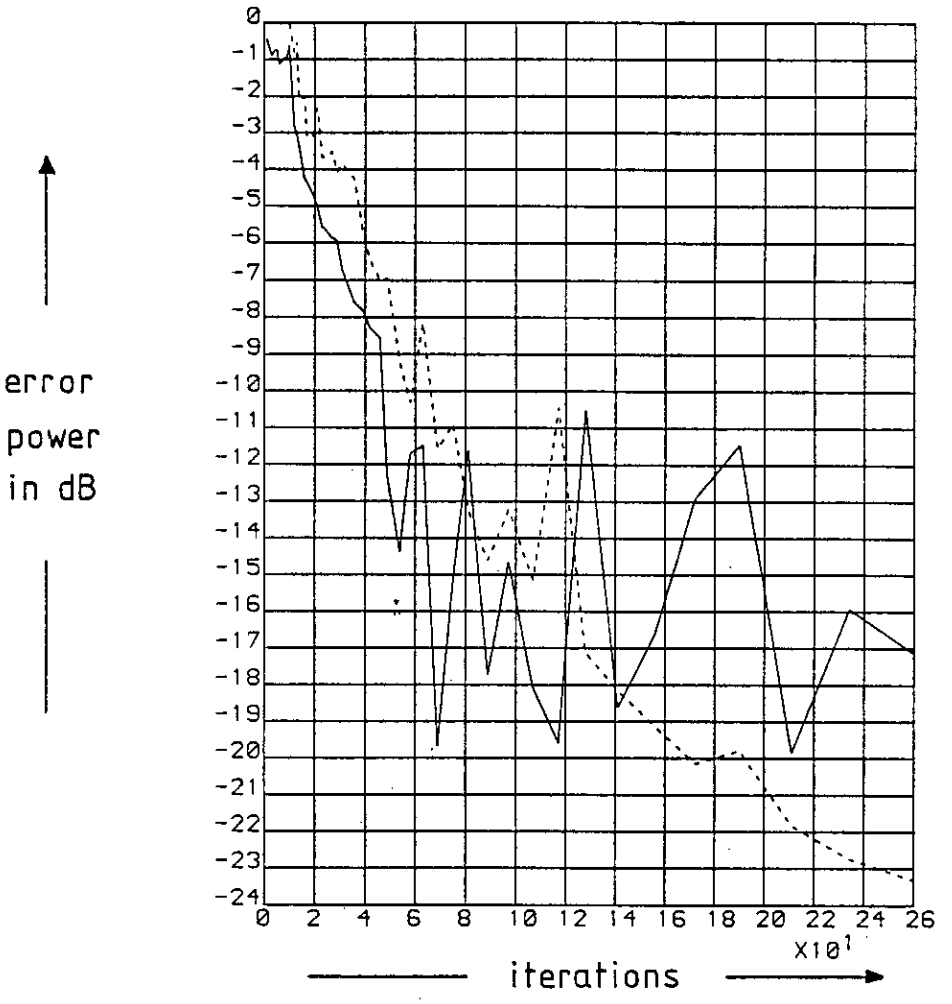


Fig 5.6 Converged error -vs- stepsize for constant-mu lattice and transversal equalisers



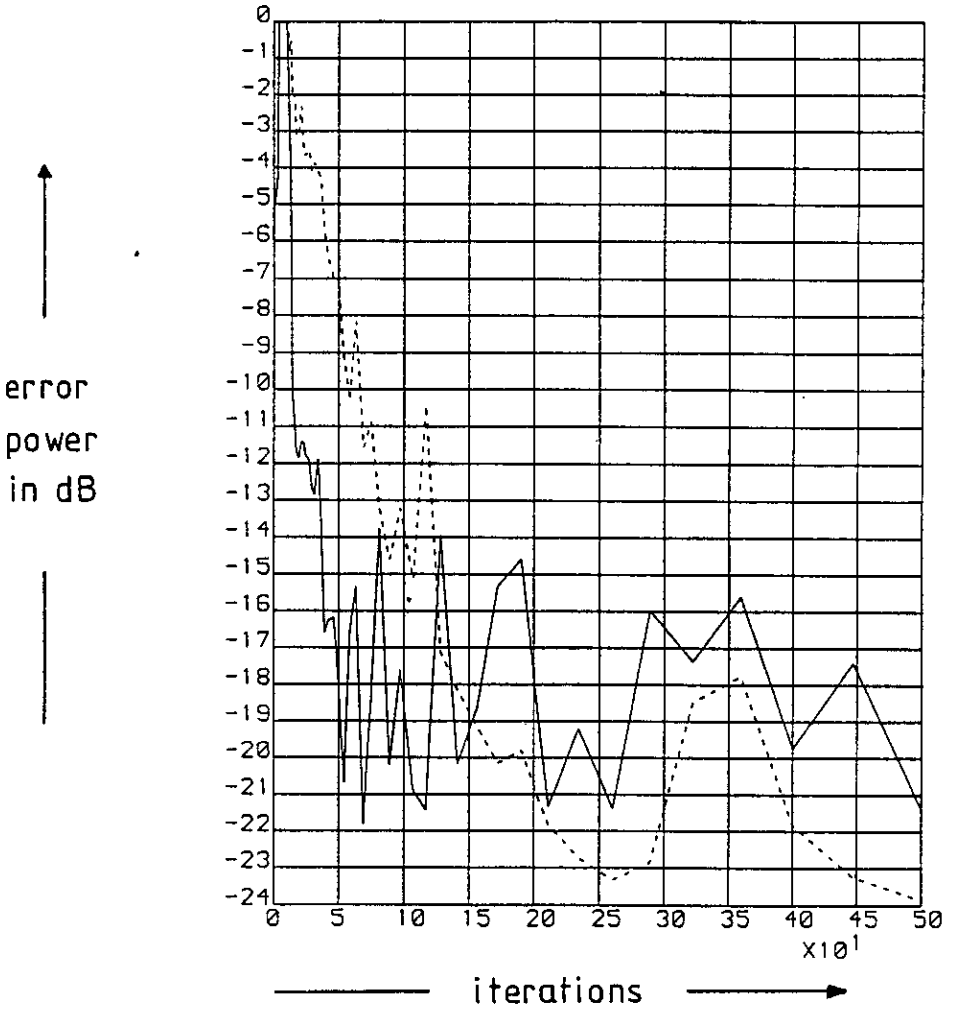
equalisers of 8, 11, 16, 22 & 32 taps
dotted line: lattice
solid line: transversal

Fig 5.7 Lattice and transversal equalisers converging on a channel of low distortion



8 tap equalisers
channel eigenvalue ratio: 49
dotted line: transversal
solid line: lattice

Fig 5.8 Transversal and lattice equalisers converging on a heavily distorted real channel



8 tap equalisers

channel eigenvalue ratio: 49

dotted line: transversal

solid line: lattice using high start and stepsize 0.25 optimum

Fig 5.9a Transversal and lattice equalisers converging on a heavily distorted real channel

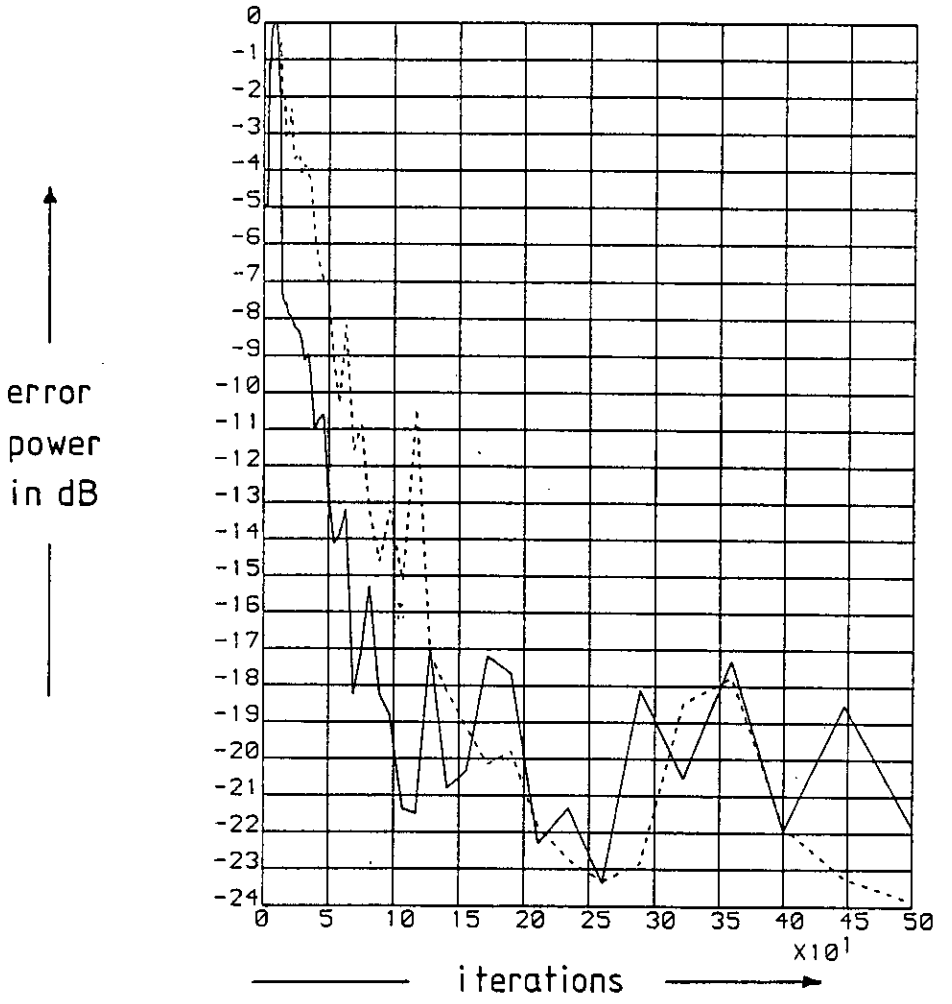
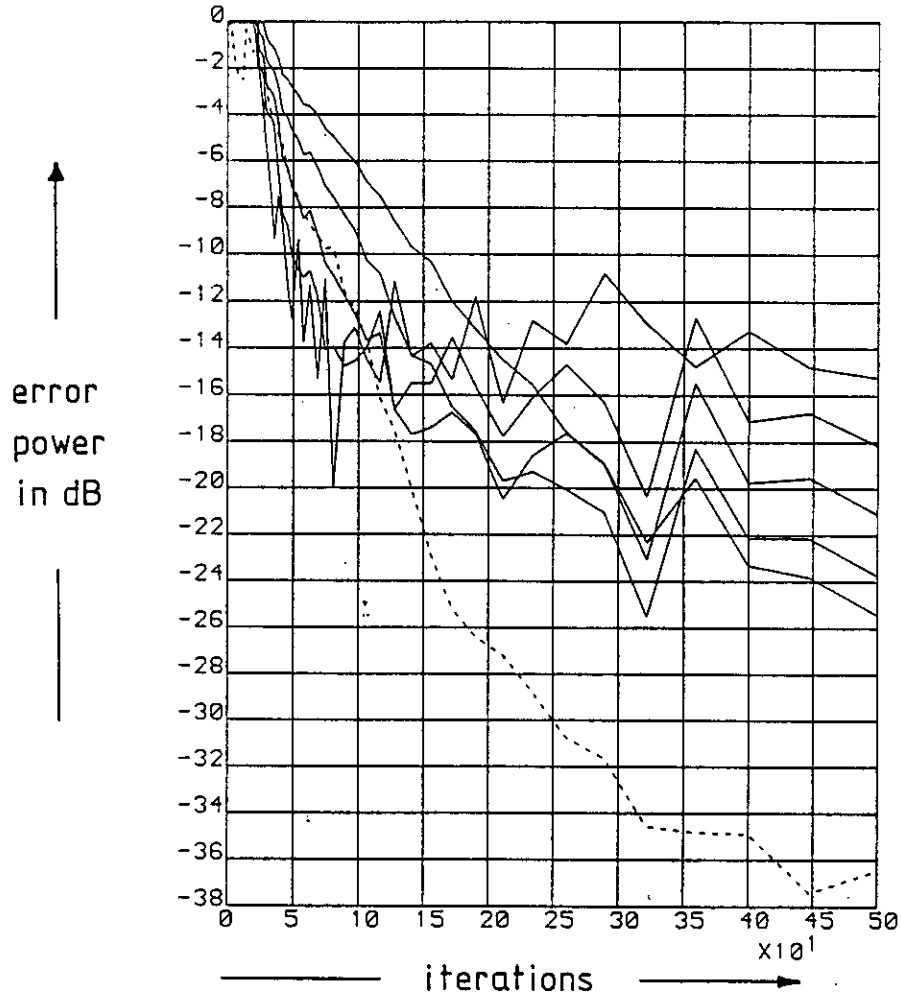
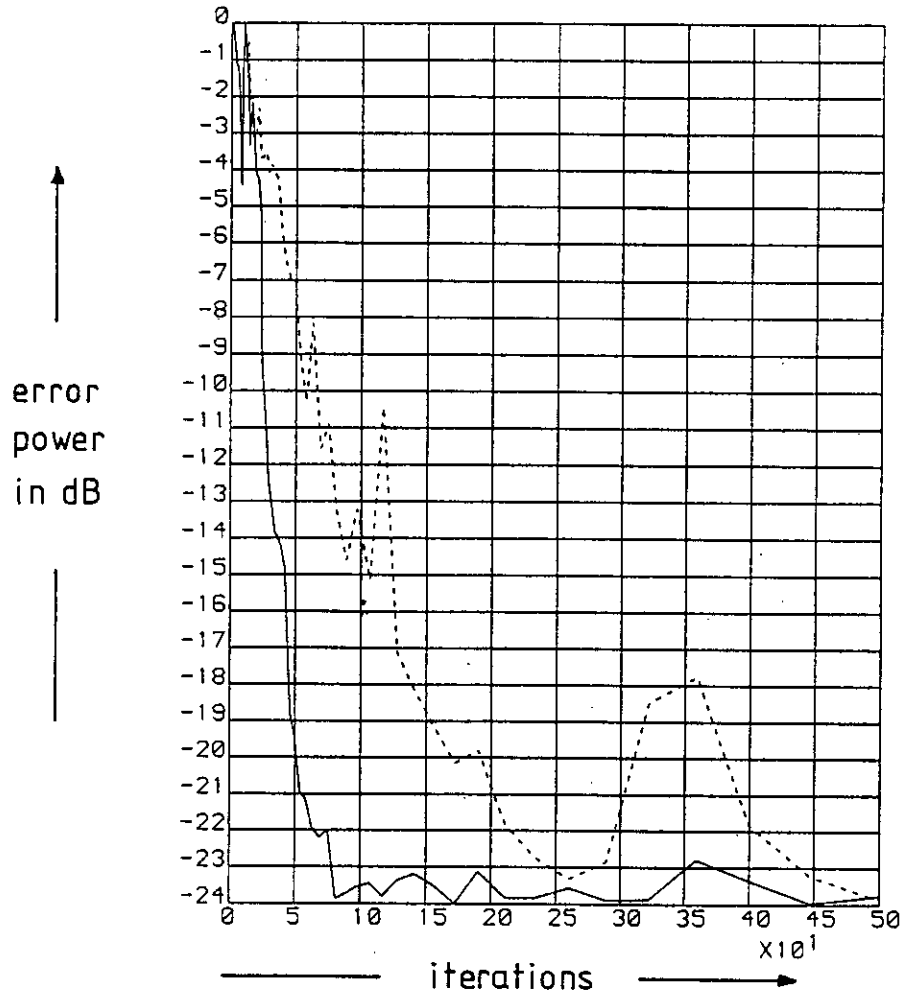


Fig 5.9b As 5.9a, but lattice using a stepsize of 0.35 optimum



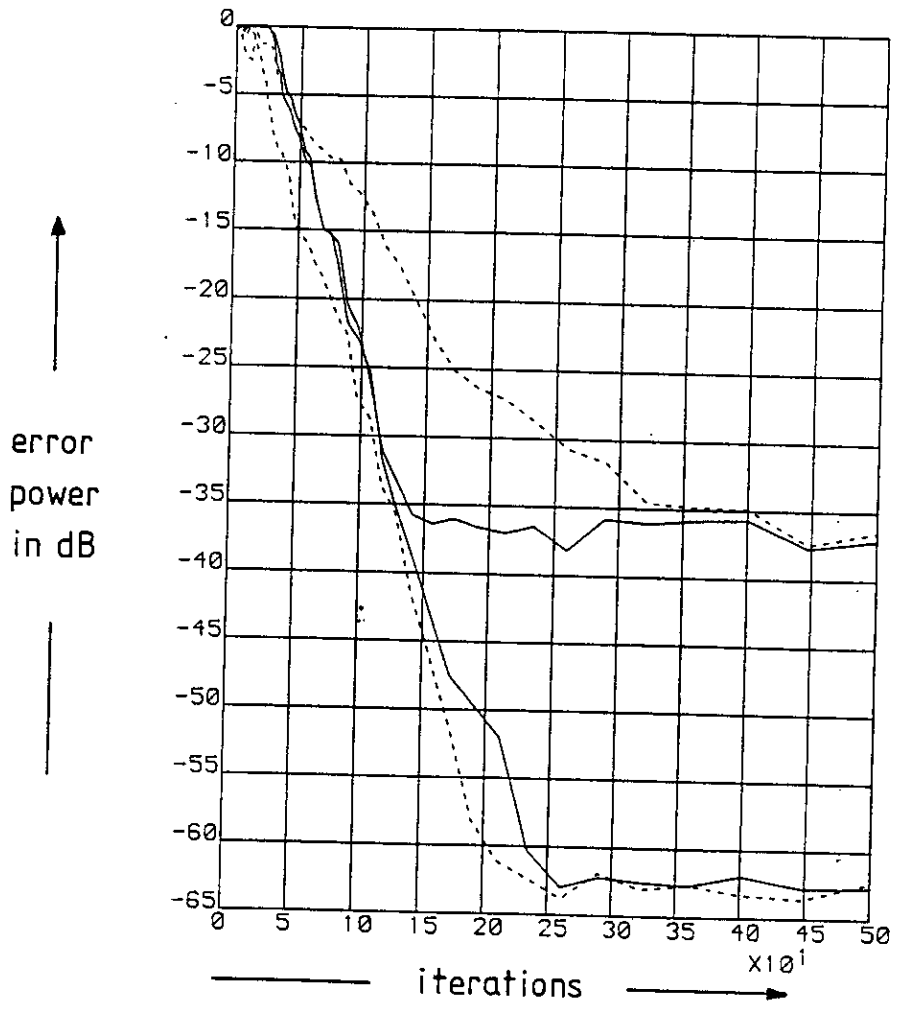
high start used
lattice steady-state stepsizes (x optimum) of:
0·707, 0·5, 0·35, 0·25 & 0·17

Fig 5.9c 16 tap lattice algorithm, converging on a heavily distorted complex channel



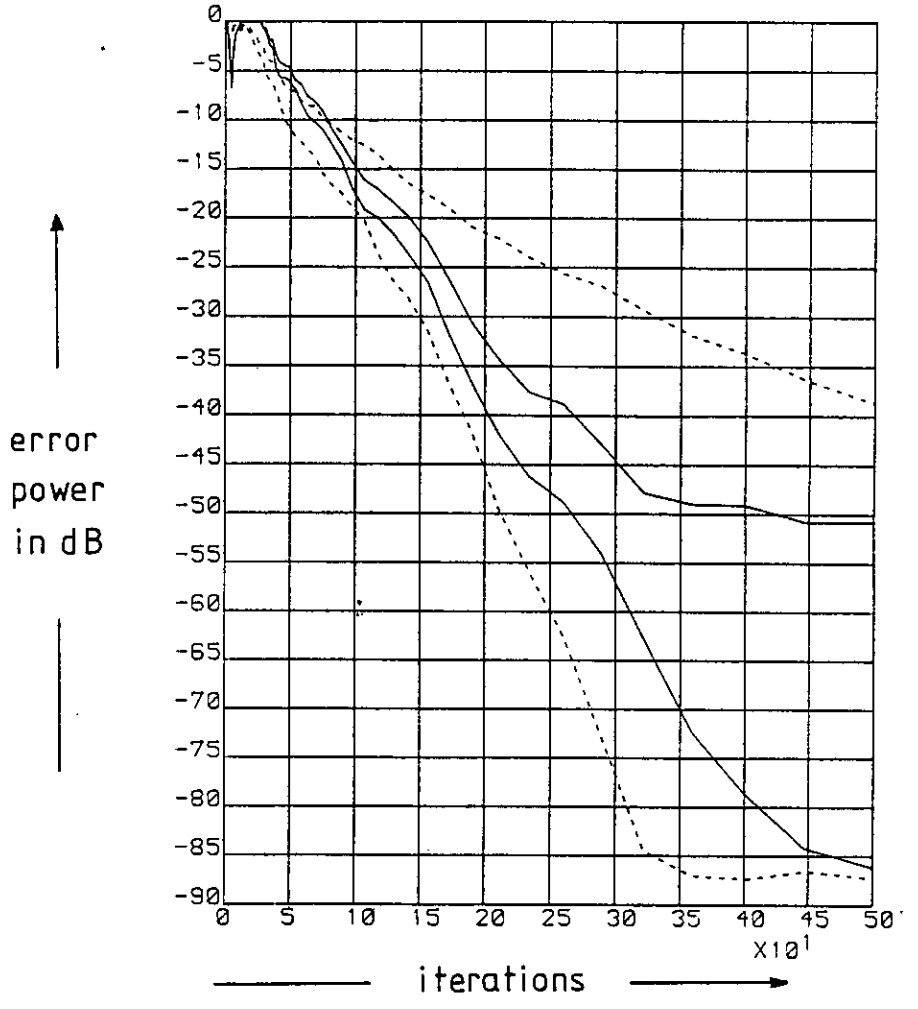
8 tap equalisers on real channel
dotted line: transversal
solid line: lattice

Fig 5.10a Timed lattice with global combiner



dotted line: transversal
solid line: lattice

Fig 5.10b As 5.10a, but using 16 tap complex equalisers on well and badly conditioned channels



dotted line: transversal

solid line: lattice

Fig 5.10c As 5.10a, but with 22 tap complex equalisers

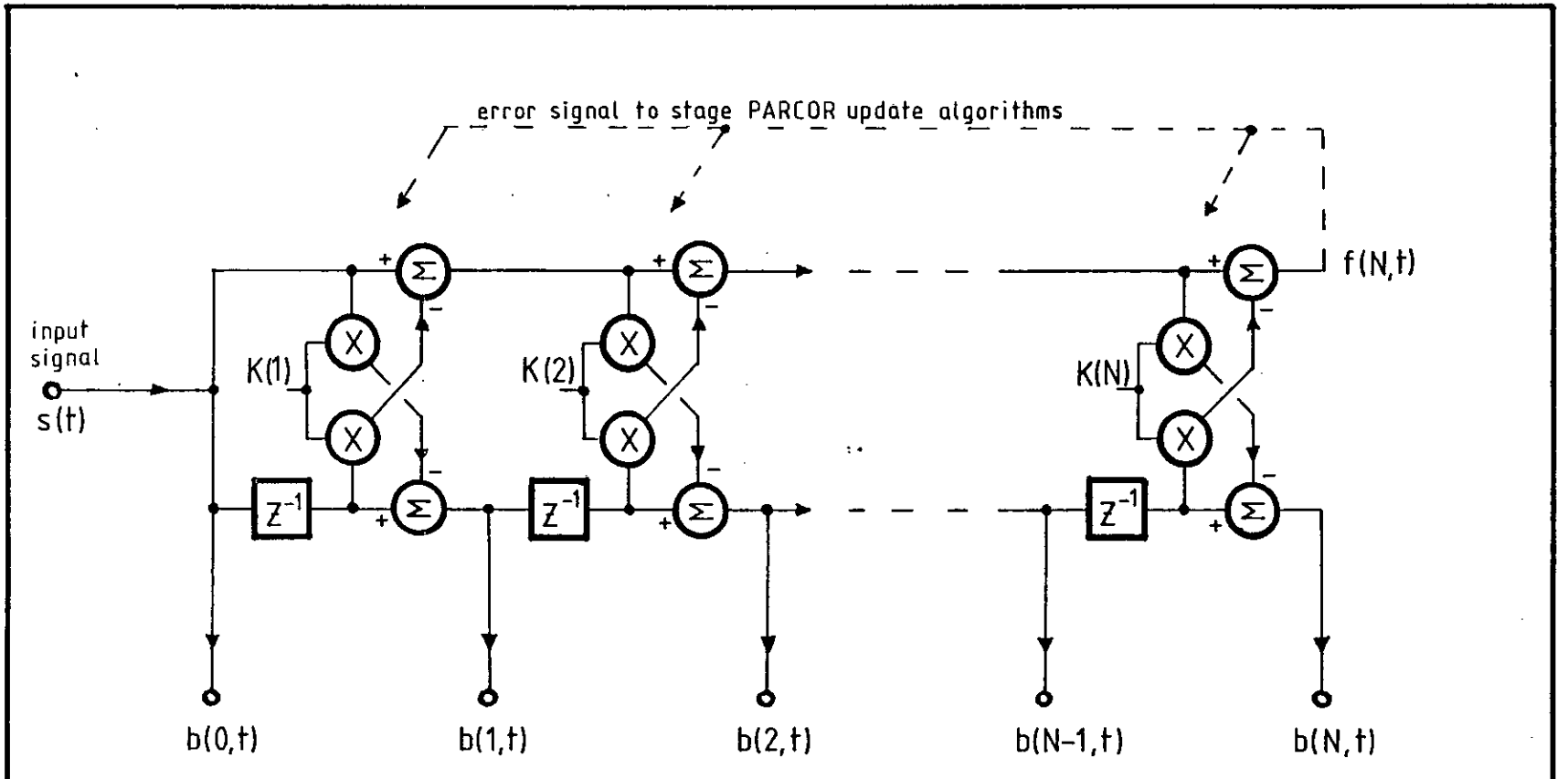
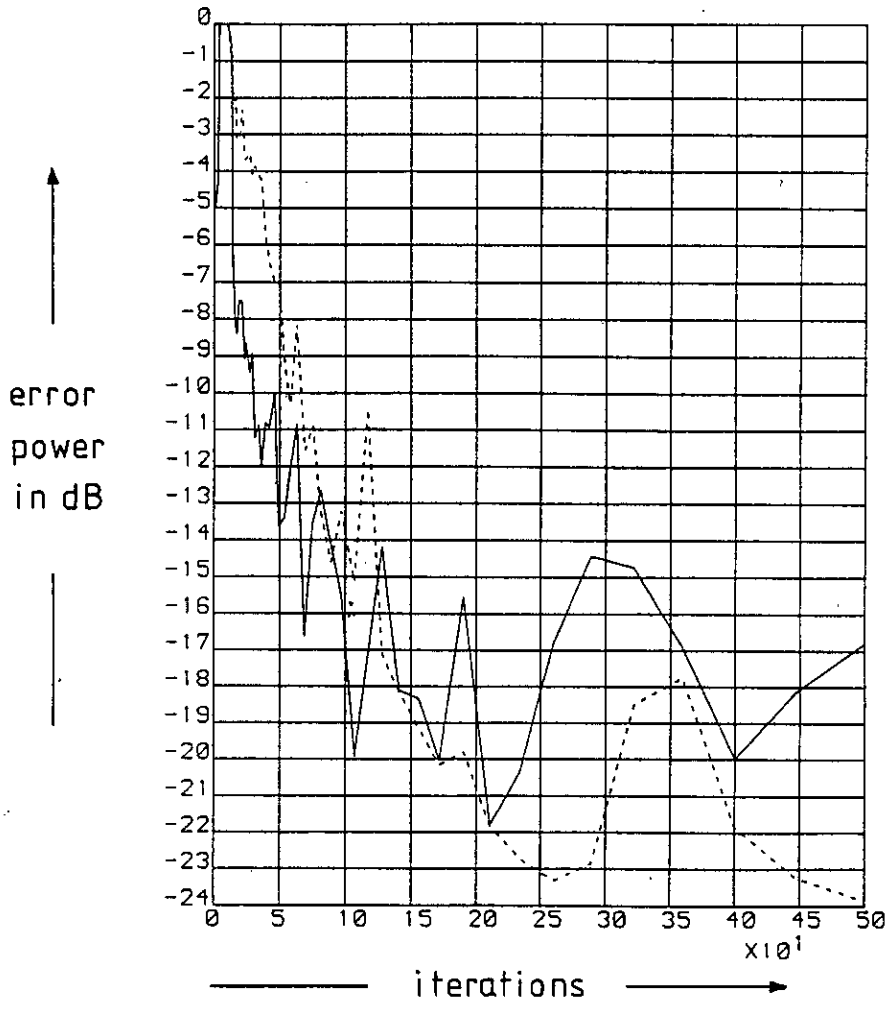


Fig 5.11 The global lattice PARCOR update algorithm



dotted line: transversal

solid line: lattice

Fig 5.12 Results from an 8 tap global lattice algorithm on a real channel

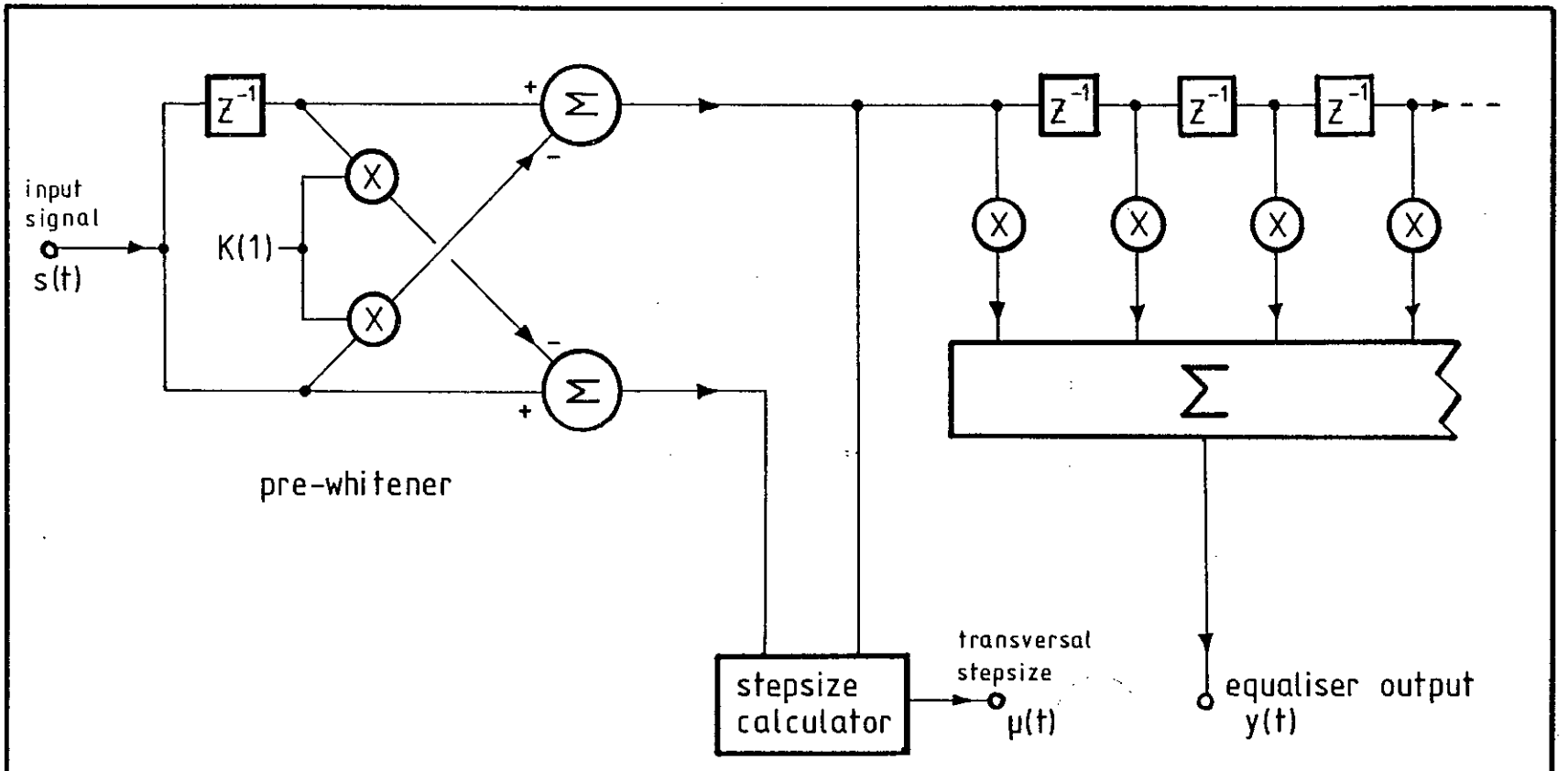
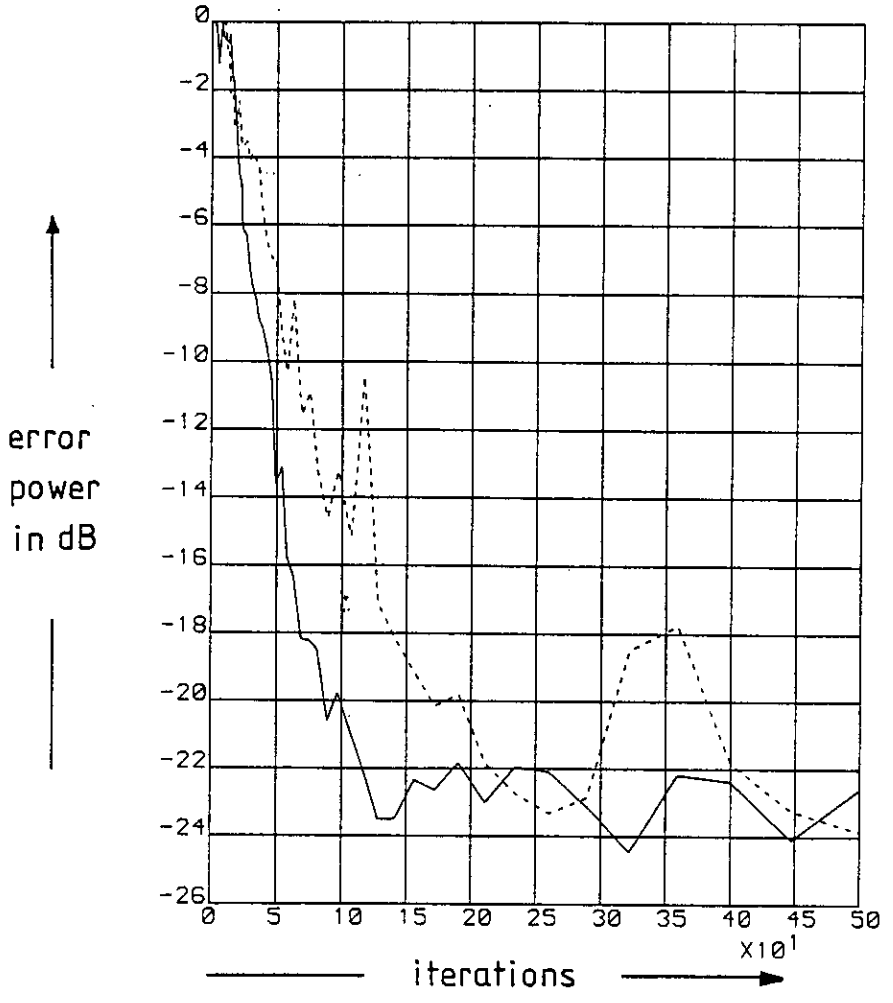


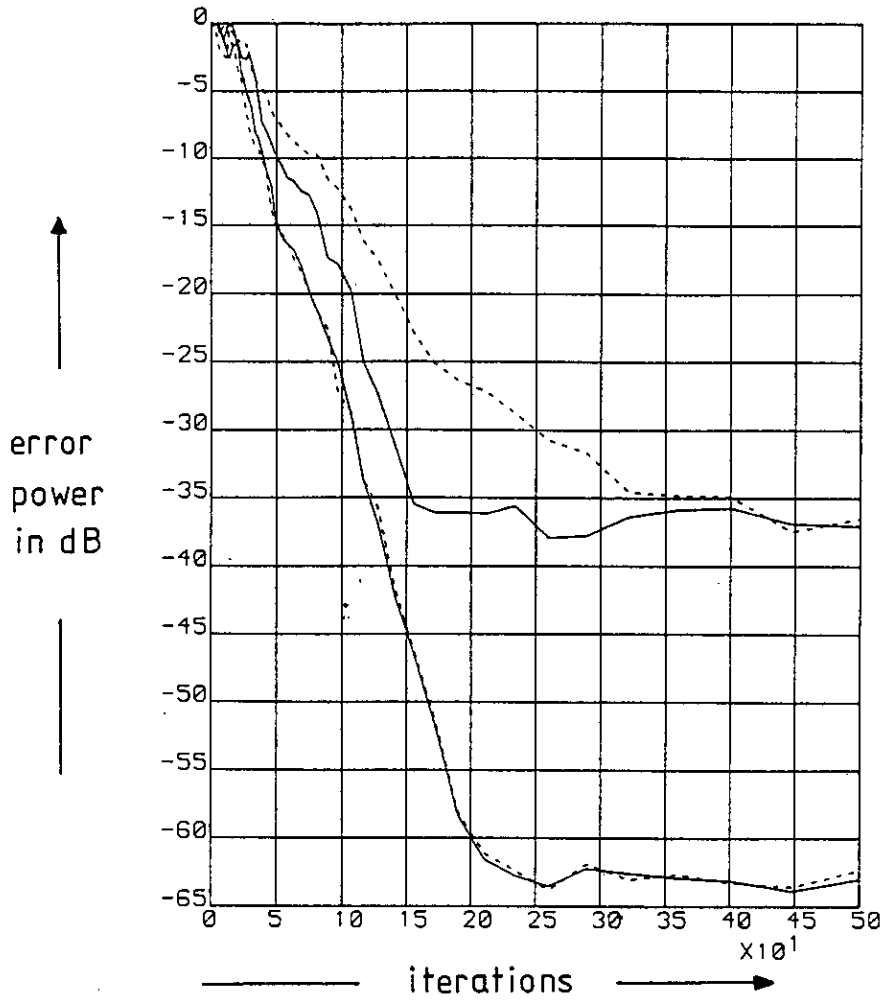
Fig 5.13 A transversal adaptive filter with pre-whitener (adaptive algorithm not shown)



dotted line: normal transversal

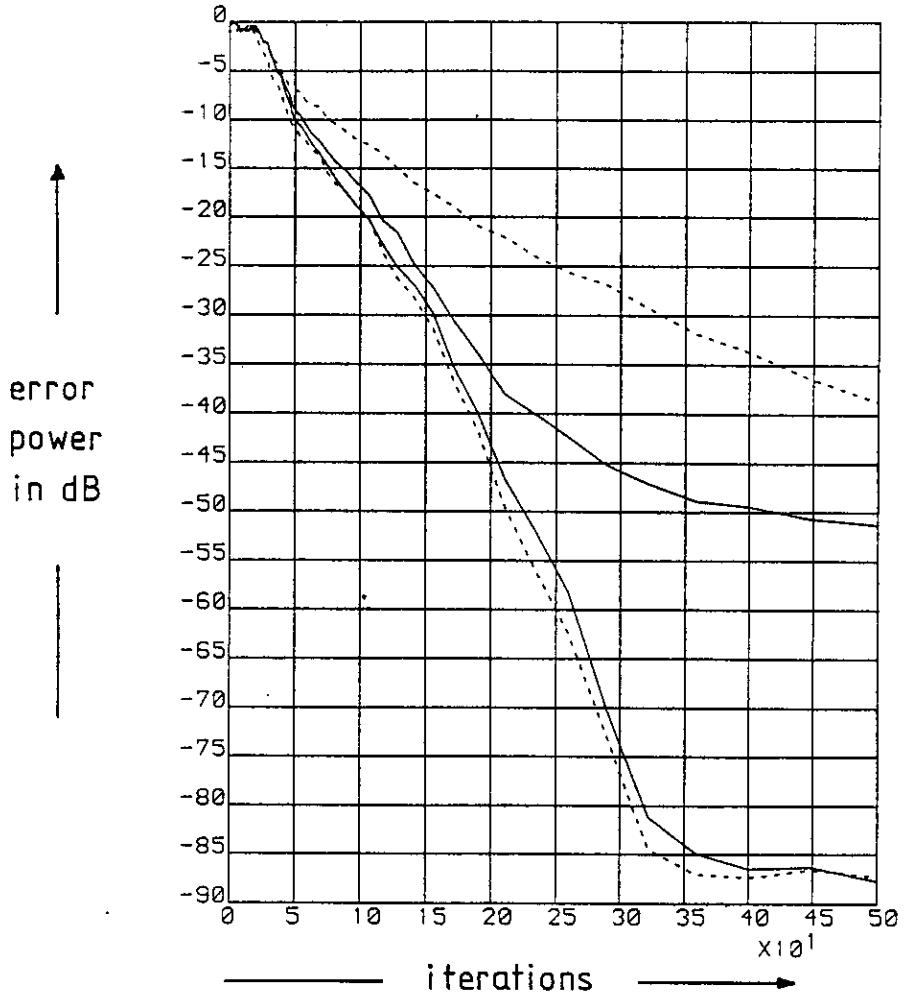
solid line: pre-whitened transversal

Fig 5.14a Convergence of an 8 tap transversal filter with a lattice pre-whitener



dotted line: normal transversal
solid line: pre-whitened transversal

Fig 5.14b As 5.14a, but with a 16 tap equaliser on complex channels



dotted line: normal transversal
solid line: pre-whitened transversal

Fig 5.14c As 5.14a, but using 22 tap equalisers on complex channels

Chapter Six

ADAPTIVE LATTICE DESIGN AND PERFORMANCE EVALUATION

This chapter and the next evaluate various ways of designing adaptive lattice structures and equalisers. This chapter deals with the discrete component realisation. The discussion covers the available technologies and the trade-offs between price, speed and accuracy. Two implementations illustrate the problems and constraints of the different approaches.

Section one is an evaluation of the technologies available. The limitations of analogue and digital approaches are discussed. A third category, analogue-digital is introduced. This hybrid of two existing approaches is becoming more attractive with the advent of digitally programmable switched-capacitor filters and multiplying digital to analogue converters (MDAC).

Section two covers an analogue lattice structure which was configured as an adaptive echo canceller, one of the rare applications in which a lattice structure may act as an equaliser. The section explains the limited success of the approach and offers guidance for further work into the field.

A digital adaptive equaliser is the subject of section three. Outwardly this took the form of an adaptive filter. Its performance is evaluated, operating as an adaptive filter, data equaliser and spectrum analyser. Section four is the chapter summary.

6.1 Evaluation of Technology

6.1.1 Analogue

In purely analogue implementations the multipliers are four-quadrant transconductance types, using MOS or bipolar circuitry. These suffer from the usual problems of analogue circuits, namely thermal drift of DC and AC transfer characteristics. These problems may be circumvented by trimming and feedback circuitry.

Operational amplifiers perform the usual functions of addition, subtraction and integration (averaging). Thus exponential averaging is more convenient than that using other window functions. Circuits may be trimmed to adjust their DC transfer characteristics but the combination of feedback circuitry and high gain obviate this need for all but the most critical cases.

Division in analogue circuitry is often avoided, as methods are even less stable than for multiplication. The implementation usually involve an analogue multiplier in a negative feedback circuit. This makes four-quadrant operation (i.e. changes of sign) difficult. Algorithms may be modified using expanded forms so that the division is eliminated. An example is the Taylor expansion used in the stepsize recursion in section 7.3.2.

Delay elements in discrete time filtering are often implemented in surface acoustic wave (SAW) or charge coupled device (CCD) technology [76]. In the lattice, delay elements are needed in unit quantities throughout the circuit, so chains of delays are less useful. Capacitor memories are normally used as delays [54], often wired as conventional sample-and-hold circuits.

One applications for a CCD delay would be in a hitherto untried multiplexed analogue scheme. The signals to be delayed until the next signal sample would be fed into the CCD delay. There would be as many elements of delay as there were stages, the signals from successive stages being fed in succession into the delay line.

Another way of using CCD technology would be to use a hybrid lattice structure (cf section 3.5.3), in which the fast-convergence properties of the lattice are transferred to the transversal structure.

6.1.2 Digital

There are two styles of digital arithmetic, bit-serial and bit-parallel. The bit-serial approach is by far the simpler, requiring

one- or two-wire busses and simpler circuit elements. However, for a given technology the parallel approach is faster, processing data in word-sized blocks instead of bit-by-bit. The bit-serial approach is discussed in depth in chapter seven.

The bit-parallel approach requires wide busses, which make high precision expensive. The specialised components needed for addition and multiplication are also expensive, so much so that multipliers are normally multiplexed. This multiplexing puts great demands on the control and timing logic and reduces the bandwidth of the processor.

The respective advantages and disadvantages of serial and paralel approaches notwithstanding, the parallel approach is currently the more popular. This has led to a wider variety of available components in bipolar, CMOS, NMOS and other technologies. In spite of multiplexing and pin-out problems, most microprocessors and microprocessor-based systems use bit-parallel arithmetic. The popularity of the approach and the availability of the components have enhanced one another in a positive feedback loop, making the bit-parallel approach a natural choice.

Another set of alternatives is presented in the sequencing of the arithmetic operation. In a multiplexed structure the same processor shares its time between many operations. A microprocessor is an extreme example of this style of design. The other end of the range of alternatives is pipelining, in which all or some of the calculations are performed simultaneously by different sections of the hardware. A fully pipelined approach is discussed in chapter seven.

The input to a system is normally analogue, so digital systems often need analogue-to-digital converters at their inputs and digital-to-analogue convertors at their outputs. Nevertheless, in spite of this increased cost, the digital approach is preferred to the analogue. Digital precision is far higher, substituting truncation errors for the analogue problems of noise, drift and manufacturing tolerances.

6.1.3 Analogue and Digital

In some systems it is advantageous to mix analogue and digital approaches, using each for its most efficient feature. Thus complex arithmetic manipulation is accomplished in the digital domain, while the speed and economy of analogue circuitry are maintained. Analogue-to-digital converters of various kinds are used to convert between the two domains.

Multiplication is particularly attractive in mixed-domain circuits. Programmable switched-capacitor filters [54] may be used for this. Another approach is the multiplying digital-to-analogue converter (MDAC), which is an extremely linear way to multiply an analogue signal by a digital one.

The charge-leakage effects of analogue memories and analogue delays are avoided by digital memories, which can remember a number almost indefinitely without degradation. Non-linear manipulations in the digital domain are simplified using read-only memories as look-up tables. Examples are inversion, squares, roots and trigonometical functions.

6.2 An Analogue Lattice Structure

An analogue lattice structure [77] was constructed as part of a final-year student project, in which the author assisted in the design and evaluation stages. The ultimate objective was to design a cheap adaptive echo canceller.

The impulse response of a channel with an echo is called 'one-sided' if the channel causes solely post-echo. This is because the main lobe is the first coefficient of the impulse response, and the lesser terms never precede it. This kind of impulse response lends itself well to approximation by an AR function [55], so a pure forward PE filter or the output of the forward channel of an adaptive lattice may be used as an equaliser. This is one of the rare occasions when a lattice structure may be used in equalisation.

6.2.1 Construction

The lattice structure was constructed with four-quadrant transconductance multipliers type AD 533 and sample-and-hold circuits HA2425. The algorithm selected was the Mead algorithm [13]:

$$K(n+1,t+1) = K(n+1,t) + \mu.(f(n,t).b(n+1,t) + b(n, t-1).f(n+1,t)) \quad (6.1)$$

Where K is the single valued PARCOR coefficient for the stage and μ is the stepsize. Rapid convergence would have required an adaptive recursion for μ, the stepsize. Instead, this was given a fixed value. The Mead algorithm is illustrated in figure 6.1, where the PARCOR update recursion is performed with the aid of an integrator. Since the signal presented to the input of the analogue integrator was only valid during part of the switching cycle, it was preceded by a sample-and-hold circuit.

The timing for the sample-and-hold circuits was derived from a decoded four-bit TTL clock, driven by a free-running oscillator of variable frequency. An input sampling frequency of 10 kHz was used, making the logic clock frequency 160 kHz.

The stages of the lattice structure were clocked in reverse sequence; last stage first, followed by the second last and so on till the input stage. This ensured that the backward signal sample used in the calculations was that retained from the previous signal input sample, as required by the algorithm.

6.2.2 Test Results

Three tests were performed on the finished structure. An echo cancellation test was performed on a non-adaptive four-stage structure using manually adjustable PARCOR values. A two stage adaptive structure was subsequently tested with D.C. and sinusoidal signals.

The pulse with the echo to be cancelled had an impulse response of $1 + 0.5z^{-1}$. It was calculated that the Wiener solution using the

four stages available would give an error signal approximately 26 dB below the main impulse. In fact, the structure was trimmed manually to give a truncation of the IIR AR solution. This involved trimming the first stage to totally remove the echo at z^{-1}, then trimming the second stage to remove residual echo at z^{-2} and so on. The remaining echo at z^{-5} was approximately 24 dB below the main impulse, showing that the truncated AR solution was quite close to the Wiener coefficient values. Figures 6.2 and 6.3 show the filter trimmed to minimise the echo, and the filter impulse response.

On applying a DC signal to an adaptive lattice, the PARCOR value of the first stage should attain a value of -1, cutting off any signal to subsequent stages. In fact a 5V input gave a 0.1V output, an attenuation of 34 dB, a respectable result for an analogue circuit. It was not possible to measure the actual PARCOR values as the gains of the multipliers were not calibrated.

Section 2.3.7.1 explains that when a sinusoid is input to a lattice structure, the second PARCOR coefficient is equal to -1 and the output of the second stage is zero on either channel. The input sinewave for the test was 500 Hz. Thus the peak-to-peak signal at the output of the first stage should be attenuated by a factor of Sin(500/2 .10000), i.e. 0.3090. Figure 6.4 and 6.5 show that this is approximately true. In figure 6.4 the forward and backward signals at the output of the first stage are shown. It is seen that they are approximately in antiphase and that a second stage PARCOR value of -1 should cancel them out. The oscilloscope trace in the output from the second stage should be zero. However, figure 6.5 shows that it is in fact 0.1 of the input signal peak-to-peak voltage, an attenuation of 20 dB. The poor performance in this test was attributed to the manufacturing tolerances in the multipliers and the rudimentary trimming techniques which were deployed.

6.2.3 Comments

The analogue lattice appears to have performed slightly below expectations. The performance as a manually set, non-adaptive filter was as expected, but the adaptive loop appeared to introduce errors.

There are two main error mechanisms in analogue multipliers: DC offset and gain. In the lattice structure the only parameter to be trimmed was the DC offset in the signal path. The gains of the PARCOR coefficient multipliers were left unmatched. The manufacturing tolerances of $\pm$ 10% did not cause significant degradation in the echo-canceller application.

The analogue lattice structure was constructed as a feasibility study with a view to IC implementation. In integrated circuit multipliers, the characteristics are fairly well matched by virtue of sharing the same geometry and the same processing. However, even on IC's the DC offsets of the multipliers would be trimmed independently off-chip. The signal adjustment here was probably too approximate for an adaptive system.

The advantage of an adaptive system is that component imperfections are compensated by the feedback loop itself. Thus a mismatch in the gain of a multiplier in a transversal adaptive filter would not itself cause errors on the output. However it is important that for stationary signals both lattice PARCOR coefficients have the same value. One way of avoiding the problems of multiplier gain mismatch would be the two-value lattice algorithm of figure 6.6. This would allow each PARCOR coefficient to independently reach the correct value.

The sampling speed of the circuit was severely limited by the maximum speeds of the sample-and-hold circuits. These in turn were limited by the impedances of the circuits. Recent advances in switched capacitor techniques [54] have shown that cancellations of up to 60dB are possible using analogue techniques. These techniques, however, dispense with conventional transconductance multipliers and sample-and-hold circuits, using MDAC's and switched capacitors instead.

6.3 A Digital Lattice Equaliser

A 16-tap digital lattice equaliser was built in order to study the constraints of bit-parallel digital design. The multiplication implied in the algorithm necessitated the use of a parallel multiplier chip. The cost of these circuits meant that only one could be used, forcing a multiplexed arithmetic approach with associated bus structure.

Initially, the Mead algorithm, equation 6.1, was selected. This used a single-valued PARCOR coefficient which was updated using two averaged correlations per recursion. A method was devised of updating the stepsize recursively, but this was not implemented due to a lack of time. Instead, a fixed stepsize was selected manually and applied to all stages, by a control on the front panel.

It was noted that adaptive filters are generally very difficult to service. When a fault develops it is very difficult to locate, since the data is stochastic and convergence requires many iterations. Twelve-bit accuracy was chosen for the arithmetic, this being the maximum commensurate with component availability. The convergence of the filter coefficients was found to be uneven and imprecise, so their accuracy was increased to 23 bits. It is a property of adaptive filters that the accuracy needed for the stochastic convergence process is greater than that needed for the filtering operation itself.

The following is a brief description of the circuitry. The circuit diagrams themselves are in appendix A.

6.3.1 Construction

The equaliser was built on double-eurocard sized wire-wrap boards. The card size was selected so as to minimise backplane wiring. This policy was so successful that the backplane carried only data busses, the control bus and the power rails. The system was partitioned into I/O circuitry, memory and arithmetic units.

The design was centred around the arithmetic unit. The algorithm was broken down into a series of multiply-and-add operations and the unit was built accordingly. Three input busses were used, one for each of the input coefficients. A single output bus returned the results to memory. The arithmetic unit contained the parallel multiplier chip, which was operated at maximum speed. This established the clock rate of the equaliser.

Figure 6.7 shows the schematic block diagram of the equaliser. The arithmetic communicates with the memory unit, which supplies it with data and receives its output. Input and output is achievable via the memory unit to the ADC and DAC blocks. The system clock drives the control circuitry which is microprogrammed to control memory and processor operations.

Figure 6.8 is a photograph of the front panel. The photograph shows the variable delay on the trimming signal input and the switch to select the output stage. The switch to manually select the stepsize is visible, as is that to switch to automatic stepsize calculation. Automatic stepsize calculation and DFB were to be added to the equaliser, but subsequently this was dropped in favour of the integrated design, chapter 7.

6.3.1.1 The Arithmetic Unit

Figure 6.9 shows the block diagram of the arithmetic unit. The diagam shows the signal flow paths but not the control circuitry. The circuit is a multiply-and-add configuration. The two numbers to be multiplied appearing on input busses 1 and 2, and the addend appearing on bus number 3. The numbers are clocked into latches and the multiplication proceeds. After the result of the multiplication is valid it is held in one of two latches (marked R). One latch feeds the adder while the other provides a means of holding a result until the next cycle. The overflow detector checks the MSB's of adder input and output words. If an overflow has occured it sets the output to the most extreme valid number in the permitted range. The output appears on bus 4.

The timing of the unit is based on the operation time of the TRW 12 bit x 12 bit multiplier. This was conservatively estimated to be approximately 150 ns. This was multiplied by three to give an arithmetic cycle comprising I/O, multiply and add.

The multiplier chip takes in two 12-bit words and supplies a 23-bit result. Most of the calculations use only 12 bits, so the output of the multiplier is rounded. Coefficient update recursions, however, worked with 23-bit precision from the output of the multiplier to the output of bus 4. The path from bus 3 to the adder was also extended when enhanced accuracy was required.

Table 6A gives the instruction set of the unit. Instructions were latched into the unit from the control bus at the same time as the data.

6.3.1.2 Control Logic

The lattice equaliser arithmetic unit executed eight operations per stage of the filter. There were sixteen stages and a new data value was input and output after the sixteenth stage. The timing is summarised in table 6B. This permitted a sampling frequency of 17.3 kHz and thus a Nyquist frequency of 8.6 kHz. Table 6C gives the algorithm used and the operation code fed to the arithmetic unit for each equation thereof.

Since the algorithm required no conditional instruction, the timing was obtained by dividing down a free-running clock using synchronous counters. The counter outputs were decoded to provide control signals for various parts of the circuit. In most cases it was sufficient to feed the counter output into a simple decoder. However, a Schottky EPROM was used to supply complex logic functions, such as the operational code for the arithmetic unit.

6.3.1.3 The Memory Unit

The memory unit's operation is shown in figure 6.10. It consists of a number of latches and tri-state buffers. The control logic triggers the latch, which takes a value from bus 4 at the appropriate moment. This control logic directs the tri-state buffers to put the value onto one of busses 1 - 3 at the appropriate time.

The basic mode of operation is modified according to the needs of the coefficient. For example, there are facilities for latching out y(16,t) and e(16,t) for output via a DAC. The value of f(0,t) and b(0,t) reached busses 1 - 3 from an ADC circuit and not the normal latch. The values of μ are taken from a switch on the front panel and not a latch. In general, however, the operation follows figure 6.10.

6.3.1.4 Input-Output Circuitry

The input-output circuitry was relatively simple in concept. The input signals were taken via a sample-and-hold circuit through an ADC to a latch. Tri-state switches selected the latch each time the zeroeth stage was processed, instead of the normal f and b signal registers. The variable delay for the training signal was made using a schottky RAM with an adder on its address lines. The front panel switch selected the difference between input and output address, which was fed into the adder.

The output signals were latched from the error and equalised signal registers. The time of latching was controlled by the output stage selector on the front panel. The latches fed a DAC which in turn fed an operational amplifier as analogue buffer. No anti-aliassing filters were used on inputs or outputs as it was felt that they would complicate rather than simplify measurements of performance, introducing ISI of their own.

Most measurements were done with the aid of an oscilloscope. Nevertheless, a bar-LED display was included in the front panel design so that the reduction of the error signal during convergence

could be observed. The design of the driving circuitry was simple and novel: If the signal held by the output latch was positive, other binary ones in the latch were permitted to set bistables, turning on bars in the display. Thus over a period of 40 ms a picture was built up of which output lines were active, i.e. the magnitude of the output signal. The latches were reset every 40 ms, a frequency above the human flicker frequency. The effect presented to the human eye was that of a bar whose height varied with the logarithm of the error power.

