# SIGNAL PROCESSING IN A LOUDSPEAKING TELEPHONE.

Author: Colin Robert South.

Submitted for the degree of Ph.D.

The University of Aston in Birmingham.

February 1985.

LIST OF CONTENTS.

The University of Aston in Birmingham.

# Signal Processing in a Loudspeaking Telephone.

Author: Colin Robert South.

Submitted for the degree of Ph.D.

February 1985.

## SUMMARY.

One of the major problems associated with communication via a loudspeaking telephone (LST) is that, using analogue processing, duplex transmission is limited to low-loss lines and produces a low acoustic output. An architecture for an instrument has been developed and tested, which uses digital signal processing to provide duplex transmission between a LST and a telephone handset over most of the B.T. network.

Digital adaptive-filters are used in the duplex LST to cancel coupling between the loudspeaker and microphone, and across the transmit to receive paths of the 2-to-4-wire converter. Normal movement of a person in the acoustic path causes a loss of stability by increasing the level of coupling from the loudspeaker to the microphone, since there is a lag associated with the adaptive filters learning about a non-stationary path. Control of the loop stability and the level of sidetone heard by the handset user is by a microprocessor, which continually monitors the system and regulates the gain. The result is a system which offers the best compromise available based on a set of measured parameters. A theory has been developed which gives the loop stability requirement based on the error between the parameters of the filter and those of the unknown path.

A programme to develop a low-cost adaptive filter in LSI produced a unique architecture which has a number of features not available in any similar system. These include automatic compensation for the rate of adaptation over a 36 dB range of input level, 4 rates of adaptation (with a maximum of 165 dB/s), plus the ability to cascade up to 4 filters without loss of performance. A complete theory has been developed to determine the adaptation which can be achieved using finite-precision arithmetic. This enabled the development of an architecture which distributed the normalisation required to achieve optimum rate of adaptation over the useful input range. Comparison of theory and measurement for the adaptive filter show very close agreement.

A single experimental LST was built and tested on connections to handset telephones over the BT network. The LST demonstrated that duplex transmission was feasible using signal processing and produced a more comfortable means of communication between people than methods employing deep voice-switching to regulate the local-loop gain. Although, with the current level of processing power, it is not a panacea and attention must be directed toward the physical acoustic isolation between loudspeaker and microphone.

Indexing terms: Duplex Transmission, Loudspeaking Telephone, Adaptive Filter, Gain regulation.

## LIST OF CONTENTS.

## LISTS OF FIGURES AND TABLES.

**TABLES.**

**ACKNOWLEDGEMENTS.**

Professor J.E. Flood for his assistance in the latter part of the project.

Professor M. Ackroyd, Mr. C.E. Rowlands, and Dr. K. White for their early assistance.

Mr. A.V. Lewis for his assistance during the project, particularly with the timing specification for the DAF and for analogue circuits in the experimental LST.

Mr. K. Murray for his TTL implementation of the multiplier and triple accumulator.

Mr. C. Taylor for his TTL implementation of the memory and for the computer controlled measurement equipment.

Mr. P. Jones for construction of cases for the prototype LST.

Mr. J. Harbridge and Mr. A. Cox for LSI design of the convergence circuit.

Dr. C. Walmsley and Mr. D Talbot, of Walmsley Microelctronics, for LSI design of the multiplier-triple-accumulator and memory circuits respectively.

Mr. P. See for measurements on subjective assessment of the depth of voice switching

Mrs. Pat Hicks for her work on all the diagrams.

Pat, Claire and Alex for their long sufferance during the writing of the thesis.

The Director of Research for permission to carry out the necessary research at BTR Labs and to publish this thesis.

## SYMBOLS.

| | |
|---|---|
| $a$ | feedforward coefficient in IIR filter |
| $a_a$ | electrical gain in acoustic path |
| $a_e$ | electrical gain in hybrid path |
| $A_1$ | Attenuation from person to LST |
| $A_m$ | Attenuation from loudspeaker to microphone |
| $b$ | feedback coefficient in IIR filter |
| $\mathbf{B}$ | vector of feedback coefficients |
| $\beta$ | vector of input, output-signal values of IIR filter |
| $C$ | system constant |
| $c$ | in hardware equations – cascade coefficient |
| $c$ | velocity of light |
| $d$ | misadjustment noise in hardware filter |
| $\mathbf{D}$ | vector of misadjustment noise d |
| $D$ | spectrum of misadjustment vector |
| $\Delta f$ | frequency resolution |
| $\Delta h$ | update (correlation) component |
| $\Delta t$ | time resolution |
| $\Delta \mathbf{H}$ | vector of update component |
| $\delta$ | quantisation step (subscript indicates source) |
| $\delta_h$ | least-significant-digit (impulse response) |
| $e$ | error |
| $E$ | energy |
| $\xi$ | misadjustment error in normalised filter |
| $f$ | noise component of hardware filter |
| $f$ | frequency in Hz |
| $f_r$ | relative frequency |
| $g$ | scaler for filter output |
| $G$ | Gain required for LST |

| | |
|---|---|
| $\gamma$ | measure of error of expected energy from instantaneous estimate. |
| $\gamma_s$ | as above but containing SHL quantisation contribution. |
| $\gamma_a$ | as above but containing PCM quantisation contribution. |
| $h$ | coefficient of impulse response |
| $H$ | vector of h |
| $h^*$ | estimate of impulse-response coefficient |
| $H^*$ | vector $h^*$ |
| $H, H^*$ | spectrum of H and $H^*$ |
| $I$ | unit vector |
| $J$ | function to be minimised in SER algorithm |
| $k$ | additional scaler for hardware-filter output |
| $L$ | Line loss |
| $\lambda$ | term used for brevity |
| $m$ | bit size of data |
| $M_a, M_e$ | loss across acoustic and hybrid paths in dBs |
| $n$ | noise (subscript indicates source) |
| $N$ | vector of n (subscript indicates source) |
| $N_s$ | vector of quantistion noise from SHL algorithm |
| $n_a$ | quantisation noise from either a-to-d or d-to-a conversion |
| $n_{a\_d}$ | quantisation noise from a-d converter |
| $n_{d\_a}$ | quantisation noise from d-a converter |
| $n_p$ | noise present in reference channel |
| $n_t$ | quantisation noise from finite precision in convolution |
| $n_r$ | quantisation noise from truncation of hardware-filter output |
| $n_b$ | quantisation noise from truncation of $2\mu e$ bus |
| $n_h$ | quantisation noise from truncation of $\Delta h$ bus |
| NF | upper bound on number of feedforward coefficients |
| NB | number of feedback coefficients |
| $P$ | correlation matrix |

| | |
|---|---|
| $q$ | scaler for $\Delta_h$ |
| $Q$ | coefficient related to q |
| $q_i, q_o$ | signals |
| $Q_i, Q_o$ | spectrum of signals above |
| $r$ | replica (output of filter) |
| $r$ | complex frequency in Chapter 6 |
| $\mathbf{R}$ | vector of r |
| $R$ | spectrum of $\mathbf{R}$ |
| $\rho$ | coefficient vector for recursive filter |
| $\sigma$ | standard deviation (subscript indicates source, see n) |
| $\sigma_q$ | relates to noise in filter from finite precision of $\mathbf{H}$ |
| $\sigma_f$, $\sigma_v$ | relates to cross-correlation noises in filter |
| $S_a$ | LST to LST loop gain |
| $S_l$ | Local LST loop gain |
| $S_t$ | Sidtone in dB |
| $t_1$ | scaler for normalisation of $2\mu e$ bus |
| $t_2$ | scaler for normalisation of $\Delta$ bus |
| $T_{hl}$ | Trans-hybrid loss |
| $u$ | noise component in hardware filter |
| $\mu$ | scaler for update (correlation) term |
| $\mu_f$ | reference to feedforward coefficients |
| $\mu_b$ | reference to feedback coefficients |
| $v$ | noise component in hardware filter |
| $v_r, v_t$ | variable gains in LST loop |
| $V_r, V_t$ | log of above |
| $W$ | number of coefficients in impulse response |
| $x$ | input signal |
| $\mathbf{X}$ | vector of x |
| $y$ | output signal from reference channel |
| $z$ | complex time in Chapter 6 |

|  |  |
|---|---|
| Z | term used for brevity |
| $\eta$ | update component (in recursive filter) |

N.B. All bold type characters refer to vectors of terms in normal print.

## OPERATORS.

|  |  |
|---|---|
| $*$ | convolution |
| E | expectation |
| $\nabla$ | differentiation w.r.t subscript |
| trunc | truncation of binary word |
| $\rightarrow$ | tends towards |

## ABBREVIATIONS

|  |  |
|---|---|
| ARMA | autoregressive moving-average |
| apx | approximate |
| DAF | digital adaptive-filter |
| FIR | finite impulse-response |
| $f(2\mu)$ | function of update coefficient |
| $f(2\mu e)$ | function of update coefficient times error |
| IIR | infinite impulse-response |
| LMS | least mean-square |
| LSI | large-scale integration |
| LST | Loudspeaker telephone |
| opt | optimum |
| SER | sequential regression |
| SHL | shift-left algorithm |
| SPL | Sound pressure level |
| SSX | sum-of-squares of x |

# 1. INTRODUCTION.

## 1.1 Background.

In an ideal situation a loudspeaking telephone (LST) would recreate the quality of a conversation held between people, equal to that held in a room where the participants are a normal distance apart. If the communication medium introduces degradations, then those participating in the conversation will experience difficulties, or discomfort and will require increased concentration and listening effort. The main degradations applying to current LSTs are twofold. First, there is the impairment commonly known as the barrel (or bathroom) effect, where speech picked up by a single microphone appears hollow and blurred. This is caused by the talker being a distance from the microphone comparable to the transmitted wavelength (which is approximately 1m at 300 Hz and .1m at 3 kHz), in an environment containing reflecting surfaces. Complete cancellation takes place when transmitted and reflected waves are 180° out of phase with each other. Examination of the frequency domain of a transmitted signal, measured some distance form the source, shows a series of holes in the spectrum [101], which give the characteristic sound of someone speaking from within a barrel. For the binaural listener, the brain is extremely effective in removing the degradation caused by reverberation. The effect can be observed by closing off one ear during a conversation. Figure 1.1 shows two measurements of sound pressure level (SPL) for a loudspeaker 500 mm away from an LST on a table in an anechoic chamber. Plot 2 shows the frequency distribution at a position just in front of the loudspeaker and plot 1 shows the resultant distribution as detected by the LST microphone. There is a deep notch at 5 kHz due to wave cancellation. The microphone in both measurements was the same and the attenuation over the 150 Hz to 4 kHz band was 18 dB.

Techniques for real-time signal processing to reduce the barrel effect have been examined for both the time and frequency domains. In the former the

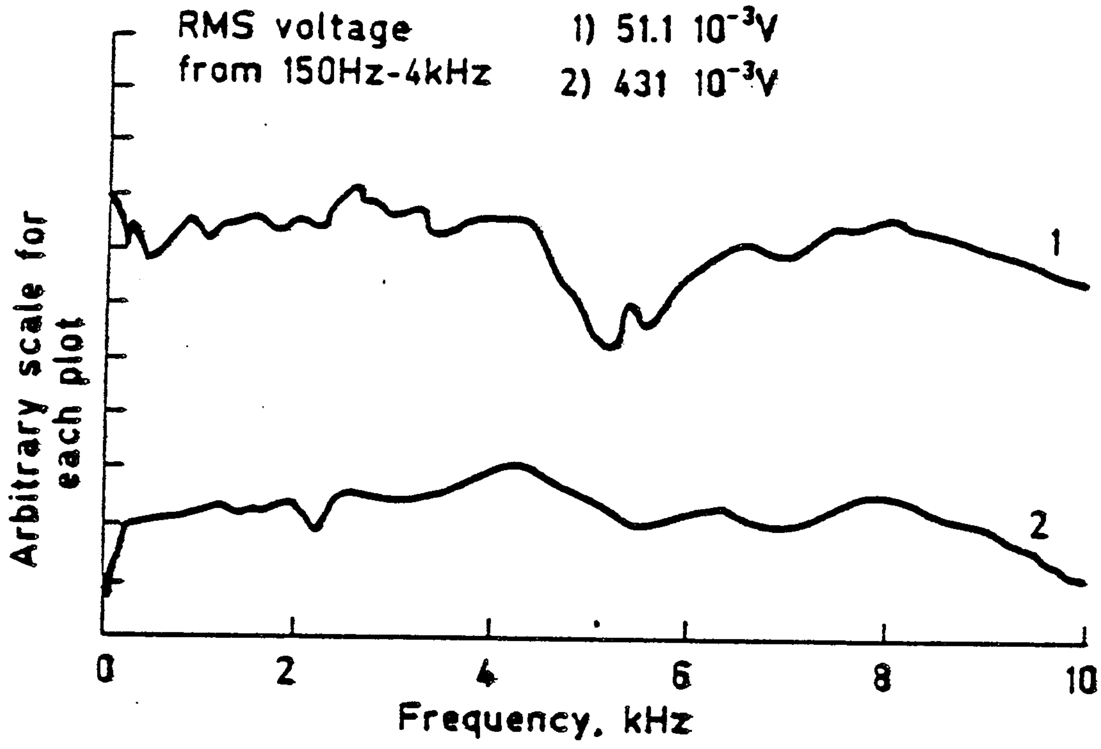


FIG. 1.1 SPL AT 1) LST MICROPHONE POSITION
2) ARTIFICIAL MOUTH POSITION

most effective technique employs a nonlinear process called 'centre-clipping' [102]. It is based on the fact that the reverberant tail of the signal is low level, thus, a threshold is set below which the signal is forced to zero. The disadvantage is that distortion is introduced. With high-speed digital signal processing it would be possible to apply an interpolating formula to provide smoothing for signals within the threshold and reduce the observed distortion.

A multi-microphone technique was assessed [103] which could treat early echoes. Since the microphones are at different positions, early echoes arrived at different times at each of the microphones and consequently the resulting nulls are at different frequencies. The input to each microphone is passed through a bank of band-pass filters covering the spectrum. The energy in each filter is measured and that with the maximum is transmitted. Considerable improvement was found for two microphones in small rooms, increasing the number of microphones did not provide any significant improvement.

The second major degradation associated with LSTs comes from the processing required to provide sufficient gain to recreate speech-levels which would match conversation conditions and maintain a stable system. It is this problem which is addressed by the following work.

## 1.2 The stability problem.

Consider the situation where talker A is d metres from his microphone, then the sound pressure level (SPL) at the microphone is the SPL to be produced at the ear of the listener B. While this statement simplifies analysis, it has been shown [104] that people have a preferred listening-level of approximately 80 dBa (SPL) at a distance of 1 m from the loudspeaker. Consequently, if any gain adjustment is provided, users will adjust both for line loss and for variation of talker level.

For the case of two identical LSTs connected to each other as shown in figure 1.2. Talkers A and B, defined at a reference position, are at a distance $d_2$ from their LSTs with a corresponding acoustic-attenuation of $A_1$ dB. The distance between loudspeaker and microphone is $d_1$ with an acoustic attenuation of $A_m$ dB. Speaker A produces x dBa at his microphone, which is to be reproduced at the reference position of listener B. The line loss is L dB and 3.5 dB loss is assumed through each hybrid (2-to-4 wire converter). There is coupling across each hybrid giving a trans-hybrid loss of $T_{h1}$ dB. The gain G required by each LST is the sum of losses from microphone to ear.

$$G = 7 + L + A_1 \quad \text{dB} \qquad \ldots (1.1)$$

NB there is a transduction term of Volts/Pascal associated with the loudspeaker and microphone. It is assumed that normalizing gains are applied to each to make them equal and, therefore, removing them from the equation.

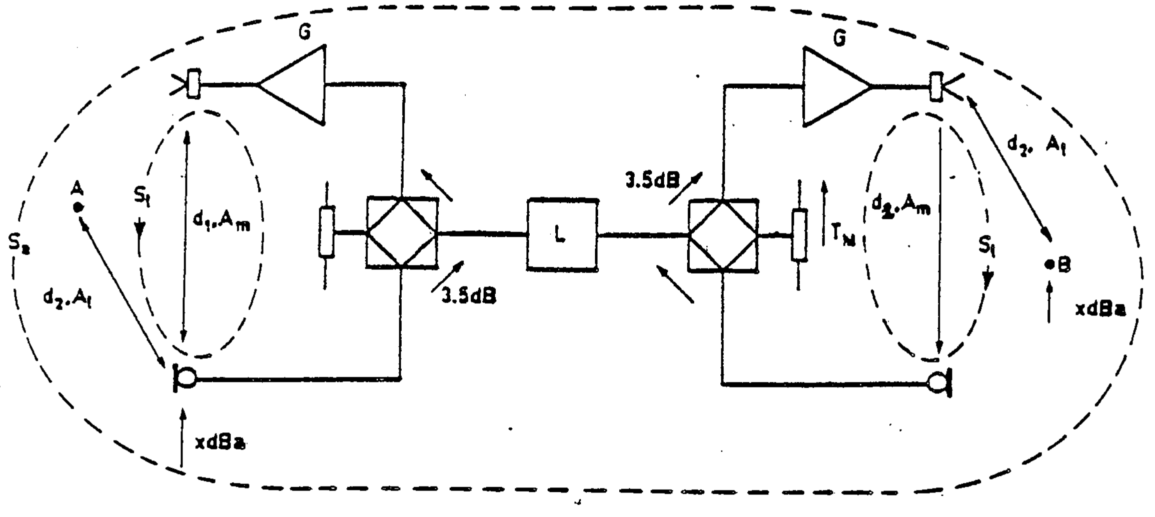


FIG. 1.2 LST TO LST CONNECTION

There are three closed loops due to coupling. One is an end-to-end loop caused by acoustic coupling between each loudspeaker and microphone, the other two are local loops with coupling around the hybrid and acoustic path.

Consider the loop gain $S_a$ from end to end.

$$S_a = -7 - L + G - A_m - 7 - L + G - A_m \quad \text{dB} \qquad \dots (1.2)$$

Substituting equation (1.1)

$$S_a = 2(A_1 - A_m) \quad \text{dB} \qquad \dots (1.3)$$

For stability $S_a < 0$ dB, therefore

$$(A_1 - A_m) < 0 \text{ dB} \qquad \dots (1.4)$$

i.e. the loss between loudspeaker and microphone must be greater than that from the talker to microphone.

The inverse-square law does not apply to a listener (or talker) sitting at a table in a small room for distances greater than 1 metre from the loudspeaker (or microphone) [104]. Figure 1.3 is derived from [104] and shows three SPL measurements, made at a point 25 mm in front of the subject's mouth, together with the inverse-square law. NB the reference level for dBa is $2^{-5}$ N/m², 0 dBPa = 94 dBa. The reference distance is from a plane defined to be 25 mm from an equivalent point-source sound and is called the 'lip ring' position [105], conforming with CCITT recommendations on speech measurements [106]. The attenuation is approximately 28 dB for the region of 800 mm to 1250 mm from the lip ring. With due cognisance of the difficulty in determining $A_1$, measurements were made of the $(A_1 - A_m)$ relation for 4 LSTs [107] in an anechoic chamber, for distance $d_2$ = 500 mm. The results are reproduced in Table 1.1.

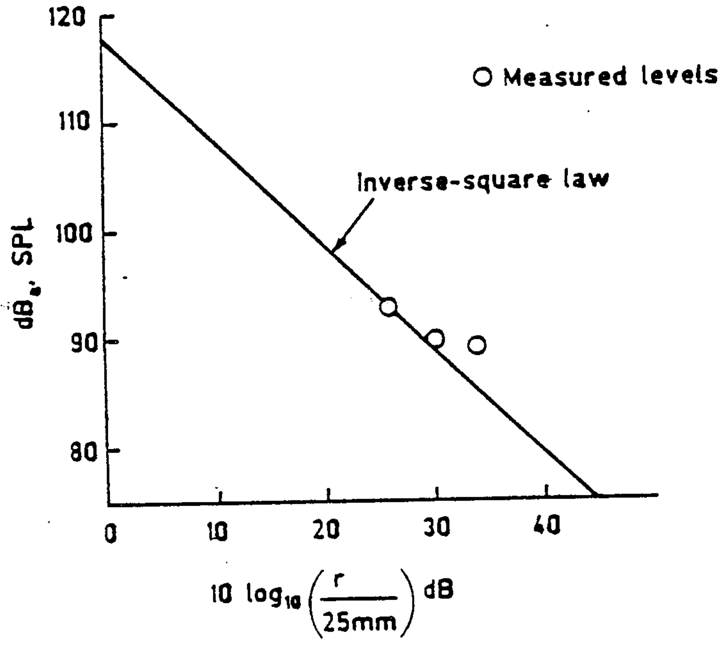


FIG 1.3 MEASURED SPL AT LISTENERS LIP-RING AGAINST DISTANCE rmm FROM THE LIP-RING OF A LOUDSPEAKER ON A TABLE

| LST No. | $A_1-A_m$ dB |
|---|---|
| 1 | 6.5 |
| 2 | 4.6 |
| 3 | 16.9 |
| 4 | 0.0 |

Table 1.1 $A_1-A_m$ relationship for 4 LSTs with $d_2$ = 500 mm.

LSTs 1 and 3 were commercially-produced items, while LST 2 was a version of LST 1 with a repositioned microphone. LST 4 was made for the project and incorporated some acoustic treatment (e.g. damping of case and transducer mounting, etc.). Thus, it is possible to approach an end-to-end loop gain of unity with a carefully designed case but, this does not comply with the requirement of equation (1.4).

Similarly for the local loop, the gain is

$$S_1 = -T_{h1} + G - A_m \quad \text{dB} \qquad \ldots (1.5)$$

Substituting equation(1.1)

$$S_1 = (7 + L) - T_{h1} + (A_1 - A_m) \quad \text{dB} \qquad \ldots (1.6)$$

Assuming that $(A_1 - A_m)$ is $> 0$ dB then the only reduction of $S_1$ will come from the trans-hybrid loss, which could vary from 6 dB to 18 dB.

Figure 1.4 shows the cumulative probability of having a connection with attenuation less than that given on the abscissa [108]. The value is dependent on the percentage of calls made only through the local exchange. It shows that the worst-case line loss is of the order of 30 dB, that 20% of calls would lie in the range 20 dB to 30 dB and 30% of calls in the range 15 dB to 20 dB. Substituting for best and worst case values of

| | |
|---|---|
| line loss | 0 to 30 dB |
| $T_{h1}$ | 6 to 18 dB |

and $A_1 - A_m = 0$ dB, then $S_1$ lies in the range

$$-11 \text{ dB} < S_1 < 31 \text{ dB}$$

Thus, more than 31 dB of suppression is required in the local loop to maintain stability for all calls over the BT network.

## 1.3 Current solutions.

The only solution, of which the author is aware, that is available to the public, relies on exploiting the general unidirectional properties of speech. Stability is achieved by inserting an attenuator into one of the transmission

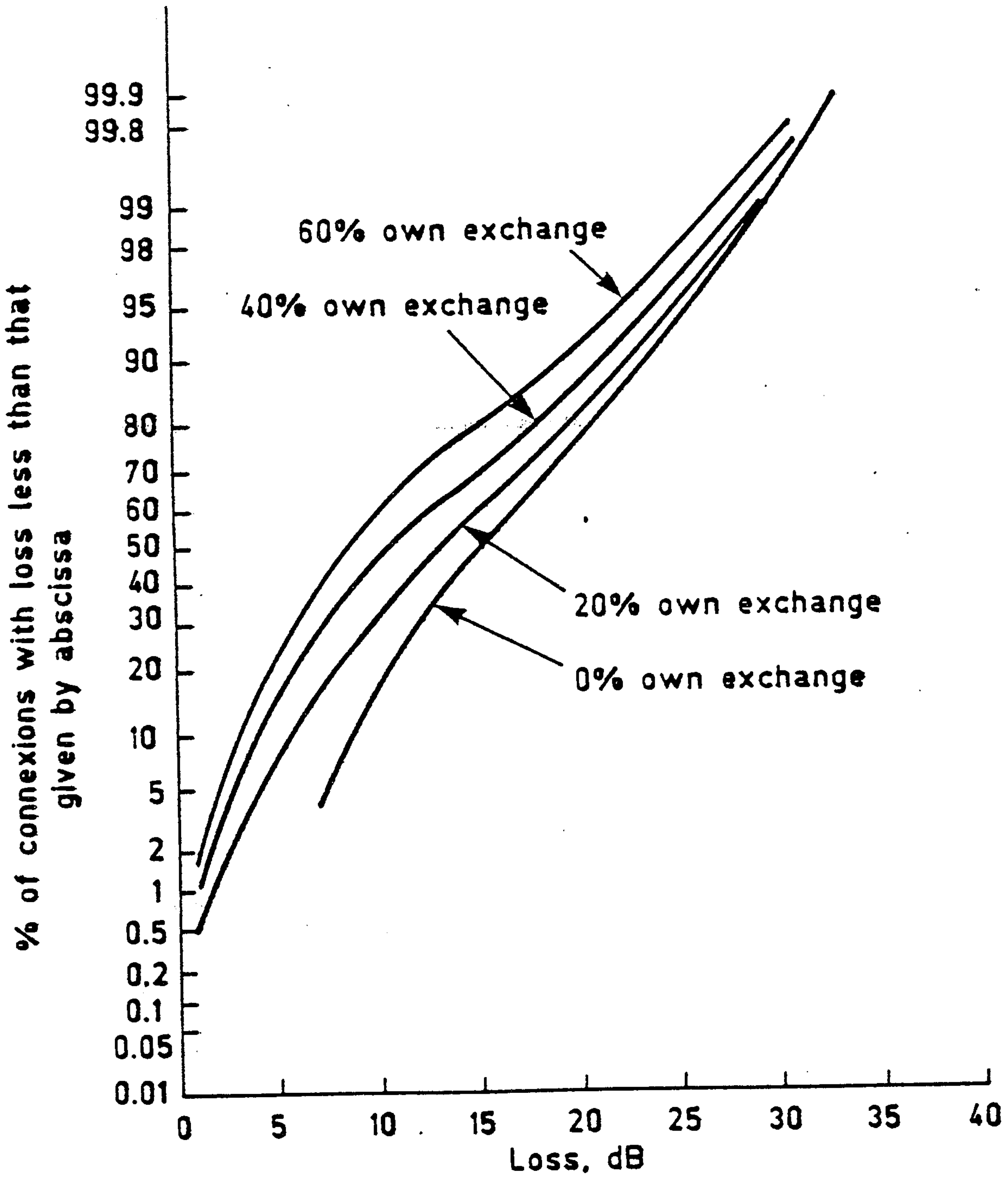


FIG. 1.4 DISTRIBUTIONS OF LOSS FOR VARYING PROPORTIONS OF OWN EXCHANGE TRAFFIC

paths, effectively giving simplex operation. An early proposal that the attenuator be activated from some measurement of speech was by Mc Millan and Doust in 1933 [109]. Their LST operated over a 4-wire circuit and used gas filled 'trigger' valves to turn on or off either the transmit or receive amplifiers depending on a balance network determining the direction of transmission. Soon after this the technique was applied to 2-wire amplifiers [110,111] and in 1935 L.H. Paddle [112] applied the technique to LSTs operating over 2-wire circuits. The principle has become known as voice switching. Since that time there have been many publications and patents relating to methods of attempting to discriminate between transmit and receive so that the attenuator can be transferred appropriately [113 - 126].

Early types of LST employed a bang-bang principle to switch the attenuator [118], later versions used a technique to ramp attenuation into one path and out of the other, giving a subjective improvement [127]. Changeover from transmit to receive requires a 'hangover period', to prevent the LST mis-operating by going into receive on the reverberant tail of the transmitted signal. This causes one of the most serious degradations, which is initial-syllable clipping, resulting in confusion as the natural response is to interject and ask for clarification. Apart from not hearing interruptions, the other significant degradation is perceived as the unpleasant feeling of listening to a 'live' (reverberant) environment, but talking to a 'dead' one. The effect is exacerbated by the apparent rise and fall of the received level of noise generated in the distant room.

A. Rickaby [128] investigated duplex transmission over a 2-wire link by various techniques which proved impracticable (or impossible). e.g. perfect balancing of the hybrid, acoustic-wave cancellation with additional loudspeakers, directionality of transducers. One interesting method, called band elimination, which was claimed to give good results, applied two interlaced comb-filters to the transmit and receive paths as shown in figure

1.5. This proved impracticable a) because it would only work with a similar LST, and b) because of the cost of building the filters. The result of this work was the A.T.E Type 49 loudspeaking telephone, which provided a measure of duplex working by using a variable depth voice-switch and separation of the loudspeaker and microphone to control the loop stability. The major disadvantage reported was low volume.

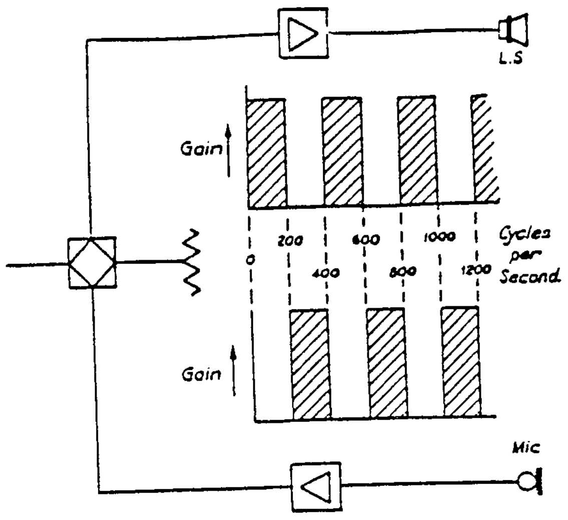


Figure 1.5 Elimination of howling by restricting the microphone and loudspeaker amplifiers to alternate frequency bands. As the bands are complementary there is no risk of feed-back.

Two techniques have been reported which control the loop stability by gain regulation methods. In [129] the gain of the loudspeaker amplifier is regulated by a circuit which compares the signal levels in send and receive paths, such that the modulus of gain in the electro-acoustic loop is always less than unity. A novel method is described in [130] where the loop is forced to oscillate just beyond the upper frequency using a narrow notch (ceramic) filter. There is little energy at this frequency and it is audible only if the ear is placed close to the loudspeaker. The loop gain is measured at the resonant frequency and used with a scaling factor to control the loop gain in the telephony band. Both techniques suffer from low acoustic output.

## 1.4 New Design Philosophy.

The first significant step toward a fully-duplex LST which came to the notice of the author was the proposal to use two adaptive filters, modelling the impulse response of the acoustic and hybrid coupling paths [131,132], which is reproduced in figure 1.6. Consider a signal y from the loudspeaker passing through the acoustic medium and arriving at the microphone, modified to $y_a$. The same signal is passed through the adaptive filter to produce an estimate of $y_a$. The two are subtracted and the difference (or error) signal is transmitted to line, thus, cancelling the received signal from the transmit path. The parameters of the filter, which are usually, although not necessarily, the impulse-response coefficients, are adjusted during a receive condition to drive the difference signal toward zero. Obviously, some method is needed to prevent the filter from altering its parameters during a period of double-talk. Exactly the same discussion applies to signals coupling across

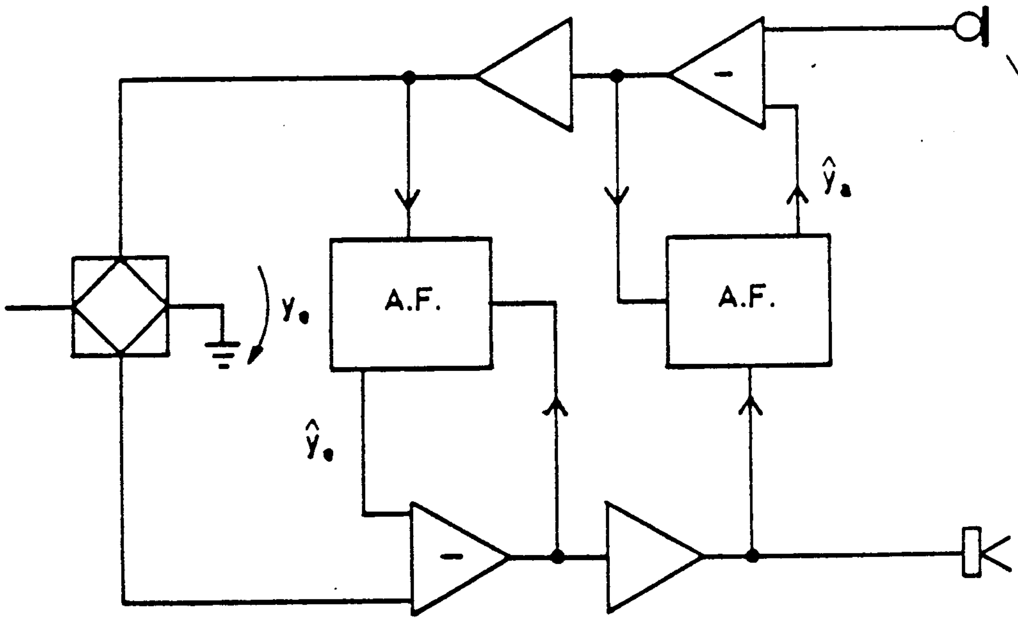


FIG. 1.6 ADAPTIVE FILTERS MODELLING ACOUSTIC AND HYBID PATHS (from [131] )

the hybrid path due to mismatch of the balance impedance. Thus, assuming perfect cancellation, the transmit and receive paths are isolated from each other and full duplex transmission can take place. It is interesting to note that in 1974 Berkley [101] gave a verbal description of the technique.

Even if perfect adaptive-filters with infinite impulse-responses were practical, the method as described in [131,132] can be shown to be impractical on two counts associated with the non-stationary characteristic of the acoustic path. These are loop stability and sidetone as perceived by the distant talker. All adaptive filters require a finite time to learn an impulse response and will have a lag associated with a non-stationary path. Non-stationarity is mainly due to random movement of the human body, which represents a significant mass in the acoustic path. Slight changes in body position, by just a few centimetres, will completely destroy the adaptation achieved. If movement occurs while the distant user is talking, there would be a lag in modelling the acoustic path, resulting in either high sidetone to the talker or instability, depending on the loss in the transmission path. For movement by the near end user while speaking or during silence, the result would be instability or high sidetone when the distant user began talking. The problem is compounded by limitations in terms of length of impulse response for practical adaptive-filters, compared with that required to model a room. The net result is poor modelling performance of the acoustic path.

To overcome this problem, Berkley [101] proposed using a centre-clipper with variable threshold in a number of band-pass filters (to reduce the inherent distortion). While such a technique might prove advantageous for a stationary path, it does not cover the non-stationary case. The author et al. proposed a gain-regulation solution [133], described in figure 1.7. In this implementation a microprocessor (known as the controller) is used to measure the loop gain continuously and adjusts the value of one attenuator in the nominally quiet direction to maintain stability in the local loop. It also controls which adaptive filter is fixed and which is adapting, depending on the direction of speech. The technique has the advantage of offering the best solution dependent on the variable parameters. For example, at the beginning of a call the filters would be set to zero and the system would start up as fully voice-switched. After adapting to the impulse responses of the hybrid and acoustic path, attenuation is removed from the listening direction. The LST thus becomes duplex and one talker can hear the other person during double-talk.

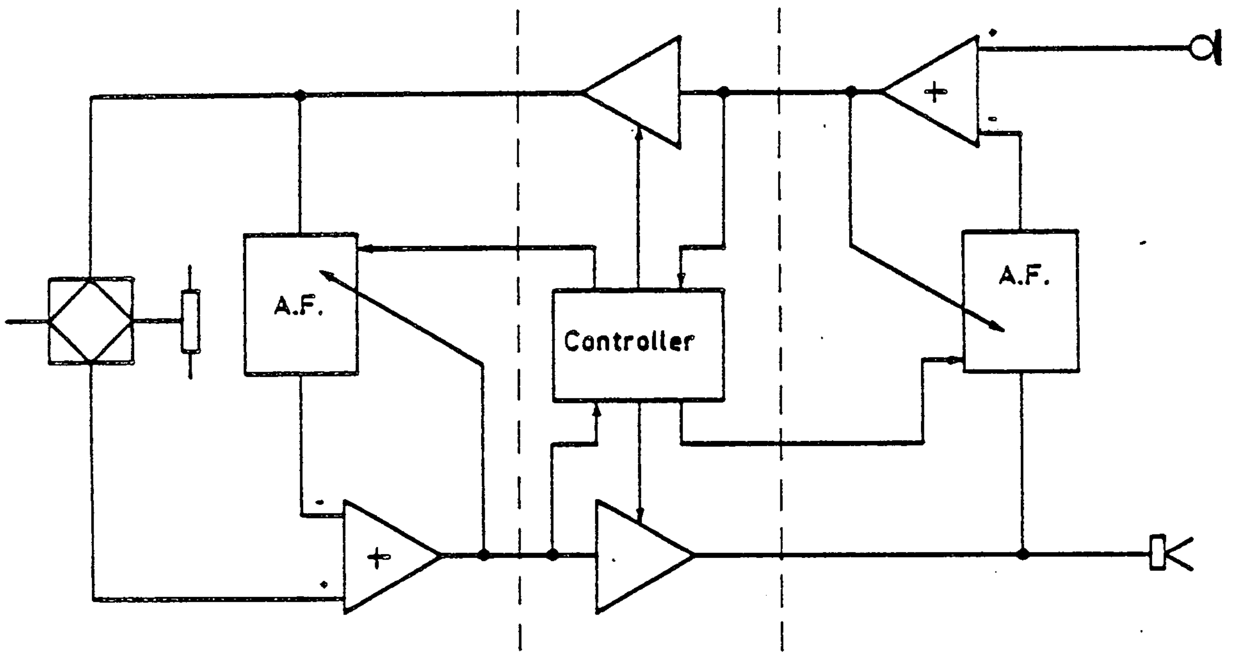


FIG. 1.7 LST WITH ADAPTIVE FILTERS AND GAIN REGULATION VIA A CONTROLLER

A single LST, using the technique described, was built to test the feasibility of the system. It proved to be successful and an engineered version was required. The thesis describes the work carried out to achieve a suitable adaptive-filter, also the controller as implemented in the feasibility study. The former is covered in Chapters 2 to 5, for theoretical performance, design, implementation in LSI and measured performance. Chapter 6 deals with the theoretical stability and sidtone performance of an adaptive LST, while Chapter 7 describes an experimental duplex LST.

# 2. THE ADAPTIVE FILTER.

## 2.1 Discussion of requirements.

Since the adaptive filter would require a considerable financial investment by British Telecommunications and was primarily intended to be incorporated into a loudspeaking telephone to be sold in the retail market, it was necessary for the final device to be bound by certain constraints.

i) Low cost.

The cost need not necessarily be the minimum, but one which the instrument could support in a competitive market.

ii) Implementation in LSI.

Because of system complexity and a low-cost requirement, it was essential that the filter be realized in LSI.

iii) Maximum functional utilization.

There is a limited number of hardware multipliers, dividers and adders, all requiring numerical overflow detection and correction, which can be included in a low-cost LSI solution. Bus interconnection can be used to multiplex data between functional operations, introducing time constraints within the sample period (nominally 125 μs for the telephone bandwidth).

iv) Maximum speed of adaptation.

An adaptive filter modelling the telephone network will, in general, see a stable impedance (assuming no phase roll [201]) and could have an arbitrary adaptation speed. However, the acoustic path is unstable: small movements of the human body can produce significant changes in the impulse response, possibly occurring during a silence or transmit condition when no signal is available to train the filter. To hold the LST stable with minimum inserted attenuation, it is necessary to track changes as quickly as possible while a signal is received.

v) Adaptation in the presence of continuous noise.

The telephone network produces noise which can be regarded as continuous relative to adaptation times, e.g. quantisation, cross-talk, thermal etc. [202] as well as impulsive noise e.g. switch chatter. The acoustic path suffers particularly from noise (due to external wind noise on buildings, ventilation, machinery, passing vehicles, etc.) which can be considered as continuous, as well as impulsive noise (from slow-speed printers or furniture movement, etc.).

vi) Wide range of impulse-response lengths.

Within the national network the impulse responses of end-to-end connections have been measured [203]. For the majority of circuits they lie in the range 0 to 30ms, and occasionally some circuits of up to 50ms have been found. This wide range is due to band-pass filter roll-off characteristics. Rooms typically have reverberation times of 500ms to 1500ms [204] depending on volume, construction and furnishing materials. This measure is defined as the time taken for energy to fall 60dB from its original value and approximates the time the ear perceives reverberation after cessation of a loud source in a quiet room. An approximation is given by :

$$T = 0.163 \frac{V}{-A.\ln(1-\alpha)} \qquad \ldots (2.1)$$

Where 'V' is the volume, 'A' the total surface area and 'α' is the absorption coefficient. Measurements show that the reverberation time is a function of the frequency band used.

vii) Adaptation on Speech.

While it is possible to use a suitable training signal for an adaptive filter to model a stable dispersive medium, in a LST environment the only signal available is transmitted or received speech plus local noise.

viii) Stability.

The filter must be stable within the arithmetic precision available from hardware functions, constrained by (ii) and (iii) above.

In relation to the above conditions, two adaptive filter architectures are considered. The first is a recursive filter with infinite impulse response (IIR), based on the autoregressive moving-average (ARMA) model, and the second is a transversal filter having a finite impulse response (FIR).

## 2.2 Recursive IIR Adaptive Filter.

### 2.2.1 Architectures.

Initial consideration, of length of impulse response and low processing overhead, may suggest that this architecture would offer the optimum solution for an adaptive filter (since the acoustic path produces a very long impulse response). However, on further examination (which will be considered in the remainder of this section) this structure may not prove so attractive.

The order of an IIR filter must be compared with that of the path to be modelled. It is probable that the order of such a practical filter would be in the range 10 to 100, this would constitute the number of poles available for modelling. Whereas the number of modes supported in a typical room in the 0 to 4kHz band would number thousands. An approximation for the number of normal modes [204] in a large room is given by :

$$N = \frac{4}{3}.\pi.f_{lim}^{3}.\frac{V}{c^{3}} \qquad \ldots (2.2)$$

where 'V' is the volume, 'f' the frequency band and 'c' the velocity of sound.

In consequence, a practical filter must converge under the condition that the number of poles available would be considerably less than the number of poles in the path being modelled.

The following is a synopsis of the modern work published to date on this architecture, with some extensions and observations on the problem of maintaining stability.

Appendix A is a reproduction of the work of S. A. White [205], who proposed an adaptive recursive filter based on the autoregressive moving-average (ARMA) structure. Applying the method of steepest descent (a gradient search technique) and using the instantaneous error as the parameter for convergence, White derived equations for updating the feedforward and feedback coefficients for a response modelling connection as shown in figure (A.1). In this work an arbitrary function of the error criterion to be minimised was proposed, i.e Fn = F(e) (equation (A.4)), the derivative of this function with respect to error required for the descent method can then be found. e.g. taking only the sign of the error has a derivative of the modulus of error. To solve the coefficient update algorithm exactly (equation (A.8)) requires the substitution of equation (A.18). A hardware scheme was proposed (reproduced in figure A.2), which provided the basis of an IIR adaptive filter. Since the forward and backward coefficients plus past input and output signal values must be stored within the system to compute the replica (i.e. the output of the filter), this solution has no additional storage penalty. No work was reported on stability of the scheme.

Stearns and Elliot [206] reported a computer simulation of White's algorithm for a two-pole modelling experiment, using arbitrary values for $\mu$ (a scaler for the gradient-search technique which is bounded to ensure stability). Experimenting with various numbers of forward and backward coefficients, it was suggested that the stability of the system improved as the number of coefficients was increased.

Feintuch [207] proposed an approximation to White's algorithm, reducing equation (A.18) to the first term only, ignoring the recursive component of the gradient term. It was shown that, with arbitrary (not reported) values for $\mu$ the system converged via computer simulation, for a correlated signal-extraction experiment. A discussion followed in the literature [208,209]. The general consensus was that the approximation was important, although doubts were cast as to the validity of derivation. Also, that the performance was not amenable to analysis and no conditions for stability had been established.

A relevant observation in the context of practically modelling an acoustic path showed [208] that when Feintuch's approximation was applied to a plant (path being modelled) with order greater than the filter, the algorithm either failed to converge, or converged to a local minimum on the mean-square-error surface, depending on the initial conditions.

The point was taken up by Parikh and Ahmed [210], who showed that White's algorithm did converge to the nearest local minimum, depending on initial values of filter coefficients. They further developed [211] a sequential regression algorithm (SER), which was demonstrated by simulation to converge to the least-mean-squares-error condition, using a second-order plant and a first-order filter, irrespective of the initial conditions. Their contribution was to change the criterion to be minimised from the least-mean-squares of error to one including the energy in the coefficient vector $\rho$ (Newton's Method).

i.e. introducing a function to be minimised :

$$J(\rho) = \mu \sum_{i}(y(i) - r(i))^2 + \rho^T\rho \qquad \ldots (2.3)$$

the resulting recurrence formula was :

$$\rho_k = \rho_{k-1} + \mu\, P^{-1}\, e(k)\, \nabla_\rho(e) \qquad \ldots (2.4)$$

This is the descent method, but includes a correlation matrix P, the inverse being solved recursively by :

$$P_k^{-1} = P_{k-1}^{-1} - \frac{1}{\gamma} P_{k-1}^{-1}\, \lambda_k\, \beta_k^T\, P_{k-1}^{-1} \qquad \ldots (2.5)$$

where

$$\gamma = \frac{1}{\mu} + \beta_k^T\, P_{k-1}^{-1}\, \lambda_k \qquad \ldots (2.6)$$

is a scaler and $\lambda$ is given by Equation (A.18).

There is an increase in the complexity of solution over White's algorithm, i.e. if the total number of forward (NF + 1) and backward (NB) coefficients is N = (NF + NB + 1) then $N^2$ additional storage elements are required, plus $2(N^2 + N)$ multiply / accumulates, $N^2$ additions and (more difficult) $N^2$ divisions, to solve $P^{-1}$.

To improve the speed of adaptation David [212,213] proposed a normalised RLMS algorithm which provided a variable scaler to $\mu$ the diagonal matrix controlling the magnitude of the update component (equation (A.9))

$$\mu_{NRLMS} = \frac{1}{\beta^T\beta}\, \mu \qquad \ldots (2.7)$$

where $\beta$ is the vector of input/output values equation (A.6). The quotient is directly related to the rate of change of the performance surface. In [214] David also compared all the algorithms for speed of adaptation, using a reference path modelling experiment, with plant and filter of the same order. An architecture called the 'equation error' in which 'y' (the output from the reference path) replaces 'r' (the filter output) [215] was also tested. Proving to have very fast convergence, the algorithm is inherently unstable. This is because any noise in the reference channel directly alters filter coefficients without restriction and could cause poles to leave the unit circle. Thus, it is not considered further because of the high noise content of 'y' in LST environment. For a simple reference path, it was shown that the

SER algorithm demonstrated the best performance, followed by the NRLMS algorithm. The paper also showed the progress of coefficient changes toward the least-mean-square error as the algorithms adapted. Of the five single-run learning curves shown, two came perilously close to instability, i.e poles moving toward the edge of the unit circle.

## 2.2.2 Stability.

Analytic determination of conditions for stability is very difficult since the filter is nonlinear and generally a function of several criteria. McMurray [216] derived some simple stability criteria, based on a frozen-time viewpoint, for Feituch's algorithm. i.e. instead of using statistical analysis and requiring that on average some function of the error must reduce, he chose a single cycle during convergence and required that the error squared produced in the same cycle, but using the updated coefficient vector, should be reduced.

Using the nomenclature of Appendix A and applying a general solution, let $\tilde{e}_n$ be the error within cycle 'n' calculated from the updated coefficient vector $\rho(n+1)$ equation (A.8), then the stability criterion is expressed :

$$e_n^2 >= \tilde{e}_n^2 \qquad \ldots (2.8)$$

Substituting the desired signal (the output of the reference channel) 'y(n)' and modified replica '$\tilde{r}(n)$' on the r.h.s. :

$$e_n^2 >= (y_n - \tilde{r}_n)^2 \qquad \ldots (2.9)$$

Let the function of error to be minimised be the instantaneous error squared, then equation (A.8) becomes

$$\rho_{n+1} = \rho_n + 2.e_n.\mu.\nabla_\rho(e_n) \qquad \ldots (2.10)$$

Substituting equation (2.10) into (2.9) then :

$$e_n^2 >= (y_n - (\rho_n + 2.e_n.\mu.\nabla_\rho(e_n))^T\beta)^2 \qquad \ldots (2.11)$$

Let

$$\eta = 2.\mu.\nabla_{\rho}(e_n)^T\beta \qquad \ldots (2.12)$$

then equation (2.11) reduces to :

$$1 >= (1 - \eta)^2 \qquad \ldots (2.13)$$

therefore

$$0 < \eta < 2 \qquad \ldots (2.14)$$

Equation (2.14) is a general stability condition which must be obeyed each cycle. The descent algorithm can be substituted to determine the complexity of arithmetic required to maintain stability. Consider Feintuch's algorithm (as in [216] by McMurry), then :

$$\eta = 2.\mu.\beta^T\beta \qquad \ldots (2.15)$$

therefore the diagonal matrix $\mu$ must be bounded by

$$0 < \mu < -\frac{1}{\beta^T\beta} \qquad \ldots (2.16)$$

N.B. equation (2.7) can be found from the lower bound. It is assumed that $\mu$ consists of two values; $\mu_f$ which is applied to the feedforward coefficients and $\mu_b$ applied to the feedback coefficients. Substituting for $\mu$ and expanding equation (2.15) gives the main stability criterion at sample 'n' :

$$\eta = 2.\mu_f.\sum_{k=0}^{NF} x^2(n - k) + 2.\mu_b.\sum_{m=1}^{NB} r^2(n - m) \qquad \ldots (2.17)$$

also

$$\mu_f, \mu_b > 0 \qquad \ldots (2.18)$$

which constrains equation (2.17) to be always positive. This produces a stability triangle with co-ordinates $\mu_f$ and $\mu_b$ i.e.

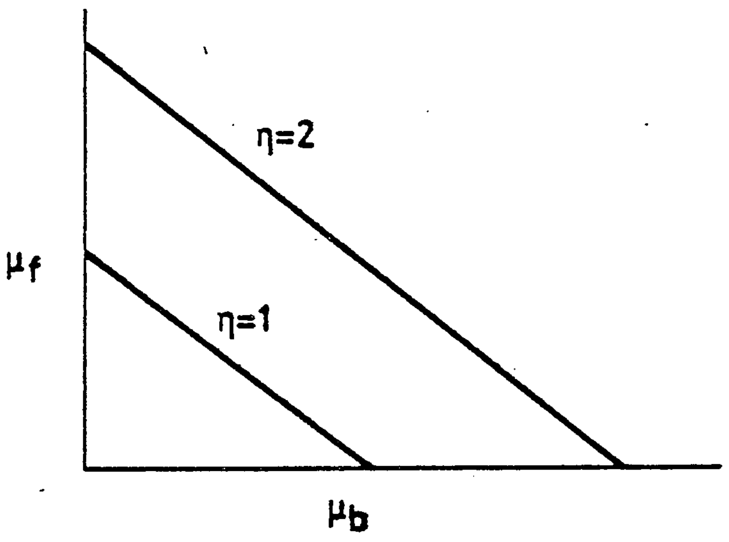


FIG. 2.1 STABILITY TRIANGLE

As the order of the filter is increased by increasing the number of forward (NF) and backward (NB) coefficients, the area within the triangle is reduced. For experimental evidence, McMurray applied his stability criterion to Feintuch's algorithm for a variety of $\mu_f$ and $\mu_b$, but using an estimate for the sum of squares of 'x' and 'r'. (It should be emphasised that by using an estimate of these parameters the condition required for stability was not obeyed). It was shown using Monte Carlo simulation on a fourth-order filter and second-order plant, that curves could be found for the probability of the filter becoming unstable. A loose translation of the results required that $0 < \eta < 1$ for the filter to have a 90% probability of being stable.

The same stability criterion can be applied in general to the gradient-search method :

$$\tilde{e}_n = y_n - (\rho_n + 2.e_n.\mu.\nabla_\rho(e_n))^T\beta \qquad \ldots (2.19)$$

$$\tilde{e}_n = e_n.(1 - 2.\mu.\nabla_\rho(e_n)^T\beta) \qquad \ldots (2.20)$$

$$\tilde{e}^2 <= e_n{}^2.(1 - \eta)^2 \qquad \ldots (2.21)$$

The squared term in parenthesis must be less than unity, therefore :

$$0 < \eta < 2 \qquad \ldots (2.22)$$

Each of the algorithms can be substituted. For White [205] -

$$\eta = -2.\mu.(\beta + (B^T\acute{R})^T\beta \qquad \ldots (2.23)$$

Where $B^T R'$ is a vector, derived from

$$B^T = [b_1, b_2, \ldots b_{NB}] \qquad \ldots (2.24)$$

is the vector of feedback coefficients, and

$$\acute{R}^T = \frac{\partial}{\partial e}[r_{n-1}, r_{n-2} \ldots r_{n-NB}] \qquad \ldots (2.25)$$

is the vector of differentiated filter output values. Applying values for the diagonal matrix $\mu$ of $\mu_f$ and $\mu_b$ in equation (2.23)

$$\eta = -2.\mu_f.(X^T X + (B^T\acute{R})^T X) - 2.\mu_b.(R^T R + B^T\acute{R})^T R \qquad \ldots (2.26)$$

To generate a positive $\eta$ it is insufficient to set $\mu_f$ and $\mu_b$ negative; a cycle-by-cycle determination of the terms in parenthesis is required to ensure they are positive. Similarly, for the SER algorithm :

$$\eta = -2.\mu(P^{-1}(\beta + (B^T\acute{R})^T\beta \qquad \ldots (2.27)$$

It can be seen that using this technique would substantially increase the processing required each cycle.

An algorithm was proposed by Treichler [217], called the simple hyperstable adaptive recursive filter (SHARF), which was heuristically derived based on the principle of hyperstability [218]. This is a time-domain technique requiring that the inputs to both the feedforward and feedback paths result in outputs of the same sign. For this case the denominator of the system transfer function (equation (A.2)) can never be zero; therefore the system is always stable. The resulting update for the jth forward and backward coefficients at sample instant 'n' was :

$$a_{j,n+1} = a_{j,n} + \mu_f.\xi_n.r_{n-j} \qquad \ldots (2.28)$$

$$b_{j,n+1} = b_{j,n} + \mu_b.\xi_n.x_{n-j} \qquad \ldots (2.29)$$

where

$$\xi_{n+1} = e_n + \sum_{i=1}^{I} c_i.e_{k-i} \qquad \ldots (2.30)$$

and $0 < c_i < -1$

i.e. the update is based on a smoothed-error condition. Computer simulation (with I = 1) showed that the algorithm did converge albeit very slowly, but if the condition for hyherstability was relaxed ($c_i = -2.5$) the second-order modelling experiment converged more rapidly, but was still slow in terms real time taken to adapt.

In conclusion it is evident that modelling the acoustic path, with a filter of lower order than the path, would require the recursive adaptive filter to be implemented using the form of Parikh's sequential regression algorithm; since it is known to converge for this condition. It is also the most complex to implement in hardware (of those reported), particularly in relation to providing a non-simple division. Also, to ensure stability a considerable processing overhead would be required each cycle to test a number of criteria, and if necessary, perform some remedial action. The main reason for rejecting this architecture was that in 1981, when a decision had to be made, it was not realised that David's stability criteria could, or should, have been made each cycle. It was believed that the filter, using finite-precision arithmetic, would have had a significant probability of being unstable and therefore unlikely to enable a loudspeaking telephone provide duplex transmission for a continuous length of time.

## 2.3 The FIR Adaptive Filter.

### 2.3.1 Introduction.

The current field of 'real time' adaptive filters grew out of a paper by Widrow [219]. The major contribution of this and a subsequent paper [220] on an application of beam forming an aerial was a practical solution of the Wiener-Hopf [221] optimum filter. A simplified derivation of Widrow's solution can be found based on statistical properties of signals, rather than the deterministic approach of Widrow [222]. In a reference channel modelling connection shown in figure 2.2, where the outputs of the channel (which contains a noise generator 'n' uncorrelated with the input signal 'x') and adaptive filter are subtracted to form an error signal. The coefficients of a transversal filter are altered to reduce some performance criterion of error to a minimum. Using vector notation, then, at time sample 'j' :

$$e_j = y_j + n_j - r_j \qquad \ldots (2.31)$$

$$e_j = y_j + n_j - X_j^T H_j^* \qquad \ldots (2.32)$$

Squaring and assuming the 'j'th sample :

$$e^2 = y^2 + n^2 + X^T H^* \, X^T H^* - 2.y.X^T H^* - 2.n.X^T H^* \qquad \ldots (2.33)$$

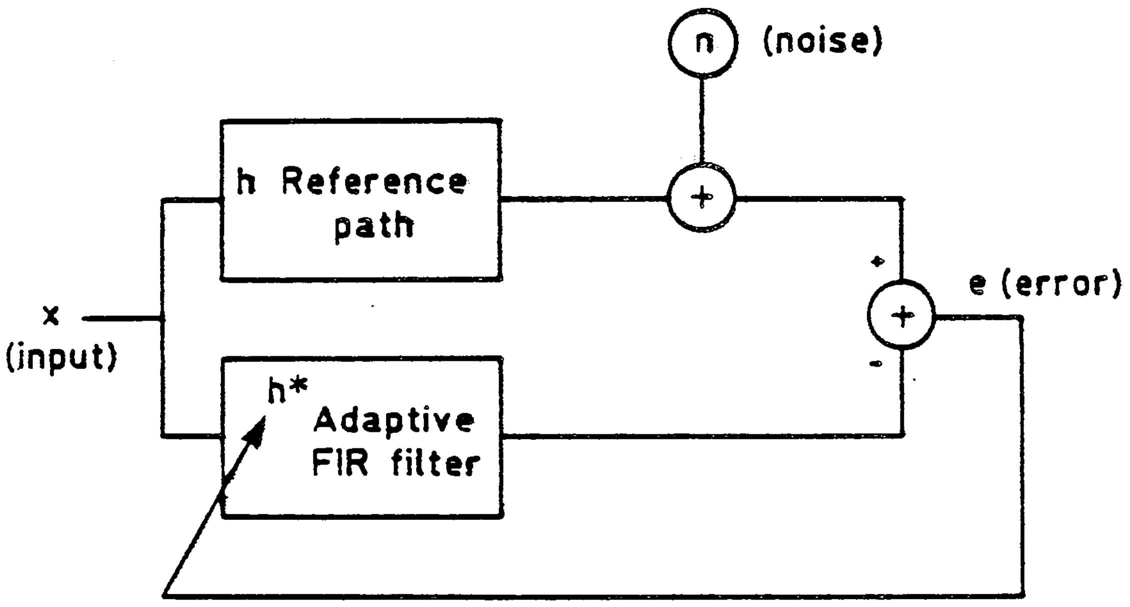


FIG. 2.2 MODELLING A REFERENCE PATH WITH AN ADAPTIVE FIR FILTER

Assume that 'n' is a normal deviate with zero mean and variance $\sigma_n^2$, i.e. :

$$E(n) = 0, \; E(n^2) = \sigma_n^2 \qquad \ldots (2.34)$$

Then

$$E(\bar{e}^2) = \bar{y}^2 + \bar{n}^2 + H^{*T} E(XX^T) H^* - 2.E(y.X)^T H^* \qquad \ldots (2.35)$$

Differentiating with respect to $H^*$ :

$$\nabla_{H^*}(\bar{e}^2) = 2.E(XX^T)H^* - 2.E(y.X) \qquad \ldots (2.36)$$

The optimum vector $H_{LMS}^*$ is found by equating the gradient to zero :

$$H^*_{LMS} = E(XX^T)^{-1}\ E(y.X) \qquad \ldots (2.37)$$

This is the Wiener-Hopf equation. It can be solved by taking a series of measurements to find expectations of the autocorrelation and cross-correlation terms (involving a large amount of storage), inverting the former and solving for the vector $H^*$.