6.3.1.5 Features not Implemented

Provision was made for reprogramming the control logic to provide extra features to the equaliser. These were not implemented due to lack of time.

The decision feedback (DFB) feature would have replaced the external training signal with an internally developed one. The training signal in DFB is calculated by detecting the equaliser output and assuming that the detected value is correct. The ideal detector input corresponding to that detector output is then used as a training signal.

In order to maintain a reliable rate of convergence, an equaliser must have an adaptive stepsize recursion. A typical recursion is equation 3.27, involving a square, a multiplication, an addition and an inversion. It would have been possible to simplify the PARCOR update recursion so that it only needed one correlation, and to use the multiplication cycle thus freed for a stepsize update. The square and invert functions would have been implemented using ROM memories as look-up tables.

There was in fact an automatic stepsize calculator incorporated into the hardware. This was not adaptive, but supplied continuously reducing values of stepsize to the coefficient update recursions. Thus the stepsize and algorithm noise would reduce with time as the adaptive algorithm converged, an idea investigated more fully in [7]. Although designed and constructed, this section of the equaliser was

never tested.

6.3.2 Experimental Results

Two types of tests were performed on the digital lattice equaliser: Firstly it was tested as an adaptive filter in filtering, equalisation and cancellation tests. Secondly it was tested as a spectrum analyser.

The adaptive filter tests used as test signals sinewaves and convolved data sequences. The data sequences were generated by constructing a pseudo-random binary sequence (PRBS) generator which gave a sequence length of 2^{24}-1. The output from the generator was convolved with a raised cosine impulse response to create a test signal for the equaliser.

6.3.2.1 The Equaliser as an Adaptive Filter

6.3.2.1.1 Sinewave Filtering

In this test two sines were applied to the s(t) input of the equaliser. These were of frequencies 2.1 and 3.3 kHz. The input to the d(t) input was the 2.1 kHz sinewave. One may see that the ideal filter solution which minimises the error power is that the filter notches out the 3.3 kHz sinewave and outputs the 2.1 kHz sinewave in phase with the d(t) signal. If this is done perfectly there will be no error power whatsoever.

The stepsize of the filter was set at minimum (0.0005)so as to minimise algorithm noise. Convergence then took approximately 10s. Figure 6.11a shows the time-domain oscilloscope traces of input and output signals, showing that the predicted solution had been achieved. Figure 6.11b is the spectrum of the input s(t); Figure 6.11c is the spectrum of the output, y(t).

The bar-LED display showed that 8 bits of cancellation had been achieved, 48 dB, a figure that agreed with oscilloscope and spectrum analyser results. It was noted that since a pair of sinewaves is

completely predictable by a four-stage lattice, the prediction-error Q(4) being zero, only three lattice stages were necessary for this test. This was confirmed by using the panel switches to truncate in the filter.

6.3.2.1.2 Sinewave Cancellation

Here, two sinewaves are supplied to the d(t) input, 2.1 and 3.3 kHz. Only one, 2.1 kHz was supplied to the s(t) input. The solution minimising the error power output will supply a sinewave to the y(t) output of the correct phase and amplitude to completely cancel one of those on the d(t) signal. Since the other sinewave does not appear on the s(t) input, it may not be cancelled from the d(t) signal and appears unattenuated on the e(t) output.

Figure 6.12a is an oscilloscope trace showing filter input and output in the time domain. Since there is a large signal on the e(t) output, it is not possible to estimate the depth of cancellation without a spectrum analyser. Figure 6.12b and c are the spectra of the outputs of the y(t) and e(t) parts respectively. They show a cancellation level of 50-60 dB. Owing to the behaviour of lattice structure when fed sinusoidal inputs, only one lattice stage and two sidetaps were necessary for this result.

6.3.2.1.3 Channel Equalisation

In this test a PN code generator was used to generate a bipolar binary sequence of length $2^{24}-1$. This sequence was fed directly into the training signal input of the filter. The sequence was then convolved with a raised cosine impulse response (cf R11 in appendix D). The convolution output was fed into the s(t) input to the filter. The equaliser should have been able to de-convolve the PN signal with a high degree of accuracy. The ideal error attenuation was calculated by computer to be -80 dB, athough the hardware could never attain this because of truncation errors etc.

The top set of traces, figure 6.13a, show the adaptive lattice equaliser converging onto the channel. The d(t) signal shows the

binary PN input to the channel. This is delayed and used in the equaliser as a training signal. The topmost trace is s(t), the channel output. The convolution of the binary input with the three-point symmettrical impulse response gives six possible combinations, each producing a different voltage level. This gives the trace a layered appearance. The e(t) signal is the error output from the equaliser. The adaptive algorithm was switched on 0.5 ms after the start of the trace (0.5 divisions). The convergence is rapid as the stepsize has been set to 0.12, giving an error 18 dB below the equalised output. It is difficult to decide when the filter has converged, as convergence is asymptotic, but a visual estimate is 4.5 divisions or 80 samples. The y(t) signal is the equaliser output. It is shown starting from zero, as all taps are initialised to zero. It builds up gradually to become a noisy estimate of the d(t) signal.

The lower set of traces, figure 6.13b, is a multiple trace exposure showing the eye diagrams of the converged system. The timebase is synchronised to the d(t) signal and a comparison with the synchronised portion of the y(t) signal shows that the delay through the filter is 8 samples, making the impulse response of the equaliser maximally symmettrical. A stepsize of 0.06 was used, giving an error signal e(t) 21 dB below the output of the equaliser. The error signal is white and non-coherent, so it appears on the screen as a bright band rather than a trace.

6.3.2.2 The Equaliser as a Spectrum Analyser

6.3.2.2.1 Spectral Estimation of 5 sinusoids

The technique of autoregressive spectral estimation was described earlier in section 2.3.7.2. This experiment was carried out in two stages. Firstly a computer simulation was done to test the accuracy of the technique itself. Then the digital hardware was fed a similar test signal, the PARCOR coefficients were read out on a logic analyser and another estimate was formed based on the hardware values. The latter was then compared with a spectrum analyser display of the same test signal.

The test signal used was a combination of 5 sinusoids, of equal magnitude, with a background of white noise. In the computer simulation the noise level was set at -23 dB relative to one of the sinusoids, all of which had the same magnitude. Having calculated the coefficients of the PE filter, the power spectrum was calculated using a 256-point Fourier transform. This gave a frequency resolution of 67 Hz between the frequency estimates or 'bins' of the transform. The content of each 'bin' was then inverted to give the power spectral estimate of the test signal.

Figure 6.14 shows the power spectral estimates for different orders of PE filter. The surface between the different orders has been filled in by a simple linear interpolation in order to show how the details of the spectrum build up with increasing order.

Measurements were taken directly from the surface plot of relative spectral density, since the calculation involved no absolute power estimate. In table 6D the point of reference is the mean height of the five spectral peaks. The results show that the estimation of frequency is of the same order as the resolution, indicating that a longer Fourier transform would have increased accuracy proportionately. The estimates of amplitude fell within a 16 dB range, not a particularly accurate result. The estimation of the noise level was difficult, owing to the ripples in the spectral floor, but the estimate was within 1dB of the theoretical figure of -44dB.

The overall view is that in this case the resolution of frequency is limited only by the resolution of the Fourier transform. The resolution of amplitude information is far less reliable, although the overall picture appears correct to the human eye.

Figure 6.15 shows the hardware test signal, measured on an audio spectrum analyser. A Krohn-Hite tunable filter was used as an anti-aliassing filter, set to a cut-off frequency of 6kHz. The effect of the anti-aliassing filter may be observed from the noise floor of the analyser trace. The test signal itself was made by tape-recording the outputs of 5 signal generators on a cassette recorder, thus

guaranteeing the repeatability of the results.

The test signal was then fed into the lattice hardware. The stepsize was initially set high for most rapid convergence. As the test proceeded the stepsize was manually reduced to the minimum level (0.0005). The PARCOR coefficients were then read out on a logic analyser and then fed into the computer by hand for the spectral estimation. Again, no power estimate was taken from the digital hardware, so the spectral estimate was a relative one. Figure 6.16 shows the resulting spectral surface, calculated from the hardware coefficients.

Table 6D compares the spectrum analyser and digital hardware results. The frequency estimates fall within a range of 120 Hz, twice the resolution of the fourier transform. This increase in variance is probably due to errors in reading the display of the spectrum analyser. The estimates of amplitude fell within a range of 17 dB, a result comparable to that for the computer simulation of the technique.

A spectrum of this particular nature is an exacting test of the AR technique. The amplitude of a sinewave is inversely proportional to the distance of its respective pole pair from the unit circle of the z-plane. Under low-noise conditions this distance is small, so inaccuracies in estimating the position of the poles cause large errors in amplitude estimates.

Another source of error is the Fourier transform itself. This may be visualised by picturing a Fourier transform as a bank of narrowband filters. A sinusoid not in the centre of a 'bin' or filter will be attenuated by up to 4dB [56], causing an error in its amplitude estimate. The Pisarenko technique was mentioned in section 2.3.7.1. By calculating the zeroes of the PE filter coefficients directly instead of via a Fourier transform a more accurate result would have been obtained. Furthermore, the Pisarenko technique, which models signals as sinusoids in noise, would have been more computationally efficient, since only five values were required.

6.3.2.2.2 Spectral Estimation of Human Speech

A test recording was made of a human male saying 'ee' as in 'feed'. The vowel was sung rather than said so as to maintain a constant fundamental frequency. The test recording was then played into the spectrum analyser and digital hardware, as in the 5-sinusoids experiment.

Figure 6.17 shows the spectrum analyser display and figure 6.18 shows the autoregressive spectral estimate. Visually comparing the two, one is impressed by the overall simularity of shape. The result from the digital hardware is simpler, whereas the analyser display is more detailed but noisier. Both show the tailoring off of background noise with frequency, caused by the anti-aliassing filter.

Although the AR estimate is perfectly adequate for LPC and human ears, it is an example of how spectral detail becomes averaged, merged or rounded when the filter order is low.

The two displays were compared by measuring the heights of the spectral peaks. This was not easy as the spectrum analyser sometimes displayed two adjacent peaks while the spectral surface showed one. Table 6E gives the comparison.

It must be pointed out that the high power in the low-frequency components of the spectra is a characteristic of human speech. In LPC work this is compensated by the use of pre-emphasis and de-emphasis filters. Thus the higher frequency components, carrying the intelligence of the speech, are given greater weight in the encoding process.

The table shows a high degree of accuracy in estimating spectral density, but a low accuracy is estimating the frequencies of the spectral peaks. The peak detected by the analyser at 4200 Hz has been completely missed by the lattice, and the peaks at 5900 and 6600 Hz have been merged into one.

In this kind of spectrum, features are rounded and ill-defined.

It is thus to be expected that there should be a wide tolerance in estimating their exact location. Nevertheless, a very high degree of accuracy is shown in estimating spectral density, far higher then in the 5-sinusoids experiment. This is probably because the poles of the speech signal are not so close to the unit circle, making errors in estimating position less critical.

6.4 Summary

The purpose of this chapter was to review the construction methods of building lattice structures and equalisers. The analogue approach appears especially attractive because of the low price of the components and the lack of need for converters between the analogue and digital domains. However, analogue implementations need careful individual trimming if they are to perform to reasonable standards. Some of the mismatches between individual components may be compensated by the closed-loop nature of the gradient algorithm and the use of two-valued algorithms. In general, however, the discrete analogue approach with transconductance multipliers seems to bring problems which outweigh its low cost.

The discrete digital equaliser with its complement of 160 IC's is a more precise, though expensive, way of implementing a lattice. The high cost of parallel multipliers necessitates a multiplexed arithmetic approach which limits operation to audio bandwidths.

The tests of spectral estimation using the digital hardware have revealed the stength and limitations of this method. Certainly it gives a very 'human oriented' picture of the spectrum emphasising spectral peaks and concentrating on stationary features. Narrow-band signals have been shown to give a high precision in frequency but a low precision in amplitude, a result of the stability criterion of the AR model. Wideband features are given with more precision, however. The reduction in precision in the frequency estimates in the wideband signal were probably more due to the wideband nature of the signal than to the techniques themselves.

One may ask whether discrete component adaptive filters have a

place in future systems. As the cost of custom LSI drops the compactness of LSI brings a range of higher-quality techniques. Thus programmable switched capacitor implementations will become more viable. The next chapter reports a custom LSI approach to lattice design.

INSTRUCTION	OP-CODE	DECIMAL EQUIVALENT
Multiply and add to number on bus 3	1 0 0	4
Multiply and subtract from number on bus 3	0 0 0	0
Multiply and save in register R2 (bus 4 = bus 3 - R1, result of previous multiplication)	0 1 0	2
Multiply and add register R2 contents	1 1 0	6
Multiply and add to number on bus 3 (extended precision)	1 0 1	5

TABLE 6A The Instruction set of the arithmetic unit.

System clock	2 x 75 ns = 150 ns (6.6 MHz)
Arithmetic cycle	3 x 150 ns = 450 ns (2.2 MHz)
Stage cycle	8 x 450 ns = 3.6 us (227 kHz)
Sample frequency	16 x 3.6 us = 57.6 us (17.3 kHz)
Nyquist	2 x 57.6 us = 115.2 us (8.6 kHz)

Table 6B Digital lattice system timing

Cycle No	Bus 3	Bus 1	Bus 2	Bus 4	OP Code
0	f(n,t)	- K(n+1,t).	b(n,t-1)	= f(n+1,t)	0
1	b(n,t-1)	-K(n+1,t).	f(n,t)	= b(n+1),t	0
2	y(n,t)	+ G(n,t).	b(n,t-1)	= y(n+1,t)	4
		(G(n,t).	b(n,t-1)	= R1)	
3	e(n,t)	+ R1		= e(n+1,t)	2
		(f(n+1,t).	b(n,t-1)	= R2)	
4	0	+ e(n+1,t).	b(n,t-1)	= EB	4
5	G(n,t)	+ EB	. $\mu_G(n,t)$	= G(n,t+1)	5
6	R2	+ f(n,t).	b(n+1,t)	= FB	6
7	K(n,t)	+ FB	. $\mu_K(n,t)$	= K(n,t+1)	5

R1, R2 Multiplier output latches

FB, EB Temporary intermediate variables

Latch Contents and temporary variables shown ringed to indicate use

Table 6C Lattice equaliser algorithm

Frequency (Hz)		Amplitude in dB	
Nominal	AR estimate	Nominal	AR estimate
1500	1500 (+0)	0	- 6.8
2100	2090 (-10)	0	- 3.6
3300	3340 (+40)	0	+ 4.5
3900	3930 (+30)	0	+ 9.3
5100	5125 (+25)	0	- 3.6
Noise	----	- 44	- 45

a) Evaluation of autoregressive techniques by computer simulation

Frequency (Hz)		Amplitude in dB	
Analyser	AR (HW)	Analyser	AR, HW
1400	1430 (+30)	2	0.5 (-1.5)
1950	2020 (+70)	-0.5	3 (3.5)
3350	3320 (-30)	1	10.5 (9.5)
3750	3740 (-10)	-1	-8.5 (-7.5)
5100	5050 (-50)	-2	-5.5 (-3.5)
Noise	---	-40	-44.5 (-4.5)

b) Comparison between spectrum analyser measurements and autoregressive estimate from digital hardware

Table 6D Spectral estimation of 5 sinusoids on a background of white noise. Differences between measurements shown bracketed.

Frequency (Hz)			Amplitude (dB)	
Textbook	Analyser	AR (HW)	Analyser	AR (HW)
270	200	200 (0)	0	+3 (+3)
2290	2150	2321 (+171)	-28	-27 (+1)
3010	3000	3100 (+1000)	-18	-21 (-3)
	4200	-	-40	-
	5200	4860 (-340)	-31	-32 (-1)
	5900	6273	-42	-42 (0)
	6600	-	-40	-

Table 6E A Comparison of spectral peak positioning of the vowel 'ee'. Textbook formant frequencies from [57].

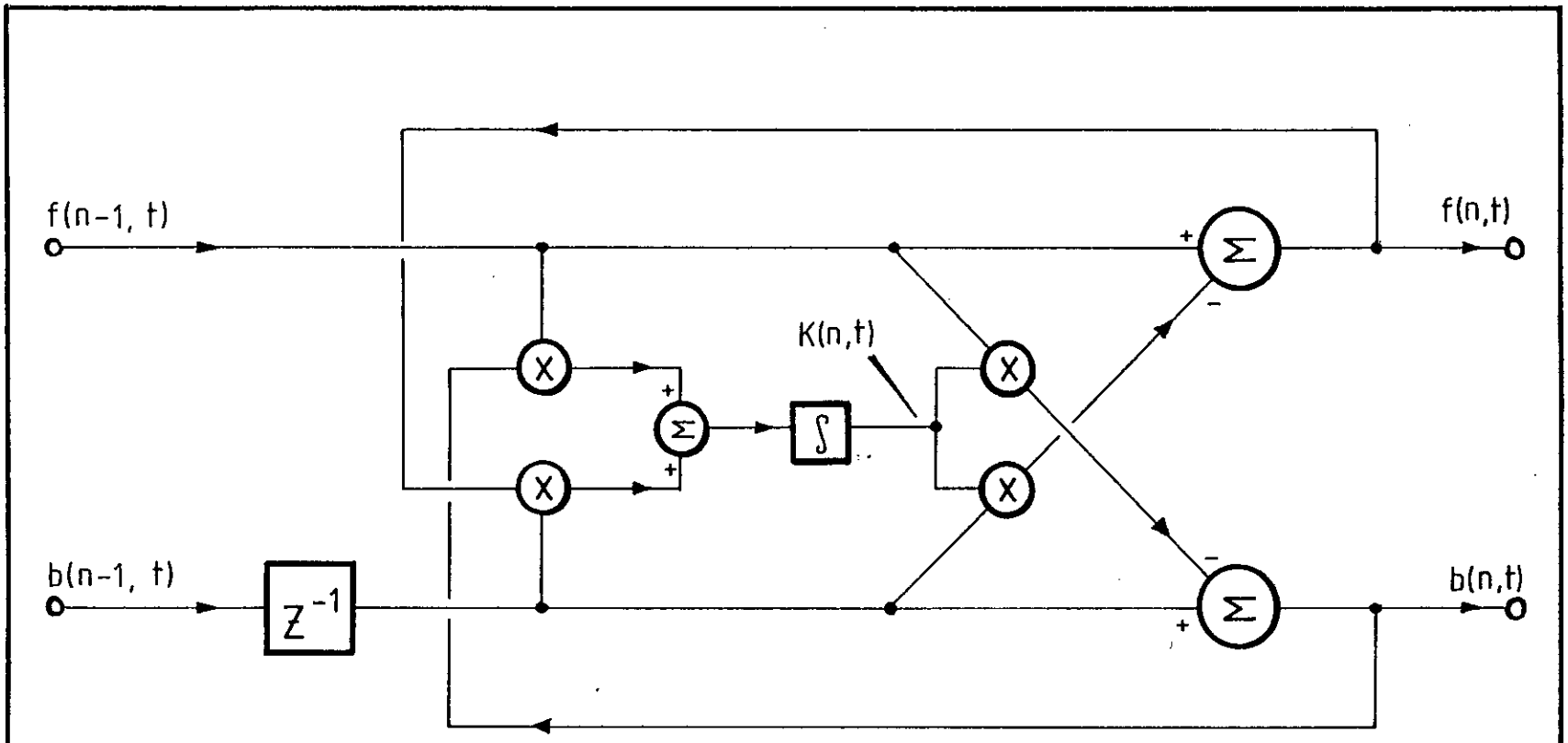
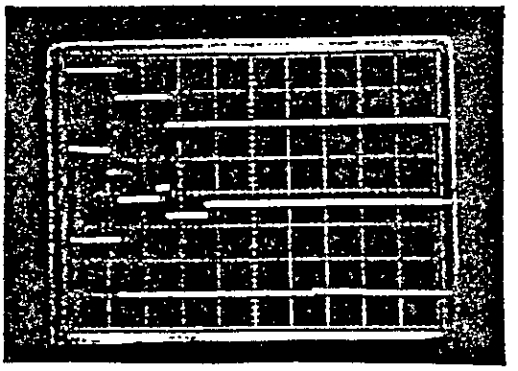


Fig 6.1 Flowchart of the Mead lattice algorithm

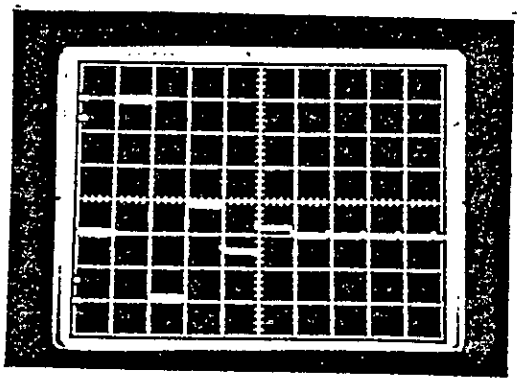


a

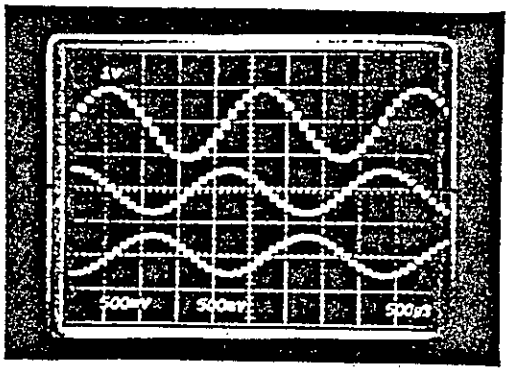
b

c

a) input signal with post-echo
b) forward signal output from first stage
c) forward signal output from fourth stage



above: Fig 6.2 Lattice echo canceller
below: Fig 6.3 Echo canceller impulse response



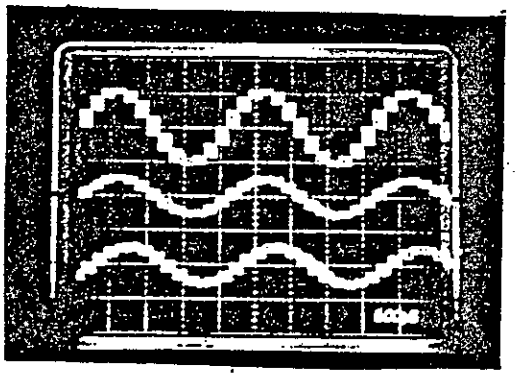
a

b

c

a) input signal
b) forward signal output from first stage
c) backward signal output from first stage

(vert: 10V/div, horiz: 500μs/div)



a (10V/div)

b (2V/div)

c (2V/div)

a) input signal
b) forward signal output from second stage
c) backward signal output from second stage

(horiz: 500μs/div)

above: Fig 6.4 Sinusoidal input (stage 1)
below: Fig 6.5 Sinusoidal input (stage 2)

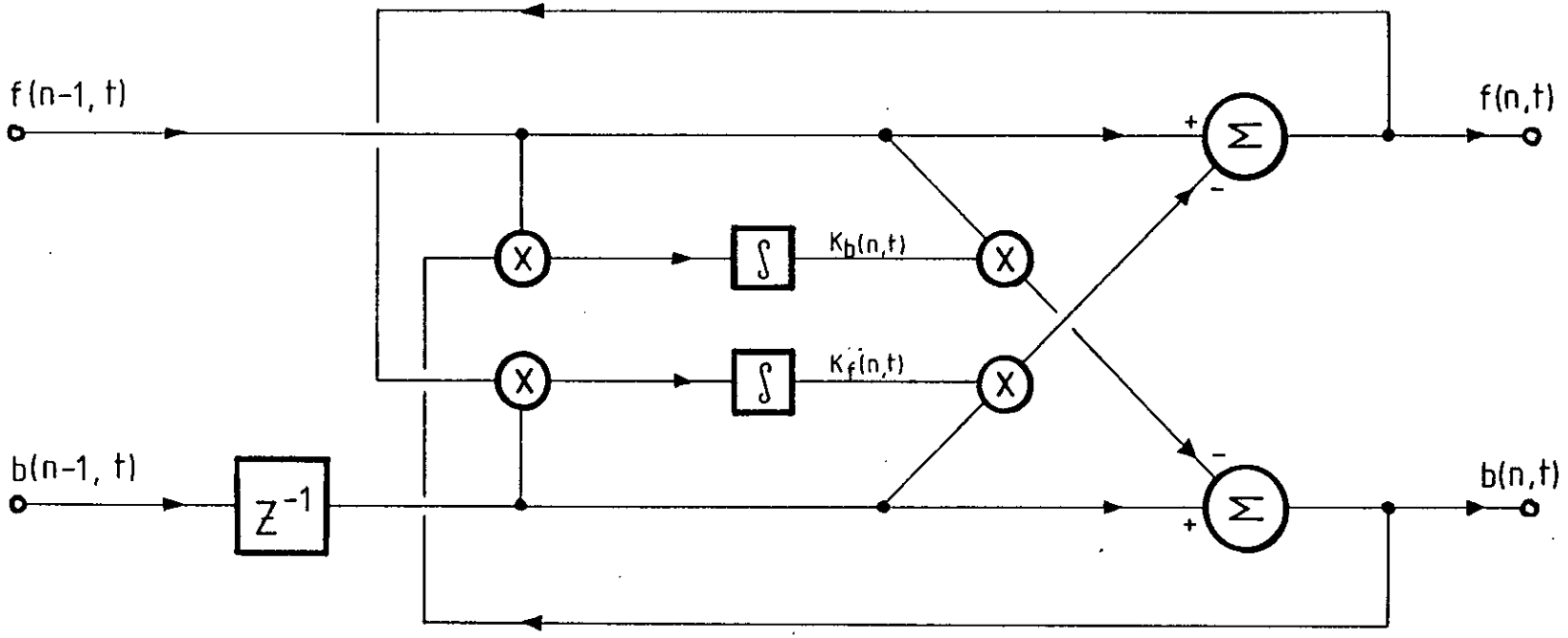


Fig 6.6 Flowchart of two-valued Mead lattice algorithm

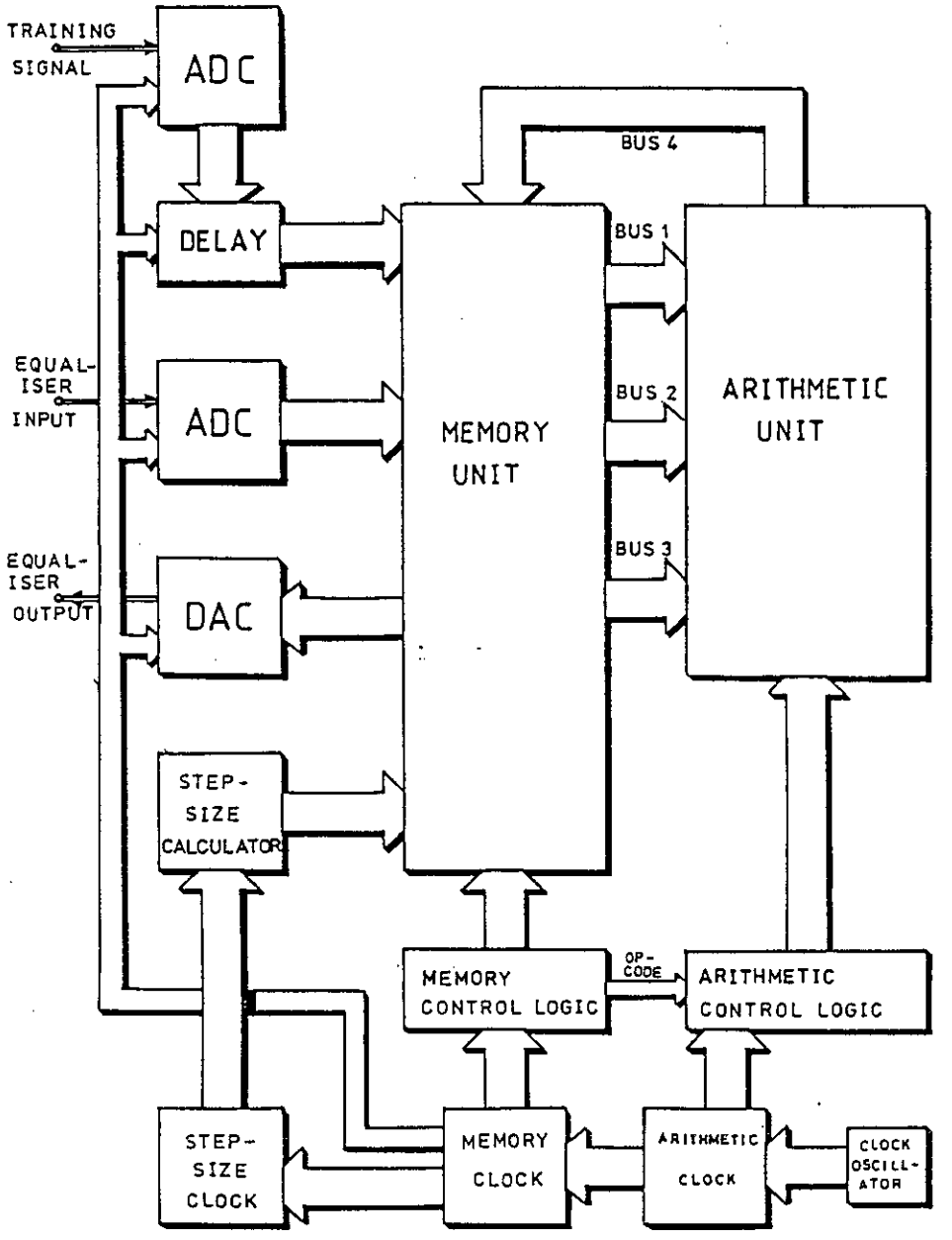


Fig 6.7 Digital lattice equaliser block diagram

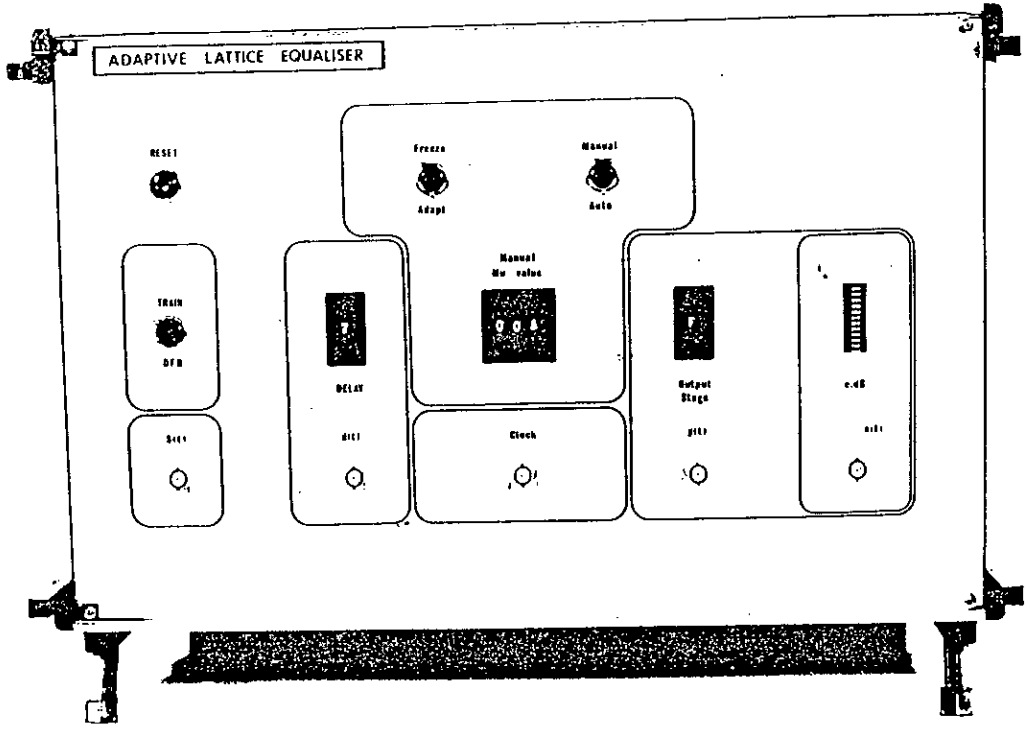


Fig 6.8 Digital lattice equaliser front panel

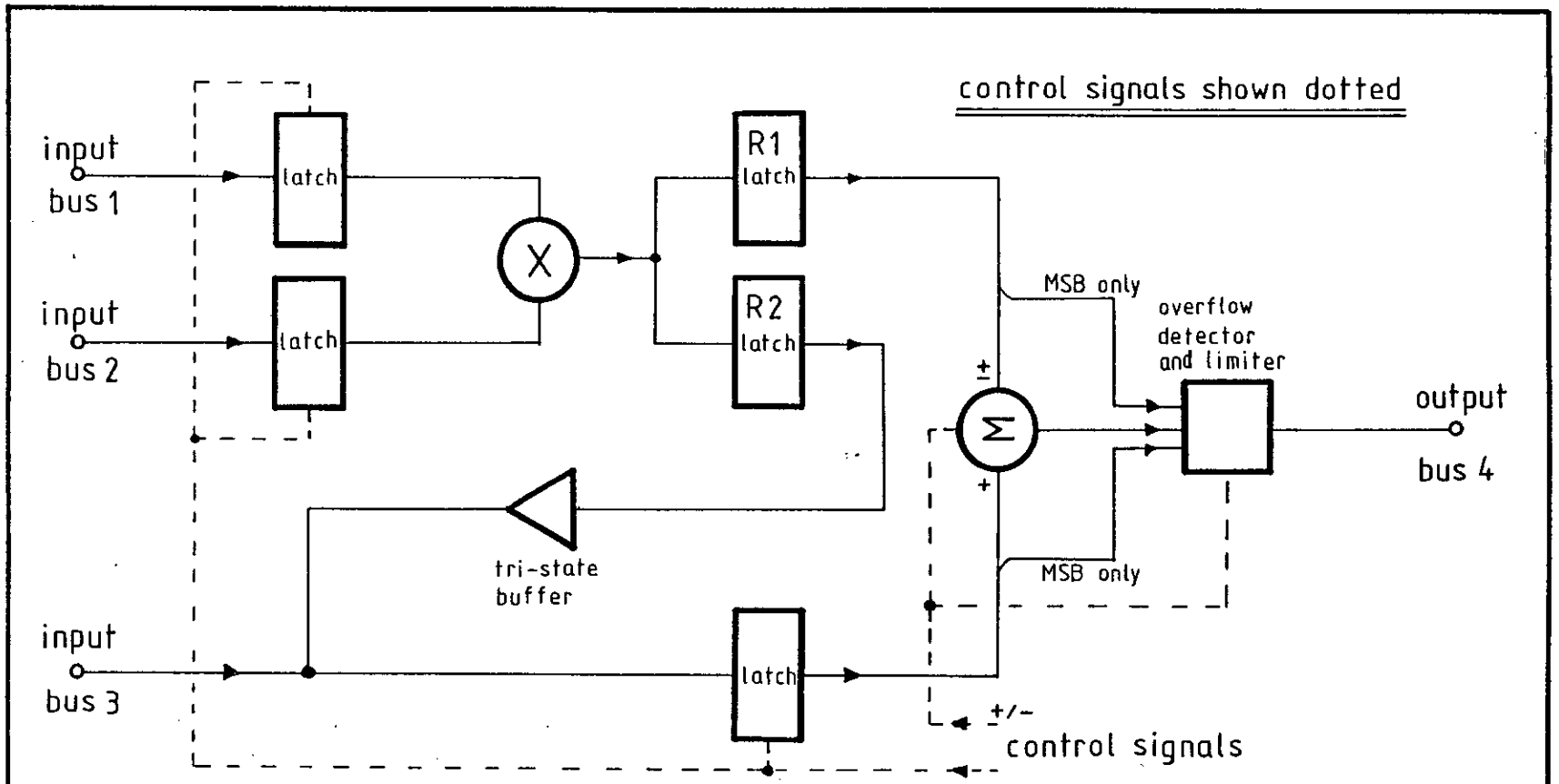


Fig 6.9 Block diagram of arithmetic unit

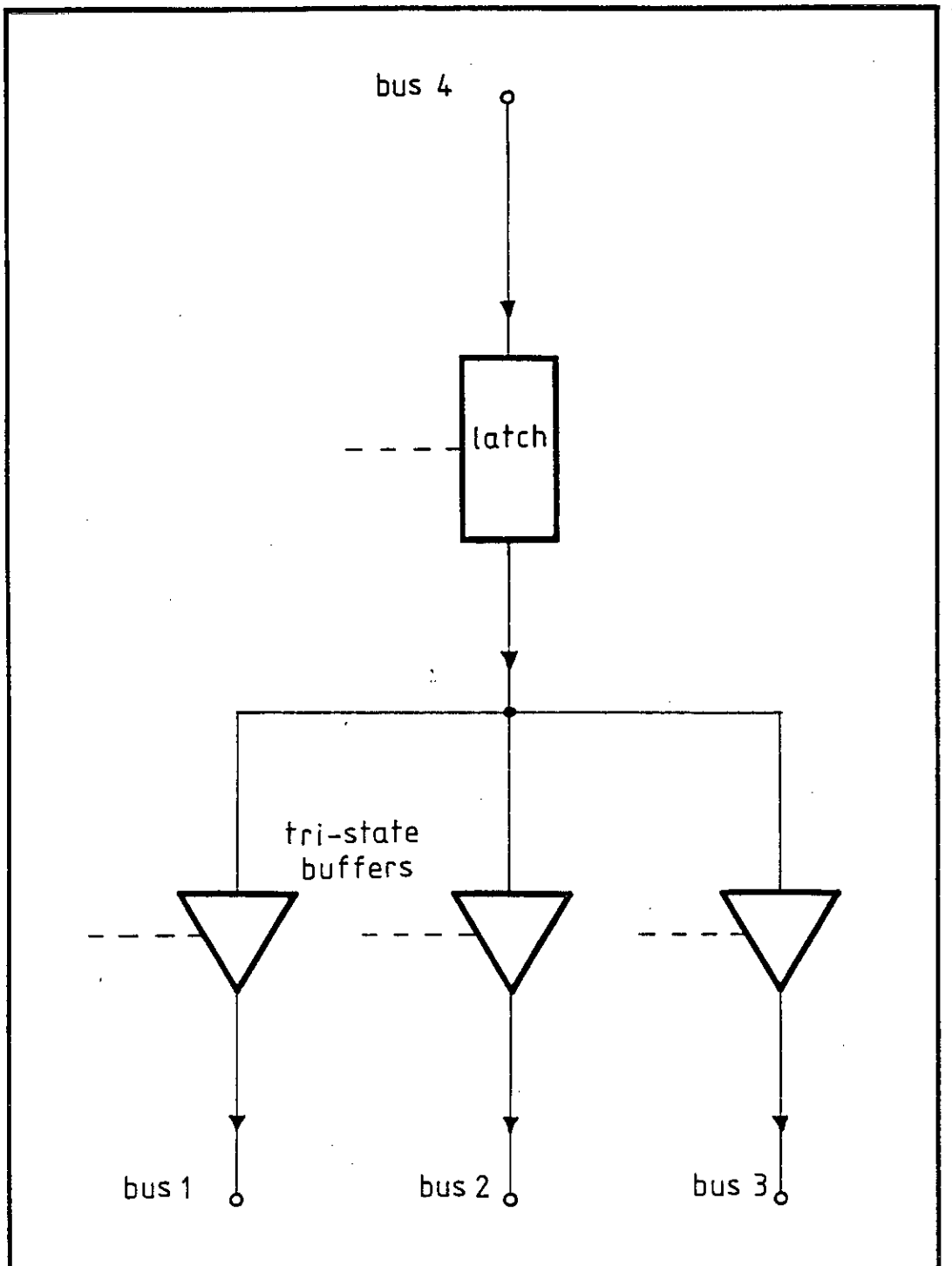
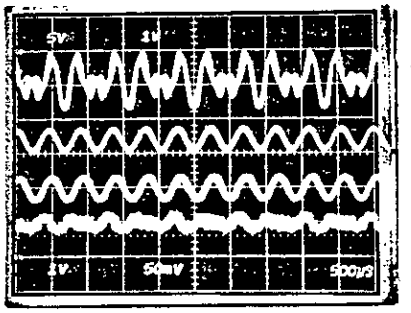
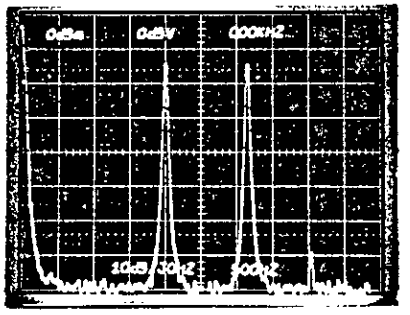


Fig 6.10 Memory unit operation
(control signals shown dotted)



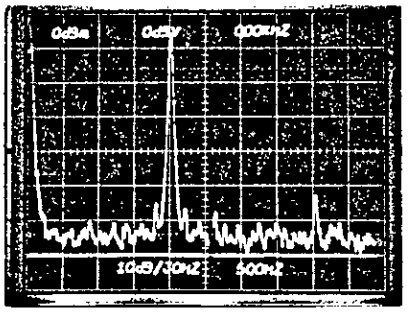
– s(t) (5V/div)
– d(t) (1V/div)
– y(t) (1V/div)
– e(t) (50mV/div)
(horiz: 500μS/div)

oscilloscope trace of input and output signals



vert: 10 dB/div
horiz: 500 Hz/div

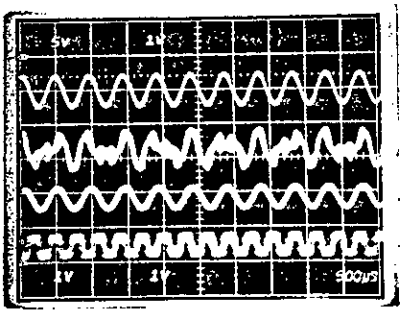
spectrum of s(t) showing two sinusoids



vert: 10 dB/div
horiz: 500 Hz/div

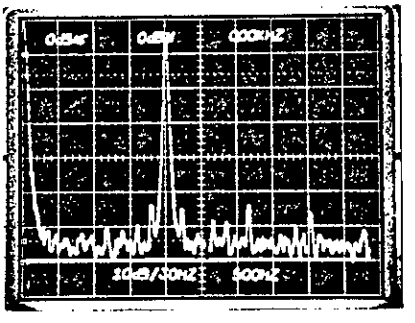
spectrum of y(t) showing one sinusoid

Fig 6.11 Sinewave filtering test



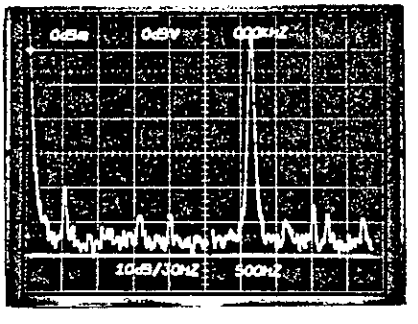
s(t) (5V/div)
d(t) (1V/div)
y(t) (1V/div)
e(t) (1V/div)
(horiz: 500μS/div)

oscilloscope trace of input and output signals



vert: 10dB/div
horiz: 500Hz/div

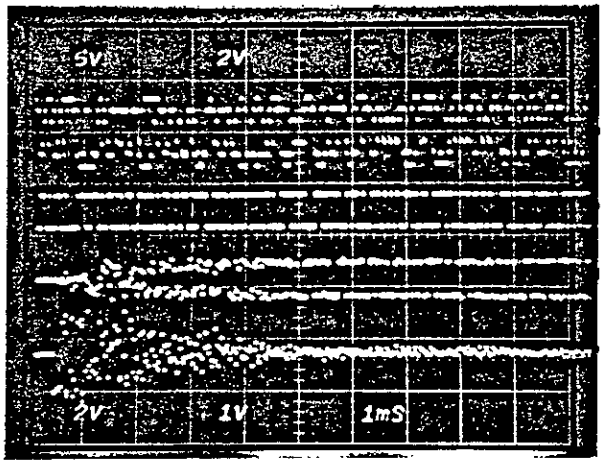
spectrum of y(t) showing 2.1 kHz sinusoid



vert: 10dB/div
horiz: 500 Hz/div

spectrum of e(t) showing 3.3 kHz sinusoid

Fig 6.12 Sinewave cancellation test



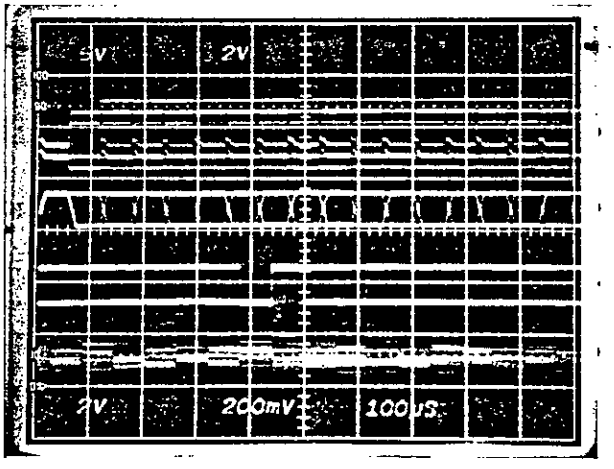
s(t) (5V/ div)

d(t) (2V/div)

y(t) (2V/div)

e(t) (1V/div)

horiz: 1mS / div



s(t) (5V/div)

d(t) (2V/div)

y(t) (2V/div)

e(t)
(200 mV/div)

horiz: 100 µS/div

Fig 6.13 Operation as a data equaliser

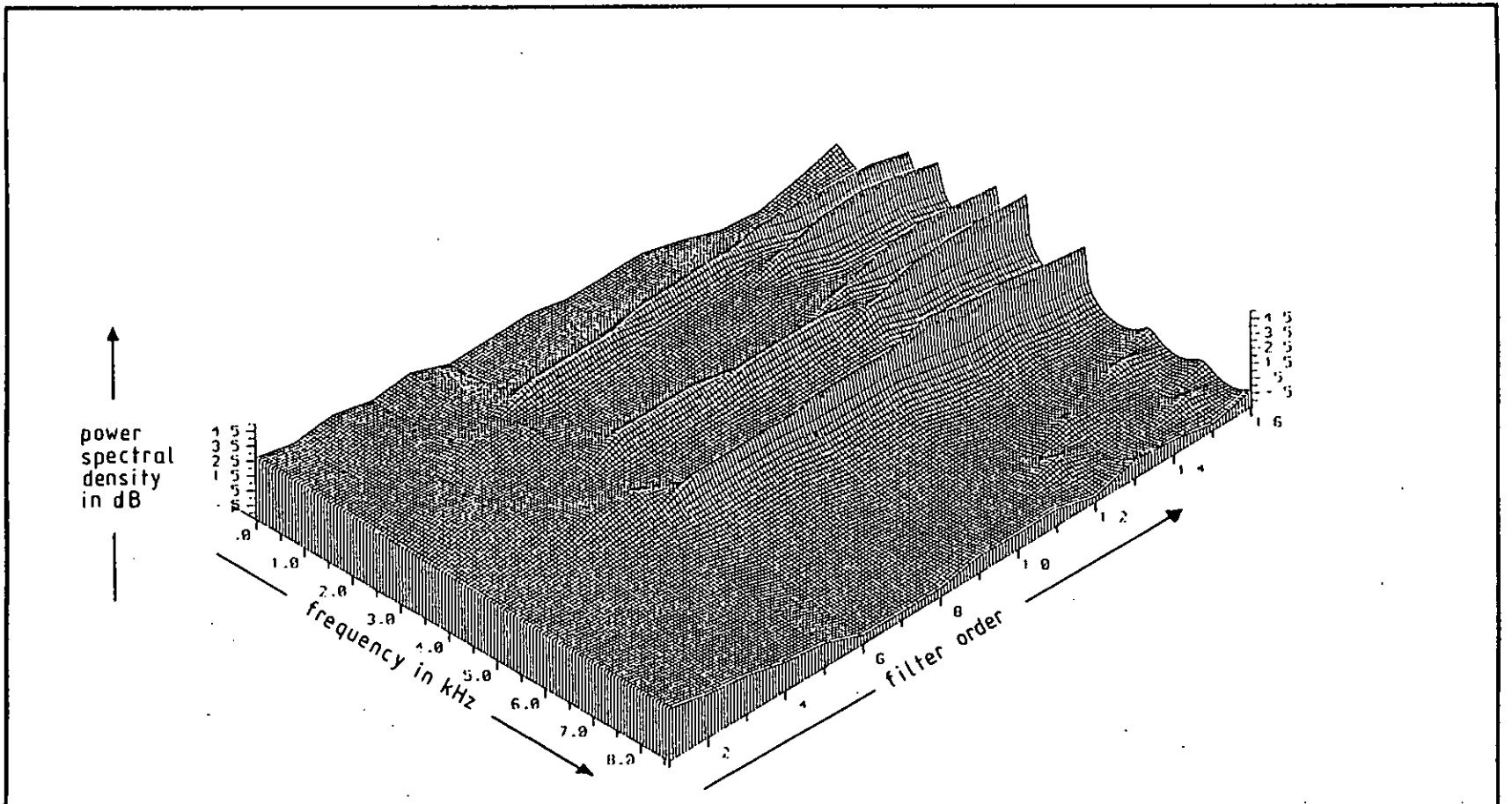
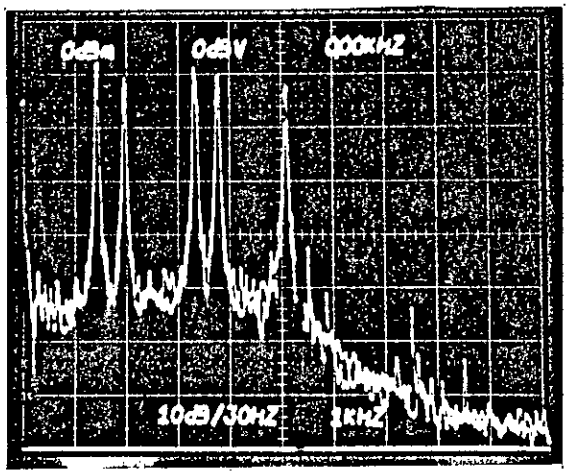


Fig 6.14 Computer simulation of autoregressive spectral estimate of 5 sinusoid signal



vert: 10 dB / div

horiz: 1 kHz / div

Fig 6.15 Hardware test signal, measured on a spectrum analyser

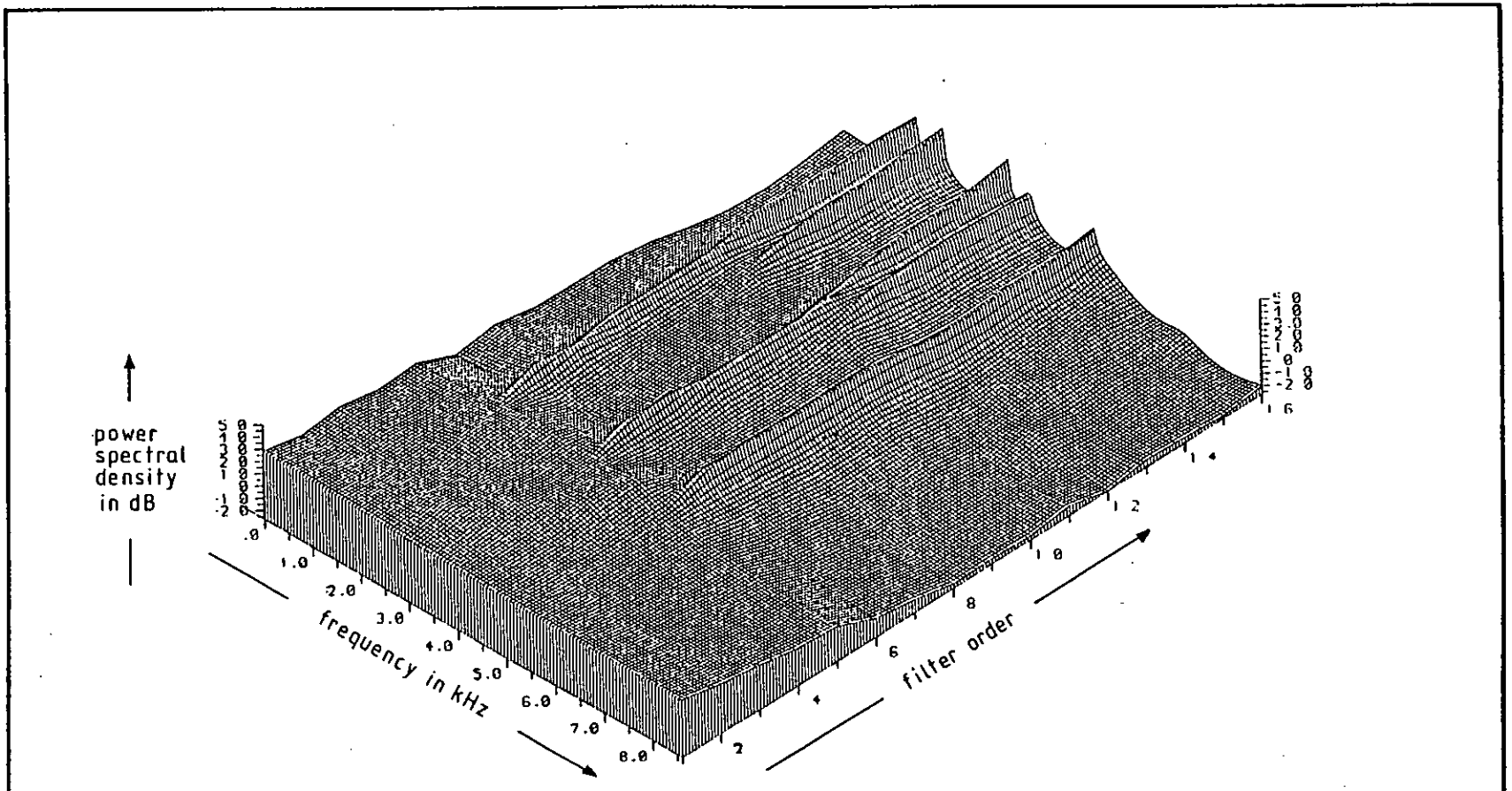
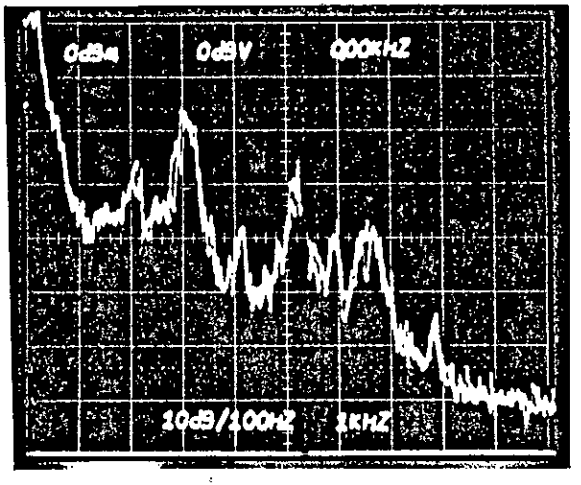


Fig 6.16 Spectral estimate of 5 sinusoid signal, measured using experimental hardware



vert: 10 dB / div

horiz: 1kHz / div

Fig 6.17 Spectrum analyser display of male saying 'ee' as in 'feed'

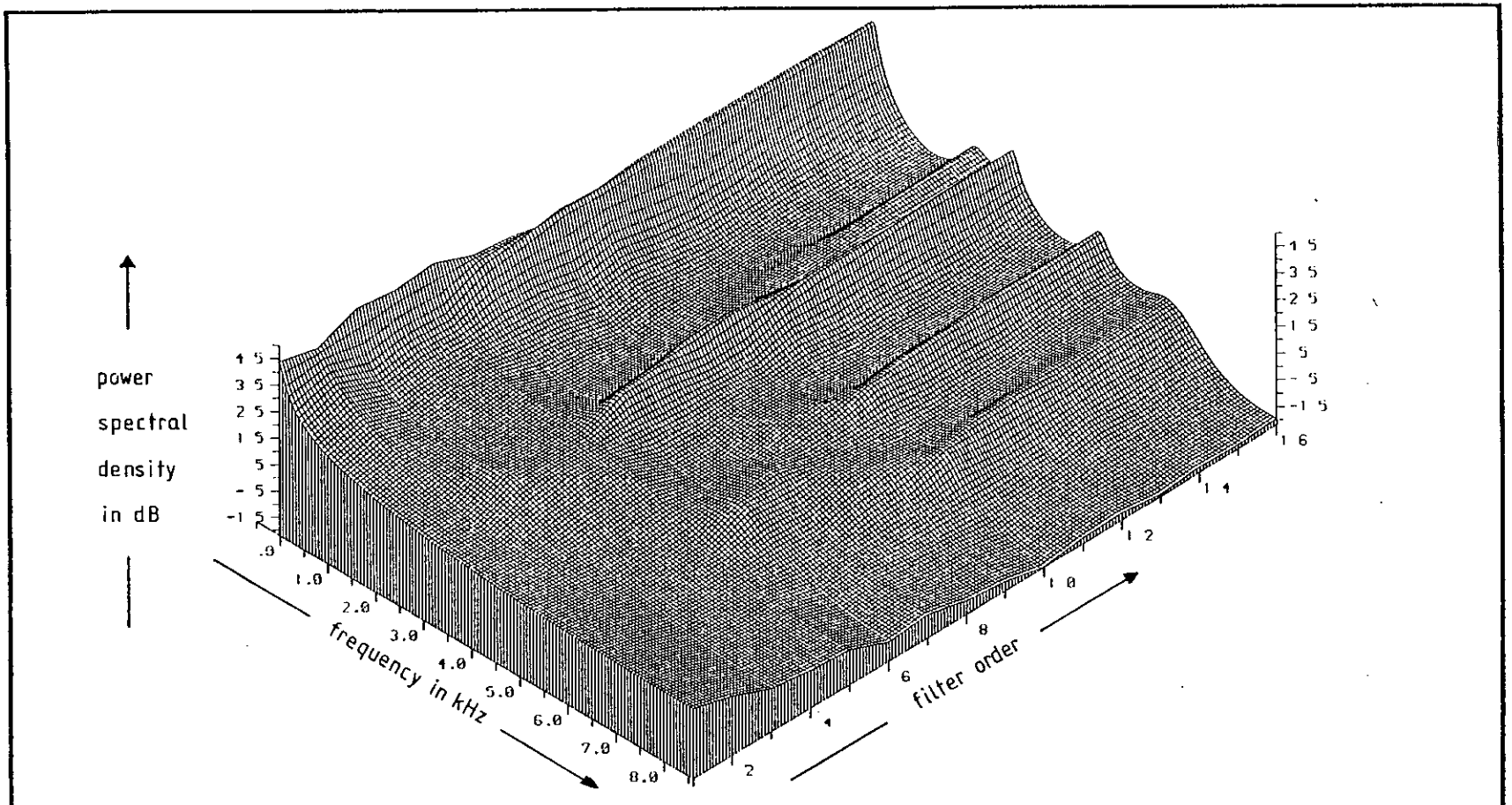


Fig 6.18 Hardware AR spectral estimate of male saying 'ee' as in 'feed'

Chapter Seven

A LATTICE STRUCTURE CHIP SET

This chapter describes the design and testing of a lattice structure (adaptive PE filter) using the FIRST silicon compiler. FIRST is an acronym for Fast Implementation of Real-time Signal Transforms. (58). The software tools offered by the FIRST system permitted the design to be exhaustively tested in an accurate computer simulation. Only when it was certain that the chip set performed as planned would it actually be fabricated in silicon.