Widrow [219] proposed that the instantaneous value of 'e' be used in place of the mean; then the minimum value of gradient can be found by a simple search technique (method of steepest descent). Here the vector $H^*$ is updated by :

$$H^*_{j+1} = H^*_j + \Delta H^*_j \qquad \ldots (2.38)$$

The update term $\Delta H^*$ is the gradient of some function of an error performance surface, then :

$$\Delta H^* = \mu.\nabla_{H^*}(f(e)) \qquad \ldots (2.39)$$

where $\mu$ is a 'step-size' scaler to ensure stability.

Equation (2.39) can be written :

$$\nabla_{H^*}(f(e)) = \frac{\partial f(e)}{\partial e}\ .\ \nabla_{H^*}(e) \qquad \ldots (2.40)$$

Let $Fn' = \dfrac{\partial f(e)}{\partial e}$

Differentiating equation(2.32) and substituting for the above, then equation (2.38) becomes :

$$H^*_{j+1} = H^*_j - \mu.Fn'_j.X_j \qquad \ldots (2.41)$$

The optimality condition Fn must be a positive even function with monotonically (and odd) nondecreasing derivative. Two functions are shown in figure 2.3.

Continuing with $Fn = e^2$, then equation (2.41) becomes :

$$H^*_{j+1} = H^*_j - 2.\mu.e_j.X_j \qquad \ldots (2.42)$$

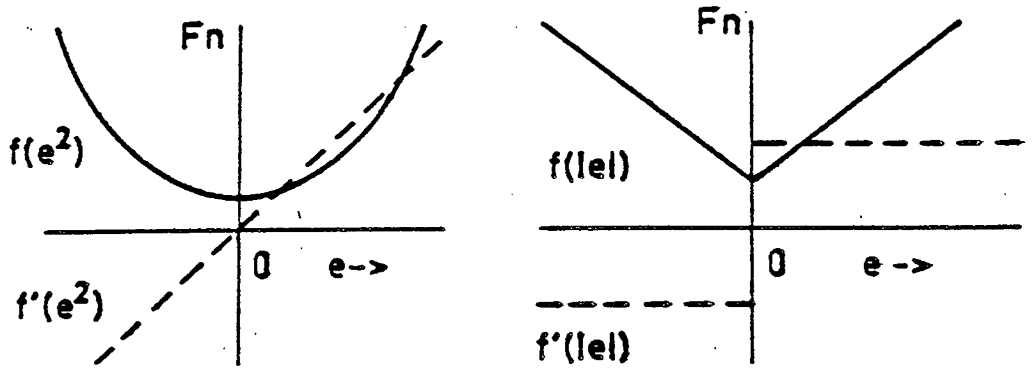


FIG.2.3 OPTIMALITY CONDITION AND DERIVATIVE

A misadjustment vector **D** is defined as :

$$\mathbf{D} = \mathbf{H} - \mathbf{H}^* \qquad \ldots (2.43)$$

Then, substitute eqn.(2.41) into eqn.(2.42) and let

$$e_j = \xi_j + n_j \qquad \ldots (2.44)$$

$$\mathbf{D}_{j+1} = \mathbf{D}_j + 2.\mu.(\xi_j + n_j).\mathbf{X}_j \qquad \ldots (2.45)$$

Form the square of the Euclidean vector length :

$$||\mathbf{D}_{j+1}||^2 = ||\mathbf{D}_j||^2 + 4.\mu.(\xi_j + n_j).\mathbf{D}_j^T\mathbf{X}_j + 4.\mu^2.(\xi_j + n_j)^2.||\mathbf{X}_j||^2 \qquad \ldots (2.46)$$

Take expectations and assume the 'j'th sample unless written otherwise

$$E(||\mathbf{D}_{j+1}||^2) = E(||\mathbf{D}_j||^2) + 4.\mu.E(\xi).E(\mathbf{D}^T\mathbf{X}) + 4.\mu^2.E(||\mathbf{X}||^2).(E(\xi^2) + \sigma_n^2) \qquad \ldots (2.47)$$

but $E(\mathbf{D}^T\mathbf{X}) = E(\xi)$, therefore eqn.(2.47) becomes :

$$E(||\mathbf{D}_{j+1}||^2) = E(||\mathbf{D}_j||^2) + 4.\mu.E(\xi)^2\left[1 + \mu.E(||\mathbf{X}||^2)\left[1 + \frac{\sigma_n^2}{E(\xi)^2}\right]\right] \qquad \ldots (2.48)$$

Convergence is assured if $E(||\mathbf{D}_{j+1}||^2) < E(||\mathbf{D}_j||^2)$, i.e. on average this condition must apply. Then the right-hand side of eqn.(2.48) must be less than or equal to zero :

$$4.\mu.E(\xi)^2\left[1 + \mu.E(||\mathbf{X}||^2)\left[1 + \frac{\sigma_n^2}{E(\xi)^2}\right]\right] =< 0 \qquad \ldots (2.49)$$

Therefore the stability condition is given by :

$$0 > \mu > \frac{-1}{E(||\mathbf{X}||^2).\left[1 + \frac{\sigma_n^2}{E(\xi)^2}\right]} \qquad \ldots (2.50)$$

Also, as time tends to infinity $E(||\mathbf{D}_{j+1}||^2) = E(||\mathbf{D}_j||^2)$, then the misadjustment noise power is :

$$E(\xi)^2 = \sigma_\xi^2 = \frac{-\mu.E(||\mathbf{X}||^2).\sigma_n^2}{1 + \mu.E(||\mathbf{X}||^2)} \qquad \ldots (2.51)$$

Widrow [222] proved that, the using equation (2.42) as a solution to the Wiener-Hopf filter, the approximation converged to an unbiased estimate of the filter's coefficients as time tended toward infinity.

The strength of the LMS algorithm (equation (2.42)), sometimes called the Widrow-Hoff equation, is that, while the performance function is quadratic ($e^2$), the gradient with respect to the coefficients $\nabla_H(e^2)$ is a linear function of those coefficients. This means that there are no local minima on the performance surface; thus the filter can start from any initial value and it will converge toward the Wiener-Hopf solution. It also means that the algorithm is very robust, in that certain intentional (or unintentional) changes can be made to the algorithm and the system will still converge toward the Wiener filter, albeit at a slower rate.

It is worth pointing out that current theory is based on the assumption that all samples are statistically independent (and independent of H). This is not the case for most signals, particularly so for speech which has high correlation between adjacent samples. Daniel reported [223] the effect of replacing the assumption of independent samples by an assumption of asymptotic independence. The nature of speech precludes any deterministic prediction of performance, but future work will be directed toward obtaining an estimate of the performance using speech.

## 2.3.2 Finite Precision.

The problem of finite precision in practical applications was first addressed by Gitlin [224], who investigated the limit imposed on performance by fixing the precision of filter coefficients. The method was to consider a non-adaptive infinite-precision system which was then impaired by a quantisation noise associated with truncation. The effective quantisation noise contributed to the error was found to be :

$$\sigma_q^2 \simeq W.\sigma_h^2.\sigma_x^2 \qquad \ldots (2.52)$$

where W is the number of coefficients, $\sigma_h^2$ is the quantisation noise associated with the least-significant digit of H and $\sigma_x^2$ is the power of the input signal.

Perry [225] considered the implication of connecting analogue-to-digital converters between a digital adaptive filter and an analogue reference channel. He also stated that, in a practical system, some truncation of the output should take place after each multiplication in the update (or correlation) term, to prevent bus sizes becoming too large. No calculation on the limitation of performance was made, but it was proved that, given the quantisation noises considered, the LMS algorithm still converged to an unbiased estimate of the Wiener filter. It was also suggested that normalisation of arithmetic would reduce the dynamic range, because the reference input signal must be scaled such that the inner product of the coefficient vector and the reference input vector (i.e. the convolution $X^TH$) cannot exceed the analogue-to-digital range. This is not necessarily true, as will be shown later.

## 2.3.3 Algorithmic Approximations.

A number of systems have been developed with the emphasis toward reducing the arithmetic processing needed in either the convolution or correlation terms (or both). For the latter equation (2.42) can be replaced by :

$$H_{j+1} = H_j + f(2\mu).f(e).g(x) \qquad \ldots (2.53)$$

where the update term components are replaced by some function of their value.

Moschner [226] showed that for $g(x) = \text{sign}(x)$, which he called the 'clipped-data algorithm', the filter still converged to an unbiased estimate of the Wiener filter, but with a speed reduction caused by a smaller word size ($\Delta h$) compared with the conventional algorithm. i.e. :

$$\Delta h_{\text{clipped data}} = \frac{1}{\sigma_x} \cdot \sqrt{\frac{2}{\pi}} \cdot \Delta h_{\text{conventional LMS}} \qquad \ldots (2.54)$$

In [226] the bounds on $\mu$ are given by :

$$0 < \mu < \sqrt{\frac{\pi}{2}} \cdot \sigma_x \cdot \frac{1}{\lambda_{\max}} \qquad \ldots (2.55)$$

where $\lambda_{\max}$ is the maximum eigenvalue of $E(\mathbf{XX}^T)$ and corresponds to $E(||\mathbf{X}||^2)$ in the derivation above.

Consequently the rate of adaptation can be increased by scaling the update value by $\sigma_x$, the standard deviation of the input signal. This is not an easy task to perform in hardware, as the square-root (or an estimate of the square-root) of the input power must be found. There is a small increase in the steady state misadjustment noise, being $\pi/2$ greater than that of the full LMS algorithm (corresponding to a 2 dB reduction in adaptation at time equal to infinity). A number of other approximations for $f(e).g(x)$ were also examined to determine their effect on speed of adaptation, stability criterion and performance.

An approximation for a function of error which would increase the speed of adaptation in the LMS algorithm, without forming a square-root, was suggested by the author [227]. Using a binary integer format an approximate value of $\Delta h^*$ would be obtained if 'e' was shifted left (equivalent to multiplication) by the most-significant bit of 'x'. i.e. writing $\tilde{x} = g(x)$ then let :

$$\tilde{x} = \text{sign}(x).2^{\text{trunc}(\log_2(|x|)} \qquad \ldots (2.56)$$

This would retain more of the information in 'x' than a fully-clipped-data algorithm where $\tilde{x} = \text{sign}(x)$, thus improving the speed of adaptation and still being relatively simple to implement in MOS technology. This is called the shift-left (SHL) algorithm.

**Lemma.**

Let the random variables 'u' and 'v' be normal deviates with expectation :

$$E(v) = 0,\ E(u^2) = \sigma_u^2,\ E(v^2) = \sigma_v^2,\ E(u.v) = \rho.\sigma_u.\sigma_v \qquad \ldots (2.57)$$

with :

$$\tilde{v} = \text{sign}(v).2^{\text{trunc}(\log_2(|v|)} \qquad \ldots (2.58)$$

Then :

$$E(u.\tilde{v}) = \sqrt{\frac{2}{\pi}}.A.E(u.v) \qquad \ldots (2.59)$$

where A is an error term equal to unity for small $\sigma_v$, falling to 0.9 as $\sigma_u$ increases.

**Proof.**

Define a random variable

$$z = \frac{u}{\sigma_u} - \rho.\frac{v}{\sigma_v} \qquad \ldots (2.60)$$

Since 'z' is the difference of two Gaussian variables, it is also a Gaussian variable.

$$E(z.v) = E\left[\left[\frac{u}{\sigma_u} - \rho.\frac{v}{\sigma_v}\right].v\right] \qquad \ldots (2.61)$$

$$E(z.v) = \rho.\frac{\sigma_u}{\sigma_u}.\sigma_v - \rho.\frac{\sigma_v^2}{\sigma_v} = 0 \qquad \ldots (2.62)$$

Therefore 'z' and 'v' are uncorrelated and, since they are Gaussian, they must also be independent. Let us define :

$$\tilde{v} = \text{sign}(v).2^{\text{trunc}(\log_2(|v|)} \qquad \ldots (2.63)$$

for

for $|v| > 4$

$\tilde{v} = 0$ for $v = 0$

$$E(z.\tilde{v}) = E(z).E(\tilde{v}) = E(z).0 \qquad \ldots (2.64)$$

Since v has zero mean and normal distribution :

$$E(z.\tilde{v}) = E\left[\left[\frac{u}{\sigma_u} - \rho.\frac{v}{\sigma_v}\right].\tilde{v}\right] = 0 \qquad \ldots (2.65)$$

Therefore

$$E(u.\tilde{v}) = \rho.\frac{\sigma_u}{\sigma_v}.E(v.\tilde{v}) \qquad \ldots (2.66)$$

To determine $E(v.\tilde{v})$ take the first moment :

$$E(v.\tilde{v}) = \int_{-\infty}^{+\infty} v.\tilde{v}.\frac{1}{\sqrt{2\pi}.\sigma_v}.\exp\left[-\frac{1}{2}\left[\frac{v}{\sigma_v}\right]^2\right].dv \qquad \ldots (2.67)$$

Since $\quad v.\tilde{v} = |v.\tilde{v}|$

$$E(v.\tilde{v}) = \frac{1}{\sigma_v}\sqrt{\frac{2}{\pi}}\int_{0}^{+\infty} v.\tilde{v}.\exp\left[-\frac{1}{2}\left[\frac{v}{\sigma_v}\right]^2\right].dv \qquad \ldots (2.68)$$

Solving numerically

$$E(v.\tilde{v}) = \sqrt{\frac{2}{\pi}}.A.\sigma_v^2 \qquad \ldots (2.69)$$

where A = 1 for small $\sigma_v$ (1 to 8) and approximately equal to 0.9 for $\sigma_v > 32$.

$$E(u.\tilde{v}) = \rho.\frac{\sigma_u}{\sigma_v}.A.\sqrt{\frac{2}{\pi}}.\sigma_v^2 \qquad \ldots (2.70)$$

$$E(u.\tilde{v}) = A.\sqrt{\frac{2}{\pi}}.E(u.v) \qquad \ldots (2.71)$$

**Bias of SHL Algorithm.**

From equation (2.42) the estimate of the gradient term is given by :

$$E(\nabla_{H^*}(\bar{e}^2)) = \nabla_{H^*}(e_j^2) = 2.e_j.\nabla_{H^*}(e_j) \qquad \ldots (2.72)$$

Substituting for $x = \tilde{x}$ in the SHL approximation and apply the expectation operator to the right hand side of equation (2.72) then :

$$E(\nabla_{H^*}(\bar{e}^2)) = -2.E(\hat{X}(y - X^TH^*)) \qquad \ldots (2.73)$$

Substitute equation (2.71) and let $\gamma = A.\sqrt{\frac{2}{\pi}}$

$$E(\nabla_{H^*}(\bar{e}^2)) = -2.\gamma.E(y.X) + 2.\gamma.E(XX^T)H^* \qquad \ldots (2.74)$$

$$E(\nabla_{H^*}(\bar{e}^2)) = \gamma.E(\nabla_{H^*}(\bar{e}^2) \qquad \ldots (2.75)$$

Since $\gamma$ is a constant then, for a given weight vector the gradient estimate is unbiased.

This provides a near maximum word size (hence increased speed of adaptation) and yet can be relatively simple to implement with a fast barrel-shifter. It is called the SHL ( shift-left) algorithm. The update component $\Delta H$ becomes a scaled version of the full implementation (equation (2.59)), i.e. :

$$\Delta h_{SHL} = A.\sqrt{\frac{2}{\pi}}\,\Delta h_{\text{conventional LMS}} \qquad \ldots (2.76)$$

Where A = 1 for small 'x' and A = 0.9 for large 'x'. The algorithm converges to an optimum solution in a time between the fully-clipped-data method and the conventional LMS method. Comparing equations (2.76) and (2.54), the former does not need an estimate of the standard deviation of input signal to compensate for the lost information in 'x'.

## 2.3.4 Hardware Systems.

Paul [228] produced a filter (for commercial purposes) using discrete components, based on the 'clipped-data algorithm' with 256 coefficients. A 12 channel digital echo-canceller built by Duttweiler [229] was studied by British Telecommunications in 1979 to measure its subjective performance on trunk circuits. This is an interesting device which interfaces directly to a p.c.m. system and utilizes the pseudo-logarithmic coding [230] of the signal to derive its floating-point representation and simplify the convolution process. The adaptation performance is severely curtailed by the p.c.m quantisation noise (as will be shown later); consequently loss of precison in other parts of the system could be tolerated. Although reported in 1978, few details of system implementation were given, except that compensation of the size of $\Delta H$ was provided over a portion of the input-signal range (approximately 16 - 20 dB from the table of results given), to improve speed of adaptation at low input-signal levels. Compensation is achieved by calculating $\mu$ as a function of input level. It was later reported [231] the functions substituted in equation (2.53) for the correlation multiplication were:

$$f(e).g(x) \doteq sign(e).2^{trunc(\log_2|e|)}.sign(x).2^{trunc(\log_2|x|)} \quad \ldots (2.77)$$

The filter had 128 coefficients and used floating-point arithmetic (with a five-bit mantissa). Maximum rate of adaptation was 70 dB/s over a 10 dB input range, falling with reduced input signal level. Using a white-noise source, the maximum adaptation achieved was 28 dB. Because of the low value of adaptation, which is further reduced when using a speech input signal, a nonlinear 'centre-clipper' was included, setting error signals below a threshold to zero and hence removing low-level echoes at the cost of introducing distortion. This architecture was implemented in the first true VLSI design [232] produced by Bell. Even by current standards the device was large, measuring 8x9 mm and achieving a low yield from fabrication. In all respects it was similar in performance to the discrete system.

## 2.3.5 Theoretical Performance.

To design a complete adaptive filter having all the attributes listed in section 2.1 would of necessity require some partitioning into separate LSI blocks. This is evident from the Bell VLSI design [232] which was too small in terms of the number of coefficients (having a 16ms impulse response at an 8 kHz sampling rate) and too large to manufacture economically. The number of output pins which can economically be connected to an LSI device is limited, thus partitioning sets restrictions on the size of binary-data values which can be transmitted between LSI blocks. Also, a more precise arithmetic is required than that used in the Bell device to obtain better performance. A full understanding was needed for the mechanism causing loss of performance by the introduction of finite precision bounds within a system.

The observations of Perry [225] can be extended to an analytic measure of performance (adaptation), and stability, of an adaptive filter with implementation limitations which introduce quantisation noise. The architecture is general to the Widrow-Hoff LMS algorithm and it is shown that the necessary normalisation can be distributed through the system.

Two methods of connection are shown; figure 2.4 corresponds to the system used in the loudspeaking telephone and figure 2.5 with that used in measurement of performance.

Assume that all numbers have infinite precision. Then, division is modelled by bit shifting (with infinite precision) and addition of a quantisation noise for any truncation error. It is also assumed that the impulse response of the path being modelled is bounded in time by the product of sample rate and number of filter coefficients 'W'.

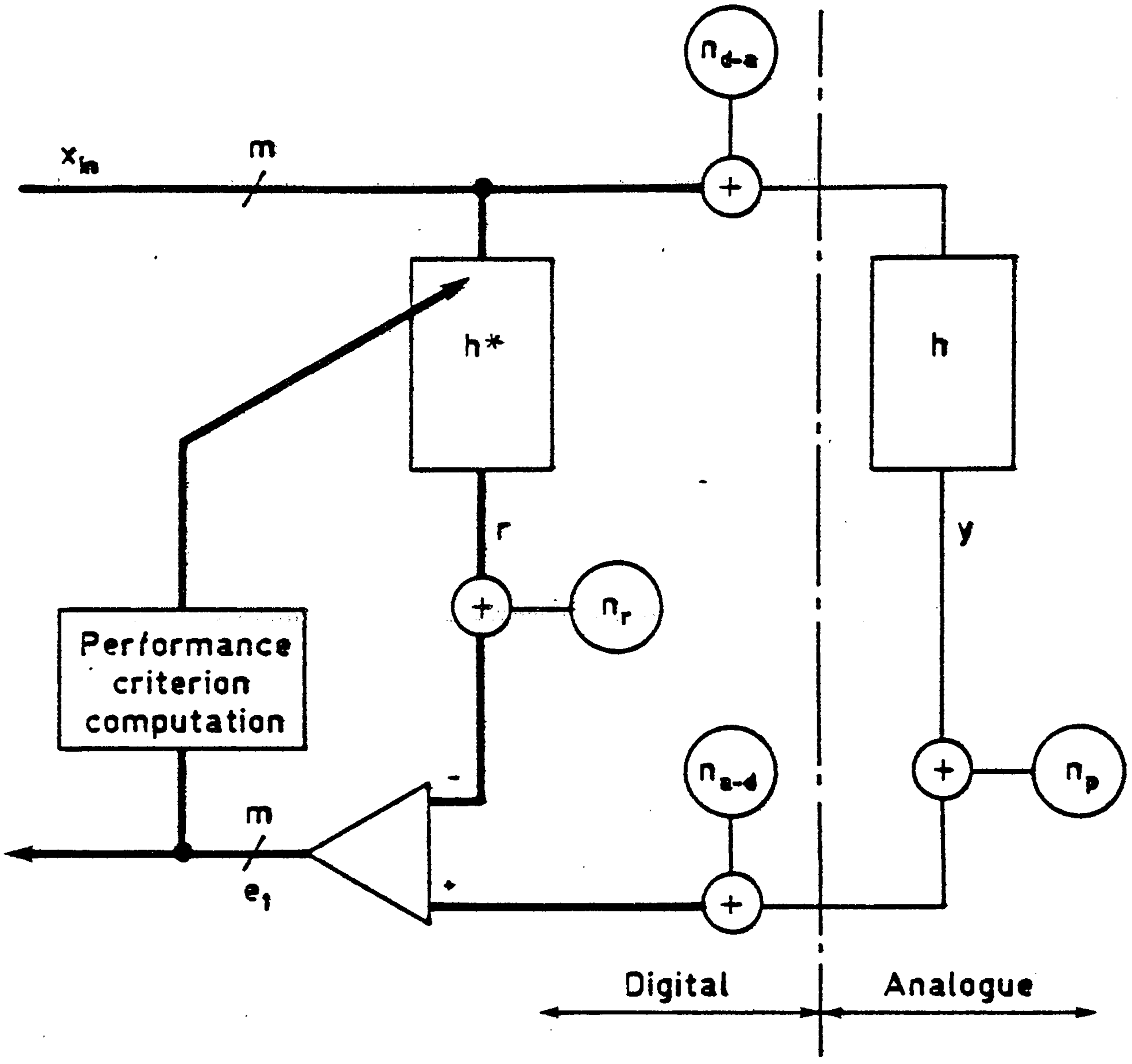


FIG. 2.4 AN ADAPTIVE FILTER WITH DIGITAL INPUT

| | | |
|---|---|---|
| Where | x | - is the input signal (digital), |
| | y | - the desired output from the path to be modelled, |
| | h | - the impulse response of the path, |
| | $h^*$ | - the estimate of h, |
| | r | - the replica of y, |
| | $n_r$ | - quantisation noise associated with scaling replica, |
| | $n_{a\_d}$, $n_{d\_a}$ | - quatisation noise of analogue/digital conversion, |
| | m | - bit size of data, |
| | $n_p$ | - noise in path, |
| | $e_1$ | - the error output from the filter. |

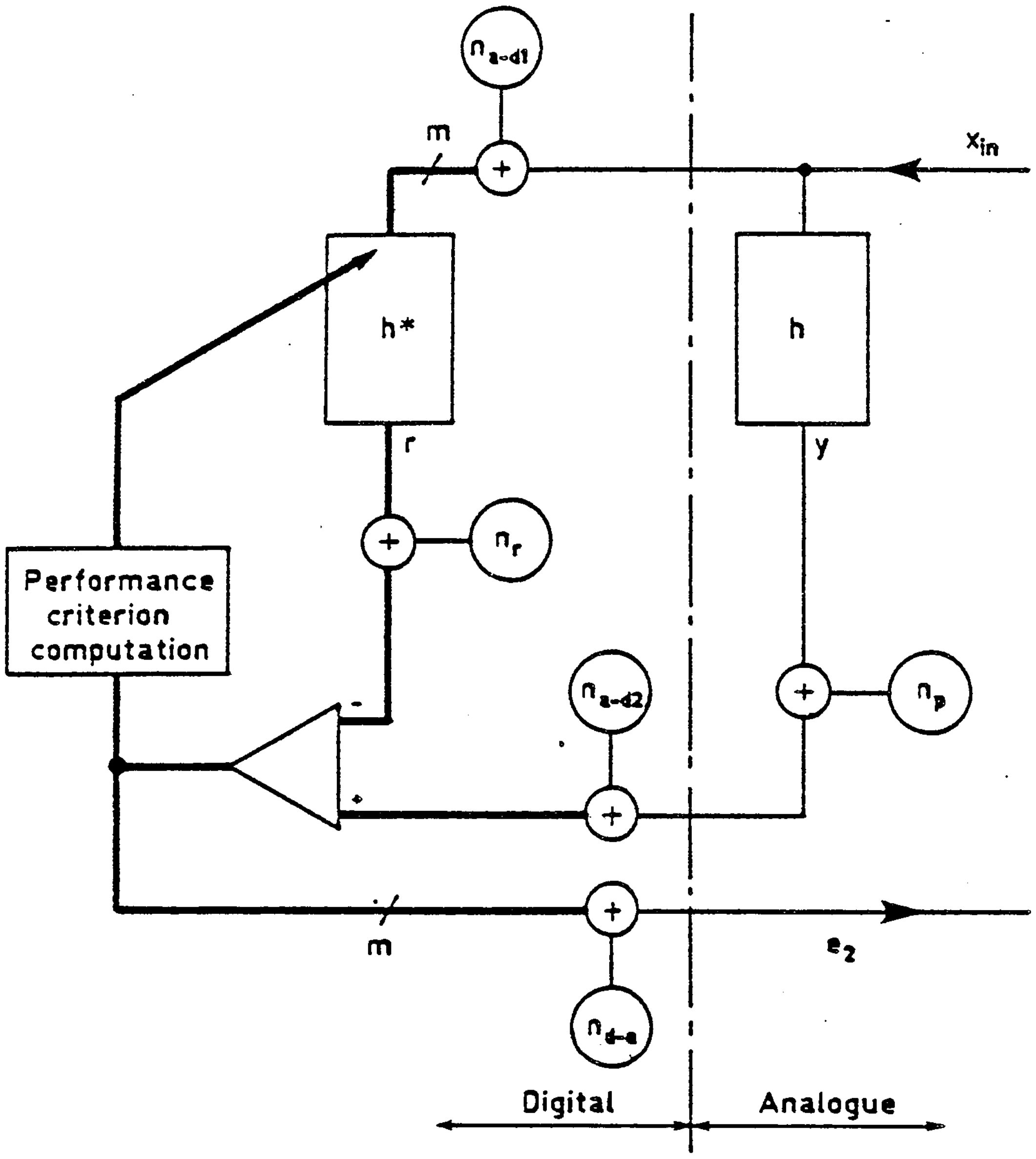


FIG.2.5 AN ADAPTIVE FILTER MODELLING AN ANALOGUE PATH

Writing the equation for error in fig. 2.4 at a sample period 'j'

$$e_{1,j} = \frac{1}{g}\sum_{i=0}^{\infty}(x_{j-i} + n_{d\_a,j}).h_i + n_{a\_d} + n_p -$$

$$\frac{1}{g.k}\sum_{i=0}^{N-1} x_{j-i}.(h^*_{i,j} + n_{t,i,j}) - n_{r,j} \qquad \ldots (2.78)$$

N.B. 'x' is an 'm' bit word, but is defined to be of infinite precision. If the effect of reducing the precision of 'x' in the convolution is required to be known, then $(x + n_x)$ is substituted for 'x', where '$n_x$' is the quantisation noise associated with the truncation of 'x'. Also, any scaling of 'x' must be included in the equation. '$n_t$' is the quantisation of $h^*$ used in the convolution and is not necessarily the precision to which $h^*$ is stored. 'g' is a scaler which puts the replica into a useful number field. The term 'number field' is not correct in the strict mathematical sense, which would imply modulo g arithmetic. 'g' should define a 'restricted-number range', since in any implementation numbers exceeding $|g|$ are truncated. The variable 'k' is an optional additional division factor, to scale the replica for impulse responses where there is a priori knowledge that the maximum value of $h^*$ can be scaled by 'k' to lie in the number field of the filter. '$n_r$' is the quantisation noise due to truncation of the replica and is a function of 'k'.

From figure 2.5

$$e_{2,j} = \frac{1}{g} \sum_{i=0}^{\infty} x_{j-i} \cdot h_i + n_{a\_d2,j} + n_{d\_a,j} + n_p -$$

$$\frac{1}{g.k} \sum_{i=0}^{W-1} (x_{j-i} + n_{a\_d1,i,j}) \cdot (h^*_{i,j} + n_{t,i,j}) - n_{r,j} \qquad \ldots (2.79)$$

Comparing equations (2.78) and (2.79) it is seen that the latter has two additional noise components due to extra stages of analogue-to-digital conversion. This is because 'x' and 'e' in both figures are defined to be of infinite precision.

Expanding equation (2.79), then in vector notation :

$$e_{2,j} = \frac{1}{g} X_j^T H + n_{a\_d2,j} + n_{d\_a,j} + n_p -$$

$$\frac{1}{g.k} (X_j^T H_j^* + X_j^T N_{t,j} + N_{a\_d1,j}^T H_j^* + N_{a\_d1,j}^T N_{t,j}) - n_{r,j}$$

Let

$$D_j = H - \frac{H_j^*}{k} \qquad \ldots (2.81)$$

be the misadjustment vector.

Then

$$e_{2,j} = \frac{1}{g} X_j^T D_j + n_{a\_d2,j} + n_{d\_a,j} + n_p -$$

$$\frac{1}{g.k} (X_j^T \; N_{t,j} + N_{a\_d1,j}^T \; H_j^* + N_{a\_d1,j}^T \; N_{t,j}) \; - n_{r,j} \qquad \ldots (2.82)$$

The error consists of seven terms (lumping analogue / digital conversion noises together) on the r.h.s. of equation (2.82). From left to right, these are :

1). misadjustment noise which is a function of the input.

2). 'a-to-d' and 'd-to-a' quantisation noise interjected between the analogue path and filter.

3) noise generated within the path being modelled.

4). a function of the input signal and the quantisation noise due to the finite precision of H used in the convolution.

5). a component due to 'a-to-d' quantisation noise but dependent on the actual value of $H^*$.

6). a function of the 'a-to-d' quantisation noise and finite impulse-response precision used in the convolution.

7). quantisation noise due to division and truncation of the replica (r).

For brevity let

$$d_j = \frac{1}{g} X_j^T D_j \qquad \ldots (2.83)$$

the misadjustment noise.

$$v_j = \frac{1}{g} N_{a\_d1,j}^T H_j^* \qquad \ldots (2.84)$$

the H dependent noise.

$$f_j = \frac{1}{g.k} X_j^T N_{t,j} \qquad \text{... (2.85)}$$

and

$$u_j = \frac{1}{g.k} N_{a\_d1,j}^T N_{t,j} \qquad \text{... (2.86)}$$

Then equation (2.82) becomes :

$$e_{2,j} = d_j + n_{a\_d2,j} + n_{d\_a,j} + n_p - v_j - f_j - u_j - n_{r,j} \qquad \text{... (2.87)}$$

Following the usual method of steepest descent to find a set of $H^*$ which will give a minimum $e^2$ (where 'e' is the instantaneous error [222]) :

$$H_{j+1}^* = H_j^* + \mu\nabla_{H^*}(e^2) \qquad \text{... (2.88)}$$

$$H_{j+1}^* = H_j^* + 2.\mu.e.\nabla_{H^*}(e) \qquad \text{... (2.89)}$$

Differentiating equation (2.79) w.r.t. $H^*$ and substituting in equation (2.89)

$$H_{j+1}^* = H_j^* - \frac{2.\mu}{g.k}.(e_{2,j} - n_{d\_a,j}).(X_j + N_{a\_d1,j}) \qquad \text{... (2.90)}$$

N.B. the error used in the descent algorithm does not contain the a-to-d component '$n_{d\_a}$.

The component subtracted from $H_j^*$ is the update term for the filter and requires two separate multiplications, i.e. (substitute $e_j = e_{2,j} - n_{d\_a,j}$)

$$\left[\frac{2.\mu}{g.k}\right] . (e_j) . (X_j + N_{a\_d1,j}) \qquad \text{... (2.91)}$$

The first term is a scaler dependent on the system implementation. After the formation of each product it is assumed that the result will be divided by a term (t) to bring it into a useful number range. Depending on the precision of the result of division, a quantisation noise is added at that point in the system. Substituting equation (2.90) into (2.91), then :

$$D_{j+1} = D_j + \frac{1}{k}\left[\left[\frac{2.\mu.(e_j)}{g.k}.\frac{1}{t_1} + n_{b,j}\right].(X_j + N_{a\_d1,j}).\frac{1}{t_2} + N_{h,j}\right] \qquad \text{... (2.92)}$$

where $t_1$ and $t_2$ are the divisors with corresponding quantisation components $n_b$ and $N_h$.

Taking the square of the Euclidean vector length :

$$||\mathbf{D}_{j+1}||^2 = ||\mathbf{D}_j||^2 + \left[\frac{2.\mu}{g.k.k.t_1.t_2}\right]^2.\left[e_j + \frac{g.k}{2.\mu}.t_1.n_{b,j}\right]^2 *$$

$$||\mathbf{X}_j + \mathbf{N}_{a\_d1,j}||^2 + ||\frac{\mathbf{N}_{h,j}}{k}||^2 + \frac{4.\mu}{g.k.k.t_1.t_2}.\left[e_j + \frac{g.k}{2.\mu}.t_1.n_{b,j}\right]$$

$$* \mathbf{D}_j^T (\mathbf{X}_j + \mathbf{N}_{a\_d1,j}) + 2.\mathbf{D}_j^T \mathbf{N}_{h,j} +$$

$$\left[\frac{2.\mu}{g.k.k.t_1.t_2}.\left[e_j + \frac{g.k}{2.\mu}.t_1.n_{b,j}\right]\right].(\mathbf{X}_j + \mathbf{N}_{a\_d1,j})^T \frac{\mathbf{N}_{h,j}}{k} \qquad \ldots (2.93)$$

Taking expectations and assuming that the noise contributions are independent random deviates with zero mean and variance $\sigma^2$, gives :

$$E(n) = 0, \quad E(n_m^2) = \sigma_m^2, \quad E(d^2) = \sigma_d^2,$$

$$E(||\mathbf{N}_{a\_d1}||^2) = W.\sigma^2_{a\_d1} \quad \text{etc.} \qquad \ldots (2.94)$$

The $E(||\mathbf{X}||^2)$ will be retained, since for speech-type signals no estimate of the power of the input signal can be made. Then equation (2.93) reduces to

$$E(||\mathbf{D}_{j+1}||^2) = E(||\mathbf{D}_j||^2) +$$

$$\left[\frac{2.\mu}{g.k.k.t_1.t_2}\right]^2.\left[\sigma_d^2 + \sigma^2_{a\_d2} + \sigma_u^2 + \sigma_v^2 + \sigma_f^2 + \sigma_p^2 + \sigma_r^2 + \left[\frac{g.k}{2.\mu}.t_1\right]^2.\sigma_b^2\right]$$

$$* (E||\mathbf{X}||^2 + W.\sigma^2_{a\_d1}) + \frac{W}{k^2}.\sigma_h^2 + \frac{4.\mu}{k^2.t_1.t_2}.\sigma_d^2 \qquad \ldots (2.95)$$

Stability

For convergence it is seen from equation (2.95) that :

$$E(||\mathbf{D}_j||^2) < E(||\mathbf{D}_{j+1}||^2) \qquad \ldots (2.96)$$

Therefore, the right hand terms in equation (2.95) must be less than zero. This function is quadratic in $(2.\mu/g.k)$, where the negative region lies between its roots.

For brevity let :

$$\sigma_n^2 = \sigma^2_{a\_d2} + \sigma_u^2 + \sigma_v^2 + \sigma_f^2 + \sigma_p^2 + \sigma_u^2 + \sigma_r^2 \qquad \ldots (2.97)$$

and

$$\left[1 + W.\sigma_{a\_d1}^2.\frac{1}{E||\mathbf{X}||^2}\right] = \gamma \qquad \text{... (2.98)}$$

The roots are found from :

$$\left[\frac{2.\mu}{g.k}\right]^2 . \frac{1}{(k.t_1.t_2)^2}.(\sigma_d^2 + \sigma_n^2).E||\mathbf{X}||^2.\gamma + \left[\frac{2.\mu}{g.k}\right].\frac{2.g.\sigma_d^2}{k.t_1.t_2} +$$

$$E||\mathbf{X}||^2.\frac{\gamma}{(k.t_2)^2}.\sigma_b^2 + \frac{1}{k^2}.W.\sigma_h^2 = 0 \qquad \text{... (2.99)}$$

Let $$Z = (1 + \frac{1}{\sigma_d^2}.\sigma_n^2) \qquad \text{... (2.100)}$$

then rearranging

$$\left[\frac{2.\mu}{g.k}\right]^2 + \frac{2.\mu}{g.k} . \frac{2.g.k.t_1.t_2}{Z.\gamma.E||\mathbf{X}||^2} +$$

$$\frac{1}{Z.\gamma.E||\mathbf{X}||^2.\sigma_d^2}.(\gamma.E||\mathbf{X}||^2.t_1^2.\sigma_b^2 + t_1^2.t_2^2.W.\sigma_h^2) = 0 \qquad \text{... (2.101)}$$

This is a quadratic equation of the form $z^2 + b.z + c = 0$

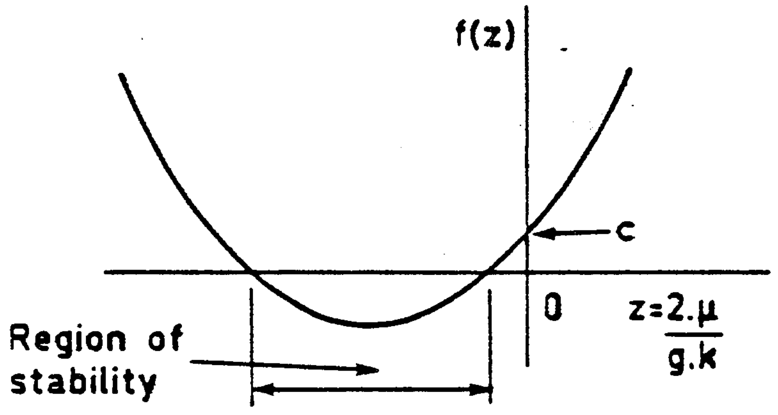


FIG. 2.6 SCHEMATIC OF QUADRATIC EQUATION

Depending on the coefficients, the region (2.μ/g.k) is negative from a minimum value to a maximum which is less than zero, as shown in fig.2.6. The practical formation of this factor requires a division; since its implementation is system-dependent, the actual value will be replaced by a function $f(2.\mu/g.k)$ which will allow a nonlinear solution.

The roots of equation (2.101) are given by :

$$f\left[\frac{2.\mu}{g.k}\right]\Bigg|_{=0} = -\frac{g.k}{\gamma.E||\mathbf{X}||^2.Z}.t_1.t_2 \; *$$
$$\left[1 \pm \mathrm{SQRT}\left[1 - E||\mathbf{X}||^2.\frac{\gamma.Z}{(\sigma_d.g.k.t_1.t_2)^2}.(\gamma.E||\mathbf{X}||^2.t_1^2.\gamma.\sigma_b^2 + t_1^2.t_2^2.W.\sigma_h^2)\right]\right] \qquad \text{... (2.102)}$$

The term subtracted from unity inside the square root is very small and can be neglected. Substituting for $\gamma$ and $Z$ then :

$$0 < f\left[\frac{2.\mu}{g.k}\right] < \frac{-2.g.k.t_1.t_2}{E||\mathbf{X}||^2.(1 + W.\sigma_{a\_d1}^2.\frac{1}{E||\mathbf{X}||^2}).(1 + \sigma_n^2.\frac{1}{\sigma_d^2})} \qquad \text{... (2.103)}$$

**Adaptation**

As time tends to infinity (in a practical system this could vary from milliseconds to minutes) it is expected that

$$E(||\mathbf{D}_{j+1}||^2) = E(||\mathbf{D}_j||^2) \qquad \text{... (2.104)}$$

Substituting into equation (2.95) and rearranging to obtain the unknown misadjustment noise :

$$\sigma_d^2 = -\frac{\left[\sigma_n^2 + \dfrac{\sigma_b^2.t_1^2}{f\left[\frac{2.\mu}{g.k}\right]^2} + \dfrac{W.t_1^2.t_2^2.\sigma_h^2}{f\left[\frac{2.\mu}{g.k}\right]^2.E||\mathbf{X}||^2.\gamma}\right]}{\left[1 + \dfrac{2.g.k.t_1.t_2}{f\left[\frac{2.\mu}{g.k}\right].E||\mathbf{X}||^2.\gamma}\right]} \qquad \text{... (2.105)}$$

Squaring equation (2.87) and taking expectations, as in equation (2.94), then the variance of error ($e_2$) is given by :

$$\sigma_e^2 = \sigma_d^2 + \sigma_{a\_d2}^2 + \sigma_{d\_a}^2 + \sigma_u^2 + \sigma_v^2 + \sigma_f^2 + \sigma_p^2 + \sigma_r^2 \qquad \ldots (2.106)$$

Repeating the above procedure with equation (2.78) results in similar equations (2.103) and (2.105) for stability and misadjustment respectively, with exceptions that $\gamma = 1$, $n_{a\_d2},u$ and $\sigma_u^2 = 0$.
Also, equation (2.84) becomes :

$$v_j = \frac{1}{g} N_{a\_d,j}^T H_j \qquad \ldots (2.107)$$

It may be noted that the quantisation noise contribution due to the finite-precision storage of the impulse response H can be found as the third term on the right of equation (2.105). Substituting for the largest negative value of the function $f(2.\mu/g.k)$ in equation (2.103) and writing

$$W.\sigma_x^2 = E(||\mathbf{X}||^2) \qquad \ldots (2.108)$$

gives :

$$\sigma_q^2 \simeq W^2.\sigma_h^2.\sigma_x^2 \qquad \ldots (2.109)$$

Comparing this with equation (2.52), i.e. the value obtained by Gitlin [224] and Perry [225], which did not consider the full LMS algorithm, then this noise contribution is proportional to the number of coefficients squared and not simply the number of coefficients.

Summarising, in 1981 when a decision was required on the type of adaptive filter which would be incorporated in a commercially viable loudspeaking telephone, the FIR filter appeared a more tractable problem than the IIR type. While the latter would provide an area for future work, it was decided that no working LSI circuit could be realised in the timescale required, which was approximately two years. Consequently the available effort was directed toward a solution of a FIR adaptive filter.

# 3. EVOLUTION OF AN ARCHITECTURE FOR A DAF IN LSI.

## 3.1 Introduction.

Because of the high cost of analogue/digital conversion, the final LST design would have a two-channel converter interfacing between the analogue environment and a totally digital regime comprising the adaptive filters and controller. The circuit which evolved for the digital adaptive filter (DAF) is based on the distributed-normalisation architecture [301,302] developed in Chapter 2. The properties derived from this architecture are exploited in the trading of increasing internally generated quantisation noise against increasing speed of adaptation, for a system that must severely constrain the precision of arithmetic operations.

Although the original design was based on an approximation to the LMS algorithm [222], called the SHL (shift-left) algorithm, developed in section 2.3.3, this was not used in the final hardware system. In discussion with LSI designers, a fixed-point binary multiplier had previously been designed, which could provide the necessary precision and speed requirement. It was claimed that the total area occupied by the multiplier was of a comparable size to that of a barrel shifter required for the SHL design. A further advantage was a considerable saving in design time. In the final analysis this decision may not have been the best. As LSI design progressed it became evident, from computer simulation, that the multiplier had a high power consumption and space became at a premium with the placement of large parallel buses.

Similar reasons were proposed by LSI designers to use fixed-point arithmetic, where the dynamic range requirements would indicate that a floating-point solution was a better alternative. The former is readily made up of logic cells which can be repeated many times, giving a low fault liability, since thorough testing by simulation is more easily done, and design time is low. The opposite is true for the latter which requires a large amount of random logic, e.g. to scale the exponent from the mantissa, provide rounding after truncation and overflow protection.

To avoid the problem of low yield of LSI devices associated with large area, a limit of between 3 and 4 $mm^2$ was set on the size of an individual chip. This imposed a requirement to partition the filter into sections, which, in LSI, would have a high gate utilization efficiency.

The environment for the DAF was well known, i.e. the characteristics of paths to be modelled, the level of adaptation required, and that it must work in a double-talk situation, which is the general case for normal free conversation. A technique was required which would prevent an interrupting talker from destroying the impulse response previously learnt, and yet be able to track a rapidly-changing impulse response. The method used was first proposed by Ochai [303], and required two FIR filters, one being adaptive and having the facility to transfer coefficients to the other filter. The LSI design would incorporate both filters and exploit the area efficiency gain by multiplexing the convolution process between both.

## 3.2 Some Performance Parameters.

Consider some of the properties of performance obtained from equations (2.105), the misadjustment noise, and (2.106), the total error-power. In the latter it is seen that with the exception of misadjustment noise ($\sigma_d^2$), the error is a sum of independent noise contributions associated with the external reference path i.e.

$(\sigma^2_{a\_d2}, \sigma^2_p, \sigma^2_{a\_d}, \sigma^2_v$ and $\sigma^2_u)$

and arithmetic precision within the DAF ($\sigma_f^2$ and $\sigma_r^2$), which can either be measured or estimated. For example, the contribution from $\sigma_v^2$ (equation (2.84)) is due to the quantisation noise added to the input signal x and passed through either the reference path or the DAF, depending on the method of use, but not both (see figs (2.4) and (2.5)). This can only be found by measurement of the actual impulse response, but in practical applications is generally found to be small. To obtain an estimate for comparison of theory and measurement, equation (2.84) is squared and the expectation taken :

$$\sigma_v^2 = E\left[\frac{1}{g^2}.(N_a^T H)^2\right] \quad \dots (3.1)$$

where $N_a$ is the vector of analogue/digital quantisation noise $n_a$ associated with the impulse response H, and g is a normalising scaler. E is the expectation operator. The quantisation noise is assumed to have zero mean and variance $\sigma_a^2$ (as in equation (2.94)), the latter given by [304] :

$$\sigma_a^2 = \frac{1}{12}.\delta_a^2 \quad \dots (3.2)$$

where $\delta_a$ is the value of the least-significant digit of the analogue-to-digital conversion. A uniform distribution is assumed for the probability of a value falling within the range of plus or minus half the least significant digit. Equation(3.1) reduces to :

$$\sigma_v^2 = \frac{\delta_a^2}{12.g^2}.E\left[\sum_{i=0}^{W-1} h_i^2\right] \quad \dots (3.3)$$

for a filter with W coefficients.

A number of impulse responses which could be reproduced (using a programmable filter) were recorded and used for comparison of theoretical performance with measurements of the hardware filter.

Applying the same technique of squaring and taking expectations, similar expressions can be found for the remaining noise contributions in equation (2.106). From equation (2.85) the noise produced by limiting the precision of H in the convolution can be found :

$$\sigma_f^2 = \frac{W}{(g.k)^2}.\frac{1}{12}.\delta_t^2.\sigma_x^2 \quad \dots (3.4)$$

where $\delta_t$ is the least-significant digit of h used in the convolution; here the Expectation of $x^2$ is taken to be its variance $\sigma_x^2$.

From equation (2.86) :

$$\sigma_u^2 = \frac{W}{(g.k)^2} \cdot \frac{1}{144} \cdot \delta_a^2 \cdot \delta_t^2 \qquad \ldots (3.5)$$

This noise is very small and occurs in the measurement mode only (figure 2.5). It is proportional to the product of the a-to-d quantisation noise present on the input signal and the quantisation noise produced by truncation of the impulse response when calculating a convolution.

Finally the quantisation noise produced by truncation of replica (value of convolution) is similar to equation (3.2) :

$$\sigma_r^2 = \frac{1}{12} \cdot \delta_r^2 \qquad \ldots (3.6)$$

where $\delta_r$ is the least significant digit remaining after truncation of the replica.

Consider the misadjustment noise $\sigma_d^2$, equation (2.105), which contains all the previous noise components plus contributions from truncation of products from each multiplication in the update term. The denominator provides a condition for stability which is found to be optimum over the range of input signal level. Let the largest value for the denominator be -1, since, if a larger value, between -1 and 0, was used the misadjustment noise $\sigma_d^2$ would increase beyond the nominal values contained in the numerator). Then :

$$1 + \frac{2.g.k.t_1.t_2}{f\left[\frac{2.\mu}{g.k}\right].E(||\mathbf{X}||^2).\gamma} <= -1 \qquad \ldots (3.7)$$

Assume $\gamma = 1$ (given by equation (2.98)), then :

$$f\left[\frac{2.\mu}{g.k}\right]_{opt.} = - \frac{g.k.t_1.t_2}{E(||\mathbf{X}||^2)} \qquad \ldots (3.8)$$

will give the optimum speed of adaptation for small input-signal levels.

The misadjustment noise ($\sigma_d^2$) resulting from use of the above stability criterion is found for a white-noise input signal where :

$$W.\sigma_x^2 = E(||\mathbf{X}||^2) \qquad \ldots (3.9)$$

By substituting equations (3.8) and (3.9) into equation (2.105), then :

$$\sigma^2{}_d = \sigma_n^2 + \sigma_b^2.\left[\frac{W.\sigma_x^2}{g.k.t_2}\right]^2 + \sigma_h^2\left[\frac{W.\sigma_x}{g.k}\right]^2 \qquad \ldots (3.10)$$

Rewriting equation (3.10) as a sum of noise powers :

$$\sigma_d^2 = \sigma_n^2 + \sigma^2_{(2\mu e)} + \sigma^2_{(\Delta h)} \qquad \ldots (3.11)$$

where $\sigma^2_{(2\mu e)}$ is the quantisation noise associated with truncation of the product f(2μ).e (see equation (2.91)), i.e. the 2μe bus. From equation (3.10) this is proportional to the square of the number of coefficients and the square of the input power, and inversely proportional to the square of $t_2$, the scaler required for the 'Δh' bus (see equation (2.92)). It would therefore be a minimum for $t_1 = 2^0$, although it will be shown later that there is a considerable advantage to be gained, in terms of speed of adaptation across a range of input levels, for a particular hardware solution, if a larger value of $t_1$ is used.

$\sigma^2_{(\Delta h)}$ is the quantisation noise associated with the Δh bus and is proportional to the input power and the square of the number of coefficients.

As the input-signal level tends to zero ($\sigma_x^2 \rightarrow 0$) then, from equation (3.10), the misadjustment noise becomes equal to the sum of noises given by equation (2.97), i.e. $\sigma_d^2 = \sigma_n^2$. Substituting this into equation (2.103), the stability criterion, also produces equation (3.8). It should be noted that for large input-signal levels the ratio ($\sigma_n^2/\sigma_d^2$) becomes small, which, in equation (2.103), results in a reduced speed of adaptation. This is a small penalty to pay for having stability over the whole range of input-signal level.

Thus, all the quantisation noise contributions produced by the filter are determined by the product of a scaler and the least-significant digit used in the filter hardware.

## 3.3 System Parameters.

It is necessary to determine those parameters which may be fixed by system limitations. Consider the parameter g which scales the result of convolution (see equation (2.78)). To maximise adaptation (defined as the ratio of input-signal level to error power) the quantisation noise from a-to-d and d-to-a conversion should be a minimum. Here the balance is between cost and performance, the lowest in both categories being 8-bit PCM. Figure 3.1 [305,306] shows that, currently, cost increases logarithmically with the number of bits of precision. The minimum size which will handle the wide dynamic range of speech is 12 bits. This was chosen for the system, but without companding, to derive the maximum performance. The effect of companding will be shown later. Two's complement arithmetic was used throughout the hardware, consequently the 12-bit number range is from $2^{11}-1$ to $-2^{11}$.

In an LST environment, a filter may have to model a reference channel with zero loss or gain (i.e. the acoustic path). Allowing the filter to model a zero-ohm (straight wire) channel would enable some paths with gain to be accommodated; the limit is then set by the largest value of h in the impulse response which can be stored. A different design criterion was used in the Bell VLSI echo-canceller [232], which required a 6dB loss across the reference path. This was because the designed role was to cancel echoes across a two-to-four-wire converter (a hybrid), having a minimum 6dB loss. Using the nomenclature of Chapter 2 and referring to figure 2.4, the output from the filter on a zero-ohm path (where $h_0 = 1$) is :

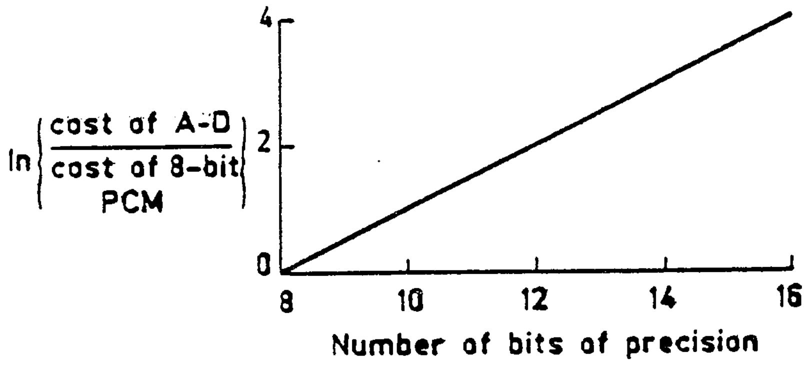


FIG. 3.1 RELATIVE COST OF A-D CONVERSION AGAINST COST OF A-D CONVERSION RELATIVE TO AN 8bit pcm SYSTEM, AGAINST NUMBER OF bits PRECISION AT 1982 PRICES

$$r = \frac{1}{g}.h_0.x \qquad \ldots (3.12)$$

Since x and r must lie in the same number range and g must be an integer power-of-two for simple division, then $g = 2^{11}$, i.e. the modulus of the filter's number range. However, if by some measurement it can be determined that the maximum value of h can be scaled by k (where k is an integer power-of-two) and still lie in the system number range, then g can be increased by k to exploit the improvement in precision of h.

Normalisation of numbers within the hardware is most easily achieved by utilising the product $t_1.t_2$. The nominal value of this term is dependent on the minimum peak-to-rms ratio of the input signal x and the number of filter coefficients. An expression may be found from equation (3.8), which must have its largest value (minus one) for the maximum input power.

Let equation (3.8) equal minus one and substitute equation (3.9). Then:

$$t_1.t_2 = \frac{W}{g.k}.\sigma_x^2 \qquad \ldots (3.13)$$

If the filter is required to handle full-scale sine-waves within its number range, then the peak-to-rms ratio of the input signal x is given by :

$$g.\frac{1}{\sigma_{x(max)}} = \sqrt{2} \qquad \ldots (3.14)$$

Substitute in equation (3.13)

$$t_1.t_2 = \frac{W.g}{2.k} \qquad \ldots (3.15)$$

Similarly, if the input-signal was known to be truncated white noise, where the peak $g = 4.\sigma_x$ (the scaling of the rms must be an integer power-of-two to enable division by $t_1.t_2$ by bit shifting the binary point), then :

$$t_1.t_2 = \frac{W.g}{2^4.k} \qquad \ldots (3.16)$$

Thus, there is an advantage to be gained, both in terms of adaptation speed and range of input-level compensation, if equation (3.13) is solved given a priori knowledge of the general statistical distribution of the input signal. D. L. Richards [307] has analysed the statistical properties of speech signals, and showed that the probability of a signal exceeding four standard-deviations was 0.005, when using a particular microphone. To confirm this result, using an electret microphone, a signal measurement was carried out for a number of talkers. Figure 3.2 shows a typical relative frequency distribution of a speech signal, measured over a 300Hz to 4kHz bandwidth. It is interesting to note that the maximum peak-to-rms ratio is 17.8dB. All the measurements made showed a characteristic logarithmic relationship for the skirts of the distribution. Thus, assuming symmetry about the centre, and ignoring the zero component; then, for one half of the distribution, the relative frequency ($f_r$) can be approximated by :

$$f_r = e^{-1.197\sigma}.e^{-2.993} \qquad \ldots (3.17)$$

where $\sigma$ is the standard deviation of the input signal.

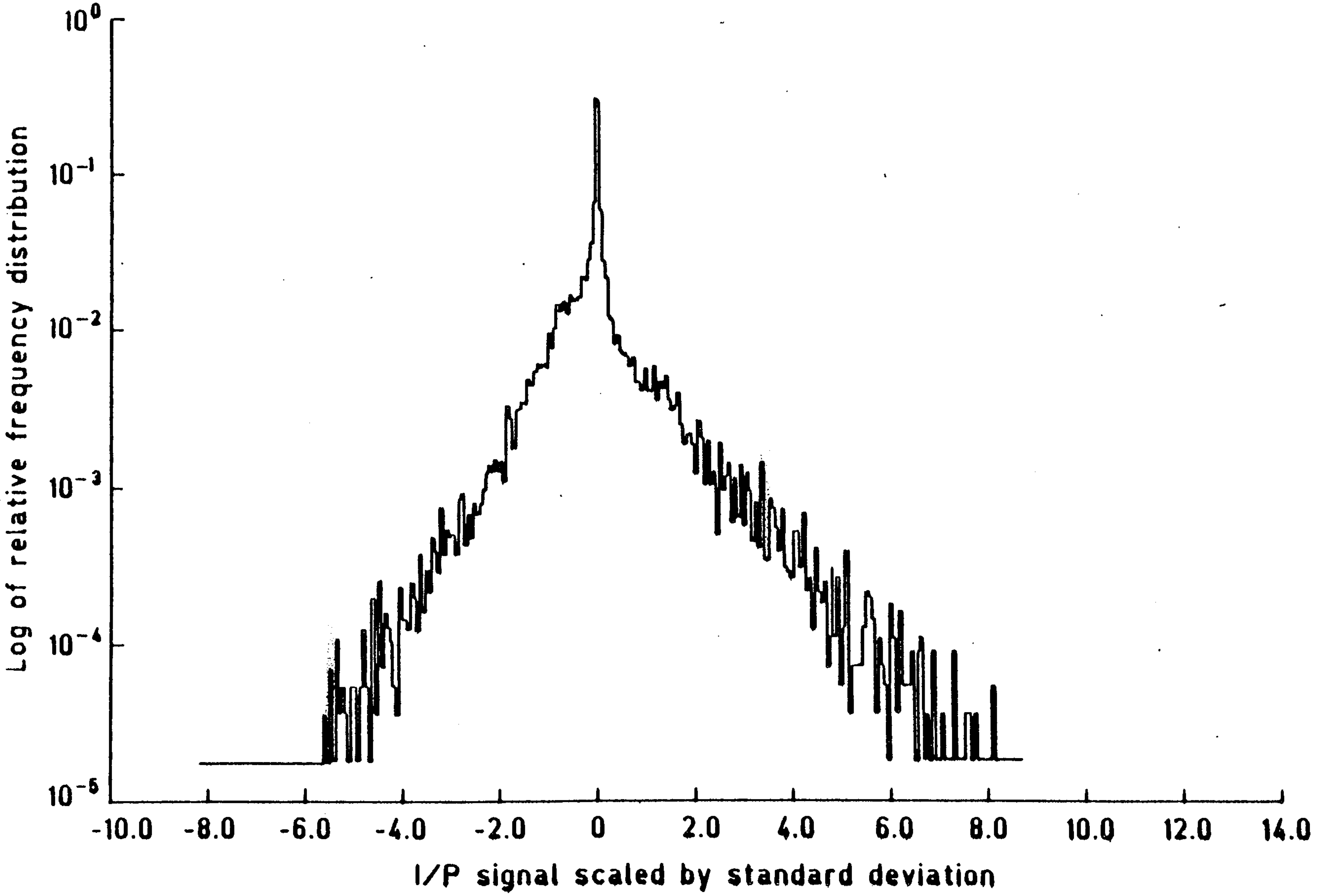


FIG. 3.2 PROBABILITY DISTRIBUTION OF SPEECH

This approximation (equation (3.17)) removes the silence component and very low level signals, so that only active speech is included. The total area for the half distribution is easily found, by integration between the limits of $\sigma = 0$ to $\infty$, to be 0.04189 ($e^{-3.173}$). The active-speech distribution can be renormalised by a scaler $\alpha$, i.e.:

$$\text{Area} = 0.5 = 0.04189 \; . \; \alpha$$

The relative frequency of active speech, for a half distribution, becomes :

$$f_r = e^{-1.197\sigma} . e^{-0.5133} \qquad \ldots (3.18)$$

The probability that a signal lies within the range $\sigma_1$ to $\sigma_2$ (of the half distribution) is found by integration between limits and is given by :

$$P(\sigma_1 < x < \sigma_2) = 0.5(e^{-1.197\sigma_1} - e^{-1.197\sigma_2}) \qquad \ldots (3.19)$$

Thus, the probability that the signal x exceeds four standard-deviations is equal to 0.008, which is very close to the value of 0.005 measured by Richards [307].

If speech was the only signal to be used in the filter, then 12dB of headroom (corresponding to $4\sigma$ clipping of the signal) would be needed between the modulus of the maximum peak, given by g, and the RMS (i.e. $\sigma$). Equation (3.16) would then be the most suitable to calculate the value of $t_1.t_2$. The main advantage comes when the compensation achieved by calculating the update coefficient $f(2\mu)$ can only be implemented over a finite range. If equation (3.16) is used instead of equation (3.15) then lower-level signals are compensated, obtaining improved speed of adaptation in the low-level region.

It was envisaged that the DAF would have wide application outside echo cancellation, including processing sine-waves. Consequently the basic filter was designed using equation (3.15), but a bonding option was provided to implement $g/\sigma_x = 2$. This is a small range change but it could be increased at a later date (should there be a demand for the change).

The final component required to solve equation (3.15) is the number of coefficients W, which is dependent on how the hardware system is organised and the time required to perform each multiply and accumulate for the convolution. In the following section, which describes the hardware evolution, W was constrained to be a maximum of $2^8$ (256 coefficients), which if substituted into equation (3.15) and letting $k = 2^0$ gives $t_1.t_2 = 2^{18}$.

## 3.4 Some Aspects of Hardware Evolution.

### 3.4.1 Generation of $f(2\mu)$.

If an estimate had to be made of the expectation of input power ($E(||\mathbf{X}||^2)$) for a speech signal, a pessimistic value would be needed to ensure stability. This is because the RMS level of speech varies during a conversation. In consequence, the update coefficient $f(2\mu)$ would be smaller than optimum, giving a lower speed of adaptation. Alternatively, the instantaneous value of $||\mathbf{X}||^2$ can be calculated, for each cycle, with some degree of complexity, and used in place of its expected value. i.e equation (3.8) becomes:

$$f\left[\frac{2.\mu}{g.k}\right]_{opt.} = -\frac{g.k.t_1.t_2}{||\mathbf{X}||^2} \qquad \ldots (3.20)$$

where the subscript 'opt' is an abbreviation of 'optimum'.
This is equivalent to calculating the total energy stored in the filter, which is then used as an estimate of the power. Thus the approximation is valid for a filter with a finite number of coefficients. The obvious advantage is the ability to track rapidly varying talker levels and provide optimum speed of adaptation for any type of input signal.

The behaviour of this approximation, applied to the hardware, can be examined for a Gaussian input signal, which theoretically has a constant power given by its variance $\sigma^2$. Summing the squares of x over W terms (the number of coefficients) will produce a variable having a chi-square ($\chi^2_{[W]}$)

distribution [308], which, for large $W$, is also a Gaussian distribution (by the central-limit theorem), having a mean of $W\sigma^2$ and standard deviation equal to $\sqrt{2W}.\sigma^2$. Forming the reciprocal of $||\mathbf{X}||^2$ in a hardware implementation is most simply achieved by an approximation to an integer power-of-two. Equation (3.20) then becomes:

$$f\left[\frac{2.\mu}{g.k}\right]_{apx.} = -\frac{g.k.t_1.t_2}{2^{\mathrm{trunc}(\log_2(||\mathbf{X}||^2))}} \qquad \ldots (3.21)$$

where the operator 'trunc' removes any value below the binary point, and the subscript 'apx' indicates an approximation. This approximation is nonuniform quantisation.

Now consider an input signal with its variance ($\sigma_x^2$) lying centrally between two integer powers of two, given by $2^n$ and $2^{n+1}$. The mean of the chi-square distribution ($\chi^2_{[W]}$) is therefore:

$$W.\sigma_x^2 = W.\frac{(2^n + 2^{n+1})}{2} = W.3.2^{n-1} \qquad \ldots (3.22)$$

The number of standard deviations from the mean to a boundary edge, where the 'trunc' operator in the denominator of equation (3.21) would alter the value of $f(2.\mu)$, is therefore:

$$\frac{W.2^{n+1} - W.3.2^{n-1}}{\sqrt{2.W}.3.2^{n-1}} = \frac{1}{3}\sqrt{\frac{W}{2}} \qquad \ldots (3.23)$$

Thus, knowing the number of coefficients $W$, the probability that the estimate of power will wander into adjacent slots can be found from normal distribution tables. e.g. for $W = 2^8$ :

$$P[2^n < \chi^2_{[W]} > 2^{n+1}] = 2.6\ 10^{-4}$$

This is very small, as would be expected for large $W$, indicating the accuracy of the approximation. Obviously, as the input power approaches the edge of a boundary change the transfer between adjacent appoximate values for f(2μ) becomes more frequent.

Substituting $C = g.k.t_1.t_2$ in equation (3.21), the approximation can be written :

$$f\begin{bmatrix}\underline{2}.\mu\\ g.k\end{bmatrix}_{apx.} = \frac{C}{||\mathbf{X}||^2} \Longrightarrow \frac{C}{2^{\mathrm{trunc}(\log_2(||\mathbf{X}||^2))}} \qquad \ldots (3.24)$$

To implement this approximation consider Table (3.1), which shows the truth table of an input-to-output function to solve equation (3.24), over an n bit range of $||\mathbf{X}||^2$. '?' is a don't-care condition.

| input = $\frac{\|\|\mathbf{X}\|\|^2}{C}$ | | | | | | output | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| sign | $2^{m+n}$ | $2^{m+n-1}$ | ... | $2^{m-1}$ | $2^m$ | sign | $2^n$ | $2^{n-1}$ | ... | $2^1$ | $2^0$ |
| 0 | 1 | ? | ... | ? | ? | 0 | 0 | 0 | ... | 0 | 1 |
| 0 | 0 | 1 | | ? | ? | 0 | 0 | 0 | | 1 | 0 |
| . | | | | | . | . | | | | | . |
| . | | | | | . | . | | | | | . |
| 0 | 0 | 0 | | 1 | ? | 0 | 0 | 1 | | 0 | 0 |
| 0 | 0 | 0 | | 0 | 1 | 0 | 1 | 0 | | 0 | 0 |
| 0 | 0 | 0 | ... | 0 | 0 | 0 | 1 | 1 | ... | 1 | 1 |

Table 3.1 Truth table for reciprocal approximation.

The upper input value of $2^{n+m+1} - 1$ is set by the expected peak-to-rms ratio of input signal, as described in Section 3.3. The lower input value of $2^m-1$ may be set by a hardware constraint; e.g. a limit on the size of a data transfer between the computation of the sum-of-squares of x ( given the abreviation 'SSX') and the generation of $f(2\mu)$. Consequently, these two bounds restrict the range over which compensation for input level can be applied.

The simplest method to perform the function required in Table 3.1 uses combinatorial logic, which is shown in figure 3.3 with 'AND' and 'OR' gates.

### 3.4.2 Timing strategy.

A general timing strategy is required for the filter, which will allow the multiplexing of functional blocks (e.g. multipliers), data transfer between processing elements (e.g. input and output or transfers between separate LSI circuits), and most important, the actual implementation of the LMS algorithm (equation (2.42) or (2.90)). To elaborate on the latter point, only one reference [309] has explicitly described the sequence of operations (although, by deduction from the number of coefficients or the performance, other systems must use a similar approach). In this, a limit cycle is introduced by solving the LMS algorithm in the following manner:

i) convolution (serial format) in cycle $n_j$,

ii) compute error in cycle $n_j$

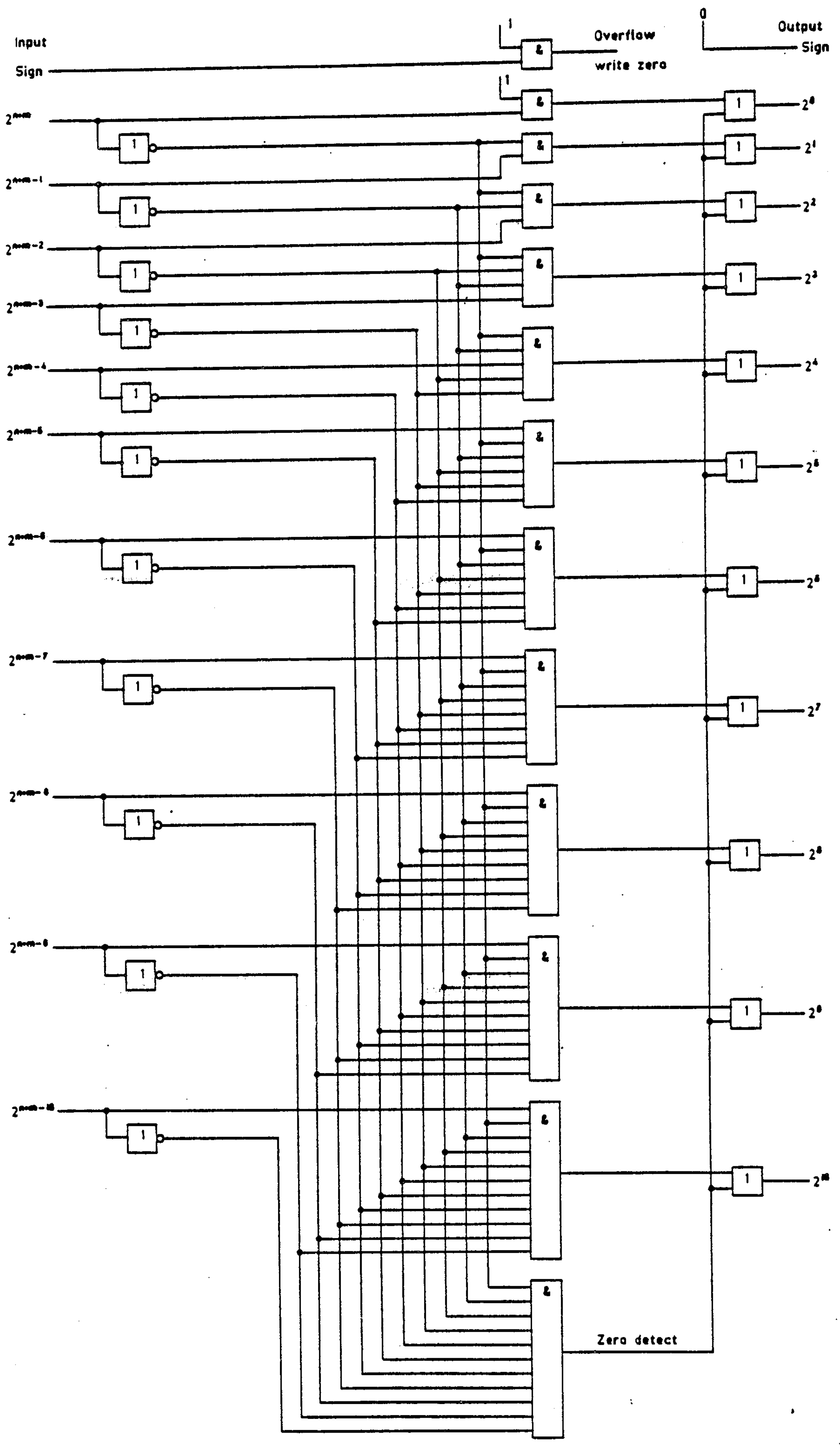
iii) update impulse response in cycle $n_{j+1}$.

Thus equation (2.42) becomes:

$$\mathbf{H}_{j+1} = \mathbf{H}_j - 2.\mu.e_{j-1}.\mathbf{X}_{j-1} \quad \text{... (3.25)}$$

It is seen that the value of error used to update the impulse response in now one cycle removed from the value of the impulse response which produced that error.

The introduction of a limit cycle has a number of effects. Of minor significance is the need of an additional storage element, which is required to hold the value of $x_{j-W-1}$ for a filter with W coefficients. More important is the reduction of the rate of adaptation (approximately halved) and the


Input
Sign
1
Overflow
write zero
0
Output
Sign
2^{n+m}
2^{n+m-1}
2^{n+m-2}
2^{n+m-3}
2^{n+m-4}
2^{n+m-5}
2^{n+m-6}
2^{n+m-7}
2^{n+m-8}
2^{n+m-9}
2^{n+m-10}
2^0
2^1
2^2
2^3
2^4
2^5
2^6
2^7
2^8
2^9
2^10
Zero detect

minimum achievable mean-square error, which has been found from measurements (reproduced in Chapter 5) to increase by approximately 6dB.

Let us define a process cycle as that period required to perform a pair of operations, e.g. a multiply and accumulate, or a read and write. Then, assuming that the convolution process is multiplexed onto a single multiplier-accumulator, for a filter with W coefficients a process cycle is defined to be:

$$t_b = \frac{T}{W} \qquad \dots (3.26)$$

where T is the sample period, i.e. there are W process-cycles in an audio cycle. It may be noted that the throughput, in terms of binary words, is 2W. This nomenclature has been defined for a single filter, whereas the hardware was designed to incorporate two filters (the second filter being a normal transversal type), in which the two convolutions are multiplexed onto a single multiplier-accumulator.

The process cycle becomes the basic unit of time within the sample period, with data changes at most parts of the filter occurring in a half process cycle.

Referring to equation (2.90), by stealing a minimum of three process cycles at the end of each audio cycle, the algorithm can be implemented without introducing a limit cycle. In the first stolen process cycle both the factor f(2μ) and the error are calculated (with parallel processing); in the second the product f(2μ).e is formed. Consequently, with some parallel processing, the update component for the first value in the impulse response

$$\Delta h_0 = f(2\mu).e.x_{j-1} \qquad \dots (3.27)$$

can be added to $h_0$ in the third process cycle before it is used in the convolution of the following audio cycle. The number of coefficients available for use in the convolution is therefore reduced by three. This is the essence of a 'burst-pause' strategy, where update and convolution are performed during the 'burst' period, and a number of process cycles are stolen to form a 'pause', enabling data transfers to take place, as well as forming part of the update term. This technique enables filters to be cascaded without introducing a limit cycle.

A clock waveform, called the '$\Delta$ clock', can be defined to partition the audio cycle into a nominal burst and pause region, shown in figure 3.4. The term 'nominal' is used because within the hardware a number of timing skews cause the actual burst-pause period, allocated to different parts of the system, to vary either side of the $\Delta$ clock.

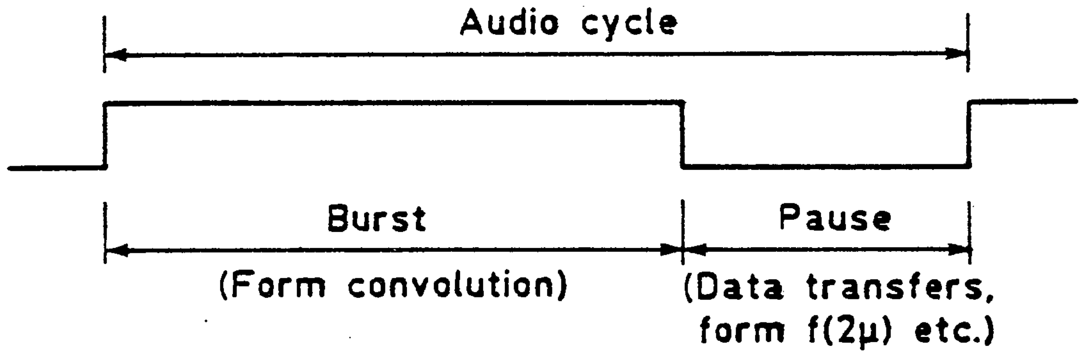


FIG. 3.4 SCHEMATIC OF Δ CLOCK

The transfer of data with a 'pause' must be carried out synchronously, to facilitate the cascading of separate filter units. It is necessary to pass the correct input-data x to each unit, where partial replicas ($r = X^T H$) and sums-of-squares of x ($||X||^2$) are calculated. These must be output from each filter to external accumulators to obtain total values and then returned to each filter, all in a time prior to the three-process-cycle sequence described above. A serial data-transfer and accumulate method was chosen to minimise the number of external pins on a filter. The maximum number of data bits which can be output, accumulated and returned to each filter is a compromise depending on

a) the number of filters which are to be cascaded,

b) the range of SSX over which compensation of f(2μ) will be applied,

c) the precision of the replica and

d) the number of process cycles that are required for a system overhead.

The result is a consequent loss of usable coefficients, as the pause expands to accommodate the data transfer.

As an example, consider a filter having $W = 2^8$ coefficients (i.e. the maximum number in a single filter unit, for this particular hardware solution) which must accommodate a full-scale sine-wave input. The maximum value of SSX is therefore:

$$||x||^2_{max} = W.\left[\frac{g}{\sqrt{2}}\right]^2 = 2^{29} \qquad \ldots (3.28)$$

and sets the upper limit in table 3.1 for a single filter, i.e. $m+n+1 = 29$, and thus fixes the value of C in equation (3.24), i.e. :

$$C = g.k.t_1.t_2 = 2^{29} \qquad \ldots (3.29)$$

where, from previous work, $g = 2^{11}$, $k = 2^0$, $t_1 t_2 = 2^{18}$, and 'C' is a system parameter fixed by the architecture used to generate f(2μ). (N.B. this is simply a repeat of the argument developed in equations (3.13), (3.14) and (3.15)).

Calculating f(2μ) over the full range of SSX would require 30 bits to be serially transferred, taking 15 process cycles, plus a further p process cycles which are needed for serial addition in a cascade mode and the computation of f(2μ).e. Similarly, the formation of a composite replica, to full precision, would require a minimum of (L/2 + p) process cycles, where H is held to L bits precision.

An arbitrary limit was set on the precision of the replica r to be that of the desired signal y, i.e. 12 bits. However, the sign is extended by 2 bits to enable serial accumulation of partial replicas in a cascade mode. Thus 7 process cycles are required for serial transfer. One process cycle was allocated, in p, for external serial addition in a cascade mode. This provides cascading of up to 4 filters, since each synchronous serial addition introduces a half process-cycle delay. A total of 11 process cycles have therefore been allocated to the pause so far. A further 3 were allocated to the convolution multiplier, which utilized a modified Booth's algorithm [310] and introduced an equivalent delay. Finally, 1 process cycle was allocated to the computation of SSX. A total of 16 process cycles were allocated to the pause period. The actual number of useful coefficients in a single hardware filter is therefore reduced to 240. Connecting 4 filters in a fully-cascaded system would then have a maximum of 960 coefficients.

The limit of 12 bits of data plus 2 bits of sign extension for serial transfer also applies to SSX and sets the value n in Table 3.1 to 10. Consequently, for a full-scale sine-wave input, in Table 3.1 m = 18 and n = 10. Calculation of f(2μ) is therefore restricted to a range of 36dB (one bit change corresponds to 6dB). When the input signal falls below the compensated region, f(2μ) remains constant (see Table 3.1); then the maximum size of Δh (the update term) falls as the input level, reducing the speed of adaptation.

The quantisation noise associated with replica truncation ($\sigma_r^2$) can now be determined, for this particular hardware implementation, by substituting $\delta_r$ = 1 in equation (3.6), which is the least-significant digit in the 12-bit number range.

During the pause SSX is calculated for the following audio cycle from:

$$SSX_{j+1} = SSX_j - x_{j-239}^2 + x_{j+1}^2 \qquad \ldots(3.30)$$

This algorithm requires two multiply-accumulates each audio cycle, but introduces a slight complication at initialisation when provision must be made for the first 240 values of $x^2$ to accumulate in the SSX register before adaptation begins. During the pause, convolution of valid data is suspended; it is therefore desirable to utilize the hardware multiplier, within this period, to solve equation (3.30).

## 3.4.3 Q shifting.

Up till this point, only the optimum value of $f(2\mu)$ for speed of adaptation has been considered, which is given by equation (3.21) for this hardware design. Examination of equation (2.105) shows that the misadjustment noise ($\sigma_d^2$) is reduced as the denominator is made more negative; the limiting condition is found from the exact solution of the stability criterion, given by equation (2.102). It is very desirable to achieve this advantage, since the filter can adapt to a better solution in the presence of external noise (given by $\sigma_p^2$). The obvious method is to reduce $f(2\mu)$ directly by a scaling factor q less than unity :

$$f(2\mu) = q.f\left[\frac{2.\mu}{g.k}\right]_{opt} \qquad \ldots (3.31)$$

where, to a good approximation, $0 < q < 1$.

But $f(2\mu)$ has been fixed by equation (3.24) and the choice of fixed-point arithmetic. A simple alternative is to introduce further division of $\Delta h$ by extending the binary-point shifting technique used for normalisation. Defining q to be :

$$q = 2^{-Q} \qquad (Q = 0,1,2, \quad \ldots \text{upper bound}) \qquad \ldots (3.32)$$

and, ignoring the noise components in the system, the update algorithm becomes:

$$H_{j+1} = H_j + f\left[\frac{2.\mu}{g.k}\right]_{apx} . \frac{1}{2^Q.t_1t_2}.e_j.X_j \qquad \ldots (3.33)$$

which effectively reduces $f(2\mu)$.

To implement equation (3.33) with a number of values of Q requires a barrel shifter between the resultant $\Delta h_{apx}$ and the adder which performs the update. A further use for this barrel shifter comes in cascade mode. It was shown in equations (3.28) and (3.29) that, for a single filter with $W = 2^8$ coefficients, the system constant C equals $2^{29}$. If the system is then doubled in W, i.e. to $2^9$, then the constant C is doubled to $2^{30}$; but the value of $f(2\mu)$ has been fixed for a single filter. Consequently, an additional cascade factor $2^{-c}$ must be introduced in similar fashion to that for Q shift.

Finally, the scaler k, described in Chapter 2, (which may be used to improve the precision of arithmetic when the peak of the impulse response is known to be smaller than g by a factor of k), when set to unity for $f(2\mu)$ in equation (3.29) can be similarly implemented, using the barrel shifter and scaling the update component by $2^K$.

Equation (3.33) becomes :

$$2^k H_{j+1} = 2^k H_j + C\left[\frac{2^c}{f(||X||^2)}\right] \cdot \frac{2^K}{2^{(Q+c)} \cdot t_1 t_2} \cdot e_j \cdot X_j \qquad \ldots (3.34)$$

The exact application to the barrel shifter will be shown later.

## 3.5 Linking Internal Numbers and Analogue Measurements.

Although the filter was designed for full-scale sine-waves virtually all measurements are made with white-noise input-signals. This is because the primary application is to hold a loop stable, consequently the performance measurement of most importance is the adaptation across the whole frequency band.

A white-noise signal, as measured on an analogue power meter or represented as expected values of standard deviation, can be related to the corresponding values of $f(2\mu)$ generated in the filter. First, the analogue voltage must be related to the number range of the filter. To obtain good voltage resolution from a-to-d and d-to-a conversion, a peak-to-peak voltage of 20 V was allocated to the total number range of 4096 (i.e. -2048 to 2047 in two's complement arithmetic). The incremental voltage $\delta$ is therefore $4.885\ 10^{-3}$ volts. A power meter, measuring in dBm, assumes a voltage terminated in 600 ohms (i.e. 1 mW corresponds to a voltage of 0.77 V in 600 ohms). Therefore, to convert a power measurement relative to voltage (in dBV) to a power measurement relative to 1 mW (in dBm) the voltage must be scaled by 1.29 (1/0.77), which is equivalent to adding 2.22dB to the voltage representation. Therefore, the input signal (of any waveform), measured in dB relative to 1 mW (dBm), is related to the RMS value in the filter number-range ($\sigma_x$) by:

$$x^2 = 20.\log_{10}(\sigma_x.\delta) + 2.22 \text{ dBm} \qquad \ldots (3.35)$$

Table 3.2 shows the relationship between $x^2$ and $f(2\mu)$ used in a single filter with W = 240 coefficients, where

$$\log_2(E(||X||^2) = \log_2(240.\sigma_x^2) \qquad \ldots (3.36)$$

| Input power | Values within filter. | | | $\log_2\left[f\left[\frac{2.\mu}{g.k}\right]_{apx}\right]$ | |
|---|---|---|---|---|---|
| $x^2$ dBm | $\log_2(E(\|\|X\|\|^2))$. | $\sigma_x$. (W=240) | $\log_2(\sigma_x)$. | $C=2^{29}$ | $C=2^{28}$ |
| 19.49 | 29 | 1456.65 | 10.55 | overflow | overflow |
| 16.48 | 28 | 1057.68 | 10.04 | 0 | overflow |
| 13.46 | 27 | 747.82 | 9.55 | 1 | 0 |
| 10.45 | 26 | 528.79 | 9.04 | 2 | 1 |
| 7.45 | 25 | 373.91 | 8.55 | 3 | 2 |
| 4.45 | 24 | 264.40 | 8.04 | 4 | 3 |
| 1.43 | 23 | 186.96 | 7.54 | 5 | 4 |
| -1.58 | 22 | 132.20 | 7.04 | 6 | 5 |
| -4.59 | 21 | 93.48 | 6.54 | 7 | 6 |
| -7.60 | 20 | 66.10 | 6.04 | 8 | 7 |
| -10.61 | 19 | 46.74 | 5.54 | 9 | 8 |
| -13.62 | 18 | 33.05 | 5.04 | 10 | 9 |
| -16.63 | 17 | 23.37 | 4.54 | 11 | 10 |
| -19.64 | 16 | 16.52 | 4.04 | 11 | 11 |
| -22.65 | 15 | 11.68 | 3.55 | 11 | 11 |

Table 3.2 Relation between powers within a single filter.

A column for $C = 2^{28}$ has been included, as this was provided as a bonding option in the hardware design and is described in Chapter 5.

Within the compensated region, the approximation for $f(2\mu)$ remains constant over a 3dB input-signal range. Its effect on the update algorithm can be modelled as an effective q factor change. i.e. from equations (3.21) and (3.31)

$$q = f\left[\frac{2.\mu}{g.k}\right] . \frac{2^{\text{trunc}(\log_2 ||X||^2)}}{g.k.t_1.t_2} \qquad \ldots (3.37)$$

Therefore, when $||X||^2 = 2^n$ $(n = 0,1,2, .. 11)$, then $q = 1$.

When $||X||^2 \rightarrow 2^{n+1}$, then $q \rightarrow 0.5$.

Since $||X||^2$ has a chi-square distribution q will behave nonlinearly as $f(2\mu)$ changes value, with constant input-power. The effect on performance is more easily measured on the hardware than modelled analytically.

A similar q-factor approach can be taken when the input level falls below the compensated range. Then, as $||X||^2$ continues to fall q falls proportionally, with its corresponding reduction in speed of adaptation and improved performance with a noisy reference channel.

## 3.6 Functional Partitioning of the DAF.

Figure 3.5 shows the simple parallel-processing requirement for a single adaptive filter, partitioned into three blocks (A,B,C). Block A details the x input-output, and storage of vectors X and $H^*$. Block B shows the convolution process and generation of $f(2\mu)$, and block C the calculation of the update component $\Delta h$. Included are the binary-point shifting operations $(g, k, t_1, t_2)$ and their associated quantisation-noise contributions due to truncation. Minimal timing information has been provided in this diagram, showing only ( by clock pulses) the number of operations per audio cycle.

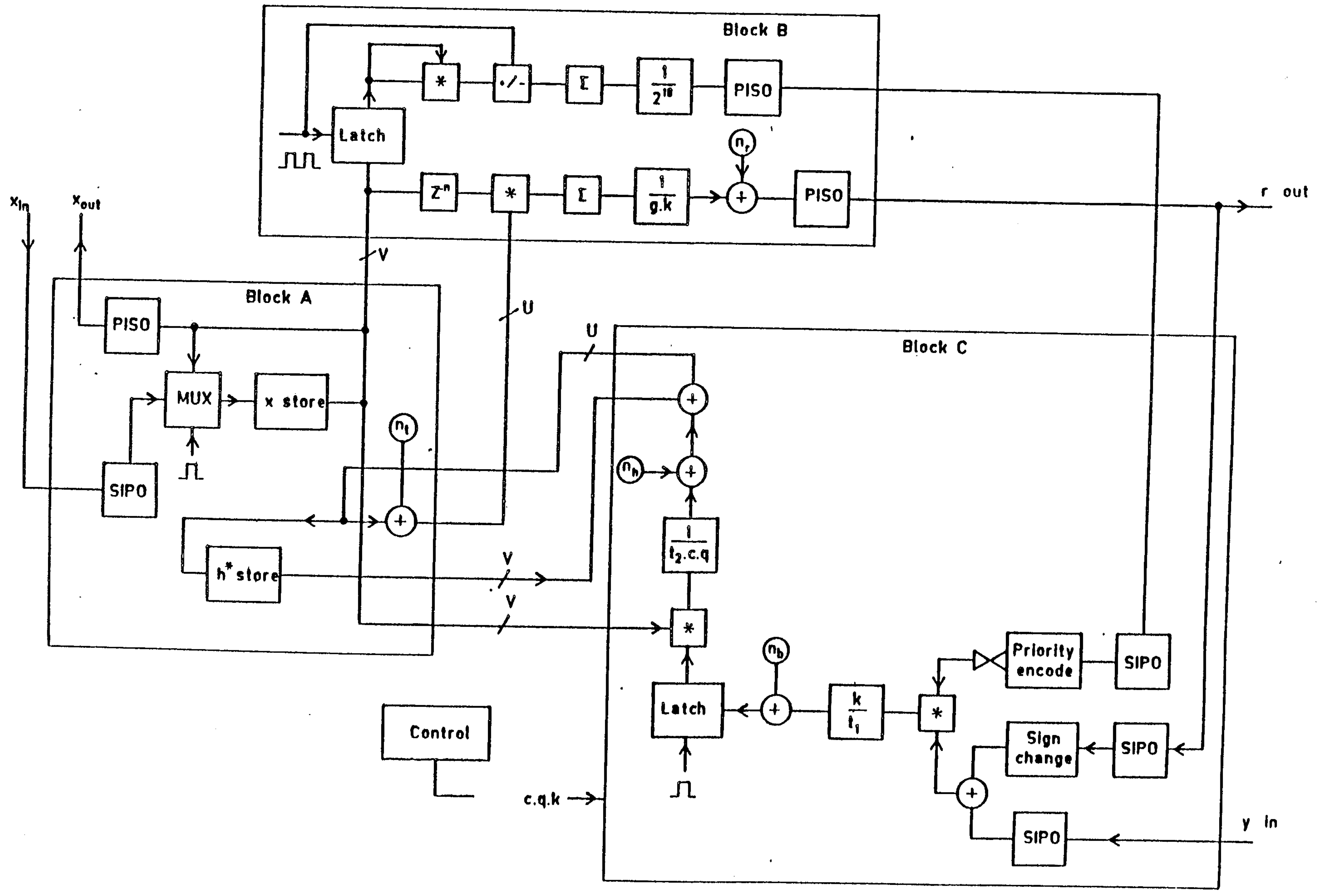


FIG. 3.5 SCHEMATIC OF PARALLEL PROCESSING REQUIREMENT OF DAF

The partitioning described, which evolved from a study of the topological layout of data processing and data transfer within a filter, was used in the hardware implementation. The topological study, which is very difficult to describe accurately, consisted of a 'paper' experiment, in which X, $H_f$ and $H_b$ were written on slips of paper and stored on rotating wheels (modelling shift-registers), with data passed over buses drawn on a paper base. The result was the synchronous transfer of data between separate logic areas, during an audio cycle, as shown in figure 3.5, multiplexed onto two buses labelled U and V. This provided a significant step forward in the commitment to an LSI design and allowed one solution of an adaptive filter with efficient use of silicon area, high yield and low cost.

The pin count for these buses set a limit on the precision of data transfer, where each was a maximum of 16 bits (set by a limit of 64 pins on block C). Thus, the maximum precision for storage of $H^*$, within the filter was 16 bits.

## 3.7 The $t_1.t_2$ Product.

It has been previously stated that, from consideration of equation (2.105), $t_1 = 2^0$ gives minimum misadjustment noise . It is now possible to examine alternative values, based on an approach requiring maximum speed of adaptation over a wide range of input level, constrained by having the correlation multiplier with less than full precision.

Consider a reference path which produces low noise and is lossless. Before adaptation begins, assuming that initially $H^* = 0$, the RMS value of the error signal is equal to that of the input signal. From Table 3.2, the expected RMS value on the $2\mu e$ bus (i.e. the output from a 12-by-12 bit multiply) over the compensated region, can be found from the sum of columns 4 and 5, i.e. for this initial condition :

$$10.55 < \log_2(f(2\mu).e)_{rms} < 15.54 \qquad \ldots (3.38)$$

N.B. as $\log_2(\sigma_x)$ passes through a value recorded in Table 3.2, the output $\log_2(f(2\mu)_{apx})$ changes by an integer amount, remaining constant until $\log_2(\sigma_x)$ passes through the next recorded value. Hence, an overflow occurs for $\log_2(\sigma_x) > 10.55$.

If the peak-to-rms ratio $(g/\sigma_e)$ is $2^2$, then the full dynamic range on the $2\mu e$ bus covers 18 bits (sign + 15 +2), which must be input to an 18 by 12 bit multiplier to form the product $(2\mu e).x$, without loss of precision. Block C in figure 3.5 shows two separate multipliers to perform the calculation $f(2\mu)$.error.x, but $f(2\mu)$.error is required only once per audio cycle and can be formed in the 'pause'. Both products were therefore multiplexed onto a single 12-by-12 bit multiplier. The dynamic range which could accommodate the $f(2\mu).e$ word without peak clipping is therefore:

$$3 < \log_2(f(2\mu).e)_{rms} < 9 \qquad \dots (3.39)$$

i.e. sign + 9 + 2 = 12 bits

This would require $t_1 = 2^6$ , leaving six bits below the binary point which are truncated, thus introducing a quantisation noise ($\sigma_b^2 = 1/12$). Figure 3.6 shows a representation of the number range on the $f(2\mu).e$ bus before and after division and truncation.

```
         |<-------------      sign.log2(|f(2μ).e|)        ------------------>|
         |
         |            |<-- No. range at start -->|
         |            |      of convergence      |
         |
t1 = 2^0 |s 19 18 17 16 15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0
t1 = 2^6 |            s 10  9  8  7  6  5  4  3  2  1  0.-1- 2 -3 -4 -5 -6
         |            |<--   selected output range    -->|
```

Figure 3.6 Input-output number range on $f(2\mu).e$ bus.

The actual speed of adaptation is dependent on hardware implementation, which will include overflow detection and correction. A computer program was written which exactly simulated the numerical processes in the LSI architecture. Figure 3.7 is derived from that program and shows the relationship between input level and the time taken to reach 50% of maximum adaptation, for a number of $t_1$, $t_2$ values. Although the hardware was designed using $t_1 = 2^6$ and $t_2 = 2^{12}$, it is seen from figure 3.7 that there is little advantage in increasing $t_1$ above $2^4$. The corresponding effect on performance will be described in Chapter 5.

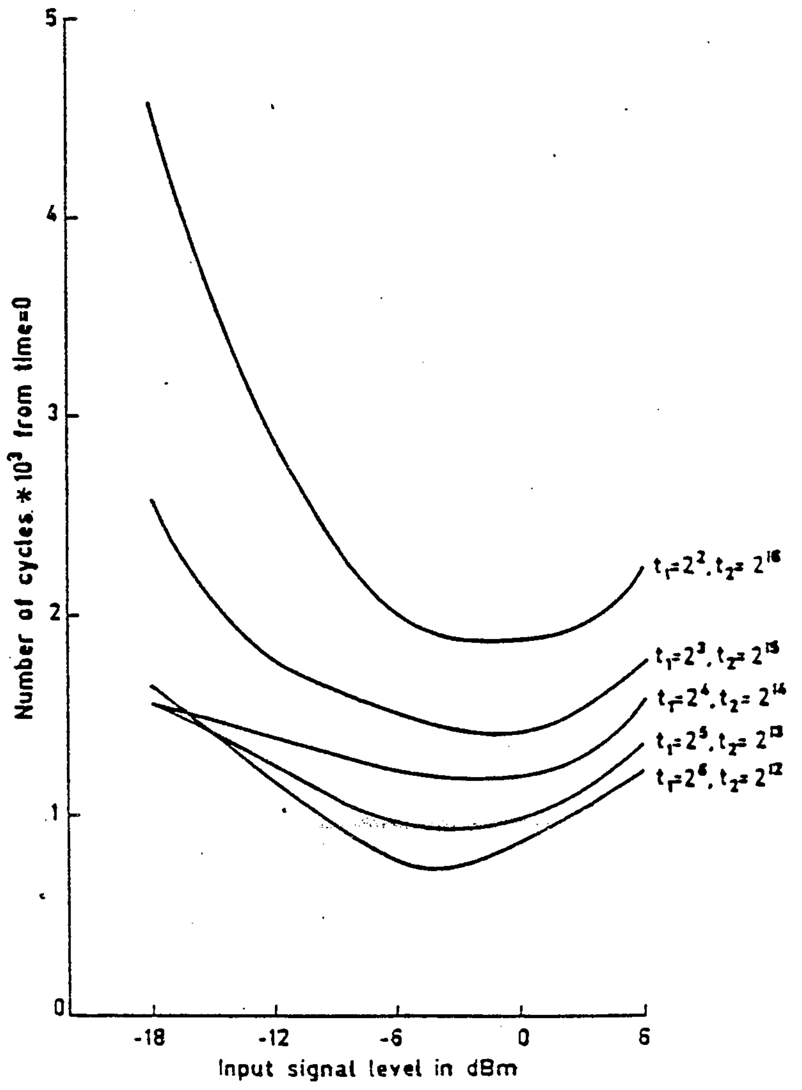


FIG. 3.7 GRAPH OF NUMBER OF CYCLES TO REACH 50% OF MAXIMUM ADAPTION AGAINST INPUT SIGNAL LEVEL FOR RANGE OF $t_1, t_2$

The limit on the internal storage size of $H^*$, and hence $\Delta h$, was set by the maximum size of the U bus, i.e. 16 bits. 12 bits are required for the number range g, leaving 4 bits below the binary point. Figure 3.8 shows the 16 bits selected from the $\Delta h$ bus, with $t_2 = 2^{12}$. The quantisation noise ($\sigma_h^2$) introduced by the truncation of values below the least-significant digit ($2^{-4}$) is similar to equation (3.2), i.e. :

$$\sigma_h^2 = \frac{2^{-8}}{12} \qquad \ldots (3.40)$$

| | $\longleftarrow$ sign.$\log_2(\lvert f(2\mu).e.x\rvert)$ $\longrightarrow$ | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $t_2 = 2^0$ | s | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 ... |
| $t2 = 2^{12}$ ($q = 2^0$) | s | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0. | -1 | -2 | -3 | -4 | | | |
| | $\longleftarrow$ bit selection for $\Delta h$ $\longrightarrow$ | | | | | | | | | | | | | | | $\leftarrow$truncated$\rightarrow$ | | | |

Figure 3.8 Bit selection from $\Delta h$ bus.

N.B. 8 bits are truncated from the full precision bus to form the output bus.

## 3.8 Truncation in Two's Complement.

Two's-complement numbering was used in the filter, but this has the disadvantage of introducing an offset when truncated, given by:

$$\text{offset} = -\frac{2^n - 1}{2^{n+1}} \qquad \ldots (3.41)$$

where n is the number of bits removed by truncation. The offset tends to $-2^{-1}$ as n increases. In most cases this offset is little more than a nuisance during measurement, except for the LMS algorithm which is an integrating loop, i.e. it causes accumulation of $\Delta h$ to a final value. It is essential to remove this offset to prevent the response simply accumulating to its most negative value. A reasonably simple rounding technique is to add the first bit below the truncation point to the total data value. The corrected offset is given by:

$$\text{corrected offset} = \frac{1}{2^n} \qquad \ldots (3.42)$$

which tends to zero as n increases.

It was shown above that, for the f(2μ).e bus n = 6 and for the Δh bus n = 8, which would produce a small offset. This offset, plus any d.c. component present in the output from the reference path but not in the input signal, causes the impulse response $H^*$ to take up a low-frequency component, which is not present in the reference path. This is demonstrated on the upper trace of figure 3.9, which is an oscillogram of an arbitrary impulse response, measured after approximately one hour. Complete removal of this erroneous condition can be achieved by introducing a small (one bit) offset into the input vector **X**, providing a reference. The lower trace of figure 3.9 shows the same impulse response, but with an offset applied to **X**.

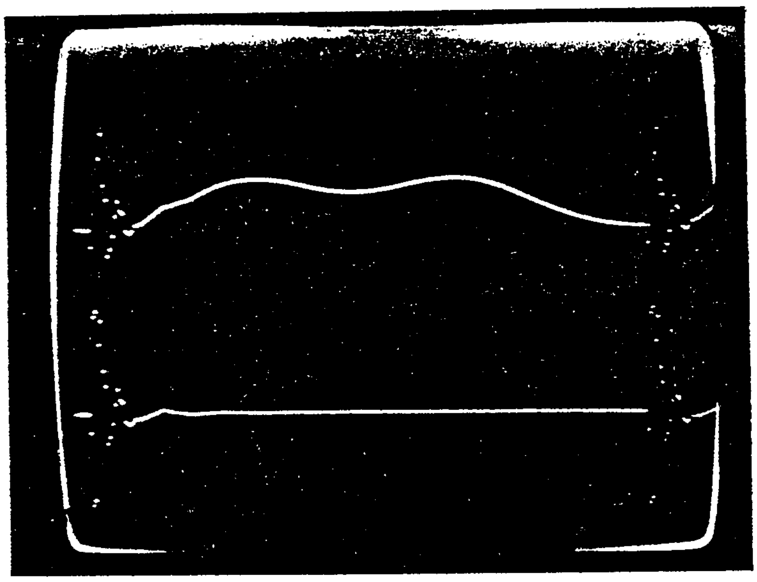


Figure 3.9 Oscillogram of an impulse response
(a) with error offset
(b) with corrected offset.

It was also necessary to bandlimit the input frequency range to

$$\frac{1}{WT} < f < \frac{1}{2T} \qquad \text{... (3.43)}$$

where T is the sample period. Frequencies below the lower limit cause the filter impulse-response to become occasionally unstable, producing a loss of a few dB adaptation.

## 3.9 Foreground-background. Multiplexing Two Filters.

Early experiments, with a prototype loudspeaking telephone, showed that it was essential to have a technique preventing the filter coefficients (which hold the loop stable) from being corrupted, or worse, totally destroyed, during the learning mode by intermittent double-talk, i.e. introducing a time-dependent noise-component $\sigma_p^2$ in equation (2.105). One method, which achieved this objective, was proposed by Ochai [303] and employed two filters connected in a foreground-background mode. A normal transversal-filter was placed in the transmission path, or foreground, the coefficients for this filter being obtained from a background adaptive-filter. Coefficient transfer can be initiated when a comparison of the error output from each filter shows that the background filter has better performance than the foreground filter. The technique has the advantage of being able to track a changing impulse response rapidly with the adaptive filter, but it allows a control option, based on performance measurement, before discarding the previous estimate of the impulse response.

It was decided to incorporate this technique into the filter architecture, such that the convolution of two separate impulse-responses and a single input vector would be multiplexed onto the multiplier-accumulator shown in block B of figure 3.5. The total number of coefficients is arranged to be a power of two, because of availability of low-cost crystals for binary multiples of the fundamental sampling frequency of the telephone bandwidth (8 kHz). A system with two impulse responses, each consisting of $2^9$ coefficients, would have 244ns to do a read, multiply and accumulate (corresponding to 4.096 $10^6$ multiply-accumulates per second), requiring $2^{10}$ process cycles. Designing a ripple-through multiplier to achieve this speed directly in LSI would restrict the fabrication process to either bi-polar technology, with its high heat dissipation, or CMOS, with its need for large surface area. Both are undesirable for a low-cost system.

The problem was overcome by using a multiplier based on the Modified Booth's Algorithm [310]. In this technique the multiplier is split into overlapping triplets which are decoded to operate on the multiplicand. It requires a time skew of n/2 clock cycles between the input and output latches, where n is the number of bits in the multiplier. The x input is 12 bits, introducing 6 clock cycles delay (corresponding to 3 process cycles). The number of bits in the multiplicand h determines the quantisation noise $n_t$ (with power $\sigma_t^2$) introduced to the system.

An initial design allocated 12 bits from the 16 bit $H^*$ store. Later this was upgraded to the full 16 bits, when LSI designers showed that it could be incorporated for a small additional area cost. The improvement obtained is described in Section 5.5 on performance of a single filter-section.

Data values X, $H_f$ and $H_b$ (where the subscript refers to either foreground or background) are transported from the storage device to the convolution device on the U and V buses (figure 3.5). It is therefore relatively simple to multiplex the correct delays before the inputs to the multiplier, so that the effect of timing skews can be removed.

Table 3.3 lists the data types and the number of bits precision which were used in the final hardware design.

## 3.10 Conclusions.

This chapter has discussed the design evolution of hardware, based on desired characteristics conflicting with physical limitations imposed by LSI size, cost, pin-count etc., which was carried out before the actual circuit design began. Enough information is presented to specify completely the performance, in terms of adaptation, using the theory developed in Section 2.3.5. However, another design objective was to achieve the optimum rate of adaptation, which could be reduced under external control. The actual value is not amenable to analysis, except under some simplifying approximations [222] described in Section 5.5, although the coefficient $f(2\mu)$ which produces this condition is easily found. It is necessary to model numerically (or build) the system, with all its nonlinear overflow detection and correction, to obtain values for the rate of adaptation.

The decision to build LSI devices was based on the arguments above, which were translated into logical functions described in Chapter 4, consequently the physical and computer modelling were carried out in parallel with the LSI design. Only one parameter proved to be suspect in the arguments presented above, this was the ratio of the values $t_1$, $t_2$. It will be shown, in Section 5.4, that the resulting loss of adaptation is small when the input signal has a broadband spectrum.

| Data type | No. of bits precision |
|---|---|
| x: input | 12 |
| y: desired input | 12 |
| r: replica output | 12 |
| e: error output | 12 |
| SSX: sum-of-squares of x | 12 |
| $h_b$: adapt impulse-response | 16 |
| $h_f$: stored impulse-response | 16 |
| convolution multiplier | 12 by 16 |
| correlation multiplier | 12 by 12 |

Table 3.3 Precision within hardware filter.

# 4 HARDWARE IMPLEMENTATION OF THE DAF.

## 4.1 Introduction.

The architecture for a single DAF unit, to be implemented as a low-cost LSI solution, was based on partitioning the total logic into three areas which were interconnected with both parallel and serial data buses as shown schematically in figure 4.1. N.B figure 4.1 corresponds to the functional layout in figure 3.5, but incorporates the additional 'foreground' FIR filter (Section 3.9). The technique of partitioning and interfacing on buses proved a major advantage in providing flexibility in use of the system, but a disadvantage in terms of the number of pins required for each device. Referring to figure 4.1, the functional areas are :

A: Memory (MEM), comprising 3 circulating shift-registers to store vectors $\mathbf{X}$ (the input signal), $\mathbf{H}_s$ (stored or foreground impulse-response), and $\mathbf{H}_a$ (adapt or background impulse-response).

B: Multiplier-accumulator (MAC), comprising a multiplier with it's output multiplexed between 3 separate accumulators. One accumulator stores the convolution term $\mathbf{X}^T\mathbf{H}_a$, another is for the convolution term $\mathbf{X}^T\mathbf{H}_s$ and one is for the vector norm $||\mathbf{X}||^2$ (SSX).

C: Convergence (CONGE), comprising a number of functional areas,

i) formation of two errors, for the foreground and background filters, i.e. equation (2.78 or 2.79).

ii) generation of $f(2\mu)$, as described in Section 3.4.1.

iii) correlation multiplier, to form the two products described by equation (2.91).

iv) scaling and Q-shifting, to implement the coefficients k,q and c, described in Section 3.4.3.

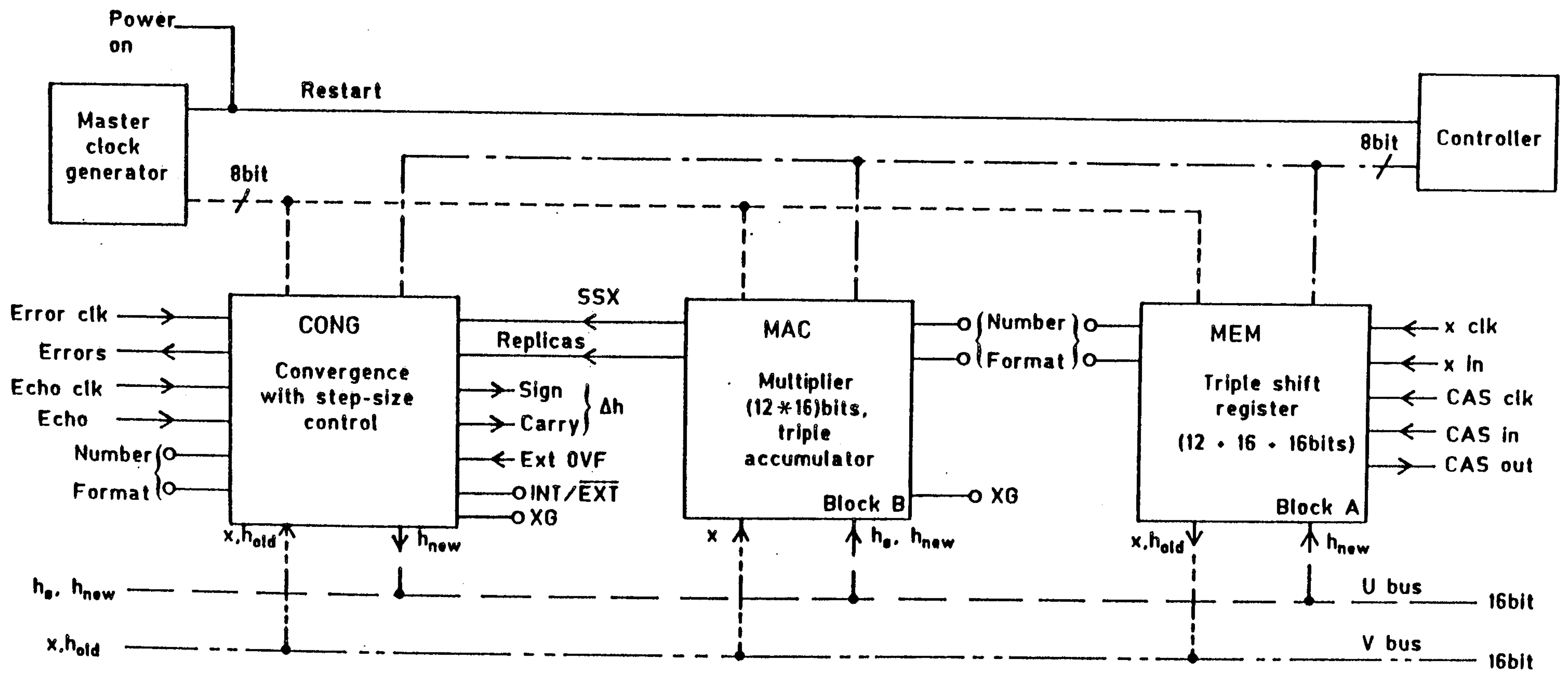


FIG. 4.1 PARTITIONING AND INTERCONNECTION OF A SINGLE DAF

v) updating $H_a$, i.e. equation (3.34).

Each of the three areas, in figure 4.1, was intended to be a separate LSI device having suitable buffers to read, or write, to appropriate buses. A slight change was made after consideration of the device-yield to silicon-area ratio, which showed a lower cost by splitting the memory into two identical devices, with half a binary word in each. Additionally a pin was required to designate the device containing the MSB and provision was made for the system to interface to a multiplicity of numbering schemes.

The final system incorporated all the elements described in Chapter 3, i.e. Q-shifting, cascading etc, plus an optional facility to reorganize the LSI devices on a larger U and V bus, with additional storage and external adders, to implement very low q-factors.

The task of implementing the LSI architecture was executed in four overlapping phases. Initially a paper exercise to produce the functional design was carried out. The output from this phase was input to three separate filter implementations i.e., TTL emulation, computer simulation and LSI design. The logical working of the three approaches to the single functional design enabled each to be compared to the other and to the theoretical performance predicted in Section 2.3.5. The result of cross-checking three systems gave a very high degree of confidence in the final architecture.

The functional layout was produced by considering each of the three logic areas in figure 4.1 and sub-dividing into smaller logical entities having well-defined interfaces between sub-areas of logic. For example generation of $f(2\mu)$, described in Sections 3.4.1 and 3.4.2, requires a serial input SSX, with overflow protection and correction, and is input to a sub-area of logic which has an input to output relationship defined by table 3.1 in Chapter 3. In this manner each of the three solutions (i.e. TTL emulation, computer simulation and LSI) can implement the logic in a way best suited to the resources available to the individual technique. Then each filter implementation can be compared at the input or output of any interface.

A complete TTL emulation of the filter was built, where each of the sub-areas was modelled and interconnected on either a system bus (defined to be those shown on figure 4.1) or a local bus. The model was a true emulator, designed to be able to exchange a group of boards in TTL for a single board containing the LSI circuit. This served the dual role of testing the filter logic in real time and a being valuable tool to debug the LSI circuit.

Each of the real-time systems (i.e. the emulator and LSI ) was designed to operate as a 'state-machine', i.e. one which could be run at any master clock-frequency up to a maximum, limited by propagation delays on buses and devices in critical paths. A computer simulation program was written (the simulation is not included in this thesis because of its bulk), which modelled each logic area, numerically, exactly as the real-time models (but not containing any explicit timing information except at process-cycle intervals), so that values on the interfaces between logic areas could be compared between the TTL model (or LSI) and the computer program. The real-time and simulation methods were cross-checked by supplying each with identical data-values (generated on the computer). If a difference was detected, the numeric values at the interfaces were compared and the fault traced. The technique of linking hardware and software proved a very powerful debug tool, which, because of the self-correcting nature of an adaptive filter (giving partial working) considerably reduced the time taken to obtain a working design with a performance matching theoretical prediction. Finally the logic areas were converted to LSI, based on 3μm rules using NMOS technology, which features high speed and gate packing-density.

## 4.2 Master Clock-generator.

The master clock-generator, shown in figure 4.1, was a separate circuit (actually built in TTL for the emulator and on a CMOS ULA for the LSI). This enabled one clock generator to drive any number of DAFs in a system, with suitable buffering, and avoid synchronization problems. The waveforms, described in Appendix B.1, were derived from a crystal oscillator (16.384 MHz for 8 kHz sampling) and comprise of an initialisation, clock called SYNCLR, the burst-pause clock, called DELTA ($\Delta$) and 6 others which are used to define various process-cycle timing within the system.

## 4.3 The Convergence Circuit.

As stated in Section 4.1, a number of functions are performed by this circuit (which are covered in detail in Appendix B.2). The following is a brief description of the functions implemented on the device.

### 4.3.1 Local- strobe-generator.

A number of locally generated strobes are required to gate data serially onto the circuit, determine actual burst-pause periods for different logic areas and provide latch pulses at specific times. These are all derived from waveforms from the master clock-generator.

### 4.3.2 Formation of input-level compensation factor $f(2\mu)$.

The $f(2\mu)$ factor, in a number system which is normalised to unity, is (equation 2.50) approximately the reciprocal of the total energy stored the X-input register (SSX). SSX is calculated on the multiplier-accumulator chip (MAC in figure 4.1), using equation (3.30), and is serially output to the convergence chip (CONGE in figure 4.1) arriving on pin SIX (serial input X). The system is prevented from starting up for W audio cycles (by holding SYNCLR low in the master clock-generator), while the initial value of SSX is formed ($W <= 240$ for a single filter but is fixed at 240 coefficients for the current LSI implementation). In the event of cascading systems the SSX outputs from each filter, up to a maximum of 4, are summed externally and the result applied to each convergence chip, where the scale factor $2^{-c}$ (see equation

3.34) is introduced via the control bus. i.e. SSX is twelve bits long at the convergence chip irrespective of the number of cascaded filters, therefore the filter must be informed to scale $\Delta h$ accordingly.

Input-overflow protection is included after the whole SSX word has been input, by examining the sign bit (bit 11), which must be zero since the number is always positive. If overflow is detected, the filter is prevented from making any changes to $H_{adapt}$ for the following audio-cycle. The number is then truncated to the largest power-of-two and the bit order reversed (forming the reciprocal). The resultant value is the solution of equation (3.24). Additionally a WIDE/NARROW facility is included, to restrict the compensated range from twelve bits to eight bits, to be used where the signal-to-noise ratio of the path being modelled is low. This is equivalent to reducing the 'q-factor' for low-level signals (see last paragraph of Section 3.5).

### 4.3.3 Serial-data input and error output.

The data input refers to the acquisition of both replicas, stored and adapt (calculated on the multiplier-accumulator), plus the echo, which are transferred serially. The term 'echo' is used exactly as 'desired signal' and refers to the signal out of the reference path. Subtraction of replicas from the echo forms two errors, adapt and stored. The adapt error is routed back into the filter to form part of the update algorithm and both errors are output from the filter. A facility has been provided to enable a DAF to interface to 4 of the common numbering schemes, i.e. two's complement, offset binary, and their complements. A detailed description is given in Appendix B.2.3.

### 4.3.4 Correlation multiplier and scaler.

The correlation multiplier forms two separate products in each audio-cycle. One is the scaler product:

$$f(2\mu e) = f(2\mu).\text{error} \qquad \dots (4.1)$$

and the other is the vector product:

$$\Delta \mathbf{H} = f(2\mu e).\mathbf{X} \qquad \ldots (4.2)$$

(calculated serially) as required by equation (2.91). Section 3.4.2 outlines the argument for computing equation (4.2) during the pause. A detailed description is given in Appendix B.2.4. Overflow detection and correction is carried out on the output of the multiplier, because at this point the range of $\Delta h$ is invariant. A barrel-shifter is used after the multiplier to scale the variables $f(2\mu)$ by k and $f(2\mu e)$ by q and the cascade level. Finally, the old impulse-response is updated by adding it to $\Delta h$ in a 16-bit adder (see Appendix B.2.5).

4.3.5 The control bus (CBUS).

The function of this part of the convergence circuit is to provide an interface between the adaptive filter and an external controller. For the latter, this can range from a simple set of switches to a microprocessor. The interface provides a mechanism to change the filter's operating parameters in response to varying external conditions and enable maximum utilization of performance.

Control data is multiplexed onto the 8-bit bus (C0 to C7) and addressed to one of two internal registers. Provision to address two separate adaptive filters has been incorporated, to accommodate the system requirements of the LST (i.e. adaptive filters modelling two different reference-paths). A detailed description is given in Appendix B.2.6. Table 4.1 shows the CBUS pin designations and the corresponding signal description. Two outputs are provided on C7 to inform the controller whether a value of $H_{adapt}$ has fallen between full scale and half full-scale. This can be used to operate the DAF in it's optimum number-range by externally adjusting the gain of the reference path being modelled.

| C7 o/p | C6 i/p | C5 i/p | C4 i/p | C3 i/p | C2 i/p | C1 Filter Address | C0 Register Address |
|---|---|---|---|---|---|---|---|
| $H_{adapt}$ > HFS* < FS | Update | Recirc | K | Q1 | Q0 | Address bit | 0 |
| $H_{adapt}$ OVF | Down-date | ZeroH | $\overline{W}$/N | Cas1 | Cas0 | Address bit | 1 |

* HFS - half full-scale, FS - full scale

Table 4.1 Control Bus Signal Descriptions

Referring to the CBUS lines in Table 4.1 then:

| | |
|---|---|
| C0 | - is an internal register address to demultiplex input-data on C2 to C6 and multiplex output-data on C7 |
| C1 | - is a filter address bit, used in conjunction with another pin designated AC/NEL (<u>AC</u>oustic/<u>N</u>ot<u>EL</u>ectric), to permit control of two separate DAFs. |
| Q1,Q0 | - is a 2-bit word to specify the q-factor value which controls the speed of adaptation. |
| Cas1, Cas0 | - is a 2-bit word to specify the number of DAFs that are cascaded (00 = single, 01 = dual, 10 and 11 = quad). |
| K | - sets the gain of $H_{adapt}$ relative to that of the reference path (0 = equivalent, 1 = multiply by 2). |
| $\overline{W}$/N | - restricts the range of compensation ($f(2\mu)$), for adaptation on a noisy reference path. |
| Recirc | - stops adaptation, i.e. $\Delta H$ is set to 0, thus $H_{adapt}$ recirculates. |
| ZeroH | - initialises $H_{adapt}$ to zero. |
| Update | - copies the values of $H_{adapt}$ into $H_{store}$. |
| Downdate | - initialises $H_{adapt}$ to values in $H_{store}$. |
| $H_{adapt}$ >HFS < FS | - is set if any value in $H_{adapt}$ exceeds half full-scale but does not overflow. |
| $H_{adapt}$ OVF | - is set if any value in $H_{adapt}$ overflows. |

## 4.4 Multiplier and Triple Accumulator.

The circuit is required to perform two convolutions, in a serial manner, to determine the stored and adapt replicas ($r_s$ and $r_a$), plus calculation of the sum-of-squares of the input-signal x (SSX). Replicas $r_a$ and $r_s$ are checked for overflow, scaled by g.k, truncated, rounded and output serially to the convergence device. Each convolution requires the summation of 240 products per audio cycle, during the burst period. The timing allows for the formation 256 product pairs. Thus it is relatively simple to multiplex the two products required to solve for SSX (equation (3.30)) into the pause period when garbage products are formed. A similar argument applies to the adders required. Two are needed for the convolution, but the arithmetic for SSX (equation (3.30)) can be multiplexed onto one of the adders in the pause period. A detailed description is given in Appendix B.3.