Section one explains the pipelining possible with the bit-serial arithmetic employed in the FIRST system. Section two describes the compiler itself. The third section explains the steps necessary to design and build a system. The problems with the arithmetic precision and ranges are contrasted with those in the multiplexed bit-parallel approach of chapter six. Section four describes the testing of the system, both debugging and performance evaluation, while the chapter is summarised in section five.

The silicon compiler is described in this chapter with reference to only one of the 5 chips in the set. Appendix C contains the full set of designs. The floor-plans of the integrated circuits are included, but it was found impossible to reproduce the masks for the NMOS process in the format of a Ph.D thesis.

7.1 Bit-serial Arithmetic

In parallel arithmetic, all the bits of an input word are presented to an arithmetic element simultaneously. The output bits appear simultaneously and the internal logic is complicated because of the number of logic variables involved. In bit-serial arithmetic the bits of a word are presented sequentially, one bit at a time, starting with the LSB. The output emerges sequentially, after a latency delay. Internal logic is simplified because arithmetic operations usually only have to deal with one significant bit of input at a time.

The simplicity of operation of a bit-serial arithmetic element leads to a considerable reduction in chip surface area. Whereas the TTL lattice implementation had one large chip which was the parallel multiplier, there are chips in the bit-serial lattice set which contain three multipliers in addition to other arithmetic elements.

The presence of numbers of elements on the same chip naturally leads to pipelining, where the outputs of elements are fed to other elements to form arithmetic operators. Shift-register delays are used to ensure that pairs of coefficients are presented to elements in synchronism. The other advantage of pipelining in this way is that new data may be input to a group of elements before previous results have emerged from the output. Thus a highly efficient structure is created in which all elements operate continuously and simultaneously.

Having developed a pipelined group of elements, or "operations", it may be used continuously for one operation, or multiplexed between many. An example is the lattice chip set, which represents one stage of a lattice structure. When it is acting as stage one, its input is connected to the signal input using a multiplexer element. When it acts as another stage, its input is the output from the operation on a previous stage, suitably delayed and synchronised using a shift-register. The PARCOR coefficient value is calculated recursively, using previously calculated PARCOR values. These are stored and input at the appropriate moment using long shift-register delays.

By contrast, the multiplier in the TTL equaliser was used in every operation of the arithmetic unit. This represents a higher degree of multiplexing. Since pipelining was difficult to achieve in the parallel hardware, the multiplier was only utilised for 33% of an arithmetic cycle.

In 1968 a design for a touch-tone dialling receiver was published [74], which used a bit-serial multiplexed architecture. In 1981, R.F. Lyon [59] proposed an architectural philosophy for fast implementation of signal-processing designs, based on a bit-serial

element structure.

7.2 The FIRST Silicon Compiler

A silicon compiler is one which takes an algorithm definition in a high-level language and translates it into a design for an LSI chip which will execute that algorithm.

The FIRST system has a library of silicon designs corresponding to elements. These may be fixed designs, as in adders, or variable ones, such as shift-registers of various lengths. The high-level language instructs the compiler to place certain of these designs onto a chip and interconnect them in a certain order.

The interconnections are made using signal names. The high-level language consists of a series of calls to elements or "primitives" (Fig. 7.1), as other languages make calls to subroutines. The signal names are used as arguments to the calls, to say how the element is to be wired to others.

The library of primitives has all basic arithmetic elements likely to be used. Nevertheless, it is possible to add to the library when necessary. Groups of elements are called "operators", which are themselves able to be called and used as units. A chip is defined by specifying the elements and operators present on the chip, along with the interconnections as signal names in the arguments to the calls. In a similar way, chips are assembled into a subsystem, which can be tested as a whole on a software simulator.

7.3 System Design

7.3.1 Methodology

The lattice chip set was designed in seven stages. First the algorithm was specified, defined by table 5A. The stepsize recursion presented problems as it involved a division. Since no division element is yet available to the silicon compiler, the recursion was inverted using a Taylor expansion, as described in section 7.3.2

Having defined the algorithm, it was divided into logical units. The lattice structure algorithm naturally divides into the signal path, the PARCOR recursion and the stepsize recursion. Flow diagrams were then drawn for each logical unit. Where a unit included a recursion, it was left outside the diagram. Fig. 7.2 shows the flow diagram for the lattice signal path. The delay implicit in the backward channel equations has also been left outside.

The flow diagrams were then turned into operators by adding synchronising delays and a control network. In the FIRST bit-serial system, each data signal path is accompanied by a second path containing control impulses. The control impulse synchronises an element, a multiplier for instance, by arriving at the same time as the LSB of the input signals. In Fig. 7.2 the signal path has been delayed to synchronise with the signals passing through the multipliers. Both inputs to the adders then arrive at the same time as the adder control pulse.

Fig. 7.2 also introduces the "wavefront rule" used to simplify and thus shorten the design work. Under the rule all inputs to an operator enter synchronously. Outputs are also synchronous. This assisted timing calculations during design. The rule was also extended to chips so that it would be easier to test physical packages.

The operators were then assembled into chips. Usually one chip contained one operator and a few extra arithmetic elements. The input and output pads were also specified in the chip description, along with the order of the pads around the chip. Judicious ordering of the pads greatly simplifies pin bonding and later printed circuit design.

Fig. 7.3 shows the lattice chip flow diagram. Normally recursion within a chip was avoided but here it proved necessary for system timing. A multiplexer is used to ground the input to the sample-length delay for system initialisation. The limiters are on the chip to truncate the coefficient of a later multiplication. Since the

physical size of a multiplier is related to its coefficient length, this makes it smaller and limits system word-growth.

The penultimate stage was to add recursion and multiplexing to the system. For the stepsize and PARCOR recursions this involved designing operators to return the output to the calculation input after a delay of precisely one input sample. The system was multiplexed over 16 stages by bringing the output of one stage to the input of the next in the lattice chip recursion. The loop was broken by a multiplexer for every new input signal sample. Multiplexers were used to initialise recursive loops.

Finally the precisions of the words were determined at every point in the system. This allowed word-growth and coefficient length to be monitored. At this point new elements were added as necessary, to truncate and limit words to the correct format.

The design of the chip-set was then assembled into a system description ready for testing. Fig. 7.4 shows the chip-set with interconnections. Fig. 7.5 shows the finished floor-plan of the lattice chip, showing the elements arranged around a field of interconnection tracks and the chip pads on the perimeter.

7.3.2 The Stepsize Recursion

The simulations of chapter four showed that a PARCOR recursion stepsize of 1/40 relative to unity input power was a reasonable value. In some cases [38] the stepsize is started at a high value to give a rapid initial convergence. The stepsize value then returns to more normal levels as its own recursion (equation 3.27) progresses. After the coefficients of the structure have converged to values close to their optimal ones, the stepsize is reduced even further. With this low stepsize, typically 1/100 relative to unity input power, the filter can track gradual changes in its input signal statistics.

For IC designs using the silicon compiler, equation 3.27 was not suitable, since it involved an inversion. There is no arithmetic

element in the library of primitives capable of dividing or inverting, so an alternative technique was found. The first-order Taylor expansion of equations 3.29 and 3.30 is:

$$\mu(n, t+1) = \mu(n,t).(1 + \alpha) - \beta.\mu^2(n,t).(1 + \alpha)^2.s^2(n,t) \quad (7.1)$$

This is equivalent to increasing the stepsize by a small percentage, and then reducing it by an amount proportional to the power estimate. Clearly the $(1 + \alpha)$ term in the first term on the RHS of the equation is essential to the recursion. The $(1 + \alpha)$ term in the second term, however, may be dropped, causing only a slighter reduction in accuracy. This gives the recursion:

$$\mu(n,t+1) = \mu(n,t).(1 + \alpha - \beta.\mu(n,t).s^2(n,t)) \quad (7.2)$$

Simulation showed that this Taylor stepsize recursion was slightly less stable than that of equation 3.29. For given values of α and β, the stepsize had a slightly higher variance for a given data sequence. It was nevertheless decided to simplify the calculation further by setting β equal to unity and dropping it from the calculation.

Two further problems arose: With a zero signal input, the value of μ would ramp upwards indefinitely. Furthermore, if a noise impulse on the signal input raised the power estimate term above unity, the stepsize would go negative. It would not normally recover from this abnormal condition. The problems were both solved using limiters. One was placed within the stepsize recursion to limit the value of the stepsize to a maximum of 0.25. The other was put at the output of the power estimate subcalculation, to ensure that it never rose above unity. Figure 7.6 is a flowchart of the stepsize recursion used.

7.3.3 Decision Feedback

Decision directed feedback (DFB) is the means by which a modem equaliser can continue to adapt after its training sequence has ended. The output for the modem detector is assumed to be correct

and this is used as the training signal for the side-tap recursion.

The question of DFB was considered because it was intended at a later date to make an equaliser based on the lattice structure chip set. In this context it was discovered that the bit-serial multiplexed philosophy of the FIRST silicon compiler did not accommodate DFB very well.

The lattice chip-set is designed to process one stage of a filter. The set is multiplexed to process each stage in turn. However, the pipelining implies that the outputs from the final stage are valid when the initial stages of the filter have already started to process the next signal input. With DFB, however, the training signal value can only be obtained after the output from the final stage. The value of the training signal must then be used to update the coefficients of the first stages before another signal sample is accepted.

There are two solutions to the problem, neither of which is elegant. The filter may execute a number of empty cycles after each output, to permit the coefficient calculations to catch up. Alternatively, the filter could use an old coefficient value for its calculations. The former solution would reduce processing speed appreciably, while the latter would reduce stability and increase algorithm noise.

7.3.4 Arithmetic Precision

The TTL hardware used fractional two's complement arithmetic to precisions of 12 or 24 bits. All add functions used limiting to prevent range overflow. Unfortunately there are several reasons why this simple scheme of range definitions does not work in a FIRST implementation.

In the FIRST chip set, each arithmetic element in the calculation is a separate area of silicon. It is no longer practical to follow each additional element with its own limiter as one may do in a central processor. This implies a less rigorous control over

word-growth, which must be allowed for in later stages. The area covered by a multiplier is dependent upon the precision of its coefficient. It is therefore desirable to minimise word-length while retaining accuracy of calculation.

The reciprocal relationship between input power and stepsize renders fractional two's complement arithmetic impracticable. When the input power to any stage is low, the stepsize must grow to a value sometimes exceeding unity. This problem was not encountered in the TTL version, as a recursive stepsize was not provided. Word-growth in the lattice signals also causes numerical significance to occasionally exceed unity.

The lattice signals are limited by the ADC to a range of ± 1. Word-growth down the lattice would take the signal outside this range and as bits may only be added in pairs, a range of ± 4 was allowed. The twelve-bit accuracy of the ADC was used throughout the signal path.

Having established the range of the signal, the ranges of the other variables were calculated. First the minimum rms value of the variable's variance was calculated for all operating conditions. The precision of the variable was then set three bits below their value. For example, the signal power-stepsize product determines the level of the algorithm noise on the PARCOR coefficient. For the PARCOR recursion to work properly, the precision of the coefficient must be better than the rms value of this stochastic process. This set the precision of the PARCOR coefficients to an accuracy of 2^{-17}. The PARCOR coefficients were also used in truncated form in the signal filter section, where accuracy could be traded for a reduced chip size.

The maximum possible values of variables were also calculated in order to establish the word-length of the signal. The word-length was taken as the range between the maximum value and the precision of the variable.

When calculating multiplier coefficients precisions, the word-

length of the variable was used. In some cases this resulted in too high a coefficient precision and the word-length had to be reduced. In some cases the maximum theoretical value was statistically unlikely and a maximum equal to eight times the maximum rms value was taken. If Gaussian statistical behaviour were assumed, this arbitrary threshold would be crossed once every $\sim 10^{15}$ calculations.

Single-precision multipliers truncate or round an output to minimise word-growth. At certain points in the stepsize and PARCOR recursion, this would have left the minimum rms value of the output below the multiplier output precision. The problem was solved using a double-precision multiplier and reformatting its two output streams into one word once more.

The system word-length is the maximum precision to which a number may be maintained within the system, without using multiple bit-streams. The system word-length is determined by the delay around the f-channel of the lattice filter, where the signal must double back around one stage to become the input to the next. This delay established the system word-length as 26 bits. This easily accommodated the maximum precision actually required for variable values, which was 19 bits.

7.4 System Tests

System testing fell naturally into two phases: fault-finding and result collection. In the fault-finding phase a simulator was used to check that the design contained no trivial mistakes. In the result collection phase, test signals were entered into the simulator model of the chip to evaluate the performance of the design. The latter phase may also be regarded as fault-finding but on a far deeper level. Ideally these tests should be followed by a redesign of the chip set, before finally committing the set to silicon. This was not, however, done.

7.4.1 Fault-finding

The FIRST software suite has a simulator associated with it. Previous to the existence of the compiler, tests and checking were performed on paper, using more cumbersome and error-prone software aids and on samples of chips themselves. This extended the time needed to design a chip, since mask-making and chip production can take upwards of six weeks, and the first test-chips may need modifications. The FIRST simulator, then, ensures that the silicon implementation is right first time.

Whereas the silicon compiler compiles an entire system, it generates a software description for the use of the simulator. The simulator uses this description to precisely simulate the operation of the system from one clock cycle to the next. The simulator will halt on serious errors, such as an incorrect system word-length. It may also print warnings about incorrect synchronisation between the elements. During simulation of operation, it takes input files, in which the designer has specified DC levels, noise level and various sinusoids, and inputs them to the system model. During operation, any node or system output may be monitored to ensure correct operation.

The system timing is a problem presented by FIRST silicon structures which is not normally encountered in conventional languages and compilers. Each arithmetic element is fed a control pulse, which arrives synchronously with the LSB of the input signal or signals. The designer must use delay elements to ensure that all signals arrive simultaneously. Should they not do so, the simulator will give an error message and continue with the simulation. The information in the error message may then be used to correct the synchronisation errors in later versions.

Having synchronised the chip-set, the next problem is arithmetic accuracy. Another problem with the FIRST system common to other systems is that of numerical significance. Since compiler and simulator perceive signals as simple streams of bits, the significance and compatibility of numbers is left to the designer to

calculate and decide.

Two main methods were used to detect design errors: A set of input data was fed to the section of circuit to be tested. Every element was then checked to ensure that its output was correct. Correct behaviour was then checked by feeding the section a test signal for a large number of iterations. Thus the stepsize recursion was checked and found to cause the stepsize to float at the correct value. The PARCOR recursion was tested by feeding the system a DC input and monitoring the PARCOR coefficient value so it asymptotically approached unity. After a fault had been corrected, it approached smoothly to within 0.005 of unity.

7.4.2 Performance Tests

After the first PARCOR coefficient had been found to converge smoothly on a DC input, a sinusoid was input to test the convergence of both first and second PARCOR coefficients. The next stage was a composite signal, comprising 5 sinusoids and a little noise. This test signal is described in section 6.3.2.2.1. The object in this test was to determine the accuracy of convergence and not to test the rate at which the system could converge. Accordingly the stepsize was started at a high value and gradually reduced over many iterations. The system was run for 1000 iterations, so that it was certain that steady-state values had been attained. The final steady-state stepsize was set to $1/(166.Q(0))$, where $Q(0)$ is the mean input signal power.

Fig. 5.3 shows the PARCOR coefficients of the simulator model converging onto the test signal. The effect of the reducing stepsize on the fluctuation in the signal is clearly visible in coefficients K(1) and K(4). The coefficients move straight to their steady-state values and fluctuate about them, the variance of the fluctuations reducing gradually with time and stepsize. The greater fluctuation in coefficients K(8) and K(12) are as a result of the build-up of noise down the lattice, as described in section 5.2.3.

Fig. 7.7 shows the autoregressive estimate of the spectrum, based

on the values of the PARCOR coefficients after 1000 iterations. It appears identical to figure 6.14, which was calculated using exact, rather than estimated signal parameters. Fig. 7.8 is a graph of the sixteenth-order spectrum only.

The frequencies of the spectral peaks show a mean deviation from nominal of -12.6 Hz, entirely satisfactory when it is considered that the resolution of the Fourier transform is 67 Hz. The maximum deviation from nominal was 23.6 Hz.

The amplitudes of the five spectral peaks were measured in decibels and the mean was taken as a reference point. The maximum deviation from the mean of a spectral peak was +3.9 dB. The noise floor was estimated at approximately -47 dB, a value 3 dB below its nominal value.

7.5 Summary

This chapter has described a method of designing and testing signal processing algorithms. The advantage of this system is that it is possible to verify a set of integrated circuits long before they are processed and produced. The lattice chip-set described here was designed with an NMOS process in view. The five chips have sizes of approximately 5 mm x 5 mm when a 5-micron feature size is employed. The maximum sample rate will then be approximately 20 KHz. Should a 2.5-micron CMOS process be employed, the whole structure will fit on one 6 mm x 6 mm chip and be capable of an 80 KHz sample rate.

The bit-serial arithmetic approach enables a higher level of refinement in algorithm design, as the bit-precisions throughout the circuit may be individually set by the designer. The simulator of the silicon compiler is useful in its own right as a method of evaluating algorithms in finite-precision implementations.

This chapter also reports the recasting of the stepsize recursion so as to eliminate the division operator. The division operator would have been a major stumbling-block to the implementation of the

algorithm in any technology. Its replacement with a Taylor expansion is a significant step in the development of the gradient lattice algorithm.

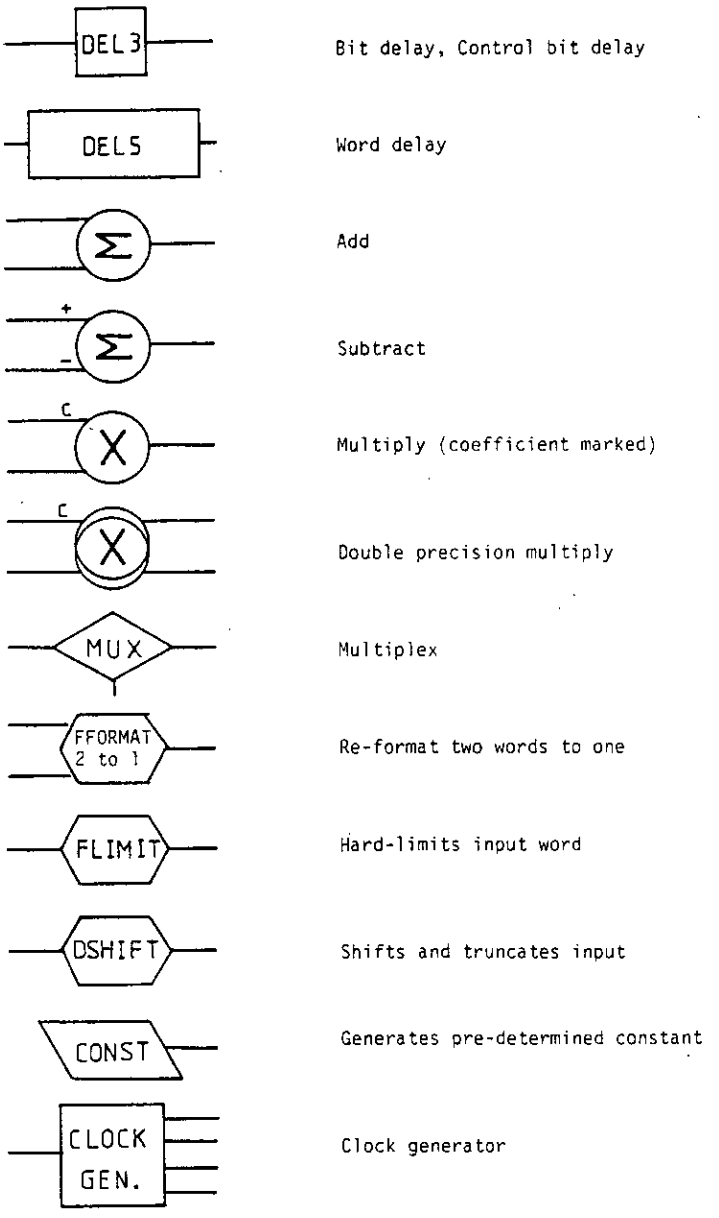


Fig 7.1 Primitives used in operators

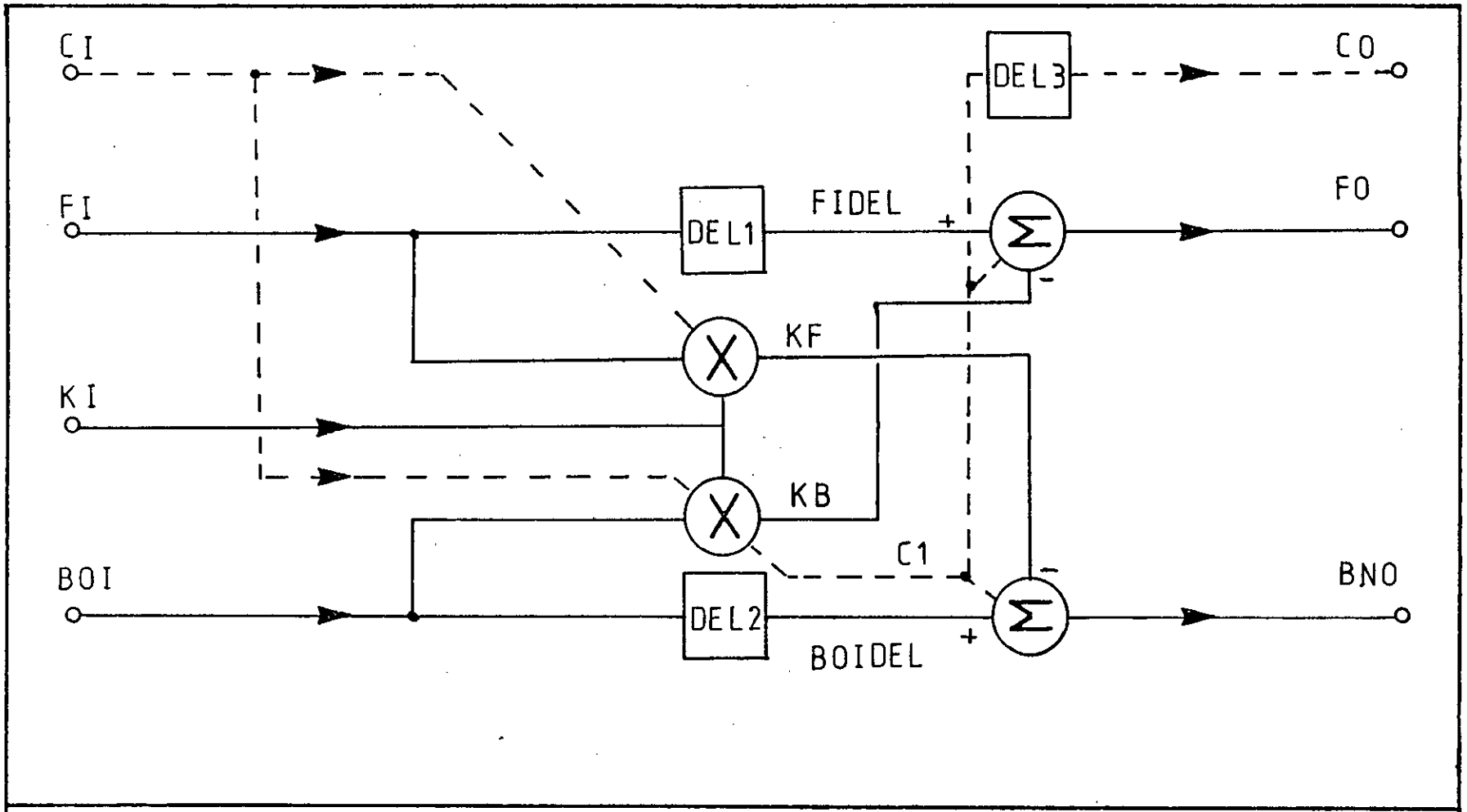


Fig 7.2 Lattice operator flowchart

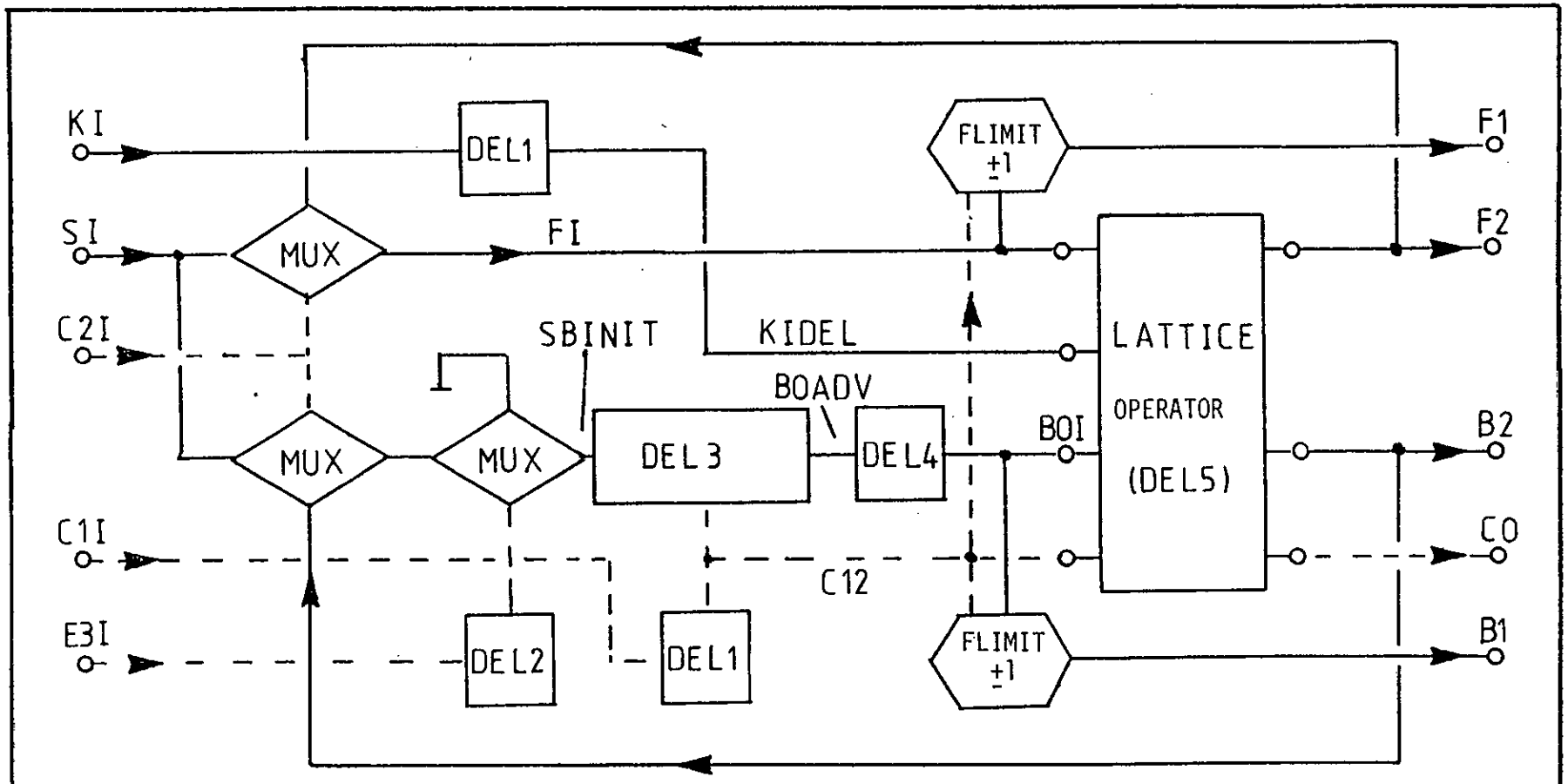


Fig 7.3 Lattice chip flowchart

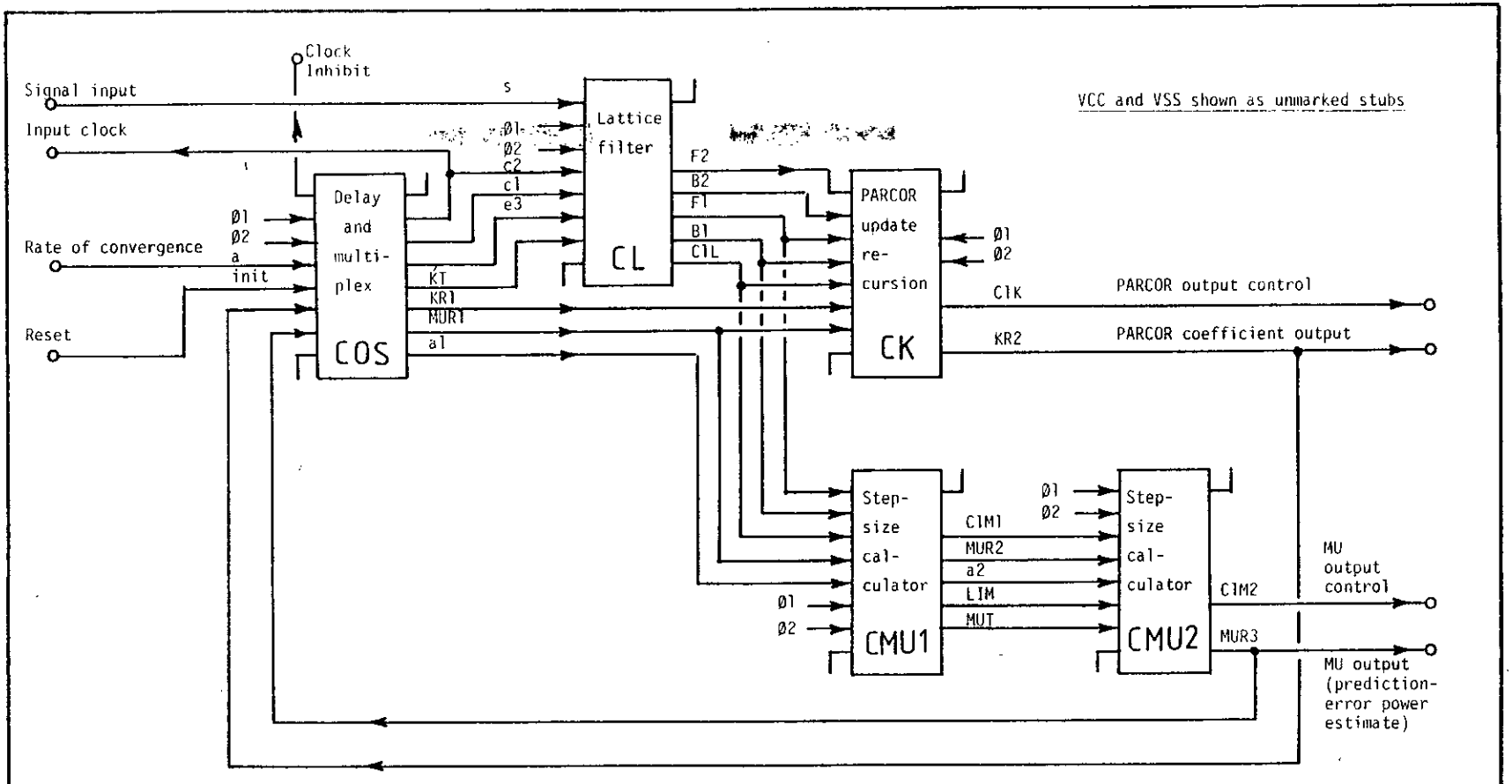


Fig 7.4 Lattice PE system, chip interconnections

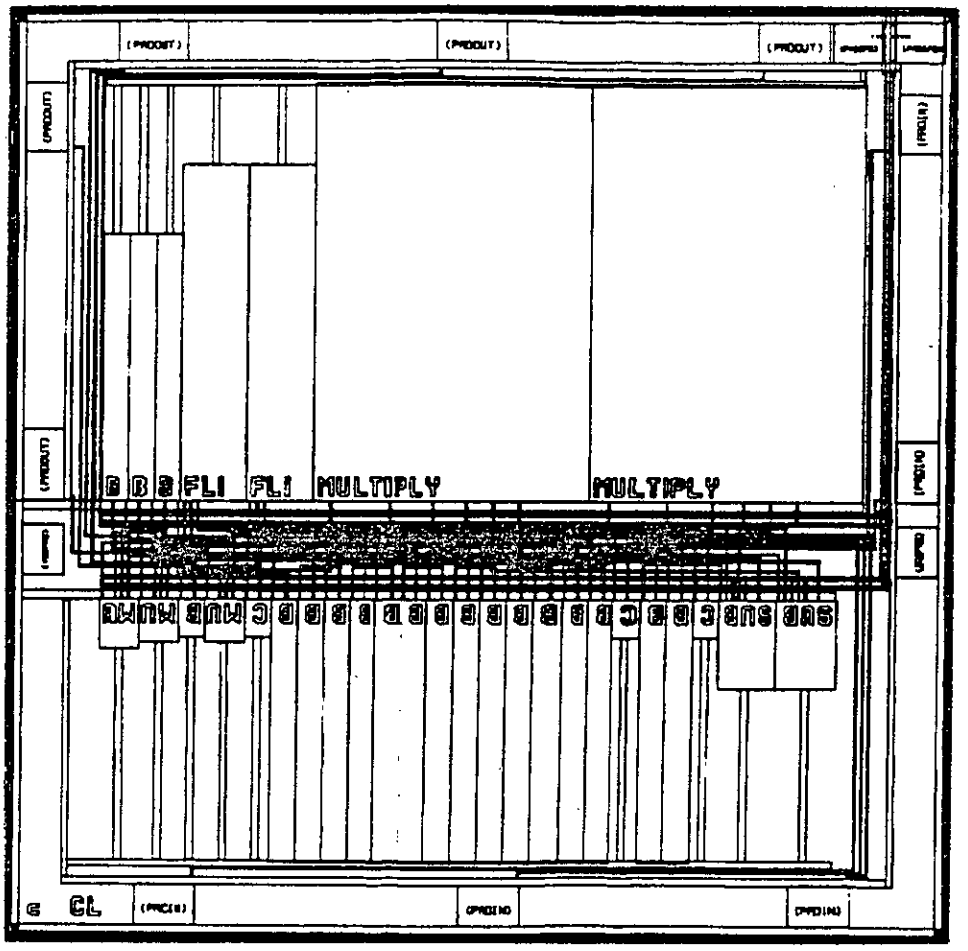


Fig 7.5 Lattice chip floor plan

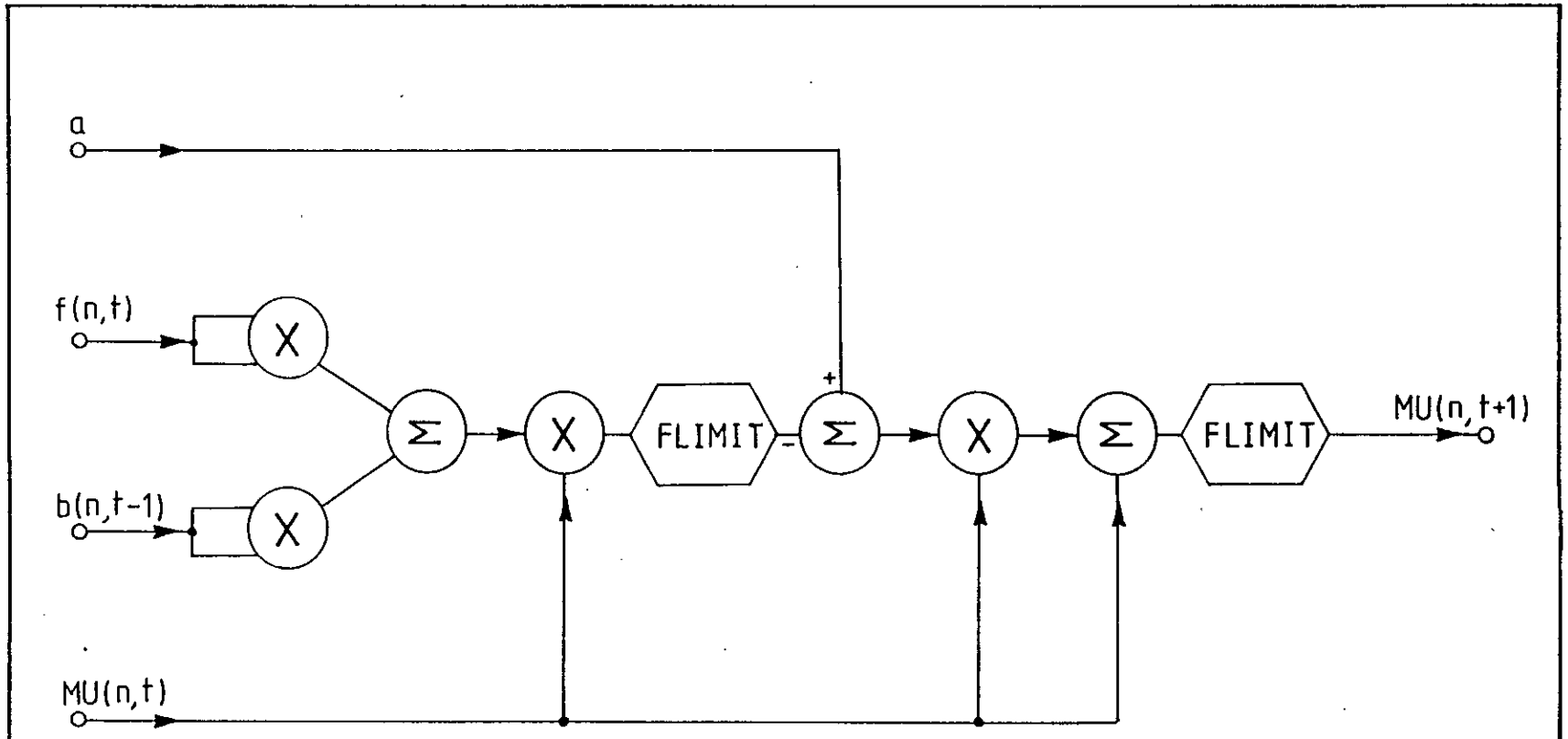


Fig 7.6 Stepsize recursion flowchart

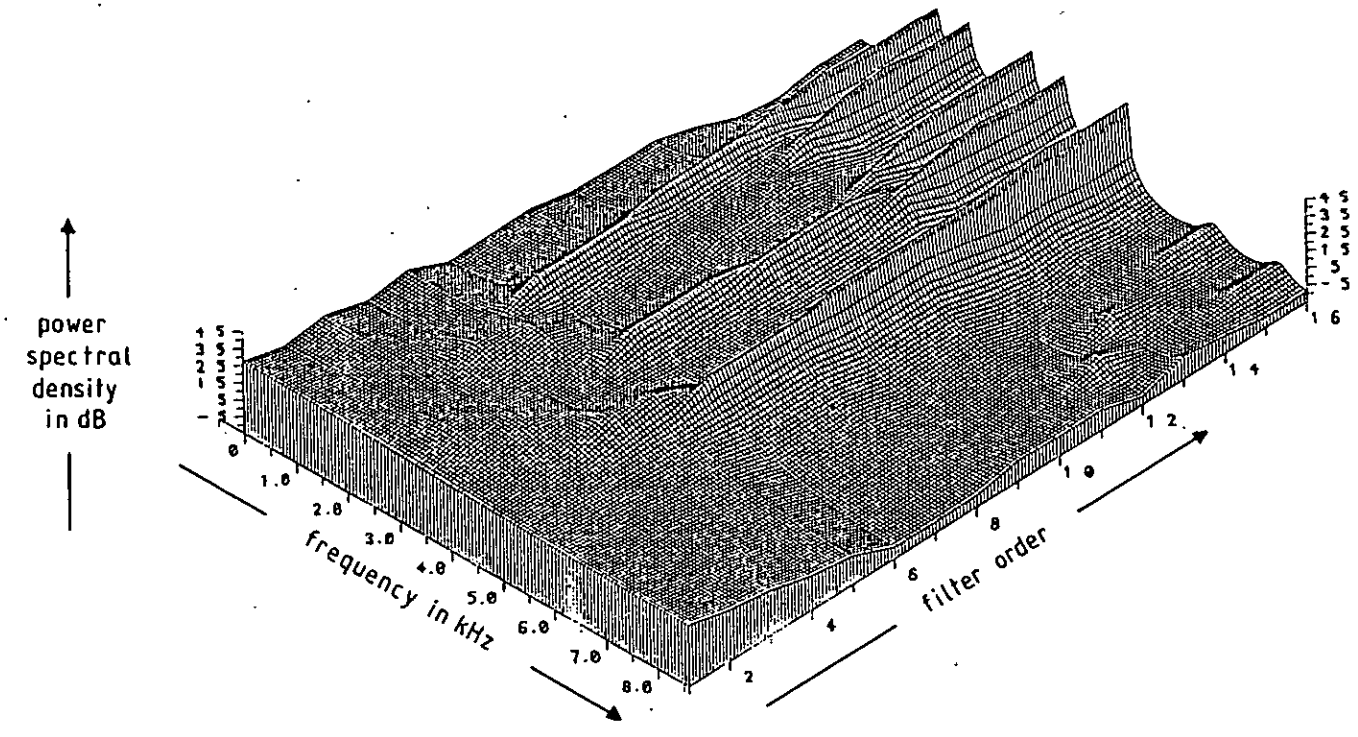


Fig 7.7 AR spectral estimate of 5 sinusoid signal, based on PARCOR coefficient values from FIRST simulation

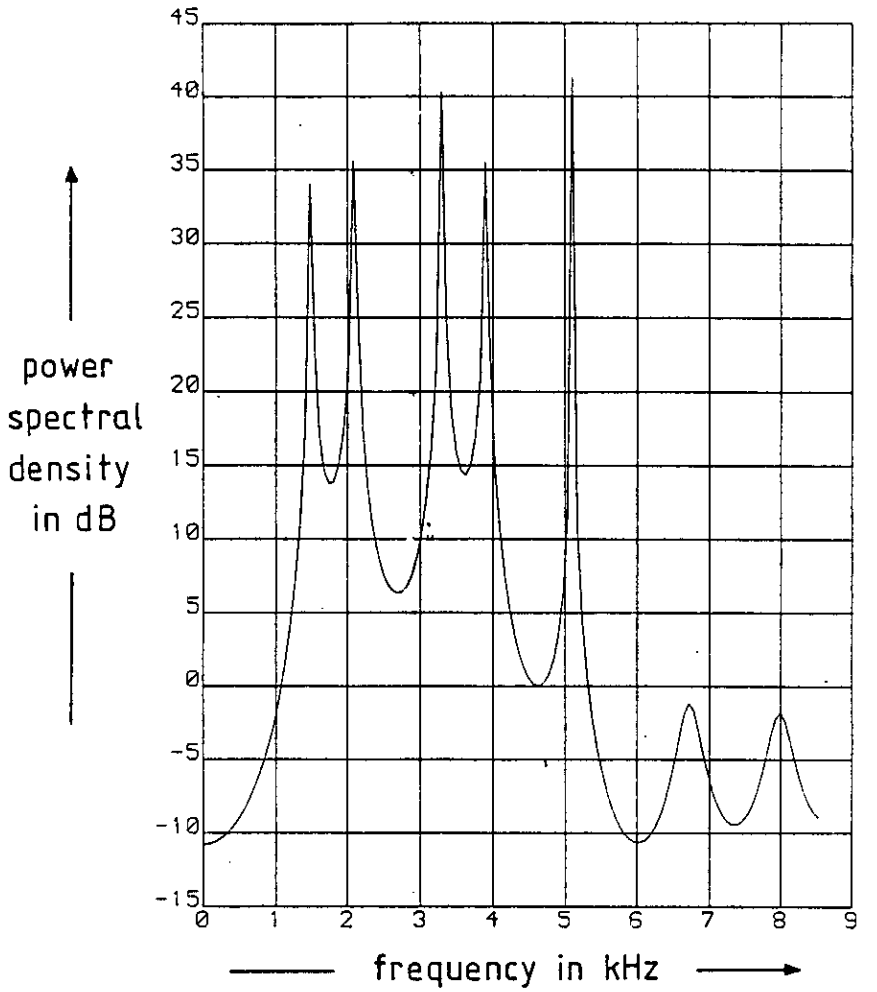


Fig 7.8 16th order AR spectral estimate of 5 sinusoid signal, based on PARCOR values from FIRST simulation

Chapter Eight

LATTICE FILTER APPLICATIONS IN SIGNAL ESTIMATION

The body of published literature on lattice filters is considerable. This thesis has concentrated on gradient adaptive lattice designs and their applications in data equalisation. This chapter attempts to expand their applications by giving an overview of other relevant signal processing areas. The chapter is loosely partitioned into frequency domain, time domain and other application areas.

In the section covering the frequency domain, the subjects of linear prediction, whitening and enhancement are discussed. The first is used mainly for speech processing, while the remainder are used in various aspects of radar signal processing. The time-domain applications covered are the use of the equaliser as an adaptive filter and the PE structure as an echo canceller. The other miscellaneous examples are a non-adaptive lattice ARMA structure and a direction-finding technique suitable for passive SONAR.

8.1 Frequency Domain

The frequency domain applications almost all depend on the spectral whitening effect of the lattice structure. This was discussed in depth in section 2.3.7. A major application is spectral estimation by AR modelling. The AR model may then be analysed into a function of spectral density, as in conventional AR spectral estimation, or a collection of sinusoids on a white background, as in Pisarenko harmonic decomposition.

8.1.1 Linear Predictive Coding (LPC)

In the linear predictive encoding of speech [61], use is made of the fact that the human ear gains information from the power spectral content of speech. The ear is relatively insensitive to phase,

although secondary phenomena, such as the stereo effect, may depend on it. Thus it is not necessary to transmit the full spectrum of a speech signal for it to be understood; all that is needed is an approximate outline of the spectrum, plus a few details concerning the excitation of the spectral envelope. These details are whether the sound is voiced or unvoiced, its pitch and its amplitude. In frequency domain vocoding, the spectral outline is measured from the power output of a bank of bandpass filters [60]. In LPC, the information is measured and transmitted as a set of coefficients for an AR model. Speech encoding or "vocoding" is a method of reducing the bandwidth of the transmitted signal.

Fig. 8.1 shows a typical LPC system. The spectral envelope is estimated as AR coefficients in the PE coefficient estimator. This may be an adaptive lattice structure or it may operate on the autocovariance function of a frame of data. Data is operated on in "frames" of 30 ms, since it is assumed that the nature of the speech signal will not change significantly within that period (a strong analogy with the optical flicker frequency).

The advantage of using an adaptive lattice in this application is that the power estimate of the excitation signal may be calculated directly from its output. This excitation power estimate determines the amplitude of the excitation signal for the AR synthesising filter. Another advantage of the lattice is that the AR filter may also be of lattice form. The AR lattice will be discussed in section 8.1.3.1. The PARCOR coefficients may then be transmitted and used directly, without any transformation.

A voiced/unvoiced decision must be made for the selection of synthesiser excitation type. On the basis of this decision, the synthesising filter is excited with either a repetitive impulse or white noise. Having made the decision that the segment of speech is voiced, the pitch must be calculated and relayed to the impulse generator. A method of deciding excitation type is to examine the autocorrelation function of the input signal [62], especially the first term. If its value lies below a certain threshold, the input signal is regarded as unvoiced. Above the threshold, the signal is

regarded as voiced with the value of the autocorrelation term indicating the pitch information.

8.1.1.1 Helium Speech Unscrambling

When divers dive to great depths they do not normally breathe air, for medical reasons they breathe a helium-oxygen mixture. The lower density of the gas in their vocal tracts makes the resonances of speech rise in frequency. Since communication is vital for survival during emergencies, a helium speech decoder is an important safety item.

A feature of helium speech is that the excitation of the vocal tract by the vocal chords does not depend on gas density and therefore does not alter in frequency. An unscrambler must lower the resonant, formant frequencies while leaving the excitation frequency unaltered. An alteration in sampling frequency will not alone suffice.

An LPC-based unscrambling technique has recently been developed [63]. It operates in 3 stages, as shown in Fig. 8.2. First the input signal is whitened using a PE filter to leave only the excitation frequency. The spectral envelope of the speech signal is also calculated and transformed to that of normal speech using a non-linear transform. The AR synthesising filter coefficients are then calculated and the filter excited using the previously derived excitation signal. The output of the synthesising filter then resembles normal speech. The advantage of this method is that the original excitation signal is retained, so voiced/unvoiced decisions do not need to be made [64].

In the process shown in Fig. 8.2, the PE filter coefficients are calculated via the first twelve lags of the autocovariance function. The power spectrum is calculated by doing a Fourier transform on a 30 ms frame of data. After the non-linear transformation another Fourier transform is performed to obtain the first twelve lags of the normal speech autocovariance function. The AR filter coefficients are then calculated via the Levinson-Durbin recursion.

An adaptive lattice structure would be computationally more efficient, as shown in Fig. 8.3. The lattice alone would serve to whiten the incoming signal to separate the excitation signal from the spectral envelope. The coefficient values of the lattice would be transformed to give an AR estimate of the power spectrum. The spectral warping and AR filter synthesis would then be performed as normal.

The process of Fig. 8.2 has already been demonstrated to give more superior results than any other currently in use. It is hoped that after further development, an adaptive lattice will aid underwater communications in the North Sea.

8.1.2 Spectral Whitening

The prediction error filter's spectral whitening properties were described in section 2.3.7. This adaptive whitening in the frequency domain is similar to the effect of AGC in the time domain. Any spectral features are detected and removed as the adaptive filter adapts its transfer function.

In dynamically changing signals, the stepsize is analogous to the time-constant of an AGC system. Thus a PE filter with a long time-constant would attentuate a keyed CW signal as though it were a continuous CW signal of proportionally lower power. A fast-reacting filter would notch an FM signal as though it were a sine whose frequency was slowly changing; a slowly-reacting filter would perceive it as a broadband signal, however.

In general, a whitening filter improves broadband signals at the expense of narrow-band ones. When the spectral density of the whole spectrum is normalised, power becomes proportional to bandwidth. Thus a whitening filter would automatically notch out unwanted heterodynes or other narrow-band signals from a wanted wideband transmission.

8.1.2.1 A Radar Application

The following section is paraphased from C.J. Gibson's Ph.D thesis and [30]:

An air-traffic control radar set transmits short monotonic pulses of energy and then listens for echoes from objects (such as aircraft) in the environment. The time delay between pulse and echo indicates the object's range. While this is happening, the antenna slowly scans in a clockwise direction to cover the entire area in azimuth.

The half power beamwidth of the antenna is typically two degrees and this determines the azimuthal resolution. The pulse width determines the spatial resolution, which is about 100 m. This enables the observed area to be split into cells whose size is determined by the resolution.

In typical radars between ten and twenty pulses may pass through a cell with each revolution of the antenna. An object in the cell may return an echo from each of these pulses, forming a time series representing the object. When detected coherently these echoes have amplitude, temporal displacement and frequency displacement. The temporal displacement indicates the range cell in which the object is located. The amplitude indicates its radar cross section and thus its approximate size. The frequency displacement (doppler) gives the object's radial velocity component with respect to the antenna.

Echo returns may be roughly divided into two types: targets and clutter. The target is the object of interest to an observer, such as a moving aircraft. It is usually small, sharply defined and moving with respect to the antenna. This means that its echoes are low in amplitude, sharply defined and usually have frequency displacement and a changing temporal displacement. Clutter is caused by items of low interest to an observer, such as the ground, sea waves, stationery objects and weather phenomena. They are often large in amplitude, less well defined and have low frequency or temporal displacement. Several processing techniques are available to minimise the clutter signals while retaining target signals.

In the moving target indicator (MTI) it is assumed that the clutter is stationary or moving slowly whereas targets are moving more quickly. Thus a high-pass filter will remove the clutter echoes while passing the doppler-shifted target echoes. The drawback of this method is that some weather phenomena create doppler-shifted signals, which pass the filter to obscure the target echoes.

Another anti-clutter filter uses a bank of frequency filters on an FFT to separate out the frequency components of the signal. If the frequency components are then each subjected to an AGC then slowly changing clutter signals will be reduced in amplitude whereas short target echoes will pass through before the AGC acts. Limiting the low-frequency bins will reduce clutter but may also obscure slow-moving targets. Indeed, since the number of bins is usually low (about 8), there is always a risk that a target will be obscured by clutter in one of the bins.

A PE filter with a slow time constant will have time to adapt to the slowly-changing clutter. Being a slowly-changing signal it will have a low prediction-error. The time-constant of the filter will not allow it to adapt to the quickly-changing target signal, which is passed by the filter.

Alternatively a PE filter with a fast time-constant will adapt to both target and clutter. Its output will then be proportional to the rate of change of frequency and amplitude of the echoes. Thus target signals will pass more readily than clutter signals, having a greater rate of change.

Compared with transversal filters in this application, lattice types have been reported to be stable with a lower quantisation noise, better resolution, plus the ability to alter filter order at will and provide an almost constant rate of adaption.

8.1.3 Spectral Enhancement

Spectral enhancement is the opposite of whitening or prediction error filtering. Instead of filtering an input signal in order to remove spectral features, an attempt is made to enhance them. This improves the signal-to-noise ratio of the signal. There are two main methods of enhancement: the adaptive spectral enhancer (ASE) and the adaptive line enhancer (ALE) [68]. An extension of the adaptive line enhancer is the smoothing filter reported in [67].

8.1.3.1 Adaptive Spectral Enhancement

When viewed in the power spectral domain, matched filtering [20] consists of amplifying the various frequency components of an input signal by amounts proportional to their spectral density. Thus spectral peaks are enhanced at the expense of the wide-band noise background.

Adaptive spectral enhancement is a subject currently under research by the author. It occurs in three steps. First the incoming signal is whitened in an adaptive PE filter. The filter coefficients are then transferred to an AR filter, which therefore has a spectral power transfer function which is the inverse of the PE filter. Thus, were the AR filter to be excited with white noise, the envelope of the resulting output power spectrum would resemble that of the input to the PE filter. Instead, however, the PE input signal is applied to the AR filter, so that it outputs a signal with enhanced spectral colouration.

Fig. 8.4 is a signal flow diagram of the ASE, consisting of a recursive IIR lattice in parallel with an FIR type. The equations of operation of the non-adaptive IIR filter are given in [65]. The coefficients for the IIR filter are calculated by the FIR adaptive lattice, ensuing adaptation down a parabolic error surface towards a global minimum. (Adaptive IIR filters do not always adapt towards global minimum [36]). The range limitation of ± 1 on the PARCOR coefficients ensures that the AR filter may approach the threshold of stability but never cross it.

Fig. 8.5 shows the spectrum of a signal used to test the ASE. It comprises an unmodulated sinewave at 35 KHz, a wideband signal centred on 88 KHz and a floor of white noise. It may be seen that the spectrum may be modelled by two pairs of poles. One pair models the sharp spectral peak, while the other models the broader spectral maximum of the wideband signal. Fig. 8.6 is the power spectrum of the IIR output. It shows that the SNR of each signal has been squared, ie that the SNR in dB has been doubled. A fourth-order IIR filter was used.

It is emphasised that the ASE does not attempt to emulate the phase characteristics of a matched filter, nor is it of linear phase. Instead it delays certain frequencies, adding a temporal distortion to the signal. In view of the high gain given to narrow-band signals, it is expected that the ASE could have applications in the detection of CW doppler radar for electronic countermeasure purposes.

8.1.3.2 Adaptive Line Enhancement (ALE)

Adaptive line enhancement is a method of enhancing narrowband components of an input signal [3]. Its structure is that of a PE filter, but the output signal is taken from a different point. Equation 2.31gives the output of the PE filter, e(t) as:

$$e(t) = s(t) + \sum_{i=1}^{N} a(i).s(t-i) \qquad (8.1)$$

Using the transversal PE model of section 3.5.1, equation 2.14 may be applied:

$$e(t) = d(t) - y(t) \qquad (8.2)$$

In this case $d(t) = s(t)$, giving $y(t)$:

$$y(t) = \sum_{i=1}^{N} a(i).s(t-i) \qquad (8.3)$$

The output y(t), then, is the output of the ALE.

The PE filter operates by trying as closely as possible to predict the next signal sample. It does so by coherently averaging the predictable components in the previous samples. This averages the predictable, narrow-band, components of the input signal, so as to enhance them. The wideband components in the signal, such as noise, are less predictable and will not be summed coherently. Thus the signal presented at the summation of equation 8.2 by the filter is a replica of the narrow-band content of the signal s(t). They will be of the correct phase and amplitude. The wideband content of the signal will be reduced by the averaging process.

An example of an application will illustrate the practical implications of separating wideband and narrowband components: Sometimes it is desirable to listen to speech which has been contaminated by background music [66]. Speech has a relatively wideband nature, communication being by the creation and detection of areas of greater spectral density. Music, by comparison, is a narrowband transmission, comprising sustained sinusoids. An ALE with a suitably selected tracking rate will segregate the signal into two outputs. The wideband, whitened output will have a lower musical content while the narrowband enhanced output will have a lower speech content.

It is an interesting feature of the ALE, that although the output signal, y(t), is an accurate replica of the phase and amplitude of the narrowband components of the current signal sample, s(t), the latter sample is not involved in the averaging process. The current sample is nevertheless used in the adaption process.

The adaptive filter model at the top of Fig. 8.7 illustrates the ALE as described in equations 8.1-3. The model in the centre of Fig. 8.7 is a lattice model of the ALE. Its principle of operation is illustrated by equation 8.2, turned round to form:

$$y(t) = s(t) - e(t) \qquad (8.4)$$

The signal e(t) is the prediction error from the forward channel

of the lattice structure. It is expected that the lattice would have better tracking properties for non-stationery signals as the convergence of the algorithm is not affected by the eigenvalue ratio of the autocovariance matrix of the input signal.

Fig. 8.8 shows the behaviour of the ALE when fed a sinusoid on a background of white noise. The figure shows the output of the filter in the power spectral domain. The eighth-order filter has placed its zeroes so as to maximally attenuate the noise, while leaving the signal unattenuated.

By contrast, Fig. 8.9 shows the performance on the composite signal of Fig. 8.5. The composite signal contains narrow-band and wideband signals, with which the eighth-order ALE can do little. The graph shows that the DC component has been removed, and the noise between spectral features reduced a little, but in general the SNR has remained largely unaltered. This demonstrates the essential difference in behaviour of ALE and ASE on wideband signals.

8.1.3.3 Spectral Smoothing

The backward channel of a lattice structure may also be used in an ALE. From equation 2.33:

$$e(t) = s(t) + \sum_{i=1}^{N} r(i).s(t+i) \qquad (8.5)$$

$$\therefore y(t-N) = (s(t-N) + \sum_{i=1}^{N} r(i).s(t+i-N))-s(t-N) \qquad (8.6)$$

This shows that in the case of the backward channel, a delay equal to the order of the filter must be applied to the input signal before the subtraction analogous to equation 8.4 is made. This is clearly a less efficient way of constructing an ALE.

Inefficient through the backwards ALE model is, it allows the performance of an ALE to be improved by averaging the outputs of backward and forward models. Using f and b subscripts to denote forward and backward filter outputs and a to denote the average:

$$y_a(t - N) = (y_f(t-N) + y_b(t-N))/2 \tag{8.7}$$

$$= \left(\sum_{i=o}^{N} r(i).s(t + i - N) - s(t-n) \right. \\ \left. + \sum_{i=o}^{N} a(i).s(t-i + N) - s(t-N) \right)/2 \tag{8.8}$$

$$= s(t - N) - (f(N, t-N) + b(N,t))/2 \tag{8.9}$$

The bottom model in figure 8.7 shows the lattice smoother, as such an averaged ALE is called. This structure was proposed in [67], though not in adaptive lattice form.

An interesting property of the smoother is that the coefficients of its impulse response are symmetrical. This implies that it is a linear phase filter for all signal components, not merely narrowband ones. In a way analogous to the ALE, the output corresponding to the input sample s(t) comprises an average of all samples either side of s(t) within a range of (t ± N). The signal sample s(t), however, is not represented in the average. The output corresponding to s(t) is output after a delay equal to the order of the PE filter.

8.2 Time Domain

There are two main lattice applications which are normally visualised in the time domain. These are the use of the equaliser as a general adaptive filter and the use of the PE structure as an echo canceller. In both cases the reliable convergence and tracking characteristics are the prime advantage over other techniques.

8.2.1 Adaptive Filtering

Much has already been written in this thesis on adaptive filters. This subsection will briefly sketch a number of applications with reference to the adaptive lattice equaliser. Applications may be roughly divided into direct and inverse system modelling.

8.2.1.1 System Modelling

Fig. 8.10a shows an adaptive filter in system modelling mode. The subtraction creating the error signal is shown outside the adaptive filter for greater clarity. In the first stage of this description let signal s(2,t) be zero. The minimum error signal will then be obtained when the adaptive filter impulse response matches that of the system as closely as possible. For very close matches, e(t) will be almost zero. The y(t) signal at the filter output will approximate the unknown system output.

Let s(2,t) be a signal orthogonal to s(1,t). The adaptive filter coefficients will have the same optimum values since the covariances of the Wiener-Hopf equation (2.18) remain unaltered. The error signal e(t) will then very closely approximate the input s(2,t). Thus, although the inputs to the summation of the system output and s(2,t) are not individually available, they appear separately at the outputs of the adaptive filter.

A typical application would be a signal contaminated by mains hum. The pure signal may be regarded as input s(2,t). If another, separate, sample of the hum may be obtained, it is the input to the s(t) (ie s(1,t)) input of the system. The model assumes that there exists some unspecified transfer function, the unknown system, linking the pure hum sample and that contaminating the signal. After convergence the y(t) output matches the hum on the signal, while the e(t) output is the hum-free signal.

Fig. 8.10b shows an imperfect 2-to-4 wire telephone hybrid. Normally the adaptive filter is not present and the output of the hybrid is taken directly to the 4- wire output. The hybrid is a form of bridge circuit, depending on accurate matching for its operation. When, as is usually the case, matching is approximate, then signal from the 4- wire input leaks round to the 4- wire output and causes a distant telephone user to perceive an echo on the line. The adaptive filter in system modelling mode adaptively removes any trace of the 4- wire input signal from the 4- wire output signal, removing the echo completely.

Algorithm noise in an adaptive filter is normally dependant upon the power content of the error signal [7]. In modelling applications the error signal power is normally high, so a very low convergence factor is needed in order to avoid tap-jitter.

An adaptive lattice equaliser structure would be of use in modelling when the s(1,t) signal had a high eigenvalue ratio. An example would be a music signal or noise hum, where the spectral energy is not evenly distributed. Furthermore, were the unknown system to gradually change, such as when a heart monitoring subject moves closer to a source of hum, a lattice would track the change more quickly than an equivalent transversal type.

8.2.1.2 Inverse System Modelling

Fig. 8.11 shows a diagram of an adaptive filter used in inverse system modelling. Here, the output of the unknown system, s(t), is fed to the filter input. The unknown system's input, s(1,t) is fed to the d(t) input. The solution minimising the error power is that the filter output resembles as closely as possible s(1,t), the input to the unknown system. Assuming that input s(2,t) is zero, the power content of the error signal should be very low after convergence.

The input to the adaptive filter will certainly not be white in the arrangement, so a lattice equaliser structure will converge and track faster than a transversal type. This mode of operation is that used in data channel equalisation, as discussed in chapters 2 - 6.

When deterministic system inputs are used, the Wiener-Hopf equation for an adaptive filter becomes insoluble as the inversion of a singular matrix is required (c.f. section 3.3.1). Under these conditions a transversal filter will converge to one of a family of possible solutions, only one of which is a matched filter solution. A lattice, however, operates using orthogonal transforms. If there are so few signal components that the input autocovariance matrix becomes singular, some sidetap signals will be set to zero (c.f. section 2.3.7.1). This reduces the order of the autocovariance

matrix to a rank which is non-singular and suitable for inversion. Thus a lattice equaliser will always converge to a matched filter solution.

It has already been shown that when s(2,t) is zero, y(t) is equal to s(1,t) and e(t) is small. If s(2.t) is fed a signal not correlated with s(1,t), the terms of the Wiener-Hopf equation will not be changed and the adaptive filter will continue to deliver s(1,t) at its output. Thus the e(t) output will closely approximate s(2,t). This is another variation of the noise cancellation model.

8.2.2 Multipath Cancellation

Radio channels with multipath are a class of echoing channels whose impulse response may be modelled by an AR filter [55]. This model is neither obvious nor universal. A channel with an echo on it is usually regarded as having an MA structure with the coefficients of the MA filter corresponding to those of the impulse response. Nevertheless, were an echoing channel able to be modelled by an AR structure, an PE filter would be able to invert it exactly, as discussed in section 2.3.7.

An AR model is subject to certain conditions governing its stability. If its transfer function is $F_a(z)$, where:

$$F_a(z) = 1/(1 + \sum_{i=1}^{N} a(i).z^{-1}) \qquad (8.11)$$

Then for stability:

$$\left| \sum_{i=1}^{N} a(i)\, z^{-1} \right| < 1, \text{ for any } z \qquad (8.12)$$

This will certainly be satisfied if:

$$\left| \sum_{i=1}^{N} a^2(i) \right| < 1 \qquad (8.13)$$

Thus the AR function is stable (ie minimum phase [28]) if, but not only if, the power of the echo is less than that of the first term of the impulse response. The first term of the impulse response

may thus be regarded as the main lobe and the echo following it is regarded as post-echo. Thus a signal with only post-echo may normally have its echo removed by an adaptive forward PE filter. In the rarer case of a signal with purely pre-echo, a backward PE filter will act as a canceller.

A lattice structure is an entirely suitable PE structure for echo cancellation in this case. Not only will it converge with a reliable speed but it will track changes in the nature of the channel. If too long an AR and PE filter model is used, algorithm noise will degrade the echo-canceller output. The lattice however has the advantage that the output may be taken from any stage. The optimum length of echo canceller may be calculated by monitoring the fall-off of the prediction-error power, Q(n), and terminating the filter where it ceases to fall significantly.

This echo canceller has a different application from that of section 8.2.1.1. The latter echo canceller can only operate at a single point in the circuit, cancelling the echo of the most troublesome component, the hybrid. A PE filter can operate at any point in the circuit. Typically it would be used immediately before the channel output to cancel as much of the echo as possible. However, it can only operate on channels which have purely pre- or post-echo, and cannot handle a channel with a mixture of echoes.

8.3 Other Lattice Applications

8.3.1 The Lattice ARMA Filter

Fig. 8.12 shows a lattice ARMA filter [69]. It is based on the recursive lattice structure shown in Fig. 8.4 . Just as it is possible to construct a one-multiplier non-recursive lattice stage, a one-multiplier recursive stage is also possible. The one-multiplier recursive lattice is said to have better round-off noise than the two-multiplier version [69].

Compared with shift-register based ARMA filters, the recursive lattice is said to have better round-off noise and greater stability.

Quantisation errors in shift-register AR filters tend to move the poles radially within the unit circle, threatening the stability of the filter. Quantisation errors in recursive lattices, however, move the poles tangentially to the unit circle, causing frequency errors instead.

8.3.2 Eigenvector Decomposition

Section 3.5.3 referred to an antenna beam-steering application in which an array of antennas is tuned to optimally receive transmission from one direction and to optimally reject those from others. The technique of eigenvector decomposition [70] uses the hybrid lattice structure to locate the bearings and signal strengths of those transmitters contributing to the received signal.

There is a duality between frequencies in a sampled data system and transmitter bearing in an antenna array [75]. Thus spectral analysis techniques may be used to locate transmitters. In the eigenvector decomposition method, the Levinson-Durbin recursion is applied to the coefficients of the hybrid lattice structure in order to form an AR model of the transmitters. Pisarenko harmonic decomposition (cf section 2.3.7.1) is then applied to the AR function in order to find the transmitter strength and bearings.

The term "eigenvector decomposition" is applied to the technique because the individual transmitter signals correspond to the non-zero eigenvalues of the input autocovariance matrix. The eigenvalues correspond to the strengths of the transmissions and the eigenvectors to their bearings. The key advantage to the approach over other Fourier based approaches to bearing measurement is that it can provide a superior resolution when operating with short blocks of data.

8.4 Summary

When such a wide range of applications is surveyed, the comments must of necessity be general. These applications are all in real-time signal processing. The spectral properties of the lattice allow

it not merely to whiten a signal, but to enhance the colouration also. The ability of the lattice to track the statistics of the input signal make it suitable for pre-processing applications, improving the signal-to-noise ratio before further processing is attempted. The tracking and whitening properties also make the lattice a suitable transform to stabilise the rate of convergence of an adaptive equaliser structure. In all, the lattice is a ubiquitous structure applicable to many fields of signal processing.

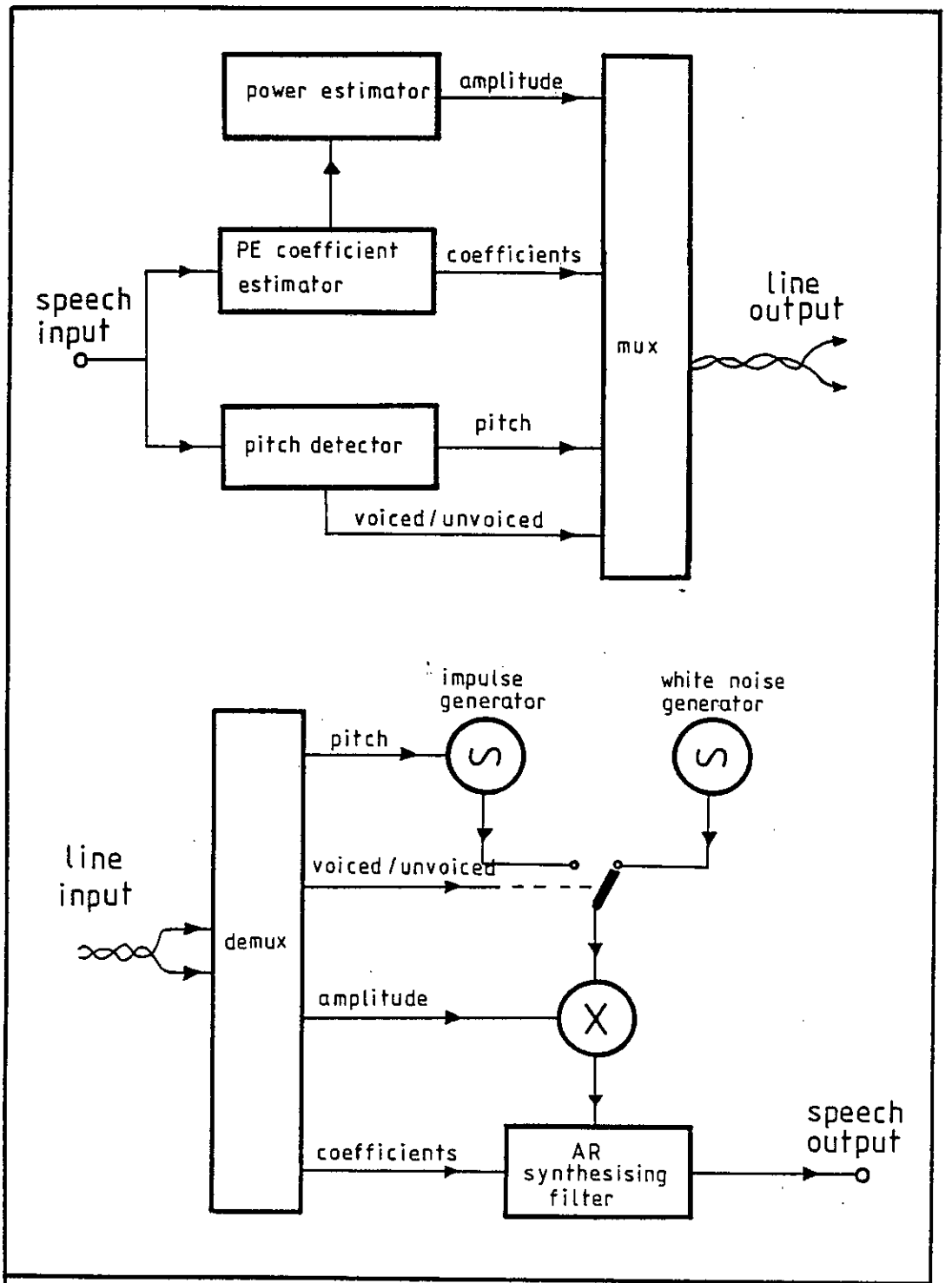


Fig 8.1 An LPC vocoding system:
(upper) Analyser and encoder
(lower) Decoder and synthesiser

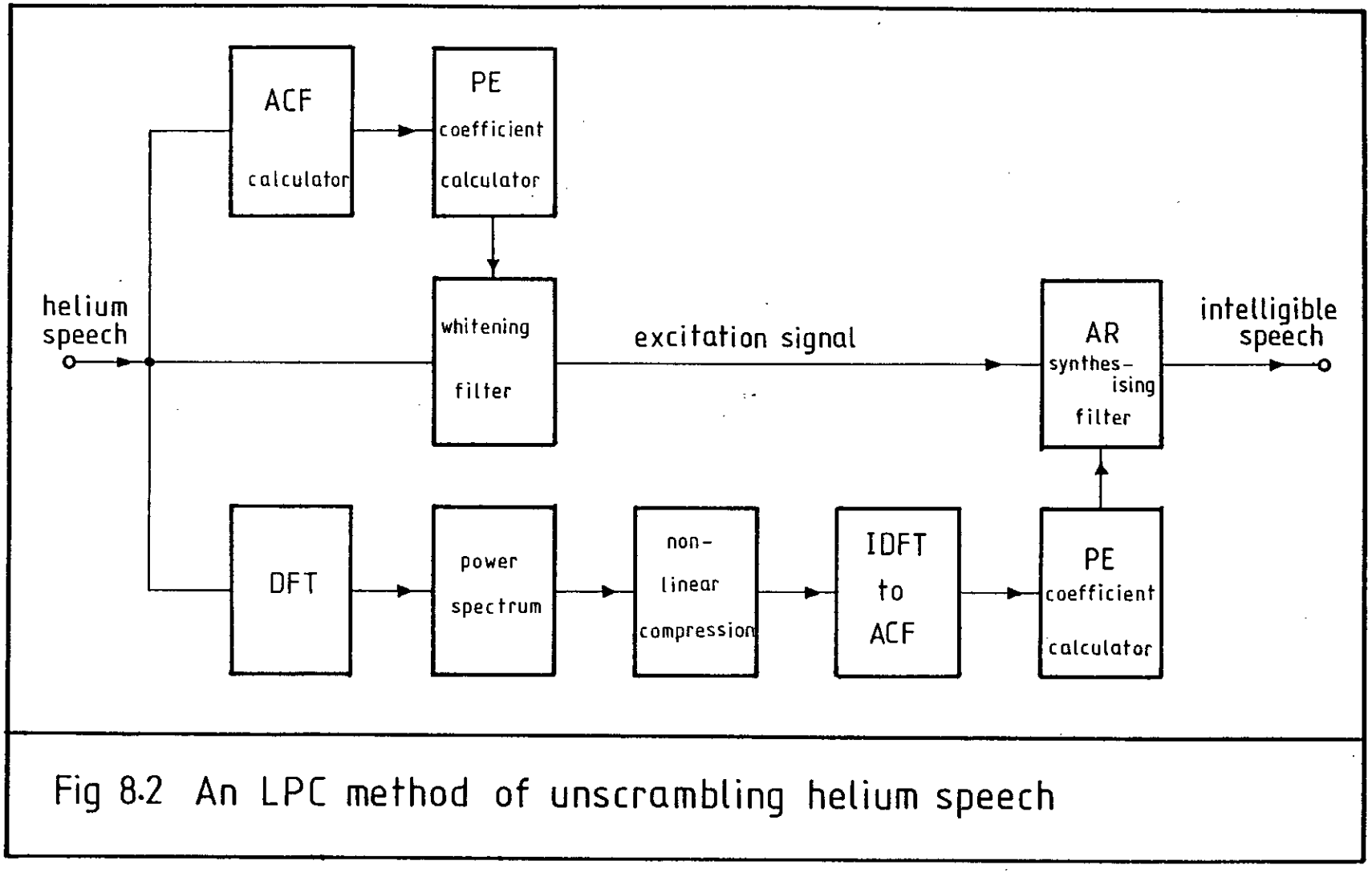


Fig 8.2 An LPC method of unscrambling helium speech

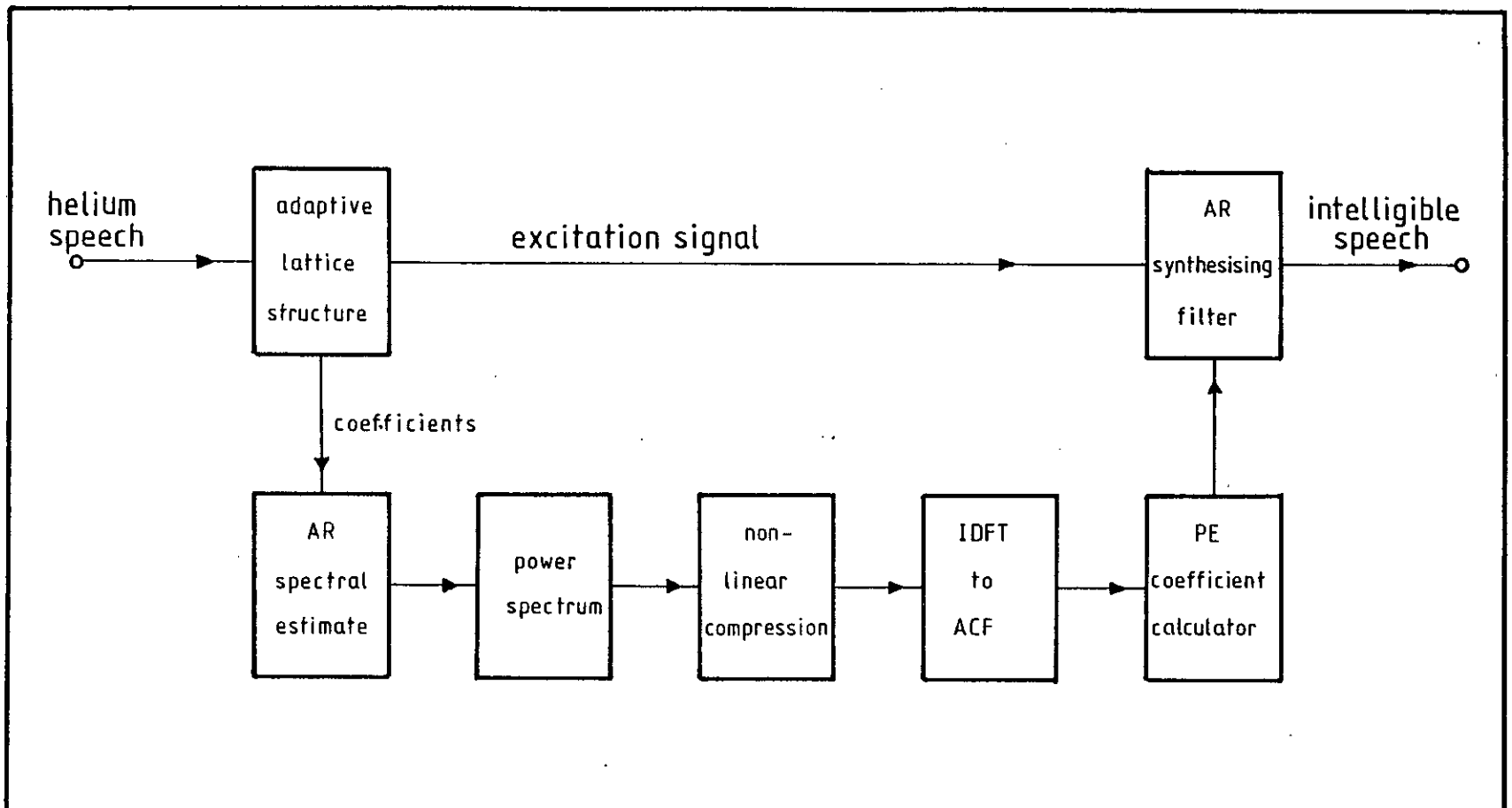


Fig 8.3 Helium speech unscrambling using an adaptive lattice structure

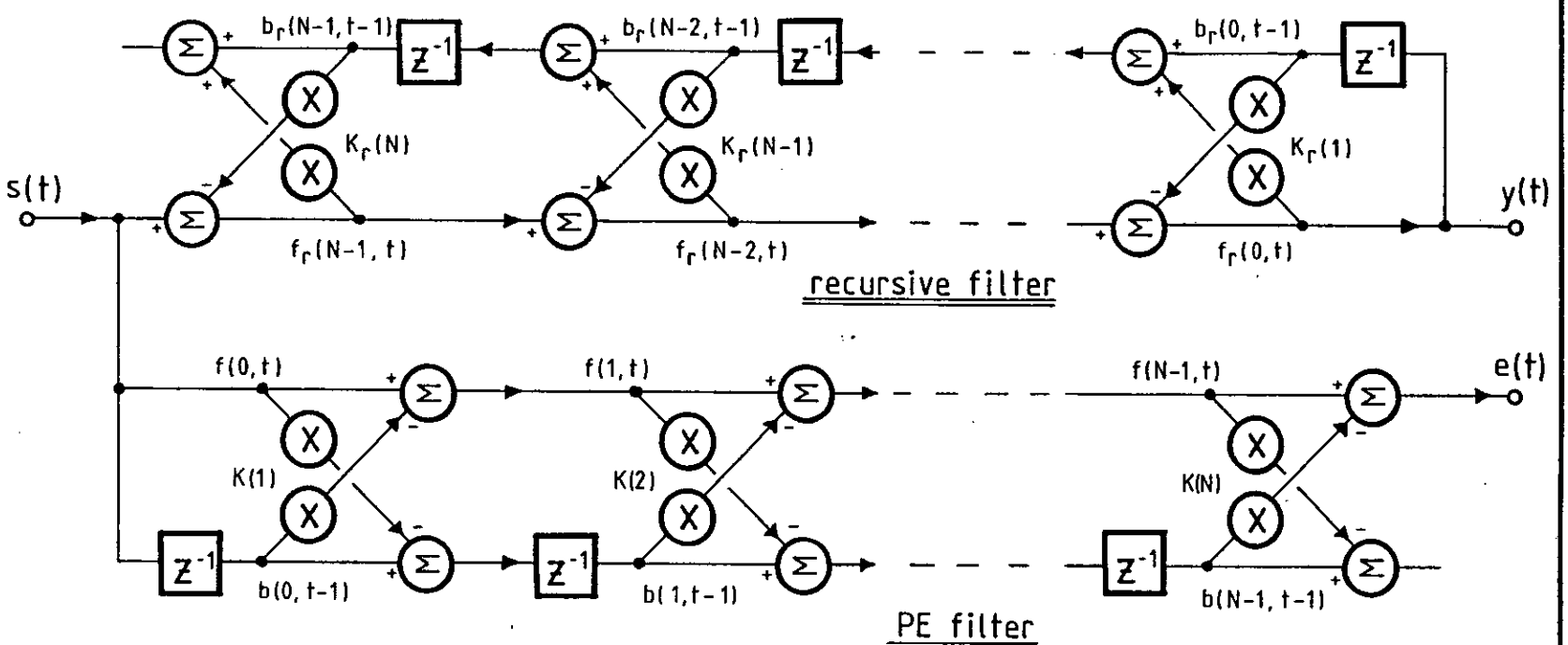


Fig 8.4 The adaptive spectral enhancer

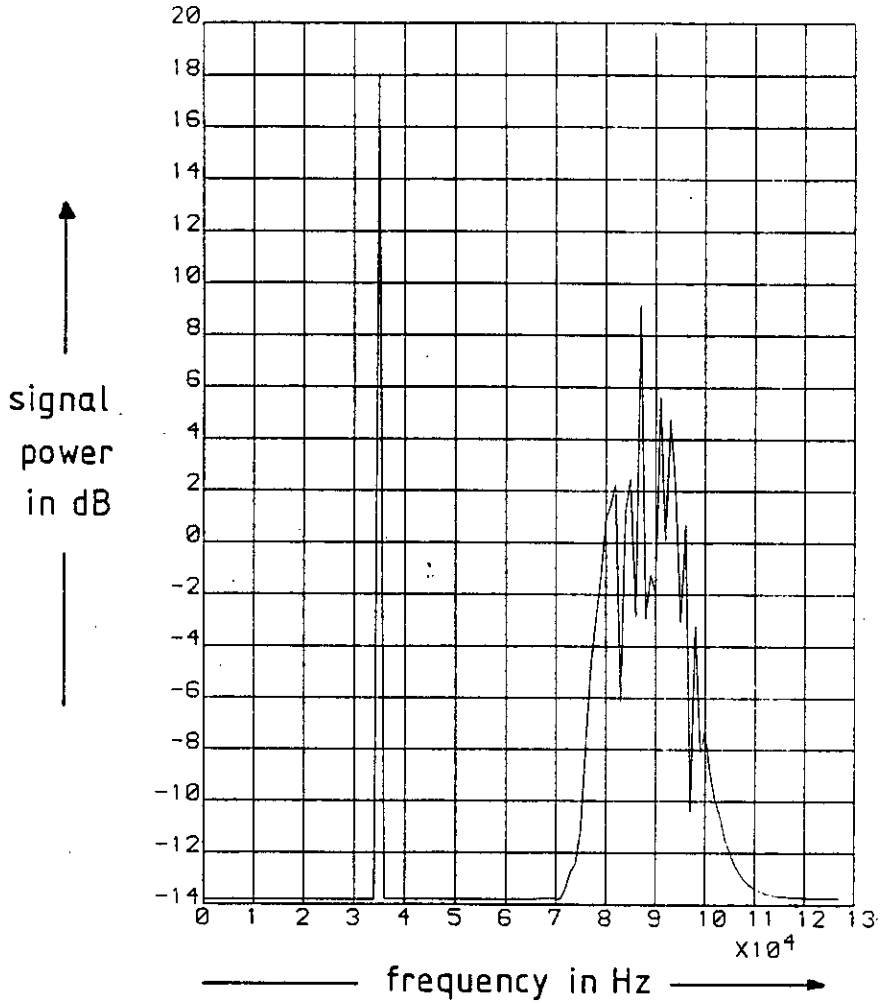


Fig 8.5 The power spectrum of the ASE test signal

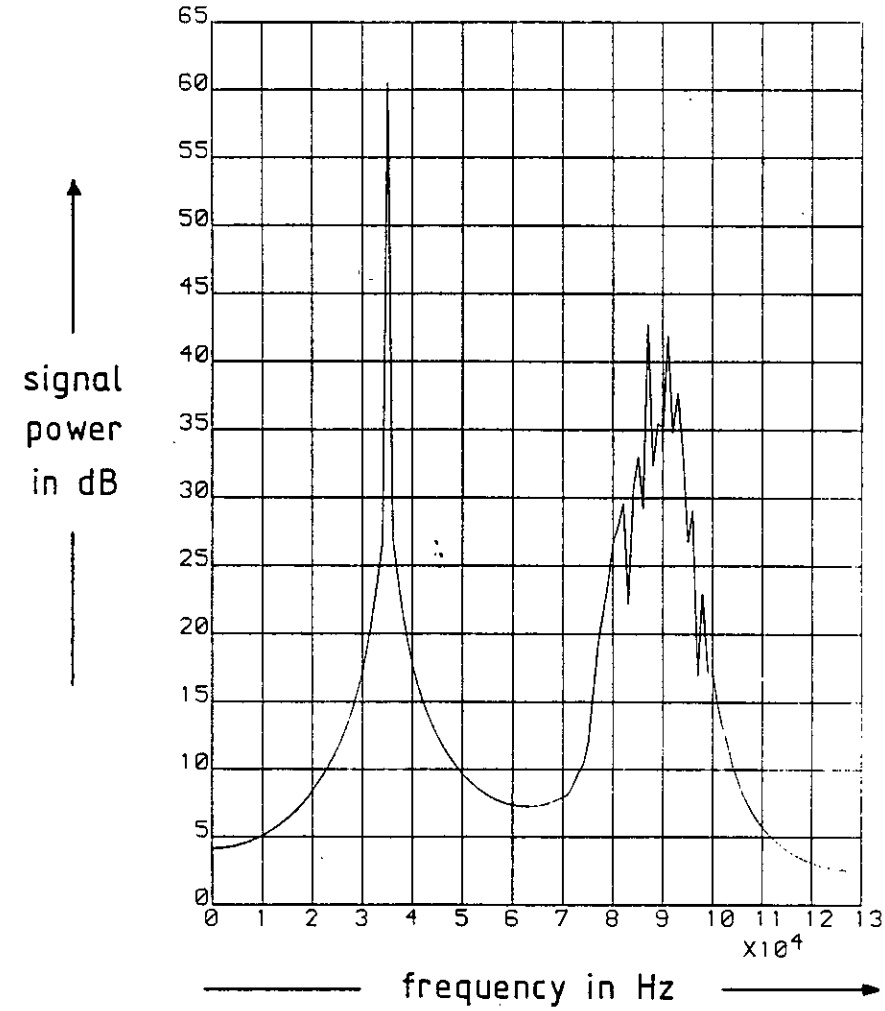


Fig 8.6 The power spectrum of the enhanced ASE output

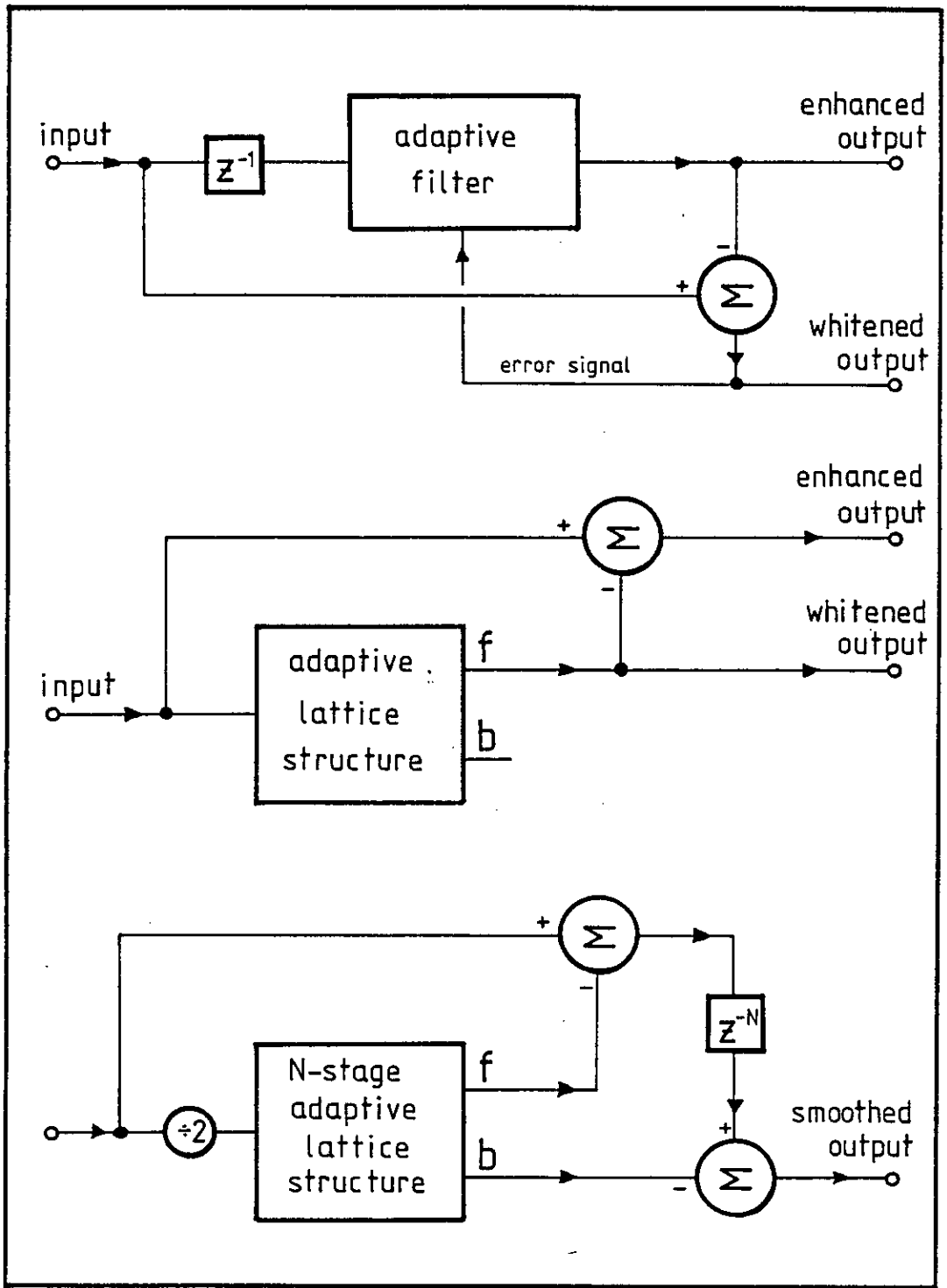


Fig 8.7 (upper) An adaptive line enhancer

(middle) An adaptive lattice line enhancer

(lower) An adaptive lattice smoother

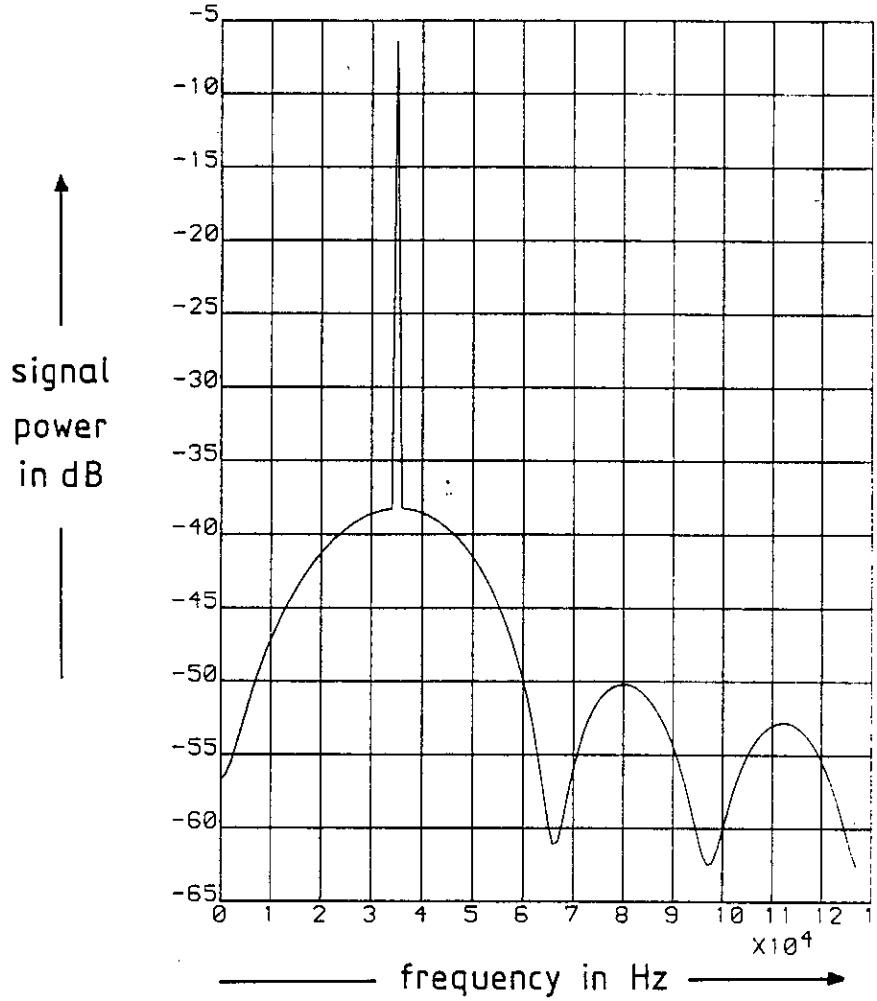


Fig 8.8 The power spectrum of the output of an ALE with a sinusoidal input

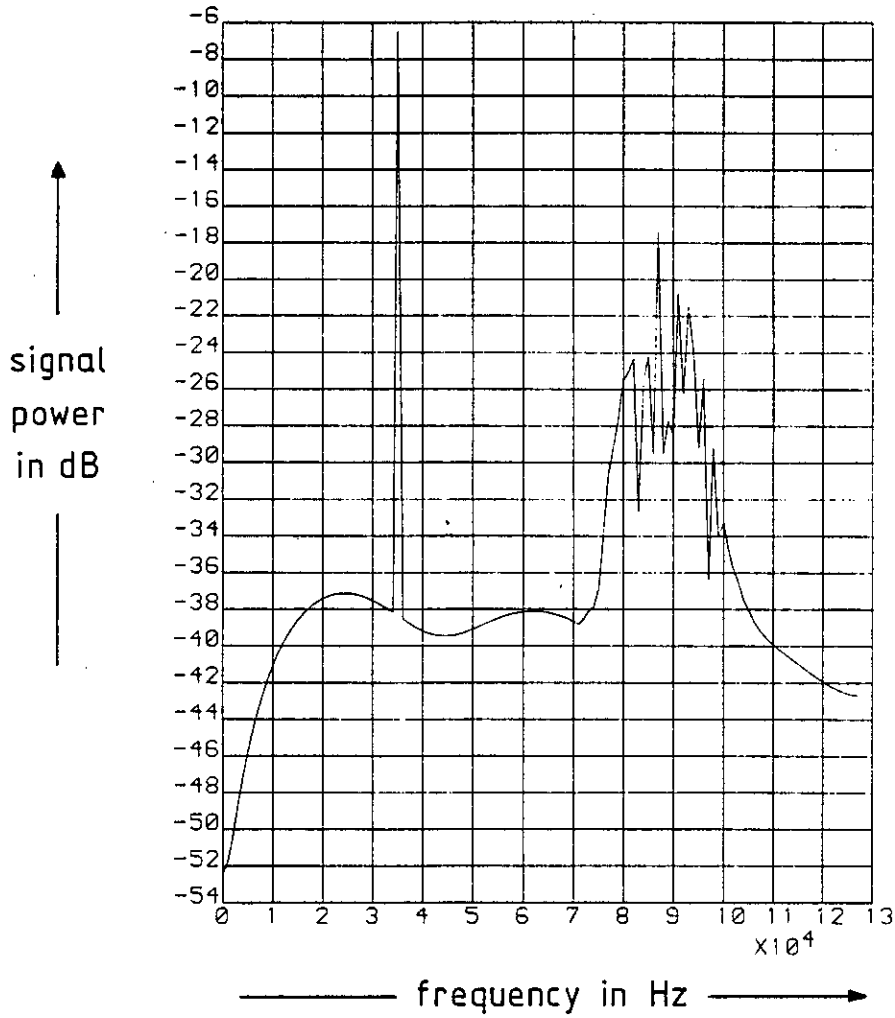


Fig 8.9 The power spectrum of the output of an ALE, when using as input the signal of fig 8.5

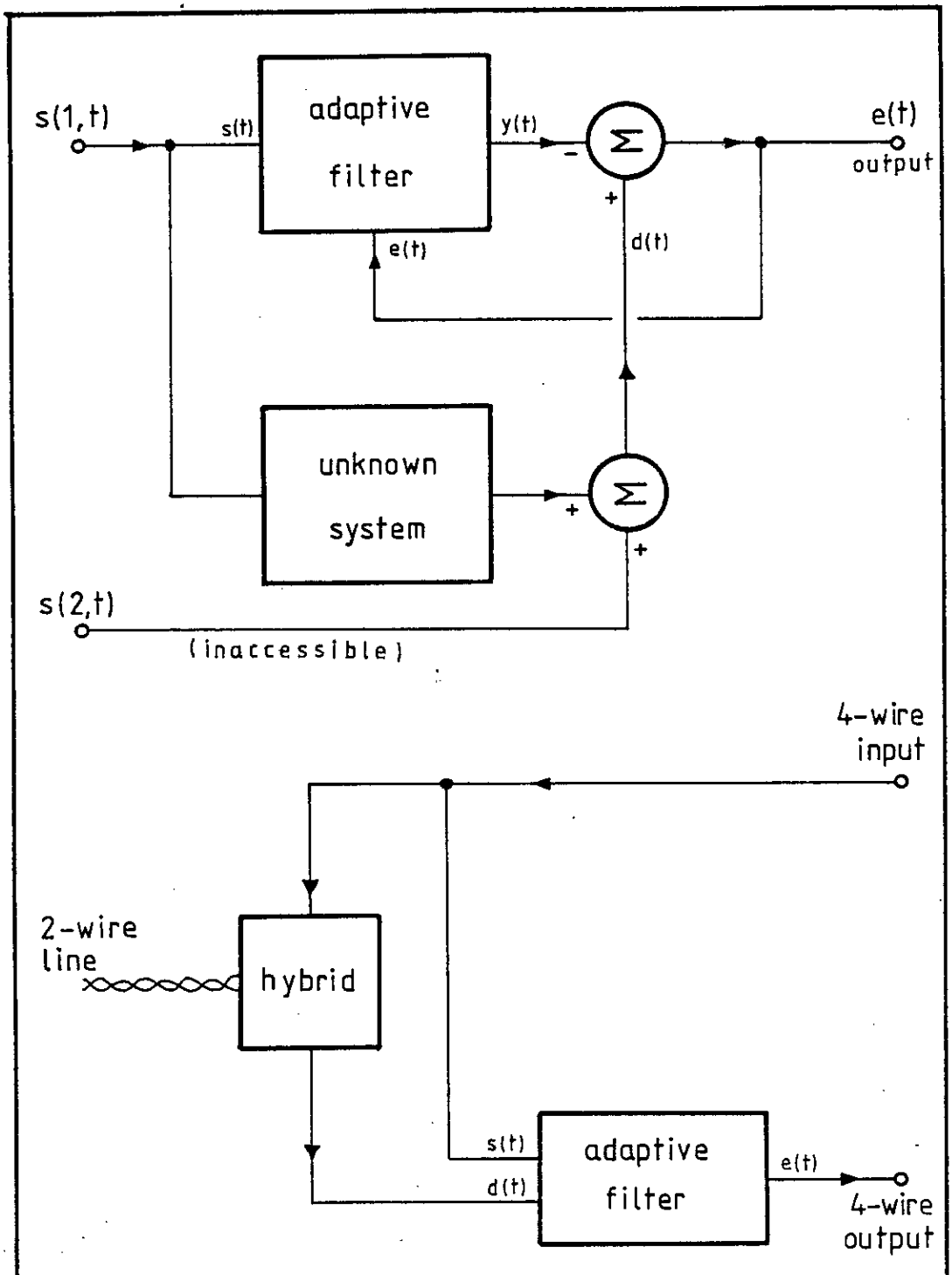


Fig 8.10 System modelling:
(upper) Signal cancellation
(lower) An echo cancellation example

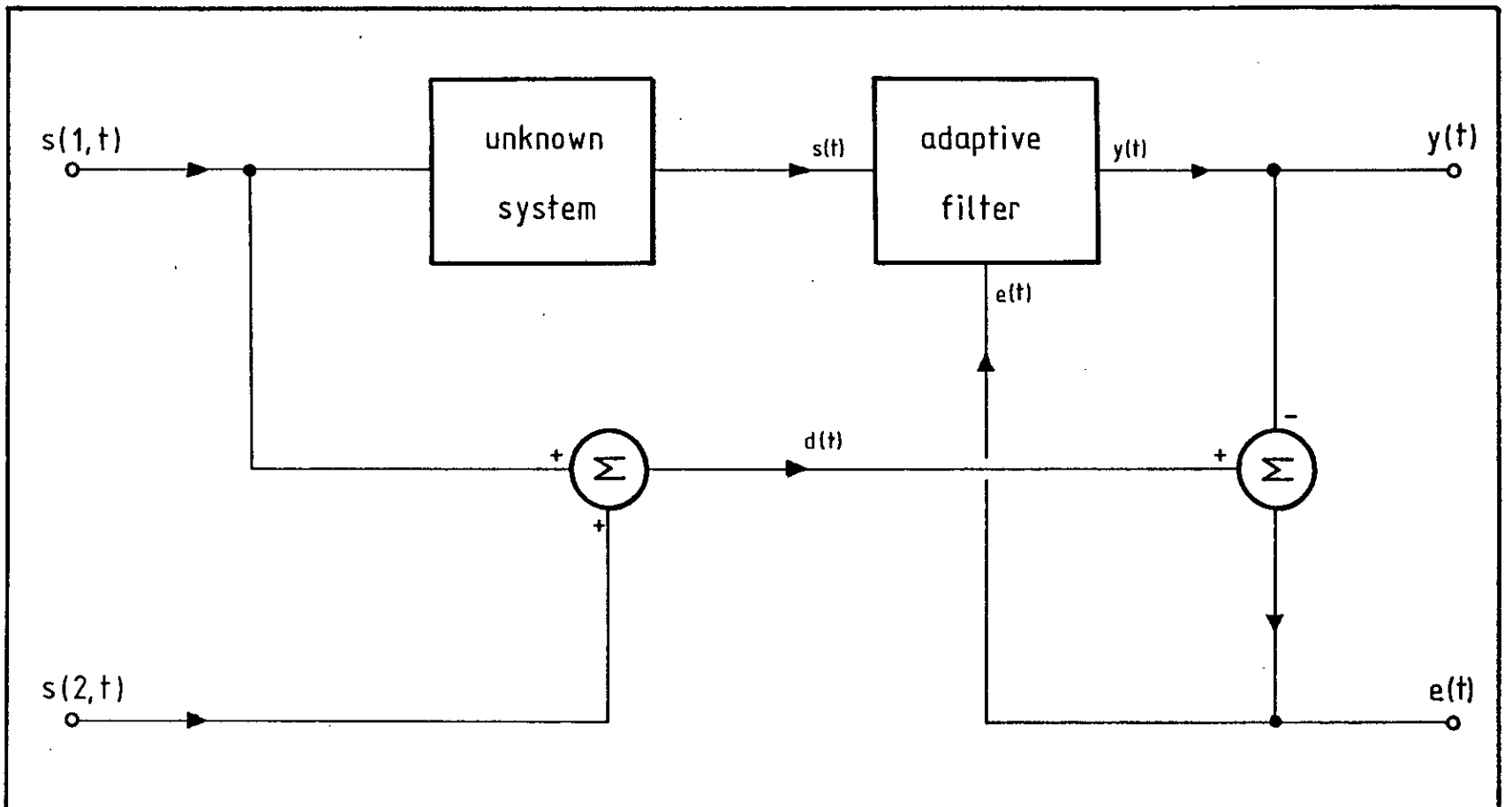


Fig 8.11 System inversion

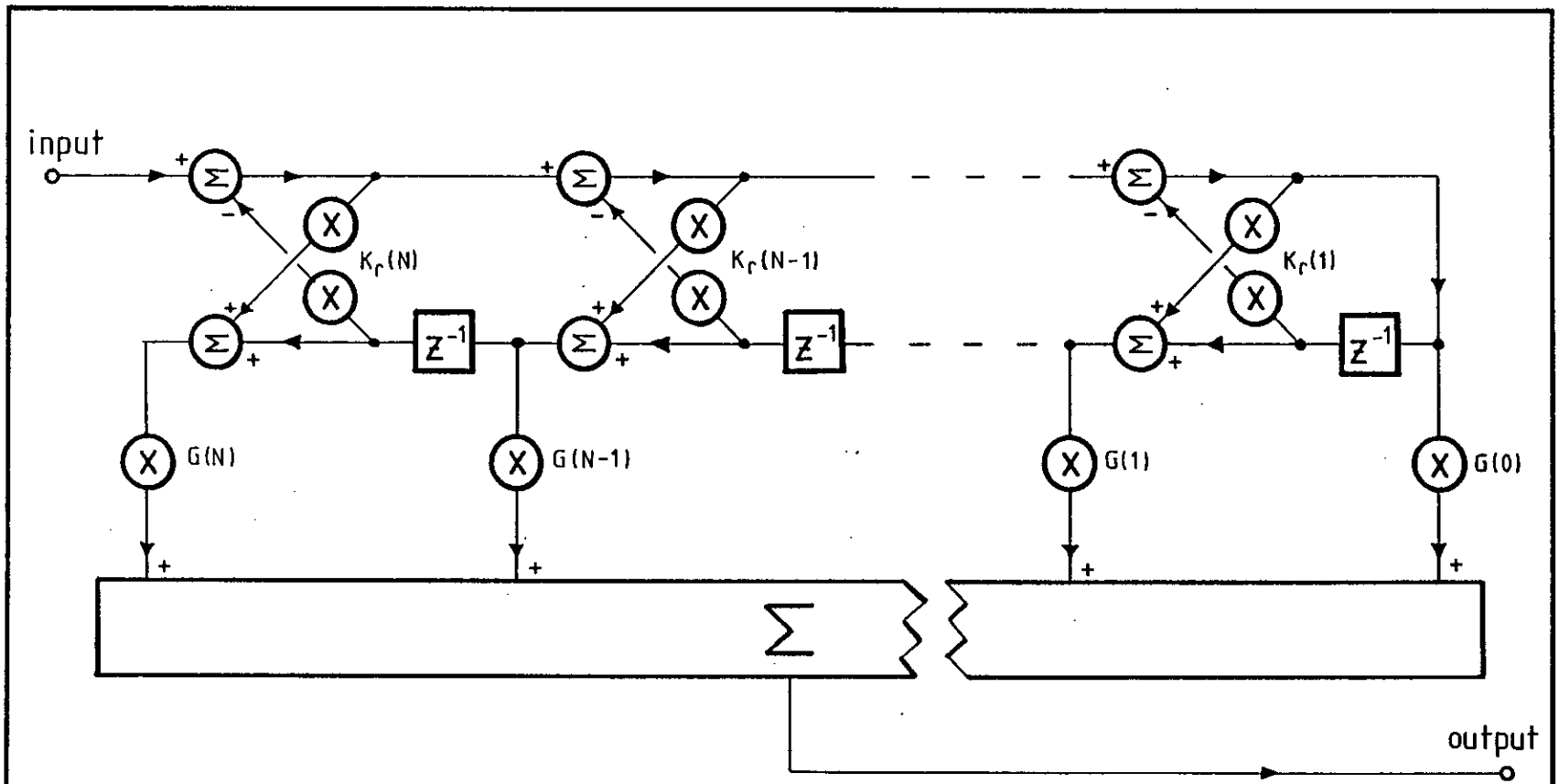


Fig 8.12 A lattice ARMA filter

Chapter Nine

CONCLUSIONS

This thesis has investigated the gradient lattice filter (equaliser) and shown that it is a major competitor to the transversal LMS adaptive filter. The higher complexity of the lattice algorithm is compensated by its more reliable rate of convergence. Another advantage, emphasised in section 3.6, is that the output of the converged equaliser may be taken from any stage, allowing Wiener solutions of arbitrary order to be selected.

The algorithm noise of the lattice structure itself has been shown to be a critical factor in circuit designs. This noise has been investigated mathematically and has been verified by simulation results. In this way the limitations of published designs have been thoroughly examined and the insight gained has led to improved lattice equaliser designs. These improved designs have been subsequently verified by computer simulation on models of typical modem channels.

Three hardware implementations of lattice filters have been constructed, in order to evaluate the available technologies. The performance of two of the implementations have been compared by evaluating the cancellation depth which can be achieved in a 2-sinusoid test. The analogue implementation, which was the simplest, offered a cancellation depth of 20dB on a system with a sampling frequency of 10kHz. An examination of the reasons behind this relatively poor performance have subsequently led to a proposed improvement in the design which will certainly yield a superior cancellation depth. A discrete component TTL digital equaliser gave a cancellation depth of 50-60 dB and a sampling frequency of 17kHz. The custom IC design was shown to be an accurate spectral estimator and to be capable of a 20 kHz sampling frequency in NMOS or an 80 kHz sampling frequency in CMOS. Its extension into a lattice equaliser can now be easily undertaken from the present stage of the work.

The lattice equaliser has been shown to be suitable for

environments where reliable convergence and tracking properties are desirable. These include polled modem systems and HF channels which suffer from fast fades due to multipath. Since telephone modem equalisation is a major potential application for the lattice, a survey of the various parameters of typical channel types has been performed. Typical channels have subsequently been selected and tested with the various lattice equaliser simulations.

The applications of the lattice to the general area of signal processing have been surveyed. While these cover most areas where adaptive filtering is appropriate, a most exciting property of the lattice is its ability to vary the colouration of the power spectral density of a signal. One application of this property is in SNR improvement.

In addition to the construction and evaluation of gradient adaptive lattice hardware designs, the main contributions of this thesis include the identification of algorithm noise as a critical factor in gradient lattice filter designs. A mathematical description has allowed quantification and the modelling of possible solutions. In this context both global and distributed LMS linear combiner algorithms have been compared and it has been determined that the more rapid tracking of the distributed algorithm is offset by an increased algorithm noise. The use of timed lattice algorithms for approximate orthogonalisation with the minimum of algorithm noise has been explored. Two new algorithms using timed techniques have been investigated: one a regular lattice equaliser with a timed lattice structure feeding sidetaps which use a global algorithm. The other is a short timed structure which precedes and speeds the convergence of a transversal adaptive filter by whitening its input signal. Both algorithms offer a reliable rate of convergence coupled with a low algorithm noise, and hence warrant further investigation for specific applications.

The mathematics of the lattice have been applied to the structurally simpler transversal filter, to develop algorithms which speed up the latter's rate of convergence. An open-loop adaptive transveral algorithm has been presented which has the innovation of

being able to truncate its own filter order in an optimum manner. Another innovation, worthy of future work, is a fast-converging closed-loop transversal design, with applications in beam-steering and passive sonar. This, again, is based on lattice mathematics.

Many other applications for lattice filters exist in addition to their use as adaptive equalisers. The most interesting of these appear at present to be in helium speech unscrambling and various forms of line enhancement, such as adaptive smoothing. The adaptive spectral enhancer, which reduces noise by enhancing spectral colouration, is an entirely new structure which is currently under investigation. This thesis has investigated gradient adaptive lattice filters with special reference to equalisation; further work should now be undertaken to evaluate the suitability of the lattice structure to these other applications also.

REFERENCES

1. A.P.Clark, Advanced data transmission systems, Pentech press, Plymouth 1977.

2. P.N.Ridout and P.Rolfe, "Transmission measurements of connexions in the switched telephone network," Post-Office engineers' journal, Vol 63, pt. 2, July 1970, pp 97-104.