A single multiplier is used, which is 12-by-16 bits. The throughput is 512 products per audio-cycle. At the 8 kHz sample rate, the time available for each product is 238.4 ns. A ripple-through system would, necessarily, require high heat dissipation in NMOS technology, making it impossible to fabricate. To overcome this problem a Modified Booth's Algorithm multiplier [306] was used, which requires a time skew of 6 clock-cycles between input and output, but presents few problems for an NMOS implementation.

Twelve bits from the sum-of-squares of x are selected according to the status of a pin XG. Bits 29 to 18 for XG = 0 and bits 28 to 17 for XG = 1 (described in Section 3.4.2). Under normal operating conditions no abnormal overflow can occur for XG = 0, since the maximum value of SSX that can be present (from full-scale input) is $240.2048^2 = 2^{29.907}$. A logic 1 on bit 29 is a valid overflow, translating to bit 11 (the sign bit) on the SSX output. This is not the case for XG = 1 and, if overflow is detected, logic 1 is forced onto bit 11, to be acted upon in the convergence device.

## 4.5 Memory.

The function of the memory circuit is to store the 3 vectors X, Hadapt and Hstore. Each H vector is 256-words long, although only 240 contain useful data. The basic structure utilizes circulating shift-registers. The technique is simple, requiring only the generation of various strobes to read from, or write to, fixed physical locations. An alternative solution could have used random access memory (RAM), requiring more complex address decoding. The RAM storage approach was used in the computer simulation. If a RAM structure had been used, it would have resulted in a physically smaller LSI device, since a RAM store needs fewer FET devices than the D-type flip-flop used in the shift register. The X vector is 257 words long, providing the time slip required (between X and H) for convolution. Additionally, the memory circuit must take in new x values and provide the appropriate old x value for cacading filters together. Finally, it must transfer data between the two H vectors, as directed by the external controller via the CBUS.

## 4.6 Cascading.

The benefit of providing the facility to cascade filter sections together is to extend further the length of impulse response which can be modelled. There is a particular economic advantage in a system, like the LST, where it is necessary to model two separate paths with different characteristics. One is the acoustic path, from loudspeaker to microphone, the other the telephone network. The latter, if terminated in a handset, is likely to have an impulse response less than 30 ms [203]. The acoustic path will have an impulse response considerably longer than that which can be accommodated in this filter design at the 8 kHz sample-rate (an analysis of this problem is discussed in Chapter 6). Hence, the cost of the LST can be kept to a minimum by providing just sufficient filter sections to model each path.

The filter has been specifically engineered to cascade either 2 or 4 sections and still retain the optimum learning performance. Three filter-sections can be cascaded, but the parameter scaling for f(2μ) will be that of a 4-section filter, giving some loss in speed of adaptation relative to a correctly scaled 3-section filter.

Audio signals that are input and output from the filter are transferred serially (to reduce the pin count) via pins CASIN and CASOUT, while all internal processing is done with data words in parallel. Therefore, to construct correctly sequenced X and H vectors, it is necessary to extract the appropriate element from the X vector and transfer it to the next section (with parallel-to-serial-to-parallel conversion), plus any data format conversion) so that the sequence of vector elements is unbroken. Assuming that $x_j$ is the newest value in the X vector, then the first value in each filter section is:

| filter section | DAF-1 | DAF-2 | DAF-3 | DAF-4 |
|---|---|---|---|---|
| first X element | $x_j$ | $x_{j-240}$ | $x_{j-480}$ | $x_{j-720}$ |
| first H element | $h^0$ | $h^{240}$ | $h^{480}$ | $h^{720}$ |

Each section will produce partial convolutions and SSX, which must be accumulated externally and the sum transferred back to the appropriate input. Figure 4.2 shows a simplified block diagram of a cascade quad. The external serial-adders produce a half-process-cycle delay, giving one-process-cycle delay for the quad connection. The partial-replica output from each section comprises 14 bits, 12 bits of data plus 2 sign-extension bits. Thus, by selecting the correctly delayed detection waveform (see Appendix B.2.1 for generation of DSTBR), via the control bus, the lower 12-bits of the final value of replica are returned to each filter section. Overflow can be checked by looking for differences the 3 MSBs on the convergence circuit.

The hardware solution is slightly different for SSX. Since the signal x will, in general, be continuous, it can be assumed that the total energy in a

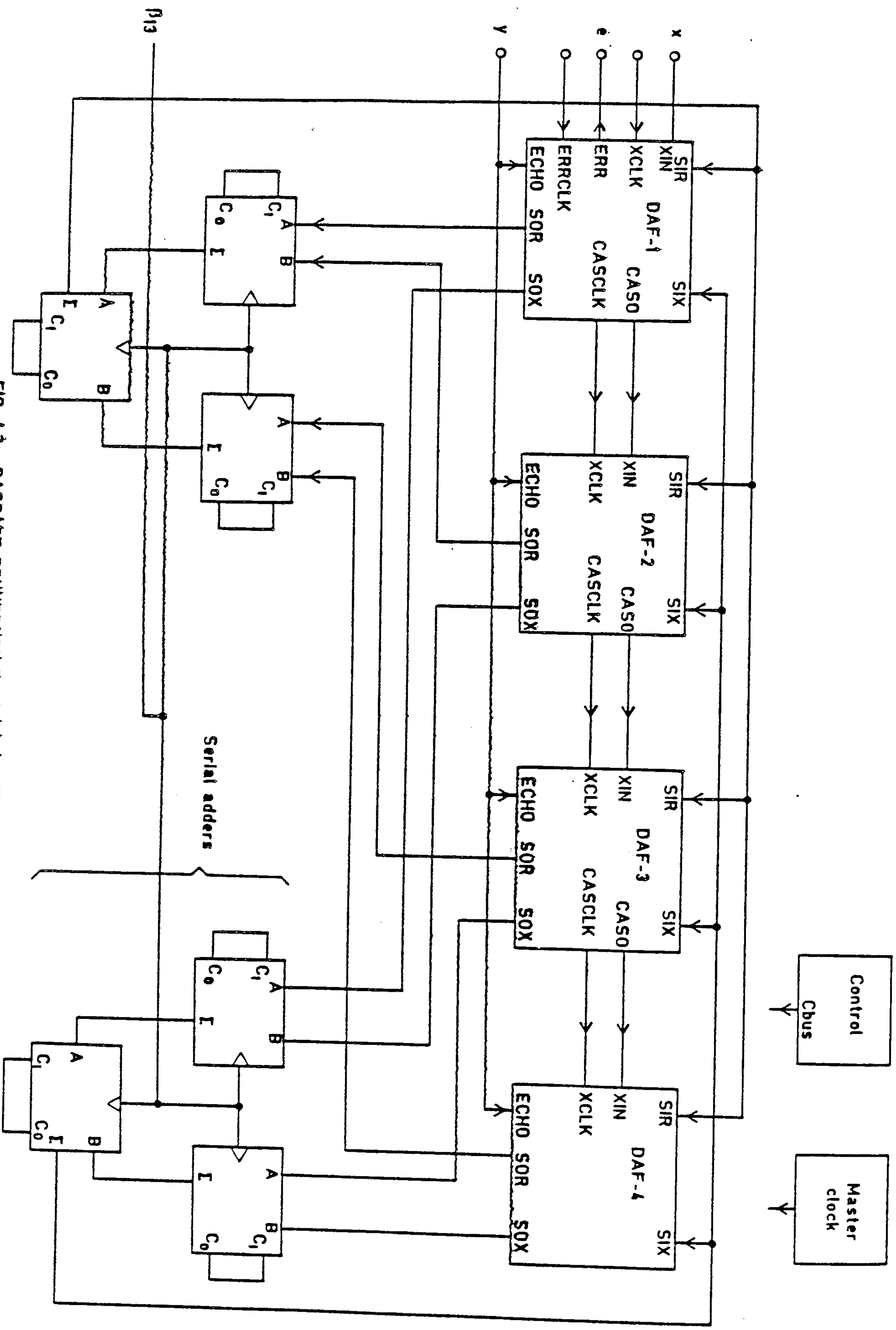


FIG. 4.2 CASCADE CONNECTION OF 4 FILTER SECTIONS

quad filter will be 4 times that in a single section. Hence, the final value of SSX can be divided by 4, i.e. selecting the upper 12 bits from the 14 bit word. The scale factor $2^c$ (the level of cascading) is then corrected on the convergence device. This approach allows for maximum utilization of the 12-bit compensation range (if the above assumption is valid). The delay introduced is one process-cycle per level of cascade; a half process-cycle for serial addition, the other half to divide by two. Overflow correction in serial mode is not possible in this implementation. Under abnormal circumstances it is possible for the filter to diverge. The condition requires the selection of the alternative compensation range ($C = 2^{28}$), pin XG = '1', for noise-like signals, and to input full-scale sine waves.

A final point concerns the application of a restart signal (see NCLR in figure 4.1), resulting in the generation of SYNCLR waveform from the master clock. If a 'soft-restart' command is issued by a controller the X vector will be full of valid data and the 240-audio-cycle delay in SYNCLR will correctly initiate each partial SSX register. For the case of a power-on restart, it is essential that sufficient time elapses (4-by-240 audio cycles for a quad filter) before adaptation begins, so that the SSX register can be filled.

## 4.7 H Extension.

A powerful Q-shift facility has been provided in the standard filter. This enables either high-speed adaptation ($q = 2^{-Q} = 1$) to track a changing impulse-response, or reduced learning time ($q < 1$) with an improvement in the adaptation for a noisy reference path. It will be shown in chapter 5 that approximately 3 dB improvement in adaptation can be achieved (for a noisy reference path) with each power-of-two reduction in q. There are applications where signal-to-noise ratios from the reference path are of the order of unity. In order to model the path, it is necessary to reduce q below the minimum provided in the standard filter. A feature of the filter design is the ability to reconnect the LSI devices on a larger U and V bus. In simple terms, the convolution (on the multiplier-accumulator) is performed on the upper portion of the bus (16 bits) , while the convergence device (CONG) is connected to the lower (16) bits. Storage of the additional bits for the two impulse-responses in on an additional memory device (MEM) and the addition $H + \Delta H$ is extended with external fast-adders. To use this facility, a pin INTernal/EXTernal is tied high, to inhibit the local overflow-detection, then the sign and carry of $\Delta h$ (which now forms a partial sum) are brought off of the device, forming part of an extended addition.The technique is described in detail in Appendix B.5, showing an example of a 8-bit increase in the U and V buses.

Little is achieved without some cost and extended Q-shifting is no exception. Results in Chapter 5 show the advantage is gained over a reduced input-signal range.

## 4.8 H Monitor.

Amongst the techniques required to examine the hardware emulator was a visual display of the adapt and stored impulse response. The advantage of having an oscillogram display is to enable the study of real time behavioural characteristics of both the filter and the impulse response. The only method currently available which would allow examination of the impulse response, in real time, would require a very fast d-to-a converter (at least 12 bits at the 2 MHz sample-rate) and a sampling oscilloscope. The system would, therefore, be very expensive to implement.

The technique developed was based on a pseudo-real-time representation of the impulse response. A gating signal (epsilon $\varepsilon$) was derived which masked the period (one process-cycle) when both $h_s^n$ and $h_a^n$ $(0<n<240)$ appear on the U bus. The 12 MSBs from each data word were stored in latches and applied to separate d-to-a converters. In the next audio-cycle epsilon slipped one process-cycle to mask $h_s^{n+1}$ and $h_a^{n+1}$. Thus, a complete impulse response is captured every 240 audio-cycles. A detailed description is given in Appendix B.6.

## 4.9 Conclusions:

The object of producing an adaptive filter in LSI was to minimise the cost, making its use in a commercially viable loudspeaking telephone an attractive solution for the provision of duplex voice-transmission. To that end the work was reasonably successful. The main costs associated with LSI devices are silicon area (also, to some extent, gate density) and the number of pins on the package. The act of partitioning the circuit into small areas imposes an increase in the pinout of each device. In this case 32, pins provide internal (to the filter) data transfer. For two out of the three devices the compromise on area and pincount resulted in low cost products. These were the multiplier-accumulator and the memory, whose final area was within 15% of the initial estimation, provided by the LSI design team. The final area of the convergence device exceeded its initial estimate by 200%. Adding to this, the large pincount factor (64 pins) resulted in an expensive device (relative to the other two). The main reasons for the large silicon area were the physical size of the correlation multiplier, the area allocated to buses and power rails and the total number of pins, requiring a minimum separation between them. Given the benefit of hindsight, the circuit would have used the shift-left algorithm (Section 2.3.3), as originally intended. This would have replaced the multiplier with a much smaller barrel-shifter, introducing a small performance penality (discussed in Section 5.8). The problem of busing presents a more fundamental problem, particularly on the convergence device where areas of random logic need be interconnected. A conceivable solution would be to distribute the processing more uniformly across the device, interconnecting active cells with short metal lines. Currently, there are no LSI design software-tools available for this type of layout. The final problem, of reducing the number of pins, could be tackled by time multiplexing the U and V buses. There would be disadvantages, particularly in terms of a minimum operating speed, since a multiplexed bus would have to operate at 8 MHz, at the 8 kHz sample-rate. This would leave little latitude for operating at sample rates greater than 8 kHz.

Figure 4.3 is a photograph of the convergence device. The device measures approximately 30.25 mm² and contains some 8500 FETs. The multiplier, centre right, covers 13% of the surface area. If this was reduced significantly, the problems of interconnecting blocks of logic and pin count would remain dominant.

A comparison of area utilization can be made with figures 4.4 and 4.5, showing the multiplier-triple-accumulator and memory device respectively. Their structure allows a more uniformly distributed approach, with consequent economic use of silicon area. The MAC measures 18.3 mm² and has some 14500 FETs and the memory 17.2 mm² with 45000 FETs.

Two final photographs are included, for comparison of scale. Figure 4.6 shows the TTL emulator, with one of the 4 boards used to model the convergence circuit in evidence. Figure 4.7 is a complete DAF on a single card and a thick -film hybrid, representing the major output from this work.

All the work described in this chapter is the subject of a patent application [401].

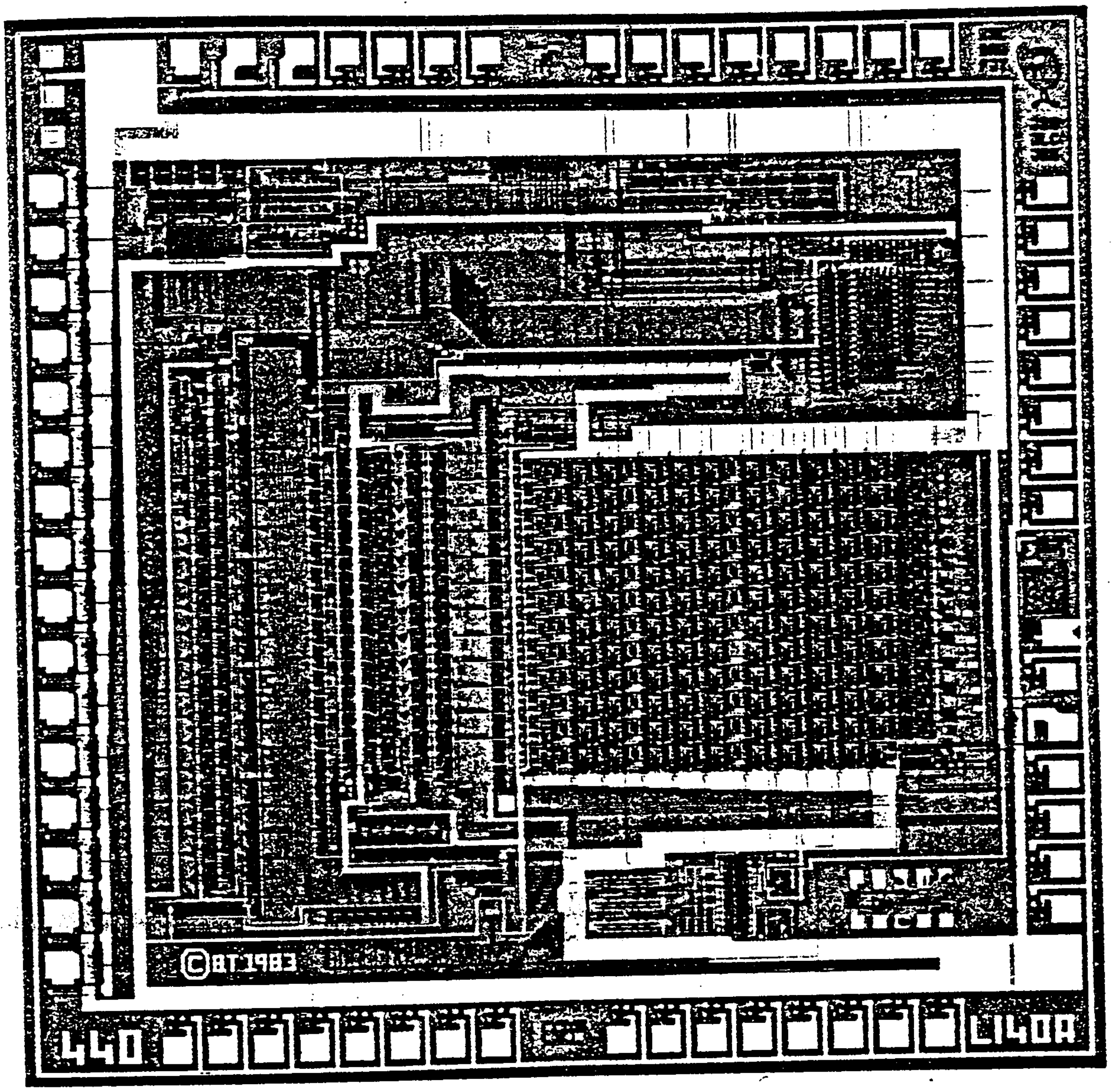


Figure 4.3 Convergence LSI

Area = 30.23 mm², 8500 FETs.

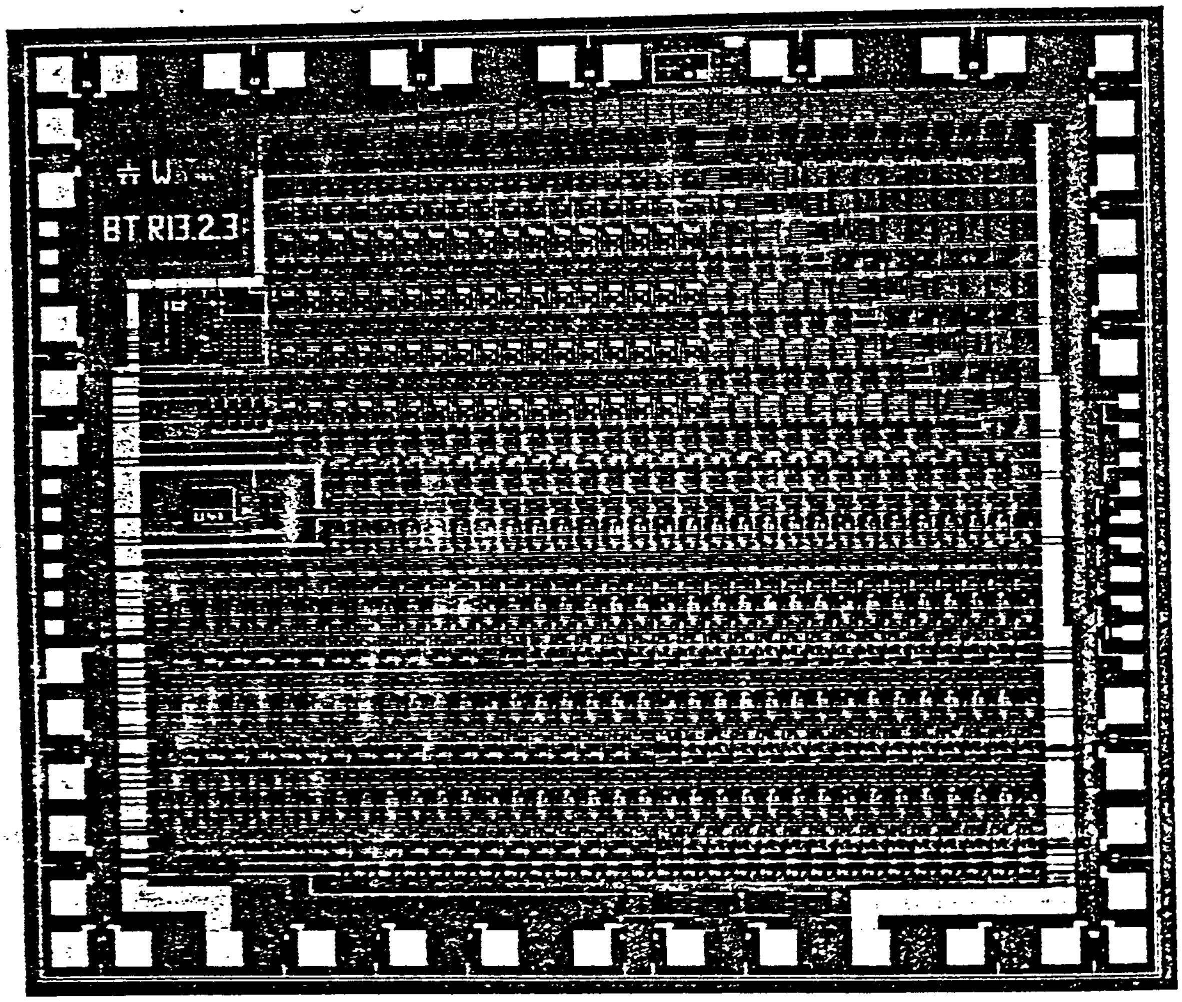


Figure 4.4 Multiplier and Triple Accumulator LSI.

Area = 18.3 mm², 13500 FETs.

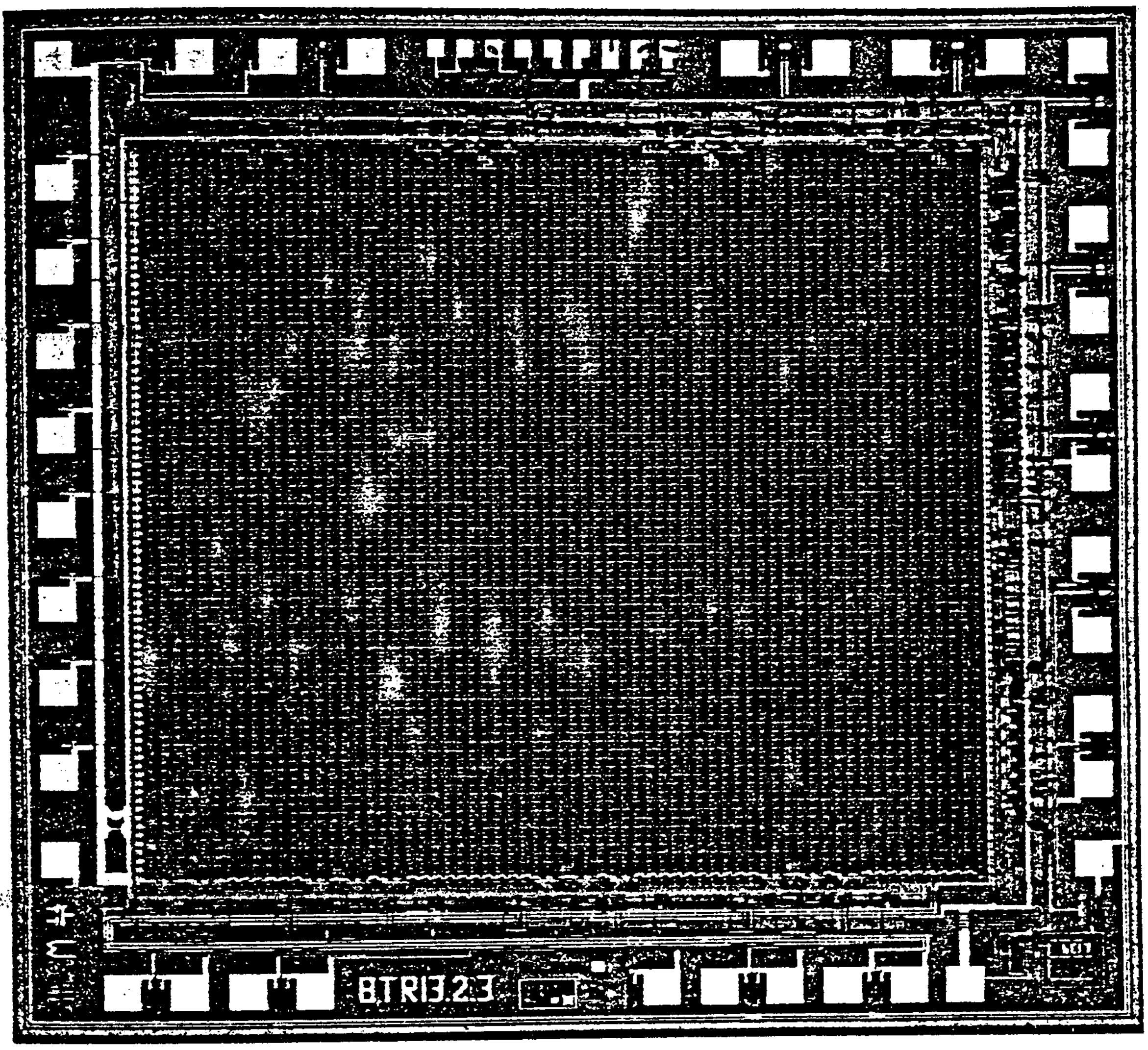


Figure 4.5 Memory in LSI.

Area = 17.2 $mm^2$, 45000 FETs.

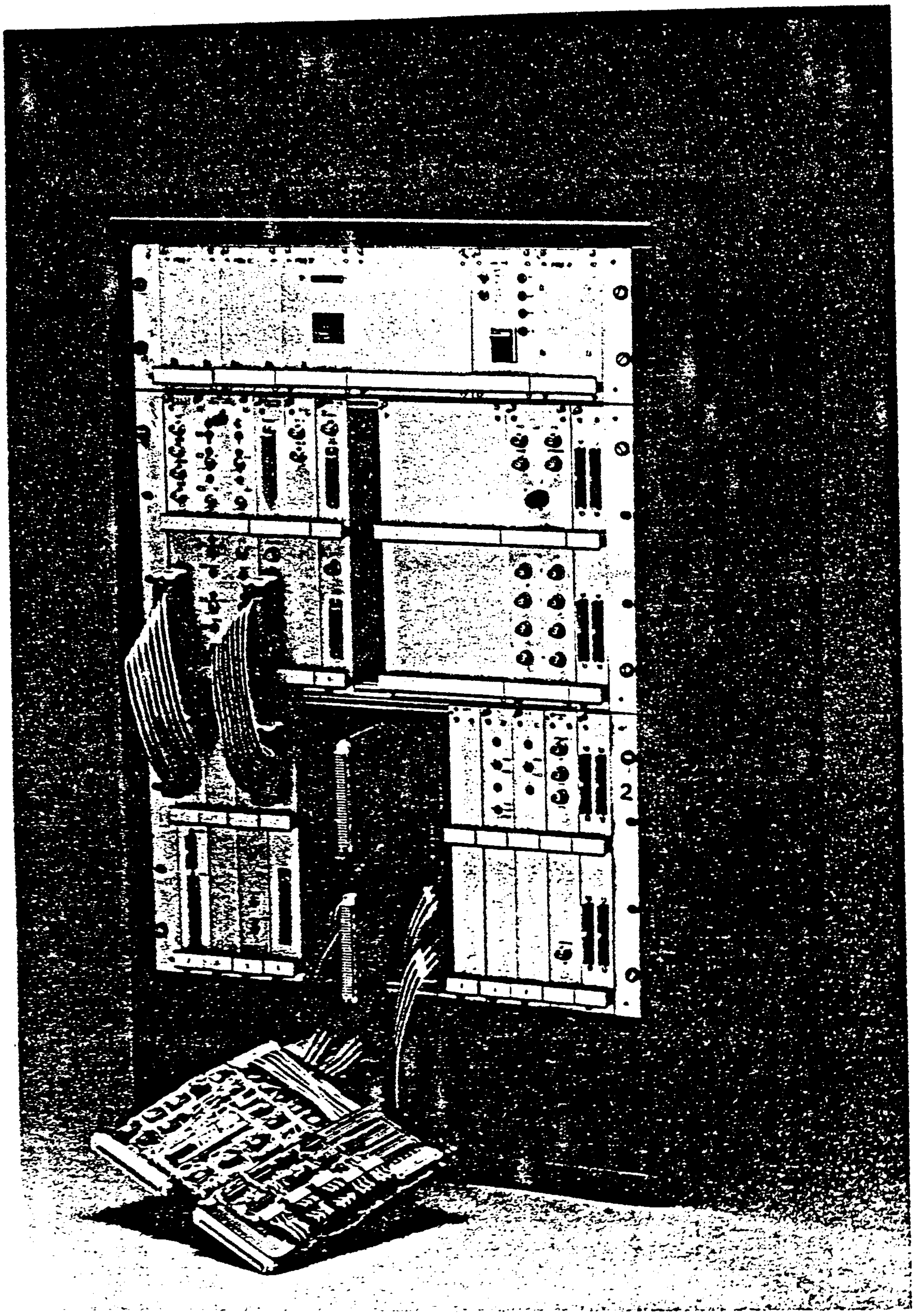


Figure 4.6 Photograph of TTL Emulator.

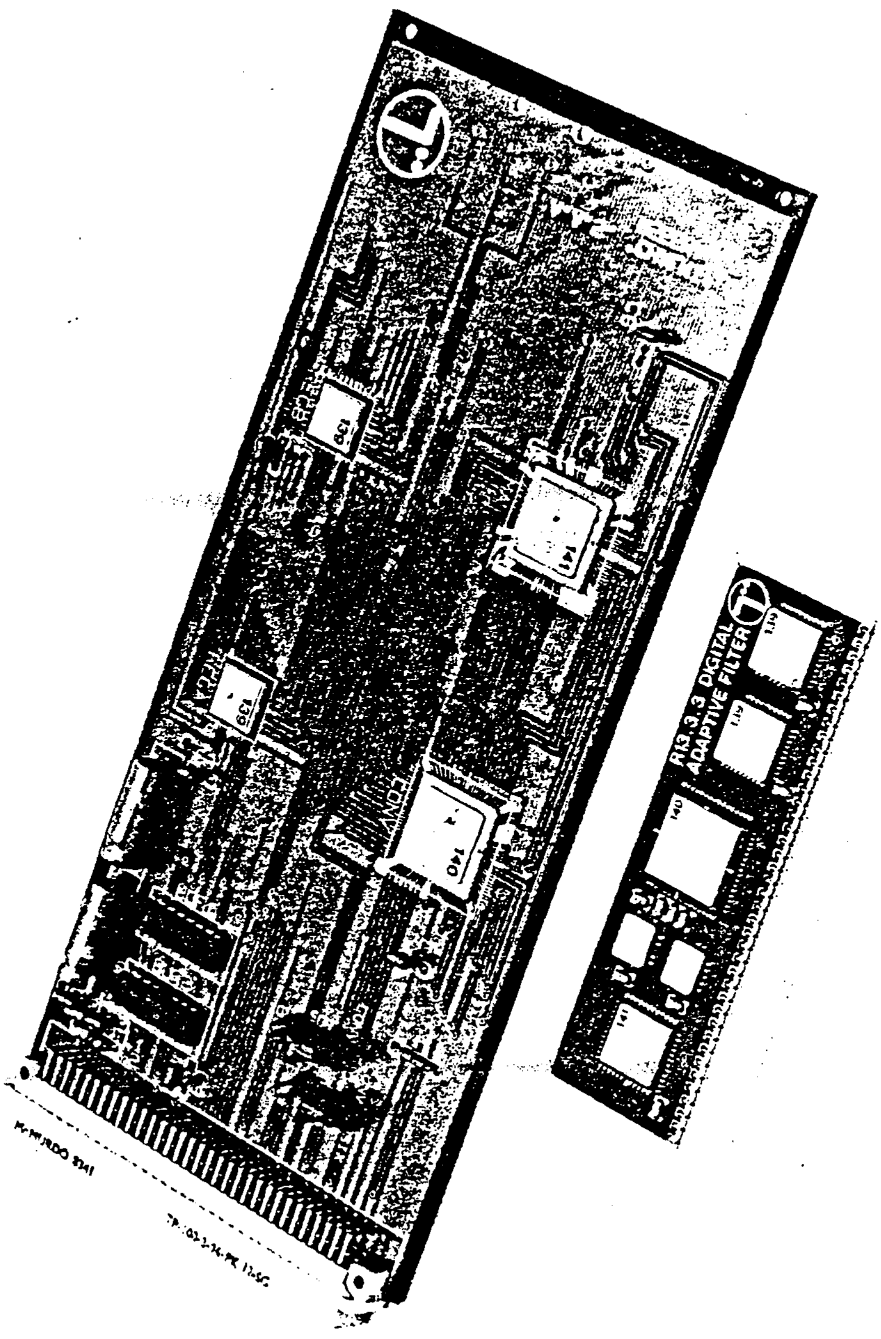


Figure 4.7 Photograph of a DAF on a PCB and a thick-film hybrid.

This page has been left intentionally blank

# 5. PERFORMANCE OF THE DAF.

## 5.1 Introduction.

The predominant emphasis in this chapter is on the performance of the filter as implemented, with occasional digressions to examine how the filter would have performed with different parameters or architecture. An example is the use of the shift-left algorithm, described in Section 2.3.3, instead of the full Widrow-Hoff algorithm.

In Chapter 2 a derivation of the achievable adaptation, for an adaptive filter with finite-precision arithmetic, was produced. All the design performance criteria were based on 3 equations, solving for stability, misadjustment noise and total noise in the error signal. Assuming that the impulse response of the path being modelled is stationary, then the stability criterion is given, to a good approximation, by equation (2.103). The misadjustment error, calculated after a period of time when adaptation has finished, is given by equation (2.105) and the total power in the error, summing all contributing noise sources, is given by equation (2.106). Equations (3.1) to (3.6) describe each of the independent noise sources in the total error (obtained from equation (2.82)) and their expected values, assuming a Gaussian input signal. Two noise components are correlated with the input signal: these are the misadjustment noise and the noise due to a finite wordsize for H in the convolution ($\sigma_d^2$) and $\sigma_f^2$ respectively). Equation (2.82) shows that the misadjustment noise is equivalent to the input signal passing through a filter which has a frequency response given by the Fourier transform of the misadjustment vector D. After infinite time the misadjustment vector will be stable and provide a filtered version of x in the error. The second x-dependent component is equivalent to x passing through a slowly-varying filter, derived from the Fourier transform of an impulse response comprised of random numbers from a uniform distribution. The theory developed in Chapter 2 gives no information on the spectral distribution of the difference vector, an extremely important factor when the filter is used to control the stability of a closed loop. In this chapter, measurements of the spectral distribution, obtained from the computer simulation, will be given and compared with results on the hardware.

A number of physical constraints applied to the LSI hardware were discussed in Chapter 3. Having partitioned the circuit in a logical (but arbitrary) manner, probably the most important limitation was the pin count, followed by the physical area of each LSI device. The former, because the packaging of LSI circuits can cost an amount equivalent to the device itself. To obtain information on the benefit of fixing the precision of arithmetic within the architecture, it is necessary to examine how the performance is affected by varying the wordsize of data paths. From this, a compromise on the two major constraints, detailed above, can achieved.

## 5.2 Constant noise contributions.

To obtain sufficient dynamic range over the telephony band, the minimum input wordsize is 12 bits, corresponding to an 8-bit pcm system, using A-law coding, which is currently an internationally agreed standard [501]. Consequently, the minimum a-to-d quantisation noise is fixed for this particular hardware implementation. Therefore, the two noise components in equation (2.106), produced by a-to-d and d-to-a conversion, introduce a fixed level into the error signal, given by equation (3.2). Using good-quality a-to-d and d-to-a converters, the accuracy of conversion is $\pm 0.5$ bits. Then, in equation (3.2), the value of $\delta_a = 1$. Introducing low-cost devices reduces the accuracy and hence increases the noise power from these sources.

The quantisation noise produced by truncating the replica (r) was arbitrarily fixed by the length of serial word which could be transmitted in the pause. This was chosen to be 12 bits, giving a resolution of $\pm 0.5$ bits for the number range used in the filter (derived from the normalisation of the replica $g = 2^{11}$). Consequently, to obtain the noise power from equation 3.6, the value of $\delta_r = 1$. When filters are cascaded together, each section produces a partial replica; these are then summed externally. Thus, the total noise power, due to replica quantisation, of a cascaded set is the sum of these noise powers, since they are produced independently. Equation (3.6) becomes:

$$\sigma_r^2 = \frac{2^c}{12}.\delta_r^2 \qquad \ldots (5.1)$$

where c is the level of cascading.

The remaining invariant noise component in the error (excluding noise generated in the reference path) is $\sigma_u^2$, given by equation (3.5). It is dependent on the resolution of the convolution, which, in the final hardware system was fixed at 16 bits, with 4 bits below the binary point. Therefore, in equation (3.5), the value of $\delta_t = 2^{-4}$. It is also dependent on the number of coefficients, W, in the impulse response. A single filter-section has 240 coefficients, and the total for a cascade system is given by $W = 2^c.240$. Table 5.1 lists the power of the constant noise components and shows that the contribution from $\sigma_u^2$ is insignificant.

| noise component | power |
|---|---|
| $\sigma_a^2$ | 0·083 |
| $\sigma_r^2$ | $2^c$.0·083 |
| $\sigma_u^2$ | $2^c$.1·552 $10^{-9}$ |

Table 5.1 Power in constant noise components.

## 5.3 Input-level dependent noise components.

The error power contains two input-level (x) dependent noise contributions, $\sigma_d^2$ and $\sigma_f^2$, given by equations (2.105) and (3.4) respectively. It was shown in section 3.4.3 that the update vector (ΔH) could be reduced by a technique called Q shifting. The hardware implementation is described in Appendix B.2.4, and effectively extends the normalisation constant $t_2$ to $t_2 2^Q$. The advantage obtained can be found by substituting the above into equation (2.105), together with equations (3.8) and (3.9) to solve for $f(2\mu)$. The misadjustment noise-power is given by:

$$\sigma_d^2 = \frac{1}{[2^{Q+1} + 1]} \left[\sigma_n^2 + \sigma_b^2\left[\frac{W.\sigma_x^2}{g.k.t_2}\right]^2 + \sigma_h^2\left[\frac{2^Q.W.\sigma_x}{g.k}\right]^2\right] \qquad \ldots (5.2)$$

The effect of introducing Q shifting can be found by comparing equations (5.2) and (3.10), the latter being without Q shifting. The result is to reduce the noise contributions from both the external noise-sources (combined in $\sigma_n^2$) and the quantisation noise from truncation of the $f(2\mu e)$ bus, but to increase the quantisation-noise contribution from truncation of the $\Delta H$ bus. Figure 5.1 shows the that the change in each component becomes 3 dB per Q shift, for Q greater than 2. The benefit of introducing Q shifting is the reduction of misadjustment noise in the presence of significant levels of noise generated in the reference path ($\sigma_p^2$).

Reduction in $\sigma_f^2$ (equation (3.4)) can only be achieved by increasing the precision of the convolution multiplier, and must be weighed against the cost of increasing the area of the multiplier-accumulator device. In the final hardware implementation, the full precision available was utilized, i.e. 12-bit x and 16-bit h words.

Expressing the separate noise-contributions in the error signal, equation (2.106), as the sum of a constant term and an input-level dependent term, adaptation can be defined as:

$$\text{Adaptation} = 20 \log_{10}\left[\frac{\sigma_x}{\sigma_e}\right]$$

$$= 20 \log_{10}(\sigma_x) - 10 \log_{10}\{P(x) + U + \sigma_p^2\} \text{ dB} \qquad \ldots (5.3)$$

where

$$P(x) = \sigma_d^2 + \sigma_f^2$$

and

$$U = \sigma^2_{a\_d2} + \sigma^2_{d\_a} + \sigma^2_u + \sigma^2_v + \sigma^2_r$$

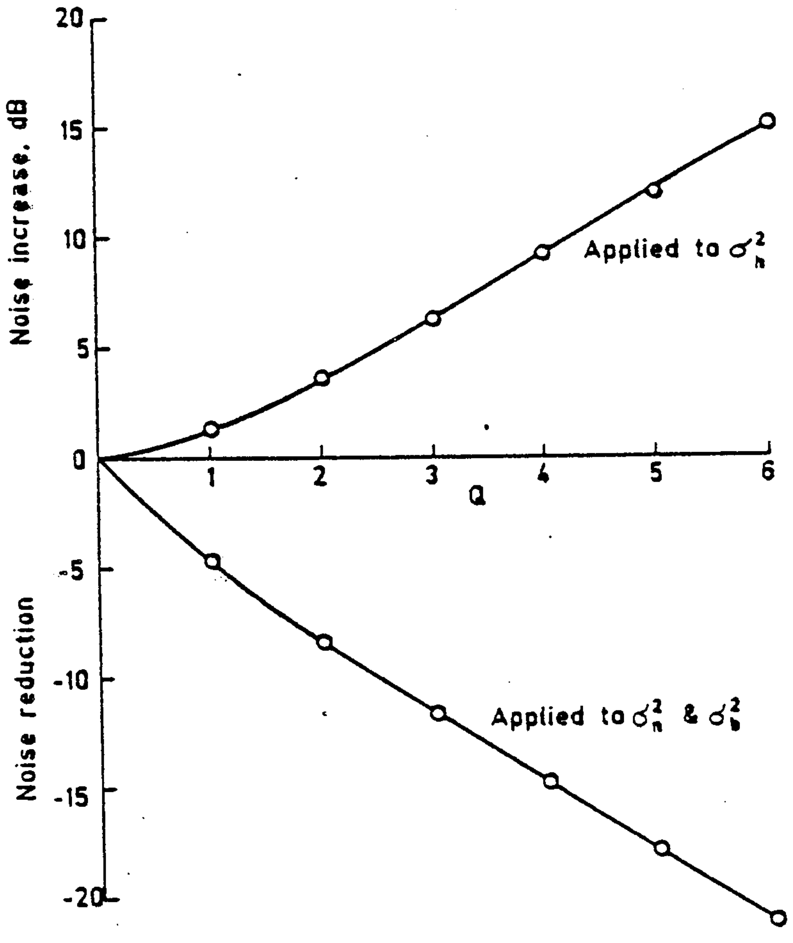


FIG. 5.1 Q SHIFT ADVANTAGE ON MISADJUSTMENT NOISE

Equation (5.3) is a convenient and general definition of adaptation, since it is independent of the reference path. More useful definitions are:

$$\text{Coherent Adaptation} = 10\ \log_{10}\left[\frac{\sigma^2_x}{P(x)}\right] \qquad \ldots (5.4)$$

which gives a measure of the component of x in the error.

Also:

$$\text{System Adaptation} = 10\ \log_{10}\left[\frac{1}{P(x)+U}\cdot\sigma^2_x\right] \qquad \ldots (5.5)$$

which gives a measure of the performance in the presence of noise in the reference path. In the loudspeaking telephone an estimate of adaptation can be made when a single talker is present, by measuring the input level to the filter and the error output. Also, an estimate can be made of the noise which is local to the reference path ($\sigma_p^2$) during a period of nominal silence. Hence, obtaining an estimate of the system adaptation. The value required to determine how stable the the loop might be is the coherent adaptation, which cannot be measured in real time. Even if it could be measured, it would be of little value, since no spectral information is available. This subject will be covered in detail later in this chapter.

Plotting adaptation (equation (5.3)) against input level for P(x) tending to zero would produce a straight line (with slope = 1) setting the maximum which can be achieved.

## 5.4 Variation of the $t_1/t_2$ ratio.

An argument was developed in section 3.7 to have a relationship between each component of the $t_1.t_2$ parameter, for normalisation, which was not an optimum in terms of misadjustment noise. Figure 5.2 shows the coherent adaptation against input level for a range of $t_1.t_2$ parameters. The first striking feature is the ripple-like nature of plot, which is caused by the nonlinear approximation for the f(2μ) (see equation (3.24)), effectively altering the q parameter over the 3 dB range of input for which f(2μ) is constant. The process is described in equation (3.37). In the computer program which calculated the coherent adaptation (reproduced in Appendix C), the value of f(2μ) is calculated at a specific input level, assuming a perfect rms value (i.e. integrated over infinite time). It was shown in section 3.4.1, that by determining the length of input vector (over a finite number W), the resulting approximate rms value would have a chi-squared distribution. Therefore, the hardware would produce a smoothed result when the input signal has a Gaussian distribution. Over the input range of -15 to 0 dBm, the coherent adaptation follows a straight line as the x-dependent contribution becomes small compared with the constant-value noise components. For input range greater than 0 dBm, the reduction of $t_2$ by a factor of 2 causes the coherent adaptation to fall, by up to 6 dB, which is also evident in equation (5.2). Below -16 dBm input the coherent adaptation begins to rise and then levels out. This is due to the effective reduction in q, caused by the input level falling below the compensated region (see Table 3.2).

The values chosen were $t_1 = 2^6$ and $t_2 = 2^{12}$ to achieve maximum speed of convergence over the compensated input range and will be used throughout the remainder of this chapter.

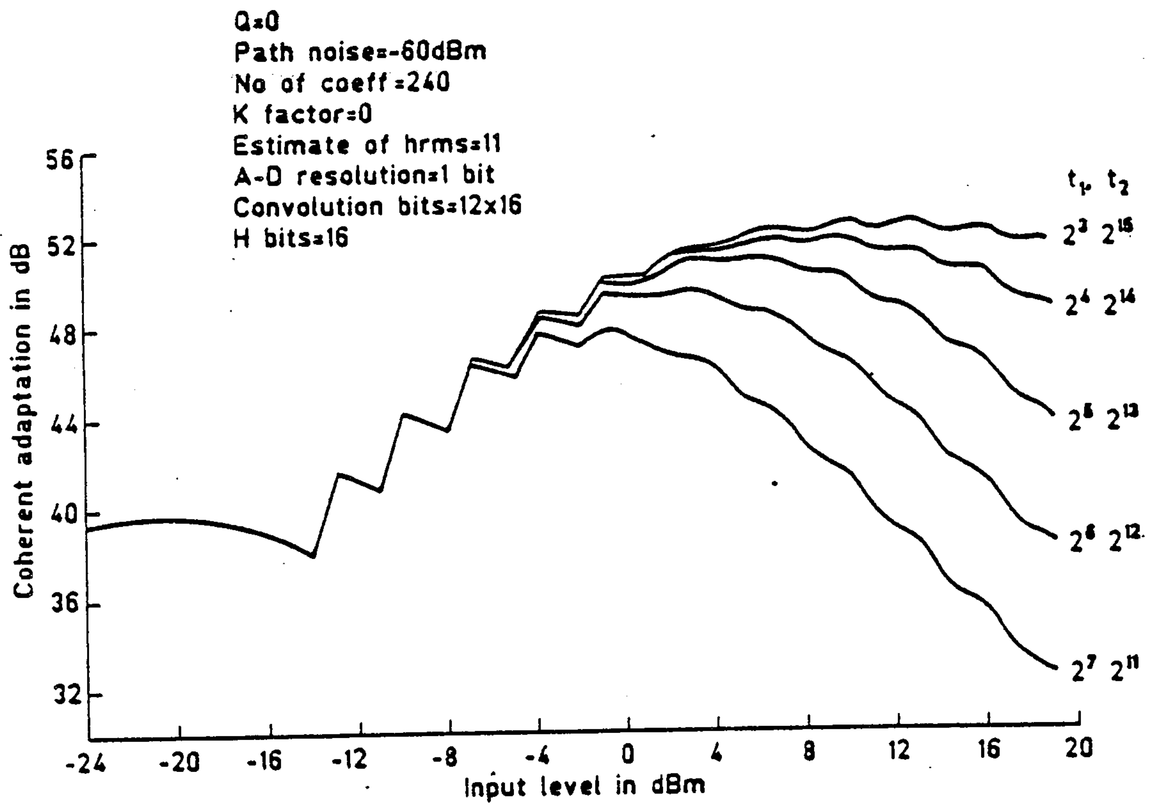


FIG. 5.2 COHERENT ADAPTATION AGAINST INPUT LEVEL FOR A RANGE OF NORMALISATION VALUES

## 5.5 Performance of a single filter-section.

The performance of the adaptive filter was measured while modelling an analogue reference path, as shown in figure 2.5. The physical reference path was a graphic equaliser having an arbitrary impulse-response, bounded within a filter section, for which the rms value of h was measured ($h_{rms} = 11$). The value of $h_{rms}$ is required to solve equation 3.3. The noise generated by the graphic equaliser was measured to be -60 dBm.

Figure 5.3 shows the theoretical and measured values of system adaptation against input level, having very good agreement. To achieve these results in the hardware measurement, it was necessary to use very-low-noise filters prior to sampling and for the reconstruction filters. The cause of the small error, confined to the straight-line region, can be found in the assumption given by equation (2.94), i.e. that the mean level of each noise component is zero. This is very difficult to achieve both in the filter and in the a-to-d conversion.

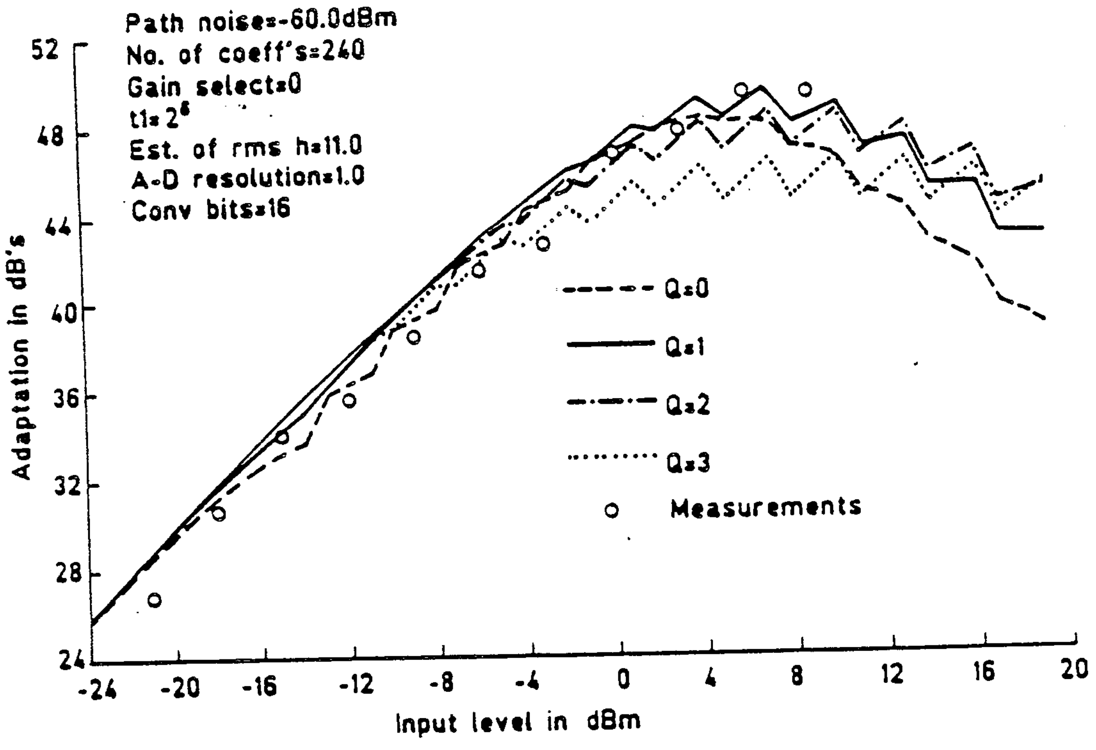


FIG 5.3 ADAPTATION OF FILTER (Measurement setup)

Analogue measurements were restricted to a maximum input of 10 dBm, to reduce clipping to a minimum, when using a Gaussian input-signal. In an attempt to obtain results equivalent to that of an ergodic system, when using a single hardware filter, the final value plotted is the mean of 10 measurements, each started from $H^* = 0$ and averaging 10 measurements after adaptation ceased ($\Delta H = 0$). The resultant standard deviation was better than 0.3 dB.

In the region of +4 dBm input, the adaptation for Q = 3 is approximately 3 dB below that for Q = 0. The reduction is expected for a low noise-level in the reference path, as described above.

## 5.5.1 Rate of adaptation.

An important parameter is the rate at which the filter learns an impulse response, particularly for the acoustic path of the LST which will have a rapidly-changing response. A learning curve for a single run is shown in figure 5.4, detailing two quantities, the adaptation and the rms of the error.

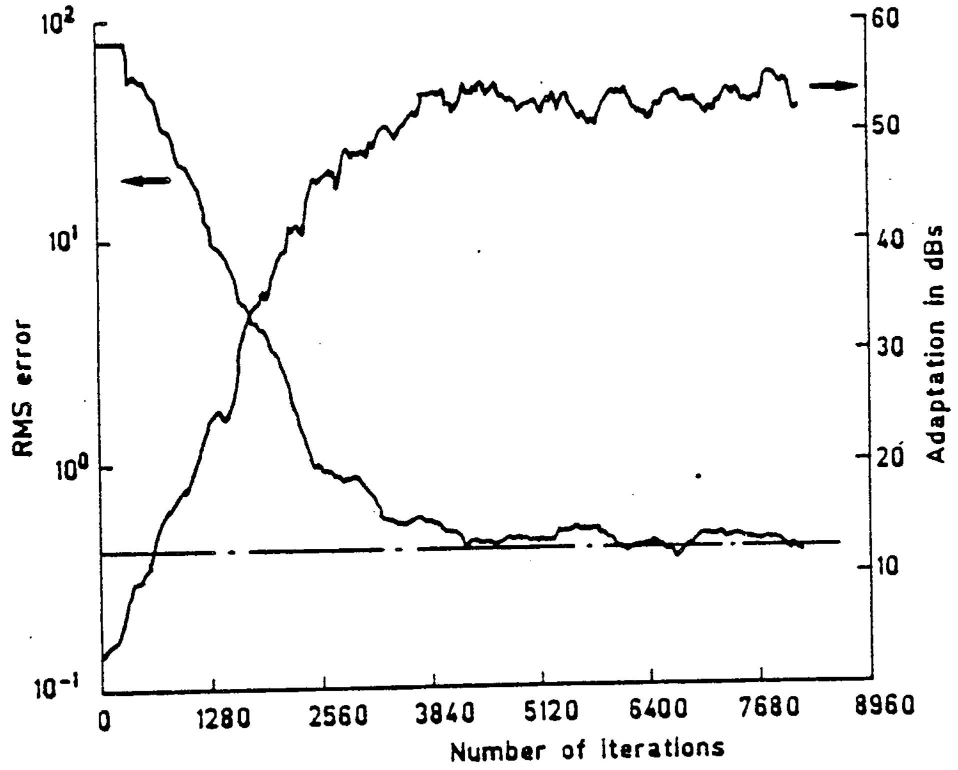


FIG. 5.4 LEARNING CURVES (ERROR & ADAPTATION) FOR THE DAF

The curves were derived from the computer simulator and show that the adaptation can vary over short periods by up to 3 dB. A more informative measurement would be an ergodic average, obtained from an ensemble of filters.

Because of the considerable time required to make an ergodic approximation for each time increment, a computer controlled measurement system, shown in schematic in form in figure 5.5, was built to test the hardware (the simulator was too expensive to run for general-purpose measurements). The measurement circuit allows known levels of Gaussian noise to be added to the reference path. In figure 5.6 two learning curves are shown for $Q = 0$ and 3, measured at an input level within the compensated range. The curve for $Q = 0$ has an initial slope of 165 dB/s, which falls to 27 dB/s at $Q = 3$.

While it is difficult to predict the exact rate at which the filter makes an estimate of the impulse response of the reference path, it is possible to obtain bounds (albeit quite large) within which the rate of adaptation should lie. Widrow [222] gives an estimate of the reduction of error power per sample time of:

$$\text{Error power reduction} = 4\mu\sigma_x^2 \text{ per sample} \qquad \dots (5.6)$$

The estimate of $2\mu$ in the hardware is nonlinear, therefore, a factor is required to account for the discrepancy. It was shown in Section 3.5 that the effect of introducing $f(2\mu)$ was equivalent to a q-factor
change between 1.0 and 0.5 over the 3 dB range of input level. Let $q(x)$ be the input-level dependent factor, then the hardware $f(2\mu)$ will lie in a range given by:

$$f(2\mu) = \frac{q(x)}{W.\sigma_x^2} \quad \text{where } 1 > q(x) > 0.5 \qquad \dots (5.7)$$

Substituting equation (5.7) into (5.6) and scaling $f(2\mu)$ by $2^Q$, then the rate of error-power reduction (adaptation rate) is given by:

$$\text{Error power reduction} = f_t.10\log_{10}(1 - \frac{q(x).2^{Q+1}}{W}) \text{ dB/s} \qquad \dots (5.8)$$

where $f_t$ is the sample rate.

Table 5.2 gives the upper and lower bounds for the rate of convergence from equation (5.8) and the measured values.

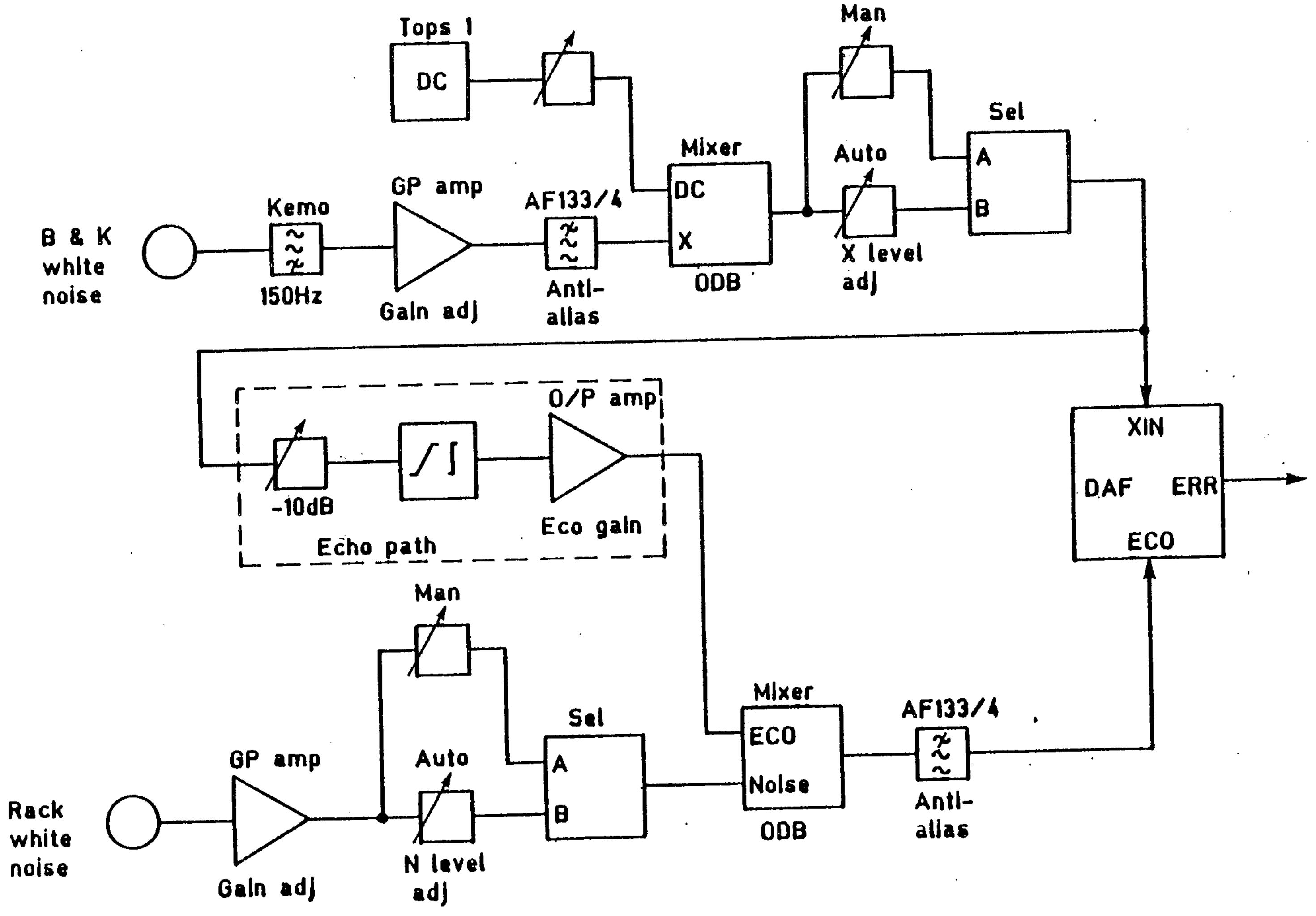


FIG. 5.5 DAF: STANDARD MEASUREMENT SYSTEM

| Q | Adaptation rate dB/s | | |
|---|---|---|---|
| | q(x) = 1 | q(x) = 0.5 | measured |
| 0 | 297·7 | 145·5 | 165 |
| 1 | 145·5 | 74·2 | 91 |
| 2 | 74·2 | 37 | 52 |
| 3 | 37 | 18·5 | 27 |

Table 5.2 Predicted and measured rates of adaptation for sample rate of 8 kHz.

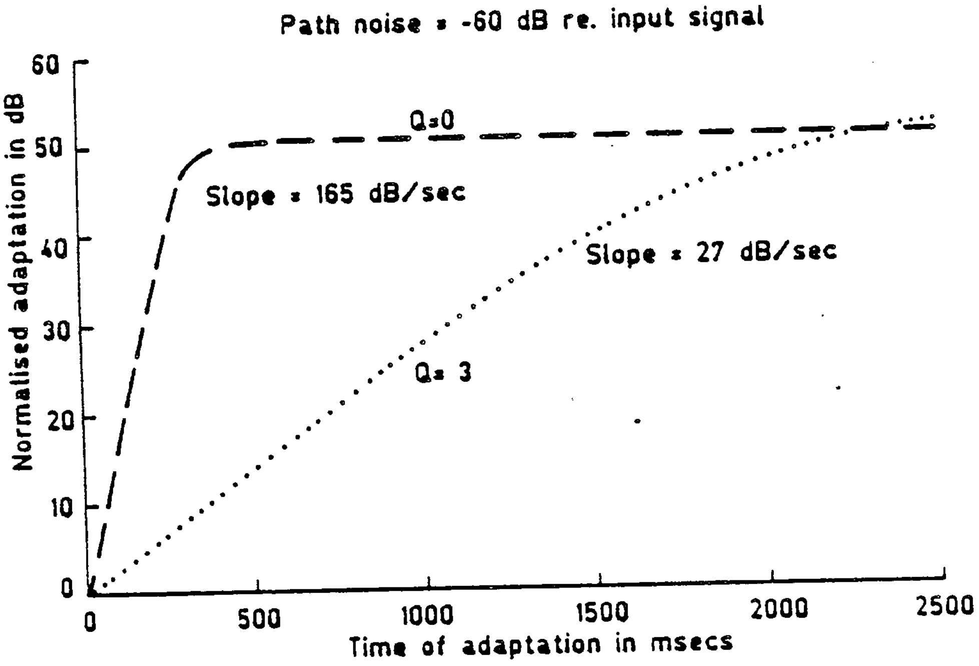


FIG. 5.6 DAF LEARNING CURVE (AVERAGE)

### 5.5.2 Spectral distribution of the misadjustment vector D.

To control an unstable loop with the DAF, it is essential that an estimate of the worst-case misadjustment noise in the frequency domain be made from an estimate in the time domain. To achieve this objective a number of measurements were carried out on the computer simulator with an arbitrary impulse response for the reference path. One such impulse response is shown in figure 5.7, which comprises 50 values from a Bessel function ($J_1(x)$), suitably scaled. The corresponding frequency-response is shown in figure 5.8.

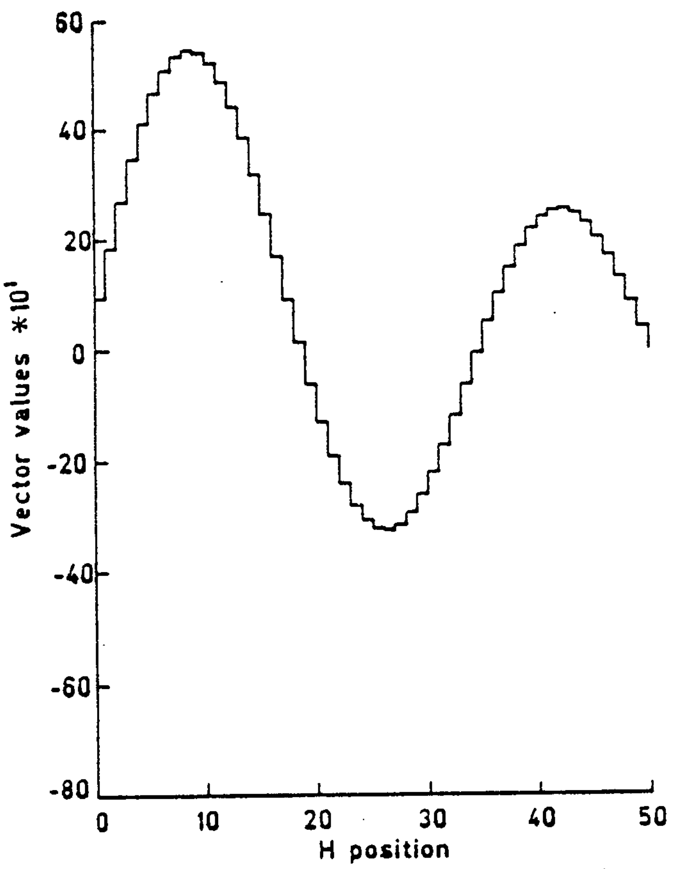


FIG 5.7 INPUT IMPULSE RESPONSE

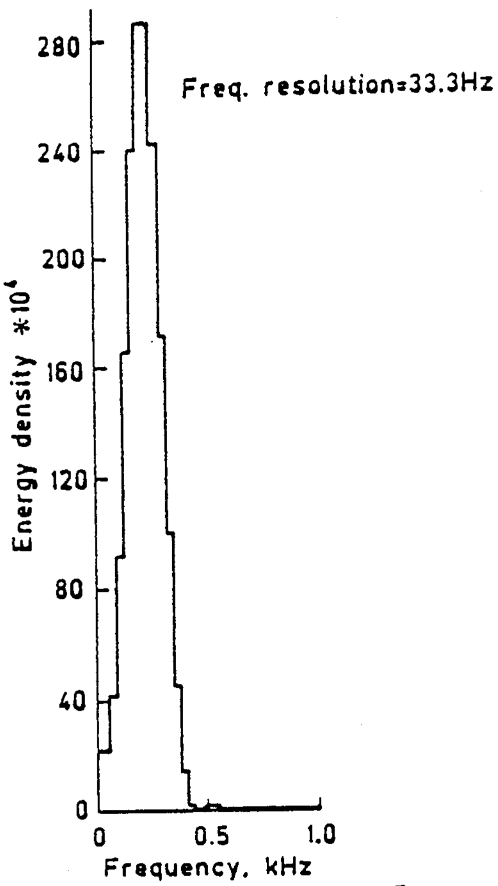


FIG 5.8 SPECTRUM OF INPUT IMPULSE-RESPONSE

Once the filter has converged to an estimate of the impulse response, the various noise components cause the misadjustment vector D (equation (2.81)) to vary in a random manner about a mean length. NB the mean length of the misadjustment vector gives another measure of the degree to which the filter has converged to the response of the reference path. In figure 5.9 the rms value of a single coefficient is plotted as a function of time (corresponding to the learning curve of figure 5.4) and shows a variation of ± 0.2 about a mean value, after the error has reached its minimum. Figure 5.10 shows a misadjustment vector at single sample-instant. Individual values in the misadjustment vector have the appearance of being randomly distributed about zero. For this particular example the mean is 0·003 and the standard deviation is 0·57. Determination of the probability that peaks will occur with some degree of regularity will give a measure of a probable frequency response,

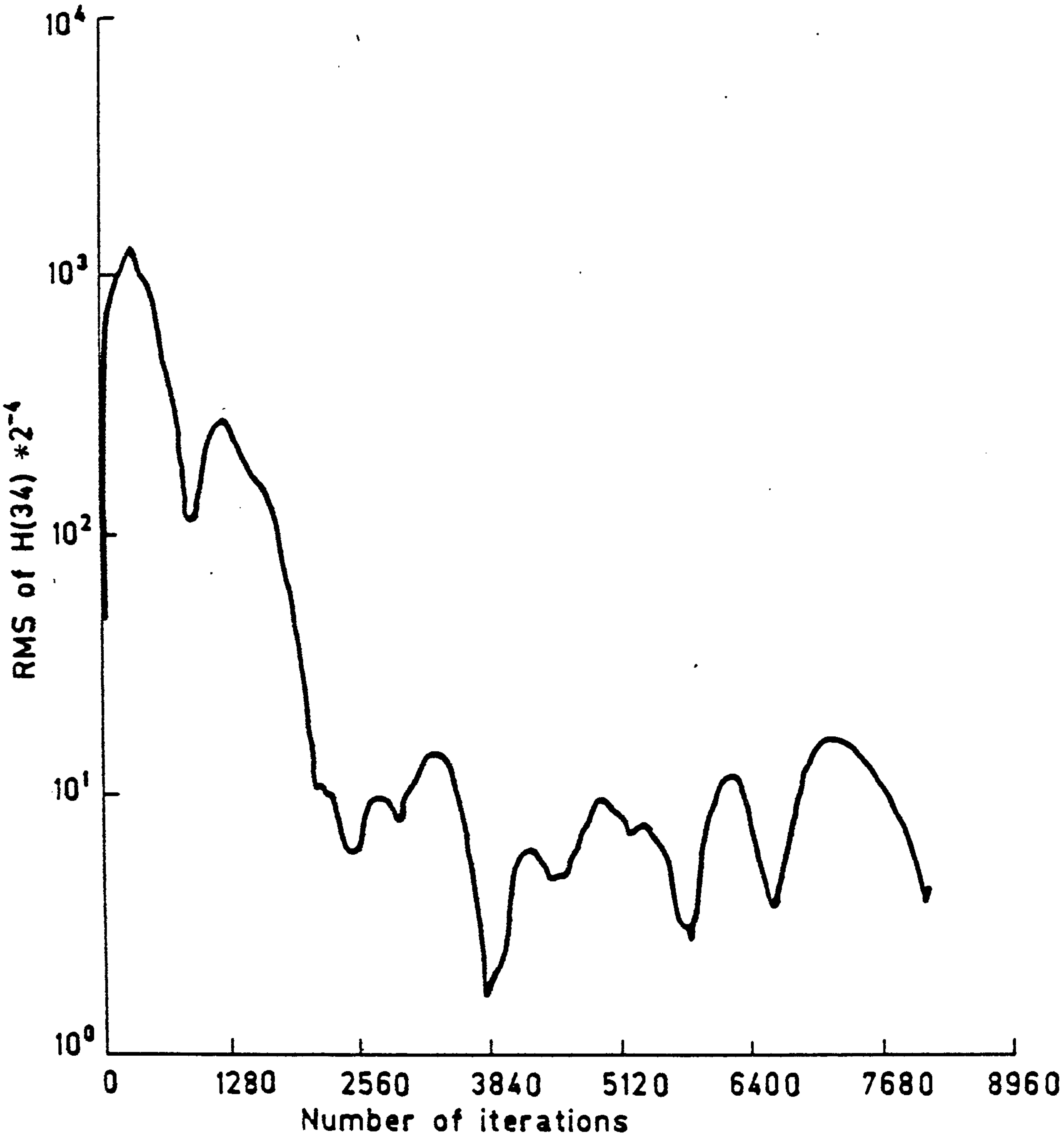


FIG. 5.9 RMS OF H(34) VALUE

which is most easily achieved by measurement.

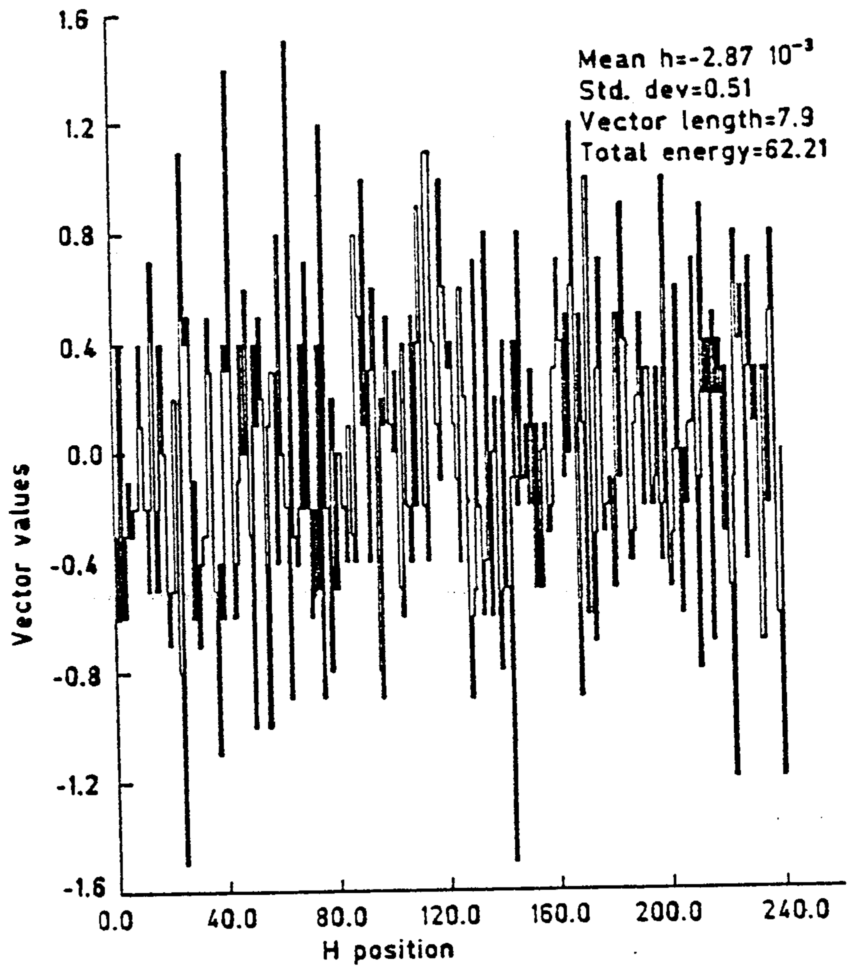


FIG. 5.10 MISADJUSTMENT VECTOR AFTER 8191 ITERATIONS

Figure 5.11 shows the frequency response corresponding to figure 5.10, and has a peak to rms ratio of 10 dB. This means that at a particular time a hole has appeared in the frequency response of the misadjustment noise, which is 10 dB below the rms. The value was the maximum found from a series of measurements. By measuring a number of misadjustment frequency responses it is possible to determine the distribution of peaks. This work is reported in Chapter 6 for Gaussian noise and will be carried out at some future date with speech.

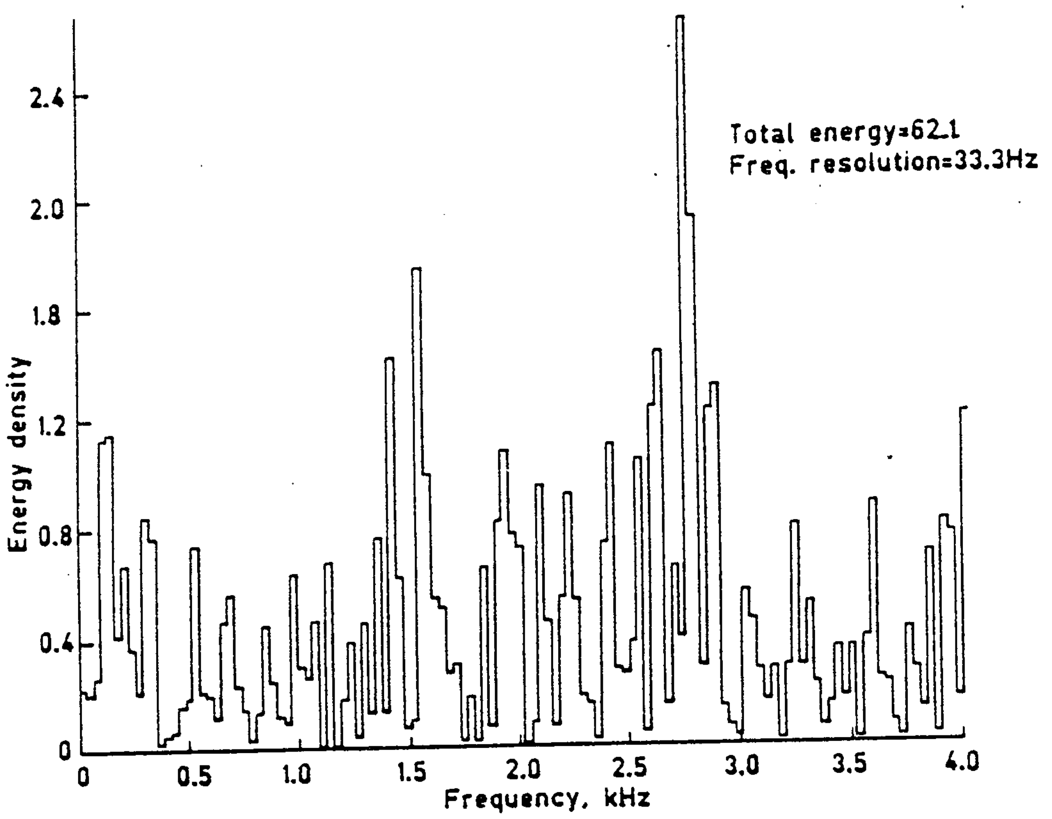


FIG. 5.11 SPECTRUM OF MISADJUSTMENT VECTOR AFTER 8191 ITERATIONS

For practical use in the loudspeaking telephone, another measure can be defined:

loop adaptation = coherent adaptation - stability margin. ... (5.9)

where, for a Gaussian input signal, the stability margin is 10 dB.

Since it is impracticable to measure the coherent adaptation in the loudspeaking telephone, a worst-case estimate is required. Figure 5.12 shows the difference between coherent and system adaptation for a range of noise powers generated in the reference path. As the input level approaches 10 dBm (the clipping level), the coherent and system adaptations become equal. Therefore, the worst case loop adaptation becomes:

$$\text{loop adaptation}\Big|_{\substack{\text{worst}\\ \text{case}}} = \text{system adaptation} - \text{stability margin} \qquad \ldots (5.10)$$

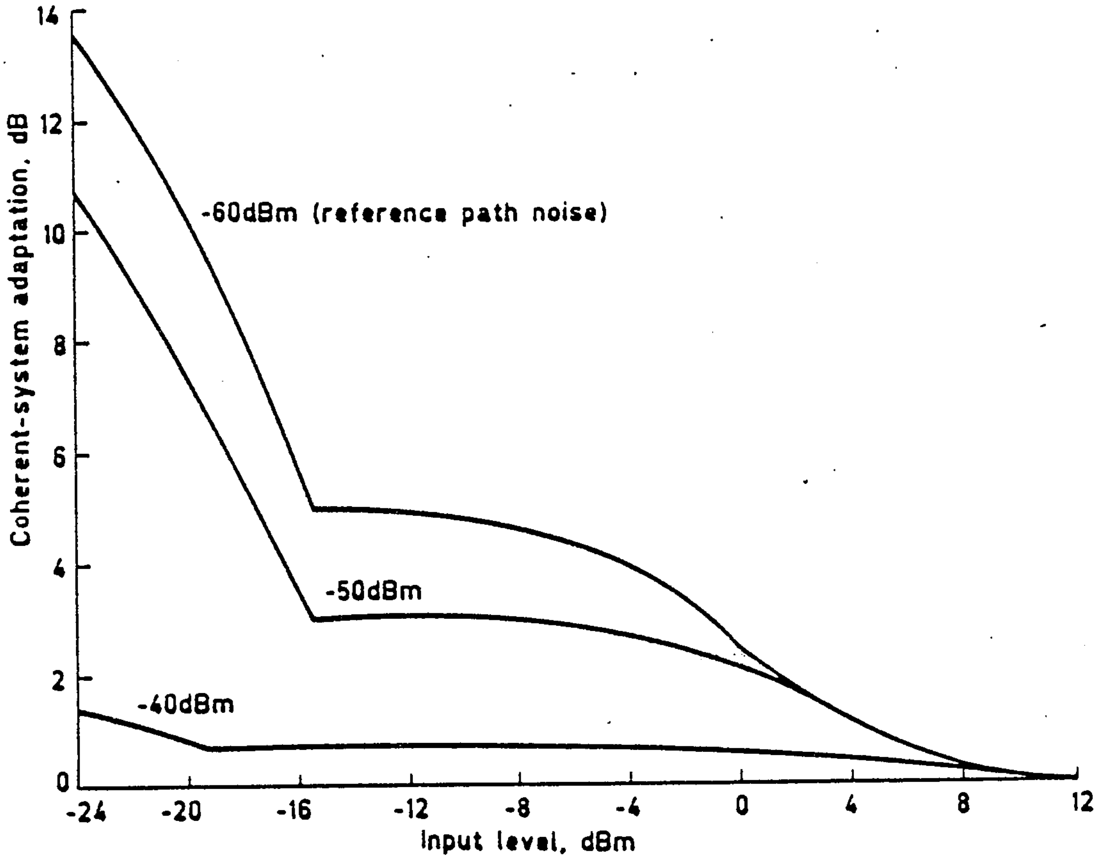


FIG. 5.12 DIFFERENCE BETWEEN COHERENT & SYSTEM ADAPTATION AGAINST INPUT LEVEL FOR VARIOUS VALUES OF REFERENCE PATH NOISE

A simple experiment was set up, shown in figure 5.13, to test the validity of equation (5.10). The DAF was allowed to converge, with the attenuator set to its maximum, and then frozen, giving the system adaptation. Attenuation was then removed and the level of error recorded at a point just before the circuit oscillated, giving the loop adaptation. The worst case stability margin found was 8 dB.

The test was not exhaustive and is really only of academic interest because the stability margin must be determined for the signal used to converge the filter, i.e. speech. For this case, the system adaptation is lower and it is anticipated that the stability margin will be greater. This is the subject of current research.

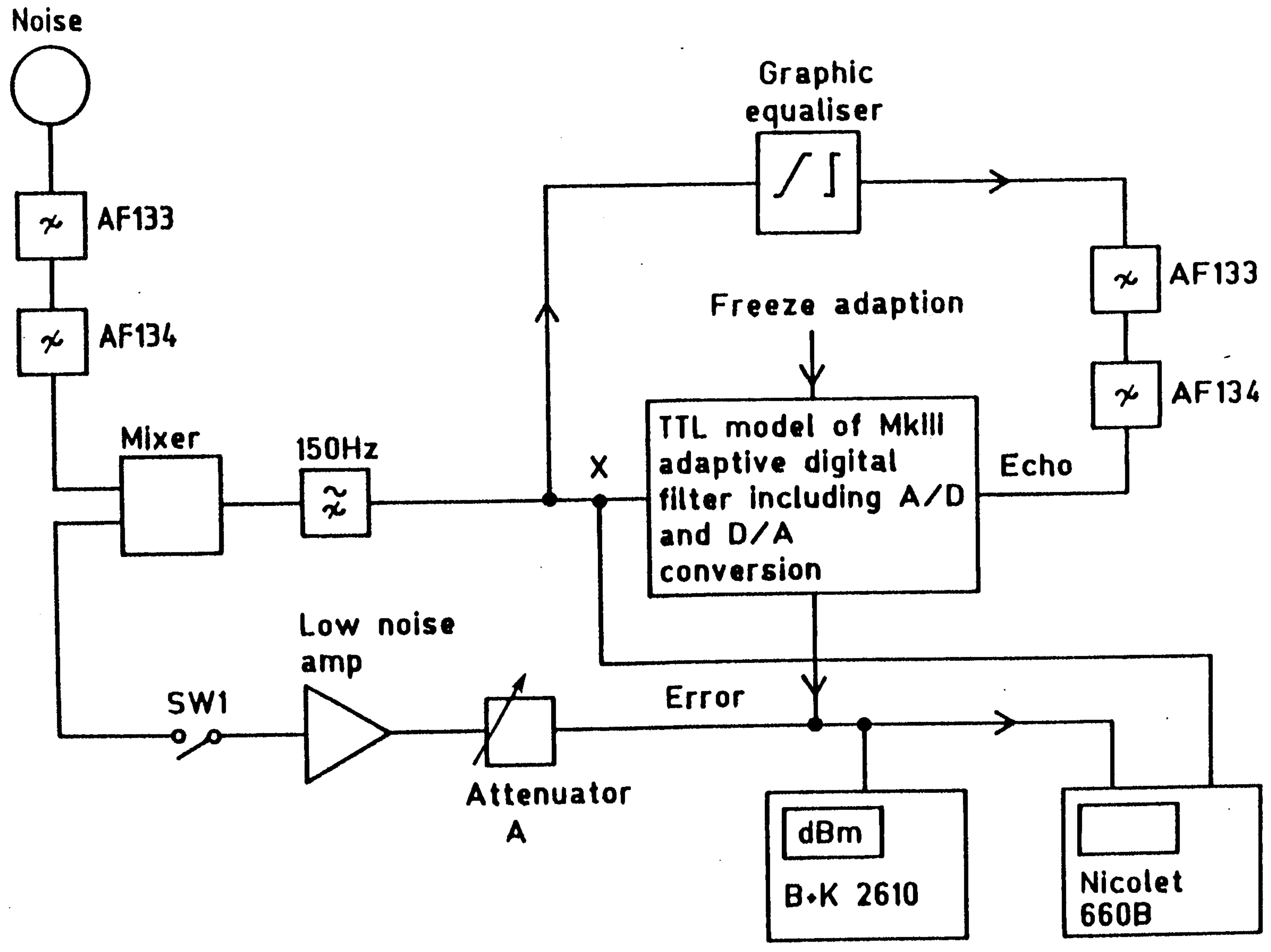


FIG. 5.13 LOOP ADAPTATION MEASUREMENT

## 6.5.3. Wordsize for convolution multiplier.

The other noise contribution which is correlated with the input signal (x) is $\sigma_f^2$ (see equations (2.85) and (3.4)). It is produced by truncation of the bus-sizes used at the input to the convolution multiplier. Figure 5.14 shows the performance improvement of using the full storage precision of 12-by-16 bit over that of a 12-by-12 bit to the multiplier. The maximum advantage is approximately 2 dB, but the contribution due to $\sigma_f^2$ has been reduced by 24 dB (6 dB per bit), effectively removing it from the equation for noise. As in the case of the misadjustment noise, this gives no indication of spectral distribution. The vector $N_t$ is derived from a uniform distribution, since it is generated by the process of quantisation. It will, therefore, have spectral properties similar to the misadjustment vector, i.e. a large peak-to-rms ratio, and would produce significant 'holes' in the frequency domain. The cost of implementation was an increase in area of the multiplier-accumulator LSI circuit of approximately one third, plus 4 additional pins. This was a small price to pay for the removal a coherent term in the primary application of loop-stability control.

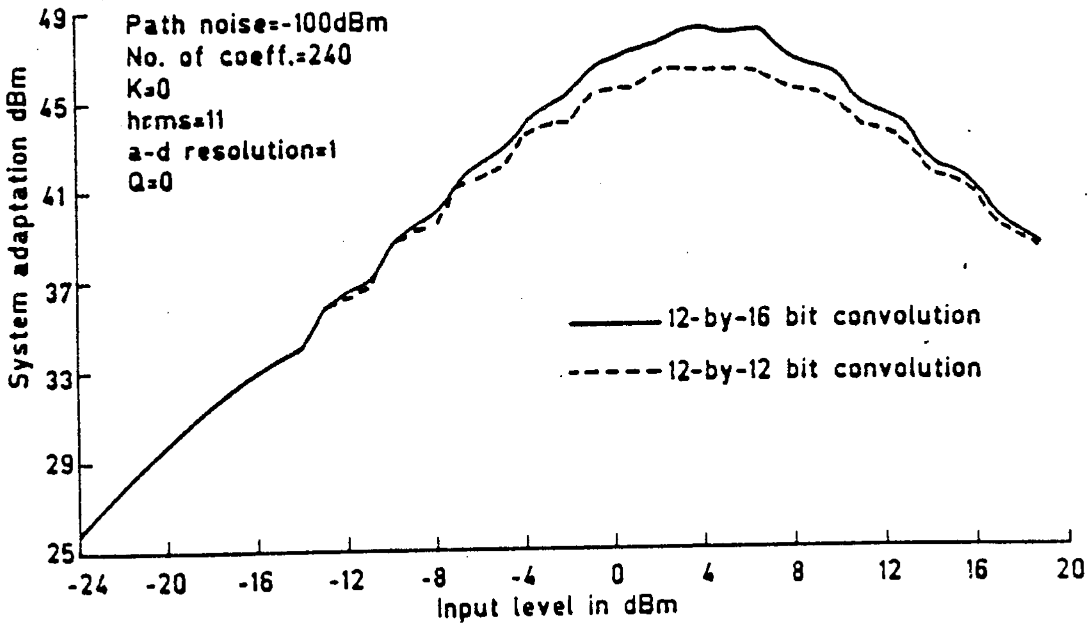


FIG. 5.14 SYSTEM ADAPTATION AGAINST INPUT LEVEL FOR TWO WORDSIZES IN THE CONVOLUTION

Effect of the k factor.

In the loudspeaking telephone one DAF will be positioned across the transmit and receive legs of the 2-to-4-wire converter, modelling the impulse response of the network. There is a minimum trans-hybrid loss of 6 dB when the 2-wire is terminated in either a short or open circuit. Provision has been made to exploit this known loss and improve the precision of the DAF (see Section 2.3.5) by increasing the normalisation constant g by a factor $k = 2^K$. The method of implementation in the hardware (described in Appendix B.2.4) uses a barrel-shifter; hence the technique has become known as g shifting. The improvement to misadjustment noise is seen in equation (5.2), where the quantisation noise due to truncation of the f(2μe) bus and the Δh bus are reduced by $k^2$. The full benefit is only achieved for a reference path with low internal noise, which is generally the case the telephone network. Figure 5.15 shows the predicted and measured advantage in terms system adaptation.

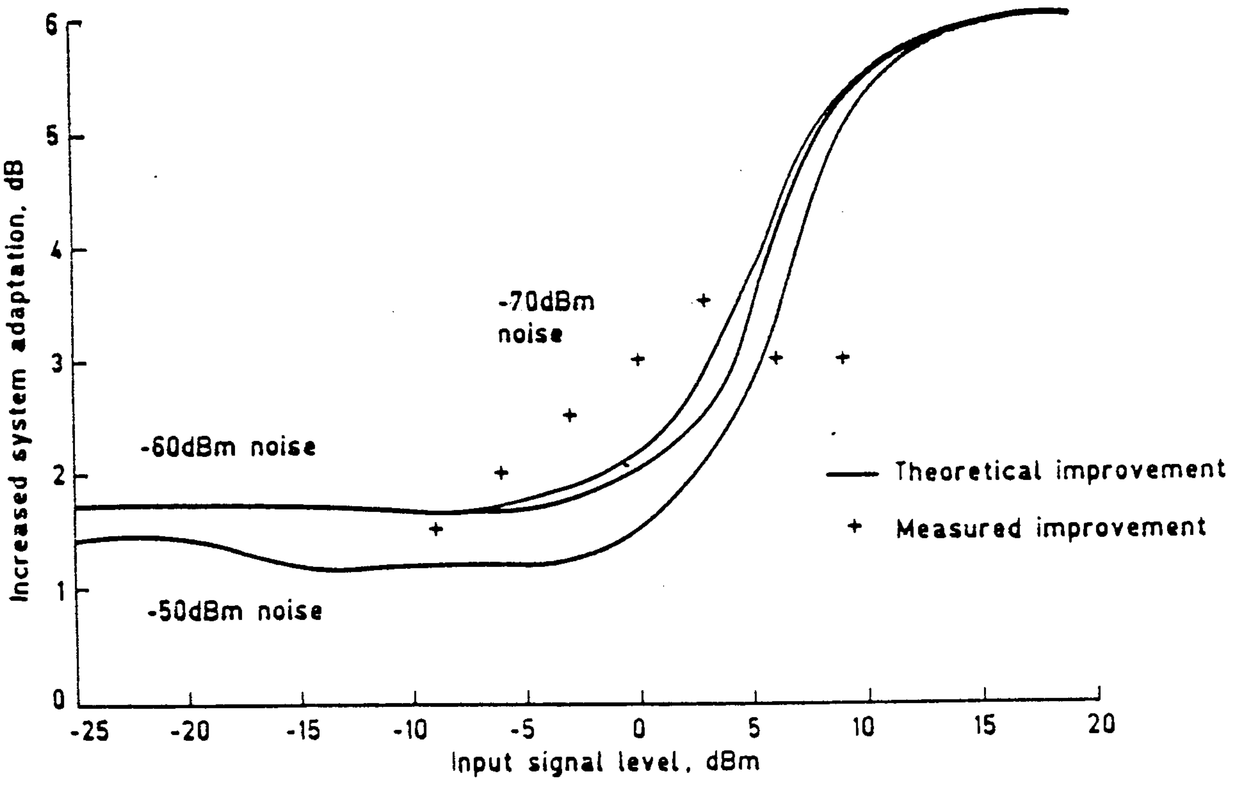


FIG. 5.15 IMPROVEMENT OF A SHIFT FOR ARBITRARY RESPONSE AGAINST SIGNAL I/P LEVEL

## 5.6 Cascading filter sections.

The increase in misadjustment noise caused by cascading filters sections is seen in equation (5.2) for increasing $W$ (the number of coefficients). Both the quantisation noises, due to truncating the $f(2\mu e)$ and $\Delta h$ buses, are increased by $W^2$. Further, the method of forming partial replicas, which are truncated, and then summed to produce the total replica, introduces an additional noise contribution. Figure 5.16 shows the system adaptation for one, two and four filters, i.e. the three cascade configurations ($Q$ = 0,1,2,3). It has also been assumed that the length of the impulse response of the reference path has increased with each cascade, such that the value of $h_{rms}$ remains constant. This is required to solve for $\sigma_v^2$ (equation (3.3)). Hardware measurements are currently only available for the single and dual DAF, due to lack of filters.

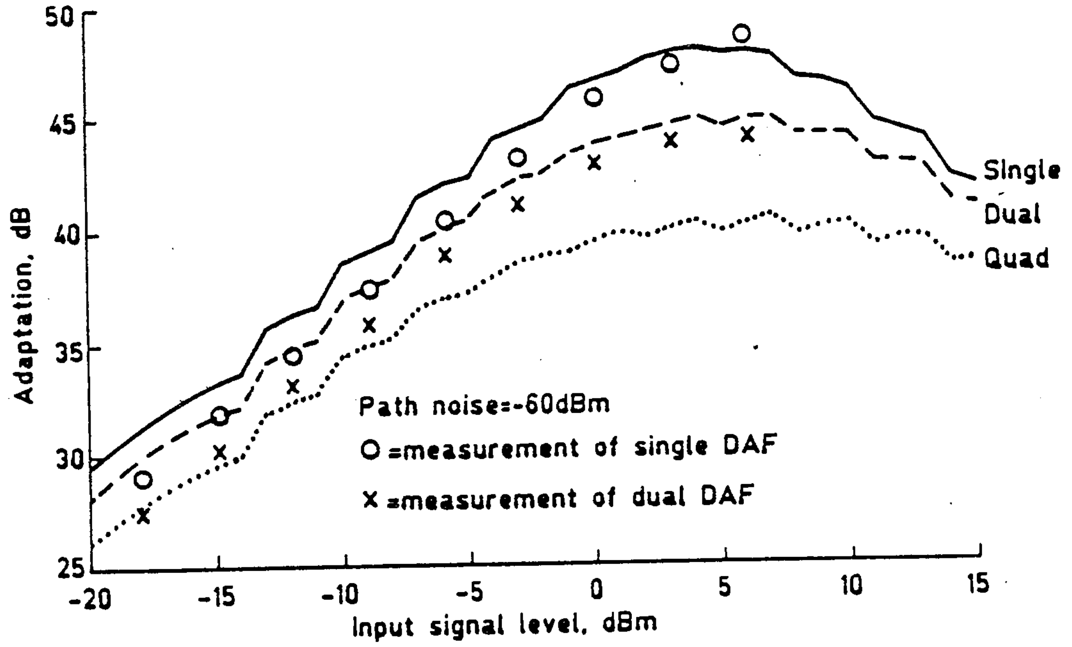


FIG. 5.16 ADAPTATION OF CASCADED DAF's

## 5.7 Q shifting and H extension.

The technique of Q shifting only has benefit for the situation where a poor echo-to-noise ratio exists in the reference path. An example is the acoustic path in the loudspeaking telephone where room noises are likely to be substantial. Equation (5.2) shows that the misadjustment noise is reduced by 3 dB per Q shift. The improvement is see in figure 5.17, which shows the increase in adaptation over that available at Q = 0 as a function of $\sigma_x^2/\sigma_p^2$ (the input signal to reference-path noise), for an input level of 6 dBm. Two points are notable:

i) the region below -30 dB signal-to-noise ratio shows the improvement rapidly diminishing, and, in the case of Q = 2 and 3, actually being worse than at Q = 0. This is also seen in figure 5.3.

ii) in the region to the right of a straight line begining at -2.5 dB on the abscissa the filter is unstable.

The architecture of the DAF allowed the Q-shift facility to be extended beyond that provided in the basic system. The method is described in Section 4.7, it requires reconfiguration of the LSI devices, plus an additional memory device and a fast adder. The structure is such that if an H extension of 4 bits is provided (using an external 4-bit adder), then the Q-shift values become Q = 4,5,6 and 7, i.e. H extension plus nominal Q-shift. The quantisation noise produced by truncating the $\Delta h$ is reduced by $2^{\theta}$, but the effect is small in terms of adaptation. Figure 5.17 shows the expected 3 dB improvement in adaptation relative to that obtained at Q = 0. Using large values of Q it is possible to achieve significant levels of adaptation in very poor echo-to-noise conditions. The penalty is the time take to achieve maximum adaptation. In figure 5.18 three learning are shown for Q = 0, 3 and 6, at an input-signal to path noise ratio of -20 dB. The abscissa requires to be scaled by the value of R to obtain the true value of time. Calculating the rate of adaptation, from equation 5.8 (and substituting q(x) = 0.75, the centre of the bounds), for the Q = 6 curve, gives 3.47 dB/s, compared with a measured value of 3.7 dB/s.

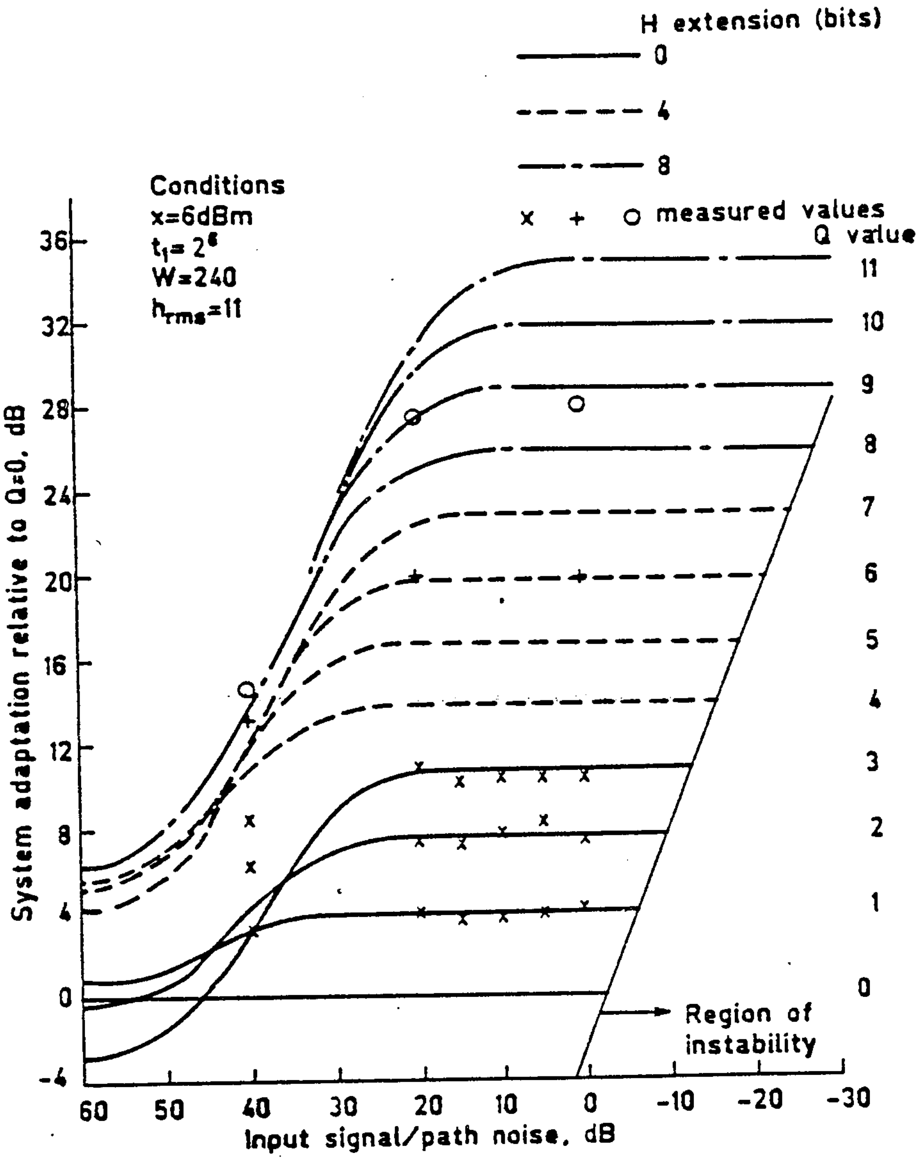


FIG. 5.17 RELATIVE ADAPTATION AGAINST INPUT SIGNAL TO PATH NOISE RATIO FOR Q SHIFT AND H EXTEND

## 5.8 Performance of the SHL algorithm.

In Section 2.3.3 a technique is described for calculating the update vector $\Delta h$ using the SHL algorithm. This method requires that the binary value of error (e) is shifted left by the sum of the most-significant bits of $f(2\mu)$ and x, filling with zeros to the right. The procedure effectively quantises the variable x to its most-significant bit and the shifting provides the multiplication. The change in x is described by equation (2.56).

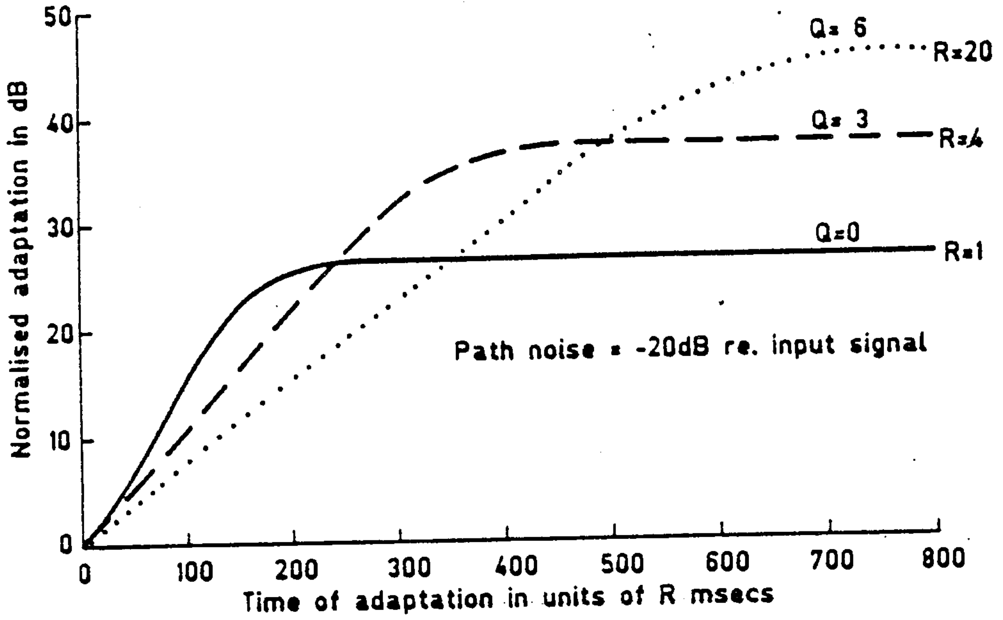


FIG. 5.18 DAF LEARNING CURVE (AVERAGE).

The quantisation of x is nonuniform, producing a noise contribution which is dependent on level. This can be substituted in the theory derived in Chapter 2 to determine the stability criterion and the misadjustment error. The quantisation noise associated with the value of x is calculated for a logarithmic quantisation law (to base 2) and then substituted for x in equation (2.90) to determine the increased contribution to the misadjustment noise. i.e.

$$X \Longrightarrow X + N_s \qquad \ldots (5.11)$$

is substituted in equation (2.90), where $N_s$ is the vector of quantisation noise produced by equation (2.56) (the shift-left algorithm). NB $N_s$ is correlated with X. Using equation (2.97), for brevity, equation (2.95) becomes:

$$E(||D_{j+1}||) = E(||D_j||) +$$

$$\left[\frac{2.\mu}{g.k.k.t_1.t_2}\right]^2.\left[\sigma_d^2 + \sigma_n^2 + \left[\frac{g.k}{2.\mu}.t_1\right]^2.\sigma_b^2\right]$$

$$* \{E(||X||^2) + E(||N_s||^2) + E(2X^TN_s) + W\sigma_{a\_d1}^2\}$$

$$+ \frac{W}{k^2}.\sigma_h^2 + \frac{4.\mu}{k^2.t_1.t_2}.\sigma_d^2 \qquad \ldots (5.12)$$

Comparing with equation (2.95), there are two additional terms: $E(||N_s||^2)$ and $E(2X^TN_s)$. The latter is due to the correlation between the signal and quantisation noise.

Resolving for the misadjustment noise ($\sigma_d^2$), then, in equation (2.105), $\gamma$ is replaced by $\gamma_s$ where:

$$\gamma_s = 1 + \frac{1}{E(||X||^2)}.\left[E(||N_x||^2) + E(2X^TN_s) + W\sigma_{a\_d1}^2\right] \qquad \ldots (5.13)$$

Depending on the value of $\gamma_s$, the stability criterion may have to be reconsidered, in comparison to that used in the full implementation. To solve for the performance of the filter (using the program in Appendix C), it is necessary to determine the components in $\gamma_s$.

Solving for $E(||N_s||^2)$ and $E(2X^TN_s)$.

Figure 5.19 shows a logarithmic representation of the quantisation applied to the x signal by the SHL algorithm. NB, in equation (2.56), $\tilde{x} = 0$ for $x = 0$. The squared error due to signals falling into the ith segment is given by:

$$E_i^2 = \sum_{x=N_i}^{N_i+M_i} (N_i - x)^2.p(x) \qquad \ldots (5.14)$$

where $N_i$ is the lowest integer output-value of the ith segment and $N_i+M_i$ is the upper integer value. p(x) is the probability that a value of x will occur.

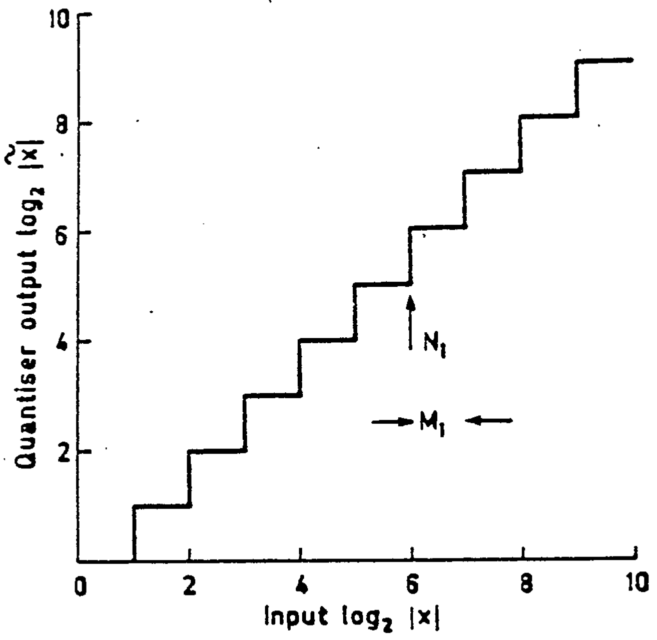


FIG. 5.19 INPUT/OUTPUT RELATIONSHIP FOR QUANTISATION IN SHL ALGORITHM

To solve equation (5.14) accurately requires knowledge of the probability distribution of the signal. However, the quantistion is very coarse and simplifying approximations [304], to produce a closed-form expression for the total noise, fail. If an input signal with a deterministic distribution is used, it is relatively easy to solve equation (5.14) by direct numerical computation (a procedure is given in Appendix C). Figure 5.20 shows the signal-to-quantisation-noise ratio for a signal having a Gaussian distribution, against input level. The worst case, over the f(2μ) compensated region, is 9.5 dB at -16 dBm. The ratio is nearly constant, and it can be substituted for the first term in equation (5.13).

The component $E(2X^TN_s)$ is more difficult to determine. Consequently, it was solved by numerical modelling (a procedure is given in Appendix C). The result is produced in figure 5.21 and shows a near constant value over the compensated region, with a worst-case signal-to-noise ratio of 3.1 db. Substituting worst-case values into equation (5.13), gives $\gamma_s$ = 1.6 compared with an approximate value of unity (assuming the a-to-d noise is negligible) for the full implementation of the update algorithm.

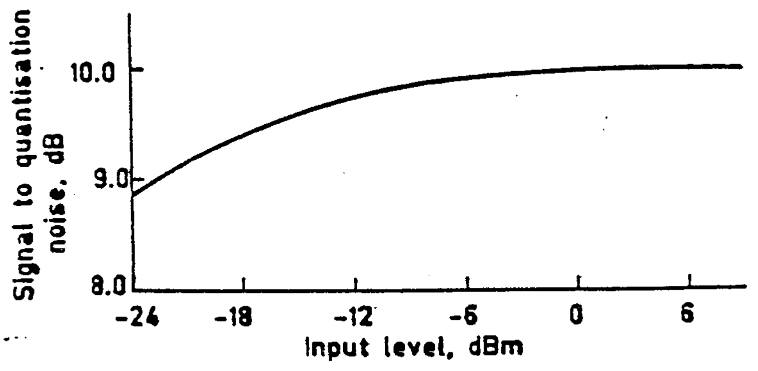


FIG. 5.20 SIGNAL TO QUANTISATION-NOISE FOR THE SHL COMPANDING LAW

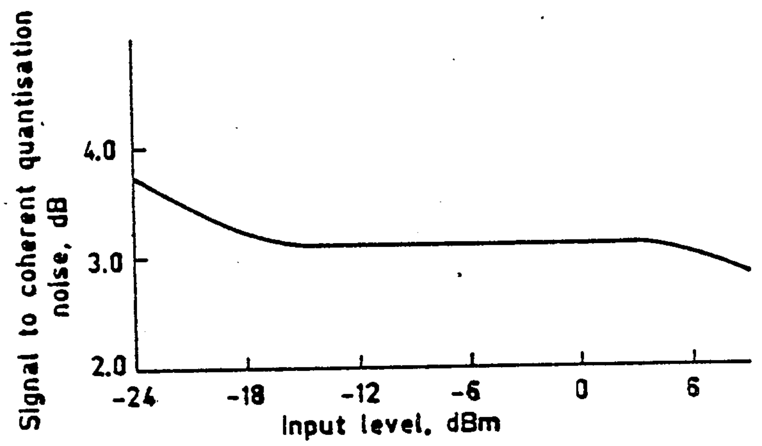


FIG. 5.21 GRAPH OF THE RATIO OF SIGNAL TO COHERENT QUANTISATION-NOISE AGAINST INPUT LEVEL

Following the derivation of equation (3.8), the optimum (fixed value) stability factor over the input range becomes:

$$f\left[\frac{2.\mu}{g.k}\right]_{opt.} = \frac{-g.k.t_1.t_2}{E(||X||^2).\gamma_s} \qquad \ldots (5.15)$$

Thus, the rate of adaptation is reduced, compared to the Widrow-Hoff algorithm by a factor of 0.64 (1/1.6). The reduction in rate of adaptation can be compared with that derived in equation (2.71), which gives a factor of 0.72. Since normalisation in the hardware implementation, using the $t_1.t_2$ product, is only altered in powers of two, the product needs be increased by a factor of 2 to remain stable. Thus, the rate of adaptation would be halved in the hardware implementation (see equation (5.7)).

Substituting equation (5.15) into (2.105), the misadjustment noise is given by:

$$\sigma_d^2 = \sigma_n^2 + \sigma^2{}_b \cdot \left[\frac{W.\sigma_x^2}{g.k.t_2} \cdot \gamma_s\right]^2 + \sigma_n^2\left[\frac{W.\sigma}{g.k}{}_x \cdot \gamma_s\right]^2 \qquad \ldots (5.16)$$

Thus, the noise components due to truncation of the f(2μe) and Δh buses are increased by a factor $\gamma_s^2$. Figure (5.22) shows the predicted coherent-adaptation over the input-signal range, for both the SHL and Widrow-Hoff algorithms. While there is a difference of approximately 2 dB at the Q = 0 value, the SHL algorithm is very near instability and would be used at a maximum of Q = 1. The graph shows there is negligible difference in coherent adaptation between Q = 0 for the Widrow-Hoff algorithm and Q = 1 for the SHL approximation. Therefore, the single advantage of using the Widrow-Hoff algorithm is to achieve the maximum rate of adaptation.

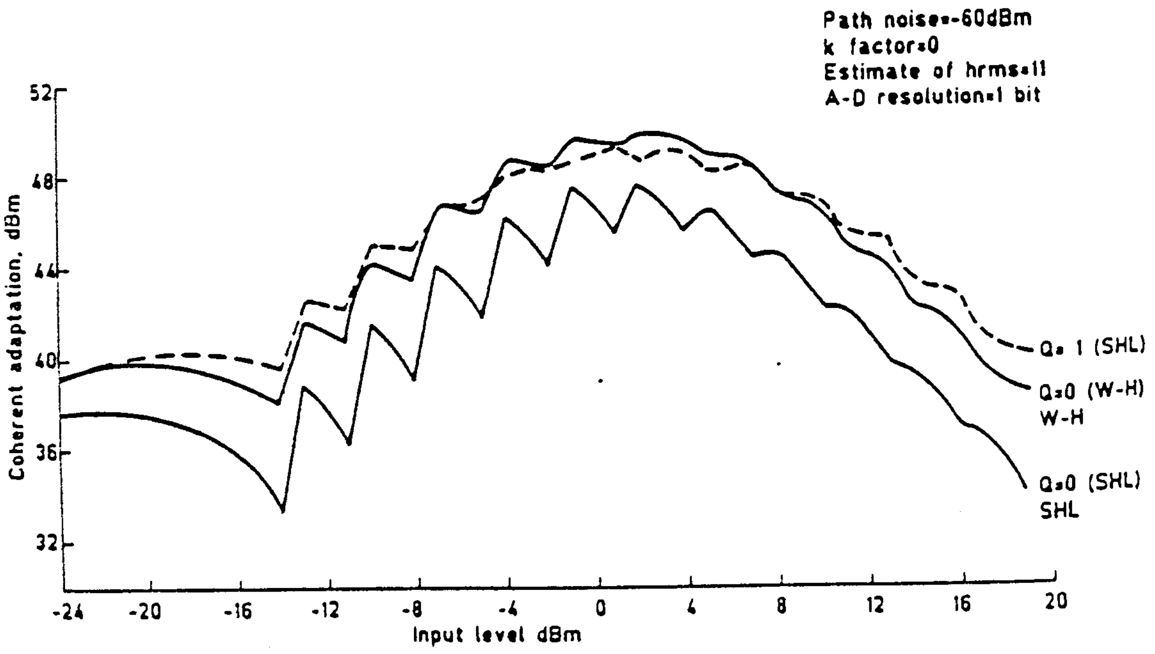


FIG.5.22 COHERENT ADAPTATION AGAINST INPUT LEVEL FOR THE SHL AND WINDOW-HOFF ALGORITHMS

## 5.9 The DAF in an 8-bit logarithmic pcm environment.

The DAF may be used directly in a digital environment, but if it is connected into a pcm signal with logarithmic quantising, e.g. 8-bit A-law [501], the performance can be calculated from theory previously described. An example of this application is in echo cancellation of the trunk telephone-network.

Consider figure 2.5, with the transmission signal having logarithmic quantising. The two noise components $n_{a\_d1}$ and $n_{a\_d2}$ are correlated with x and y (echo) respectively. Thus, the correlated cross-products in equation (2.93) must be resolved. Equation (2.95) becomes:

$$E(||\mathbf{D}_{j+1}||) = E(||\mathbf{D}_j||) + \left[\frac{2.\mu}{g.k.k.t_1.t_2}\right]^2.\left[\sigma_d^2 + \sigma_n^2 + \left[\frac{g.k.}{2.\mu}.t_1\right]^2.\sigma_b^2\right] * \{E(||\mathbf{X}||^2) + + E(2\mathbf{X}^T\mathbf{N}_{a\_d1}) + W\sigma^2_{a\_d1}\} + \frac{W}{k^2}.\sigma_h^2 + \frac{4.\mu.\sigma_d^2}{k^2.t_1.t_2} \qquad \dots (5.17)$$

Following the method described in Section 5.8, resolving for the misadjustment noise ($\sigma_d^2$), then, in equation (2.105), $\gamma$ is replaced by $\gamma_a$ where:

$$\gamma_a = 1 + \frac{1}{E(||\mathbf{X}||^2)}.\left[E(2\mathbf{X}^T\mathbf{N}_{a\_d1}) + W\sigma^2_{a\_d1}\right] \qquad \dots (5.18)$$

Both noise components in equation (5.18) can be resolved, $E(2\mathbf{X}^T\mathbf{N}_{a\_d1})$ by numerical modelling and $W\sigma^2_{a\_d1}$ by analysis [302]. Procedures are included in Appendix C. Figure 5.23 shows the ratio of signal-to-quantising-noise against input level which is applicable for both x and y, i.e. $\sigma^2_{a\_d1}$ and $\sigma^2_{a\_d2}$.