3. B.Widrow, J.R.Glover, Jr., J.M.McCool, J.Kaunitz, C.S.Williams, R.H.Hearn, J.R.Zeidler, E.Dong, Jr., J.C.Goolin, "Adaptive noise cancelling: Principles and applications," Proc IEEE, Vol 63, No 12, Dec 1975, pp 1692-1716.

4. N.Wiener, Extrapolation, interpolation and smoothing of stationary time series, with engineering applications, The Technology Press of MIT and J. Wiley and Sons Inc., NY, 1949.

5. B.Widrow, P.E.Mantey, L.J.Griffiths, B.B.Goode, "Adaptive antenna systems," Proc IEEE, Vol 55, No 12, Dec 1967, pp 2143-2159.

6. R.W.Lucky, "Techniques for adaptive equalisation of digital comunications systems," BSTJ, Vol 45, No 2, Feb 1966, pp 255-286.

7. G.Ungerboeck, "Theory on the speed of convergence in adaptive equalisers for digital communication," IBM J.Res.Dev., Vol 16, No 6, Nov 1972, pp 546-555.

8. R.W.Chang; "A new equaliser structure for fast start-up digital communication, " BSTJ, Vol 50, No 6, July-August, 1971, pp 1969-2014.

9. M.C.Goodall, "Performance of a stochastic net," Nature, Vol 185, No 4712, Feb 20th 1960, pp 557-558.

10. J.K.Lubbock, "A self-optimising non-linear filter," Proc IEE, Vol 108b, No 40, July 1961, pp 429-430.

11. J.Makhoul, "Stable and efficient lattice methods for linear prediction," IEEE Trans ASSP, Vol ASSP-25, No 5, October 1977, pp 423-428.

12. F.Itakura and S Saito, "Digital filtering techniques for speech analysis and synthesis," Proc 7th Int. Conf. Acoustics, Budapest, 1971, pp 261-264.

13. K.O.Mead and W.H.Ryder, Improvements in or relating to self-adaptive linear prediction filters, U.K patent GB 2 026 289 B, 21st April 1982.

14. L.J.Griffiths, "A continuously adaptive filter implemented as a lattice structure," IEEE Trans Int. Conf. ASSP, Hartford, May 1977, pp 683-686.

15. L.J.Griffiths, "An adaptive lattice structure for noise-cancelling applications," IEEE Trans Int. Conf. ASSP, Tulsa OK, April 1978, pp 87-90.

16. M.Morf and D.T.Lee, "Recursive least-squares ladder forms for fast parameter tracking," IEEE Int. Conf. Decision and Control, San Diego, 1979, pp 1362-1367

17. P.Butler and A.Cantonini, "Noniterative automatic equalisation," IEEE Trans Comms, Vol COM-23, No 6, June 1975, pp 621-633.

18. D.A.Linkens, "Short time-series spectral analysis of biomedical data," IEE Proc, Vol 129, Pt A, No 9, Dec 1982, pp 663-672.

19. M.J.Di Toro, "Communication in time-frequency spread media using adaptive equalisation," Proc IEEE, Vol 56, No 10, Oct 1968, pp 1653-1679.

20. P.A.Lynn, The analysis and processing of signals, Macmillan, London, 1973.

21. S.Haykin and S.Kesler, "Prediction-error filtering and maximum-entropy spectral estimation," S.Haykin (ed.), Nonlinear methods of spectral analysis, Topics in applied physics, Vol 34, Springer-Verlag, Berlin 1979.

22. D.G.Messerschmitt, "A class of generalised lattice filters," IEEE Trans ASSP, Vol ASSP-28, No 2, April 1980, pp 198-204.

23. J.Makhoul, "A class of all-zero lattice digital filters: Properties and applications," IEEE Trans ASSP, Vol ASSP-26, No 4, Aug 1978, pp 304-314.

24. B Friedlander, "Lattice forms for adaptive processing," Proc IEEE, Vol 70, No 8, Aug 1982, pp 829-867.

25. D.T.Finkbeiner, Introduction to matrices and linear transformations, W.H.Freeman, London, 1966.

26. The International dictionary of applied mathematics, D.Van Nostrand, London, 1960.

27. F.G.Stremler, Introduction to communications systems, Addison-Wesley, London, 1977.

28. F.F.Kuo, Network analysis and synthesis, Wiley and Sons, London, 1962.

29. S.M.Kay and S.L.Marple, Jr., "Spectrum analysis - a modern perspective," Proc IEEE, Vol 69, No 11, Nov 1981, pp 1380-1419.

30. C.J.Gibson and S.Haykin, "Radar performance studies of adaptive lattice clutter-suppression filters," IEE Proc, Vol 130, Pt F, No 5, August 1983, pp 357-367.

31. M.J.Hawksford and N.Rezaee, "Adaptive mean-square-error transversal equaliser" IEE Proc, Vol 128, Pt F, No 5, Oct 1981, pp 296-304.

32. M.J.Box, "a new method of constrained optimisation and a comparison with other methods," Computer Journal, Vol 8, No 1, April 1965, pp 42-52.

33. R.A.Monzingo and T.W.Miller, Introduction to adaptive arrays, J.Wiley and Sons, NY, 1980.

34. B.Widrow, J.M.McCool, M.G.Larimore, C.R.Johnson, Jr., "Stationary and non-stationary learning characteristics of the LMS adaptive filter," Proc IEEE Vol 64, No 8, Aug 1976, pp 1151-1162

35. P.L.Feintuch, "An adaptive recursive LMS filter," Proc IEEE, Vol 64, No 11, Nov 1976, pp 1622-1624.

36. C.R.Johnson, "Comments and additions to "An adaptive recursive LMS filter"," Proc IEEE, Vol 65, No 9, Sept 1977, pp 1399-1404.

37. M.Schwartz, Information transmission, modulation and noise, McGraw-Hill, London, 1970.

38. E.H.Satorius and S.T.Alexander, "Channel equalisation using adaptive lattice algorithms," IEEE Trans Comms, Vol COM-27, No 6, June 1979, pp 899-905.

39. A.B.Bageroer, "Sonar signal processing," A.V.Oppenheim (ed.), Applications of digital signal processing, Prentice-Hall, Englewood Cliffs NJ, 1978.

40. H.Widom, "Toeplitz matrices", I.I.Hirschmann, Jr. (ed.), Real and complex analysis, The Mathematical Association of America, Prentice-Hall, Englewood Cliffs NJ, 1965.

41. R.R.Bitmead, "Adaptive frequency sampling filters," IEEE Trans Circuits and Systems, Vol CAS-26, No 6, June 1981, pp 524-534.

42. K.M.Wong and Y.G.Yan, "Adaptive Walsh equaliser for data transmission," IEE Proc, Vol 130, Pt F, No 2, Mar 1983, pp 153-160.

43. A.Morgul, P.M.Grant and C.F.N.Cowan, "Wideband hybrid analog/digital frequency domain adaptive filter," Submitted for publication to IEEE Trans ASSP.

44. J.P.Burg, Maximum entropy spectral analysis, PhD dissertation, Stanford University, Stanford CA, May 1975.

45. P.Monsen, "Fading channel communications," IEEE Communications magazine, Vol 18, No 1, Jan 1980, pp 16-25.

46. M.G.Honig and D.G.Messerschmitt, "Convergence properties of an adaptive digital lattice filter," IEEE Trans Circuits and Systems, Vol CAS-28, No 6, June 1981, pp 482-493.

47. E.H.Satorius and J.D.Pack, "On the application of least-squares lattice algorithms to adaptive equalisation," IEEE Trans Comms, Vol COM-29, No 2, pp 136-142.

48. M.D.Yücel, N.Tepedelenioglu and Y.Tanik, "A fast non-iterative method for channel equalisation," Short communication and poster digest, First European Signal Processing Conference (EUSIPCO), Lausanne Switzerland, 1980, pp 155-156

49. M.J.Rutter and C.F.N.Cowan, "Open-loop transversal equaliser of optimal length," Electronics Letters, Vol 19, No 6, Mar 1983, pp 208-210.

50. K.Groves and P.MacKrill, "A line transmission simulator for testing data-transmission systems," Post-Office Engineers' Journal, Vol 63, Pt 2, July 1970, pp 117-122.

51. C.L.Monk, Transmission characteristics of connections in the switched telephone network measured in the 1967 field study, British Telecommunications Research Laboratories, Ipswitch, 1977.

52. P.F.Adams, "Adaptive filters in telecommunications," C.F.N.Cowan and P.M.Grant (ed.), Adaptive filters, Prentice-Hall, Englewood Cliffs NJ, 1984.

53. B.Widrow, "Adaptive filters," R.E.Kalman and N.de Claris (ed.), Aspects of network and system theory, Holt, Rinehart and Winston, NY, 1971.

54. R.D.Fellman, "A switched capacitor adaptive lattice filter," IEEE Trans ASSP, Vol ASSP-31, No 1, Feb 1983, pp 294-304.

55. D.R.Morgan, "Adaptive multipath cancellation for digital data communications," IEEE Trans Comms, Vol COM-26, No 9, Sept 1978, pp 1380-1390.

56. F.J.Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," Proc IEEE, Vol 66, No 1, Jan 1978, pp 51-83.

57. P.B.Denes and E.N.Pinson, The speech chain, Bell Telephone Laboratories, NY, 1963.

58. P.B.Denyer, D.Renshaw and N.Bergmann, "A silicon compiler for VLSI signal processors," Proc European Solid-State Circuits Conf., Brussels, 1982, pp 215-218

59. R.F.Lyon, "A bit-serial VLSI architectural methodology for signal processing," Proc Int. Conf. on VLSI, Edinburgh, 1981, pp 131-140.

60. J.N.Holmes, "The JSRU channel vocoder," Proc IEE, Vol 127, Pt F, No 1, Feb 1980, pp 53-60.

61. J.Makhoul, "Linear prediction, a tutorial review," Proc IEEE, Vol 63, No 4, Apr 1975, pp 561-580.

62. J.D.Markel, "The SIFT algorithm for fundamental frequency estimation," IEEE Trans AU, Vol AU-20, No 5, Dec 1972, pp 367-377.

63. G.Duncan and M.A.Jack, "A residually excited LPC processor for enhancing helium speech intelligibility," Submitted to Electronics Letters.

64. C.K.Un and D.T.Magill, "The residual-excited linear-prediction vocoder with transmission rate below 9600 kbit/s," IEEE Trans Comms, Vol COM-23, No 12, Dec 1975, pp 1466-1474.

65. R.Wiggins and L.Brantingham, "Three-chip system synthesises human speech," Electronics, Vol 51, No 18, Aug 1978, pp 109-116.

66. C.F.N.Cowan and P.M.Grant, "Adaptive filters await breakthroughs," Microwave Systems News, Vol 11, No 8, Aug 1981, pp 69-79.

67. B.Freidlander, "A recursive maximum likelihood algorithm for ARMA line enhancement," IEEE Trans ASSP, Vol ASSP-30, No 4, Aug 1982, pp 651-657.

68. J.Treichler, The spectral line enhancer - the concept, an implementation and an application, PhD dissertation, University of Stanford, Stanford CA, 1977.

69. J.M.Turner, "Use of the digital lattice structure in digital estimation and filtering," M.Kunt and F.de Coulon (ed.), Signal processing: Theories and applications, North Holland Publishing Company, NY, 1980, pp 33-41.

70. T.S.Durrani and K.C.Sharman, "Eigenfilter approaches to adaptive array processing," IEE Proc, Vol 30, Pt F, No 1, Feb 1983, pp 22-28.

71. S.Papert, "Some mathematical models of learning," C.Cherry (ed.), Information theory, Butterworths, London, 1961, pp 353-363.

72. D.Gabor, W.P.L.Wilby, R.Woodcock, "A universal non-linear filter, predictor and simulator which optimises itself by a learning process," Proc IEE, Vol 108b, No 40, July 1961, pp 422-438.

73. M.J.Rutter and P.M.Grant, "Generation of codes with good autocorrelation properties," Electronics Letters, Vol 19, No 15, July 1983, pp 571-572.

74. L.B.Jackson, K.F.Kaiser and H.S.McDonald, "An approach to implementation of digital filters," IEEE Trans Audio and Electroacoustics, Vol AU-16, No 3, Sept 1968, pp 413-421.

75. W.F.Gabriel, "Spectral analysis and adaptive array super-resolution," Proc IEEE, Vol 68, No 6, June 1980, pp 654-666.

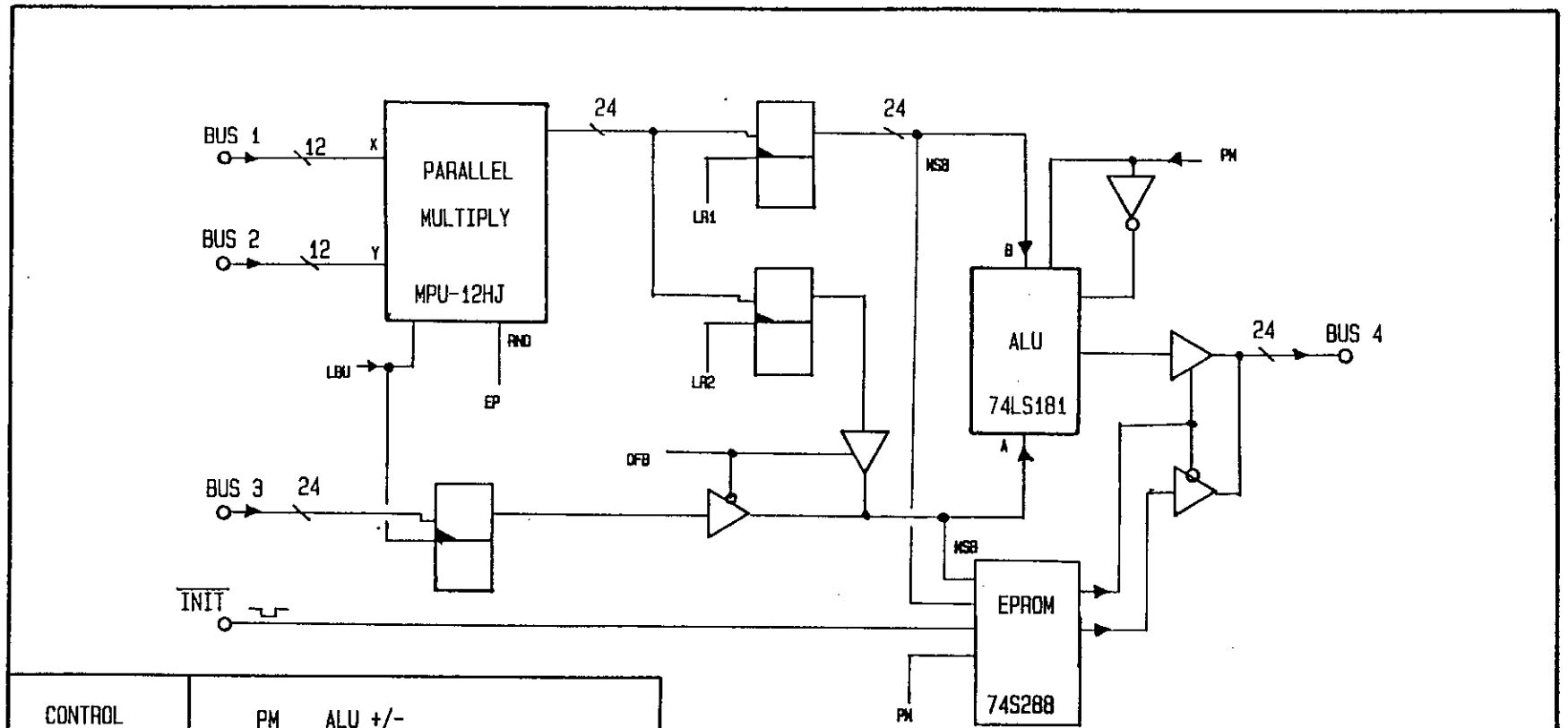
76. P.M.Grant (ed.), The impact of new technologies in signal processing, IEE Conf. publ. No 144, London, Sept 1976.

77. L.R.Rollo, The development of an analogue lattice filter, BSc Honours project report HSP 318, University of Edinburgh, May 1982.

78. C.N.Tate and C.C.Goodyear, "A real-time adaptive lattice filter for speech coding," IEE Conf. publ. No 31, April 1983, pp 3-1 to 3-4.

APPENDIX A: TTL Equaliser Circuits

In chapter 6 a design for a discrete component digital adaptive lattice equaliser was described. In this appendix specific details of the actual circuit design was described. These figures are to be read in conjunction with the general circuit description of section 6.3.1. The contents are as follows:



CONTROL SIGNALS		
	PM	ALU +/-
	LR1	LATCH REGISTER 1
	LR2	LATCH REGISTER 2
	LBU	LATCH BUSSES
	OFB	OUTPUT FEEDBACK
	EP	EXTENDED PRECISION

Fig A1.1

ARITHMETIC UNIT

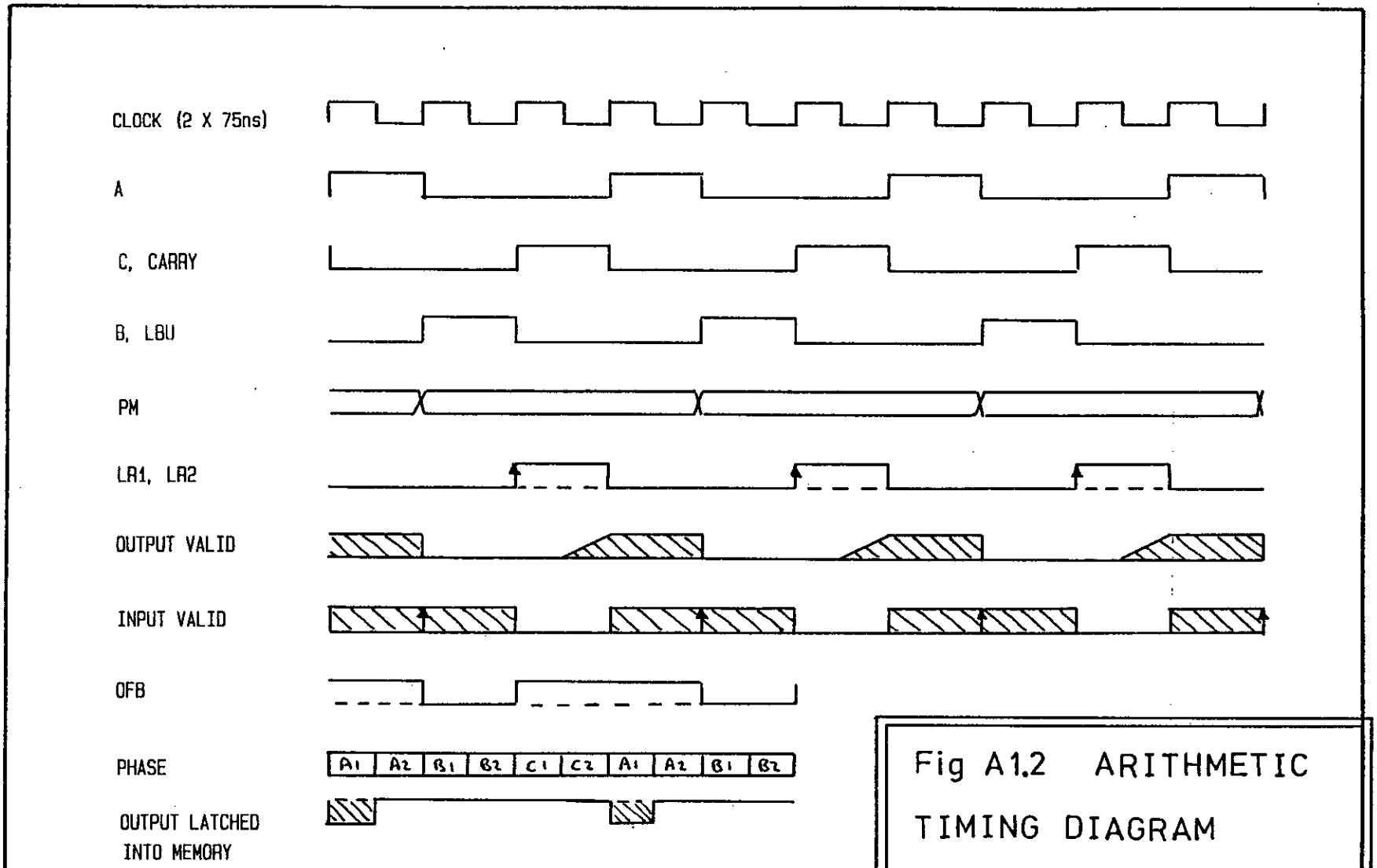


Fig A1.2 ARITHMETIC TIMING DIAGRAM

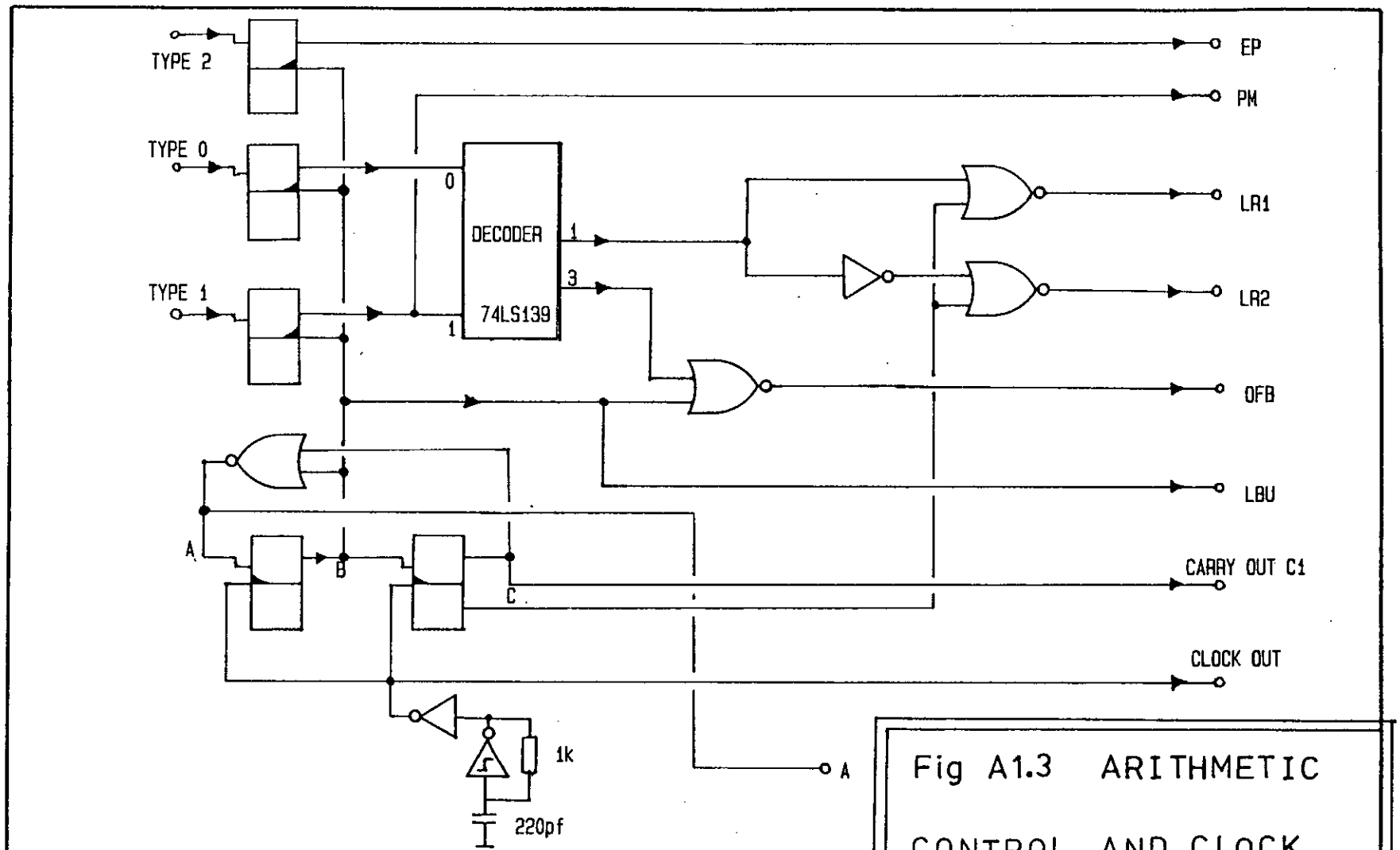


Fig A1.3 ARITHMETIC CONTROL AND CLOCK

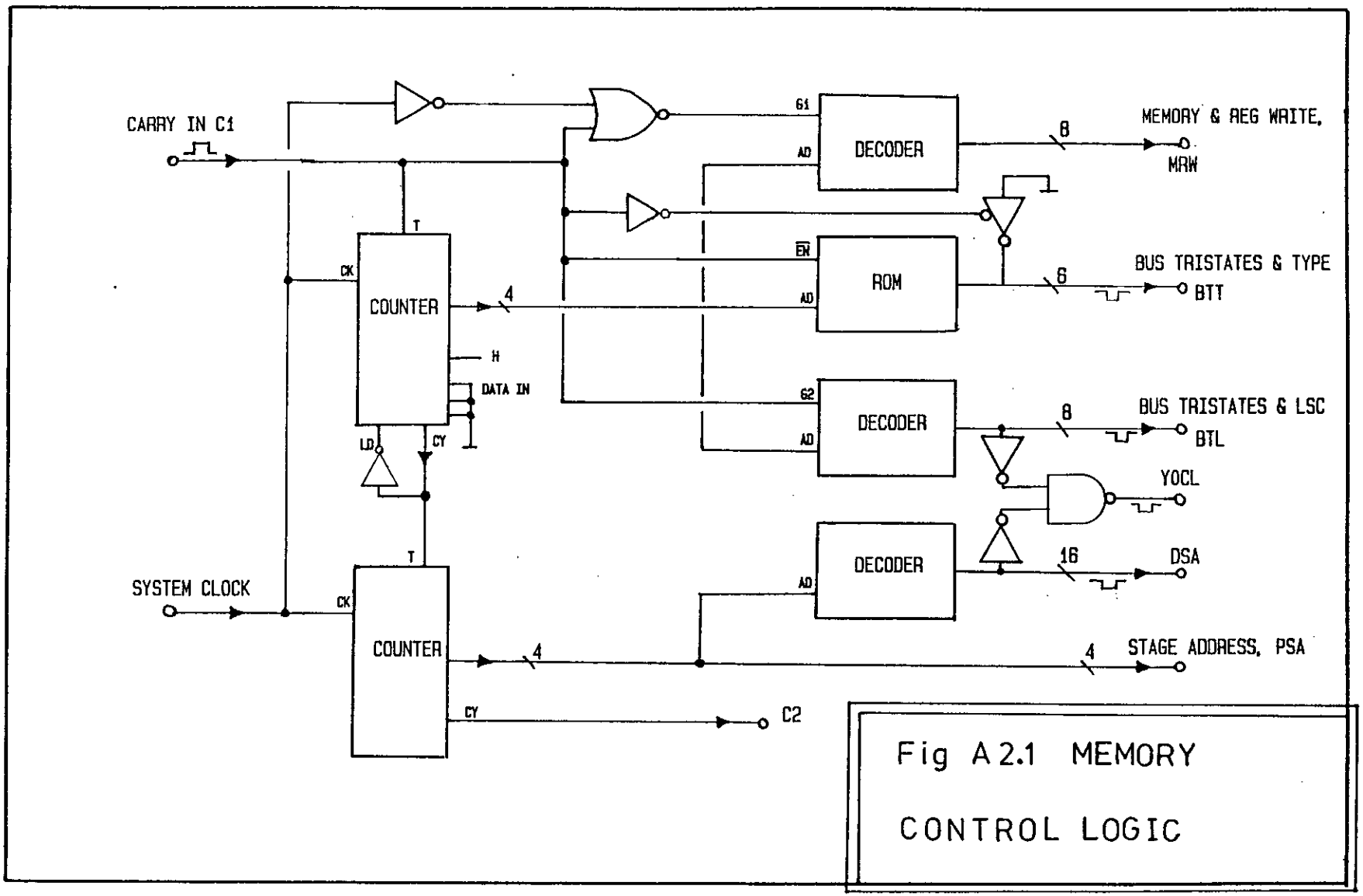


Fig A2.1 MEMORY CONTROL LOGIC

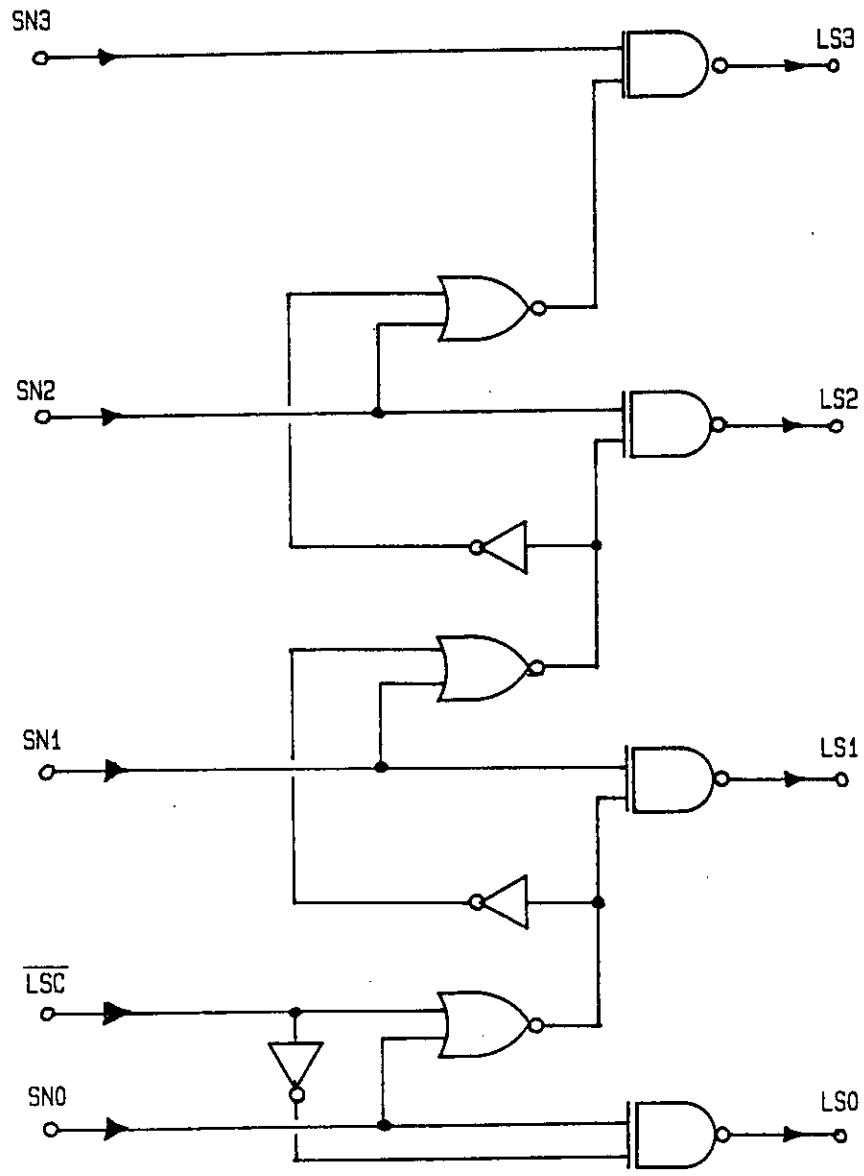


Fig A 2.2
DECREMENTING
NETWORK

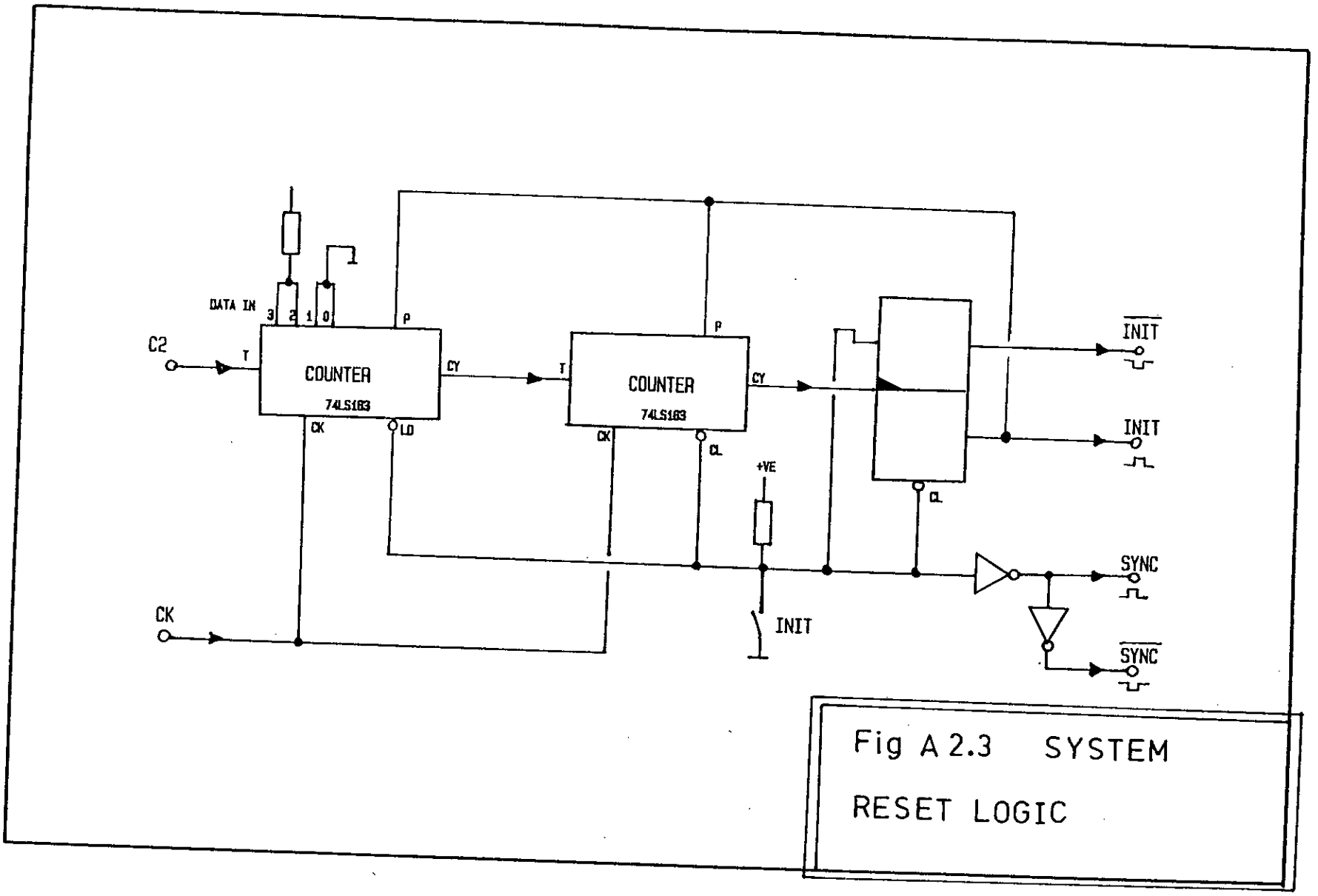


Fig A 2.3 SYSTEM RESET LOGIC

Signal Generated		Signal Used
SAOD	0	TEI, TB01
BTL	0	TB3B0, TB2F1
	1	TB3F1
	2	TB3Y, TB1G
	3	TB3E, LSC; TB1F2
	4	TB37
	5	TB3G, TB2MU2
	6	TB2BN, TB1F1
	7	TB3K, TB2MU1
	0	TB1K
	1	TB14
	2	TB2B0
	3	TYPE0
	4	TYPE1
	5	TYPE2
	0	KOL
	1	BNL
	2	F2L
	3	YL,KL
	4	EL
	5	BOL
	6	GL
	7	FIL

TABLE A2.4 MEMORY CONTROL SIGNALS GENERATED BY MEMORY CONTROL LOGIC.

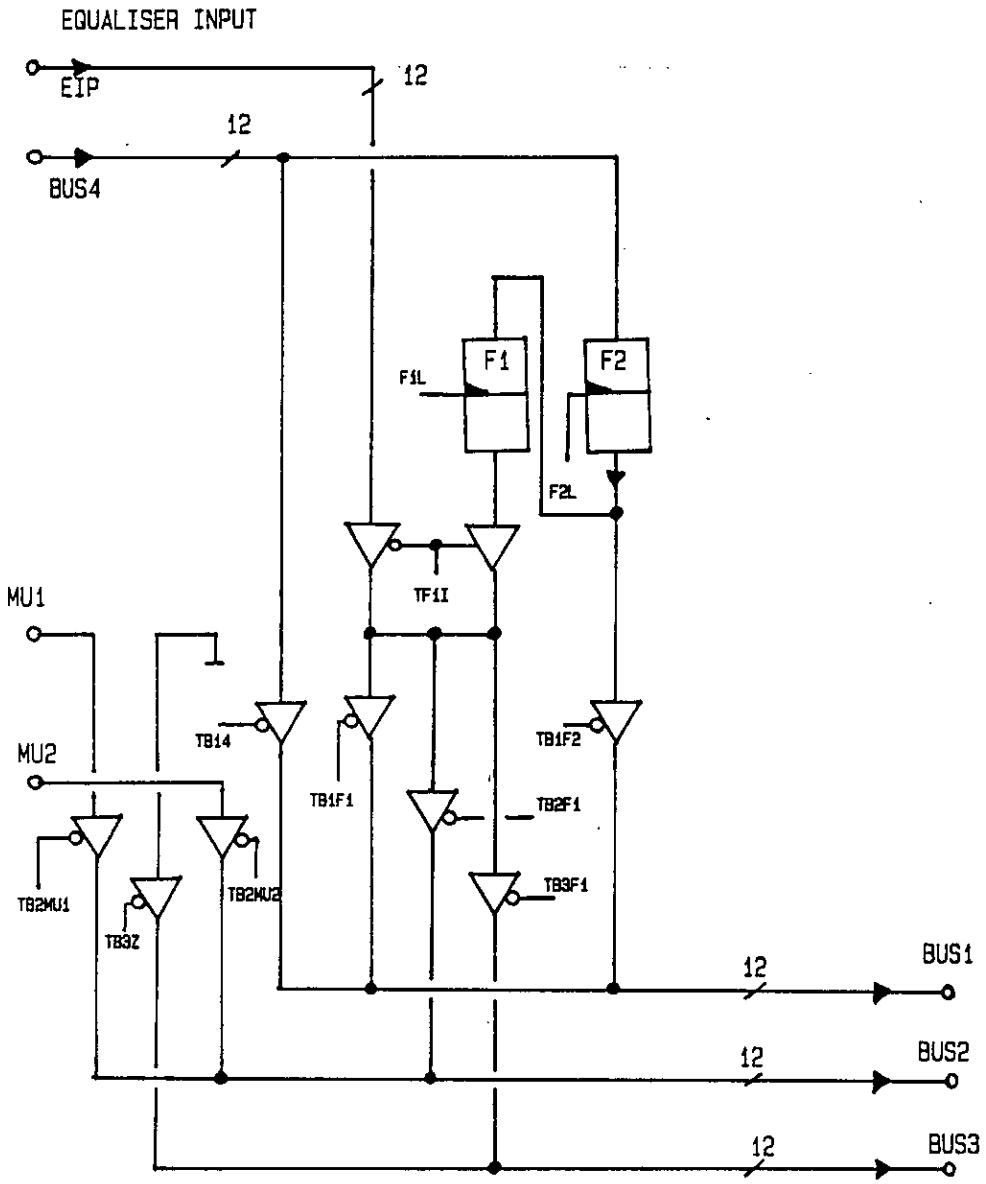


Fig A 3.1

MEMORY REGISTERS

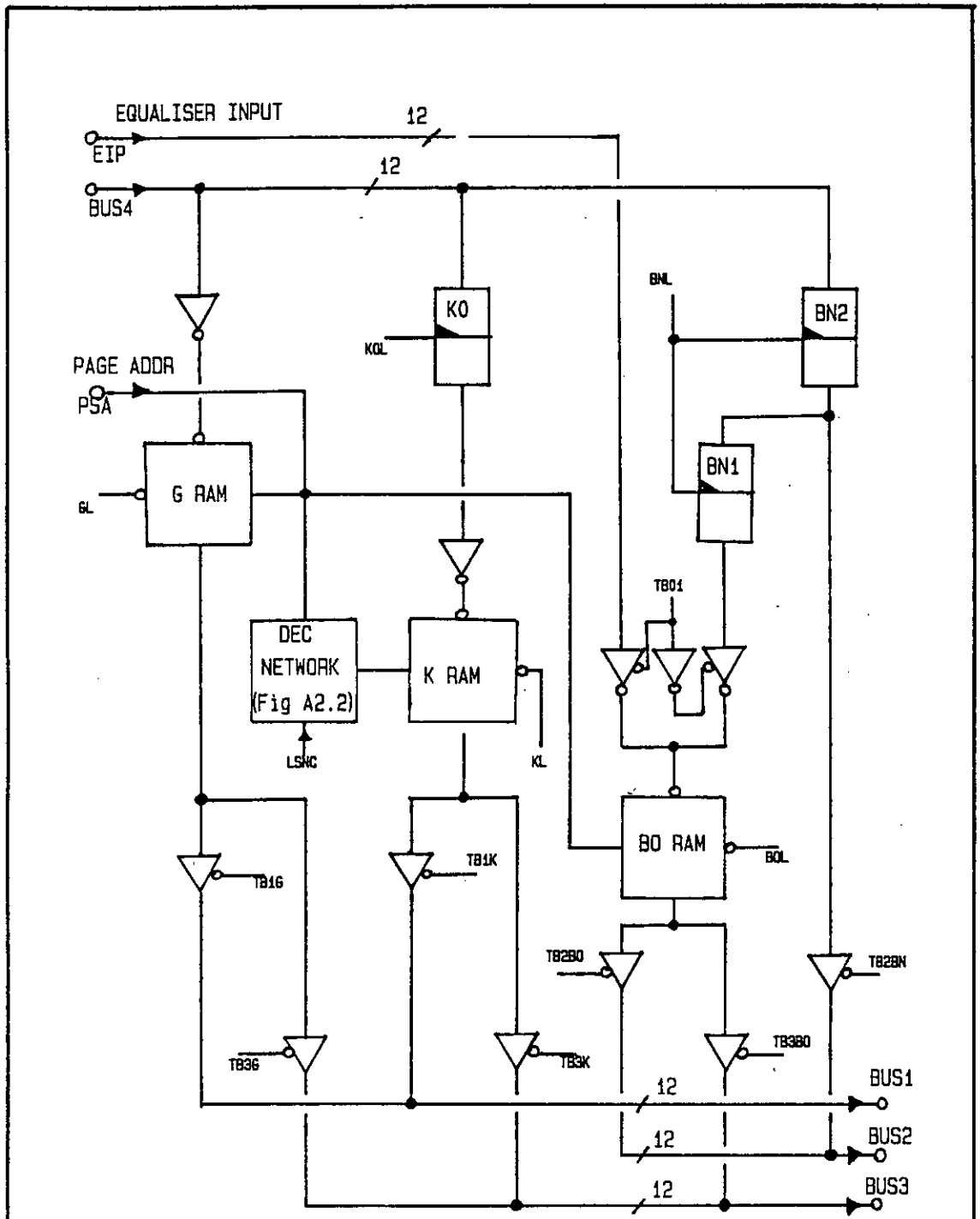


Fig A3.2

MEMORY RAM

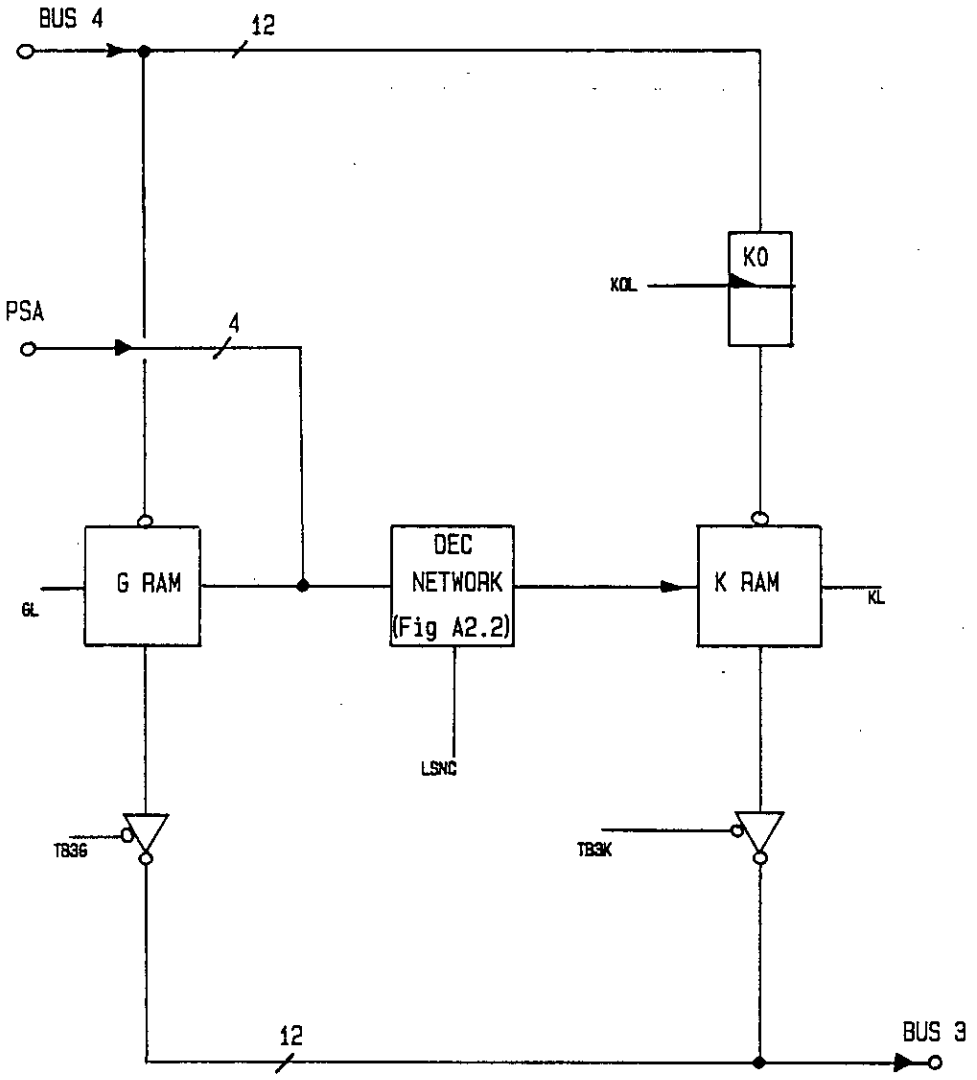


Fig A 3.3

EXTENDED PRECISION COEFFICIENTS

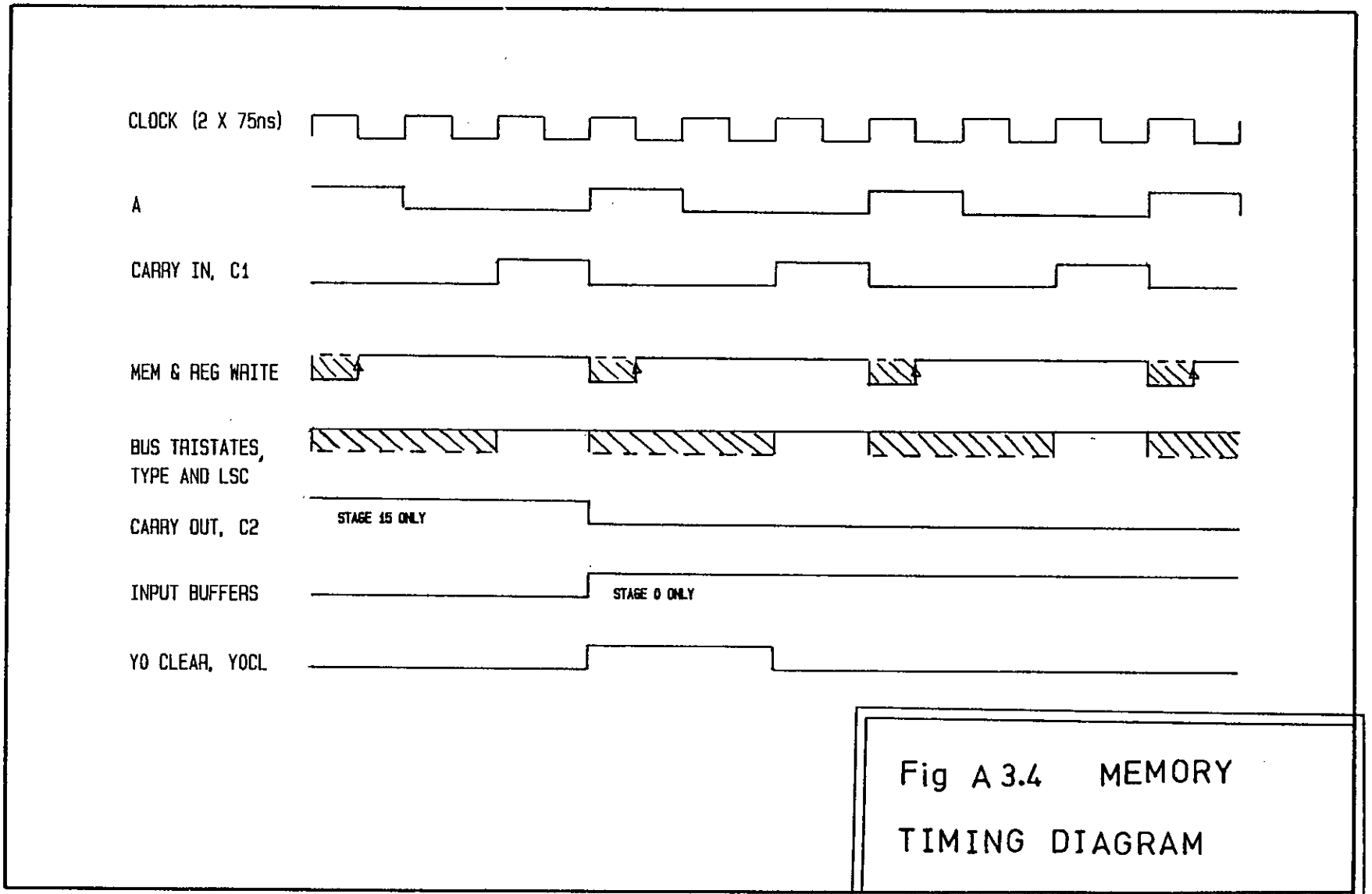


Fig A 3.4 MEMORY TIMING DIAGRAM

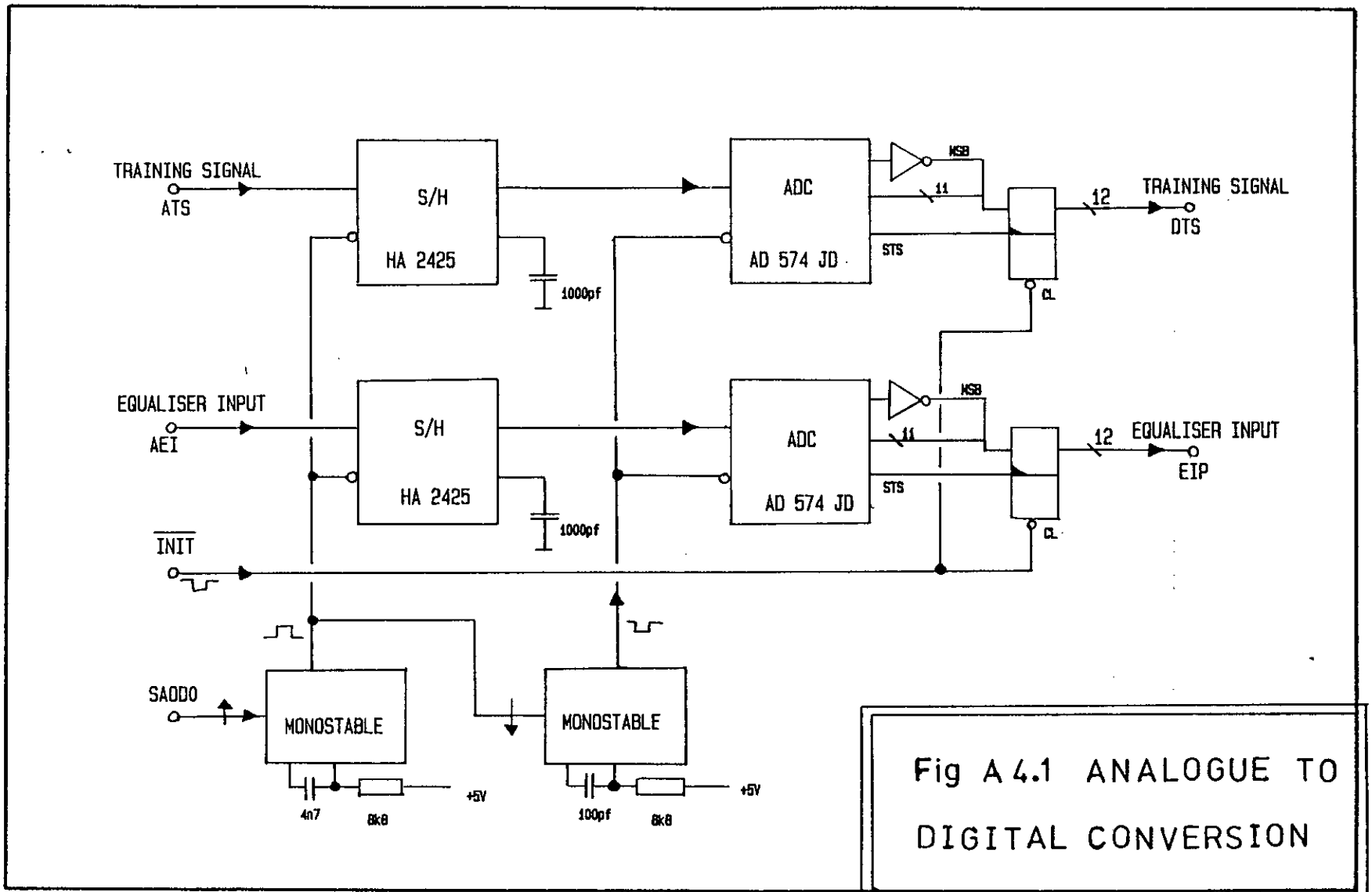


Fig A 4.1 ANALOGUE TO DIGITAL CONVERSION

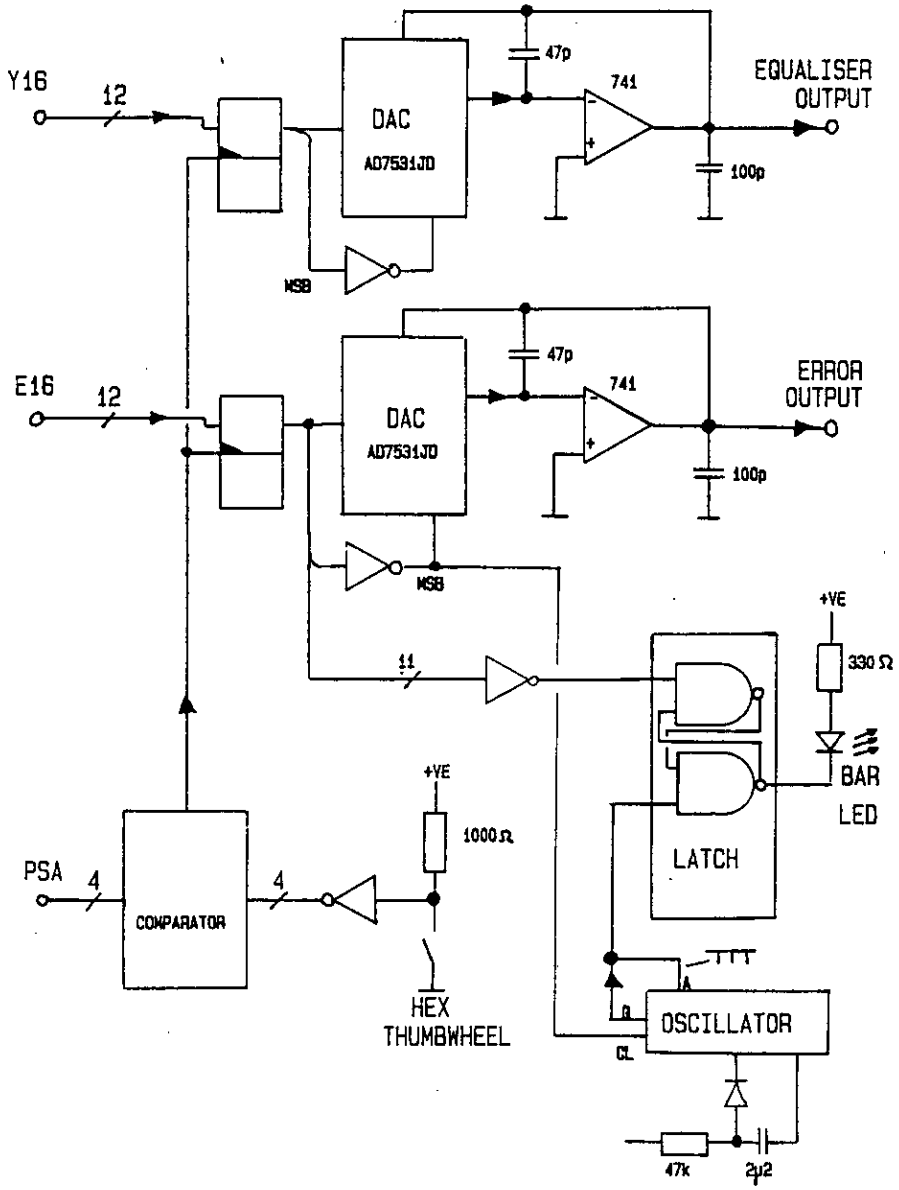


Fig A4.2

OUTPUT CIRCUIT AND DACs

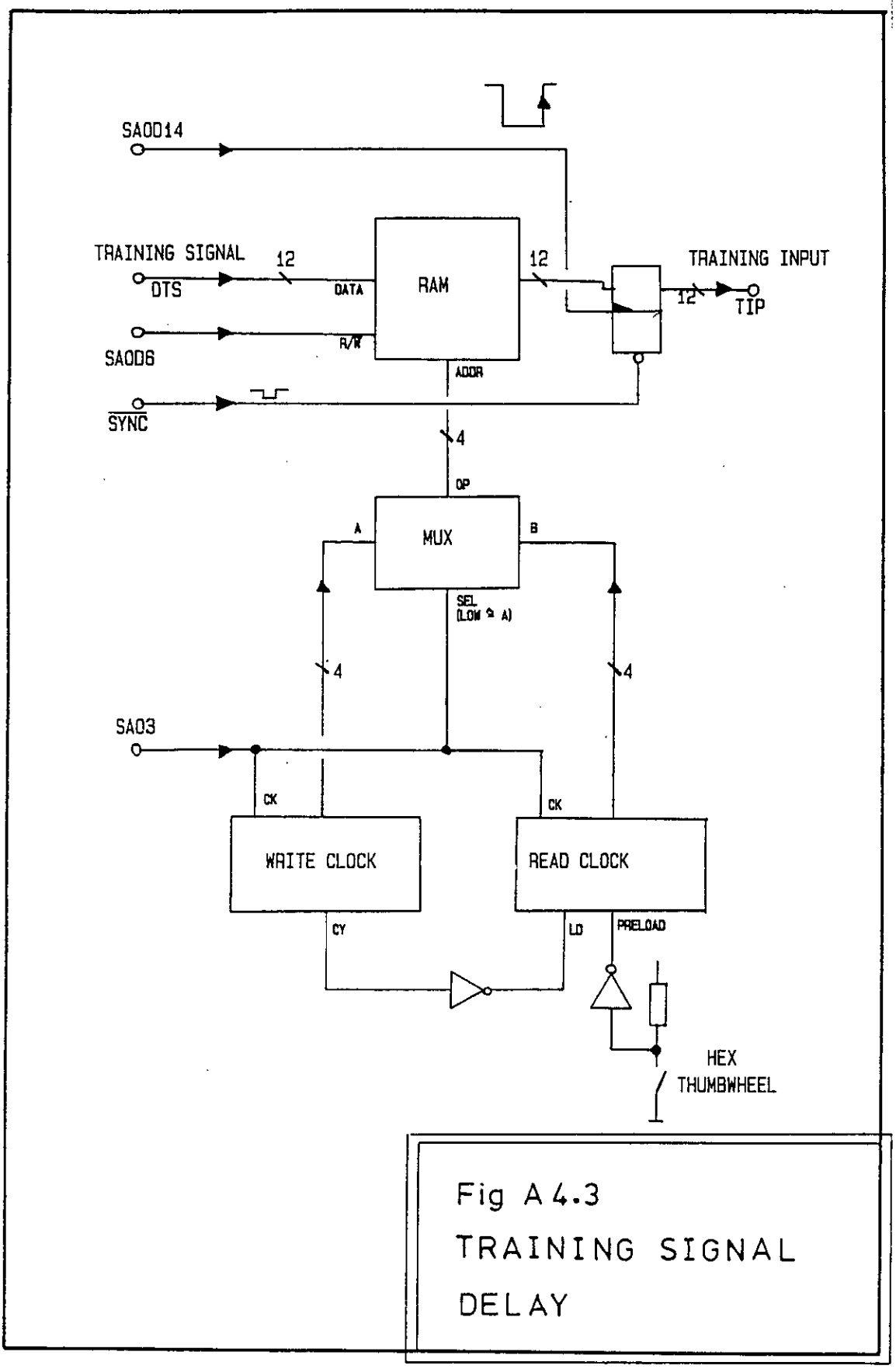


Fig A 4.3
TRAINING SIGNAL
DELAY

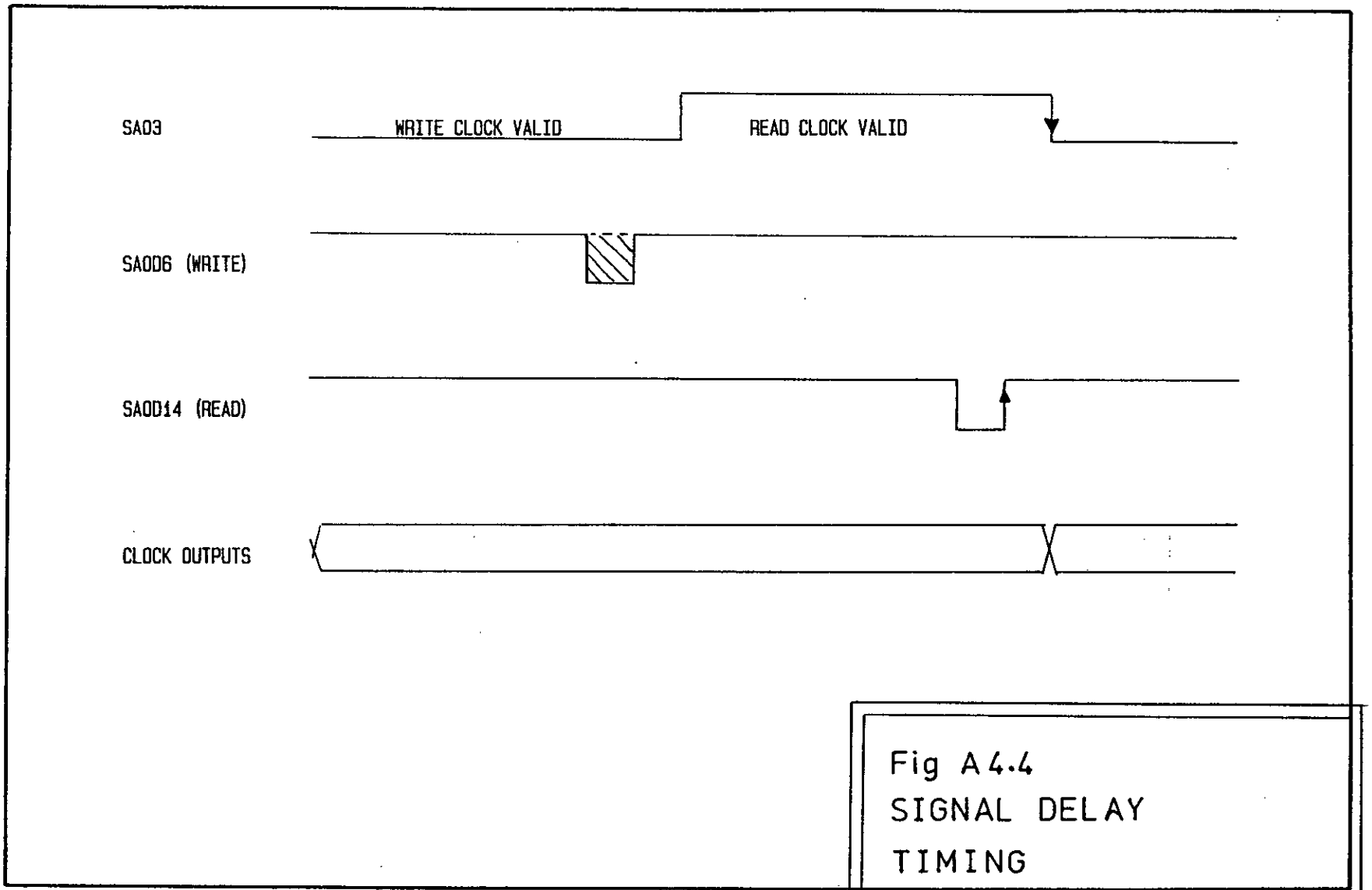


Fig A4.4
SIGNAL DELAY
TIMING

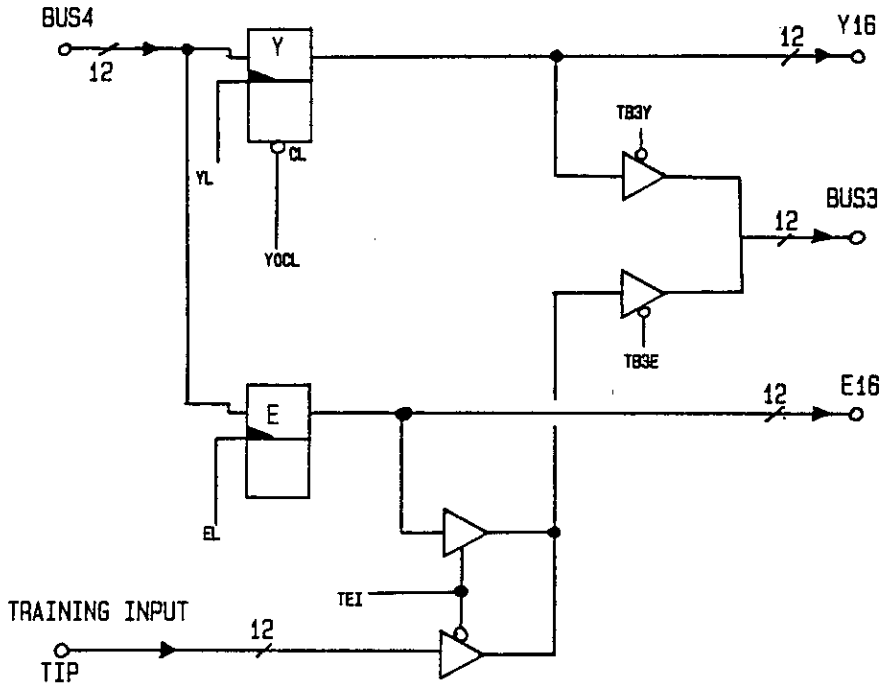


Fig 4.5

I/O BUS INTERFACE

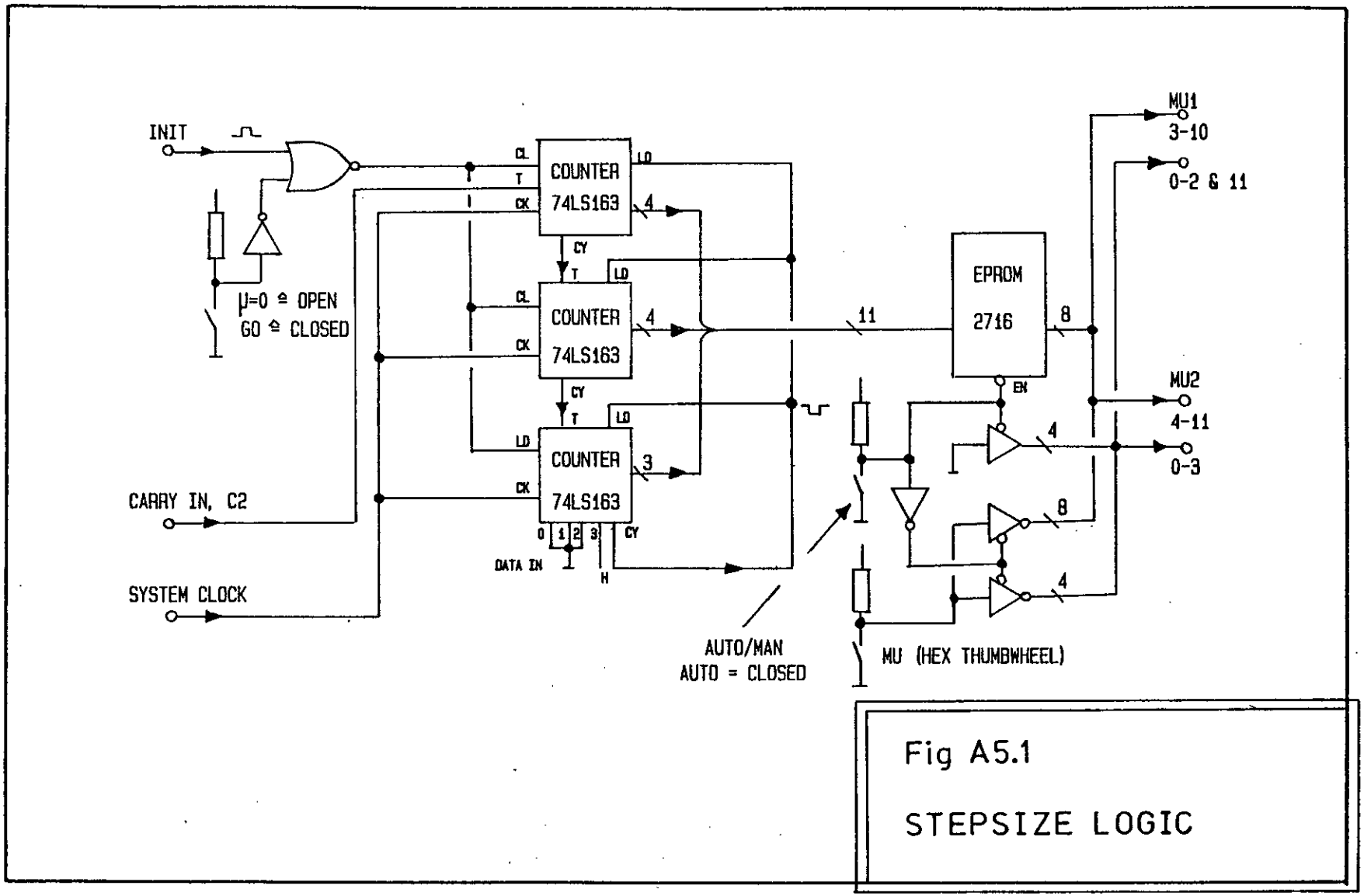


Fig A5.1

STEPSIZE LOGIC

APPENDIX B: Software

This appendix contains some of the software used in this thesis. The simulation programme and the spectral estimation programme are members of families of similar programmes, each with a slightly different purpose. Only one member of either family is included here. The programmes are all written in DEC 10 FORTRAN, a language very similar to ANSI FORTRAN. All programmes call subroutines; the subroutines themselves are listed last.

CONTENTS:

Some subroutines call NAG and GINO library packages for certain arithmetic and graphic functions. Further data on these packages is available from:

R.Hare (ed), Edinburgh DECsystem - 10 Installation Manual, University of Edinburgh, 1983.

```
      PROGRAM SATSIM ! M.J.Rutter 28.1.83
c     ************************************************************************

c     SATORIUS ALGORITHM LATTICE SIMULATOR

c     This program simulates a lattice equaliser converging onto
c     a channel.  There are facilities for making a number of
c     runs (number determined by the value of MUEND) possibly
c     with reducing MU values, and taking only the best results
c     from each.
c     The program works as follows:

c     1.  Declarations
c     2.  Initialisation
c     3.  Read in impulse response of line
c     4.  Read in filter length
c     5.  For various Mu values
c        6.  Iterate to 500
c           7.  Generate a data pair
c           8.  Run through the filter
c           9.  If iteration is a wanted value
c              10. Calculate system response
c              11. Calculate error power
c              12. If this is best result for Iteration No.
c                 13. Record it in array
c              14. Continue
c           15. Continue
c        16. Continue
c     17. Continue
c     18. Store run data on disc
c     19. End

c     ------------------------------------------------------------------


c     1.  Declarations

      REAL
     .MUREL,              ! Input MU rel to 1/(N*R(0))
     .MU1,                ! Stepsize of lattice
     .MU2,                ! Stepsize of side-taps
     .MU(2,64),           ! Stepsize array
     .ALPHA,              ! Stepsize time constant
     .BETA,               ! Stepsize scaling constant
     .AO,                 ! Power gain of channel
     .IMPRE,              ! Real part of impulse response
     .IMPIM,              ! Imaginary ....
     .REPOW,              ! Error power
     .ERRAY(3,50)         ! Array of error powers

      COMPLEX
     .IMPUL1(32),         ! Channel impulse response
     .LINE,               ! Convolved PN sequence
     .PURE,               ! Training signal
     .K(64),              ! Lattice coefficients
     .G(64),
     .BO(64),             ! Lattice signal variables
     .BN(64),
     .P(64),
     .V(64),
     .CONVOL(64)          ! System impulse response

      INTEGER
     .I,                  ! Loop counter
     .ITNO,               ! Iteration number
     .INC1,               ! Database sample counter
     .INC2,               ! Iteration tally
     .MUEND,              ! Final mu run
     .ABORT,              ! =1 if run to be aborted
     .LEN1,               ! Length of channel impulse response
     .IMPNO,              ! Code of channel impulse response
     .LEN2,               ! Length of equaliser
     .DELAY               ! Delay on training signal
```

```
c     2.  Initialisation

      OPEN(UNIT=50, FILE='IMPULS', DEVICE='DSK')
      OPEN(UNIT=60, FILE='HSAT', DEVICE='DSK', ACCESS='APPEND')

      DO 200 I = 1, 32, 1
         IMPUL1(I) = (0,0)
 0200 CONTINUE

      CALL DATA(LINE,PURE,IMPUL1,-1)
      DO 220 I = 1, 30, 1
         CALL DATA(LINE,PURE,IMPUL1,1)
 0220 CONTINUE

      DO 230 I = 1, 50, 1
         ERRAY(1,I) = 10
 0230 CONTINUE



c     3.  Read in impulse response of line

      READ(50,*) LEN1, IMPNO
      WRITE(60,301) IMPNO
 0301 FORMAT(I3, '  =IMPULSE RESPONSE NUMBER')

      A0 = 0
      DO 300 I = (35-LEN1)/2, (33+LEN1)/2, 1            ! Main lobe at tap 17
         READ(50,*) IMPRE, IMPIM
         IMPUL1(I) = CMPLX(IMPRE, IMPIM)
         A0 = A0 + IMPRE*IMPRE + IMPIM*IMPIM
 0300 CONTINUE



c     4.  Read in filter length

      TYPE 401
 0401 FORMAT(' ? ',$)
      ACCEPT *, LEN2

      DELAY = 17 + LEN2/2



c     5.  For various Mu values

      MUEND = 6
      ALPHA = 0.07
      I = 0
 0500 I = I + 1
         TYPE *,I
         ABORT = 0

         MUREL = 0.707**I
         MUREL = MUREL * 1.414
         MU2 = MUREL/(A0*LEN2*2)
         BETA = LEN2/MUREL
         MU1 = MU2 * 0.5

         CALL RESET(K,G,B0,MU,MU1,MU2)

         INC1 = 1
         INC2 = 1



c        6.  Iterate to 500

         DO 600 ITNO = 1, 500, 1
            IF (ABORT.NE.0) GOTO 600
```

```
c           7.  Generate a data pair

            CALL DATA(LINE,PURE,IMPUL1,DELAY)


c           8.  Run through the filter

            CALL SFILT(LINE,PURE,F,BN,BO,V,K,G,MU,LEN2,ALPHA,BETA)


c           9.  If iteration is a multiple of ten

            IF (ITNO.LT.INC2) GOTO 900


c              10. Calculate system response

               CALL LFILTX(K,G,IMPUL1,CONVOL,LEN2)


c              11. Calculate error power

               CALL ERRCAL(CONVOL,REPOW)

               IF (REPOW.LT.30) GOTO 1100
                  MUEND = MUEND + 1
                  ABORT = 1
                  TYPE 1110
 1110             FORMAT(' ABORT')
                  GOTO 600
 1100          CONTINUE


c              12. If this is best result for Iteration No.

               IF (ERRAY(1,INC1).LT.REPOW) GOTO 1210


c                 13. Record it in array

 1200             ERRAY(1,INC1) = REPOW
                  ERRAY(2,INC1) = MU(2,1)
                  ERRAY(3,INC1) = INC2


c              14. Continue

 1210          CONTINUE


c           15. Continue

               INC1 = INC1 + 1
               INC2 = INC1 + INT(10**(FLOAT(INC1)*ALOG10(450.5)/50))
 0900       CONTINUE


c          16. Continue

 0600     CONTINUE


c       17. Continue

        IF (I.LT.MUEND) GOTO 500
```

```
c      18. Store run data on disc

       WRITE(60,1801) LEN2
 1801 FORMAT(I3, '  = Length of filter')

       DO 1800 I = 1, 50, 1
          WRITE(60,*) ERRAY(3,I), ERRAY(1,I), ERRAY(2,I)
 1800 CONTINUE


c      19. End

       END


c      ------------------------------------------------------------
```

```
c      M.J.Rutter 14.7.82 PE10.FOR
c      ############################################################

c            AUTOREGRESSIVE SPECTRAL ESTIMATION

c      This program takes a file of HEX PARCOR coefficients, and turns
c      them into an interpolated surface of freq-vs-order.
c      It works as follows:

c      1.  Declarations and initialisation
c      2.  Generate ACF
c      3.  Calculate PE coefficients
c      4.  Calculate MEM spectrum
c      5.  Plot MEM spectrum
c      6.  End

c      ------------------------------------

c      1.  Declarations and initialisation

       REAL
      .ZPLOT(181,128),          ! Surface contours
      .ZMIN,                    ! Minimum value in ZPLOT
      .ALIN,                    ! Used to find active lines of surf.
      .PII,                     ! Angular conversion factor
      .P,                       ! Filter PE power
      .ACF(20),                 ! Autocorrelation function
      .K(20),                   ! PARCOR coefficients
      .PM(2048),                ! Filter family PE powers
      .A(256),                  ! PE filter coefficients
      .B(2048),                 ! Scratchpads used in Fourier transformations
      .SPAD(74000),
      .SPAD2(2048)

       INTEGER
      .ISPAD(256),
      .ASPECT,                  ! Nearest corner of surface
      .I,J                      ! Loop counters


c      2.  Generate ACF
      OPEN(UNIT=50,DEVICE='DSK',FILE='K.DAT')


      READ(50,101) N,F

 0101 FORMAT(I,F)
      DO 100 I = 1, N, 1
         READ(50,111) (ISPAD(J), J=1,4)
 0111    FORMAT(4A1)
         ITEMP = 256*KONVRT(ISPAD(2)) + 16*KONVRT(ISPAD(3))
         ITEMP = ITEMP + KONVRT(ISPAD(4)) + 4096*KONVRT(ISPAD(1))
         IF (ITEMP.GT.32767) ITEMP=ITEMP-65536
         K(I) = FLOAT(ITEMP)/32768
 0100 CONTINUE


c      3.  Calculate PE coefficients

      DO 300 I = 1, 16, 1
         J=I
         CALL LEVIN(K,J,ACF,A,PM,SPAD)


c      4.  Calculate MEM spectrum

         P = PM(I+1)
         J=I
         CALL MEMS(A,J,P,256,ISPAD,B,SPAD,SPAD2)
```

```
c      5.  Plot MEM spectrum

         DO 500 J = 1, 128, 1
            ZPLOT((I*12-11), J) = A(129-J)
            IF (I.EQ.1) GOTO 510
            DO 510 L = 1, 11, 1
               ZPLOT((I*12-23+L), J) = (L*ZPLOT((I*12-11),J)
                                     + (12-L)*ZPLOT((I*12-23),J))/12
 0510       CONTINUE
 0500    CONTINUE
 0300 CONTINUE


      CALL SURF(1.,8.6,16.,0.,ZPLOT,181,128,0.3,0,ISPAD,SPAD)


c      6.  End

      STOP
      END


c      ------------------------------------
```

```
      PROGRAM CHANAN ! M.J.Rutter 9.9.82
C     ###################################################################

C             CHANNEL ANALYSER

C     This program takes a channel impulse response and calculates
C     the eigenvalues of its autocovarience matrix at rank 8.
C     Then it calculates its inverse at that rank and convolves
C     it with the channel impulse response to form a system
C     impulse response.  From this it calculates the
C     relative off-lobe power in dB and eye openness.
C     It outputs max/min eigenvalue, min/power, error power and
C     eye openness.
C     Then it repeats the process for 11, 16, 22 and 32 taps.
C     It works as follows:

C     1.  Declarations.
C     2.  Initialisation.
C     3.  For taps = 8, 11, 16, 22, 32.
C        4.  Calculate eigenvalues of channel.
C        5.  Calculate inverse filter.
C        6.  Calculate system impulse response.
C        7.  Calculate error power and eye openness.
C        8.  Output data to file.
C     9.  Next taps.
C     10. End.

C     ------------------------------------------------


C     1.  Declarations.

      REAL
     .IMPRE,              ! Real component of impulse response
     .IMPIM,              ! Imaginary .....
     .EIG(32),            ! Channel eigenvalues
     .POWER,              ! Power gain of channel
     .EVR,                ! Eigenvalue ratio
     .EVMIN,              ! Min eigenvalue (rel)
     .REPOW,              ! Error power
     .FRACT               ! Eye openness

      COMPLEX
     .IMPUL1(32),         ! Input impulse response
     .IMPUL2(32),         ! Inverse impulse response
     .CONVOL(64)          ! System impulse response

      INTEGER
     .LEN1,               ! Length of IMPUL1
     .CHAN,               ! Name of channel
     .TAPS                ! Number of equaliser taps in use


C     2.  Initialisation.

      OPEN(UNIT=50, FILE='IMPULS', DEVICE='DSK')
      OPEN(UNIT=60, FILE='CHANAN', DEVICE='DSK', ACCESS='APPEND')

      READ(50,*) LEN1, CHAN
      WRITE(60,*) LEN1, CHAN

      DO 200 I = 1, LEN1, 1
         READ(50,*) IMPRE, IMPIM
         IMPUL1(I) = CMPLX(IMPRE, IMPIM)
 0200 CONTINUE


C     3.  For taps = 8, 11, 16, 22, 32.

      DO 300 I = 1, 5, 1
         TAPS = INT(8 * 2**(0.5 * FLOAT(I-1)))
```

```
c       4.  Calculate eigenvalues of channel.

        CALL EIGEN(IMPUL1,LEN1,EIG,TAPS,POWER)

        EVR = EIG(TAPS) / EIG(1)
        EVMIN = POWER / EIG(1)


c       5.  Calculate inverse filter.

        CALL WIENER(IMPUL1,LEN1,IMPUL2,TAPS)


c       6.  Calculate system impulse response.

        CALL CONV(IMPUL1,LEN1,IMPUL2,TAPS,CONVOL)


c       7.  Calculate error power and eye openness.

        CALL ERRCAL(CONVOL,REPOW)
        REPOW = 10 * ALOG10(REPOW)

        CALL EYE(CONVOL,FRACT)


c       8.  Output data to file.

        WRITE(60,*) TAPS,EVR,EVMIN,REPOW,FRACT


c     9.  Next taps.

 0300 CONTINUE


c     10. End.

      STOP 'Data appended to CHANAN.DAT'
      END


c     ------------------------------------------
```

```
      PROGRAM SPROGS     | M.J.Rutter 31.1.83
C     ************************************************************************

C                SIGNAL PROCESSING SUBROUTINES

C     Contents:

C     Calcom
C     Conv
C     Data
C     Durbin
C     Eigen
C     Errcal
C     Eye
C     Konvrt
C     Latinv
C     Levin
C     Lfilt
C     Lfiltx
C     Mems
C     Nuplo
C     Reset
C     Screen
C     Sfilt
C     Surf
C     Tfilt
C     Tfiltx
C     Wiener


      END



C     ------------------------------------------------------------------------


C     Calcomp plotter, 13.7.82
      SUBROUTINE CALCOM(XMIN,YMIN,XMAX,YMAX,XPLOT,YPLOT,POINTS,
     .                  F,ISPAD)

C     This subroutine plots a graph on the Calcomp plotter.  It is of
C     correct size for direct use in a thesis.  It adds identifying
C     text which may later be whited out:  Date-time + a real
C     number.


      REAL XMIN,YMIN,XMAX,YMAX,  ! The numerical limits of the axes
     .     XPLOT(10),YPLOT(10),  ! The positions of the points
     .     F                     ! A real for graph ID solely
      INTEGER POINTS             ! The number of points to be plotted
      INTEGER ISPAD(4)           ! Scratchpad used for date/time


      CALL CC1051
      CALL DEVPAP(220.,300.,1)
      CALL PENSEL(2,0.2,3)

      CALL AXIPOS(1,57.,94.,125.,1)
      CALL AXIPOS(1,57.,94.,175.,2)

      CALL AXISCA(1,12,XMIN,XMAX,1)
      CALL AXISCA(1,17,YMIN,YMAX,2)

      CALL NUPLO(XPLOT,YPLOT,POINTS)
      CALL NUPLO(XPLOT,YPLOT,POINTS)
      CALL NUPLO(XPLOT,YPLOT,POINTS)
      CALL NUPLO(XPLOT,YPLOT,POINTS)

      CALL MOVTO2(60.,70.)
      CALL CHAARR(ISPAD,4,5)
      CALL MOVTO2(60.,50.)
      CALL CHAFLO(F,15)
```

```
      CALL MOVTO2(0.,0.)
      CALL CHAHOL(3H+*.)
      CALL MOVTO2(210.,0.)
      CALL CHAHOL(3H+*.)
      CALL MOVTO2(210.,297.)
      CALL CHAHOL(3H+*.)
      CALL MOVTO2(0.,297.)
      CALL CHAHOL(3H+*.)

      CALL MOVTO2(41.,42.)
      CALL CHAHOL(3H.*.)
      CALL MOVTO2(185.,42.)
      CALL CHAHOL(3H.*.)
      CALL MOVTO2(185.,278.)
      CALL CHAHOL(3H.*.)
      CALL MOVTO2(41.,278.)
      CALL CHAHOL(3H.*.)

      CALL DEVEND

      RETURN
      END


C     ------------------------------------------------------------------


C     Complex convolver 20.7.82
      SUBROUTINE CONV( IMPUL1,LEN1,IMPUL2,LEN2,CONVOL)


C     This subroutine convolves two complex impulse responses.
C     The lowest array number corresponds to the earliest event.

      COMPLEX
     .IMPUL1,            ! Impulse responses
     .IMPUL2,
     .CONVOL             ! Convolution

      INTEGER
     .LEN1,LEN2,         ! Impulse response lengths
     .I,J,K              ! Loop counters

      DIMENSION IMPUL1(LEN1), IMPUL2(LEN2), CONVOL(64)



      DO 0200 I = 1, 64, 1
         CONVOL(I) = (0,0)
0200  CONTINUE



      DO 500 I = 1, LEN1, 1
         DO 550 J = 1, LEN2, 1
            K = I + J - 1
            CONVOL(K) = CONVOL(K) + IMPUL1(I) * IMPUL2(J)
0550     CONTINUE
0500  CONTINUE



      RETURN
      END


C     ------------------------------------------------------------------
```

```
c     Line simulator subroutine, 24.8.82
      SUBROUTINE DATA(LINE,PURE,IMPULS,DELAY)


c     This subroutine generates a 25 stage 2-d PN binary signal
c     and convolves it with a 32-point impulse response.  The
c     outputs are the binary and convolved signal points.
c     DELAY gives the tap from which the training signal is taken
c     A delay of 17 matches the centres of the signals for
c     32-tap impulse responses (ie the main tap is tap m=17).
c     If the equaliser is N taps long, and if the equaliser tap one
c     instantly becomes the output of the channel simulator with no
c     clocking delays, then the training signal (Pure) should come
c     from tap Train(Delay) where Delay = (M + N/2).
c     Remember that the 2-d PN signal has a power of 2, not 1.
c     The program works as follows:

c     1.  Declarations
c     1a. Initialisation
c     2.  Generate new binary data point
c     3.  Convolve data points with impulse response
c     4.  Output binary data
c     5.  Clock binary data
c     6.  return

c     ---------------------------------------------

c    1.  Declarations

      REAL
     .PNREAL,            ! Real componenent of new point
     .PNIMAG             ! Imaginary ....

      COMPLEX
     .LINE,              ! Distorted data output
     .PURE,              ! Binary data point
     .IMPULS(32),        ! Line impulse response
     .TRAIN(64)          ! Array of binary data points

      INTEGER
     .DELAY,             ! Training signal offset
     .I                  ! Loop counter


c     1a. Initialisation

      IF (DELAY.GT.0) GOTO 110
      DELAY = 1
      DO 110 I = 1, 64, 1
         TRAIN(I) = (-1., -1.)
 0110 CONTINUE


c     2.  Generate new binary data point

      PNREAL = AIMAG(TRAIN(2)) * AIMAG(TRAIN(13))
      PNIMAG = REAL(TRAIN(3)) * REAL(TRAIN(14))
      TRAIN(1) = CMPLX(SIGN(1.,PNREAL), SIGN(1.,PNIMAG))


c     3.  Convolve data points with impulse response

      LINE = (0,0)
      DO 100 I = 1, 32, 1
         LINE = LINE + IMPULS(I) * TRAIN(I)
 0100 CONTINUE


c     4.  Output binary data

      PURE = TRAIN(DELAY)
```

```
c     5.  Clock binary data

      DO 0200 I = 1, 63, 1
         TRAIN(65-I) = TRAIN(64-I)
 0200 CONTINUE


c     6.  Return

      RETURN
      END


c     ================================================================


c     Levinson Autocorr -> K, 19.8.82
      SUBROUTINE DURBIN(K,N,R,A,PM,SPAD)

c     This subroutine takes a set of autocorrelation coefficients
c     and converts them into K-values (PARCOR coefficients) and
c     PE-filter weights.
c     Since Fortran has no zeroeth element, all R, A and PM values
c     are slid up by one.  For (N+1) input values, there will be (N+1)
c     A and PM values and N K-values.
c     It works as follows:

c     1.  Declarations
c     2.  Initial calculations
c     3.  For each filter order:
c        4.  Calculate K-value
c        5.  Calculate prediction error
c        6.  Calculate PE filter coefficients
c     7.  Next filter order
c     8.  Return

c     ----------------------------------------


c     1.  Declarations

      REAL
     .K(40),              ! PARCOR Coefficients
     .R(40),              ! Autocorrelation coefficients
     .A(40),              ! PE Coefficients
     .PM(40),             ! PE Power
     .SPAD(40)            ! Scratchpad

      INTEGER
     .N,                  ! Number of PARCOR coefficients wanted
     .ORD,                ! Order of filter being currently calculated
     .I                   ! Loop counter


c     2.  Initial calculations

      DO 200 I = 1, (N+1), 1
         K(I) = 0
         PM(I) = 0
         A(I) = 0
 0200 CONTINUE

      PM(1) = R(1)
      K(1)  = R(2) / R(1)
      PM(2) = PM(1) * (1 - K(1)*K(1))
      A(1)  = 1
      A(2)  = -K(1)
```

```
c     3.  For each filter order

      IF (N.LT.2) GOTO 300
      DO 300 I = 2, N, 1

         IF ( PM(I).LT.1E-10) GOTO 300


c        4.  Calculate K-value

         K(I) = R(I+1)
         DO 400 J=1, I-1, 1
            K(I) = K(I) + A(J+1) * R(I-J+1)
 0400    CONTINUE
         K(I) = K(I) / PM(I)


c        5.  Calculate prediction error

         PM(I+1) = PM(I) * (1 - K(I)*K(I))


c        6.  Calculate PE coefficients

         A(I+1) = 0
         DO 600 J = 2, (I+1), 1
            SPAD(J) = A(J) - K(I) * A(I+2-J)
 0600    CONTINUE

         DO 610 J = 2, (I+1), 1
            A(J) = SPAD(J)
 0610    CONTINUE


c     7.  Next filter order

 0300 CONTINUE


c     8.  Return

      PM(N+2) = 42
      A(N+2)  = 42
      K(N+1)  = 42

      RETURN
      END


c     ------------------------------------------------------------------------


c     Eigenvalue calculator 21.7.82
      SUBROUTINE EIGEN( IMPUL1,LEN1,EIG,RANK,POWER)

c     This program calculates the Eigenvalues
c     of a given impulse response.
c     The program works as follows:

c     1.    Declarations
c     2.    Initialisation
c     3.    Generate autocorrelation function
c     4.    Generate autocorrelation matrix
c     5.    Call eigenvalue subroutine
c     6.    End

c     ------------------------------------------------------------------------
```

```
c     1.   Declarations

      REAL
     .POWER,
     .AR(32,32),          ! Autocorrelation matrix
     .AI(32,32),
     .EIG(32),            ! Eigenvalues
     .SPAD1(32),                 ! Scratchpads for NAG routine
     .SPAD2(32),
     .SPAD3(32)

      INTEGER
     .RANK,               ! Rank of autocorrelation matrix
     .LEN1,               ! Number of input data points
     .IFAIL,              ! Error code for NAG routine
     .I,
     .J,
     .K

      COMPLEX
     .IMPUL1(32),         ! Line impulse response
     .ACF(64)             ! Autocorrelation array


c     2.   Initialisation

      DO 0200 I = 1, 64, 1
         ACF(I) = (0,0)
 0200 CONTINUE


c     3.   Generate autocorrelation function

      DO 400 I = 1, LEN1, 1
         DO 450 J = 1, LEN1, 1
            K = I - J + 32
            ACF(K) = ACF(K) + IMPUL1(I) * CONJG(IMPUL1(J))
 0450    CONTINUE
 0400 CONTINUE
      POWER = CABS(ACF(32))


c     4.   Generate autocorrelation matrix

      DO 500 I = 1, RANK, 1
         DO 550 J = 1, RANK, 1
            K = I - J + 32
            AR(I,J) = REAL(ACF(K))
            AI(I,J) = AIMAG(ACF(K))
 0550    CONTINUE
 0500 CONTINUE


c     5.   Call eigenvalue subroutine

      IFAIL = 0
      CALL F02AWF(AR,32,AI,32,RANK,EIG,
                  SPAD1,SPAD2,SPAD3,IFAIL)


c     6.   End

      RETURN
      END


c     ------------------------------------------------------------------
```

```
c     Inverse filter error calculator, 9.9.82
      SUBROUTINE ERRCAL(CONVOL,REPOW)


c     This subroutine takes the array obtained by convolving a line
c     impulse response with a filter impulse response.  It calculates
c     the error power of the inverse filter as a fraction of that
c     in the main lobe.
c     It works as follows:

c     1.  Declarations
c     2.  Initialisation
c     3.  Find unit impulse
c     4.  Calculate error power
c     5.  Return

c     ------------------------

c     1.  Declarations

      REAL
     .RMAX,              ! Main lobe of impulse response
     .REPOW              ! Error power

      COMPLEX
     .CONVOL(64)         ! System impulse response

      INTEGER I          ! Loop counter


c     3.  Find unit impulse

      RMAX = REAL(CONVOL(1))
      DO 0600 I = 1, 64, 1
         IF (REAL(CONVOL(I)).LT.RMAX) GOTO 0600
         RMAX = REAL(CONVOL(I))
 0600 CONTINUE


c     4.  Calculate error power

      REPOW = 1 - 2*RMAX
      DO 0700 I = 1, 64, 1
         REPOW = REPOW + REAL(CONVOL(I) * CONJG(CONVOL(I)))
 0700 CONTINUE


c     5.  Return

      RETURN
      END


c     ------------------------------------------------------------------


c     Eye openness calculator, 20.7.82
      SUBROUTINE EYE(CONVOL,FRACT)


c     This subroutine takes the array obtained by convolving a line
c     impulse response with a filter impulse response.  It calculates
c     the maximum error volts of the inverse filter.
c     It expresses it as a fraction of the main lobe.
c     If Fract is negative, the eye is closed.
c     (The maths assumes a quadrature binary input.)
c     It works as follows:

c     1.  Declarations
c     2.  Initialisation
c     3.  Find unit impulse
c     4.  Calculate error power
c     5.  Return

c     ------------------------
```

```
c     1.  Declarations

      REAL
     .RMAX,              ! Main lobe of impulse response
     .FRACT              ! Total potential error volts

      COMPLEX
     .CONVOL(64)         ! System impulse response

      INTEGER I          ! Loop counter


c     3.  Find unit impulse

      RMAX = REAL(CONVOL(1))
      DO 0600 I = 1, 64, 1
         IF (REAL(CONVOL(I)).LT.RMAX) GOTO 0600
         RMAX = REAL(CONVOL(I))
 0600 CONTINUE


c     4.  Calculate error power

      RMAX = ABS(RMAX)
      FRACT = RMAX * 2
      DO 0700 I = 1, 64, 1
         FRACT = FRACT - ABS(REAL(CONVOL(I)))
         FRACT = FRACT - ABS(AIMAG(CONVOL(I)))
 0700 CONTINUE

      FRACT = FRACT / RMAX


c     5.  Return

      RETURN
      END


c     ------------------------------------------------------------------------




c     Hexadecimal conversion subroutine, 14.6.82
      INTEGER FUNCTION KONVRT(I)

c     This function converts an ASCII HEX character into an integer

      KONVRT = -999999
      IF (I.EQ.'0') KONVRT=0
      IF (I.EQ.'1') KONVRT=1
      IF (I.EQ.'2') KONVRT=2
      IF (I.EQ.'3') KONVRT=3
      IF (I.EQ.'4') KONVRT=4
      IF (I.EQ.'5') KONVRT=5
      IF (I.EQ.'6') KONVRT=6
      IF (I.EQ.'7') KONVRT=7
      IF (I.EQ.'8') KONVRT=8
      IF (I.EQ.'9') KONVRT=9
      IF (I.EQ.'A') KONVRT=10
      IF (I.EQ.'B') KONVRT=11
      IF (I.EQ.'C') KONVRT=12
      IF (I.EQ.'D') KONVRT=13
      IF (I.EQ.'E') KONVRT=14
      IF (I.EQ.'F') KONVRT=15

      RETURN
      END


c     ------------------------------------------------------------------------
```

```
c     Levinson matrix inverter, 19.8.82
      SUBROUTINE LATINV(K,N,R,A,PM,P,G,H,SPAD)

c     This subroutine takes a set of autocorrelation coefficients
c     and cross-correlation coefficients,
c     and converts them into K-values (PARCOR coefficients)
c     G-values (sidetaps) and PE-filter weights.
c     Then it calculates H, the inverse filter taps.
c     Since Fortran has no zeroeth element, all R,G,H,A and PM values
c     are slid up by one.  For (N+1) input values, there will be (N+1)
c     G,H,A and PM values and N K-values.
c     It works as follows:

c     1.  Declarations
c     2.  Initial calculations
c     3.  For each filter order:
c        4.  Calculate K-value
c        5.  Calculate prediction error
c        6.  Calculate PE filter coefficients
c        7.  Calculate lattice sidetaps
c        8.  Calculate transversal taps
c     9.  Next filter order
c     10. Return

c     ---------------------------------------


c     1.  Declarations

      REAL
     .K(40),              ! PARCOR Coefficients
     .R(40),              ! Autocorrelation coefficients
     .A(40),              ! PE Coefficients
     .PM(40),             ! PE Power
     .P(40),              ! Cross-correlation coefficients
     .G(40),              ! Lattice side-taps
     .H(40),              ! Transversal tap
     .SPAD(40)            ! Scratchpad

      INTEGER
     .N,                  ! Number of PARCOR coefficients wanted
     .ORD,                ! Order of filter being currently calculated
     .I                   ! Loop counter


c     2.  Initial calculations

      DO 200 I = 1, (N+1), 1
         K(I) = 0
         G(I) = 0
         PM(I) = 0
         A(I) = 0
         H(I) = 0
 0200 CONTINUE

      PM(1) = R(1)
      K(1)  = R(2) / R(1)
      PM(2) = PM(1) * (1 - K(1)*K(1))
      A(1)  = 1
      A(2)  = -K(1)
      G(1) = P(1) / PM(1)
      G(2) = (P(1)*A(2) + P(2)*A(1)) / PM(2)
      H(1) = G(1) + G(2)*A(2)
      H(2) = G(2)


c     3.  For each filter order

      IF (N.LT.2) GOTO 300
      DO 300 I = 2, N, 1

         IF ( PM(I).LT.1E-10) GOTO 300
```

```
c        4.  Calculate K-value

         K(I) = R(I+1)
         DO 400 J=1, I-1, 1
            K(I) = K(I) + A(J+1) * R(I-J+1)
 0400    CONTINUE
         K(I) = K(I) / PM(I)


c        5.  Calculate prediction error

         PM(I+1) = PM(I) * (1 - K(I)*K(I))


c        6.  Calculate PE coefficients

         A(I+1) = 0
         DO 600 J = 2, (I+1), 1
            SPAD(J) = A(J) - K(I) * A(I+2-J)
 0600    CONTINUE

         DO 610 J = 2, (I+1), 1
            A(J) = SPAD(J)
 0610    CONTINUE


c        7.  Calculate lattice sidetaps

         G(I+1) = 0
         DO 700 J = 1, (I+1), 1
            G(I+1) = G(I+1) + A(J)*P(I+2-J)
 0700    CONTINUE
         G(I+1) = G(I+1) / PM(I+1)


c        8.  Calculate transversal taps

         H(I+1) = G(I+1)
         DO 800 J = 1, I, 1
            H(J) = H(J) + G(I+1)*A(I+2-J)
 0800    CONTINUE


c     9.  Next filter order

 0300 CONTINUE


c     10. Return

      G(N+2) = 42
      PM(N+2) = 42
      A(N+2)  = 42
      K(N+1)  = 42

      RETURN
      END


c     ================================================================


c     Levinson-Durbin recursion, 25.6.82
      SUBROUTINE LEVIN(K,N,R,A,PM,SPAD)

c     This subroutine takes a set of lattice coefficients and converts
c     them into an autocorrelation function and a PE filter.
c     The Prediction Error is also given.
c     Since Fortran has no zeroeth element, all R and A array points
c     are slid up by one.  For N input values there will be (N+1)
c     A, PM and R output values.
```

```
      REAL
     .K(40),              ! PARCOR Coefficients
     .R(40),              ! Autocorrelation coefficients
     .A(40),              ! PE Coefficients
     .PM(40),             ! PE Power
     .SPAD(40)            ! Scratchpad

      INTEGER
     .N,                  ! Number of PARCOR coefficients supplied
     .ORD,                ! Order of filter being currently calculated
     .I                   ! Loop counter

      PM(1) = 1
      A(1) = 1
      A(2) = -K(1)
      PM(2) = 1 - K(1)*K(1)
      R(1) = 1
      R(2) = K(1)


      DO 100 ORD = 3, (N+1), 1
         R(ORD) = 0
         PM(ORD) = 0
         A(ORD) = 0
0100  CONTINUE


      IF (N.LT.2) GOTO 200
      DO 0200 ORD = 2, N, 1

         IF (PM(ORD).LT.1E-10) GOTO 0200

         R(ORD+1) = K(ORD) * PM(ORD)
         DO 0300 I = 1, (ORD-1), 1
            R(ORD+1) = R(ORD+1) - A(I+1)*R(ORD - I + 1)
0300     CONTINUE

         PM(ORD+1) = PM(ORD) * (1 - K(ORD)*K(ORD))

         A(ORD+1) = 0
         DO 0500 I = 1, ORD, 1
            SPAD(I+1) = A(I+1) - K(ORD)*A(ORD - I + 1)
0500     CONTINUE
         DO 0600 I = 1, ORD, 1
            A(I+1) = SPAD(I+1)
0600     CONTINUE

0200  CONTINUE

      PM(N+2) = 43
      A(N+2)  = 43
      R(N+2)  = 43

      RETURN
      END


C     ------------------------------------------------------------------


C     Lattice filter subroutine, 18.8.82
      SUBROUTINE LFILT(LINE,PURE,F,BN,BO,V,K,G,MU,LENGTH)


C     This subroutine acts as a digital lattice equaliser.
C     The line signal and training signal are input and are unaltered
C     on return.  The output is the equalised signal.
C     The program works as follows:

C     1.  Declarations
C     2.  Data input
C     3.  For every stage:
C        4.  Calculate lattice values
C        5.  Calculate equaliser values
C        6.  Adapt lattice coefficients
C        7.  Adapt equaliser coefficients
C        8.  Clock delays
C     9.  Next stage
C     11. Return

C     ------------------------------------
```

```
c     1.  Declarations

      REAL
     .MU(2,64)          ! Equaliser stepsizes

      COMPLEX
     .LINE,             ! Filter input
     .PURE,             ! Training input
     .F(64),            ! Filter signal values
     .BN(64),
     .BO(64),
     .V(64),
     .K(64),            ! Filter coefficients
     .G(64)

      INTEGER
     .LENGTH,           ! Filter length
     .N                 ! Loop counter


c     2.  Data input

      F(1) = LINE
      BN(1) = LINE
      V(1) = PURE


c     3.  For every stage:

      DO 0300 N = 1, LENGTH, 1


c        4.  Calculate lattice values

         F(N+1) = F(N) - K(N) * BO(N)
         BN(N+1) = BO(N) - CONJG(K(N)) * F(N)


c        5.  Calculate equaliser values

         V(N+1) = V(N) - G(N) * BN(N)


c        6.  Adapt lattice coefficients

         K(N) = K(N) + MU(1,N) * (F(N) * CONJG(BN(N+1))
                              + F(N+1) * CONJG(BO(N)))

         IF (CABS(K(N)).GT.1) K(N) = CMPLX(REAL(K(N))/CABS(K(N)),
                                           AIMAG(K(N))/CABS(K(N)))


c        7.  Adapt equaliser coefficients

         G(N) = G(N) + V(N+1) * CONJG(BN(N)) * MU(1,N)


c        8.  Clock delays

         BO(N) = BN(N)


c     9.  Next stage

 0300 CONTINUE
```

```
c     11. Return

      RETURN
      END



c     ------------------------------------------------------------------


c     Lattice convolution subroutine, 15.10.82
      SUBROUTINE LFILTX(K,G,IMPUL2,CONVOL,LENGTH)


c     This subroutine convolves the input impulse response with the
c     filter.
c     It calls subroutine Conv.
c     It works as follows:

c     1.  Declarations
c     2.  Initialisation
c     3.  Get lattice impulse response
c     4.  Convolve with channel
c     5.  Return

c     ---------------------------

c     1.  Declarations

      COMPLEX
     .K(64),              ! Filter coefficients
     .G(64),
     .XBO(64),            ! Filter signal values
     .XBN(64),
     .XF(64),
     .XV(64),
     .CONVOL(64),         ! Convolved output array
     .IMPUL1(64),         ! Filter impulse response
     .IMPUL2(64)          ! Line impulse response

      INTEGER
     .LENGTH,             ! Filter length
     .I,N                 ! Loop counters


c     2.  Initialisation

      DO 200 I = 1, 64, 1
         XBO(I) = (0,0)
 0200 CONTINUE

      XV(1) = (0.,0.)


c     3.  Get lattice impulse response

      DO 0300 I = 1, LENGTH, 1

         XF(1) = (0.,0.)
         IF (I.NE.1) GOTO 330
            XF(1) = (1.,0.)
 0330    XBN(1) = XF(1)

         DO 0370 N = 1, LENGTH, 1
            XF(N+1) = XF(N) - K(N)*XBO(N)
            XBN(N+1) = XBO(N) - CONJG(K(N))*XF(N)
            XV(N+1) = XV(N) - G(N)*XBN(N)
            XBO(N) = XBN(N)
 0370    CONTINUE

         IMPUL1(I) = - XV(LENGTH + 1)

 0300 CONTINUE
```

```
c     4.  Convolve with channel

      CALL CONV(IMPUL1,LENGTH,IMPUL2,32,CONVOL)


c     5.  Return

      RETURN
      END


c     ================================================================


c     MEM spectrum, 24.6.82
      SUBROUTINE MEMS(A,N,P,M,ISPAD,B,SPAD,SPAD2)


c     This subroutine takes the input array of (N+1)
c     PE filter coefficients and
c     uses them to calculate the MEM spectrum.

      REAL
     .A(10),              ! PE coefficients & Output array
     .B(10),              ! Scratchpad for imaginary components
     .P,                  ! PE power
     .SPAD(10),           ! Scratchpads for NAG routine
     .SPAD2(10)

      INTEGER
     .N,                  ! Number of input coefficients
     .M,                  ! Number of output points
     .ISPAD(10)           ! Scratchpad for NAG routine

      DO 100 I = N+2, M, 1
         A(I) = 0
 0100 CONTINUE
      DO 200 I = 1, M, 1
         B(I) = 0
 0200 CONTINUE

      IFAIL = 0
      CALL C06ADF(A,B,M,M,M,ISPAD,M,SPAD,M,SPAD2,M,IFAIL)

      DO 400 I = 1, M, 1
         A(I) = (A(I)*A(I) + B(I)*B(I))
         IF (A(I).LT.1E-10) A(I) = 1E-10
         A(I) = 10 * ALOG10( P*M/A(I) )
 0400 CONTINUE

      RETURN
      END


c     ================================================================


c     Graph plotting subroutine, 14.6.82
      SUBROUTINE NUPLO(XPLOT,YPLOT,POINTS)

c     This subroutine plots a graph.  The style of axes and line
c     may be altered by changing the two program lines (see manuals).

      REAL XPLOT(10),YPLOT(10)
      INTEGER POINTS

      CALL GRID(3,1,1)

      CALL GRAPOL(XPLOT,YPLOT,POINTS)

      RETURN
      END
```

```
c     ------------------------------------------------------------------

c     Filter resetting subroutine, 18.8.82
      SUBROUTINE RESET(BO,K,G,MU,MU1,MU2)

c     This subroutine resets coefficients and signal variables in
c     the filter subroutine LFILT.
c     It works as follows:

c     1.  Declarations
c     2.  Reset variables
c     3.  Return

c     ------------------

c     1.  Declarations

      REAL MU(2,64), MU1, MU2
      COMPLEX BO(64), K(64), G(64)
      INTEGER I


c     2.  Reset variables

      DO 200 I = 1, 64, 1
         BO(I) = (0,0)
         K(I) = (0,0)
         G(I) = (0,0)
         MU(1,I) = MU1
         MU(2,I) = MU2
0200  CONTINUE


c     3.  Return

      RETURN
      END


c     ------------------------------------------------------------------


c     Tektronix terminal, 27.8.82
      SUBROUTINE SCREEN(XMIN,YMIN,XMAX,YMAX,XPLOT,YPLOT,POINTS,F)

c     This subroutine plots a graph on the T4014.  It is of
c     correct size for direct use in a thesis.  It adds identifying
c     text:  Date-time + a real number.
c     If necessary, data points are altered to fit the frame.
c     Apart from that, no points are altered.


      REAL XMIN,YMIN,XMAX,YMAX,  ! The numerical limits of the axes
     .     XPLOT(10),YPLOT(10),  ! The positions of the points
     .     F                     ! A real for graph ID solely
      INTEGER POINTS             ! The number of points to be plotted
      INTEGER ISPAD(4),          ! Scratchpad used for date/time
     .        I,J                ! Used to get time


      CALL TIME(I,J)
      ISPAD(3) = I
      ISPAD(4) = J
      CALL DATE(ISPAD)

      DO 200 I = 1, POINTS, 1
         IF (XPLOT(I).GT.XMAX) XPLOT(I)=XMAX
         IF (XPLOT(I).LT.XMIN) XPLOT(I)=XMIN
         IF (YPLOT(I).GT.YMAX) YPLOT(I)=YMAX
         IF (YPLOT(I).LT.YMIN) YPLOT(I)=YMIN
0200  CONTINUE
```

```
      CALL T4014
      CALL WINDOW(2)
      CALL DEVSPE(9600)
      CALL PICCLE
      ACCEPT 101,K
 0101 FORMAT(A2)

      CALL AXIPOS(1,57.,94.,125.,1)
      CALL AXIPOS(1,57.,94.,175.,2)

      CALL AXISCA(1,12,XMIN,XMAX,1)
      CALL AXISCA(1,17,YMIN,YMAX,2)

      CALL NUPLO(XPLOT,YPLOT,POINTS)

      CALL MOVTO2(60.,70.)
      CALL CHAARR(ISPAD,4,5)
      CALL MOVTO2(60.,50.)
      CALL CHAFLO(F,15)

      CALL MOVTO2(1.,2.)
      CALL LINTO2(211.,2.)
      CALL LINTO2(211.,299.)
      CALL LINTO2(1.,299.)
      CALL LINTO2(1.,2.)

      CALL MOVTO2(41.,42.)
      CALL CHAHOL(3H.*.)
      CALL MOVTO2(185.,42.)
      CALL CHAHOL(3H.*.)
      CALL MOVTO2(185.,278.)
      CALL CHAHOL(3H.*.)
      CALL MOVTO2(41.,278.)
      CALL CHAHOL(3H.*.)

      CALL DEVEND

      ACCEPT 101, K
      IF (K.EQ.1Hc) K = 1HC
      IF (K.NE.1HC) GOTO 500
      CALL CALCOM(XMIN,YMIN,XMAX,YMAX,XPLOT,YPLOT,POINTS,
     .            F,ISPAD)

 0500 RETURN
      END


C     ================================================================


C     Lattice filter subroutine, 31.1.83
      SUBROUTINE SPILT(LINE,PURE,F,BN,BO,V,K,G,MU,LENGTH,
     .                 ALPHA,BETA,ITNO)


C     This subroutine acts as a digital lattice equaliser
C     with adaptive stepsize.
C     The line signal and training signal are input and are unaltered
C     on return.  The output is the error signal, V(LENGTH+1).
C     The program works as follows:

C     1.  Declarations
C     2.  Data input
C        3.  Calculate lattice values
C        4.  Calculate equaliser values
C        5.  Adapt lattice coefficients
C        6.  Adapt equaliser coefficients
C        7.  Adapt stepsizes
C        8.  Clock delays
C     10. Return

C     ----------------------------------
```

```
c     1.  Declarations

      REAL
     .ALPHA,              ! Stepsize time constant
     .BETA,               ! Stepsize scaling constant
     .MU(2,64)            ! Equaliser stepsizes

      COMPLEX
     .LINE,               ! Filter input
     .PURE,               ! Training input
     .F(64),              ! Filter signal values
     .BN(64),
     .BO(64),
     .V(64),
     .K(64),              ! Filter coefficients
     .G(64)

      INTEGER
     .LENGTH,             ! Filter length
     .ITNO,               ! Iteration number
     .N                   ! Loop counter


c     2.  Data input

      F(1) = LINE
      BN(1) = LINE
      V(1) = PURE


c        3.  Calculate lattice values

      DO 300 N = 1, LENGTH, 1
         F(N+1) = F(N) - K(N) * BO(N)
         BN(N+1) = BO(N) - CONJG(K(N)) * F(N)


c        4.  Calculate equaliser values

         V(N+1) = V(N) - G(N) * BN(N)
 0300 CONTINUE


c        5.  Adapt lattice coefficients

      DO 500 N = 1, LENGTH, 1

         K(N) = K(N) + MU(1,N) * (F(N) * CONJG(BN(N+1))
     .                       + F(N+1) * CONJG(BO(N)))

         IF (CABS(K(N)).GT.1) K(N) = K(N) / CABS(K(N))

 0550    CONTINUE


c        6.  Adapt equaliser coefficients

         G(N) = G(N) + V(N+1) * CONJG(BN(N)) * MU(2,N)


c        7.  Adapt stepsizes

         MU(1,N) = 1/ ((1/MU(1,N))*(1-ALPHA)
     .              + (BETA*ALPHA) * REAL(BN(N)*CONJG(BN(N))
     .                             + F(N)*CONJG(F(N))))
         MU(2,N) = 2 * MU(1,N)


c        8.  Clock delays

         BO(N) = BN(N)
 0500 CONTINUE
```

```
c     9. Return

      RETURN
      END



c     ================================================================


c     Surface plotter, 30.6.82
      SUBROUTINE SURF(XMIN,YMIN,XMAX,YMAX,ZPLOT,XDIM,YDIM,ZHT,
     .                ASPECT,ISPAD,SPAD)


c     This subroutine plots a surface on the T4014 and adds the
c     time of plotting as a serial number.  If the plot is
c     acceptable, it then repeats it on the Calcomp.  XDIM
c     must be greater than YDIM.

      REAL
     .XMIN,YMIN,XMAX,YMAX,       ! Numerical limits of axes
     .ZPLOT,                     ! 2-D array of Z-values
     .ZHT,                       ! Z-axis height ratio
     .SPAD(10)                   ! Scratchpad

      INTEGER
     .K,                         ! Switch variable
     .ASPECT,                    ! Nearest corner
     .I,J,                       ! Used to get time
     .ISPAD(10),                 ! Used for date-time
     .XDIM,YDIM,                 ! Points along axes
     .NW                         ! A dimension used by SPAD


      DIMENSION ZPLOT(XDIM,YDIM)


      NW = XDIM * (YDIM + 9)

      CALL TIME(I,J)
      CALL DATE(ISPAD)
      ISPAD(3) = I
      ISPAD(4) = J

         CALL T4014
         CALL WINDOW(2)
         CALL PICCLE

         ACCEPT 101, K
0101     FORMAT(A2)

         CALL HEIRAT(ZHT)
         CALL ISOPRJ(XDIM,XMIN,XMAX,YDIM,YMIN,YMAX,ZPLOT,
     .               ASPECT,NW,SPAD)
         CALL MOVTO2(10., 10.)
         CALL CHAARR(ISPAD,4,5)

         CALL DEVEND

      ACCEPT 101, K
      IF (K.EQ.'c') K = 'C'
      IF (K.NE.'C') GOTO 100

         CALL CC1051
         CALL WINDOW(2)
         CALL PENSEL(2, 0.2, 3)

         CALL HEIRAT(ZHT)
         CALL ISOPRJ(XDIM,XMIN,XMAX,YDIM,YMIN,YMAX,ZPLOT,
     .               ASPECT,NW,SPAD)
         CALL MOVTO2(10., 10.)
         CALL CHAARR(ISPAD,4,5)

         CALL DEVEND

0100  RETURN
      END
```

```
c     ------------------------------------------------------------

c     Transversal filter subroutine, 24.8.82
      SUBROUTINE TFILT(LINE,PURE,F,BN,BO,V,K,G,MU,LENGTH)


c     This subroutine acts as a digital transversal equaliser.
c     The line signal and training signal are input and are unaltered
c     on return.  The output is the equalised signal.
c     The program works as follows:

c     1.  Declarations
c     2.  Data input
c     3.  Calculate filter output
c     4.  Subtract from training signal
c     5.  Update the tapweights
c     6.  Clock the delays
c     7.  Return

c     ------------------------------------

c     1.  Declarations

      REAL
     .MU(2,64)          ! Filter stepsize

      COMPLEX
     .LINE,             ! Distorted input
     .PURE,             ! Training input
     .OUTPUT,           ! Equalised output
     .F(64),            ! Filter signal variables
     .BN(64),
     .BO(64),
     .V(64),
     .K(64),            ! Filter coefficients
     .G(64)

      INTEGER
     .LENGTH,           ! Filter length
     .N                 ! Loop counter


c     2.  Data input

      BO(1) = LINE


c     3.  Calculate filter output

      OUTPUT = 0
      DO 300 N = 1, LENGTH, 1
         OUTPUT = OUTPUT + G(N) * BO(N)
 0300 CONTINUE


c     4.  Subtract from the training signal

      V(LENGTH+1) = PURE - OUTPUT


c     5.  Update the tapweights

      DO 500 N = 1, LENGTH, 1
         G(N) = G(N) + MU(2,N) * V(LENGTH+1) * CONJG(BO(N))
 0500 CONTINUE


c     6.  Clock the delays

      DO 600 N = 1, (LENGTH-1), 1
         BO(LENGTH - N + 1) = BO(LENGTH - N)
 0600 CONTINUE
```

```
c     7.  Return

      RETURN
      END


c     ------------------------------------------------------------


c     Convolution subroutine, 11.8.82
      SUBROUTINE TFILTX(K,G,IMPULS,CONVOL,LENGTH)


c     This subroutine convolves the input impulse response with the
c     filter.
c     It works as follows:

c     1.  Declarations
c     2.  Initialisation
c     3.  Feed in impulse response
c     4.  Run right through filter
c     5.  Return

c     ---------------------------

c     1.  Declarations

      COMPLEX
     .K(64),              ! Filter coefficients
     .G(64),
     .XBO(64),            ! Filter signal values
     .XBN(64),
     .XF(64),
     .XV(64),
     .CONVOL(64),         ! Convolved output array
     .IMPULS(64)          ! Line impulse response

      INTEGER
     .LENGTH,             ! Filter length
     .I                   ! Loop counter


c     2.  Initialisation

      DO 0200 I = 1, 64, 1
         XBO(I) = (0,0)
         XBN(I) = (0,0)
         XF(I)  = (0,0)
         XV(I)  = (0,0)
         CONVOL(I) = (0,0)
 0200 CONTINUE


c     3.  Feed in impulse response

      DO 0300 I = 1, 32, 1
         CALL TFILT(IMPULS(I),(0,0),CONVOL(I),
     .              XF,XBN,XBO,XV,K,G,0,0,LENGTH)
 0300 CONTINUE


c     4.  Run right through filter

      DO 0400 I = 33, 64, 1
         CALL TFILT((0,0),(0,0),CONVOL(I),
     .              XF,XBN,XBO,XV,K,G,0,0,LENGTH)
 0400 CONTINUE


c     5.  Return

      RETURN
      END
```

```
      Tapweight calculator 29.7.82
      SUBROUTINE WIENER(IMPUL1,LEN1,IMPUL2,RANK)

C     This program calculates the Wiener solution of the inverse
C     of a given impulse response.
C     The program works as follows:

C     1.    Declarations
C     2.    Initialisation
C     3.    conjugate and centre input data
C     4.    Generate autocorrelation function
C     5.    Generate autocorrelation matrix
C     6.    Call inversion subroutine
C     7.    End

C     1.    Declarations

      REAL
     .SPAD1(32)          ! Scratchpads for NAG routine

      INTEGER
     .RANK,              ! Rank of autocorrelation matrix
     .LEN1,              ! Number of input data points
     .IFAIL,             ! Error code for NAG routine
     .I,
     .J,
     .K

      COMPLEX
     .A(32,32),          ! Autocorrelation matrix
     .IMPUL1(32),        ! Line impulse response
     .IMPUL2(32),        ! Inverse impulse response
     .ACF(64),           ! Autocorrelation array
     .B(64)              ! Cross-correlation array


C     2.    Initialisation

      DO 0200 I = 1, 64, 1
         ACF(I) = (0,0)
         B(I) = (0,0)
0200  CONTINUE


C     3.    conjugate and centre input data

      K = (LEN1 + RANK + 3) / 2
      DO 0350 I = 1, RANK, 1
         IF ((K-I).LT.1) GOTO 350
         B(I) = CONJG(IMPUL1(K-I))
0350  CONTINUE


C     4.    Generate autocorrelation function

      DO 0400 I = 1, LEN1, 1
         DO 0450 J = 1, LEN1, 1
            K = I - J + 32
            ACF(K) = ACF(K) + IMPUL1(I) * CONJG(IMPUL1(J))
0450     CONTINUE
0400  CONTINUE
```

```
c     5.   Generate autocorrelation matrix

      DO 0500 I = 1, RANK, 1
         DO 0550 J = 1, RANK, 1
            K = I - J + 32
            A(I,J) = ACF(K)
 0550    CONTINUE
 0500 CONTINUE


c     6.   Call inversion subroutine

      IFAIL = 0
      CALL F04ADF(A,32,B,32,RANK,1,IMPUL2,32,SPAD1,IFAIL)


c     7.   End

      RETURN
      END


c     ------------------------------------------------------------------
```