In figure 5.24 the signal-to-noise ratios for the coherent term (first term in parenthesis in equation 5.18) is plotted against input level. This shows that the term contributes significantly to to the reduction of performance over the compensated region. The predicted performance of the DAF

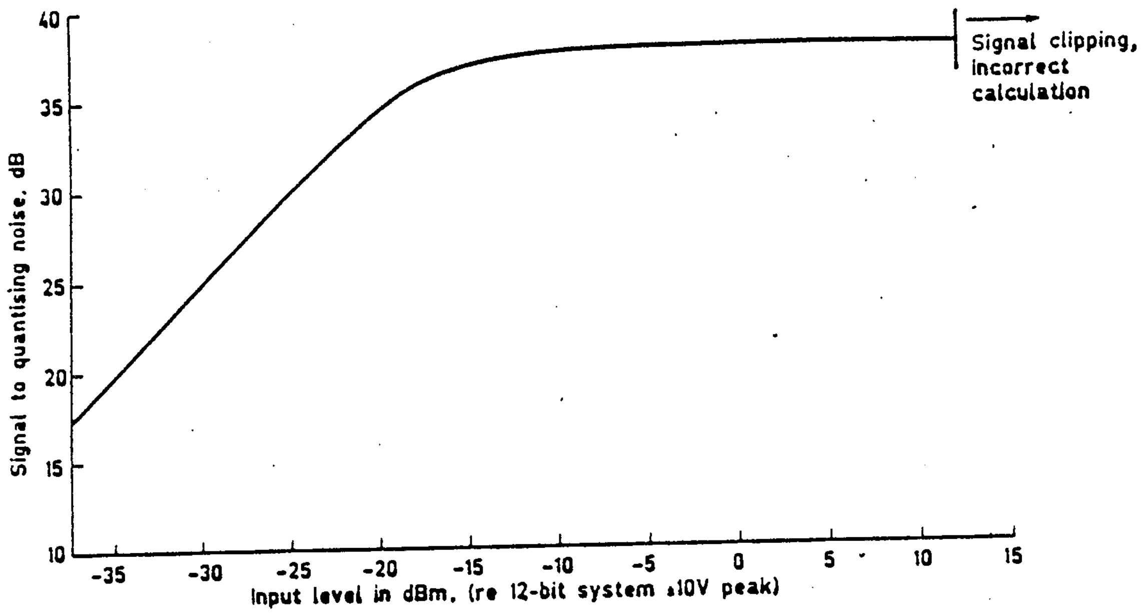


FIG. 5.23 SIGNAL TO QUANTISATION NOISE FOR INPUT TO DAF FROM 8-bit A-LAW PCM

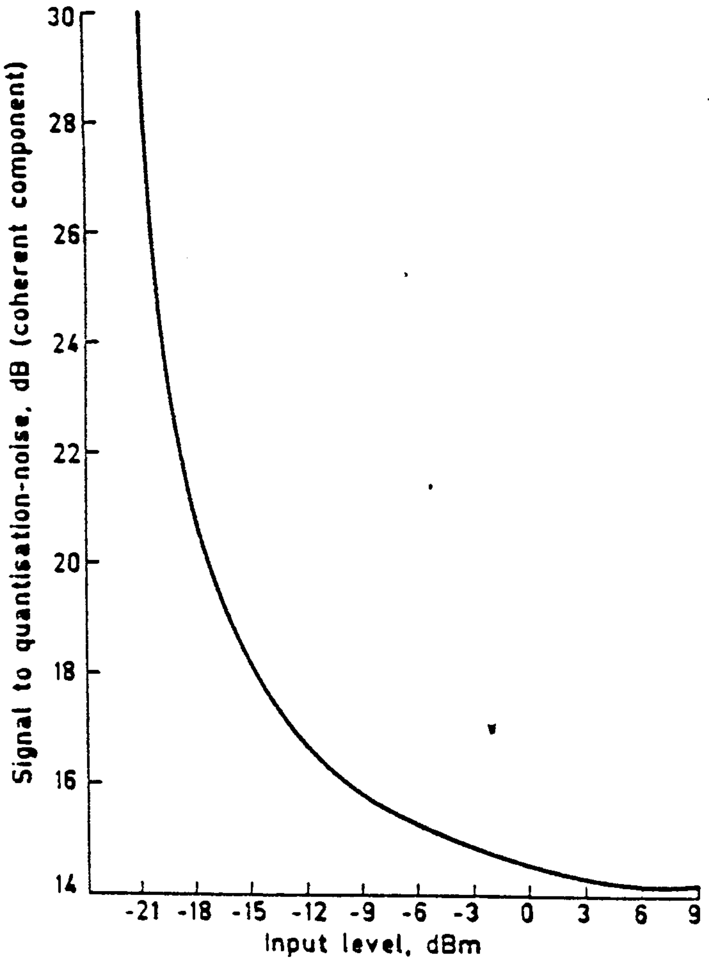


FIG. 5.24 THE SIGNAL TO QUANTISING NOISE FOR THE COHERENT COMPONENT AGAINST INPUT LEVEL

is described in two figures, 5.25 shows the coherent adaptation against input level and the improvement which can be achieved by Q shifting with noise in the reference path, i.e. a-to-d quantising noise. Figure 5.26 shows the system adaptation, where the perfomance is masked by the external quantising-noise. Current work includes the building of logarithmic-to-linear converters so that measurements can be made to test equation (5.18) in (2.105).

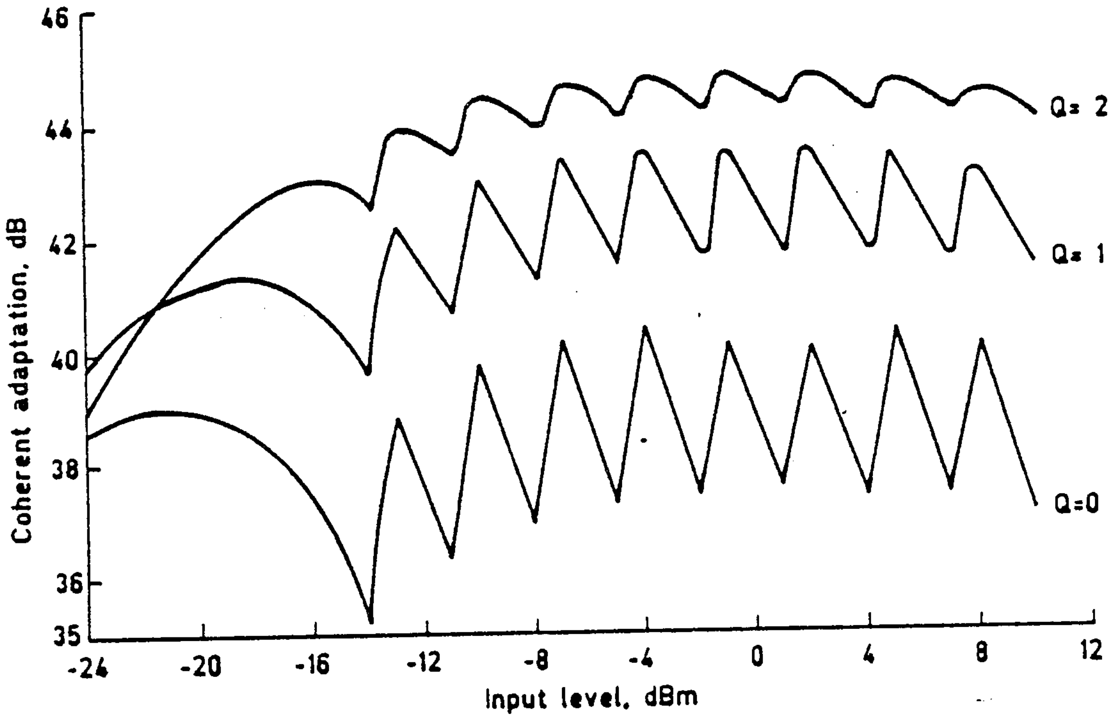


FIG.5.25 COHERENT ADAPTATION AGAINST INPUT LEVEL FOR 8-BIT PCM EXTERNAL ENVIRONMENT

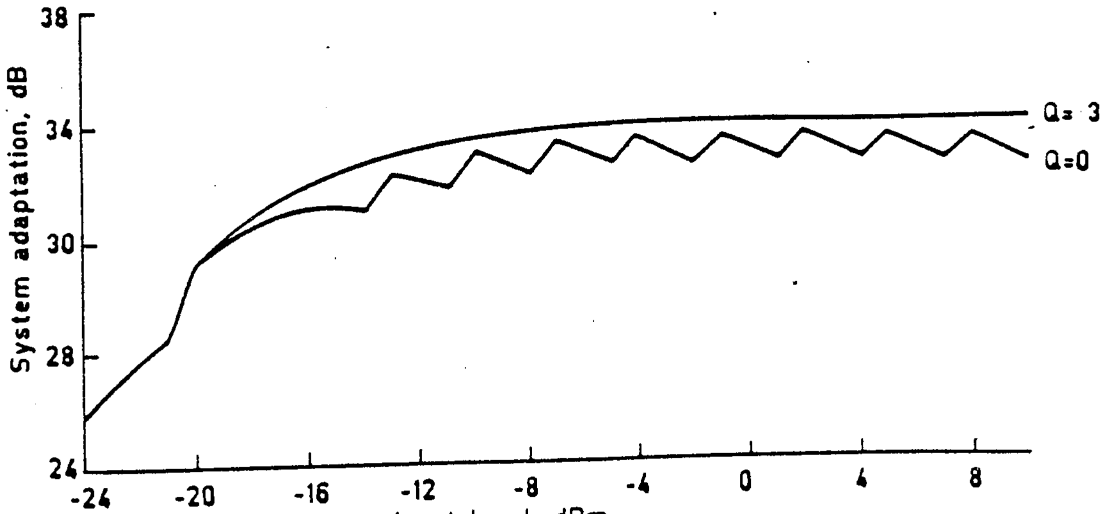


FIG. 5.26 SYSTEM ADAPTATION AGAINST INPUT LEVEL FOR 8-BIT PCM EXTERNAL ENVIRONMENT

## 5.10 Improvement with removal of limit cycling.

At the begining of this project an adaptive filter had been built [502] to test the hypothesis that a duplex LST was feasible. The filter had no facility to remove limit cycles (described in section 3.4.2). Within the constraints of the architecture, two process cycles were removed so that the error could be calculated in sufficient time. It was this work which led to the idea of the filter architecture described in Chapter 4. Figure 5.27 shows learning curves, after the original filter was debugged. The rate of adaptation is 70 dB/s. Figure 5.28 shows the learning curve with the limit cycle removed, the rate of adaptation becomes 110 dB/s, with an approximate 6 dB improvement in the final value of system adaptation.

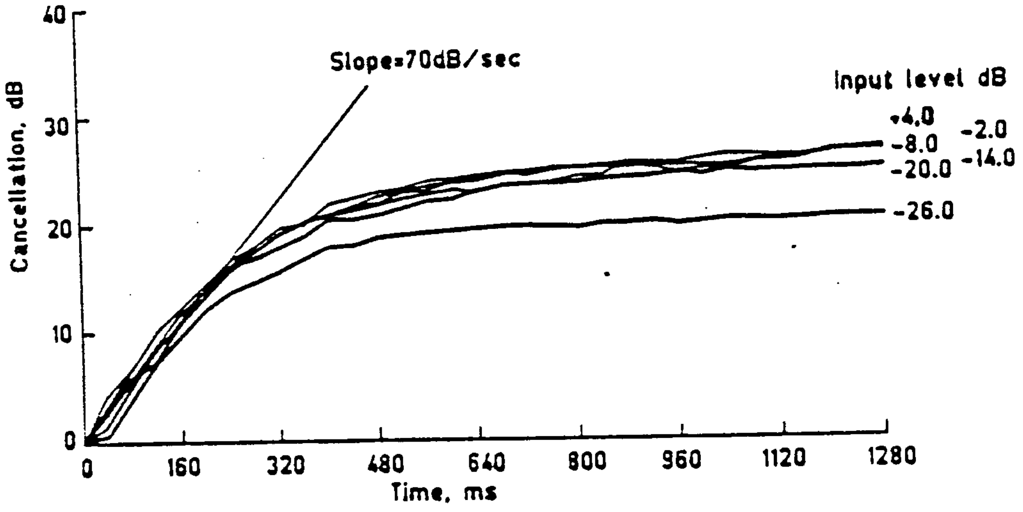


FIG. 5.27 LEARNING CURVE OF FILTER WITH LIMIT CYCLING

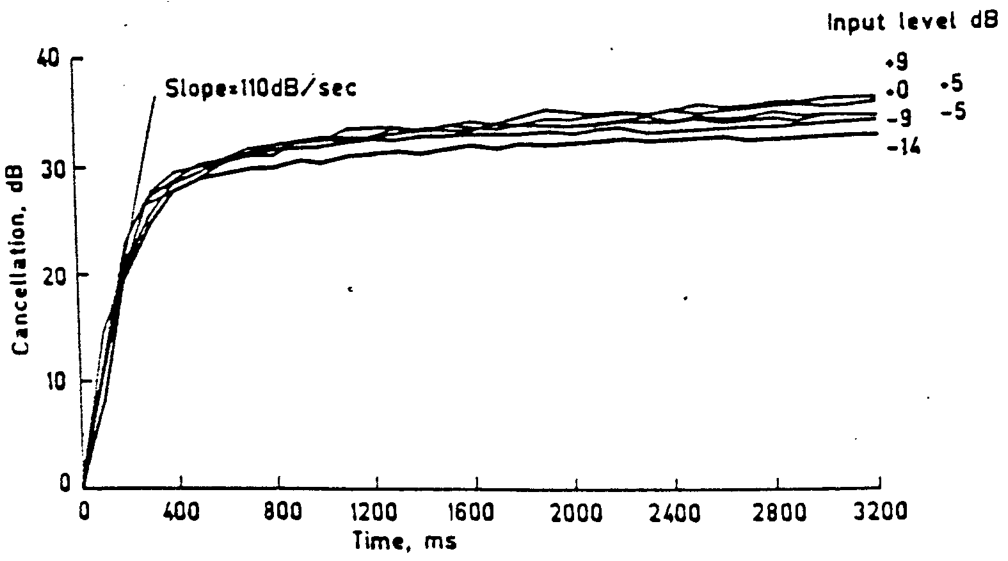


FIG. 5.28 LEARNING CURVE FOR FILTER WITH LIMIT CYCLE REMOVED

# 6. STABILITY AND SIDETONE/ECHO.

The problem associated with duplex on a LST, discussed in Chapter 1, is to maintain stability of a closed loop which would have unity gain for a significant portion of the calls made over the BT telephone network. The technique of using adaptive filters to model the acoustic path (between loudspeaker and microphone) and the hybrid path imposes a limitation on two counts. First, the impulse response of the acoustic path is considerably longer than that which can be accommodated by the current generation of adaptive filters. The second concerns the nonstationarity of the acoustic path which is caused by body movement. During a receive signal the filter can follow changes in the path, with some time lag, but during transmission or silence no update is possible.

A theory is developed which shows that the stability of the loop is dependent on the adaptation achieved and that an estimate of this can be made from real-time measurements.

## Stability Criterion for an LST.

Consider a single LST which is modelled as shown in figure 6.1 with complex impedances, represented as impulse responses, connected by gain (or loss) elements. The components $v_t$ and $v_r$ represent the variable loss in the voice switch. The suffix 'e' refers to the network side of the LST and the suffix 'a' to the acoustic side.

NB $a_{e1}.a_{e2} = a_e$, $a_{a1}.a_{a2} = a_a$

The loop gain can be determined by opening the circuit at point P or Q and injecting a signal $q_i$ or $p_i$, which is compared with the returned signal ($q_o$ or $p_o$). Since multiple convolutions are involved, a transform pair must be defined and the scaling for convolution determined.

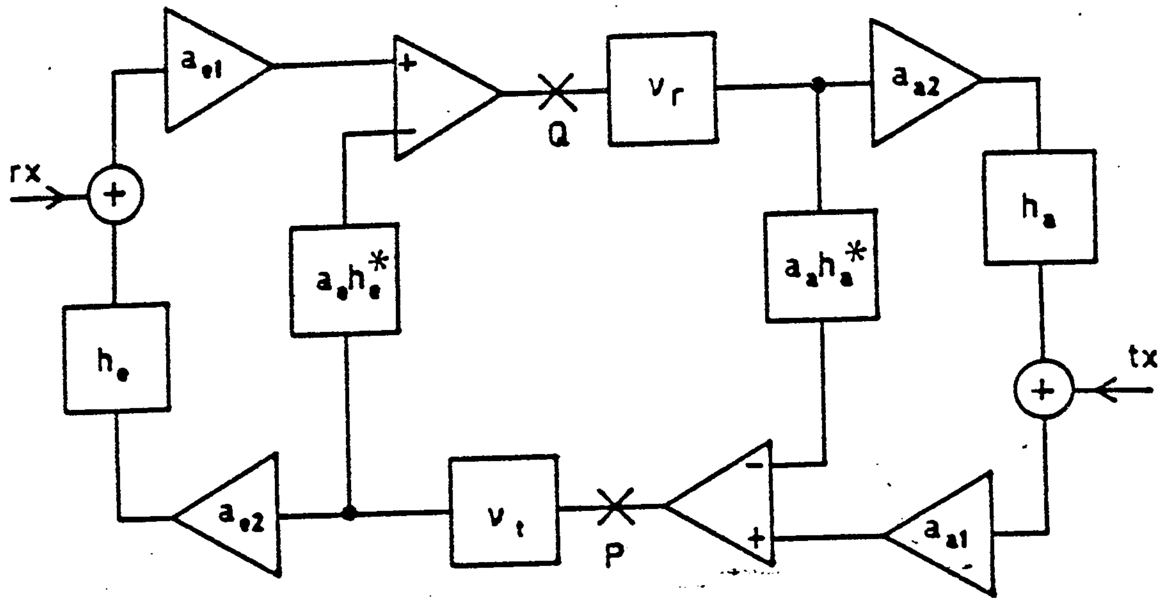


FIG. 6.1 MODEL OF VARIABLE PARAMETER LST

Defining the finite Fourier transform pair:

$$X(n.\Delta f) = \frac{1}{\sqrt{W}} \sum_{p=0}^{W-1} x(p.\Delta t).e^{-j2\pi pn/W} \qquad \ldots (6.1)$$

where $\Delta t.\Delta f = \frac{1}{W}$

and

$$x(p.\Delta t) = \frac{1}{\sqrt{W}} \sum_{n=0}^{W-1} X(n.\Delta f).e^{j2\pi pn/W} \qquad \ldots (6.2)$$

There is an advantage in using the transform pair described, rather than the usual method of scaling in one direction [601], The inherent symmetry eliminates the need to modify one of the domains to obtain Parseval's theorem, for it is easily proved that the total energy E is:

$$E = \sum_{p=0}^{W-1} |x(p.\Delta t)|^2 = \sum_{n=0}^{W-1} |X(n.\Delta f)|^2 \qquad \ldots (6.3)$$

Now consider time-domain convolution in the number range $|g|$

$$r(p.\Delta t) = \frac{1}{g}\sum_{k=0}^{W-1} x((p-k)\Delta t).h(k.\Delta t) \qquad \dots (6.4)$$

Transforming to the frequency domain using equation (6.1)

$$R(n.\Delta f) = \frac{1}{g.\sqrt{W}}\sum_{p=0}^{W-1}\sum_{k=0}^{W-1} x((p-k)\Delta t).h(k.\Delta t).e^{-j2\pi pn/W} \qquad \dots (6.5)$$

Reversing the order of summation

$$R(n.\Delta f) = \frac{1}{g.\sqrt{W}}\sum_{k=0}^{W-1} h(k.\Delta t) \sum_{p=0}^{W-1} x((p-k)\Delta t).e^{-j2\pi pn/W} \qquad \dots (6.6)$$

Let $p - k = m$

then

$$R(n.\Delta f) = \frac{1}{g.\sqrt{W}}\sum_{k=0}^{W-1} h(k.\Delta t) \sum_{m=-k}^{W-k-1} x(m.\Delta t).e^{-j2\pi(m+k)n/W} \qquad \dots (6.7)$$

Using the cyclic property of the domain

$$R(n.\Delta f) = \frac{1}{g.\sqrt{W}}\sum_{k=0}^{W-1} h(k.\Delta t).e^{-j2\pi kn/W} \sum_{m=0}^{W-1} x(m.\Delta t).e^{-j2\pi mn/W} \qquad \dots (6.8)$$

Substituting equation (6.1)

$$R(n.\Delta f) = \frac{\sqrt{W}}{g} H(n.\Delta f)\ X(n.\Delta f) \qquad \dots (6.9)$$

Thus, the coefficient $\sqrt{W}/g$ must be used when transforming a convolution in the time domain to the frequency domain.

Returning to the loop gain calculation associated with figure 6.1 and considering a break at point 'Q' then

$$q_o = \left[\left[\frac{v}{g}{}_r.a_a.q_i{*}h_a^* - \left[\frac{v}{g}{}_r.a_{a2}.q_i{*}h_a + t_x\right]a_{a1}\right]v_t\right]{*}\frac{a}{g}{}_e.h_e^*$$

$$- \left[\left[\left[\frac{v}{g}{}_r.a_a.q_i{*}h_a^* - \left[\frac{v}{g}{}_r.a_{a2}.q_i{*}h_a + t_x\right]a_{a1}\right]v_t\right]\frac{a}{g}{}_{e2}{*}h_e - r_x\right]a_{e1} \qquad \dots (6.10)$$

NB an asterisk used as a superscript means 'an estimate', when used as a dyadic operator means 'convolution'.

This simplifies to

$$q_o = \frac{v_r.v_t.a_e.a_a}{g^2}.(q_i*h_e^**h_a^* - q_i*h_a*h_e^* - q_i*h_a^**h_e + q_i*h_a*h_e)$$

$$+ \frac{v_t}{g}.a_e.a_{a1}.(t_x*h_e - t_x*h_e^*) + r_x.a_{e1} \qquad \text{... (6.11)}$$

Transforming to the frequency domain (scale by $\sqrt{W}$) and simplifying, then

$$Q_o = v_r.v_t.a_e.a_a\frac{W}{g^2}.Q_i\left[(H_a - H_a^*)(H_e - H_e^*)\right]$$

$$+ v_t.a_e.a_{a1}.\frac{\sqrt{W}}{g}.T_x\left[H_e - H_e^*\right] + R_x.a_{e1} \qquad \text{... (6.12)}$$

Let the misadjustment spectrum be defined as

$$D = a(H - H^*) \qquad \text{... (6.13)}$$

then

$$Q_o = v_r.v_t.\frac{W}{g^2}.Q_i.D_a.D_e + v_t.a_{a1}.\frac{\sqrt{W}}{g}.T_x.D_e + R_x.a_{e1} \qquad \text{... (6.14)}$$

The second term on the r.h.s. is the 'double-talk' ($T_x$) component modified by the misadjustment spectrum $D_e$. The third term is the received signal.

Therefore, during silence

$$Q_o = v_r.v_t.\frac{W}{g^2}.Q_i.D_a.D_e \qquad \text{... (6.15)}$$

For stability, then

$$1 > \left|\frac{Q_o}{Q_i}\right| = v_r.v_t.\frac{W}{g^2}.|D_a.D_e| \qquad \text{... (6.16)}$$

i.e the product of the misadjustment spectra, scaled by the loop gain, must be less than unity across the frequency band.

$D_a$ and $D_e$ are complex, therefore

$$|D_a.D_e| = |D_a|.|D_e| \qquad \text{...(6.17)}$$

An instability factor $I_f$ can be assigned to each filter,

$$I_f = \sqrt{\frac{W}{g}}\ |D|_{max} \qquad \dots (6.18)$$

Estimating the probability of instability.

To determine the distribution of $|D|$, the distribution of values in the misadjustment vector **D** must be found.

The input signal to the filter was a Gaussian deviate which was generated from a uniform deviate using the direct method ((3) page 953 [602]) and tested for departure from normality (Table 34 in [603]). Figure 6.2 shows the cumulative probability distribution plotted on a probability scale. Measured values indicate a slight serpentine shape about the ideal straight line. Nominal values for normalised skew and kurtosis are 0 and 3 respectively, the measured standard deviation of these parameters indicate that less than 1% of vectors having 240 or more values will deviate from a normal distribution. Table 6.1 gives the measured standard deviation of skew and kurtosis for three vector lengths.

Applying this Gaussian signal to the filter modelling an arbitrary impulse-response (see figure 5.7) produces a misadjustment vector, after maximum adaptation has been achieved, of the form shown in figure 5.10. Figure 6.3 is a graph of the average distribution of values in the misadjustment vector and shows that it is very close to normal. Testing for skew and kurtosis give values slightly better than those in Table 6.1. Thus, the frequency response of a vector of normal deviates must be found.

Consider the Fourier Transform of a vector of normal deviates

| Vector | | Spectrum |
|---|---|---|
| $z = \partial + j0$ | $\Longleftrightarrow$ | $r = a + jb$ |

where the distribution of r can be found.

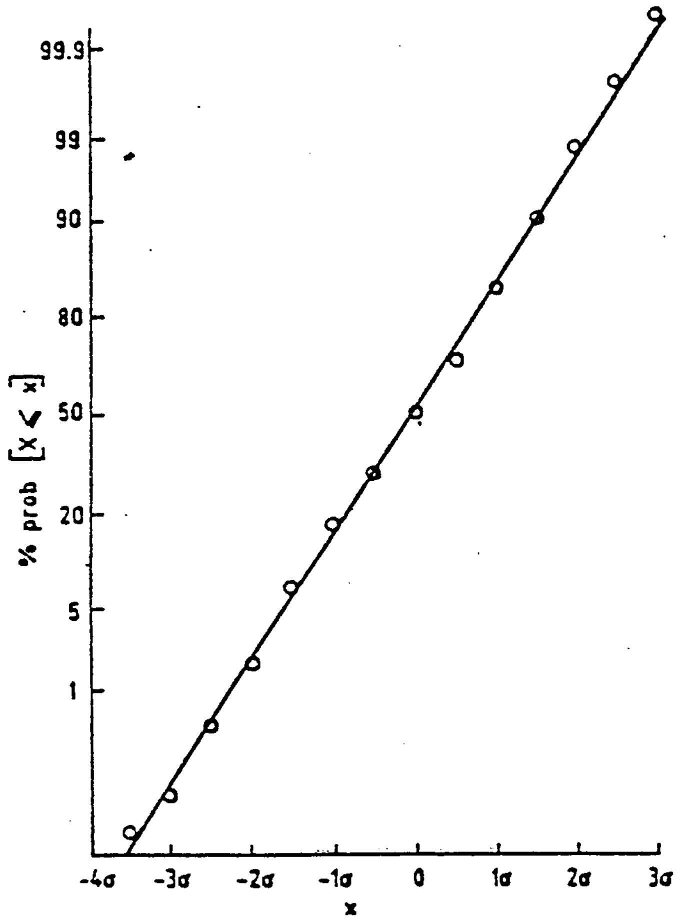


FIG. 6.2 CUMULATIVE DISTRIBUTION OF OUTPUT OF GAUSSIAN GENERATOR FROM THE DIRECT METHOD

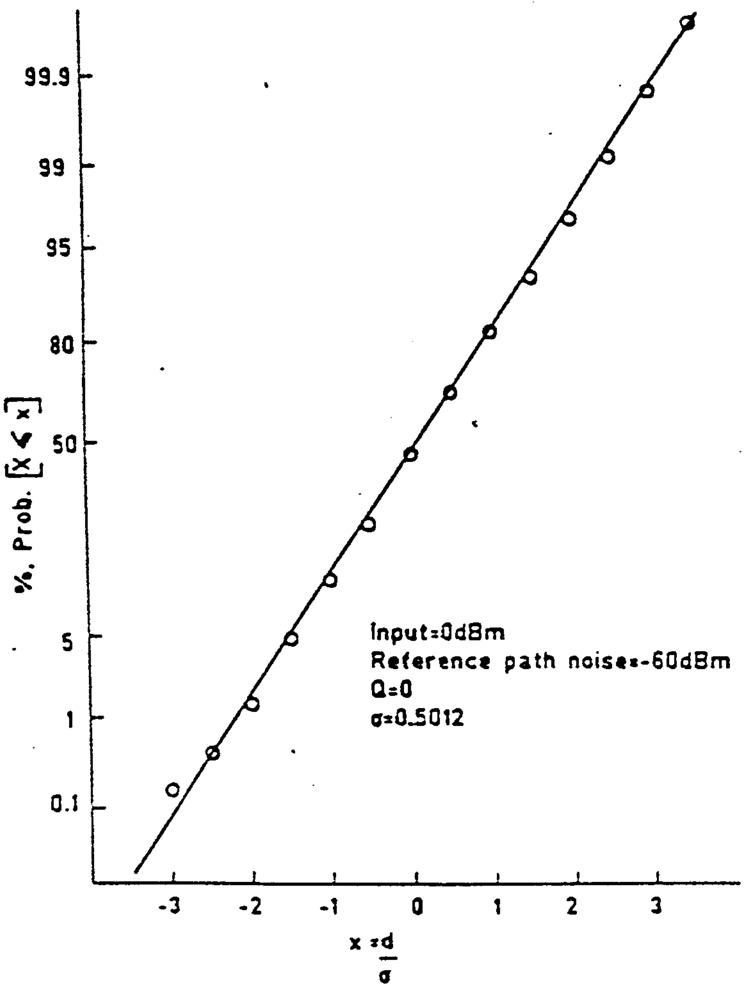


FIG. 6.3 CUMULATIVE DISTRIBUTION OF AMPLITUDE OF MISADJUSTMENT VECTOR

| Vector length | Measured standard deviation | |
|---|---|---|
| | skew | kurtosis |
| 240 | 0.14 | 0.16 |
| 480 | 0.1 | 0.14 |
| 960 | 0.75 | 0.1 |

Table 6.1 Standard deviations of skew and kurtosis for vectors of normal deviates generated by the Direct Method.

Theorem: any real combination of Gaussian variables is Gaussian.

From the above theorem a and b will have a Gaussian distribution, therefore, $|r|$ will have a Rayleigh distribution, with density

$$f(r) = \frac{2r}{\sigma^2} e^{-r^2/\sigma^2} \qquad \dots (6.19)$$

and a cumulative distribution function

$$\text{Prob}[Z <= r/\sigma] = 1 - e^{-r^2/\sigma^2} \qquad \dots (6.20)$$

which is plotted in figure 6.4

The mean is given by

$$\mu_1 = \frac{2}{\sigma^2}\int_0^\infty r^2 \, e^{-r^2/\sigma^2} \, dr \qquad \dots (6.21)$$

$$\mu_1 = \sigma\sqrt{\frac{\pi}{4}} \qquad \dots (6.22)$$

From Parseval's theorem, equation (6.3), the total energy for a W-vector and its spectrum can be written

$$W\sigma_\partial^2 = W(\sigma_r^2 + \mu_1^2) \qquad \dots (6.23)$$

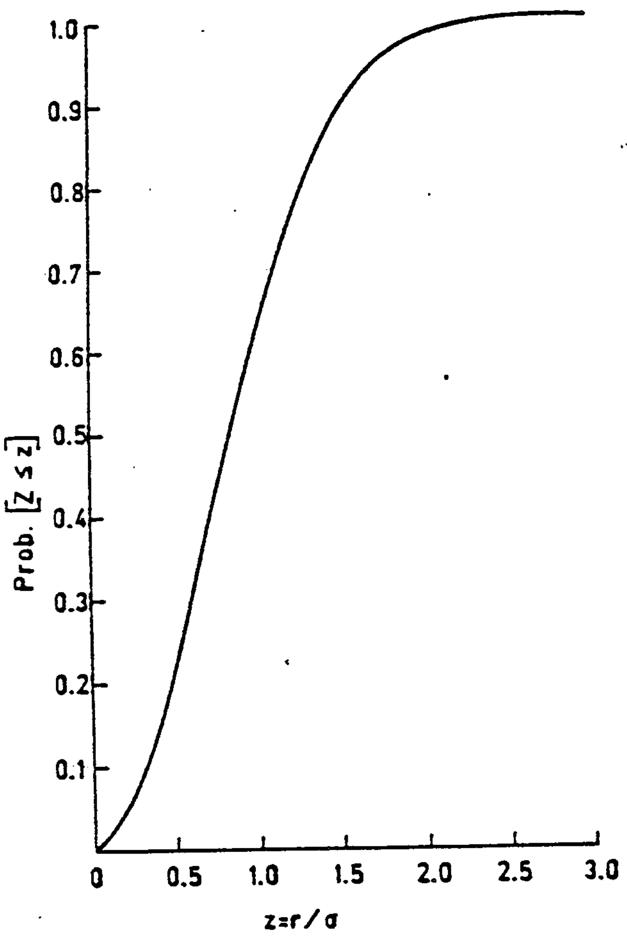


FIG 6.4 CUMULATIVE PROBABILITY OF RAYLEIGH DISTRIBUTION

Substituting equation (6.22) into (6.23), then

$$\sigma_{\partial}^{2} = \sigma_{r}^{2}\left[1 + \frac{\pi}{4}\right] \qquad \ldots (6.24)$$

Thus, the variance of $|r|$ in the frequency domain can be found from the variance of the values in the misadjustment vector.

Let

$$|D|_{max} \approx p\sigma_{r} \qquad \ldots (6.25)$$

where p is chosen to give a small probability of having values exceed $|D|_{max}$, i.e. increasing p and still complying with equation (6.16) will reduce the probability of instability.

Substituting equations (6.24) and (6.25) into (6.18), then the instability factor can be written

$$I_f = \frac{\sqrt{W}}{g} p \, \sigma_r = \frac{\sqrt{W}}{g} 0.75p \, \sigma_\partial \qquad \dots (6.26)$$

For $p = 3$, then, from equation (6.20), 0.01% of peaks will exceed $3\sigma_r$. If these values are uniformly distributed across the spectrum then the probability of peaks in both filters occurring together is 0.0001%, or, at the 8 kHz sample rate, approximately once in 125 seconds.

Rewriting equation (2.82), for the error, in a simplified form,

$$e = \frac{1}{g} X^T D + n_s + n_p \qquad \dots (6.27)$$

where $n_s$ includes all the system noise components (described in Chapter 3) and $n_p$ is the noise in the reference path.

Assuming that $X$, $D$, $n_s$ and $n_p$ are independent random deviates with zero mean and variance $\sigma^2$, then equation (6.27) can be squared and the expectation taken, giving

$$\sigma_e^2 = \frac{1}{g^2} W \sigma_x^2 \sigma_\partial^2 + \sigma_s^2 + \sigma_p^2 \qquad \dots (6.28)$$

where the subscript corresponds with components in equation (6.20) and $\sigma_\partial$ refers to the standard deviation of values in the misadjustment vector.

Rearranging

$$\frac{\sqrt{W}\sigma_\partial}{g} = \frac{1}{\sigma_x}(\sigma_e^2 - \sigma_s^2 - \sigma_p^2)^{1/2} \qquad \dots (6.29)$$

Substituting equation (6.29) into (6.26), the instability factor becomes

$$I_f = 0.75\frac{p}{\sigma_x}(\sigma_e^2 - \sigma_s^2 - \sigma_p^2)^{1/2} \qquad \dots (6.30)$$

where typically $p > 3$.

Thus, the instability factor can be estimated from the time domain and is of the form shown in equation (5.4) for coherent adaptation.

All the system-noise components forming $\sigma_s^2$ can be estimated as shown in equations (3.2) to (3.6), while a measure of $\sigma_p^2$ can be made during periods when no conversation is present. Figure 6.5 shows a plot of equation (6.26) (with p=3) against the ratio of input signal to reference-path noise. Values of $\sigma_\partial$ were measured from the simulation program for the DAF described in Chapters 3 to 5. It shows that the instability factor falls as $\sigma_p$ (the external noise) until $\sigma_s$ (the system noise) becomes the dominant noise contribution.

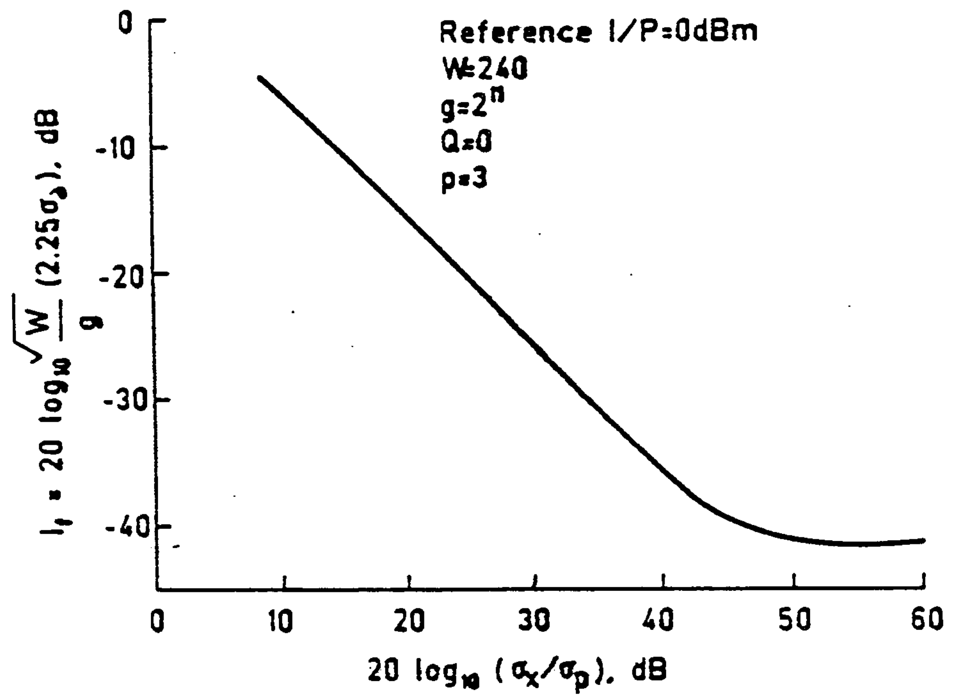


FIG. 6.5 GRAPH OF INSTABILITY FACTOR ($I_f$) AGAINST THE RATIO OF INPUT SIGNAL TO REFERENCE-PATH NOISE FOR A DAF WITH Q=0

At first sight equation (6.30) appears to be a simple solution to the stability problem, but in terms of practical application it is incorrect on two counts. First, Gaussian signals have been used throughout, whereas speech, with its distribution of non-independent samples, would be the norm. The effect would be to alter the distribution of values in the misadjustment spectrum and would require an increase in p to compensate. Secondly, it has been assumed that the impulse response of the reference path has been bounded by that of the filter. While this would be true for the filter modelling the network, it is unlikely to hold for the acoustic path (see Section 2.1). In the latter case the misadjustment vector will have components beyond W, producing a noise term which is only present during speech and correlated with the speech.

From a pragmatic viewpoint, ignoring the system noise and the reference-path noise in the measurement of adaptation (to solve equation 6.30) will lead to a pessimistic result, leaving the system with a greater stability margin.

From equation (6.29) let

$$-M \text{ dB} = 10\log_{10}\left[\frac{1}{\sigma^2_x}(\sigma_e^2 - \sigma_s^2 - \sigma_p^2)\right]$$

i.e. the misadjustment term, then, rewriting (6.16) in dB notation (using capital letters to denote a logarithmic transform) and substituting equation (6.18) and (6.30), the stability condition becomes

$$0 > V_r + V_t - M_a - M_e + 14 \quad \text{dB} \qquad \text{... (6.31)}$$

Actual values for the fixed parameters $a_e$ and $a_a$ in figure 6.1 would be distributed in such a way as to optimise the use of the dynamic range within the system and to maximise the signal-to-quantising-noise ratio. Also, the acoustic path will incorporate terms to account for the sensitivities of the loudspeaker and microphone, so that they can be matched to the electrical-signal levels. For the purpose of analysis, the components of the loop gain G, defined by equation (1.2), can be allocated to particular places in figure 6.1, so that the DAFs will be modelling paths with zero loss. This assumes that the $A_1 - A_m$ relationship for the case as defined in Chapter 1 is of the order of 0 dB (or less)

$A_{e1} = A_{e2} = 3.5$ dB - compensating for the hybrid loss.

$A_{a2} = 0$ dB

$A_{a1} = A_1$ dB - compensating for the acoustic path loss from talker to LST.

$V_r = L$ dB - compensating for the line loss.

Substituting typical values for line loss and measured values for adaptation (with speech), the requirement for $V_t$ to maintain stability can be found.

Rewriting equation (6.31)

$$V_t < M_e + M_a - V_r - 14 \text{ dB} \qquad \ldots (6.32)$$

Consider a connection of LST to telephone handset, where the adaptation of the hybrid DAF ($M_e$) would be typically 30 dB (depending on the network noise, the nonlinear noise produced in the hybrid and the training on speech) and for the acoustic DAF in the range 0 to 10 dB ($M_a$). Table 6.2 shows $V_t$ for best and worst-case conditions. (NB $V_t$ must never be greater than 0 dB.)

Thus, up to 14 dB of voice switching is needed to maintain stability on lines with loss greater than 16 dB.

| Line loss (= $V_r$) | $M_a$ | $M_e$ | $V_t$ (max) |
|---|---|---|---|
| 0 | 0 | 30 | 0 |
| 0 | 10 | 30 | 0 |
| 30 | 0 | 30 | -14 |
| 30 | 10 | 30 | -4 |

Table 6.2 Values of $V_t$ for varying levels of line loss and adaptation of the acoustic path.

The sidetone/echo problem.

Stability provides one condition which must be met, but because of the nonstationarity of the acoustic path and the consequent loss of adaptation by the DAF, the level of sidetone must also be considered. A further complication occurs due to the finite number of coefficients of the DAF, for the desired signal (echo) lying beyond the finite impulse-response of the DAF is highly correlated with the input signal. Thus, the signal returned to the talker via the acoustic path comprises the input signal convolved with the misadjustment vector (which has the first W values of a random nature), plus the remaining impulse response of the room.

At the present time there is no internationally-agreed method of determining sidetone except that of Sidetone Reference Equivalents (STRE) [604]. However this method has a number of drawbacks concerning the range of listening levels, the sidetone path through the handset or bone conduction, room noise and frequency response. The method used by BT is Sidetone Loudness Ratings with Human Sidetone Masking effect (STMR) [605]. The principle behind the preferred use of this system is that there is no subjective improvement in reducing the sidetone beyond a certain limit, set by that heard due to bone conduction. Subjective measurements [606] suggest that, for a telephone connecting to the BT network, an STMR of between 8 dB and 15 dB should be the norm. For the duplex LST this will require subjective assessment since the sidetone heard is transitory.

Consider the LST-to-handset connection shown in figure 6.6, which assumes gains in the LST distributed as in Table 6.2. Then the sidetone $S_t$, due to an acoustic loss of M dB at the LST, heard at the handset is given by

$$S_t = -(M_a + 7 + L) \qquad \ldots (6.32)$$

Substituting values for $M_a$ and line loss as above, then the range of sidetone is shown in Table 6.3.

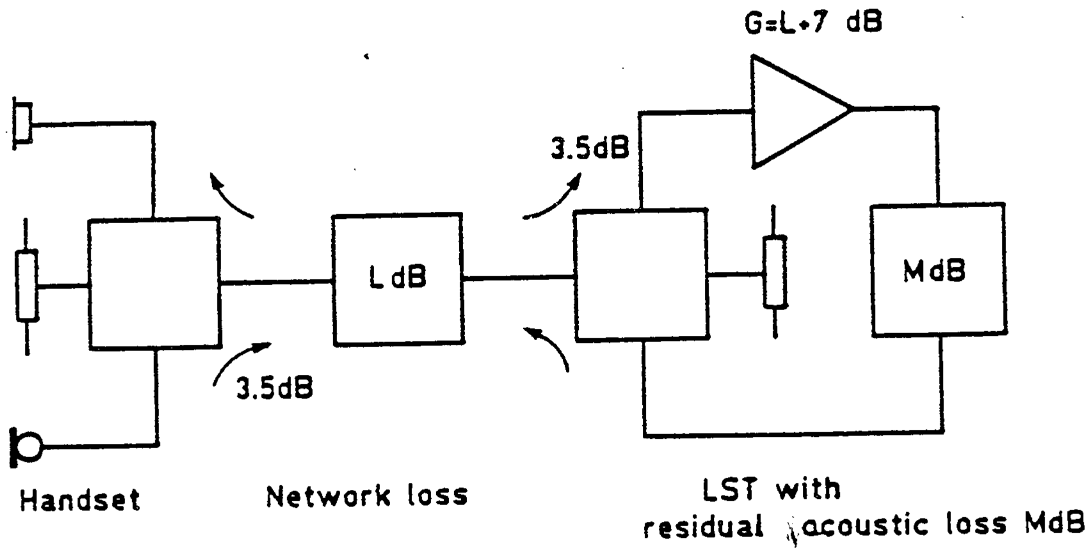


FIG. 6.6 SIDETONE PATH FOR LST TO HANDSET

| Line loss dB | $M_a$ dB | $S_t$ dB |
|---|---|---|
| 0 | 0 | -7 |
| 0 | 10 | -17 |
| 30 | 0 | -37 |
| 30 | 10 | -47 |

Table 6.3 Range of sidetone levels at handset.

Assuming that the low-line-loss case is unsatisfactory (which is the situation from preliminary listening tests on the system), the sidetone must be reduced by increasing the attenuation of the transmit path ($V_t$). The only way that the controller can obtain information about the line loss is to measure the setting of the volume control. At best this can only be very approximate, as the LST user will compensate for his preferred listening level. One solution is to have a minimum loss for $V_t$ which will cause the sidetone heard by the handset user to rise to a maximum tolerable level when no adaptation is obtained on the acoustic path. This value must be found by subjective assessment, but an initial assumption of 6 dB loss has been used, which is the level found so as to maintain duplex operation and to cause a minimum discomfort to users on very short lines, given the poor performance of the DAFs that were available.

This page has been left intentionally blank

# 7 THE EXPERIMENTAL LST.

Applying the condition for stability, determined in Chapter 6, to the case of LST—to—LST calls, the problem of nonstationarity is compounded by the need for each instrument to transmit information to the other (within the speech band) on the performance of its adaptive filters and depth of voice switching. Duplex transmission between LSTs (over a 2-wire link) could not be examined within the timescale of the project. Instead, an architecture for a controller has been developed, and implemented, which exploits the probability that the majority of calls made by an LST user (for some considerable time to come) will be to a handset. The problem of duplex transmission between LSTs is the subject of a current research project.

A further complication caused by the difficulties of modelling the acoustic path is the introduction of new degradation, which is heard as transitory sidetone (or echo) by the handset user. Under conditions of low line-loss and poor adaptation, the perceived sound can be objectionable. The subjective effect will be studied in a future research programme.

## 7.1 Design philosophy.

To achieve minimum cost for the LST, an architecture was required which would have A/D and D/A conversion for the transmit and receive channels only, as shown in figure 7.1. Analogue components comprise the hybrid (2-to-4 wire converter), the transducers, and buffer stages to maximise the dynamic range of the A/D and D/A conversions. In the digital area the adaptive filters and controller would operate.

### Program Control.

The functions required of any controller can be detailed in a top-down description. Actual implementation would be dependent on the hardware available, where those items not done in hardware must be done in software. To facilitate subjective assessment a number of options were provided, e.g.

voice-switching mode only, manual restart and initialise with a training signal. These will be included, although they may not be necessary in a production instrument.

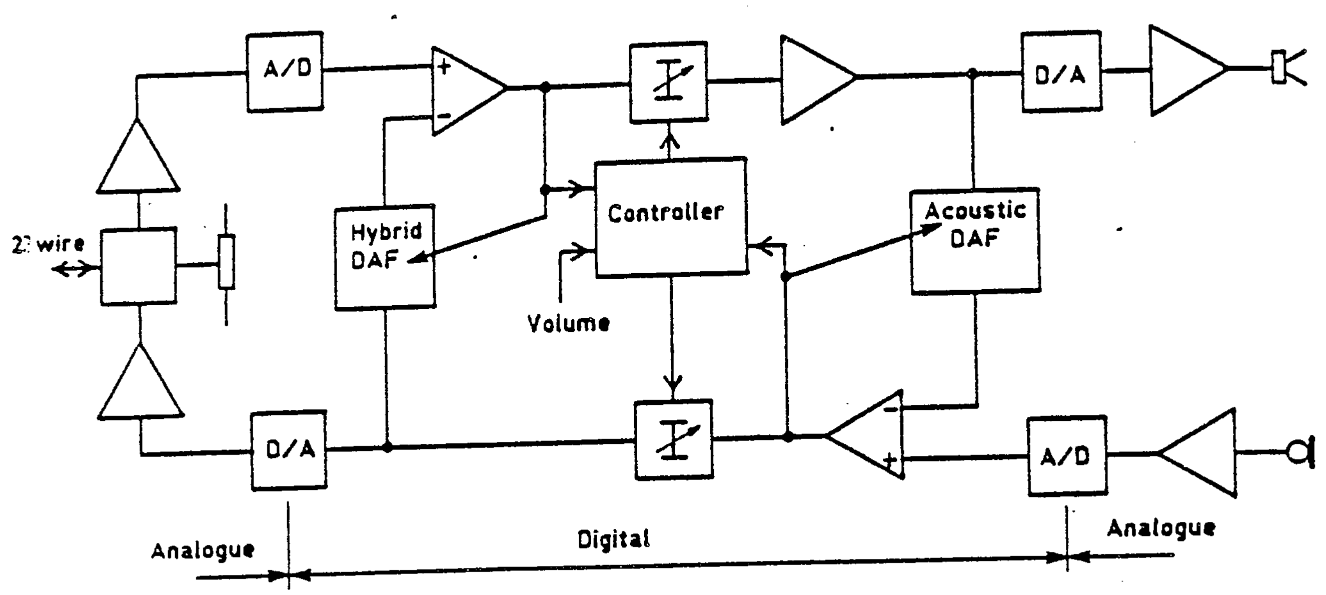


FIG. 7.1 DUPLEX LST WITH IDEAL DISTRIBUTION OF ANALOGUE & DIGITAL COMPONENTS

Top-down description.

1) Initialise the system. From power-up all devices set under software control are initialised, i.e. adaptive filters, input and output ports, constants (which may include arrays) and variables. One variable is the depth of voice switching would be set to maximum.

2) Examine an external switch to determine when LST operation is required. Whilst not in an LST mode, a controller could provide 'on-hook' facilities.

3) Detect LST mode. Scan ports and, if desirable, initialise the system with a training signal. This requires maximum attenuation in both the transmit and receive directions and a Gaussian-noise signal transmitted over one, or both, coupling paths for a short period (typically 500 ms to 2 s depending on the DAF). During this time the corresponding DAFs are put

into adapt mode. Whilst this facility is good in providing duplex operation from the start of a conversation, a short burst of noise to each end may prove objectionable to some people, particularly the distant handset user.

4) Calculate the depth of voice switching needed to maintain stability and that required to achieve an acceptable level of sidetone. The former requires the solution of equations (6.16) and (6.30) which includes both multiplication and division. Because of the number of functions performed by the controller, the calculation can only be done at a relatively slow rate (order of milliseconds) and needs to be filtered to obtain a smooth estimate.

5) Read input port to determine if still in LST mode.

6) Determine if loop is oscillating, if so then restart. This is best done using an interrupt technique.

7) Read transmit and receive levels to determine if a change in the direction of transmission has taken place. If so, then alter the state of a transmit or receive (TX-RX) comparator. It is necessary to signal the direction of transmission and if the direction has changed, so that any residue attenuation in the voice switch can be transferred either abruptly or over a period of time.

8) If transmit condition is dominant, then set up and measure the performance of the DAF across the hybrid. Set rate of adaptation (Q setting) dependent on smoothed estimates of adaptation and noise in the reference path. Set depth of voice switching from previous calculation. If a change of direction has occurred, the transfer of attenuation can be made over a finite period.

9) Estimate whether there is a speech signal present or just residual

noise in the reference path (acoustic in this case). This can be done with a smoothed trough-detector.

If only noise is present. then determine for how long the silent state has existed. It may be necessary to assume that the reference path has changed its impulse response and to ramp attenuation into the voice switch. The program can return to the beginning

10) If a signal is present, set the hybrid filter to converge. This could be for a finite period, given some external timing facility, or simply left until the program reaches this point again. For the latter case, the acoustic filter must be signalled to recirculate as there may have been a change of state in the TX-RX comparator.

11) Estimate a smoothed version of the adaptation for both the adaptive (background) filter and the programmable (foreground) filter. Performance of the background filter can deteriorate either due to adapting in double talk or to the impulse response of the path changing. Depending on which is the better, coefficients can be interchanged. If there is a consistent direction in performance, i.e. deterioration or improvement, then the enhancement is adjusted accordingly. Return program to start (4 above).

12) If at 7 the system is in receive, the steps 8 to 11 are repeated except with different timing conditions due to the probable asymmetry in performance of the two DAFs.

## 7.2 Hardware implementation.

At the beginning of the project there were no digital-signal-processing (DSP) devices available, which could perform the operations described above in the digital domain. Although preproduction information on one device (the Texas 320 [701]) had been released showing that it would be suitable to perform most (if not all) of the functions required. The method adopted to test the feasibility of the LST was to use analogue components for signal processing and interface them to a microprocessor via analogue-to-digital conversion.

To expedite the feasibility study, a commercially-available LST with analogue voice-switching and signal detection was to have foreground-background adaptive filters grafted onto it, across the acoustic and hybrid coupling paths. A block diagram [702] of the analogue LST is shown in figure 7.2. Voice switching was implemented by a pair of balanced modulators - demodulators, constructed so that a single control voltage would drive the transmit and receive attenuators in opposite directions, as shown in figure 7.3. These are called the the transmit (TX) and receive (RX) variolossers.

Operation of the control-line voltage.

The range of loss through the variolossers was 0 dB to 60 dB with the actual values inserted dependent on whether the system was in transmit, receive or idle. In the transmit state the level in the TX direction is dependent on the background noise present (measured by a noise-guard circuit) in the acoustic path, varying by up to 10 dB. This is to compensate for the fact that people tend to talk louder when in a noisy environment. When silence has been detected for approximately 1.5 seconds, the system returns to an idle state which inserts an additional 12 dB loss, into the TX variolosser, below the level determined by the noise-guard circuit. For both conditions the remaining attenuation is inserted into the RX direction, as shown in figures 7.4a and 7.4b. In the receive state the gain or loss in the RX direction is dependent on the setting of the volume control, having a 30 dB range, as shown in figure 7.4c.

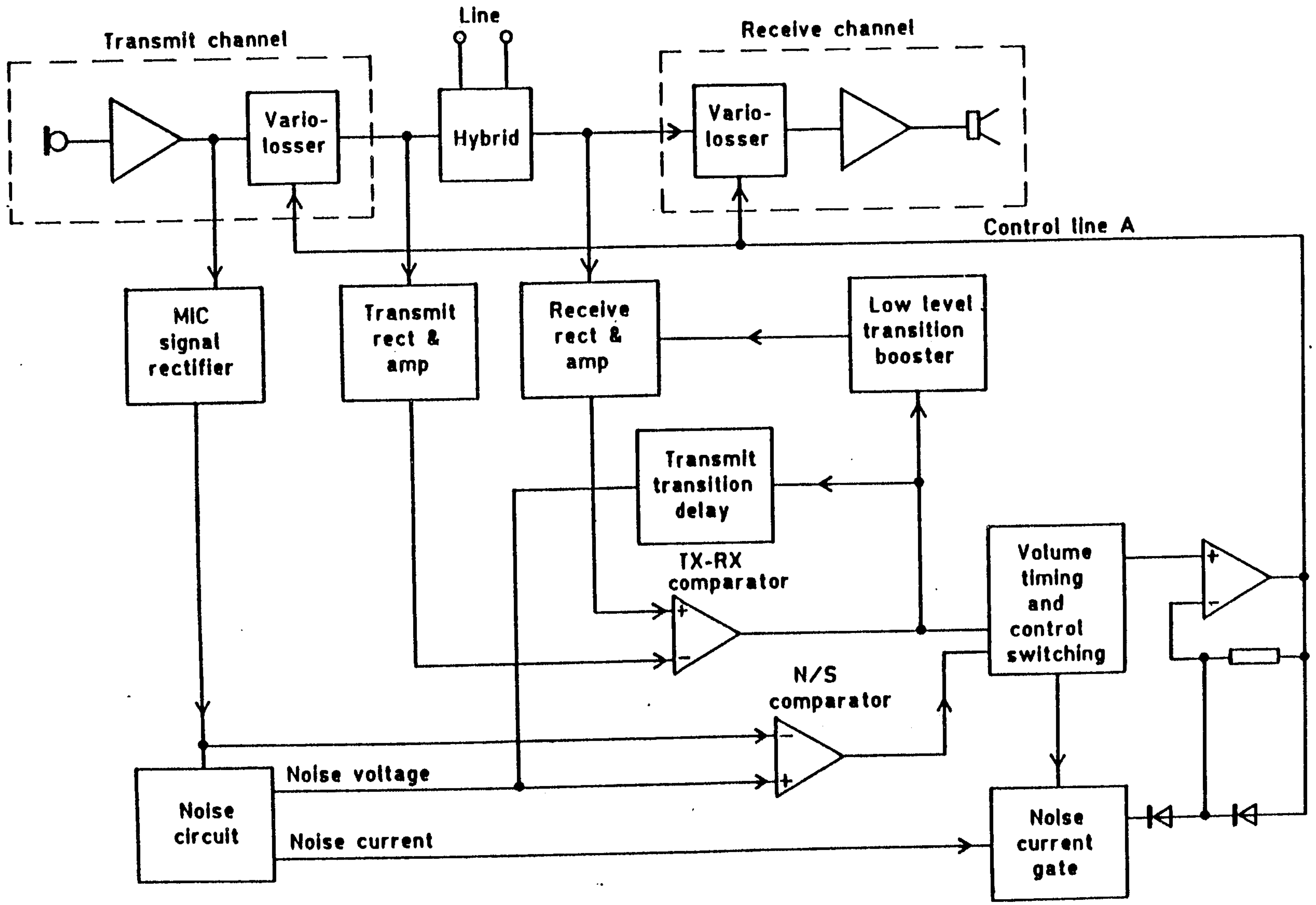


FIG. 7.2 BLOCK SCHEMATIC OF VOICE AND SWITCHING CIRCUIT

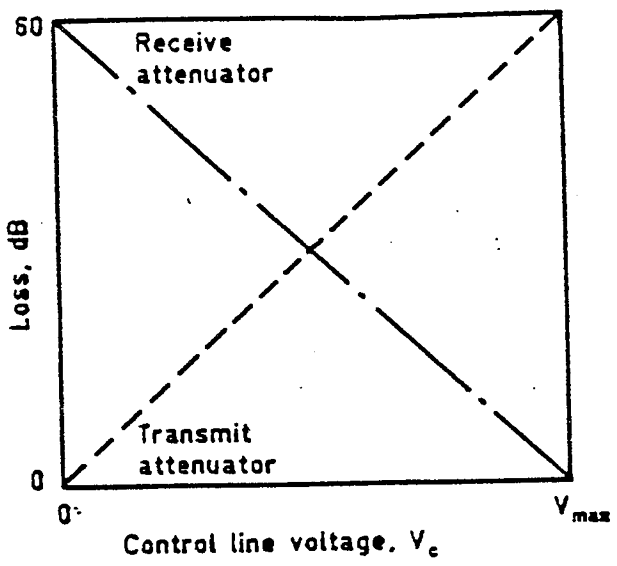


FIG. 7.3 SCHEMATIC DIAGRAM OF LOSS IN VARIOLOSSERS AGAINST CONTROL LINE VOLTAGE

Thus the voltage on the control line is the result of analogue processing of signals to derive three conditions:

a) the local noise-level,

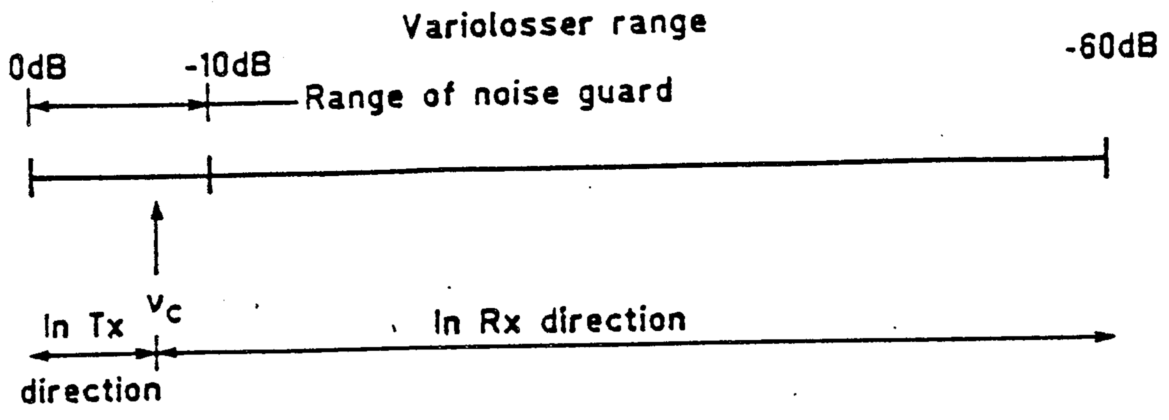
b) the idle state,

c) the volume setting.

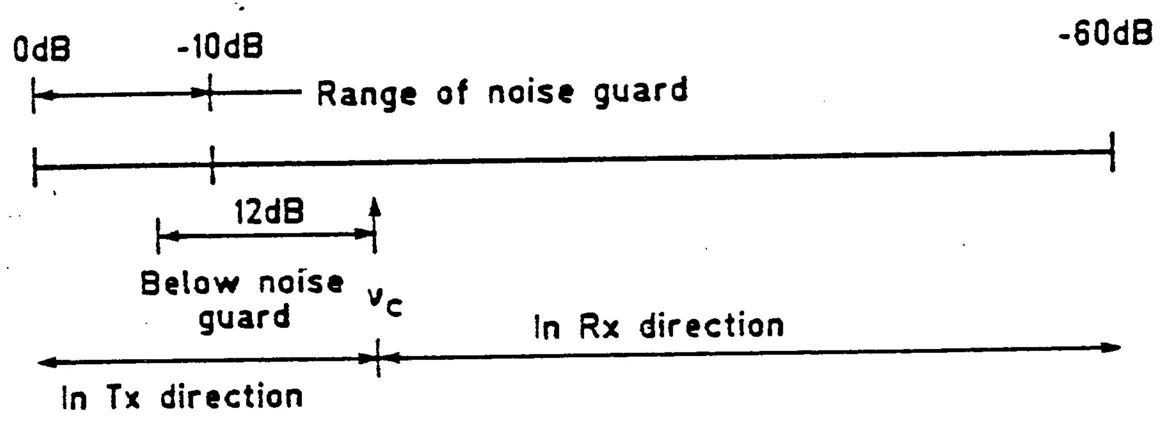
Thus, by converting the control-line voltage to a digital value, these can be detected and stored for use by the microprocessor. All transit times between transmit, idle and receive are also controlled by analogue circuits. Therefore, simply following the change in control-line voltage provides the transition delays.

## Circuit description of the analogue LST.

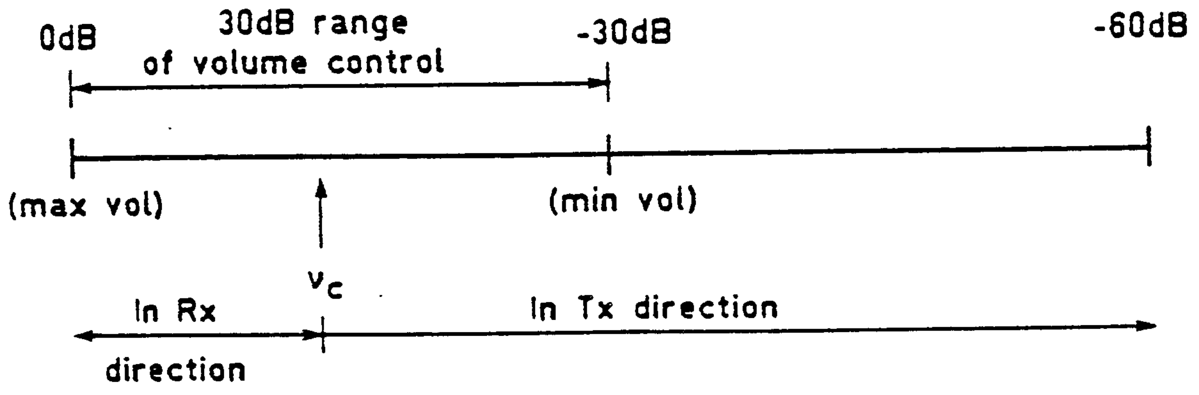
Referring to figure 7.2, the output from the TX variolosser (i.e. the signal to line) was rectified and smoothed (with a fast-attack slow-decay filter), to prevent interjection between syllables or words; similarly with the received signal (from the hybrid). The gain of the transmit rectifier and amplifier was 20 dB over that of the receive rectifier and amplifier. It may be noted that in receive mode the TX variolossers would be at maximum loss so that the LST user could not interject unless he raised the level of his voice. Similarly, when in transmit the additional gain of the transmit rectifier and



a) For transmit state



b) For idle state



c) For receive state

$v_c$ is the control line voltage
(range= 0 to 240 in processor)

FIG. 7.4 DISTRIBUTION OF GAIN IN VARIOLOSSERS 3 STATES

amplifier would prevent distant party interrupting, unless the speech level was raised sufficiently. The two smoothed estimates of TX and RX are applied to a comparator to detect a received or transmitted signal in the absence of the other, or local noise. The transmit signal from the microphone is separately applied to a rectify and smooth circuit which has a slow attack and fast decay characteristic, to follow the level of ambient noise in the room i.e. a trough detector. The circuit has two functions:

a) to reduce the signal sent to line (by increasing attenuation in the TX variolosser) as the local noise increases, because people tend to talk louder in a noisy environment,

b) to bias the LST to transmit when in an idle state (no conversation), so that distant users had a feeling of listening to a 'live' environment.

Two additional circuits were included to bias the system toward receive:

d) a low-level transition booster which operated on a TX to RX condition such that the system stayed in RX for a period, regardless of a possible change back to TX.

e) a transmit transition delay, again operating on a TX to RX condition, which increased the level of local noise measured by the noise/signal comparator so that the TX signal would have to increase further before a changeover could be accomplished.

The object of these circuits was to catch low-level RX signals (caused by line attenuation) and prevent mis-operation due to room reverberation.

Interfacing to the microprocessor and DAFs.

The DAF described in Chapters 3 to 5, was designed to overcome the limitations associated with those available for the first duplex LST (i.e to have a foreground-background architecture, Q-shift facility, automatic f(2$\mu$) compensation and cascading). To achieve foreground-background working, a pair of DAFs [502] were connected as shown in figure 7.5. The error outputs were converted to analogue, rectified and smoothed, then compared to select the one which produced the minimum level for the foreground filter. Simple gating of the recirculate/converge command from the microprocessor was incorporated to ensure that only the background filter was allowed to converge.

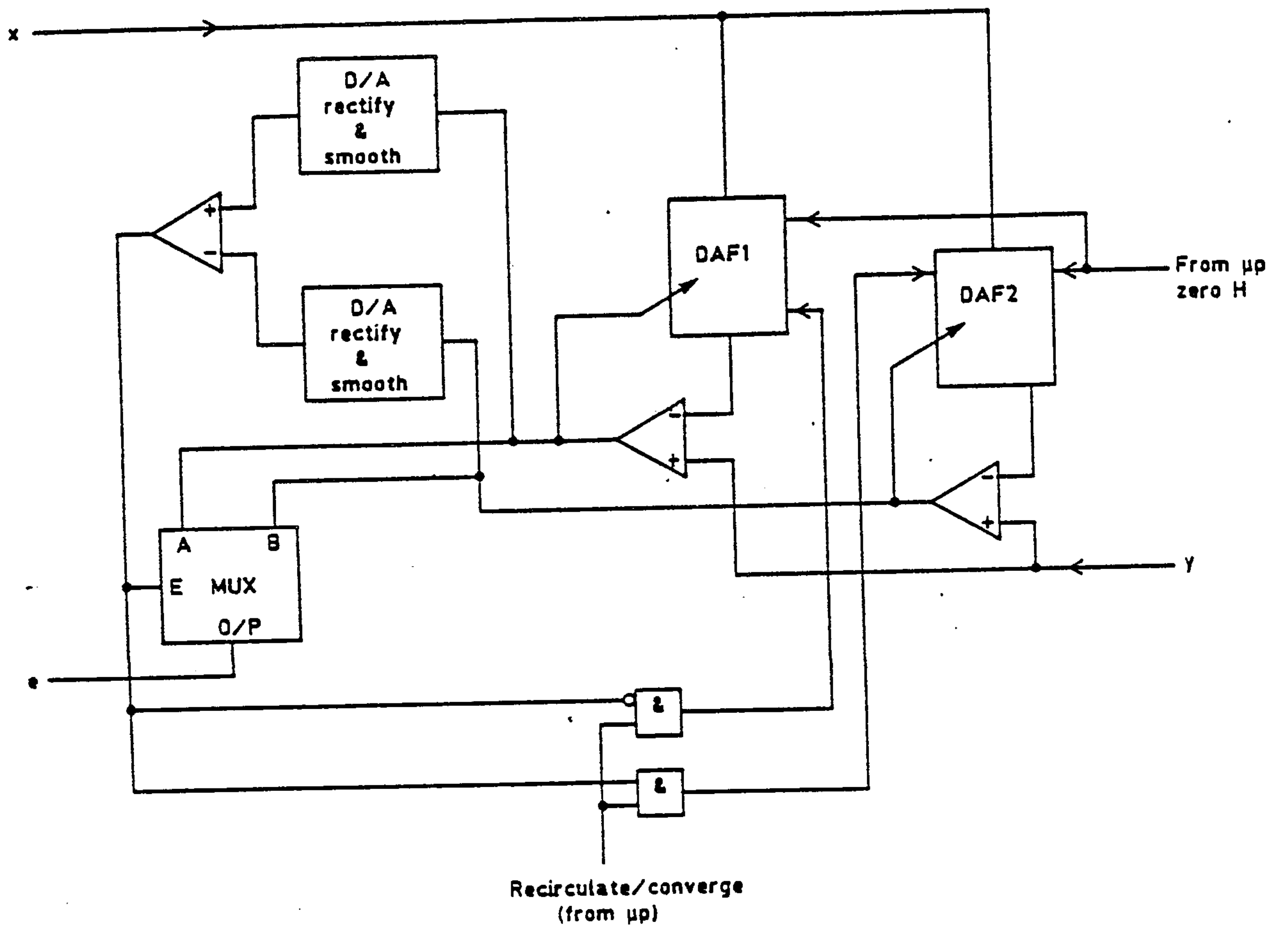


FIG. 7.5 IMPLEMENTATION OF 'FOREGROUND-BACKGROUND' WITH TWO DAFs

A schematic diagram of the full duplex LST is shown in figure 7.6, where interfaces between analogue and digital domains were via conditional buffers, mapping the analogue voltage-range to the digital number-range. Comparing the analogue circuit with that of figure 7.2, a number of changes have been incorporated (apart form the inclusion of the two foreground-background DAFs). The control voltage to each of the variolossers in the transmission paths was intercepted and directly driven by the microprocessor, whereas the control-line voltage from the analogue system is converted to digital to be read by the microprocessor. The 60 dB range (shown in figure 7.3) was mapped in to a

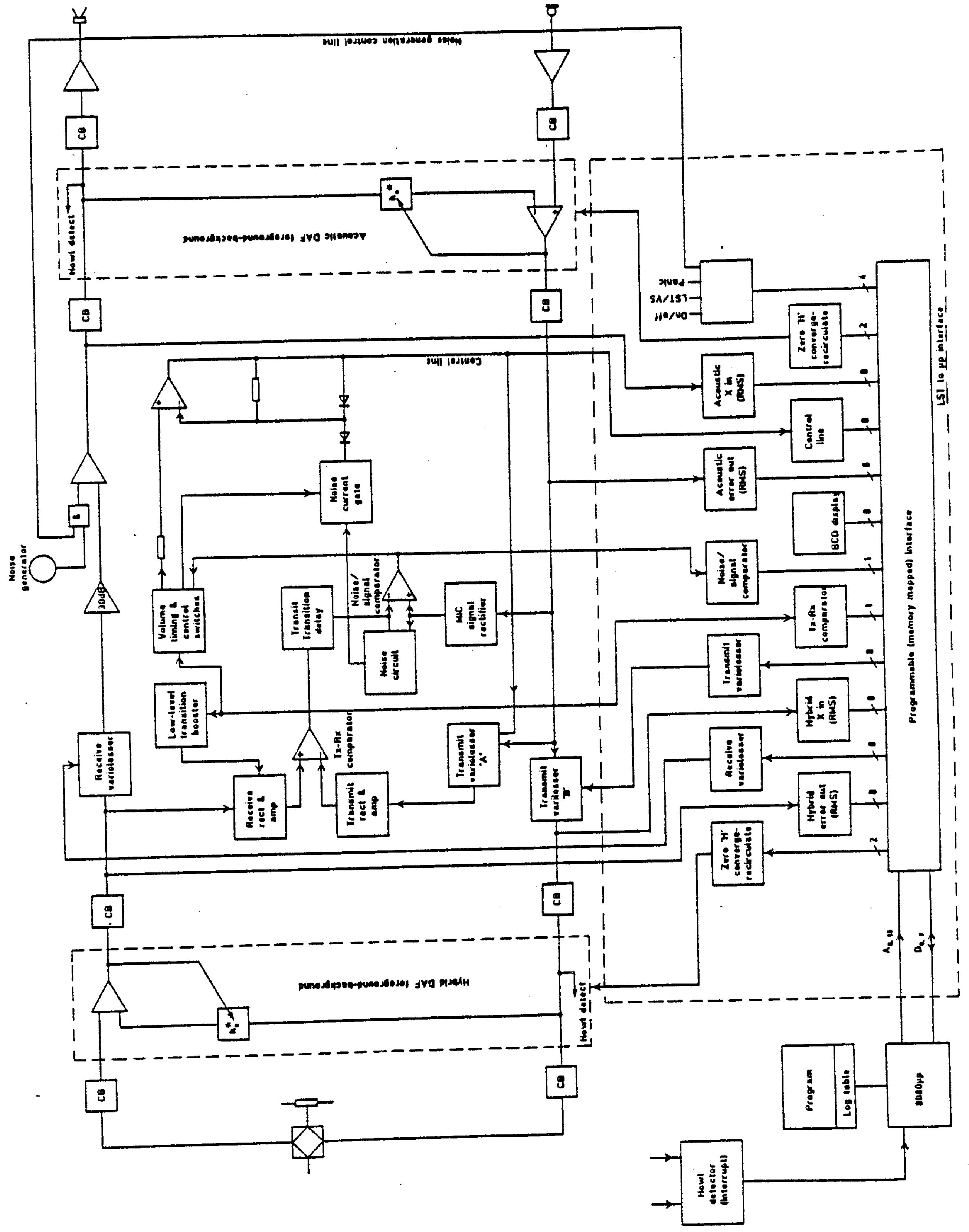


FIG. 7.6 MICROPROCESSOR CONTROL-SCHEMATIC DIAGRAM

number range of 0 to 240, giving 0.25 dB per integer change.

The measurement of adaptation was implemented by converting the analogue x and e to RMS (using an Analogue Devices component AD 536), which was then smoothed and mapped into an 8-bit A/D number range. This gave 48 dB dynamic range over the expected levels of signal in the system (+4 dBm to -44 dBm).

One component which would guarantee to prevent the good operation of the LST was the DAF modelling the acoustic path, having an impulse response of only 32 ms. i.e. no cascading facility was incorporated. Considering a LST to telephone handset conversation, then, in terms of performance, the system is asymmetric and must be treated as such. To achieve reliable operation of the analogue TX-RX comparator in the presence of poor performance of the acoustic filter, hysteresis was obtained by the inclusion of an additional transmit variolosser 'A'. Consider the system in a receive state, where the transmit variolossers 'A' and 'B' are at maximum loss and the receive 'C' is at a minimum (set by the volume control). The adaptation of the acoustic DAF is measured and multiplied by that previously obtained from the hybrid DAF, together with the loss (or gain) of the receive variolosser, which can be measured. Thus, equation (6.16) can be solved (substituting equation (6.30)) to determine the minimum attenuation needed in the transmit variolosser 'B' and still maintain loop stability. Meanwhile, the transmit variolosser 'A' is still at a maximum set by the control-line voltage. Since the adaptation of the hybrid filter can be expected to remain constant (the path is stationary) little power from the output of the transmit variolosser will be input to the receive rectifier and amplifier, thus, the operation of the TX-RX comparator is unaffected.

Consider the LST in a transmit state. Both transmit variolossers ('A' and 'B') would be at minimum attenuation, defined by the noise-guard circuit (in the range 0 dB to -10 dB, see figure 7.4a). During speech transmission the performance of the hybrid DAF is measured and attenuation removed from the receive variolosser ('C') up to an amount determined by the volume setting. The hybrid filter in normal operation will have good adaptation and minimise coupling back into the receive path (and the receive rectifier and amplifier). Thus, the TX-RX comparator will function properly.

## Detection of instability.

The problem of nonstationarity of the acoustic path will now be reconsidered. During transmit, or silence, it is most probable that the adaptation previously achieved for the acoustic path will be lost by body movement and a strategy must be found to deal with the problem. The technique adopted was to start a clock when either of the above conditions was detected and to slowly ramp attenuation into the receive path. Two additional facilities for mis-operation were included which caused the program to return to an initial condition. These were an automatic howl-detector and a manual 'panic-button'. The former is easily implemented by detecting that full-scale has occurred in both the transmit and receive directions, as shown in figure 7.7.

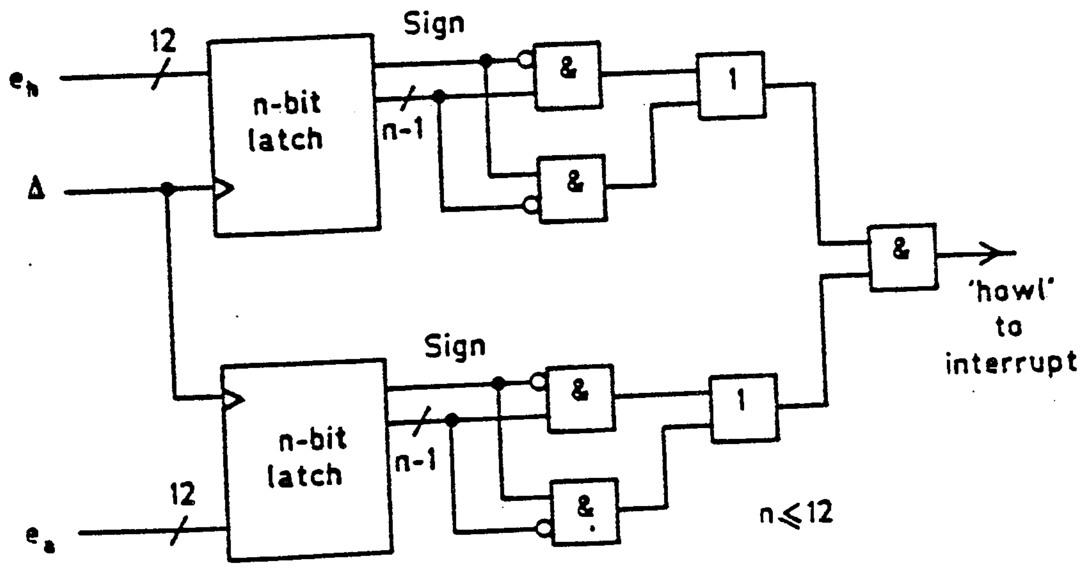


FIG. 7.7 A HOWL DETECTOR

## 7.3 Software implementation.

An Intel 8080 CPU was used as the basic controller, which connected to the LST via 4 programmable peripheral interfaces (PPI - 8255). The PPI has 3 (8-bit) ports, A,B and C, which can be assigned to read/write by sending the appropriate code to a control register. Additionally, the C port can be split into 2 (4-bit) ports. Data words for the control register are reproduced in figure D.1a and D.1b. The I/O map is shown in Appendix D.1. The software was written in PLM on an Intel MDS which provided excellent facilities for debugging in real time using an in-circuit emulator (ICE) for the 8080 device. All arithmetic in PLM is carried out in unsigned-integer form, although provision is made to determine if an overflow has occurred. Multiplication and division are also implemented, but these are computationally time consuming. To assist in the calculation of the stability margin in dB 20.log (base 10) and antilog lookup tables were provided. Numbers in the range 0 to 1023 (10 bits) were quantised into 61 logarithmic values as shown (partly) in figure 7.8. The dynamic range is 60 dB.

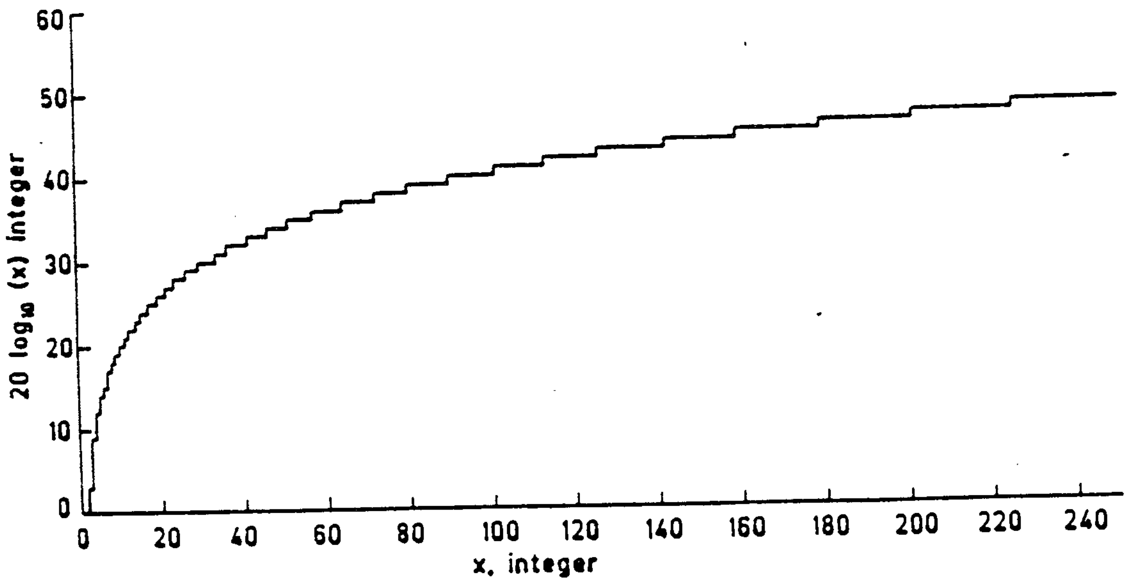


FIG. 7.8 GRAPH OF INTEGER APPROXIMATION FOR LINEAR⟷LOG TRANSFORMATION

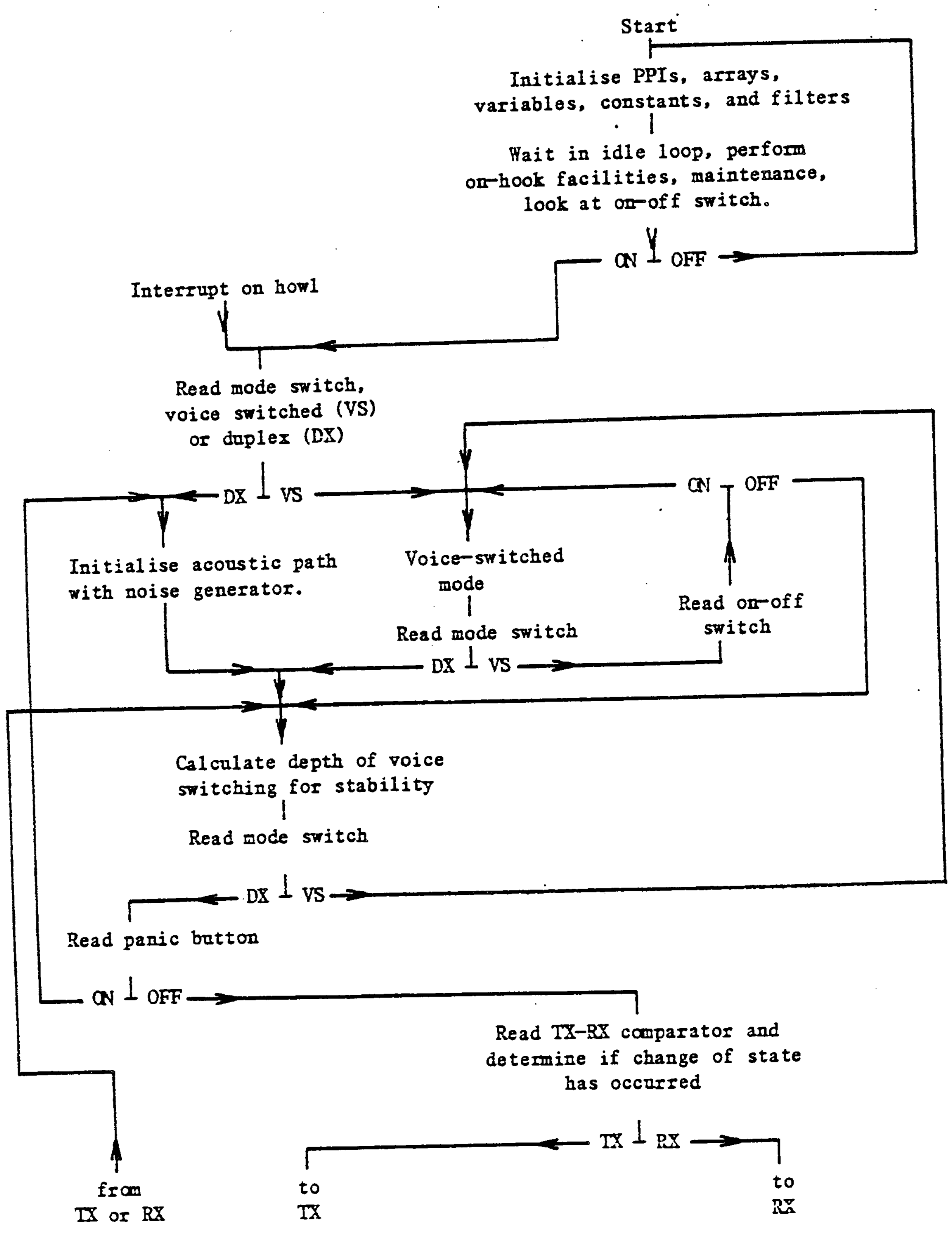
The software is reproduced in Appendix D.3 and consists of a main program plus 9 subroutines. These are

| | |
|---|---|
| MAIN | - which examines switches, signals A/D for measurements, reads in TX-RX and SIG-NOISE comparators, filters the acoustic enhancement (fast attack, slow decay) and controls the calling of all subroutines. |
| START | - sets up all PPIs and log table. |
| IDLE | - state when not in use. |
| V_SWITCH | - operates the system in fully voice-switched mode. |
| AC_HYB | - sets up the acoustic path with a noise source and estimates the overall acoustic loss available. |
| VS_DEPTH | - calculates the level of voice switching required for stability. |
| ACTPTH | - measures adaptation of acoustic path. |
| ELHYB | - measures adaptation of hybrid path. |
| VSTC | - calculates $V_t$ and $V_r$ when in a transmit state. |
| VSRC | - calculates $V_t$ and $V_r$ when in a receive state. |

Most variables used in the program are self explanatory and particular address locations are given meaningful names. Control signals sent to PPIs can be decoded from figures D.1a and D.1b.

Figure 7.9 gives a detailed flow diagram of the program which can be used in conjunction with Appendix D to see the exact method of implementation.

Figure 7.9 Flow diagram of the controller.



Start
Initialise PPIs, arrays, variables, constants, and filters
Wait in idle loop, perform on-hook facilities, maintenance, look at on-off switch.
ON OFF
Interrupt on howl
Read mode switch, voice switched (VS) or duplex (DX)
DX VS
ON OFF
Initialise acoustic path with noise generator.
Voice-switched mode
Read mode switch
Read on-off switch
DX VS
Calculate depth of voice switching for stability
Read mode switch
DX VS
Read panic button
ON OFF
Read TX-RX comparator and determine if change of state has occurred
TX RX
from TX or RX
to TX
to RX

Figure 7.9 Flow diagram of the controller continued.

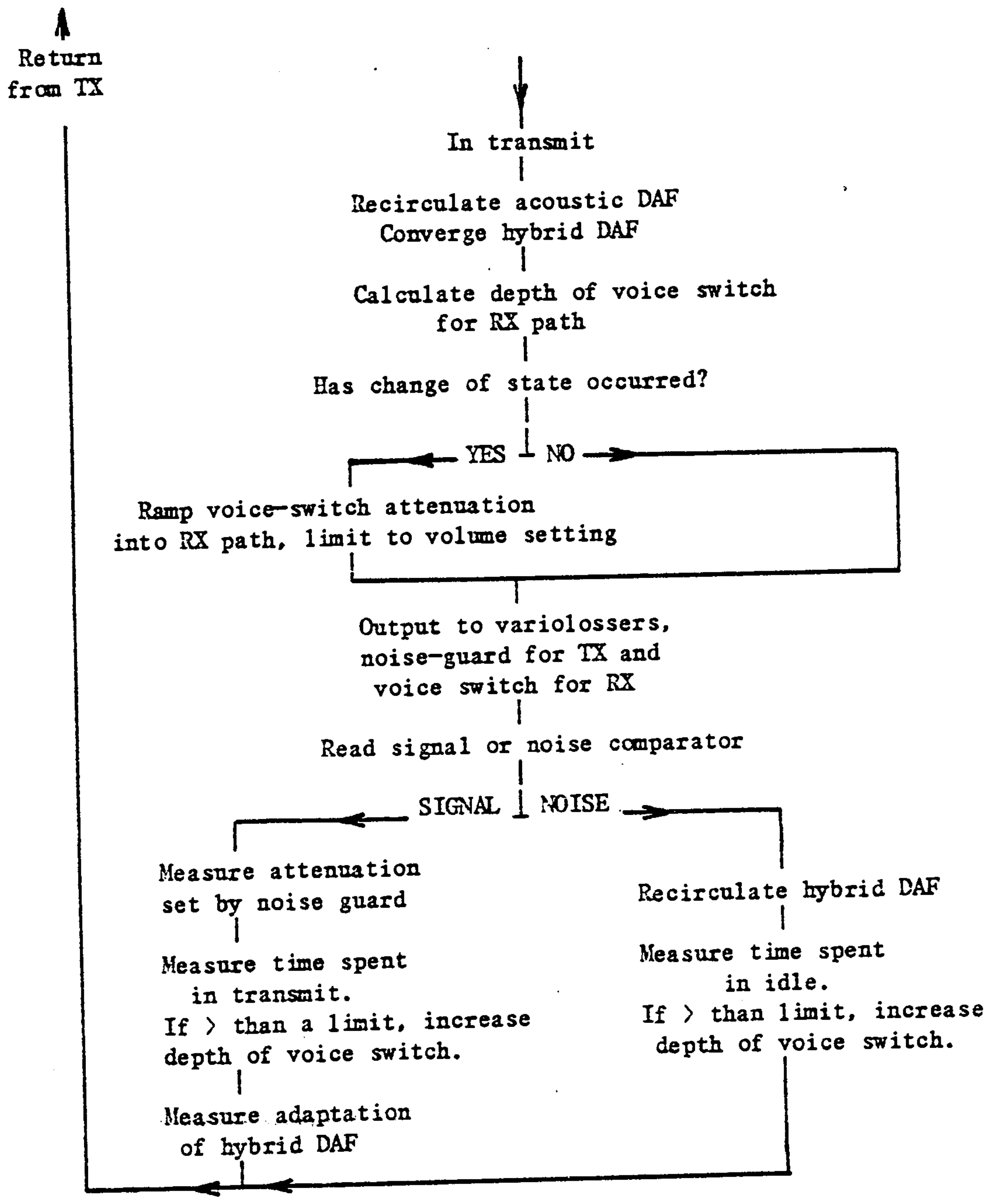
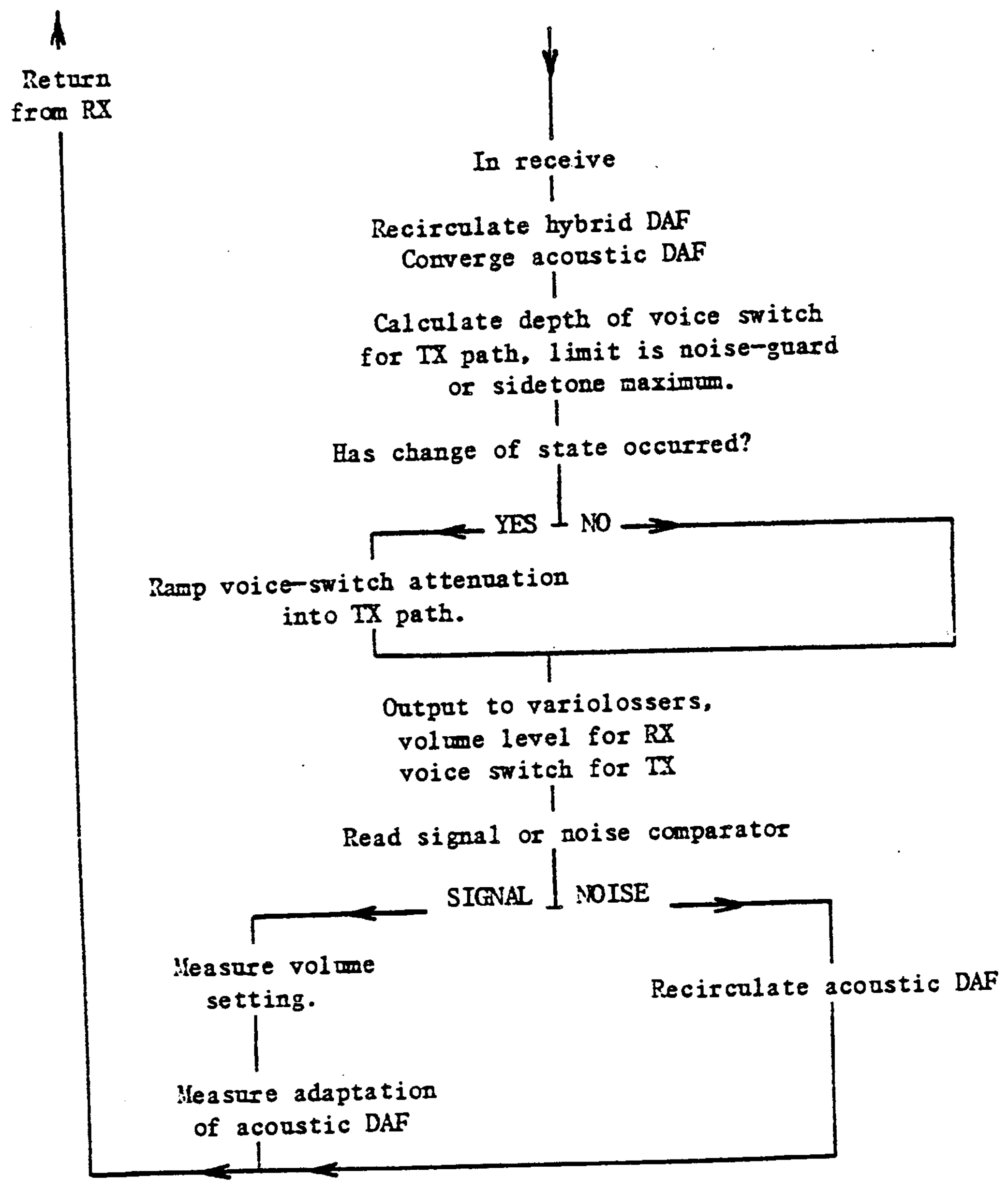


Figure 7.9 Flow diagram of the controller continued.


Return from RX
In receive
Recirculate hybrid DAF
Converge acoustic DAF
Calculate depth of voice switch for TX path, limit is noise-guard or sidetone maximum.
Has change of state occurred?
YES
NO
Ramp voice-switch attenuation into TX path.
Output to variolossers, volume level for RX voice switch for TX
Read signal or noise comparator
SIGNAL
NOISE
Measure volume setting.
Recirculate acoustic DAF
Measure adaptation of acoustic DAF

## 7.4 Performance.

The prototype system, which was built to demonstrate the feasibility of duplex transmission between an LST and a handset, was tested over a large number of calls with a range of line losses and room conditions. These tests were carried out by the team who built the system, occasionally using naive subjects to give opinions on aspects of the call. No formal subjective experiments were performed (except one described in Chapter 8 on objection and detection of variable depth voice-switching) for two main reasons. First, the equipment, with its great number of components and connections proved unreliable on a day-to-day basis. There was only one built and so any test programme undertaken could not be guaranteed to be completed. Secondly, there was only a full voice-switched system with which to compare it and, ideally, the performance should be measured against an unimpaired duplex system. A third reason was that although the system worked well under most controlled conditions, it was evident that the performance was limited by the architecture of the DAFs available [502] and the difficulty in making any changes to the analogue part of the controller. Commercial pressure deemed that, based on the performance of the prototype, all effort need be directed toward an instrument capable of being manufactured. To that end the work on DAFs in LSI and a new wholly-digital controller (using a DSP device), has been undertaken. Also, facilities are being developed to assess the subjective performance of the system.

A more flexible interpretation of the term 'duplex' emerged from using the system , based on the depth of voice switching:

| Depth of voice switching | Definition |
|---|---|
| 0 to 12 dB | Duplex |
| 12 to 20 dB | Partial Duplex |
| $>$ 20 dB | Simplex |

The measure is very crude and will obviously depend on room and system noise. It was found for the first case that, although the distant talker appeared quiet during double-talk, each word was easily understood, leading to a comfortable and interactive conversation. For the second case, speech from the distant talker appeared quiet during double-talk but difficulty was found in determining what was actually said. Never-the-less, conversations could be maintained with little effort from both parties.

The most objectionable degradation was the sidetone perceived by the distant handset user, appearing in a transitory manner as body movement in the acoustic path caused loss of adaptation. This only applies to low-loss calls, and the technique of leaving a minimum 6 dB attenuation in the transmit path while in a receive state proved partially successful. The resulting sidetone had an objectionable high frequency content. It was evident that high-frequency components were not being cancelled by the short impulse response of the DAF (32 ms) and that low frequency sidetone may not be as objectionable.

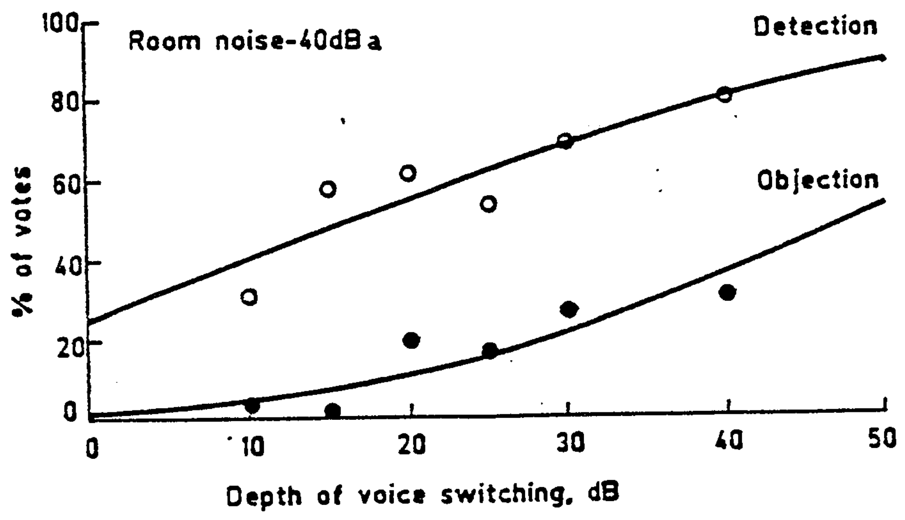
Two methods of dealing with this problem are being pursued. The first is to use the DAFs described in Chapters 3 to 5 in a cascaded mode. This will have the advantage removing more of the high-frequency signal plus the ability to adapt to a changing response with a minimum lag time. The second is to shape the gain-frequency characteristic of the voice-switch while in receive, to provide high-frequency cut and further make the sidetone less objectionable.

To obtain information on the benefit that might be achieved by reducing the depth of voice switching, an experiment was set up to assess subjectively the effect of varying the depth of voice switching from call to call (but not within a call, which is the case for the experimental LST) [703]. The experiment was symmetric with two LST users and loop stability was maintained by using a 4-wire circuit and acoustic separation of a loudspeaker and microphone. The loudspeaker was mounted near one ear and the output adjusted to be equivalent to that of a 750 mm air path. Voice switching was varied between 5 and 40 dB in two levels of room noise (simulating a quiet and a noisy office). The subjects were asked

a) if they could detect voice switching and

b) if they did, was it objectionable?

Detection/objection results are shown in figure 7.10 for the two levels of room noise with smooth curves fitted by probit analysis [704]



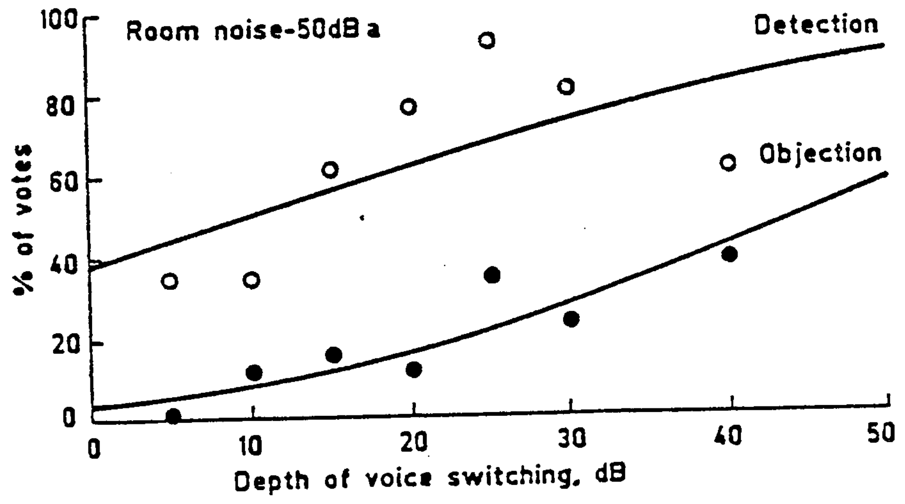


FIG. 7.10 DETECTION/OBJECTION CURVES AGAINST DEPTH OF VOICE SWITCHING FOR TWO LEVELS OF ROOM NOISE

Some points are noteworthy. There is a significant interaction between level of room noise and depth of voice switching. This might be expected, since the background noise rises and falls between transmit and receive states. The louder the noise, the more noticeable it becomes. Secondly, and surprisingly, the detection of voice switching saturates in the region of 50 dB of voice switching. A video recording was made of the whole experiment and, on examination this, it was noted that some people had a natural tendency not to interrupt until the other party had finished talking. It could also be seen that people adapted to the system; when deep voice switching prevented duplex conversation they structured their conversation accordingly. Finally, if the maximum (on average) depth of voice switching is limited to 20 dB, then approximately 40% of users find the system less objectionable.

## 7.5 Conclusions.

An architecture for a controller has been developed, which measures the adaptation of the DAFs and calculates the necessary additional attenuation needed to maintain stability. The theory developed in Chapter 6 and controller were applied to a hardware system, which demonstrated the feasibility of providing duplex transmission between an LST and handset, working in real time. Current research effort is being directed toward using the newly-developed LSI DAFs and an all-digital controller (based on the architecture described) to provide a flexible (and economic) system which can be subjectively assessed to determine the optimum performance.

# 8 FUTURE WORK.

New work generated by this project will continue along three lines: subjective assessment, LST-to-LST transmission and recursive adaptive filters. The latest LST, with all-digital processing and a cascade pair of DAFs for the acoustic path, is in an advanced stage of development. It will soon be available for subjective assessment of the new degradations it introduces during a conversation, i.e. variable sidetone and depth of voice switching. It will also be of interest to determine to what extent people move in the acoustic path when using a handsfree system.

In the application of LST-to-LST connections, two approaches are possible. First, the adaptive filters can be configured to accommodate the full impulse response of a room, but this is expensive. Second, information on parameters can be passed between LSTs, but this would require identical instruments at each end of the connection and in-band signalling. One area which could accommodate both techniques is broadband (7 kHz) conferencing. With the current generation of DAFs, to obtain an impulse response long enough to cover a room, the frequency response can be split into bands with separate DAFs applied to each frequency band. This would exploit the high-attenuation-high-frequency characteristic of typical rooms. An example is shown in figure 8.1 where the frequency band is divided in two, 0 - 1 kHz and 1 - 4 kHz. The sample rate for the lower band is 2 kHz ($\Delta t$ = 0.5 ms), therefore, a cascade of 4 filters (960 coefficients) would have an impulse response of 480 ms, sufficient to handle most rooms. A cascade of 4 filters for the upper band would have an impulse response of 120 ms. A project is in progress to apply this technique together with a controller distributed between both terminals.

Finally, an investigation of the next generation of adaptive filters using a recursive architecture is being undertaken. Initial interest will be centred on algorithms of the SER type (described in Chapter 2), but with processing to determine conditions necessary for stability at each update cycle.

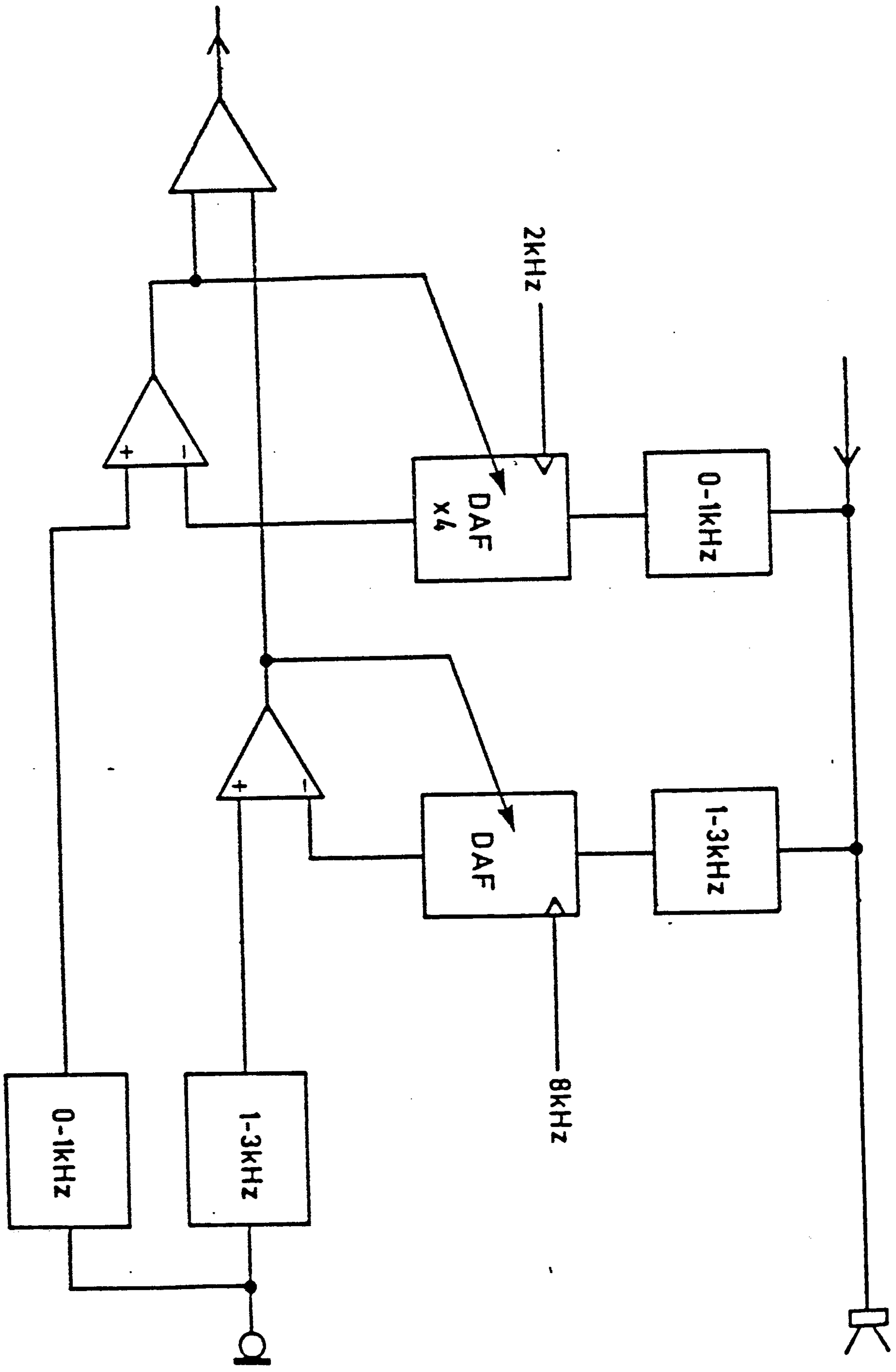


FIG. 8.1 BAND SPLITTING TO ACCOMMODATE AN ACOUSTIC IMPULSE-RESPONSE

## 9. CONCLUSIONS.

A successful and original loudspeaking telephone has been built and demonstrated, which can provide duplex transmission to a handset over the majority of connections in the BT network.

Currently, all of the LSTs available to the public use voice switching to maintain loop stability while providing sufficient acoustic power to make the instrument subjectively acceptable. A number of schemes have been proposed to provide duplex transmission, but all have failed in achieving both stability and adequate volume over the range of lossy connections in the network.

The experimental LST examined in this project utilized adaptive filters to cancel the coupling between loudspeaker and microphone in the acoustic path and between the transmit and receive ports of a hybrid. Stability is maintained by a controller which must perform a number of functions. To provide a degree of duplex transmission, it must measure the level of adaptation achieved by the filters and adjust the gain (or loss) in one of the paths, depending on its determination of the direction of transmission, at any instant in time. The stability of the system is based on obtaining an estimate of the deepest hole in the frequency spectrum of misadjustment between the physical path and the model produced in the adaptive filter. A theory has been developed which links a time-domain measurement of adaptation with that in the frequency domain. Thus, for signals which have a Gaussian amplitude distribution, a margin can be calculated which, when used with the time-domain measurements, will give a small probability of instability. In the future this work must be extended to cover speech signals. This is a more complex problem because of the coherent nature of the signal. The theory holds for the condition where the impulse response of the physical path is bounded within that of the adaptive filter. Unfortunately, this was not the case for the filters used in the acoustic path of the experimental system and stability was achieved by increasing the margin. A further consequence of using a filter with a short impulse response was the level of sidetone perceived by the handset user on low-loss connections. This required an asymmetric distribution of enhancement between the directions of transmission, such that the transmit path was restricted to have a minimum level of attenuation in the voice switch. The problem will also exist in a LST-to-LST call, although this has not been dealt with in the work reported here. It has been found that processing alone is insufficient to enable duplex transmission, and attention must be given to case design to achieve the maximum acoustic isolation between the loudspeaker and microphone.

From the calls made over the network it was found that the LST introduces two new degradations, as a result of random movement of the LST user in the acoustic path. Under this condition, there is a lag associated the adaptive filter following the change, with different results depending on which end of the connection the direction of speech is coming from. For a receive condition (at the LST), the handset user perceives a modulation of his sidetone level, which is only apparent on low-loss calls. In the transmit case, the degradation is dependent on the the method used to deal with the problem, since no signal is available to update the acoustic filter. The solution used in the experimental controller was to ramp attenuation into the receive path as a function of the length of time transmit was held. As well as introducing the usual initial syllable clipping on interrupt, this is perceived by the handset user on a low-loss connection to appear as a high level of sidetone at the start of his sentence, until the acoustic filter re-adapts. The perception of the degradation is judged to be similar to sidetone with a high-frequency content. The returned signal is complex, where, in the time domain, it is characterised by an initial period of noise, resulting from the misadjustment of the DAF (possibly producing the high frequencies), followed by the uncancelled reverberant tail of the room.

For the majority of calls, the experimental LST required much less effort than a deep (of the order of 40 dB) voice-switched system. The latter could be provided by operating a switch to compare the two types of stability control. Although the depth of voice switching is dependent on several factors and could vary throughout a call, the feeling of having a duplex conversation persisted even with an average of 18 dB attenuation in the voice switch.

The experimental LST, which was built to establish the feasibility of the system, combined both analogue and digital signal processing to carry out the functions of measurement and gain regulation. This work has led to a second-generation instrument in which the processing is totally digital, using a DSP device, and provides the prospect of an economic solution appearing in the near future.

The experimental LST demonstrated the need to have an adaptive filter with a number of characteristics not included in any of those currently available. On cost grounds, the filter also had to be implemented in LSI. The list of requirements included the need to alter functions of the filter under software control, to have large levels of adaptation, a variable rate of learning, a wide level of input-signal compensation (so that adaptation rate is independent of signal level), and a variable-length impulse response. Additionally, the architecture requires a technique to prevent the loss of filter parameters during double-talk. All these were included in a design, implemented in LSI, which partitioned the circuit into 3 separate blocks called 'convergence', 'multiply accumulate' and 'memory', and connected via a bus. While the bus has the disadvantage of producing a high pin count for the LSI devices, it allows a flexible architecture which can be exploited in other applications. Protection against corruption by double-talk was implemented by multiplexing a second FIR filter onto the system, with coefficient interchange between it and the adaptive filter. This technique allows the non-adapting filter to be used in the transmission path (the foreground). Coefficients for this filter are calculated in the adaptive filter (in the background) and transferred under software control when measurements indicate that they represent the physical path to some predetermined degree of accuracy.

A theory for the performance of the adaptive filter has been developed, which is unique in including all the noise contributions produced by bus truncation within its architecture. It is necessary to reduce the size of buses, particularly after multiplying two numbers together, to keep the size of parallel-processing components at a reasonable level. There are two multiplications associated with the update algorithm and the theory allows for a distributed normalisation scheme, where loss of adaptation at high input levels is traded for increased rate of adaptation over the usable input-signal range. The theory has also been extended to include performance using bit-shifting as an alternative to multiplication in the update algorithm. It was shown that the single advantage of using a multiplier was to obtain the maximum rate of adaptation (165 dB/s). Additionally the theory has been extended to include the performance when operated in a companded PCM environment.

The result of this work has been to demonstrate that the application of signal processing to handsfree telephony can provide the basis of a new generation of loudspeaking telephone, capable of approaching the natural conditions which exist in face-to-face conversations.

This page has been left intentionally blank

This page has been left intentionally blank

## Appendix A.

Derivation of an algorithm for adaptation of a recursive IIR filter.

S. A. White [301] proposed an adaptive recursive filter based on an autoregressive moving-average (ARMA S) structure. The response modelling connection is as shown in Figure (A.1)

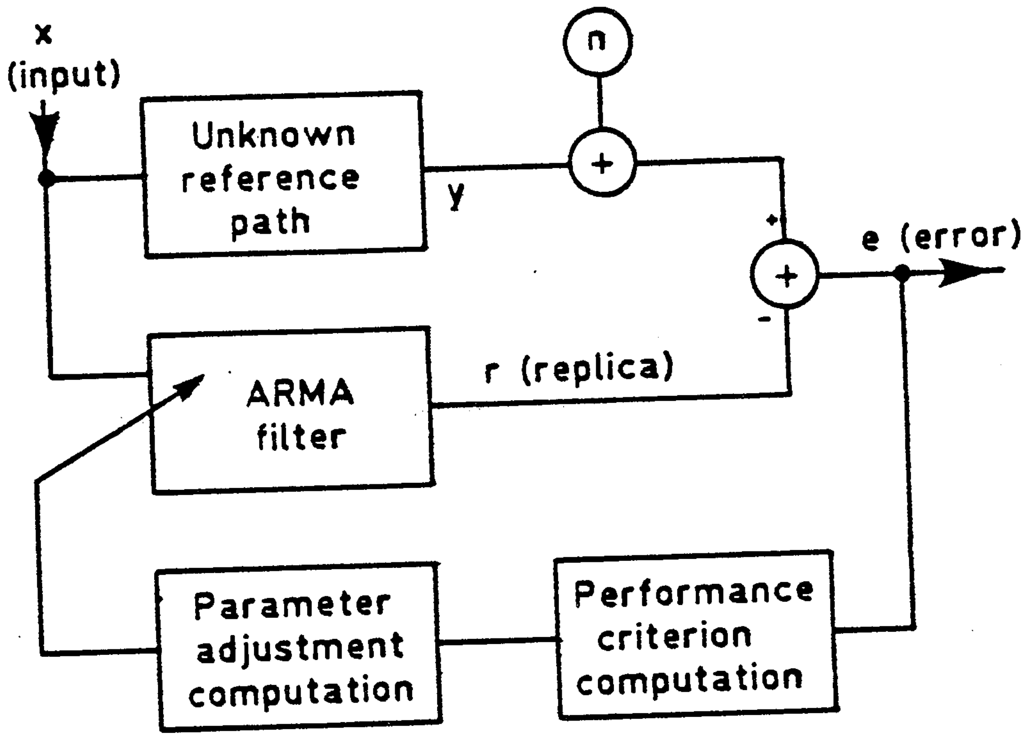


FIG. A.1 RESPONSE MODELLING SCHEMATIC

In the sampled-data domain the output of the filter 'r' (replica) is given by :

$$r_n = \sum_{k=0}^{NF} a_{k,n} \cdot x_{n-k} + \sum_{k=1}^{NB} b_{k,n} \cdot r_{n-k} \qquad \text{... (A.1)}$$

where $r_n$ = r(nT) the value of r at the nth sampling instant. All variables are normalised to lie in the range +1 to -1.