APPENDIX C; Integrated Circuit Designs

Chapter seven described an adaptive lattice structure implemented as a set of five integrated circuits. This appendix contains the full set of designs for the chip set. Only the mask designs are missing, as they are not suitable for A4 reproduction. The contents are as follows:

CONTENTS:

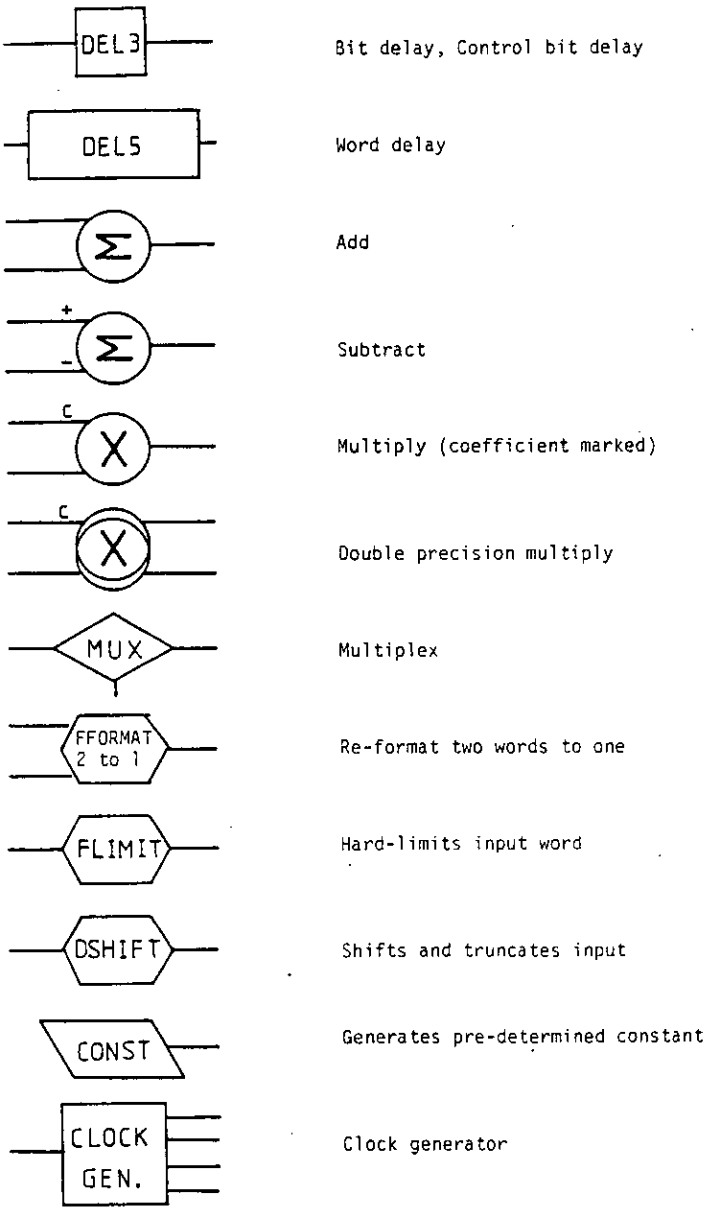


Fig C1 Primitives used in operators

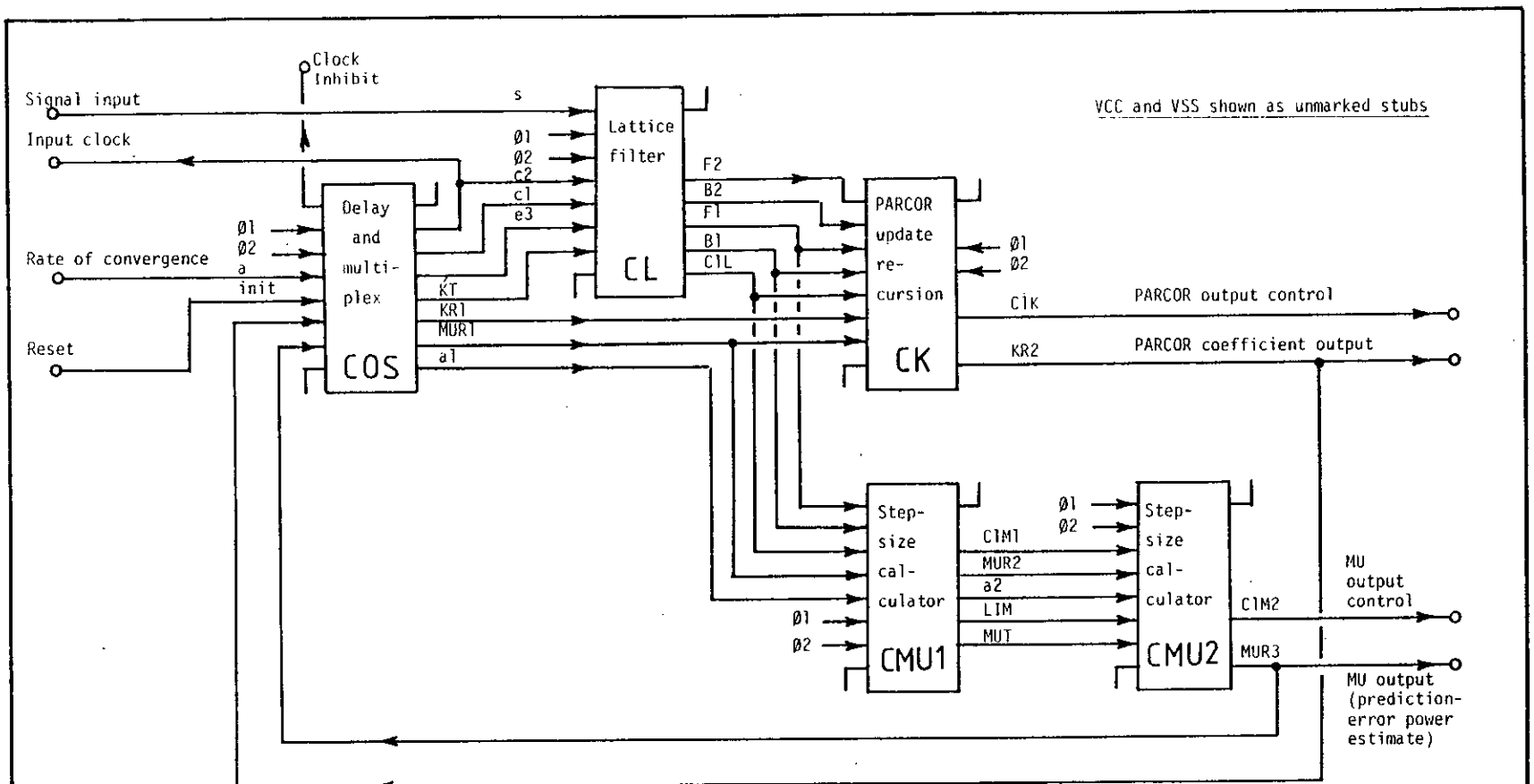


Fig C2 Lattice PE system, chip interconnections

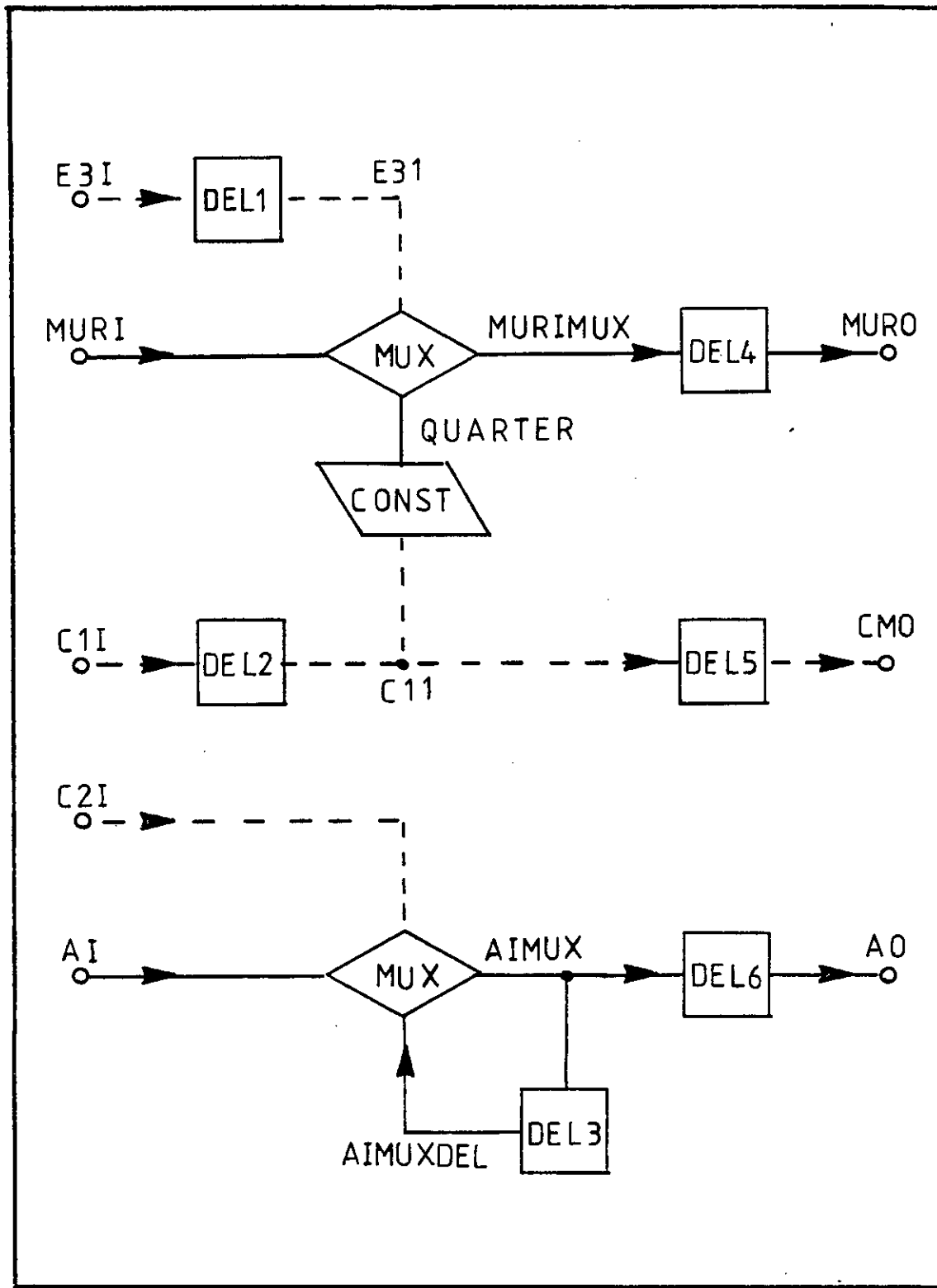


Fig C3 MUINIT operator

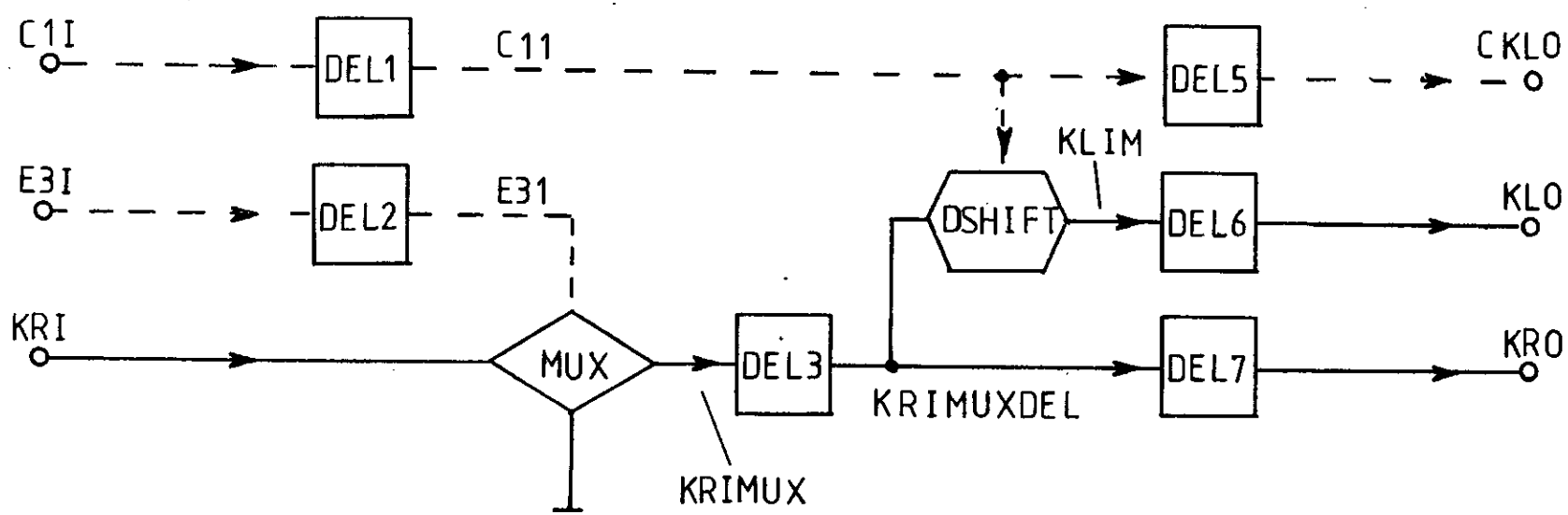


Fig C4 KINIT operator

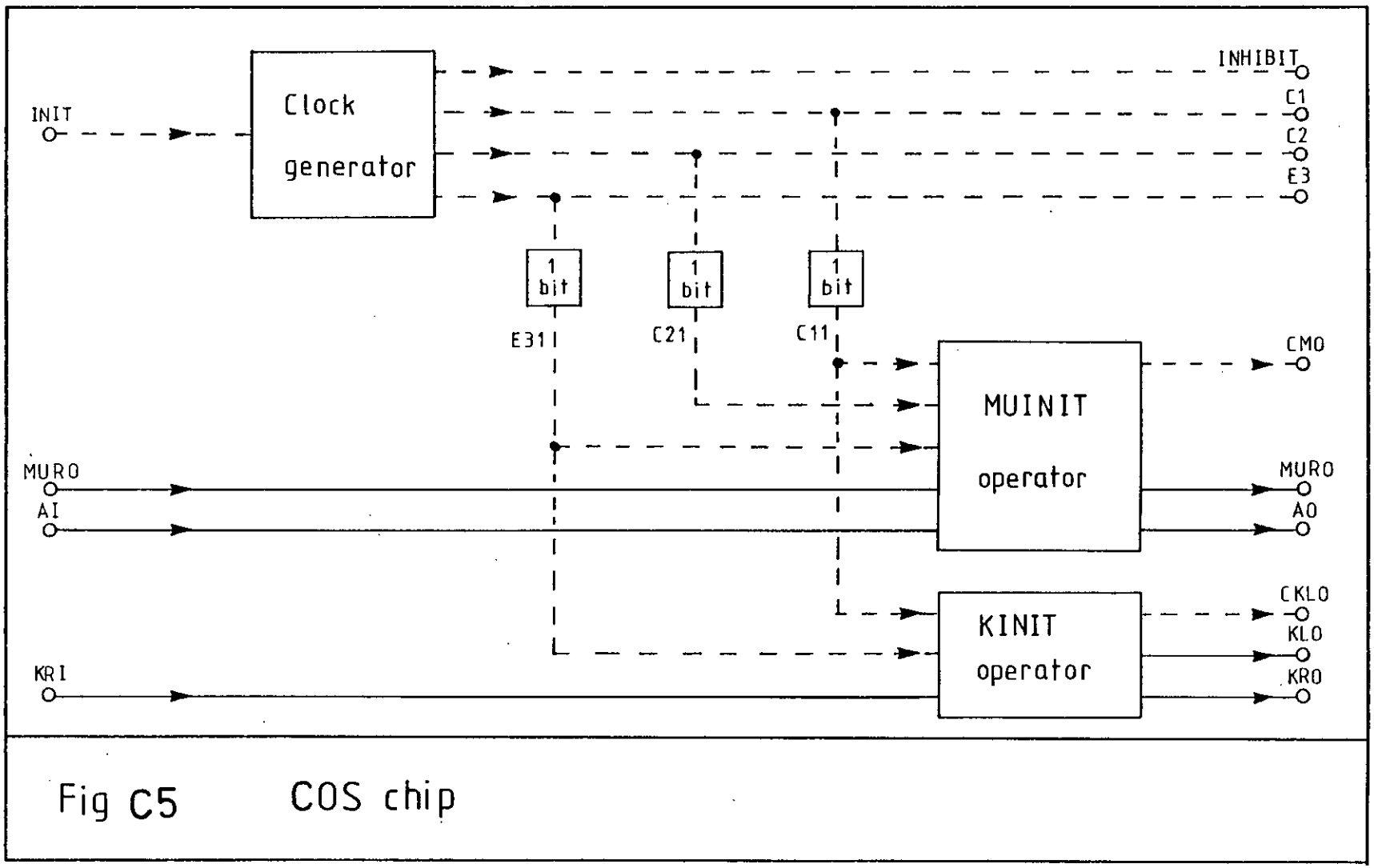


Fig C5 COS chip

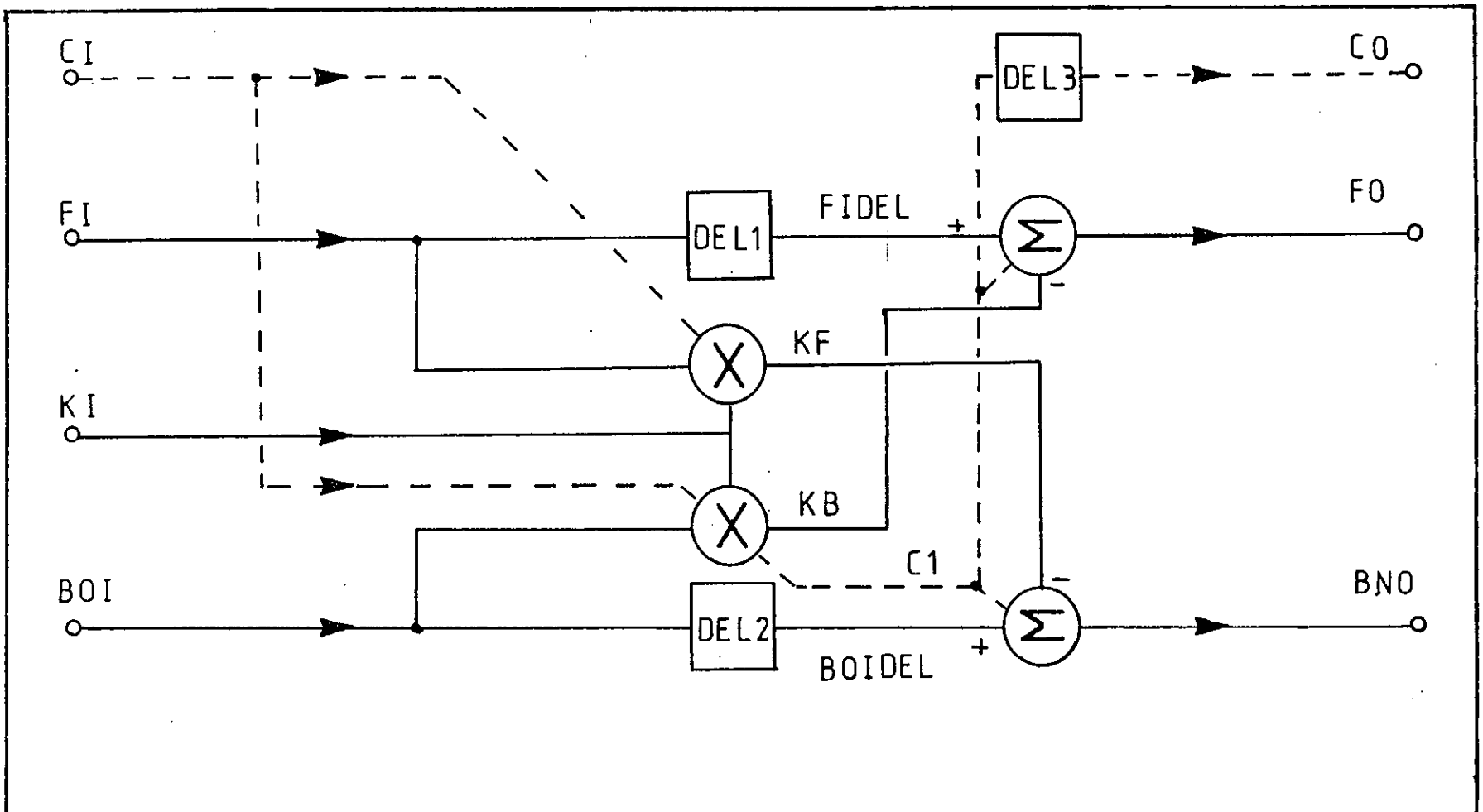


Fig C6 Lattice operator flowchart

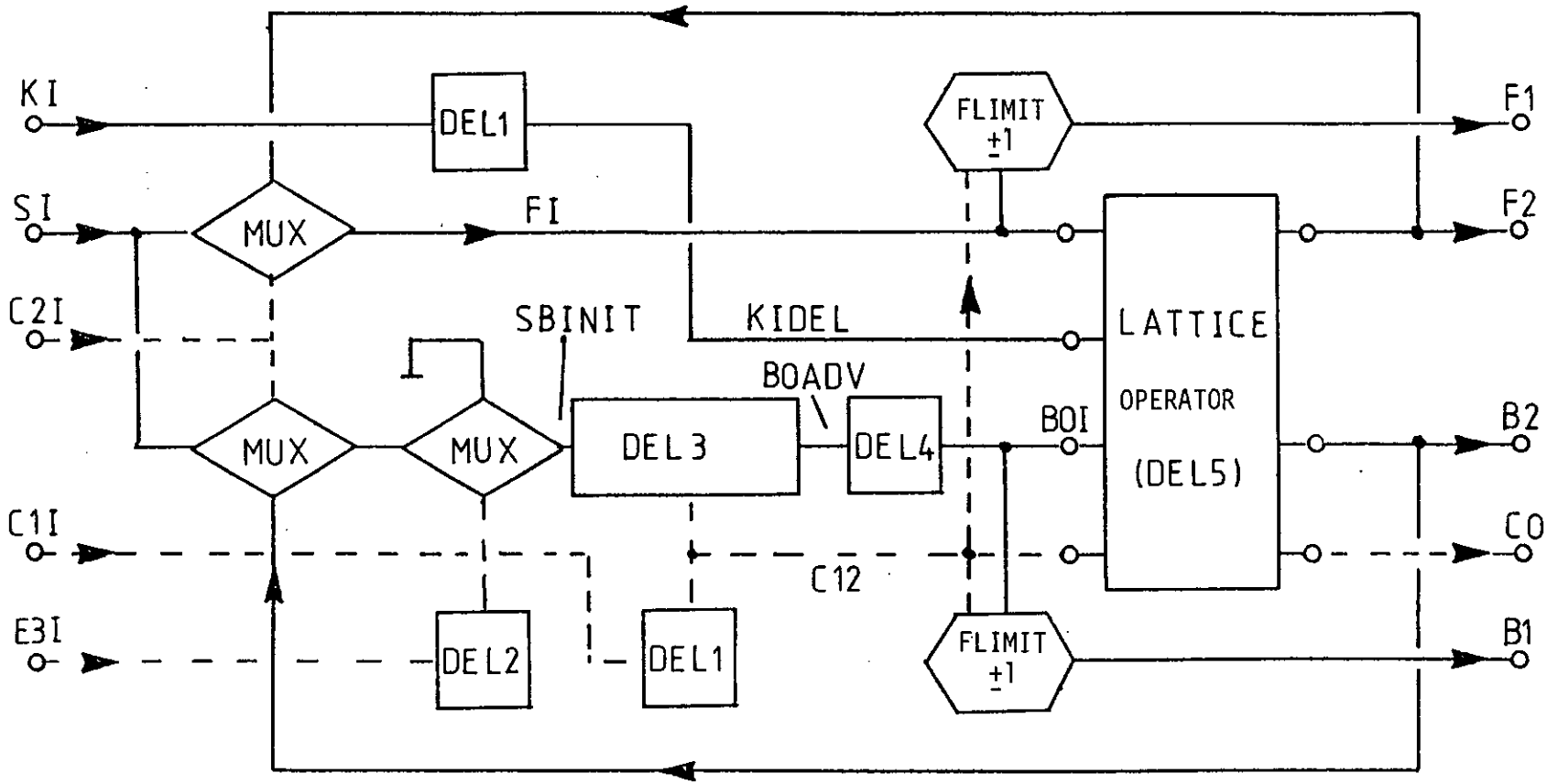


Fig C7 Lattice chip flowchart

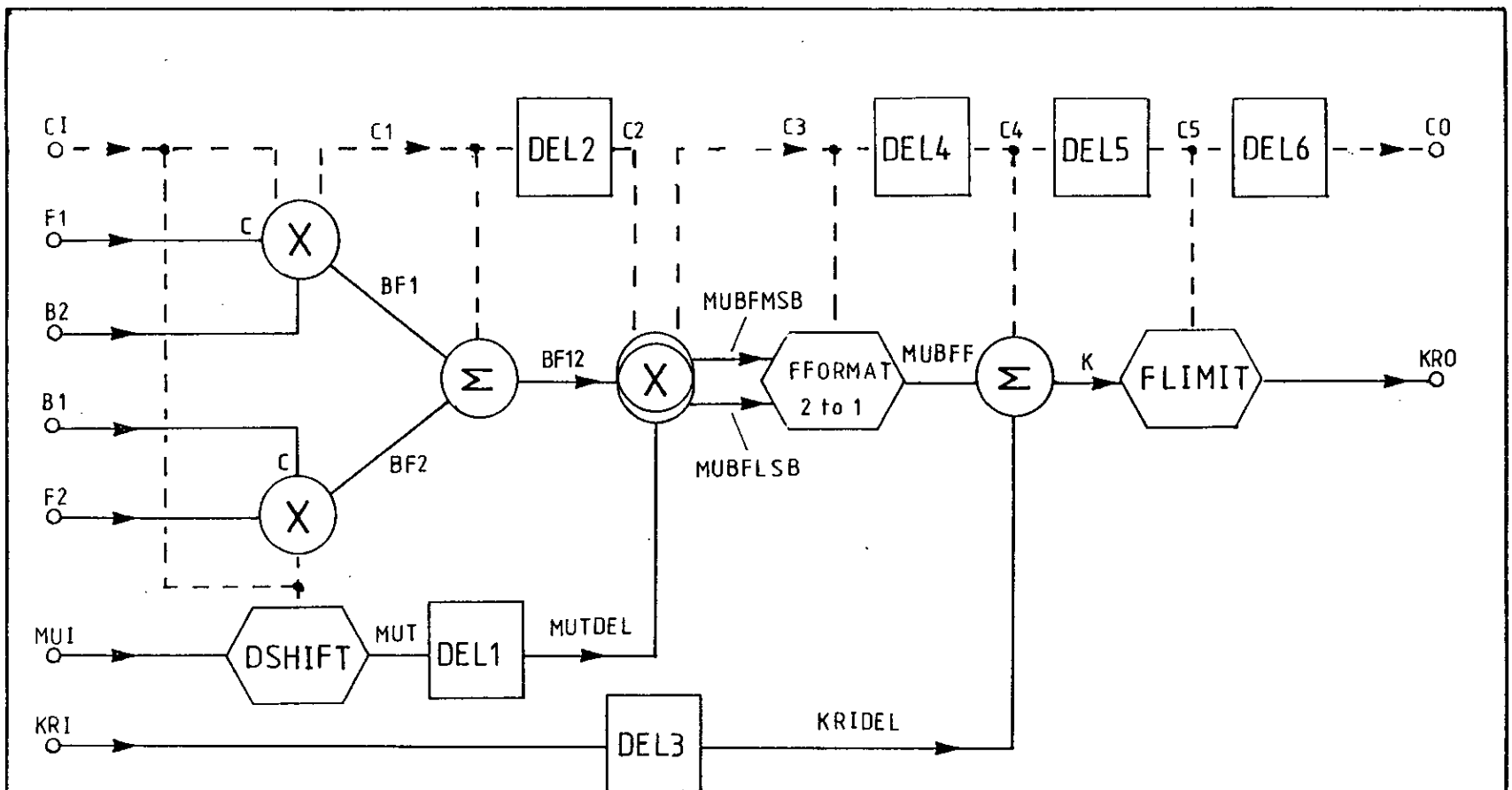


Fig C 8 K operator and chip

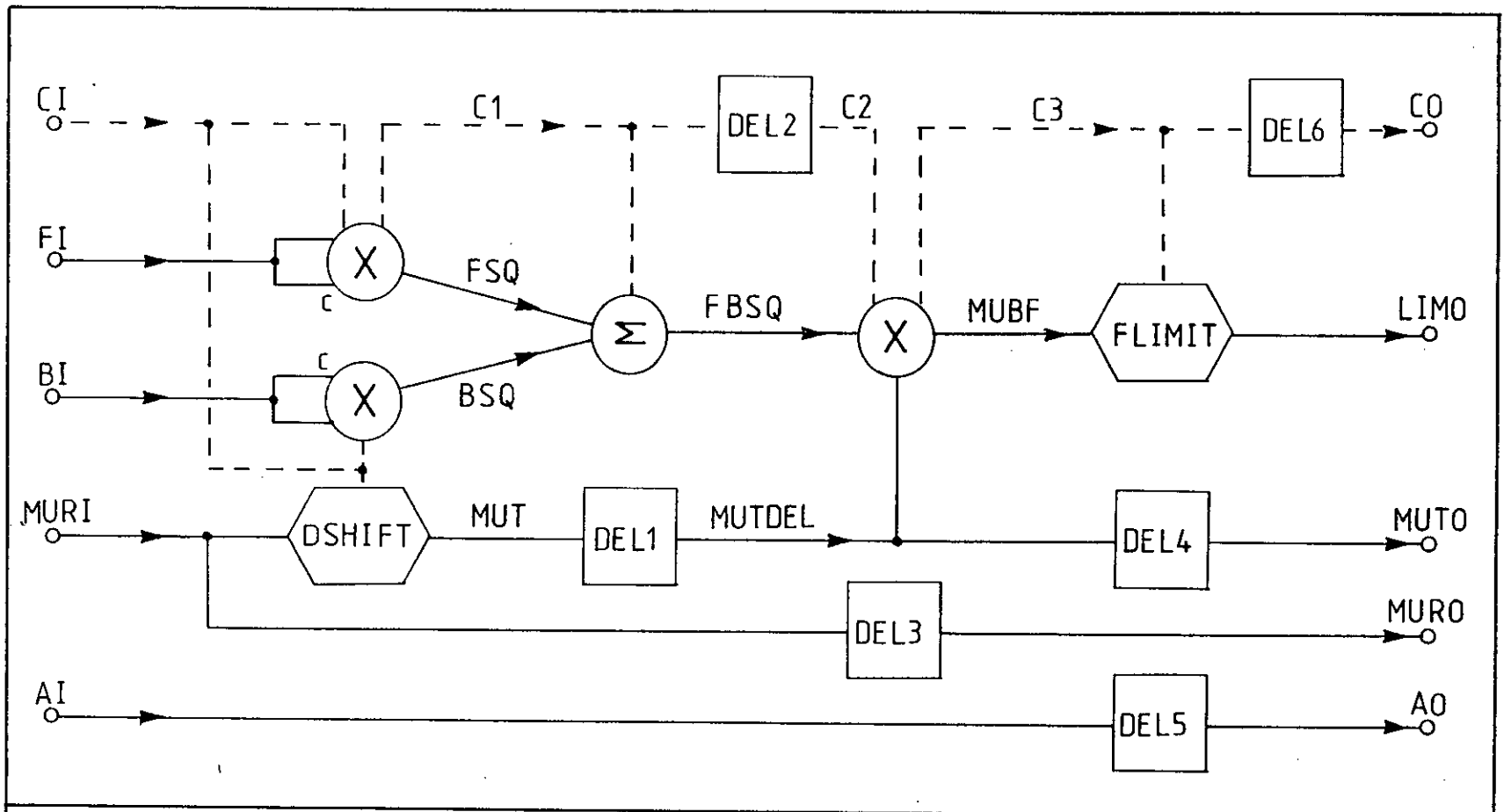


Fig C9 MU1 operator and chip

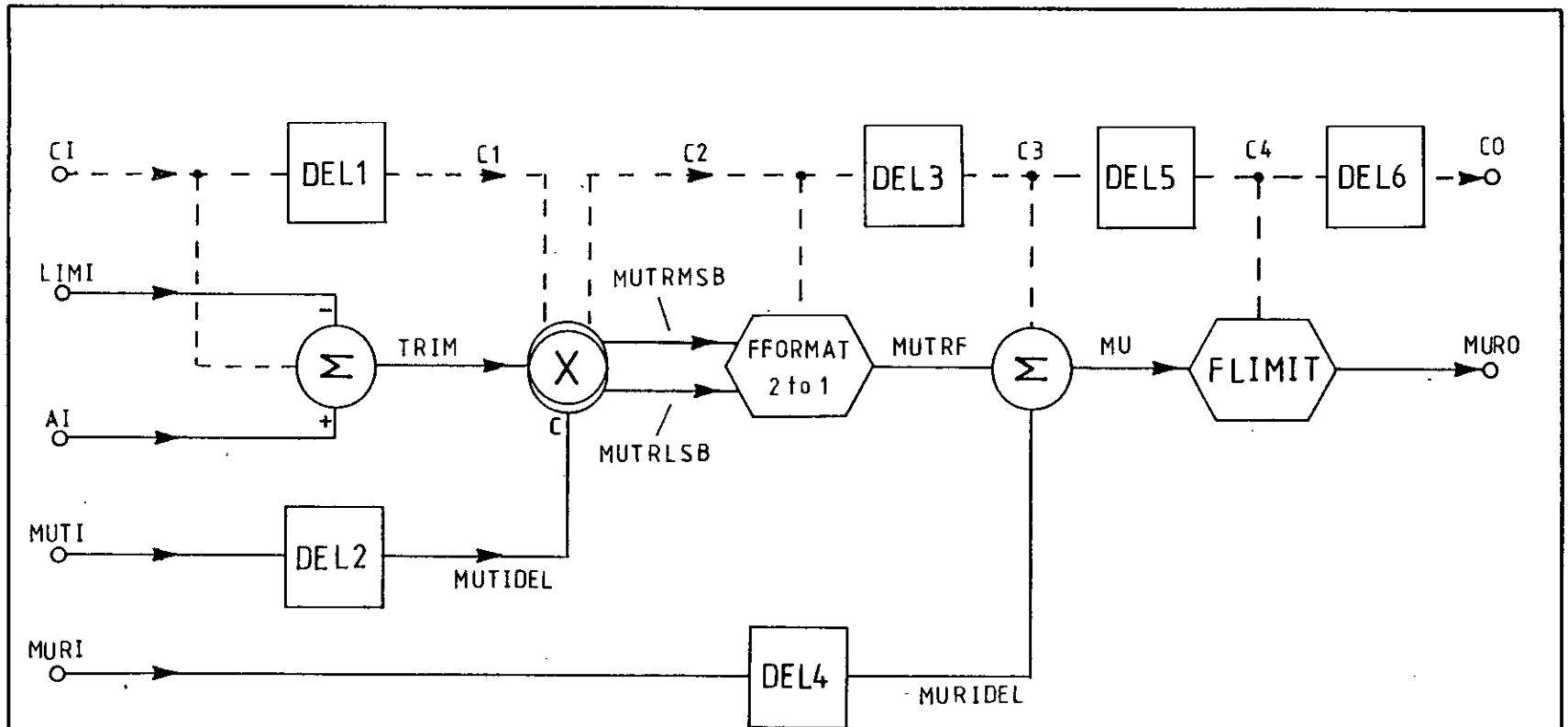


Fig C10 MU2 operator and chip

```
! SPE.FIR  M.J.Rutter 12.7.83
!#############################################################################

!      ADAPTIVE LATTICE STRUCTURE CHIP SET

!  This program uses the silicon compiler to generate a set of 5
!  integrated circuits.  Each IC contains subroutine-like operators
!  which are functional subunits.  The operators are listed before
!  the chip itself.
!  The modules are arranged as follows:

!        OPERATOR MUINIT
!        OPERATOR KINIT
!     CHIP COS

!        OPERATOR LATTICE
!     CHIP CL

!        OPERATOR K
!     CHIP CK

!        OPERATOR MU1
      CHIP CMU1

!        OPERATOR MU2
!     CHIP CMU2

!  SYSTEM PE


!#############################################################################


OPERATOR MUINIT [wordll] (cli,c2i,e3i->cmo) -
         muri, ai -> muro, ao

! code
! designer    : mjr
! last update : 9.5.83
! status      : checked, simulated
! description : Mu word delays and initialisation
! delay

   SIGNAL murimux, murimuxdel, -
          quarter, aimux, aimuxdel, del5br1, del5br2, del5br3, -
          del5br4, del5br5, del5br6, del5br7, del5br8, del5br9

   CONTROL cll, e3l, del2br1, del2br2, del2br3, del2br4, -
           del2br5, del2br6

   CONSTANT dell = 2
   CONSTANT del2a = 6
   CONSTANT del2b = 20
   CONSTANT del4 = 19
   CONSTANT del5 = 10
   CONSTANT del7 = wordll - 1
   CONSTANT del9 = 7
   CONSTANT dell0 = 1

   CBITDELAY [dell] (cli->cmo)
!  CBITDELAY [del2] (e3i->e3l)
      CBITDELAY [wordll] (e3i->del2br1)
      CBITDELAY [wordll] (del2br1->del2br2)
      CBITDELAY [wordll] (del2br2->del2br3)
      CBITDELAY [wordll] (del2br3->del2br4)
      CBITDELAY [wordll] (del2br4->del2br5)
      CBITDELAY [wordll] (del2br5->del2br6)
      CBITDELAY [del2b] (del2br6->e3l)

   CBITDELAY [del4] (cli->cll)
   CONSTGEN [wordll, 102400] (cll) -> quarter
```

```
   MULTIPLEX [1,0,0] (e31) muri, quarter -> murimux
!  WORDDELAY [del5,wordll,0] (cll) murimux -> murimuxdel
      BITDELAY [wordll] murimux -> del5br1
      BITDELAY [wordll] del5br1 -> del5br2
      BITDELAY [wordll] del5br2 -> del5br3
      BITDELAY [wordll] del5br3 -> del5br4
      BITDELAY [wordll] del5br4 -> del5br5
      BITDELAY [wordll] del5br5 -> del5br6
      BITDELAY [wordll] del5br6 -> del5br7
      BITDELAY [wordll] del5br7 -> del5br8
      BITDELAY [wordll] del5br8 -> del5br9
      BITDELAY [wordll] del5br9 -> murimuxdel
   BITDELAY [del9] murimuxdel -> muro

   MULTIPLEX [1,0,0] (c2i) aimuxdel, ai -> aimux
   BITDELAY [del7] aimux -> aimuxdel
   BITDELAY [del10] aimux -> ao

END

!=================================================================

OPERATOR KINIT [wordll,shiftl] (cli,e3i->cklo) kri -> kro, klo

! code
! designer    : mjr
! last update : 9.5.83
! status      : checked, simulated
! description : K word delays and initialisation
! delay

   SIGNAL krimux, krimuxdel, del3br1, del3br2, del3br3, del3br4, ~
          del3br5, del3br6, del3br7, del3br8, del3br9, del3br10, del7br

   CONTROL cll, e31, del2br1, del2br2, del2br3, del2br4, del2br5

   CONSTANT dell = 18
   CONSTANT del2w = 5
   CONSTANT del2b = 3
   CONSTANT del3 = 10
   CONSTANT del4 = 14
   CONSTANT del5 = 7
   CONSTANT del7 = 36
      CONSTANT del7a = del7 / 2
      CONSTANT del7b = del7 - del7a


   CBITDELAY [dell] (cli->cll)
!  CBITDELAY [del2] (e3i->e31)
      CBITDELAY [wordll] (e3i->del2br1)
      CBITDELAY [wordll] (del2br1->del2br2)
      CBITDELAY [wordll] (del2br2->del2br3)
      CBITDELAY [wordll] (del2br3->del2br4)
      CBITDELAY [wordll] (del2br4->del2br5)
      CBITDELAY [del2b] (del2br5->e31)

   MULTIPLEX [1,0,0] (e31) kri, GND -> krimux

!  WORDDELAY [del3,wordll,0] (cll) krimux -> krimuxdel
      BITDELAY [wordll] krimux -> del3br1
      BITDELAY [wordll] del3br1 -> del3br2
      BITDELAY [wordll] del3br2 -> del3br3
      BITDELAY [wordll] del3br3 -> del3br4
      BITDELAY [wordll] del3br4 -> del3br5
      BITDELAY [wordll] del3br5 -> del3br6
      BITDELAY [wordll] del3br6 -> del3br7
      BITDELAY [wordll] del3br7 -> del3br8
      BITDELAY [wordll] del3br8 -> del3br9
      BITDELAY [wordll] del3br9 -> del3br10
      BITDELAY [del4] del3br10 -> krimuxdel
   DSHIFT [shiftl,0] (cll) krimuxdel -> klo

   CBITDELAY [del5] (cll->cklo)
!  BITDELAY [del7] krimuxdel -> kro
      BITDELAY [del7a] krimuxdel -> del7br
      BITDELAY [del7b] del7br -> kro

END

!=================================================================
```

```
CHIP COS (pinit->pc1,pc2,pe3,pcmo,pcklo,pinhibit) -
     pmuri, pai, pkri -> pmuro, pao, pklo, pkro

!! code
!! designer    : mjr
!! last update : 6.5.83
!! status      : checked, simulated
!! description : clock generator, muinit & kinit
!! delay

   SIGNAL muri, ai, kri, muro, ao, klo, kro
   CONTROL init, c1, c2, e3, cmo, cklo, c11, c21, e31,inhibit

   PADORDER VDD, pinit, GND, pmuri, pai, pkri, -
            CLOCK, pc1, pc2, pe3, pcmo, pcklo, pinhibit, pmuro, pao, pklo, -
            pkro

!     PADIN (pinit->init)
      PADIN (pinit->e31)
!&&&&&
      PADIN pmuri -> muri
      PADIN pai -> ai
      PADIN pkri -> kri

      PADOUT (c1->pc1)
      PADOUT (c2->pc2)
      PADOUT (E3->pe3)
      PADOUT (cmo->pcmo)
      PADOUT (cklo->pcklo)
      PADOUT (inhibit->pinhibit)
      PADOUT muro -> pmuro
      PADOUT ao -> pao
      PADOUT klo -> pklo
      PADOUT kro -> pkro

   CONSTANT word11 = 26
   CONSTANT word12 = 26
   CONSTANT word13 = 26
   CONSTANT shift1 = 4
   CONSTANT stages = 16


!  CONTROLGENERATOR (init->inhibit,c1,c2,e3)
   CONTROLGENERATOR (gnd->inhibit,c1,c2,e3)
!&&&&
      CYCLE [word11]
      CYCLE [stages]
      EVENT
   ENDCONTROLGENERATOR


   CBITDELAY [1] (c1->c11)
   CBITDELAY [1] (c2->c21)
!  CBITDELAY [1] (e3->e31)
!&&&&&

   MUINIT [word12] (c11,c21,e31->cmo) -
          muri, ai -> muro, ao

   KINIT [word13,shift1] (c11,e31->cklo) kri -> kro, klo

END

!=================================================================================

OPERATOR LATTICE [coeff1] (ci -> co) fi, ki, boi -> fo, bno

! code
! designer    :: MJR
! last update :: 26.4.83
! status      :: checked , simulated.
! description :: Non-adaptive lattice structure
! delay       :: del1 + del3 =24

   SIGNAL fidel, kf, boidel, kb
   CONTROL c1

   CONSTANT del1 = ((3 * coeff1)/2) + 2
   CONSTANT del2 = del1
   CONSTANT del3 = 1
```

```
    BITDELAY [del1] fi -> fidel
    MULTIPLY [2,coeff1,0,0] (ci->NC) fi, ki -> kf, NC
    MULTIPLY [2,coeff1,0,0] (ci->c1) boi, ki -> kb, NC
    BITDELAY [del2] boi -> boidel

    CBITDELAY [del3] (c1->co)
    SUBTRACT [1,0,0,0] (c1) fidel, kb, GND -> fo, NC
    SUBTRACT [1,0,0,0] (c1) boidel, kf, GND -> bno, NC

END

!!=================================================================

CHIP CL (pc1i,pc2i,pe3i->pco) psi, pki -> pf1, pf2, pb1, pb2

!! code
!! designer    :: MJR
!! last update :: 27.5.83
!! status      :: checked, simulated
!! description :: lattice chip
!! delay       :: (swl+1) + del1 + 1 = 29

  SIGNAL ki, si, sb, sbinit, boi, fi, f2, b2, f1, b1, -
         kidel, del3br1, del3br2, del3br3, del3br4, del3br5, del3br6, -
         del3br7, del3br8, del3br9, del3br10, del3br11, -
         del3br12, del3br13, del3br14 , boadv, f2adv, b2adv

  CONTROL c1i, c2i, e3i, e31, c11, c12, co

  CONSTANT coeff1 = 14
  CONSTANT word11 = 26
  CONSTANT word12 = 12
  CONSTANT stages = 16
  CONSTANT del1 = 1
  CONSTANT del2 = 1
  CONSTANT del3 = stages - 1
  CONSTANT del5 = 3

  PADORDER VDD, pc1i, pc2i, pe3i, GND, psi, pki, CLOCK, -
           pco, pf1, pf2, pb1, pb2

    PADIN (pc1i -> c1i)
    PADIN (pc2i -> c2i)
    PADIN (pe3i -> e3i)
    PADIN psi -> si
    PADIN pki -> ki

    PADOUT (co -> pco)
    PADOUT f1 -> pf1
    PADOUT f2 -> pf2
    PADOUT b1 -> pb1
    PADOUT b2 -> pb2

 MULTIPLEX [1,1,0] (c2i) f2adv, si -> fi
 MULTIPLEX [1,1,0] (c2i) b2adv, si -> sb

 BITDELAY [del1] ki -> kidel
 MULTIPLEX [1,0,0] (e31) sb, GND -> sbinit
 CBITDELAY [del2] (e3i->e31)

!  WORDDELAY [del3,word11,0] (c11) sbinit -> boadv
     BITDELAY [word11] sbinit -> del3br1
     BITDELAY [word11] del3br1 -> del3br2
     BITDELAY [word11] del3br2 -> del3br3
     BITDELAY [word11] del3br3 -> del3br4
     BITDELAY [word11] del3br4 -> del3br5
     BITDELAY [word11] del3br5 -> del3br6
     BITDELAY [word11] del3br6 -> del3br7
     BITDELAY [word11] del3br7 -> del3br8
     BITDELAY [word11] del3br8 -> del3br9
     BITDELAY [word11] del3br9 -> del3br10
     BITDELAY [word11] del3br10 -> del3br11
     BITDELAY [word11] del3br11 -> del3br12
     BITDELAY [word11] del3br12 -> del3br13
     BITDELAY [word11] del3br13 -> del3br14
     BITDELAY [word11] del3br14 -> boadv
 BITDELAY [word11-1] boadv -> boi
 CBITDELAY [del2] (c1i->c11)
```

```
    FLIMIT [word11,word12,0] (c11) f1 -> f1
    FLIMIT [word11,word12,0] (c11) b0i -> b1

    LATTICE [coeff1] (c11->c12) f1, k1del, b0i -> f2adv, b2adv
    BITDELAY [del5] f2adv -> f2
    BITDELAY [del5] b2adv -> b2
    CBITDELAY [del5] (c12->co)

END

!=================================================================================

OPERATOR K [coeff1,coeff2,word11,word12,shift1] (ci->co) -
           kri, mui, b1i, f1i, b2i, f2i -> kro

! code
! designer    :: MJR
! last update :: 6.5.83
! status      :: checked , simulated.
! description :: K update recursion with range limiting
! delay       :: del3 + del5 + del6 =103

   SIGNAL mutdel, mut, kridel, bf1, bf12, k, bf2, mubf1sb, mubfmsb, -
          mubff, del3br1, del3br2

   CONTROL c1, c2, c3, c4, c5, del4br

   CONSTANT del1 = ((3*coeff1)/2) +2 +1 -(shift1+3)
   CONSTANT del2 = 1
   CONSTANT del3 = ((3*coeff1)/2) +3 +1 +((2*word12) +1)
      CONSTANT del3a = del3 / 3
      CONSTANT del3b = del3 - (2*del3a)
   CONSTANT del4 = ((2*word12)+1)
      CONSTANT del4a = word12
      CONSTANT del4b = word12 + 1
   CONSTANT del5 = 1
   CONSTANT del6 = word12 + 1


   MULTIPLY [2,coeff1,0,0] (ci->NC) b2i, f1i -> bf1, NC
   MULTIPLY [2,coeff1,0,0] (ci->c1) f2i, b1i -> bf2, NC
   DSHIFT [shift1,0] (ci) mui -> mut

   BITDELAY [del1] mut -> mutdel
   ADD [1,0,0,0] (c1) bf1, bf2, GND -> bf12, NC

   CBITDELAY [del2] (c1->c2)
   DPMULTIPLY [coeff2,0,0,0] (c2->c3) bf12, mutdel, GND -> mubf1sb, mubfmsb, NC
!  BITDELAY [del3] kri -> kridel
      BITDELAY [del3a] kri -> del3br1
      BITDELAY [del3a] del3br1 -> del3br2
      BITDELAY [del3b] del3br2 -> kridel

   FFORMAT2TO1 [word12,44,19,0,0] (c3) mubf1sb, mubfmsb -> mubff

!  CBITDELAY [del4] (c3->c4)
      CBITDELAY [del4a] (c3->del4br)
      CBITDELAY [del4b] (del4br->c4)
   ADD [1,0,0,0] (c4) mubff, kridel, GND -> k, NC

   CBITDELAY [del5] (c4->c5)
   FLIMIT [word12,17,0] (c5) k -> kro

   CBITDELAY [del6] (c5->co)
END

!=================================================================================
```

```
CHIP CK (pci->pco) pb1i, pb2i, pf1i, pf2i, pmui, pkri -> pkro

!! code
!! designer     : mjr
!! last update  : 26.4.83
!! status       : checked, simulated
!! description  : K chip
!! delay        : K + 1 =104

   SIGNAL b1i, b2i, f1i, f2i, mui, kro, kri
   CONTROL ci, co

   CONSTANT coeff1 = 12
   CONSTANT coeff2 = 8
   CONSTANT word11 = 17
   CONSTANT word12 = 26
   CONSTANT shift1 = 10

   PADORDER VDD, pci, GND, pmui, pkri, CLOCK, pb1i, pf1i, pb2i, pf2i, -
            pco, pkro

      PADIN (pci->ci)
      PADIN pb1i -> b1i
      PADIN pb2i -> b2i
      PADIN pf1i -> f1i
      PADIN pf2i -> f2i
      PADIN pmui -> mui
      PADIN pkri -> kri

      PADOUT (co->pco)
      PADOUT kro -> pkro

  K [coeff1,coeff2,word11,word12,shift1] (ci->co) -
    kri, mui, b1i, f1i, b2i, f2i -> kro

END

!=============================================================================

OPERATOR MU1 [coeff1,coeff2,word11,shift1] (ci->co) -
         fi, bi, muri, ai -> limo, muto, muro, ao

! code
! designer    : mjr
! last update: 9.5.83
! status      : checked, simulated
! description: Mu update recursion part 1
! delay       : del3 =62

   SIGNAL fsq, bsq, fbsq, mufb, mut, mutdel, del3br, del4br, del5br
   CONTROL c1, c2, c3

   CONSTANT del1 = ((3*coeff1)/2) +2 +1 -(shift1+3)
   CONSTANT del2 = 1
   CONSTANT del3 = ((3*(coeff1+coeff2))/2) +4 +1 +(word11+1)
      CONSTANT del3a = del3 / 2
      CONSTANT del3b = del3 - del3a
   CONSTANT del4 = ((3*coeff2)/2) +2 +(word11+1)
      CONSTANT del4a = del4 / 2
      CONSTANT del4b = del4 - del4a
   CONSTANT del5 = del3
      CONSTANT del5a = del5 / 2
      CONSTANT del5b = del5 - del5a
   CONSTANT del6 = (word11+1)


   MULTIPLY [2,coeff1,0,0] (ci->c1) fi, fi -> fsq, NC
   MULTIPLY [2,coeff1,0,0] (ci->NC) bi, bi -> bsq, NC
   DSHIFT [shift1,0] (ci) muri -> mut

   ADD [1,0,0,0] (c1) fsq, bsq, GND -> fbsq, NC
   BITDELAY [del1] mut -> mutdel

   CBITDELAY [del2] (c1->c2)
   MULTIPLY [2,coeff2,0,0] (c2->c3) fbsq, mutdel -> mufb, NC
!  BITDELAY [del3] muri -> muro
      BITDELAY [del3a] muri -> del3br
      BITDELAY [del3b] del3br -> muro
```

```
   FLIMIT [wordl1,8,0] (c3) mufb -> limo
!  BITDELAY [del4] mutdel -> muto
      BITDELAY [del4a] mutdel -> del4br
      BITDELAY [del4b] del4br -> muto
!  BITDELAY [del5] ai -> ao
      BITDELAY [del5a] ai -> del5br
      BITDELAY [del5b] del5br -> ao

   CBITDELAY [del6] (c3->co)

END

!=================================================================

CHIP CMU1 (pci->pco) pfi, pbi, pmuri, pai -> plimo, pmuto, pmuro, pao

!! code
!! designer    : mjr
!! last update : 6.5.83
!! status      : checked, simulated
!! description : MU part 1
!! delay       : MU1 + 1 =63

   SIGNAL fi, bi, muri, ai, limo, muto, muro, ao
   CONTROL ci, co

   CONSTANT coeff1 = 12
   CONSTANT coeff2 = 8
   CONSTANT wordl1 = 26
   CONSTANT shift1 = 10

   PADORDER VDD, pci, GND, pfi, pbi, pmuri, pai, CLOCK, -
            pco, plimo, pmuto, pmuro, pao

      PADIN (pci->ci)
      PADIN pfi -> fi
      PADIN pbi -> bi
      PADIN pmuri -> muri
      PADIN pai -> ai

      PADOUT (co->pco)
      PADOUT limo -> plimo
      PADOUT muto -> pmuto
      PADOUT muro -> pmuro
      PADOUT ao -> pao

   MU1 [coeff1,coeff2,wordl1,shift1] (ci->co) -
       fi, bi, muri, ai -> limo, muto, muro, ao

END

!=================================================================

OPERATOR MU2 [coeff1,wordl1,wordl2] (ci->co) -
         limi, ai, muti, muri -> muro

! code
! designer    : mjr
! last update : 6.5.83
! status      : checked, simulated
! description : Mu update recursion, part 2
! delay       : del4 + del5 + del6 =83

   SIGNAL trim, mutidel, mutrmsb, mutrlsb, mutrf, muridel, mu, del4br
   CONTROL c1, c2, c3, c4, del3br

   CONSTANT del1 = 1
   CONSTANT del2 = 1
   CONSTANT del3 = (2*wordl1) +1
      CONSTANT del3a = del3 / 2
      CONSTANT del3b = del3 - del3a
   CONSTANT del4 = del3 + del1 + 1
      CONSTANT del4a = del4 / 2
      CONSTANT del4b = del4 - del4a
   CONSTANT del5 = 1
   CONSTANT del6 = wordl1 +1
```

```
   CBITDELAY [del1] (ci->c1)
   SUBTRACT [1,0,0,0] (ci) ai, limi, GND -> trim, NC
   BITDELAY [del2] muti -> mutidel

   DPMULTIPLY [coeff1,0,0,0] (c1->c2) trim, mutidel, GND -> mutrlsb, mutrmsb, NC


!  CBITDELAY [del3] (c2->c3)
      CBITDELAY [del3a] (c2->del3br)
      CBITDELAY [del3b] (del3br->c3)
   FFORMAT2TO1 [word11,44,19,0,0] (c2) mutrlsb, mutrmsb -> mutrf
!  BITDELAY [del4] muri -> muridel
      BITDELAY [del4a] muri -> del4br
      BITDELAY [del4b] del4br -> muridel

   ADD [1,0,0,0] (c3) mutrf, muridel, GND -> mu, NC

   CBITDELAY [del5] (c3->c4)
   FLIMIT [word11,18,0] (c4) mu -> muro

   CBITDELAY [del6] (c4->co)

END

!-------------------------------------------------------------------------

CHIP CMU2 (pci->pco) plimi, pai, pmuti, pmuri -> pmuro

!! code
!! designer    : mjr
!! last update : 2.5.83
!! status      : checked, simulated
!! description : Mu part 2
!! delay       : CMU2 + 1 =84

   SIGNAL limi, ai, muti, muri, muro
   CONTROL ci, co

   CONSTANT coeff1 = 8
   CONSTANT word11 = 26
   CONSTANT word12 = 17

   PADORDER VDD, pci, GND, plimi, pai, pmuti, pmuri, CLOCK, pco, pmuro

      PADIN (pci->ci)
      PADIN plimi -> limi
      PADIN pai -> ai
      PADIN pmuti -> muti
      PADIN pmuri -> muri

      PADOUT (co->pco)
      PADOUT muro -> pmuro

   MU2 [coeff1,word11,word12] (ci->co) limi, ai, muti, muri -> muro

END

!-------------------------------------------------------------------------
```

```
SYSTEM PE (init->c1,c2,clk,clm2) si, a, -> mur3, kr2

! code
! designer    : mjr
! last update : 24.11.82
! status      : checked, simulated
! description : Adaptive prediction-error filter

   SIGNAL kt, krl, murl, al, fl, bl, f2, b2, -
          mur2, lim, mut2, a2

   CONTROL e3, cll, clml

   COS (init->c1,c2,e3,NC,NC,NC) -
       mur3, a, kr2 -> murl, al, kt, krl

   CL (c1,c2,e3->cll) si, kt -> fl, f2, bl, b2

   CK (cll->clk) bl, b2, fl, f2, murl, krl -> kr2

   CMU1 (cll->clml) fl, bl, murl, al -> lim, mut2, mur2, a2

   CMU2 (clml->clm2) lim, a2, mut2, mur2 -> mur3

END

ENDOFPROGRAM
```

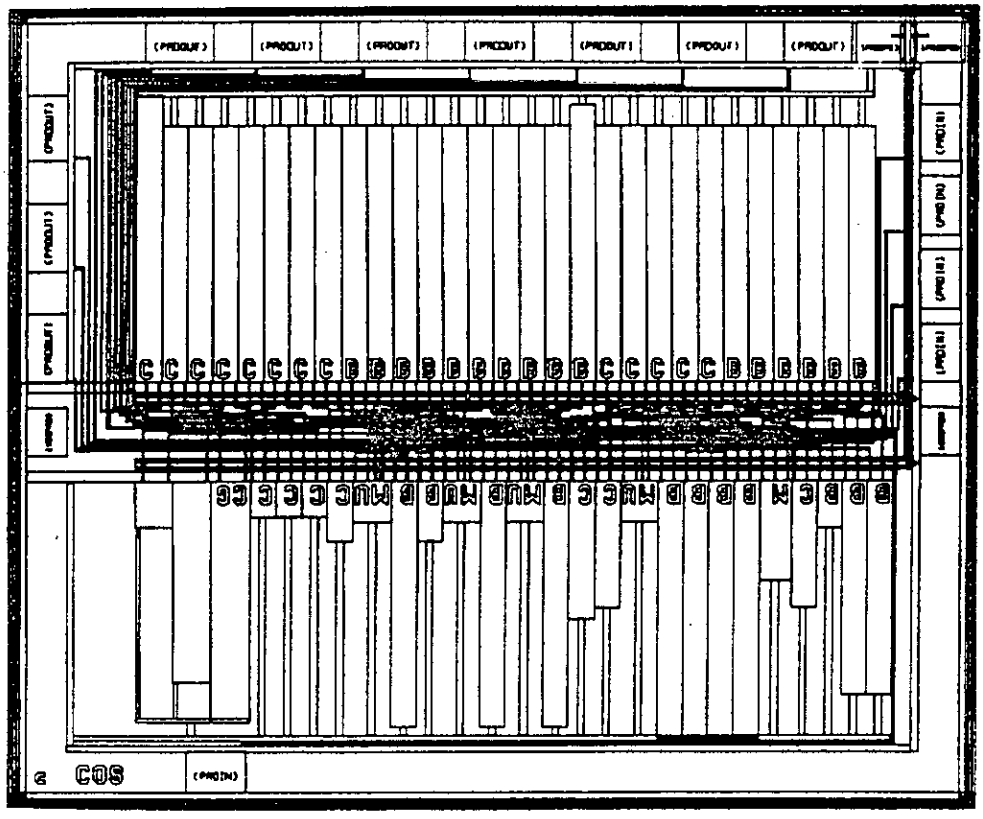


Fig C11 Chip COS floor plan.

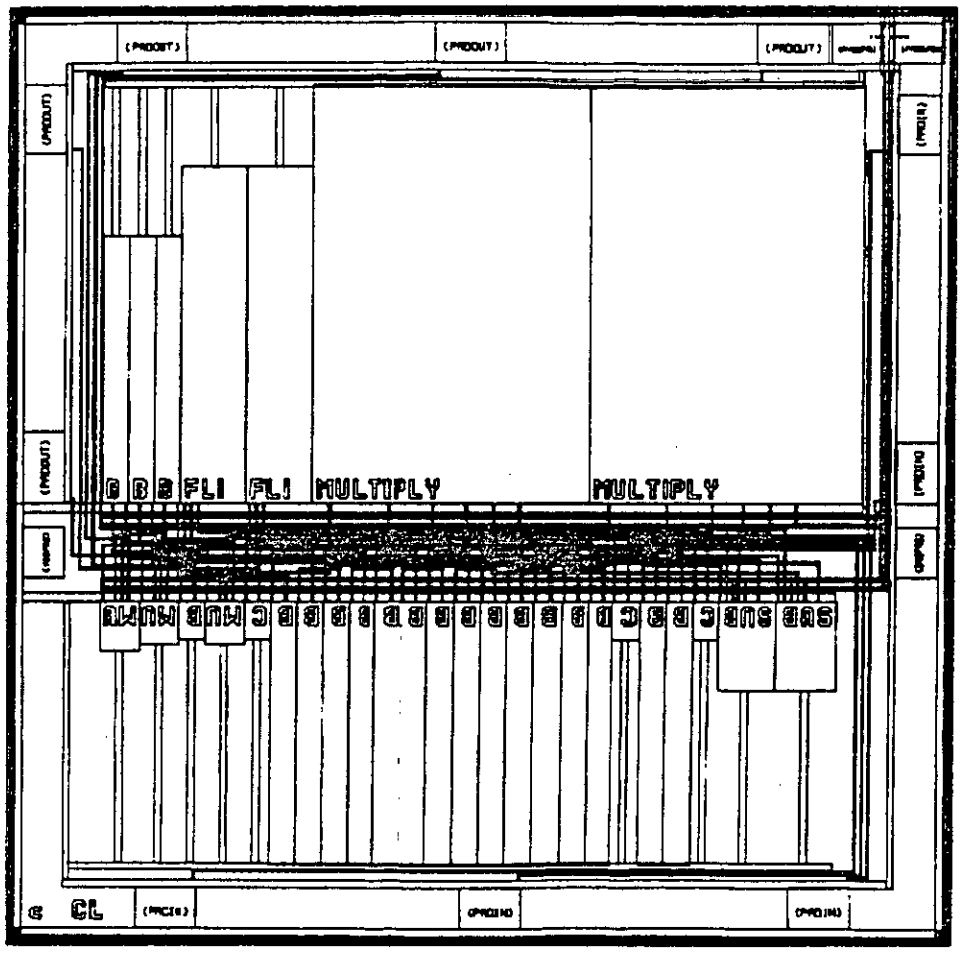


Fig C12 Chip CL floor plan

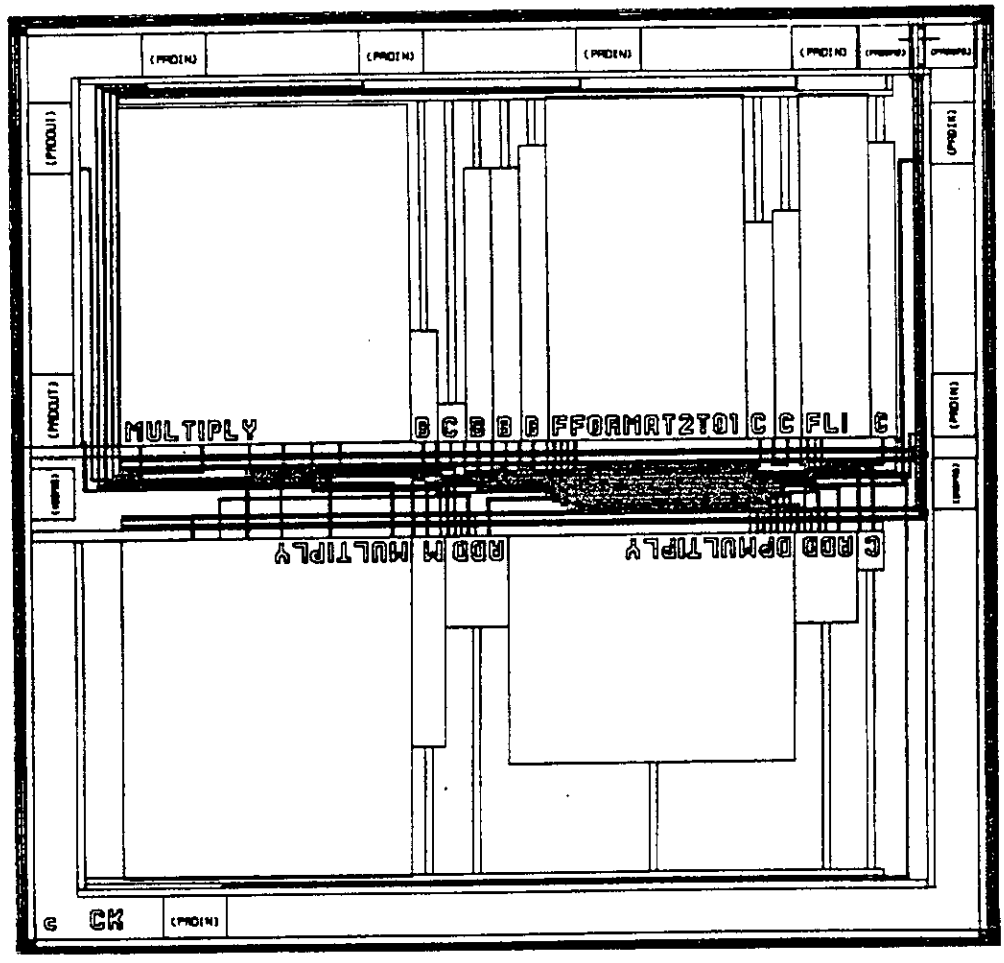


Fig C13 Chip CK floor plan

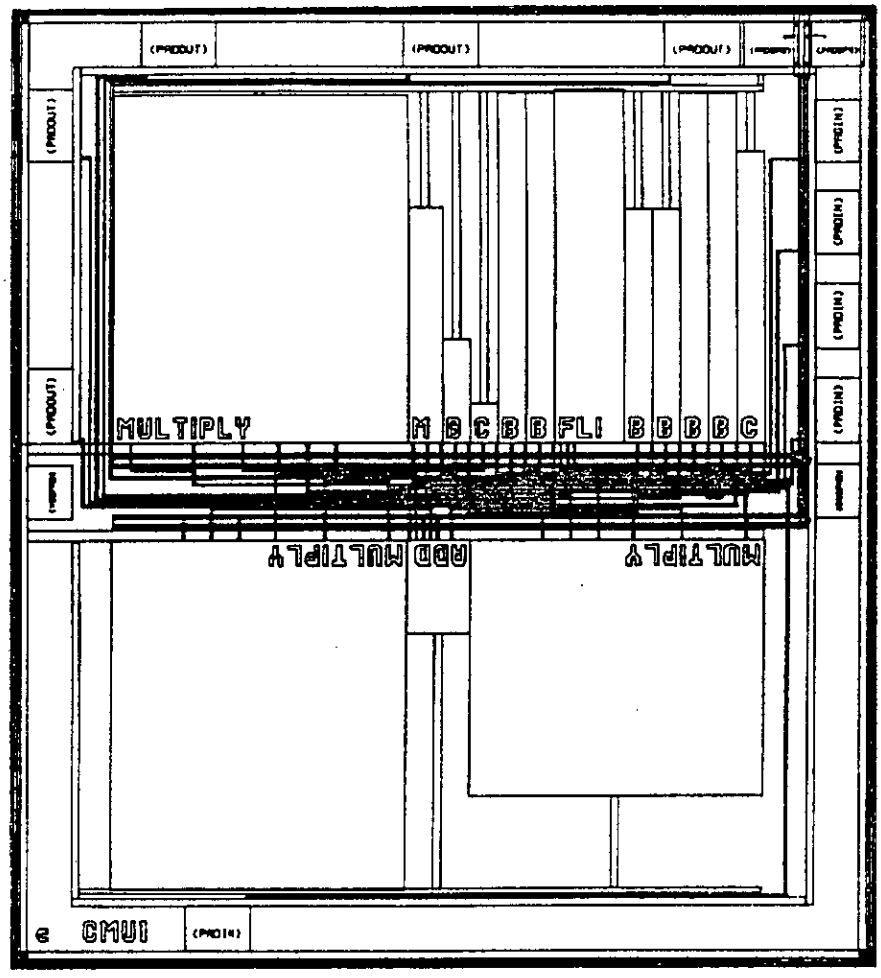


Fig C14 Chip CMU1 floor plan

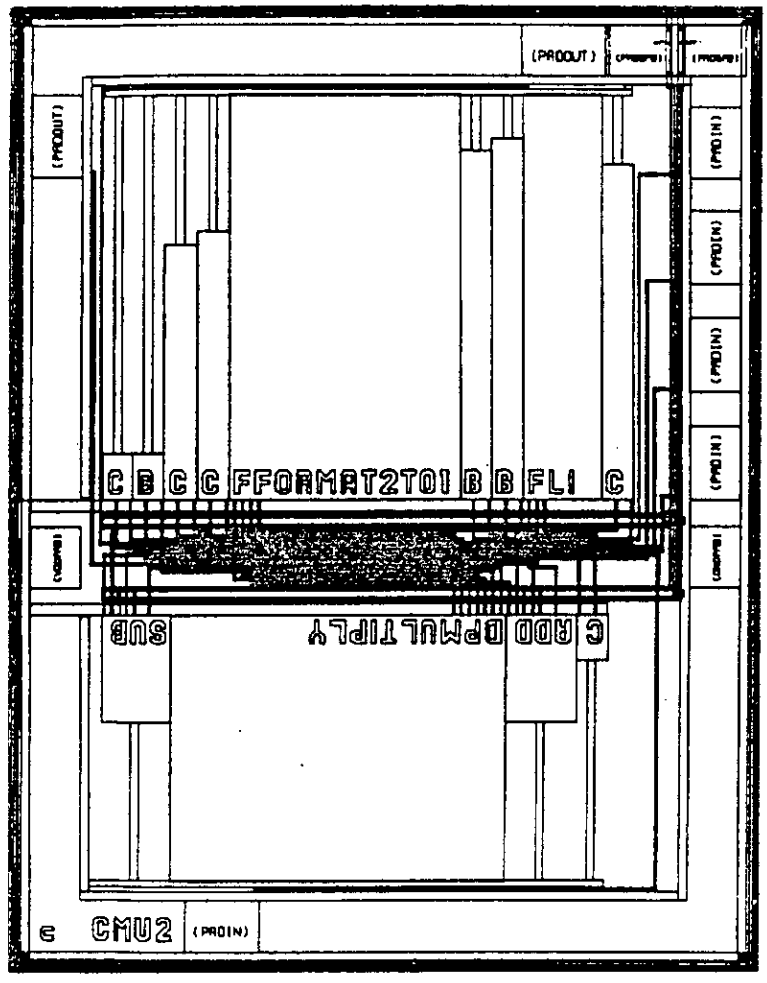
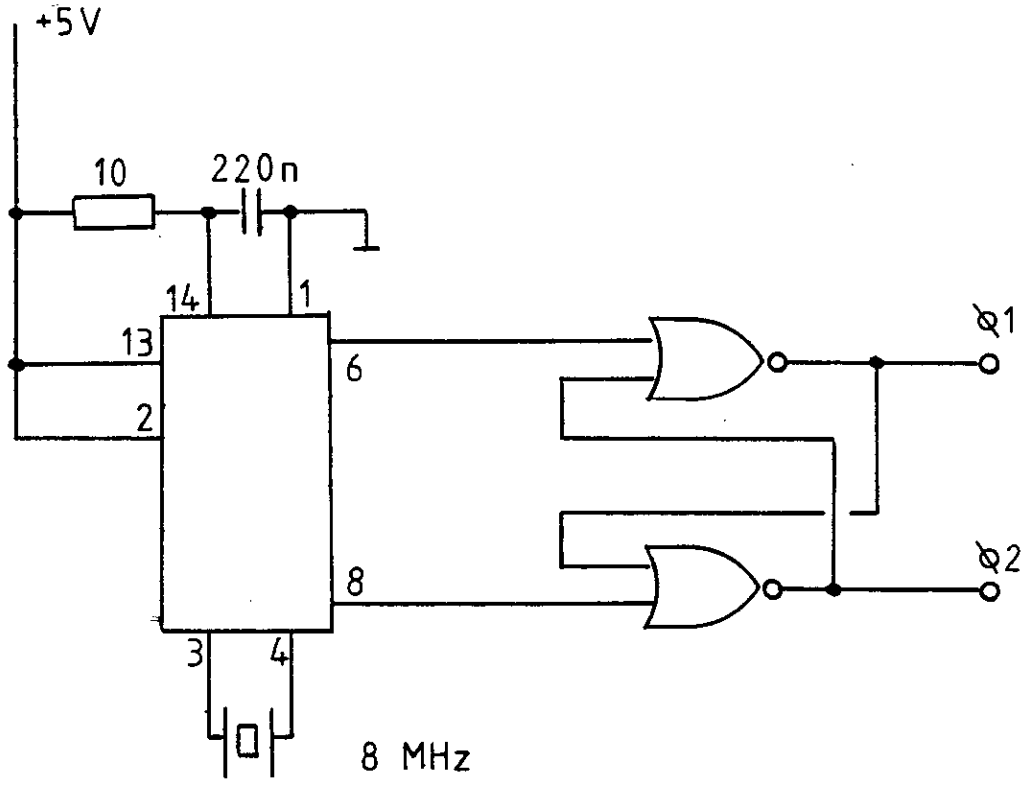


Fig C15 Chip CMU2 floor plan



74 LS 624

Fig C16 Free-running clock generator

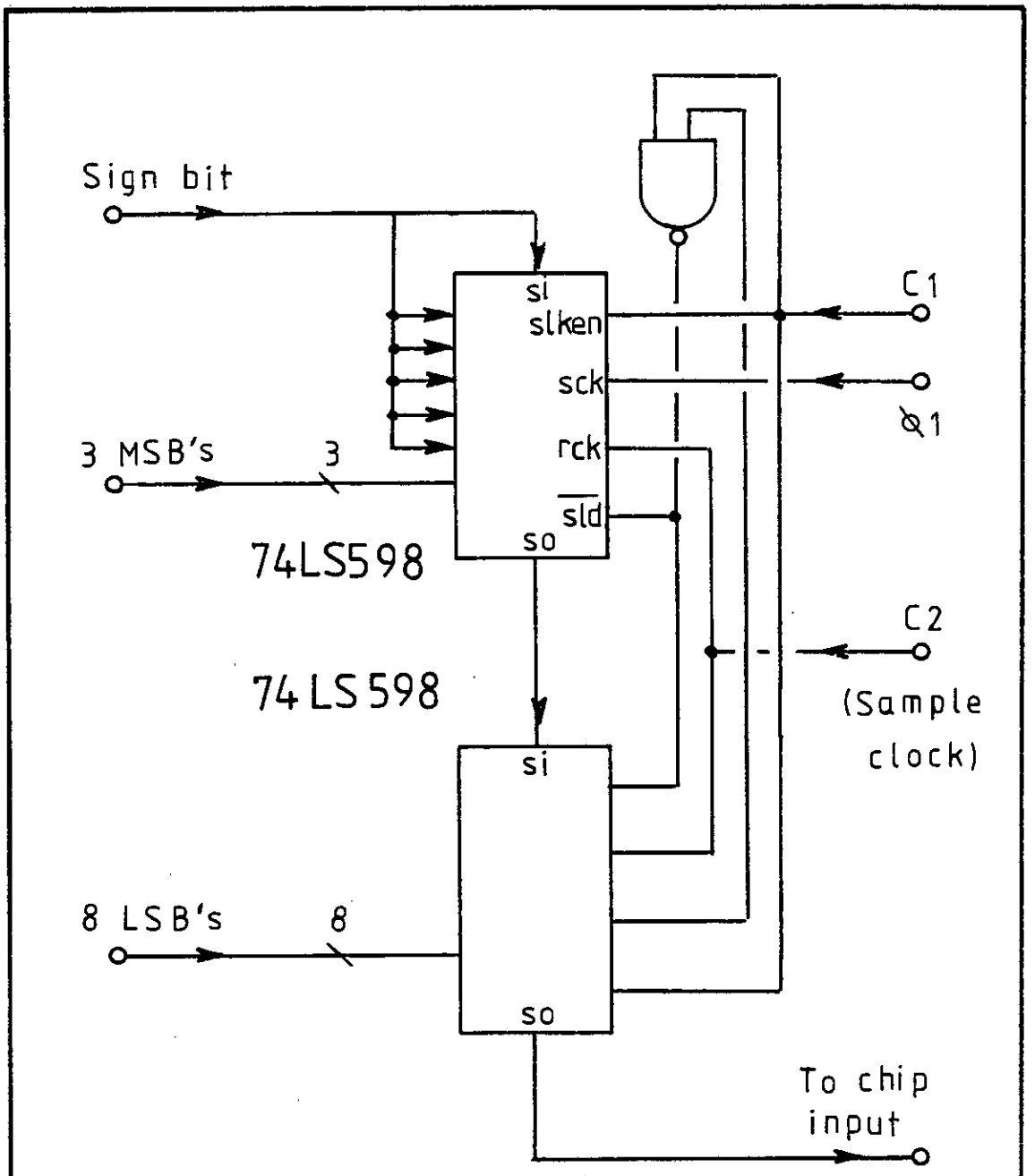


Fig C17 PISO data input interface

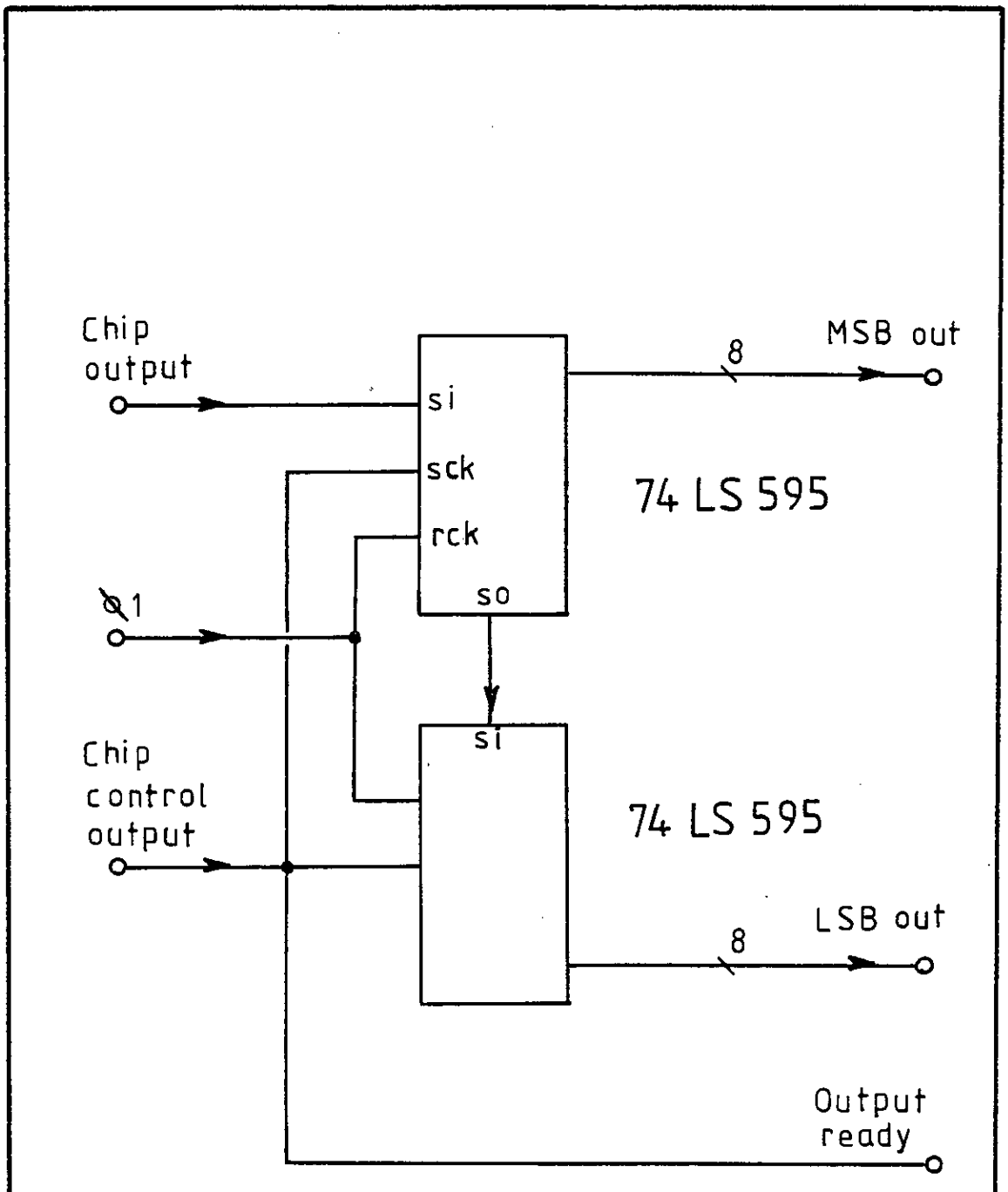


Fig C18 SIPO data output interface

APPENDIX D: BTRL Impulse Responses

Chapter 4 describes an analysis of various channel impulse responses. The responses themselves are listed here.

The listings each begin with two blank lines, followed by the channel identification letters (see section 4.5) and then another blank line. The following sixteen lines represent the sixteen points of the impulse responses. The first number on the line represents the real component, the second the imaginary component.

E10C4

```
 1.68133171856E-03,      1.73585035003E-04
 3.79758124637E-03,      4.43826722737E-03
-5.01861755846E-03,     -6.19417612450E-03
-6.46684098897E-04,      3.30503164835E-03
 1.79249838603E-02,      7.78499920470E-03
-5.50312160609E-02,     -3.28481878263E-02
 1.31696593268E-01,      8.18982132334E-02
-3.53366305977E-01,     -1.71616582357E-01
 1.00000000000E+00,     -4.17841925168E-01
 7.64244892027E-01,      8.36712146644E-01
 1.11968117046E-01,     -3.62233941727E-01
-1.21889186435E-01,      5.61335466150E-02
 6.77598875227E-02,      1.29052835481E-02
-3.89152521613E-02,     -2.04123634956E-02
 2.21654953422E-02,      1.46754803886E-02
-1.03339937966E-02,     -6.87940058365E-03
```

R11

```
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
0.27980292,             0
1,                      0
0.27980292,             0
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
```

R21

```
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
0.33646601,             0
1,                      0
0.33646601,             0
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
0,                      0
```

E15

```
-7.56113215393E-04,     +2.94581655436E-12
+5.45362786262E-04,     +3.21168520072E-03
+1.95692150250E-04,     -4.34619508846E-03
-1.75882222186E-03,     +3.76872584486E-04
+4.60161907143E-03,     +1.34090478174E-02
-9.73167692692E-03,     -4.46388901168E-02
+2.15373067596E-02,     +1.10126370213E-01
-8.05558875435E-02,     -2.82835918438E-01
+1.00000000000E-00,     -8.95333673915E-11
-8.05558875765E-02,     +2.82835918414E-01
+2.15373067363E-02,     -1.10126370200E-01
-9.73167692692E-03,     +4.46388900993E-02
+4.60161906738E-03,     -1.34090478063E-02
-1.75882221480E-03,     -3.76872593488E-04
+1.95692144114E-04,     +4.34619509825E-03
+5.45362785238E-04,     -3.21168520921E-03
```

E8C3A

```
-1.87394306222E-02,     +1.59836570441E-02
+5.11703658089E-03,     -9.52839128522E-03
-8.10296852686E-04,     +7.88419927292E-03
+1.13735167866E-02,     -1.10077689936E-02
-4.51965518294E-02,     +2.13466591082E-02
+1.17798531390E-01,     -4.74310590041E-02
-2.84472919891E-01,     +1.26855564536E-01
+4.36569776055E-01,     -7.53171201068E-01
+1.00000000000E-00,     +6.71485336616E-01
+1.26325452032E-01,     +1.98726376371E-01
+2.48533564133E-01,     -2.15694875107E-01
-2.09819608887E-01,     -7.08275856658E-02
+4.57636788008E-02,     +1.39290533279E-01
+3.58041768564E-02,     -1.01071406275E-01
-4.95065899791E-02,     +5.71519624156E-02
+3.65577010863E-02,     -2.99920013950E-02
```

E8C3

```
+4.49577812179E-03,     +1.14814517548E-03
-6.17207789415E-03,     -1.57710038991E-03
+2.24379342047E-03,     +1.75320333035E-03
+1.15230570240E-02,     -1.20364035525E-03
-4.21537183353E-02,     -6.85691157226E-04
+1.03760670664E-01,     +3.63057927225E-03
-2.50610613044E-01,     +1.46071478427E-03
+6.51240929936E-01,     -4.73070592715E-01
+1.00000000000E-00,     +7.06626229583E-01
+1.16489694412E-02,     -2.90530503807E-01
-6.47391035493E-02,     +6.32165196167E-02
+4.43442177661E-02,     -1.17619588723E-02
-2.95598227276E-02,     +1.26445253033E-03
+1.79628688442E-02,     +3.41246167581E-04
-8.20418297007E-03,     +8.30481060418E-06
+5.79956262218E-04,     -6.19930065487E-04
```

E5

```
-2.72541784654E-05,     +1.04670029832E-12
+1.93394437256E-05,     +1.13586914196E-03
+8.33999824381E-06,     -1.52627953088E-03
-6.64327123189E-05,     +8.81466474808E-05
+1.72098725737E-04,     +4.86518957601E-03
-3.64812552871E-04,     -1.60826756041E-02
+8.23173028763E-04,     +3.97660850807E-02
-3.11606077442E-03,     -1.05359591005E-01
+1.00000000000E-00,     -9.62539268379E-11
-3.11606079061E-03,     +1.05359590999E-01
+8.23173013118E-04,     -3.97660850755E-02
-3.64812551596E-04,     +1.60826755984E-02
+1.72098723284E-04,     -4.86518957203E-03
-6.64327066651E-05,     -8.81466505404E-05
+8.33999257635E-06,     +1.52627953421E-03
+1.93394417339E-05,     -1.13586914488E-03
```

E5R

```
-2.72541784654E-05,     0
+1.93394437256E-05,     0
+8.33999824381E-06,     0
-6.64327123189E-05,     0
+1.72098725737E-04,     0
-3.64812552871E-04,     0
+8.23173028763E-04,     0
-3.11606077442E-03,     0
+1.00000000000E-00,     0
-3.11606079061E-03,     0
+8.23173013118E-04,     0
-3.64812551596E-04,     0
+1.72098723284E-04,     0
-6.64327066651E-05,     0
+8.33999257635E-06,     0
+1.93394417339E-05,     0
```

R81

0,	0
0,	0
0,	0
0,	0
0,	0
0,	0
0,	0
0.4,	0
1,	0
0.4,	0
0,	0
0,	0
0,	0
0,	0
0,	0
0,	0

APPENDIX E: Publications

(* denotes paper reprinted here)

1. M.J. Rutter, P.M. Grant and C.F.N. Cowan, "Adaptive lattice filters for channel equalisation," IEE Colloquium digest 82/52, May 1982, pp 9-1 to 9-4.

2.* M.J. Rutter and C.F.N. Cowan, "Open-loop transversal equaliser of optimal length," Electronics Letters, Vol. 9, No. 6, March 1983, pp 208-210.

3. M.J. Rutter, P.M. Grant, D. Renshaw and P.B. Denyer, "Design and realisation of adaptive lattice filters," Proc. IEEE Int. Conf. ASSP, Boston, April 1983, pp 21-24.

4. S.G. Smith, C.F.N. Cowan and M.J. Rutter, "A new structure for adaptive echo cancellation," Proc. IEEE Int. Conf. ASSP, Boston, April 1983, pp 49 - 52.

5.* M.J. Rutter and P.M. Grant, "Generation of Codes with good autocorrelation properties," Electronics letters, Vol. 19, No. 15, July 1983, pp 571-572.

6. M.J. Rutter and P.M. Grant, "The application of gradient adaptive lattice filters to channel equalisation," submitted for publication to Proc. IEE, pt F.

OPEN-LOOP TRANSVERSAL EQUALISER OF OPTIMAL LENGTH

Indexing terms: Filters, Equalisers

As a rule, in equaliser training algorithms complexity increases with efficiency. Unfortunately, complex recursive algorithms do not suit high-bandwidth systems. This letter describes a nonrecursive technique for the derivation of the weight vector in an optimally variable length transversal equaliser.

Introduction: Complicated real-time training algorithms are not compatible with the operation of high-bandwidth equalisers; however, simpler algorithms converge more slowly. In the open-loop solution proposed here, the training period is used to collect data on the channel. The optimum transversal equaliser tapweights are later calculated and installed. This letter describes a way of optimising both tapweight values and filter length, minimising tap-generated truncation noise. Algorithmic notation has been used for clarity and to assist later implementation.

Hardware design: The equaliser itself is a simple transversal filter, implemented in analogue or digital form. According to the Weiner–Hopf equation,[3] the optimum tapweights $H(i)$ for a filter of length N satisfy

$$P(j) = \sum_{i=0}^{N-1} R(i-j)H(i) \tag{1}$$

where P is the crosscovariance function between equaliser input S and training input T, truncated to length N:

$$P(j) = E\langle S(i)T(i-j)\rangle \tag{2}$$

The function R is the aperiodic autocovariance function of the channel:

$$R(i-j) = E\langle S(i)S(j)\rangle \tag{3}$$

The function $E\langle\,.\,\rangle$ is the expectation operator. Thus the information to be gathered to calculate the optimum tap value is the autocovariance function of the channel $R(0 \ldots N-1)$ and the crosscovariance function $P(0 \ldots N-1)$. These are calculated using correlators in the hardware of Fig. 1. Also shown is the variable-length equaliser, but the general-purpose processor which calculates the tapweights is omitted. Accurate covariance estimates are obtained by using a code of low autocorrelation as a training sequence, after Yucel.[2]

Tapweight calculation: The tap settings are calculated using the Levinson–Trench algorithm,[1] which is best understood

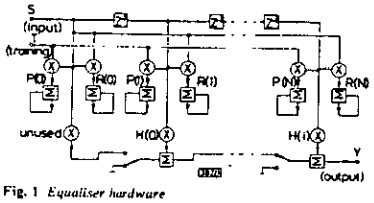


Fig. 1 *Equaliser hardware*
Processor not shown

through the lattice equaliser[4] (Fig. 2). It is emphasised that the lattice is mentioned here only as a mathematical concept. The calculated hardware taps are for a simple transversal filter.

First, the Levinson recursion[3] is used on the autocorrelation function to produce $K(1 \ldots N-1)$, the PARCOR coefficients of the lattice prediction-error filter, $Q(0 \ldots N-1)$, the

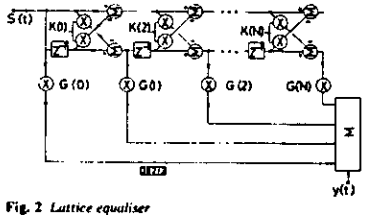


Fig. 2 *Lattice equaliser*

prediction error power at the output of each stage at the lattice and $A(0 \ldots N-1, 0 \ldots N-1)$, the pulse response seen at the output of each lattice stage. Thus $A(i, j)$ refers to the output of the ith stage j clock periods after a unit pulse has been applied to the filter input. From the properties of the lattice,

$$Q(0) = R(0) \tag{4}$$

$$A(i, i) = 1$$

$$A(i, j) = 0 \quad \text{when } j > i \text{ or } j < 0 \tag{5}$$

The recursion proceeds stage by stage:

$$K(i) = \left(\sum_{j=0}^{i-1} A(i-1, j)R(j+1)\right) \Big/ Q(i-1) \tag{6}$$

$$A(i, j) = A(i-1, j-1) - K(i)A(i-1, i-1-j) \tag{7}$$

$$Q(i) = Q(i-1)(1 - K^2(i)) \tag{8}$$

Since the outputs of the taps are mutually orthogonal, $G(0 \ldots N-1)$, the side-tap values are simply the normalised cross-correlation between the lattice stage output and the training signal, i.e.

$$G(0) = P(0)/Q(0)$$

$$G(i) = \left(\sum_{j=0}^{i} P(j)A(i, j)\right) \Big/ Q(i) \tag{9}$$

The transversal equaliser impulse response is $H(0 \ldots N-1, 0 \ldots N-1)$, where $H(i, j)$ refers to the output of an i-tap equaliser after the jth clock period:

$$H(0, 0) = G(0)$$

$$H(i, j) = H(i-1, j) + A(i, j)G(i), \quad \text{if } 0 \leq j \leq i$$
$$= 0, \text{ otherwise} \tag{10}$$

Note that the triangular nature of matrix A causes the length of H to increment at each recursion.

Equaliser length minimisation: Since D, the power content of the training sequence T, is known beforehand, and since the signals on the taps are orthogonal, the power E of the error signal (the difference between ideal and actual equaliser outputs) may be calculated within the recursion:

$$E(0) = D - Q(0)G^2(0)$$

$$E(i) = E(i-1) - Q(i)G^2(i) \tag{11}$$

The recursion is stopped when the number of taps used has caused the error to fall below a predefined threshold value.

While the Weiner–Hopf equation may appear clear-cut, there is nevertheless uncertainty in selecting the optimum section of P. A rule of thumb is to locate its maximum in the centre of the equaliser. Here, with very little extra computational effort, a number of sections of P may be tried. The recursion is stopped when one of these yields a satisfactory error level.

Simulation results: Fig. 3 shows a computer simulation of two equalisers operating on a channel of eigenvalue ratio 11. It is a graph of iteration number against mean error power, calculated by convolving channel and equaliser pulse responses. The solid line shows, for comparison, an 11-tap Widrow LMS algorithm with step size optimised for fastest convergence,[5] converging on a random binary training signal.

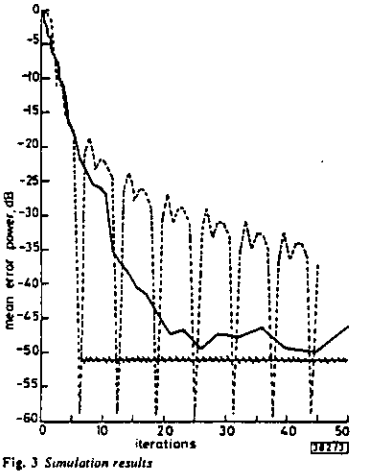


Fig. 3 *Simulation results*

——— error power of 11-tap transversal adaptive filter
- - - - error power of 11-tap transversal filter calculated by correlation
+-+-+ error power of transversal filter calculated by correlation and frozen at 10 taps after 63 iterations

The dotted line shows the 'convergence' of an 11-tap open-loop equaliser on a 63-point code of low autocorrelation. Each 63rd iteration yields an exact result. The hatched horizontal line shows how the equaliser would be used to give an optimum number of taps. After 63 iterations had given an accurate estimate of the channel's characteristics, data collection was stopped. The transversal taps were then calculated using the condition that the error power had to be lower than -50 dB. This was found to be possible using a 10-tap equaliser. The shorter equaliser was then used, giving a mean error power of $-51{\cdot}4$ dB.

Conclusions: This letter has described an open-loop equaliser structure. It operates nonrecursively measuring line characteristics using correlators and training sequences of low autocorrelation. The result is a set of tapweights for a transversal equaliser, which is calculated using lattice filter equations. This transversal equaliser is of the minimum length which can achieve a prespecified error power. The equaliser also has the advantage of using a far shorter training sequence than most adaptive filters.

Acknowledgments: The authors acknowledge the support of the UK Science & Engineering Research Council. We would also like to thank Dr. P. M. Grant for his advice and encouragement.

M. J. RUTTER
C. F. N. COWAN

Department of Electrical Engineering
University of Edinburgh
The King's Buildings, Edinburgh EH9 3JL, Scotland

4th February 1983

References

1 BUTLER, P., and CANTONINI, A.: 'Noniterative automatic equalisation', *IEEE Trans.*, 1975, COM-23, pp. 621–633
2 YUCEL, M. D., TEPEDELENLIOGHU, N., and TANIK, Y.: 'A fast noniterative method for channel equalisation'. Short Communications and Poster Digest, EUSIPCO, Lausanne, 1980, pp. 155–156; also private communication to P. M. Grant
3 HAYKIN, S., and KESLER, S.: 'Prediction-error filtering and maximum-entropy spectral analysis' *in* HAYKIN, S. (Ed.): 'Nonlinear methods of spectral analysis' (Springer-Verlag, Berlin, 1979)
4 GRIFFITHS, L. J.: 'An adaptive lattice structure for noise-cancelling applications', IEEE Trans. ICASSP, Tulsa OK, April 1978, pp. 87–90
5 UNGERBOECK, G.: 'Theory on the speed of convergence in adaptive equalisers for digital communication', *IBM J. Res. & Dev.*, 1972, 16, pp. 546–555

0013-5194/83/060208-03$1.50/0

GENERATION OF CODES WITH GOOD AUTOCORRELATION PROPERTIES

Indexing term: Codes

The letter describes a conceptually simple method of generating impulse equivalent burst and cyclic codes, using an all-pass autoregressive moving average (ARMA) filter. These infinite impulse response burst codes may be converted into cyclic codes of arbitrary length, while retaining their low autocorrelation time sidelobe performance. Their extension to orthogonal coding is explored.

Introduction: In digital communications there is often a need for spectrally white burst or cyclic codes which provide a sharp autocorrelation peak with low time sidelobes. Applications include spread spectrum communication, radar and navigation systems. Approximately white burst and cyclic codes[1] have been generated using moving average (MA) finite impulse response filters. This letter describes a method of using a digital ARMA filter to generate amplitude modulated codes which are precisely white, for subsequent detection in a MA receiving filter.

An all-pass filter[2] is employed to synthesise the codes, where the poles in the left-hand s-plane are mirrored by zones in the right-hand s-plane. This relationship generates a spectrally white impulse equivalent sequence[3] whose autocorrelation function shows a sharp peak at zero lag with low values for all other lags.

Fig. 1 shows a fifth-order ARMA filter, where the feedback (recursive) coefficients exactly mirror the feed forward coefficients to provide a transfer function:

$$T(z) = (a_n + a_{n-1}z^{-1} + \ldots + a_1 z^{-(n-1)} + z^{-n}) / (1 + a_1 z^{-1} + \ldots + a_n z^{-n}) \quad (1)$$

with an all-pass response.

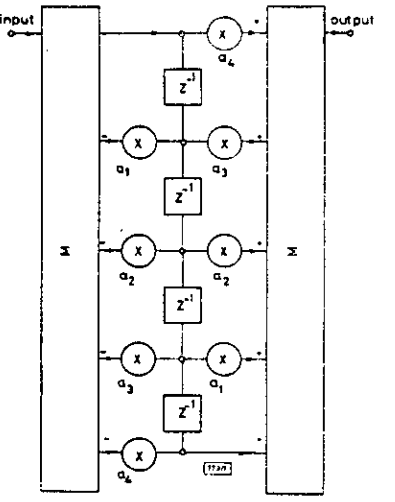


Fig. 1 *Fifth-order ARMA all-pass digital filter*

Burst code: A calculation was performed on a second-order ARMA filter with transfer function:

$$T(z) = (0{\cdot}5 + z^{-1})/(1 + 0{\cdot}5z^{-1}) = \tfrac{1}{2} + (\tfrac{3}{4})z^{-1} - (\tfrac{1}{8})z^{-2} + (\tfrac{1}{16})z^{-3} \ldots \quad (2)$$

This exhibits an infinite impulse response with most of the energy concentrated in the first few terms. Observations on a number of similarly coded sequences indicated that the energy was fairly evenly distributed over a number of terms equal to the order of the filter. Thereafter the energy decayed exponentially. Locating the zero close to the unit circle produces a filter response with slow decay and energy spread widely in time. The optimum rate of decay, however, depends on the application.[4]

The first $m = 100$ terms of the series in eqn. 2 were evaluated and the autocorrelation function was calculated. The zero lag term, corresponding to power content, was unity. All other lags had values of zero to within the calculation accuracy (10^{-18}).

Thus in a practical system this code can be generated by impulsing a second-order ARMA filter and the matched filter may be realised as a 100 tap MA filter. Huffman[3] has shown that the receiving filter can be simplified by applying a second smaller time delayed impulse to the ARMA filter to terminate the waveform to less than the full hundred samples. This introduces discontinuities at the edges of the autocorrelation function, Fig. 2. A similar degradation was observed when the MA receiving filter was truncated to a length shorter than 100 stages.

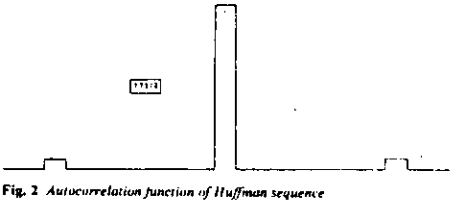


Fig. 2 *Autocorrelation function of Huffman sequence*

Cyclic codes: These burst codes are easily-generated but the receiving MA filter is of necessity long if the lags of the autocorrelation function are to be minimised. One method of overcoming this limitation is to use a cyclic code by impulsing the ARMA filter at a rate greater than the 100 bit code repetition

rate. The individual impulse response values are summed to form a composite code whose length is governed by the repetition period of the impulses.

Table 1 CYCLIC CODE OF LENGTH 6 AND ITS AUTO-CORRELATION FUNCTION

	Code sequence	Autocorrelation
0	+0·761904	+1·000
1	−0·380952	$+5{\cdot}421 \times 10^{-20}$
2	+0·190476	$-4{\cdot}065 \times 10^{-20}$
3	−0·095238	$+4{\cdot}065 \times 10^{-20}$
4	+0·047619	$-6{\cdot}098 \times 10^{-20}$
5	+0·476190	$+3{\cdot}049 \times 10^{-20}$

This generation of a cyclic code has been found to retain the exceptionally low values in the lags of the autocorrelation function of the burst code and it greatly simplifies the receiving filter which reduces to one of order equal to the number of bits in the code. Table 1 shows the impulse response for a $m = 6$ bit cyclic sequence derived from eqn. 2 and it provides the values for the lags of the autocorrelation function. If the order of this sequence is less than twice the order of the ARMA filter, then the generating filter can also be simplified from an ARMA to a MA design. A key attraction of this synthesis procedure is that codes can be generated with perfect autocorrelation properly for any length, avoiding the powers of two restriction normally experienced with pseudo noise maximal length codes.[5]

Table 2 CYCLIC CODE OF ZERO DC CONTENT AND ITS AUTO-CORRELATION FUNCTION

	Code sequence	Autocorrelation
0	+0·428571	+1·000
1	+0·285714	$+9{\cdot}485 \times 10^{-20}$
2	+0·857142	$-1{\cdot}084 \times 10^{-19}$
3	−0·428571	−1·000
4	−0·285714	$-9{\cdot}485 \times 10^{-20}$
5	−0·857142	$+1{\cdot}084 \times 10^{-19}$

These codes all have a DC content of unity. This may be alleviated by generating the codes with impulses of alternate polarity. Table 2 gives the impulse response and autocorrelation function for a cyclic $m = 6$ bit code generated from eqn. 2. The power content has doubled (due to the higher impulse rate) and the autocorrelation function is zero for all lags except zero and m 2.

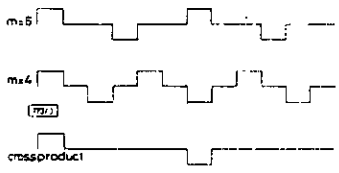


Fig. 3 *Antisymmetric cyclic orthogonal codes of length four and six and their crossproduct*

These cyclic codes can also be extended to orthogonal signalling. Fig. 3 shows the two distinct input impulse trains which could be applied to an ARMA filter to generate two cyclic codes of length four or six. The product of the two codes is another antisymmetric code of length equal to the lowest common multiple (LCM) of the two other cycle lengths (twelve). Fig. 3. If the MA filters which receive these codes are designed with an order equal to the LCM of the individual codes then the crosscorrelation between the codes is zero while the individual codes retain their antisymmetric autocorrelation response. Thus orthogonality between the codes can be achieved at the expense of an increased receiver complexity and an effective reduction in the data rate.

Summary: This letter has described a new technique for the generation burst and cyclic amplitude modulated coded waveforms with good autocorrelation properties. As these codes are related to Huffman sequences the Doppler performance can be controlled by adjusting the generating ARMA coefficients[4] to a desired ambiguity characteristic. Although our example has only incorporated amplitude modulation it is expected that it can be extended to amplitude and phase modulated waveforms by adopting a less restricted choice for the pole-zero locations in the generating filter.

Acknowledgment: The authors wish to acknowledge the sponsorship of the Science and Engineering Research Council.

M. I. RUTTER
P. M. GRANT

Department of Electrical Engineering
University of Edinburgh
King's Buildings, Edinburgh EH9 3JL, Scotland

20th May 1983

References

1 ACKROYD, M. H.: 'Amplitude and phase modulated pulse trains for radar', *Radio & Electronic. Eng.*, 1971, **41**, pp. 541–552
2 KUO, F. F.: 'Network analysis and synthesis' (Wiley and Sons Inc., 1962)
3 HUFFMAN, D. A.: 'The generation of impulse-equivalent pulse trains', *IEEE Trans.*, 1962, **IT-8**, pp. S10–S16
4 ACKROYD, M. H.: 'The design of Huffman sequences', *IEEE Trans.*, 1970, **AES-6**, pp. 790–796
5 GOLOMB, S. W. (Ed.): 'Digital communications with space applications' (Prentice-Hall, 1964)