The transfer function is given by :

$$\frac{R(z)}{X(z)} = \frac{\sum_{k=0}^{NF} a_k \cdot z^{-k}}{1 - \sum_{k=1}^{NB} b_{k,n} \cdot z^{-k}} \qquad \text{... (A.2)}$$

The error is

$$e_n = y_n - r_n \qquad \text{... (A.3)}$$

The error criterion to be minimised is given by some function of the instantaneous error :

$$Fn = F(e_n) \qquad \text{... (A.4)}$$

Let $\rho_n$ be a (NF + NB + 1) vector of coefficients

$$\rho_n^T = [a_0, a_1, \ldots a_{NF}, b_1, b_2, \ldots b_{NB}] \qquad \text{... (A.5)}$$

Let $\beta$ be a (NF + NB + 1) vector of input/output values

$$\beta_n^T = [x_n, x_{n-1}, \;..\; x_{n-NF}, r_{n-1}, r_{n-2}, \;..\; r_{n-NB}] \qquad \text{... (A.6)}$$

Then the replica can be written

$$r_n = \rho^T \beta \qquad \text{... (A.7)}$$

Using the method of steepest descent the coefficient vector $\rho$ is updated each cycle, i.e. :

$$\rho_{n+1} = \rho_n + \mu . \nabla_\rho F(e_n) \qquad \text{... (A.8)}$$

where $\mu$ is a diagonal matrix which sets a limit on the value of the update component.

$$\mu = \begin{bmatrix} \mu_{a_0} & & & & & & & \\ & \mu_{a_1} & & & & 0 & & \\ & & . & & & & & \\ & & & \mu_{a_{NF}} & & & & \\ & & & & \mu_{b_1} & & & \\ & & & & & \mu_{b_2} & & \\ & & 0 & & & & . & \\ & & & & & & & \mu_{b_{NB}} \end{bmatrix} \qquad \text{... (A.9)}$$

Differentiating equation (A.4) w.r.t. $\rho$

$$\nabla_\rho F(e_n) = \nabla_e F(e_n) \; . \; \nabla_\rho (e_n) \qquad \text{... (A.10)}$$

The components of $\nabla_\rho(e_n)$ can be determined from equations (A.1) and (A.3). Consider the component $a_{j,n}$ in the vector $\rho_n$

$$\frac{\partial e_n}{\partial a_{j,n}} = \frac{\partial}{\partial a_{j,n}} (y_n - r_n) \qquad \text{... (A.11)}$$

$$= - \frac{\partial}{\partial a_{j,n}} \left[ \sum_{k=0}^{NF} a_{k,n} . x_{n-k} + \sum_{k=1}^{NB} b_{k,n} . r_{n-k} \right] \qquad \text{... (A.12)}$$

$$= -\frac{\partial}{\partial a_{j,n}} (a_{0,n}.x_n + a_{1,n}.x_{n-1} + \ldots\ a_{NF,n}.x_{n-NF}$$

$$+ b_{1,n}.r_{n-1} + b_{2,n}.r_{n-2} + \ldots + b_{NB,n}.r_{n-NB}) \quad \ldots (A.13)$$

Collecting terms

$$\frac{\partial e_n}{\partial a_{j,n}} = - x_{n-j} - \sum_{k=1}^{NB} b_{k,n}.\frac{\partial r_{n-k}}{\partial a_{j,n}} \quad \ldots (A.14)$$

Similarly

$$\frac{\partial e_n}{\partial b_{j,n}} = - r_{n-j} - \sum_{k=1}^{NB} b_{k,n}.\frac{\partial r_{n-k}}{\partial b_{j,n}} \quad \ldots (A.15)$$

These two expressions are recursive and can be solved by their recurrence relation

$$\frac{\partial e_n}{\partial a_{j,n}} = - x_{n-j} - \sum_{k=1}^{NB} b_{k,n}\frac{\partial r_{n-k}}{\partial a_j} \quad \ldots (A.16)$$

and

$$\frac{\partial e_n}{\partial b_{j,n}} = - r_{n-j} - \sum_{k=1}^{NB} b_{k,n}\frac{\partial r_{n-k}}{\partial b_j} \quad \ldots (A.17)$$

Rewriting equations (A.16) and (A.17)

$$- \nabla_\rho(e_n) = \beta_n + \sum_{k=1}^{NB} b_{k,n}\frac{\partial r_{n-k}}{\partial \rho} \quad \ldots (A.18)$$

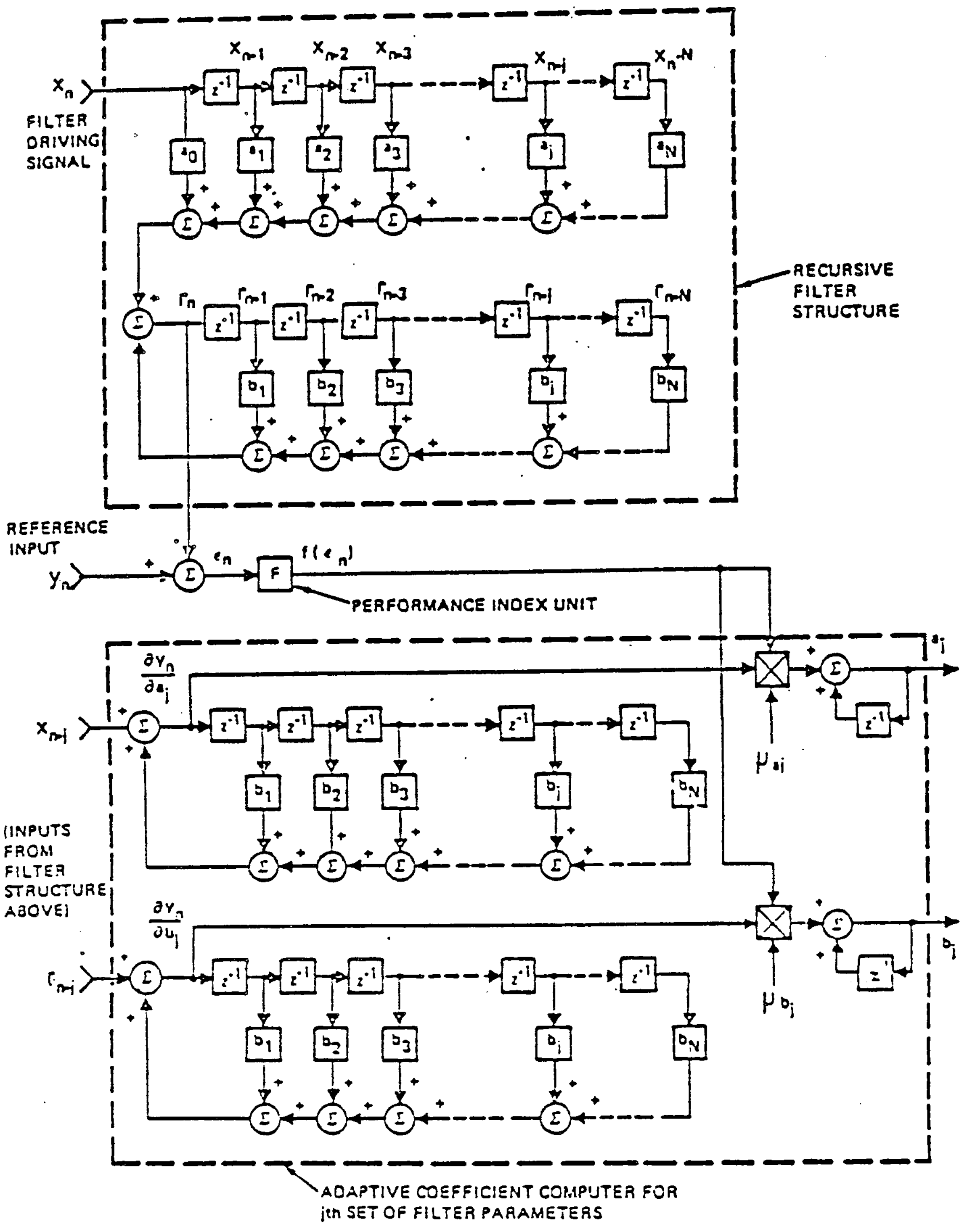


Figure A.2 Block Diagram of Adaptive Recursive Filter

## APPENDIX B - DETAILED CIRCUIT DESCRIPTION OF THE DAF.

### B.1 Master Clock Generator.

The master clock generator, shown on figure 4.1, was a separate circuit (actally built in TTL for the emulator and on a CMOS ULA for the LSI) with eight output waveforms derived from a crystal oscillator (16.384 MHz for 8kHz sampling), represented on figure B.2 as waveforms numbered 3 to 7. Two clock lines provided synchronisation at power on, to reset all counters, and at the audio-cycle rate, to define the burst-pause period. The remaining 6 clock lines provided 3 pairs of two-phase clocks. Additional clocks are required for serial-data transfer; these were generated on each LSI device. Listing the items on figure B.1, in order, from the first line:

1) Process-cycle number: the numbers 0 to 255 indicate the process-cycle number and act as a reference guide to link the various implementations.

2) Fundamental clock: this waveform is the master clock frequency derived by dividing the output of a crystal oscillator by two (to achieve a one-to-one mark/space ratio), which is then gated with a counter to produce a four phase alpha clock ($\alpha_{1-4}$), with one-to-eight mark/space ratio. Each alpha clock number is shown inside the respective mark positions on the waveform.

Master-clock output.

3) SYNCLR (synchronous clear): each of the partitioned circuits have internal clock-generators and counters derived from various master-clock waveforms. The SYNCLR provides a synchronous start for all independent clocks and counters, which is automatically reset at power-on and may be reset by an external control line, designated NOT(CLEAR) or NCLR. A more detailed description of this waveform is given in Appendix B.3, on the multiplier-accumulator, where it is used to count the correct number of audio cycles to initiate the SSX accumulator.

4) **Delta** ($\Delta$): defines a nominal burst-pause relationship with two hundred and forty process-cycles in the burst and sixteen process-cycles in the pause. It was intended that the ratio of burst to pause should be a variable, to exploit the noise improvement obtained from reducing 'W' in equation (2.105) for impulse responses bounded by WT. To achieve this, all internal clocks which control events in the pause should be counted from the rising edge of the delta clock and allowing only the termination of convolution to be fixed by the falling edge. Unfortunately this was not carried out in the LSI design of the multiplier and triple accumulator, but may be included in a subsequent rework (changes in LSI layout can prove very difficult and expensive).

Two-phase clocks:

5) **Gamma 41 and 23** ($\gamma_{41}$, $\gamma_{23}$): the suffixes indicate a relation to a phase of the alpha clock. This is a two-phase (non-overlapping) clock, having a mark-to-space ratio of 3/5, and frequency of 256 times the sample rate. It is only used by the LSI memory device to drive high-capacitance lines to the shift-registers, which were made up of D-type flip-flops.

6) **Beta 13 and 24** ($\beta_{13}$, $\beta_{24}$): a two-phase (non-overlapping) clock, with a frequency of 512 times the sample rate, and a mark-to-space ratio of 1/3. It is used to latch data into internal registers.

7) **Omega 41 and 23** ($\Omega_{41}$, $\Omega_{23}$): a two-phase (non-overlapping) clock, used to switch tri-state buffers onto the 'U' and 'V' buses. The frequency is 256 times the sample rate. The waveform is derived from the beta clock, but the falling edge is delayed by a half crystal-clock-period to extend the time available for bus settling, giving a mark-to-space ratio of 7/9.

Data-timing allocation.

8) 'U' and 'V' bus, data values: indicates the time data-values are

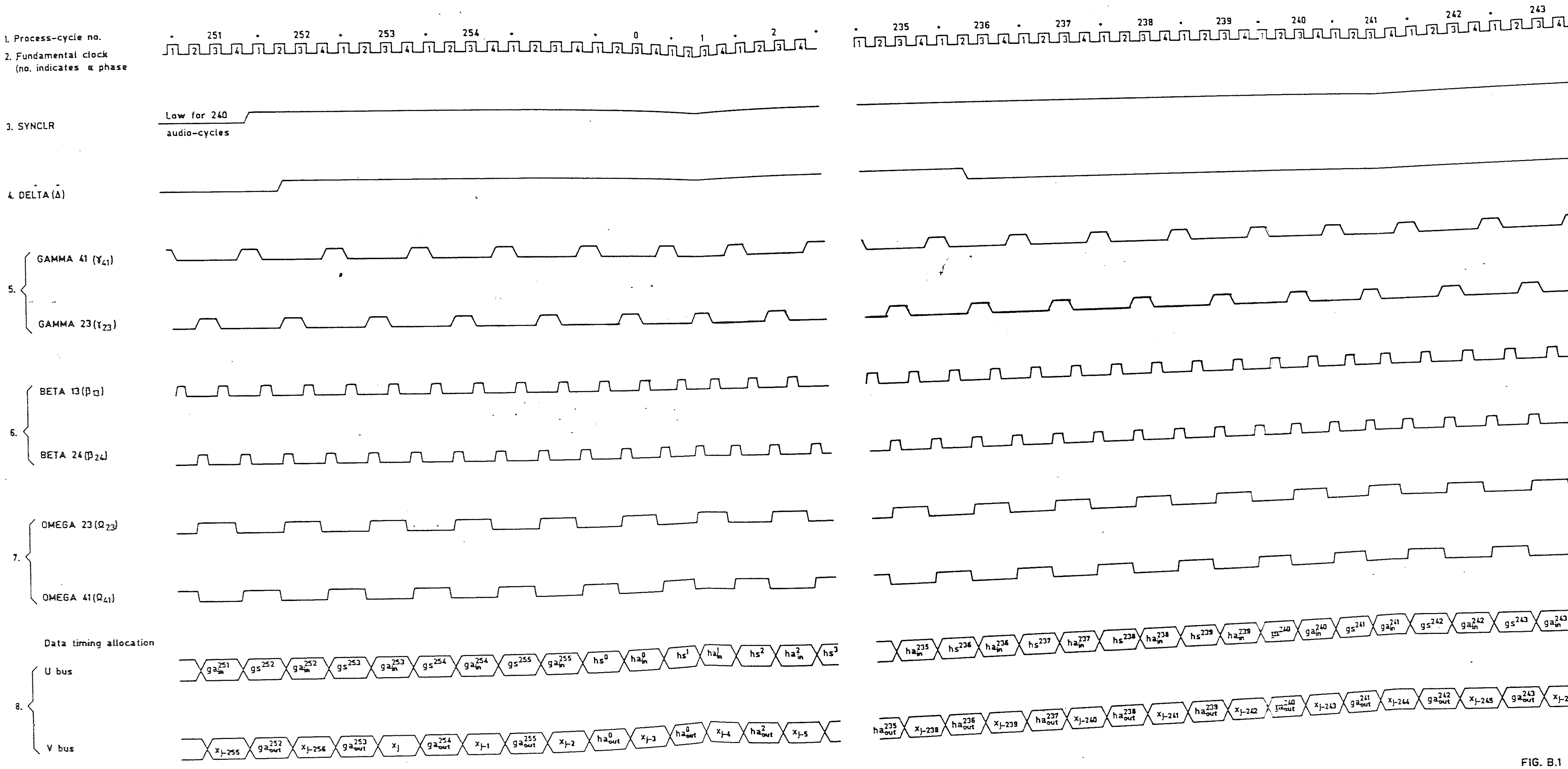

1. Process-cycle no.
2. Fundamental clock (no. indicates α phase
3. SYNCLR
Low for 240 audio-cycles
4. DELTA (Δ)
5. GAMMA 41 (γ41)
GAMMA 23 (γ23)
6. BETA 13 (β13)
BETA 24 (β24)
7. OMEGA 23 (Ω23)
OMEGA 41 (Ω41)
Data timing allocation
8. U bus
V bus
251
252
253
254
0
1
2
235
236
237
238
239
240
241
242
243

FIG. B.1

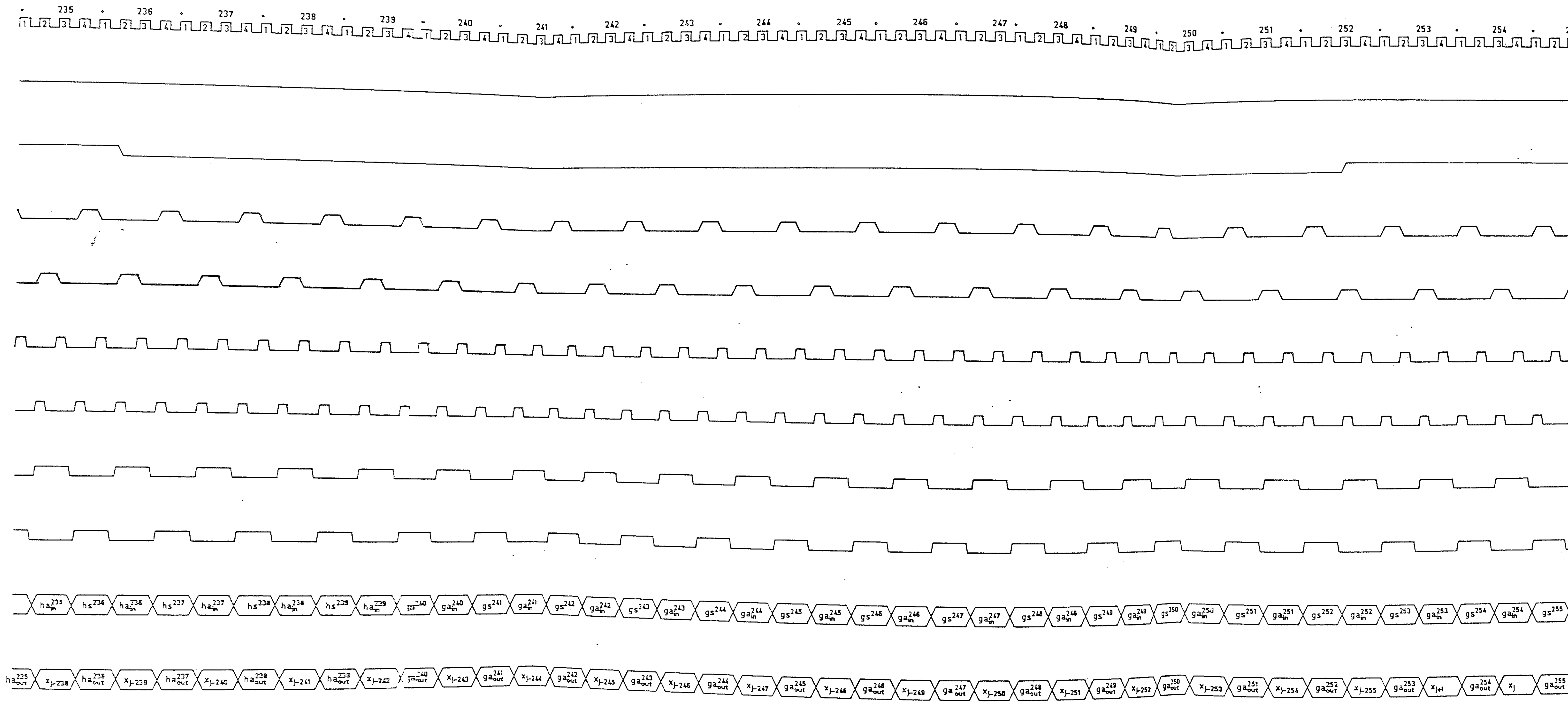


FIG. B.1 MASTER-CLOCK WAVEFORMS & DATA-BUS TIMING ALLOCATION

nominally allocated to their respective buses. Each period comprises a time for an electrical device to change state, plus a bus-settling time, before the data value is valid.

Nomenclature of data:

**for the 'U' bus:**

i) $ga_{in}^{252}$ (in process-cycle 252), refers to updated data in an unused, or garbage (g), location in the adapt (a) impulse-response storage device. i.e. since the number of data locations in the shift-register is 256, then, because of the pause, 16 will contain invalid numbers (see section 3.4.2). The superscript 252 refers to the coefficient index of a valid impulse-response.

ii) $gs^{253}$, refers to garbage in the stored (s) impulse-response.

iii) $hs^{0}$ (in process-cycle 0), refers to the first valid element in the stored impulse-response.

iv) $ha_{in}^{0}$ (in process-cycle 0), refers to the first valid element in the adapt impulse-response.

**for the 'V' bus:**

v) $x_j$ (in process-cycle 253), refers to the newest value of 'x' input (acquired in the previous audio-cycle).

vi) $ga_{out}^{253}$ (in process-cycle 253), refers to garbage out of the adapt impulse-response, before it is updated.

vii) $x_{j-1}$ (in process-cycle 254), refers to the previous value of 'x' input.

viii) $ha_{out}^{0}$ (in process-cycle 0), refers to the first valid element of the adapt impulse-response, before it is updated.

NB each phase of the alpha clock can be derived by ANDing appropriate beta and omega waveforms.

## B.2 The convergence Circuit.

### B.2.1 Local Strobe Generator.

The local strobe generator is shown on figure B.2 (left side) and associated waveforms on figure B.3. Five master-clock lines are terminated on the convergence device (SYNCLR, $\Delta$, $\beta_{13}$, $\Omega_{41}$ and $\Omega_{23}$) in the clock buffer and encoder (figure B.2, top left), where two alpha clocks are derived i.e.

$$\alpha_1 = \beta_{13}.\Omega_{41}$$

and

$$\alpha_3 = \beta_{13}.\Omega_{23}$$

Consider the $\Delta$ clock from the buffer driving the D input of an 8-bit shift-register SR-1, which is clocked alternately on $\Omega_{41}$ (first) and $\Omega_{23}$. The output, designated BURST, is therefore $\Delta$ delayed by 4 process-cycles, as shown in figure B.3. BURST is a mask defining the periods of the audio cycle when values in $H_{adapt}$ ($h^0$ to $h^{239}$) are valid (BURST high) and when garbage values ($g^{240}$ to $g^{255}$) are present (during BURST low).

The generation of two waveforms DSTBR and DSTBX, which are masks for the serial transfer of data from the multiplier-accumulator device, will be described. Both waveforms are complicated by the necessity of delaying each by a time dependent on the level of cascading (to facilitate external serial-addition). BURST is delayed 2 process-cycles in the shift-register SR-2, so that its falling edge coincides with the rising edge of $\alpha_2$ in process-cycle 242 (shown as DLY'D BURST in figure B.3). A further delay in flip-flop FF-1, clocked on $\Omega_{41}$, is introduced, and, by gating the input and inverted output of FF-1, a single low pulse, from the rising edge of $\alpha_2$ to the rising edge of $\alpha_4$, is derived (shown as waveform SigA in figure B.3). Waveform SigA provides NOT(SET) to flip-flop FF-2, whose output drives two inputs: one (called COUNTER NOT(R) in figure B.3) provides reset 'false' to a 5-bit counter (used to count $\alpha_1$ pulses, i.e. half process-cycles, to measure the time for each replica-transfer period); the other is ANDed with the output of flip-flop FF-3

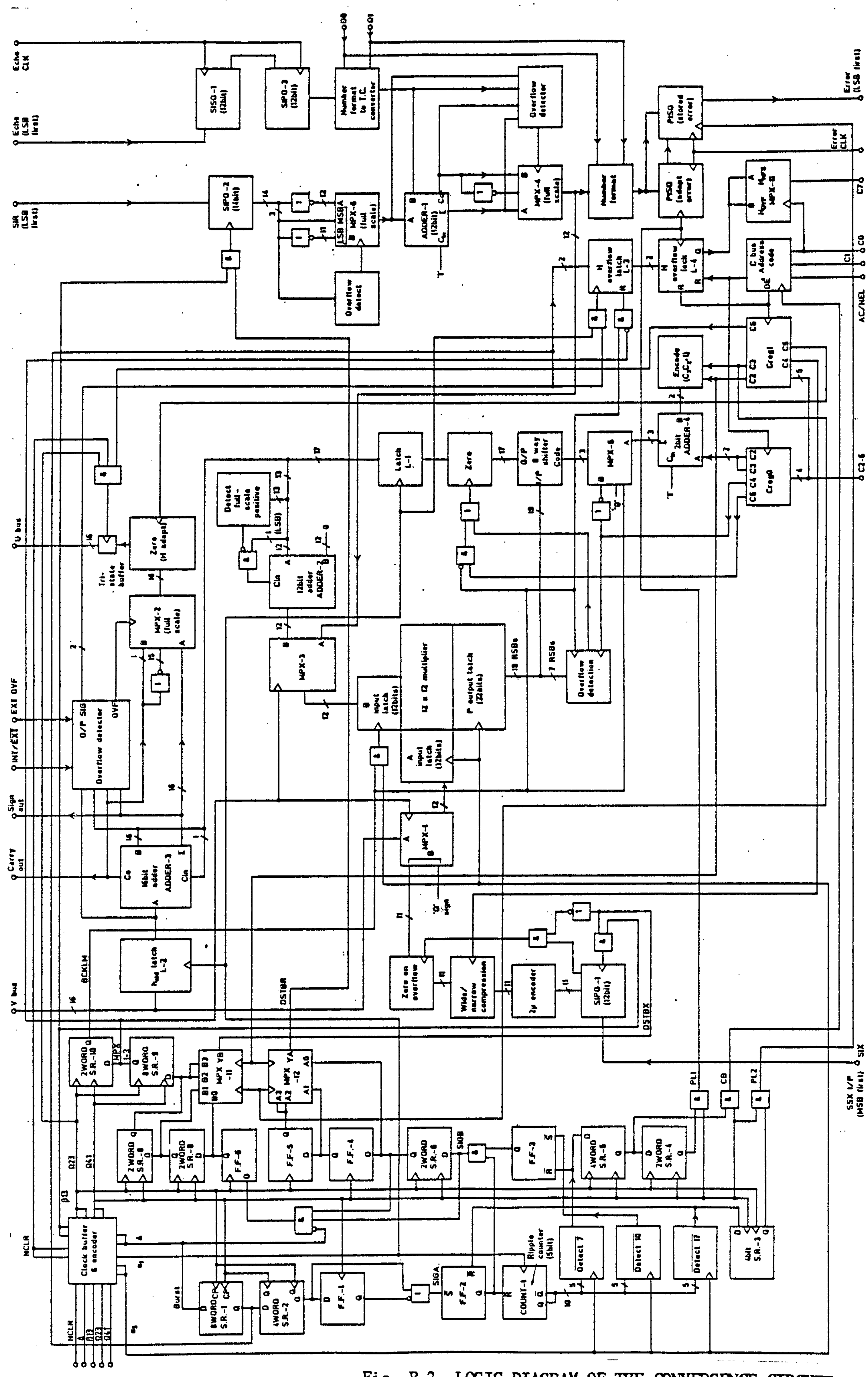


Fig. B.2 LOGIC DIAGRAM OF THE CONVERGENCE CIRCUIT.

to produce a waveform called RAW STB which goes high on the rising edge of $\alpha_2$ in process-cycle 242. Three values of the counter are detected and latched on $\alpha_3$. The first is decimal 7, i.e. counting 14 half process-cycles, equivalent to the number of bits in the replica transfer. NOT(DETECT 7) resets flip-flop FF-3, thus driving RAW STB low on $\alpha_3$ in process-cycle 249. The second, NOT(DETECT 10), then sets FF-3, driving RAW STB high on $\alpha_3$ in process-cycle 252, to begin the mask for the second replica transfer. The third count detected, NOT(DETECT 17), resets FF-1 and drives RAW STB low (14 half process-cycles later) on $\alpha_3$ in process-cycle 3; also reseting the counter to zero.

Three versions of DSTBX and DSTBR, corresponding to 3 cascade settings (single, dual and quad), are derived from RAW STB and connected to multiplexers MPX-11 and MPX-12 respectively. The choice of output is dependent on the cascade code (Cas1, Cas0) applied to the CBUS. RAW STB is delayed one process-cycle in shift-register SR-6 to produce the DSTBR waveform for a single filter (connected to input A0 of MPX-12). Flip-flops FF-4 and FF-5 provide the delays for the remaining two cascade conditions.

DSTBX comprises a pulse of 6 process-cycles duration, which is achieved by ANDing the input and output of SR-6, to remove 2 process-cycles from RAW STB, together with NOT(Δ) to remove the second RAW STB pulse. The resultant output is RAW DSTBX, which is delayed a half process-cycle in flip-flop FF-6 to provide DSTBX for a single filter (connected to input B0 of MPX-11). Two-bit shift-registers SR-7 and SR-8 produce the necessary delays, to DSTBX, for the remaining two cascade configurations.

The falling edges of two waveform masks, MPX1+3 and BCLKM, define periods within the pause for data latching on the correlation multiplier inputs. They are derived by simply delaying DSTBX the requisite number of process-cycles in shift-registers SR-9 and SR10.

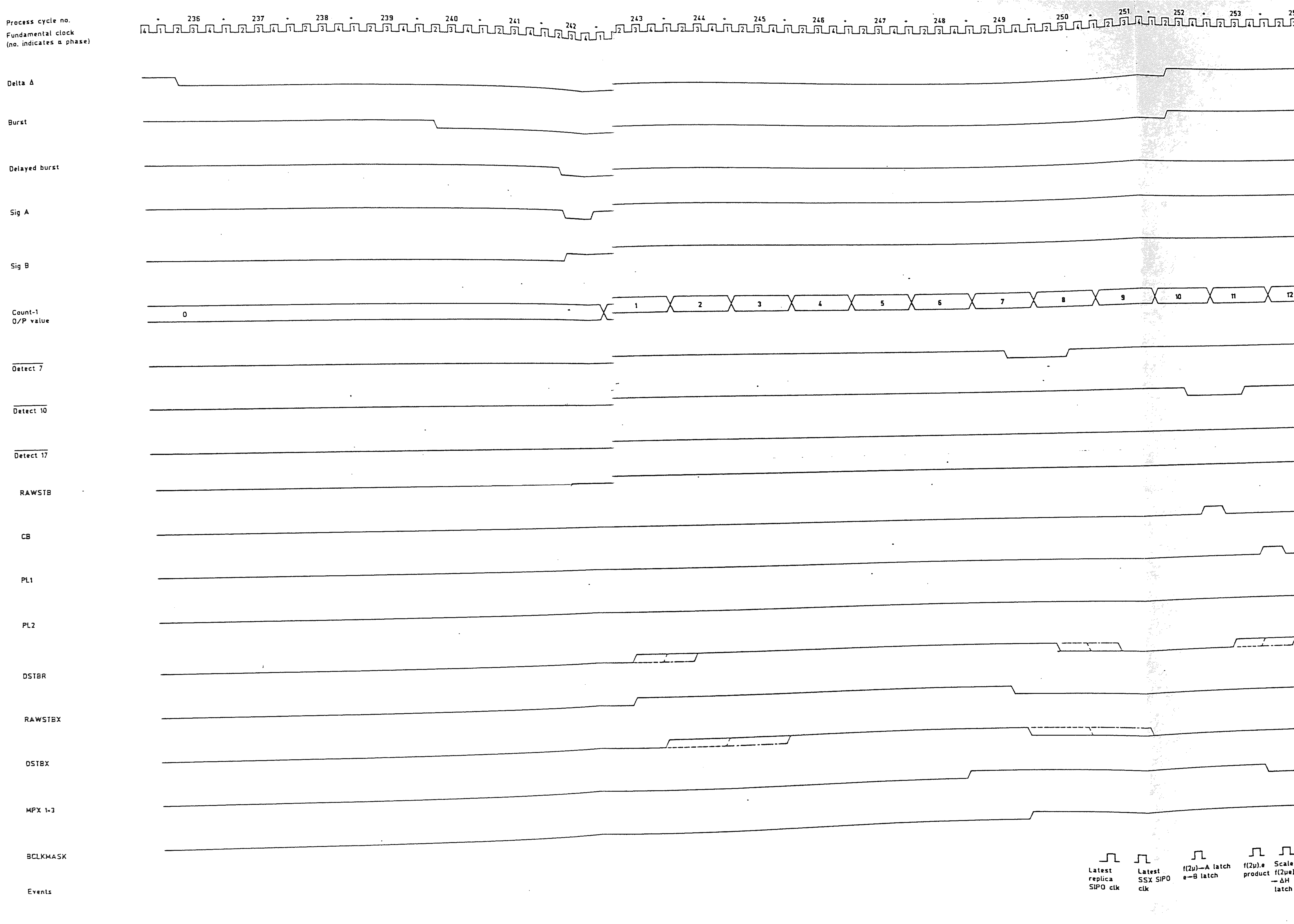


FIG. B.3 LOCAL STROBE WAVEFORMS FOR CONVERGENCE

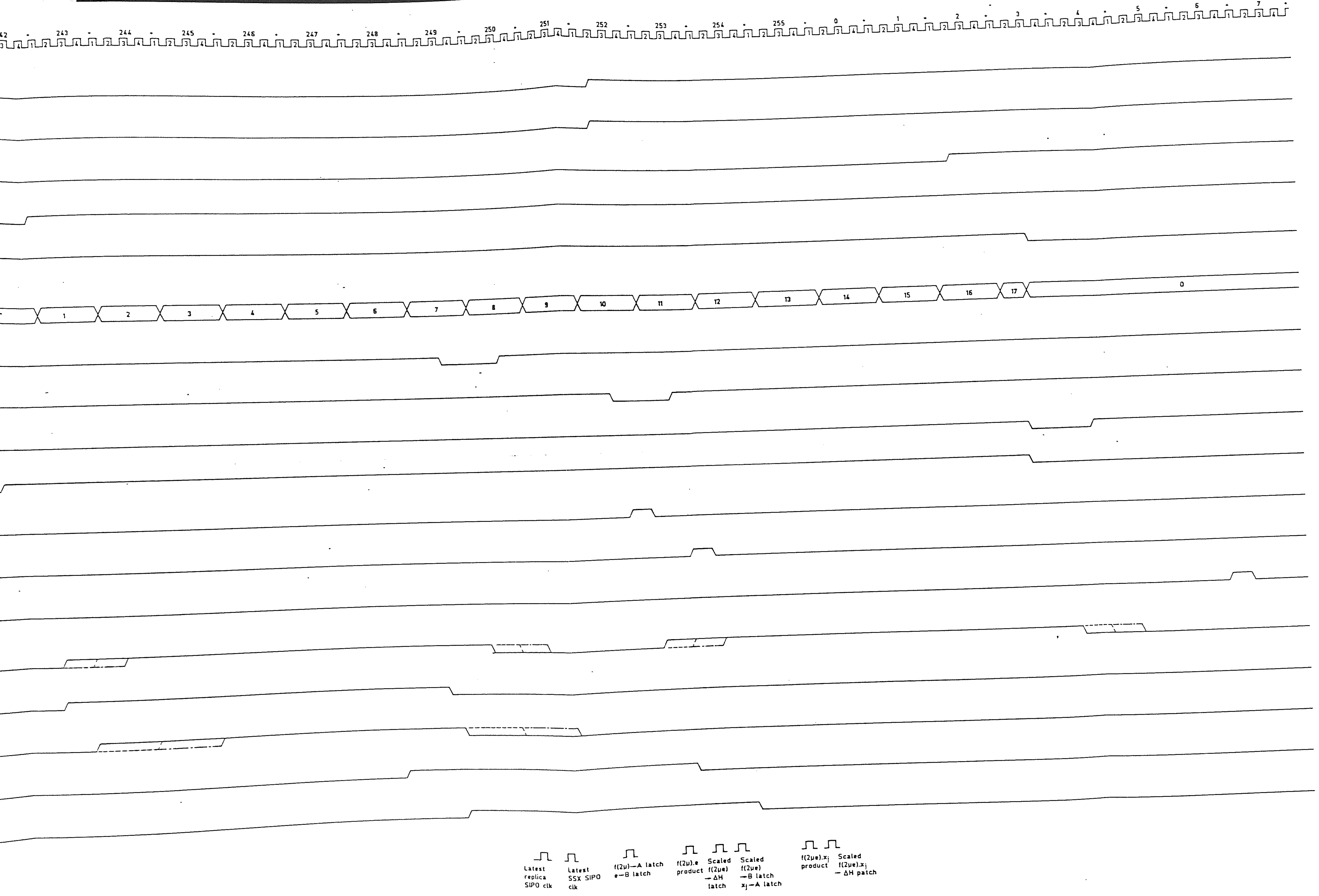


FIG. 8.3 LOCAL STROBE WAVEFORMS FOR CONVERGENCE

Three more strobes are required to latch data at specific times, CB latches data from the control bus (CBUS), PL1 and PL2 load adapt error and stored error respectively. Each is derived by delaying suitable waveforms (described above) which are ANDed with $\Omega_{41}$ to produce a single pulse. CB is obtained by delaying NOT(DETECT 10) for 4 process-cycles in shift-register SR-5 and PL1 is derived with a further 2 process-cycles delay by SR-4. PL2 uses NOT(DETECT 17) delayed 4 process-cycles in SR-3.

## B.2.2 Circuit description to calculate f(2μ).

Referring to figure 4.1, bits $2^{18}$ to $2^{29}$ (the choice of bit range is discussed in section 3.4.2) are output from the SSX accumulator (on the multiplier-accumulator chip from pin SOX) on DSTBX.$\beta_{24}$ (see figure B.3) and are latched into SIPO-1 (serial in parallel out) on DSTBX.$\beta_{13}$. In the TTL emulator, SIPO-1 was constructed from JK flip-flops (type 7473). Data output from the multiplier-accumulator changes on the rising edge of $\beta_{24}$; therefore, data must be stable by the rising edge of $\beta_{13}$ to be correctly latched in the SIPO on the falling edge of $\beta_{13}$.

Overflow is detected if bit $2^{29}$ in SSX is set high during NOT(DSTBX). NB if overflow occurs in any bits above $2^{29}$, it is detected on the multiplier-accumulator and bit $2^{2.90.2}$ is then forced high. ${}^{a}$f the alternative range of $\alpha\alpha\omega$ is selected (see ${}^{s}$able 3.2), by tying pin $\omega\gamma$ high, the overflow condition becomes valid for bit $2^{28}$. The consequence of overflow, at SIPO-1, is to inhibit the update of H during the following audio cycle (since a value for f(2μ) of less than 1 would be required for correct operation). This may be done a number of ways, e.g. to zero either f(2μ), x, error, or Δh for the whole of the cycle. The method used in the TTL emulator was to zero f(2μ) into the multiplier using a NOT(ENABLE) to set all outputs low, from multiplexer MPX1 (type 74158), which switches either f(2μ) or X to the A-input of the multiplier.

The f(2μ) used is given by equation (3.24), where C has been chosen to be $2^{29}$ (see equation (3.29)), for pin XG held at logical 0. Because SSX has 12 bits of data there are 12 possible codes for f(2μ), shown on the output of table 3.1. The code for zero input corresponds to full-scale-positive output. A simple method of generating the input-to-output relationship is shown in figure 3.3, using AND and OR gates.

The emulation employed a different technique. Using an intermediary stage to priority encode (TTL device 74148) SSX from SIPO-1 and decode the output to obtain the reciprocal (with TTL device 74154). The devices employ negative logic; consequently the NOT(Q) outputs were used on SIPO-1, the system reverting to positive logic with an inverting multiplexer, which supplied the full-scale number as its other input (switched on SSX = 0).

Wide-narrow. (W/N).

W/N is a facility to reduce the compensation range, effectively reducing the q-factor at low input-signal levels, which improves the adaptation performance for noisy reference paths.

The selection code is:

| | |
|---|---|
| wide | - logic 0 |
| narrow | - logic 1 |

W/N is selected externally and applied via the control bus; seting the maximum value of f(2μ) to $2^{8}$. Figure B.4 shows a simple logic diagram, applied to the upper three-data-bits of SSX, which performs this function.

Multiplexer MPX1.

The output of MPX1 is connected to the A-input of the multiplier. Switching, on strobe MPX1+3 (see figure B.3), between the 12-bit f(2μ) word (generated from SSX during the pause), and the most-significant 12 bits of the V bus (to obtain x during the burst).

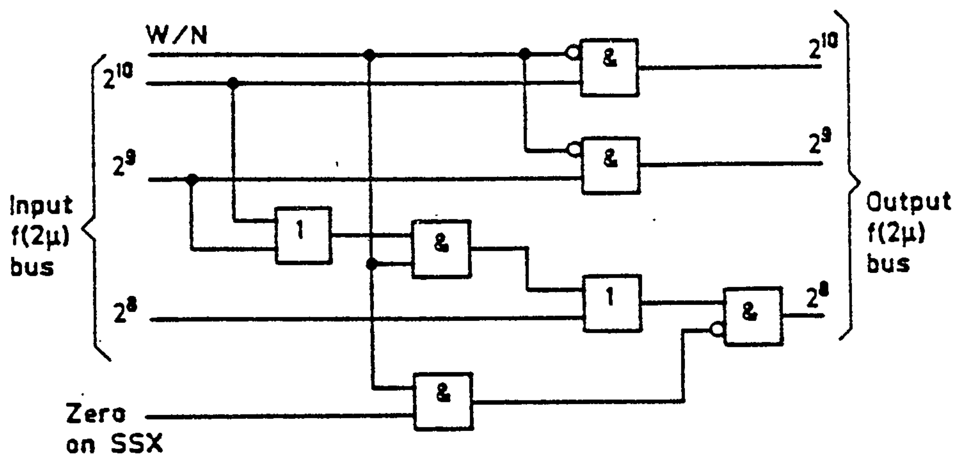


FIG. B.4 WIDE/NARROW OPTION OF f(2μ) COMPENSATION

B.2.3 Serial-data input and error output.

The serial-output replicas (SOR) are clocked off of the multiplier-accumulator during the two data-valid pulses in DSTBR (see figure B.1) on $\beta_{24}$. They enter the convergence device on pin SIR (serial-input replicas) and are latched into SIPO-2 which is 14-bits long (see figure B.2, upper right), on the falling edge of $\beta_{13}$. There are 12 data-bits plus 2 sign-extension bits, the latter are for external serial-addition in the cascade mode. DSTBR is delayed by one $\beta$ clock per level of cascade (a maximum of two for this filter), to allow time for the serial addition. Overflow detection and correction are applied to the output of SIPO-2.

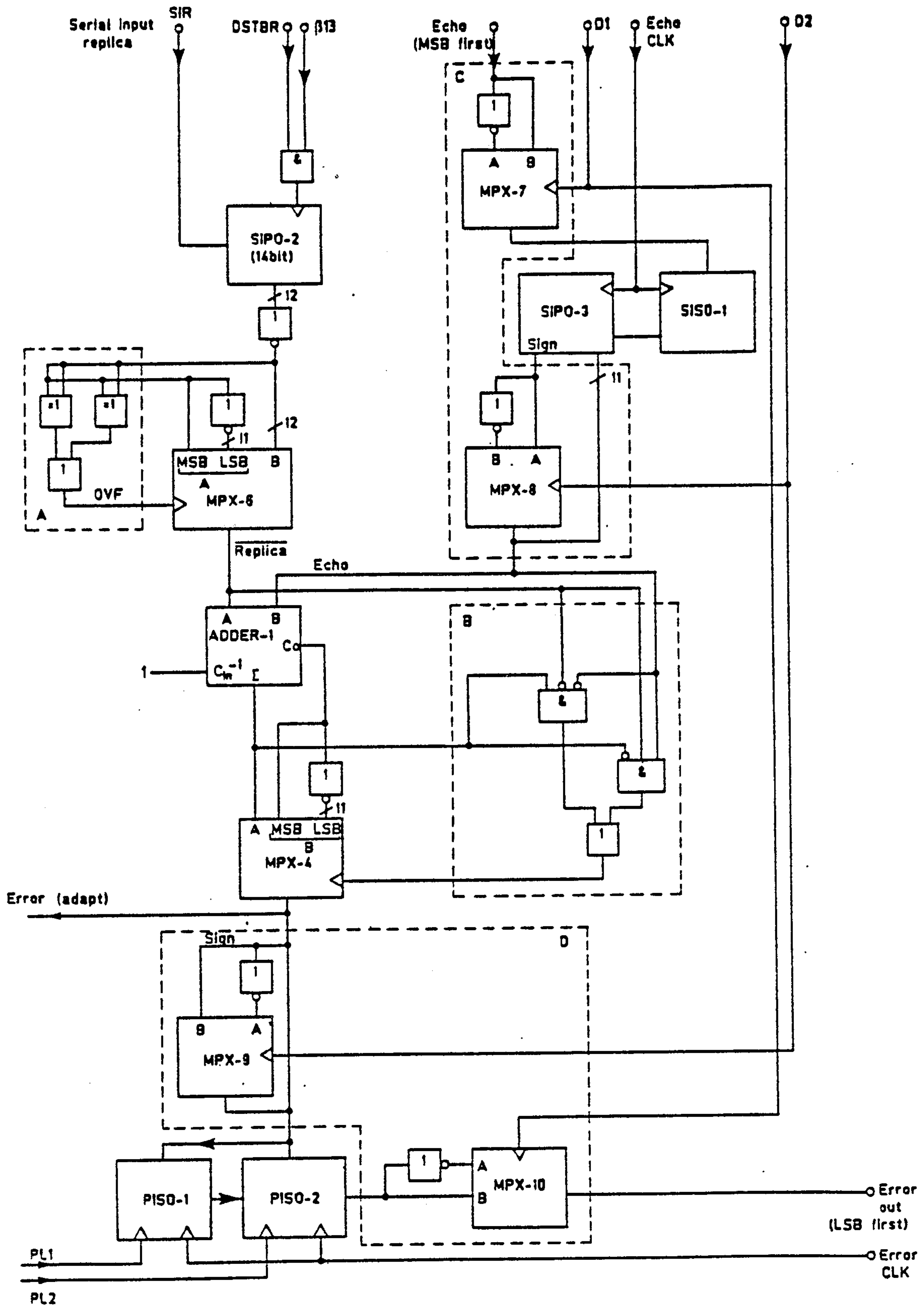
The adapt-replica ($r_a$) is valid from the falling edge of $\alpha_1$ in process-cycle 251. This sets the time limit for the desired signal to have been latched into its SIPO, from an external source, which could be an a-to-d converter or a digital store. Many a-to-d converters produce complementary-offset-binary (COB) numbers, which may be easily changed to two's complement, the number scheme in the filter. Additionally, external interfacing to alternative numbering schemes was provided by coding two pins, $D_1$ and $D_2$. The 12-bit echo is clocked into SISO-1 (serial in, serial out, i.e. a 12 bit shift-register) during the period it was sampled, and into SIPO-3 (12 bits), for use by the filter in the following audio cycle; this provides one cycle delay to compensate for the time required to perform the convolution.

To subtract the replica from the echo requires a sign change of the former and both values input to an adder (ADDER-1). This is easily done in two's complement by inverting the replica and adding 1 to the carry-in of the adder. The 'error' produced from the subtraction, is then overflow checked, corrected, and applied to the B-input of the correlation multiplier, via multiplexer MPX3, by the rising edge of $\alpha_1$ in process cycle 252. It is also restored to the external number-format and loaded into PISO-1 on strobe PL1, which has its rising edge in process cycle 252, ready to be output from the filter.

The stored replica follows in the second DSTBR pulse and is subtracted from the echo, then loaded into PISO-2. The 24 bits comprising error-adapt and error-stored are clocked off the convergence device with an external strobe after process cycle 6 in the following audio cycle (LSB first).

Figure B.5 is a detailed expansion of part of figure B.2 (right-hand side) and shows the formation of both errors. Following the signal path from the SIR input-pin on figure B.5 (top left), the replicas are clocked into SIPO-2 on DSTBR.$\beta_{11}$, LSB first. The TTL emulation used JK flip-flops (TTL device 7473) which transfer data on the falling edge of the clock. The NOT(Q) outputs were used to provide the inversion required to form the negative of the number.

A replica overflow (OVF) is detected by looking for a difference in the 3 MSB's with exclusive-or gates (as shown in the enclosed section 'A' of figure B.5). If an overflow condition occurs multiplexer MPX-6 provides full-scale output, i.e. the MSB (bit 14) to the sign and NOT(MSB) to the remaining 11 bits. It may be noted that a error is introduced when the replica is full-scale negative, i.e. -2048, as the negative of this number (+2048) does not



A Replica overflow detector

B Adder overflow detector

C Input to 2s complement conversion

D 2s complement to output conversion

Fig. B.5 INPUT OUTPUT AND SUBTRACTOR.

exist in the range. There is no need to trap this error prior to its input to the adder, as the overflow detection following the adder (ADDER-1) will determine any false condition.

Referring to the echo input on figure B.5, the 12-bit value must be entered serially into SISO-1 (a 12-bit shift-register ) prior to the rising edge of $\alpha_1$ in process cycle 251, using an external clock-generator. Two pins $D_1$ and $D_2$ are used to reformat the input data from one of four types into two's complement, with the code shown in Table B.1

The echo in the next audio-cycle will clock the preceding value into SIPO-3, giving one-audio-cycle delay required for convolution. Referring to the enclosed section 'C' in figure B.5, pin $D_1$ controls complementing at the serial input. The output of SIPO-3 is formated by putting the sign bit under the control of pin $D_2$, to determine offset binary or two's complement. The result is a delayed two's complement value, to the B-input of ADDER-1

| $D_1$ | $D_2$ | external number-format |
|---|---|---|
| 0 | 0 | two's complement (TC) |
| 0 | 1 | offset binary (OB) |
| 1 | 0 | complementary two's complement (CTC) |
| 1 | 1 | complementary offset binary (COB) |

Table B.1 Codes for external number-formats to two's complement conversion.

NB the value of stored-error must be computed before the stobe PL2. Obviously neither echo or stored-replica must alter during the formation of error. Thus, a time window is defined where new values of echo may be entered, i.e. between $\alpha_3$ in process-cycle 6 and $\alpha_1$ in process-cycle 251.

The carry input is set to 1 on ADDER-1, completing the negation of the two's complement replica.

**Overflow detection on adder output.**

The detection of overflow using exclusive-or gates as shown in figure B.6 would produce 4 gate delays. For most overflow detection requirements, time is not a critical factor. The exception occurs for the h+Δh adder (described later), where a minimum delay solution is required. It is possible to minimise the propagation time to three gate-delays using don't care conditions, as derived in Table B.2

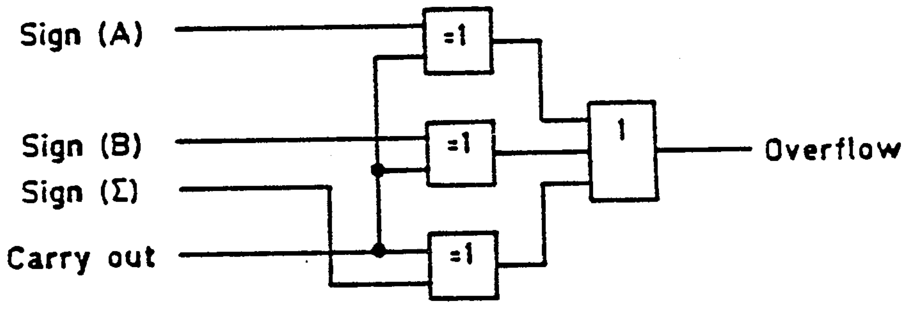


FIG. B.6 ADDER OVERFLOW-DETECTION USING EXCLUSIVE OR GATES

| input conditions | | | | output |
|---|---|---|---|---|
| sign (A)<br>A | sign (B)<br>B | sign(sum)<br>C | carry out<br>D | overflow |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | ? |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | ? |
| 0 | 1 | 0 | 0 | ? |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | ? |
| 1 | 0 | 0 | 0 | ? |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | ? |
| 1 | 1 | 0 | 0 | ? |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | ? |
| 1 | 1 | 1 | 1 | 0 |

Table B.2 Truth table for detecting overflow on an adder.
(? indicate a don't care condition)

Constructing a Karnaugh map for table B.2

| CD \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | ? | ? | ? |
| 01 | ? | 0 | 1 | 0 |
| 11 | ? | ? | 0 | ? |
| 10 | 1 | 0 | ? | 0 |

Then an overflow is given by:

$$OVF = sign(A).sign(B).\overline{sign(sum)} + \overline{sign(A)}.\overline{sign(B)}.sign(sum) \qquad \ldots (B.1)$$

The logic elements for equation (B.1) are shown in enclosed window B of figure B.5.

Error output.

The output from ADDER-1 produces two errors each audio cycle. Adapt-error is produced first and routed to multiplexer MPX-3 to form part of the update term. It is also latched into PISO-1 on strobe PL1, ready to output from the filter. Stored-error is produced subsequently and latched into PISO-2 on strobe PL2. The two PISOs are cascaded to enable an external clock (ErrorCLK) to output stored-error first (LSB first). Maximum time is therefore provided for external processing of the foreground-filter output. NB the two's complement error number is formatted to the form used for the echo.

## B.2.4 Correlation multiplier and scaler.

The correlation multiplier forms two separate products in each audio-cycle. One is the scaler product:

$$f(2\mu e) = f(2\mu).\text{error} \quad \dots (B.2)$$

and the other is the vector product:

$$\Delta \mathbf{H} = f(2\mu e).\mathbf{X} \quad \dots (B.3)$$

(calculated serially) as required by equation (2.91). Section 3.4.2 outlines the argument for computing equation (4.2) during the pause. The correlation multiplier runs continuously, consequently a number of garbage products are calculated during the pause while serial-data transfers, computation of f(2μ) and error etc. take place. f(2μ) and error are the last multiplier and multiplicand to be presented, in the pause, respectively to the A and B inputs of the correlation multiplier (see figure B.2, process-cycle 252). From the output P-latch of the multiplier the product f(2μe) is overflow detected, scaled and truncated to 12 bits, as shown in figure 3.6 (for $k = 1$ and $C = 2^{29}$). Scaling f(2μe) by the factor $k = 2$ (see equation (3.34)) can be implemented by examining the appropriate control line (C4) and setting a suitable code for a barrel-shifter (via multiplexer MPX-5) to select the correct bit-range. Having truncated a two's complement number, the value must be rounded, as described in section 3.8 (using ADDER-3). The resultant value of f(2μe) is latched back into the B-input of the multiplier (via the B-input of multiplexer MPX-3) on the last pulse in $\alpha_3$.BCLKM. At the same instant, $x_j$ (the newest value of $x$) is latched into the multiplier's A-input (from the V bus via multiplexer MPX-1).

The resultant product is $\Delta h^0$, which is stored in the P-latch of the multiplier on $\alpha_3$ in process-cycle 255. The value is examined for overflow, scaled and truncated as shown in figure 3.8 (an additional bit, $2^{-5}$, is included in the output for rounding). $\Delta h^0$ is then stored in latch L-1 on $\alpha_1$ in process-cycle 0, ready to be added to $h_0$. If an overflow on either f(2μe) or $\Delta h$ has occured, that value is zeroed. The process is continued through the burst period to complete the solution of equation (B.3).

Scaling $\Delta H$ for cascade and q-factor is acomplished by storing and adding the values (externally set) on the appropriate control lines (C2 and C3) to obtain a value for $2^{(Q+c)}$ in equation (3.34), which is then applied to the barrel-shifter during the burst period.

The Multiplier.

An architecture for a ripple-through, 12-by-12 bit multiplier, with 22-bit output, and capable of operating at the desired speed (i.e. at the 8 kHz rate having 480 ns to form a product) had been carried out by an LSI design group in BTRL. Consequently this multiplier was chosen for the convergence circuit. Referring to figure B.2, the A-input and P-output latches are clocked on $\alpha_3$. The B-input latch is clocked on $\alpha_3$.BCLKM. Data to the A-input is via the two-input multiplxer MPX-1 which switches either f(2μ) or x information (from the V bus), on the strobe MPX1+3. Input B derives data from multiplexer MPX-3 (switched on strobe MPX1+3) and receives either the error or f(2μe).

Overflow detection.

The product output from the multiplier forms a convenient position to detect overflow for both f(2μe) and ΔH, but each has slightly different requirements. Overflow on f(2μe) alters with the choice of k, while for ΔH it is dependent on the q-factor and the cascade setting. Any overflow at this point can only have been produced by an erroneous computation, e.g. noise on the desired signal or data corruption from α-particle generation. The action required is to zero the value, which is conveniently done after the barrel-shifter.

From figure 3.6 it can be seen that a valid overflow in f(2μe) occurs if any bits in the range from the MSB to $2^{17}$, for k = 1, differ from the sign bit, i.e. bits of greater order those of f(2μe). For k = 2 the overflow-bit-range becomes the MSB to $2^{16}$.

To determine the overflow condition for ΔH, substitute equation (3.24) into (3.34) and consider only the update component:

$$\Delta H = \frac{C}{||X||^2\, t_1.t_2.2^{(Q+c)}}\, e.X \qquad \ldots \text{(B.4)}$$

where the vector norm is scaled externally by $2^{-c}$.

NB the component $k = 2^K$ is ignored since it is treated in the pause.

It is necessary to determine the probable worst case conditions for e and X, which can be assumed to occur at initialisation (when $H^* = 0$), then, for a lossless reference-path:

$$E(\bar{e}^2) = E(\bar{x}^2) \qquad \ldots \text{(B.5)}$$

If an input signal limits at 4σ peaks, then, using equation (B.5), the worst-case product in the vector e.X is:

$$(e.x)_{(max)} = 16E(\bar{x}^2) \qquad \ldots \text{(B.6)}$$

Substituting equation (B.6) into equation (B.4), and let the number of coeficients in a single filter be approximately $2^8$, the largest value in the vetor $\Delta H$ is given by:

$$\Delta h_{(max)} = \frac{C\, 2^{-4}}{t_1 . t_2 . 2^{(Q+c)}} \qquad \text{.... (B.7)}$$

The system coefficients for a single filter are given by equation (3.29), where $C/t_1/t_2$ = g.k, therefore equation (B.7) becomes:

$$\Delta h_{(max)} = 2^{(7-Q-c)} \qquad \text{... (B.8)}$$

for k = 1.

Assume a filter with Q = c = 0, then $\Delta h_{(max)} = 2^7$. Referring to figure 3.8, overflow will occur if any of the bits of order greater than $2^6$ (i.e. $2^7$ approximates to $2^7-1$), in $\Delta h$ (shown as the row labled $t_2 = 2^{13}$), differ from the sign on the output of the multiplier (shown as the row labled $t_2 = 2^0$). NB for this particular multiplier the corresponding bits are $2^{21}$, $2^{20}$ and $2^{19}$.

The arguments developed above for overflow detection on the two outputs f(2μe) and $\Delta H$ are for worst cases, which are unlikely to occur in normal operation except for low-level input (where f(2μ) is a maximum) and a noisy echo. Some noise advantage can be gained by extending the overflow detection toward better-case conditions, i.e. ignoring large instantaneous values of the echo signal y, for low-level input x, at initialisation. This would reduce some contribution of noise on the echo. Consequently an additional bit was included in the overflow detection on the $\Delta H$ output.

By choosing the output of the multiplier to examine $\Delta H$ for overflow, the minimum circuit complexity is achieved, since at this point the range of $\Delta H$ is invariant.

The condition for overflow on the f(2μe) bus was based on using the system constant $C = 2^{29}$ (equation (B.8)), but it was stated in Section 3.5

that an option was provided for C = $2^{28}$ (to alter the input-signal compensation range), by tying a pin XG at logic 1. Reducing the value of C is easily implemented by shifting the selected bits in figure 3.6 one place to the right, i.e. $t_1 = 2^6$. A minor difficulty is added to the overflow of the f(2μe) bus, since selection of the correct bits is dependent on XG. f(2μe) is valid during BCLKM and must account for changes in XG and k; Table B.3 shows that an overflow condition occurs when any of the bits shown is different from the sign.

| | | overflow for bits >= | | |
|---|---|---|---|---|
| XG | k | 17 | 16 | 15 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Table B.3 Overflow for f(2μe) at multiplier output.

Figure B.7 shows the logic diagram for overflow detection on the output of the correlation multiplier for both f(2μe) and ΔH outputs.

The barrel-shifter.

Referring to figure B.2, the 19 MSBs from the output of the multiplier are input to an 8-way barrel-shifter, i.e. 2 shift positions for switching between K = 0 or 1 and 6 positions for -(Q+c). The output state of the shifter is decoded from a 3-bit binary-word. Table B.4 shows the bit positions from the multiplier bus, numbered 0 to 22 (bit 22 is the sign), as seen from the output of the shifter, where the bits are numbered from the MSB, i.e. 11 (sign) to -5 (the LSB).

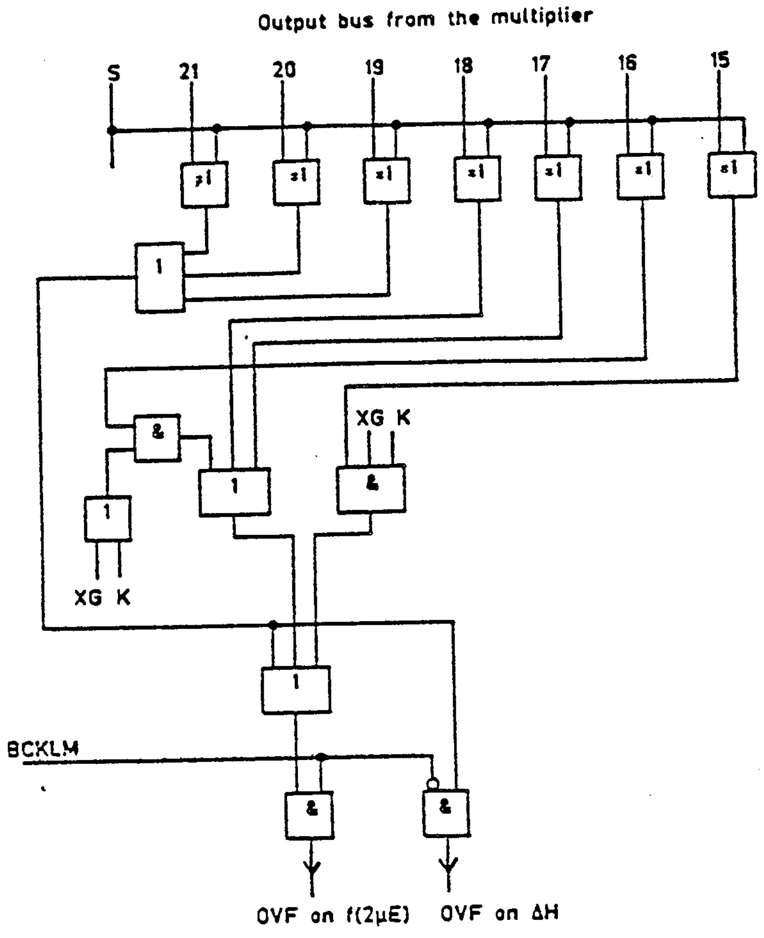


FIG. B.7 OVERFLOW DETECTION FOR f(2μE) AND ΔH

| Shift code | s | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | -2 | -3 | -4 | -5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ⟵ Output shifter field ⟶ | | | | | | | | | | | | | | | | |
| 000 | s | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | } for $C=2^{28}$ | | | |
| 001 | s | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | } | | | |
| 000 | s | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | } for $C=2^{29}$ | | | |
| 001 | s | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | } | | | |
| 010 | s | s | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 |
| 011 | s | s | s | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 100 | s | s | s | s | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 |
| 101 | s | s | s | s | s | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| 110 | s | s | s | s | s | s | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
| 111 | s | s | s | s | s | s | s | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |

Table B.4 Input-bus order relative to output for 8-way barrel-shifter.

Figure B.8 shows a partial implementation of the barrel-shifter, using transmission gates.

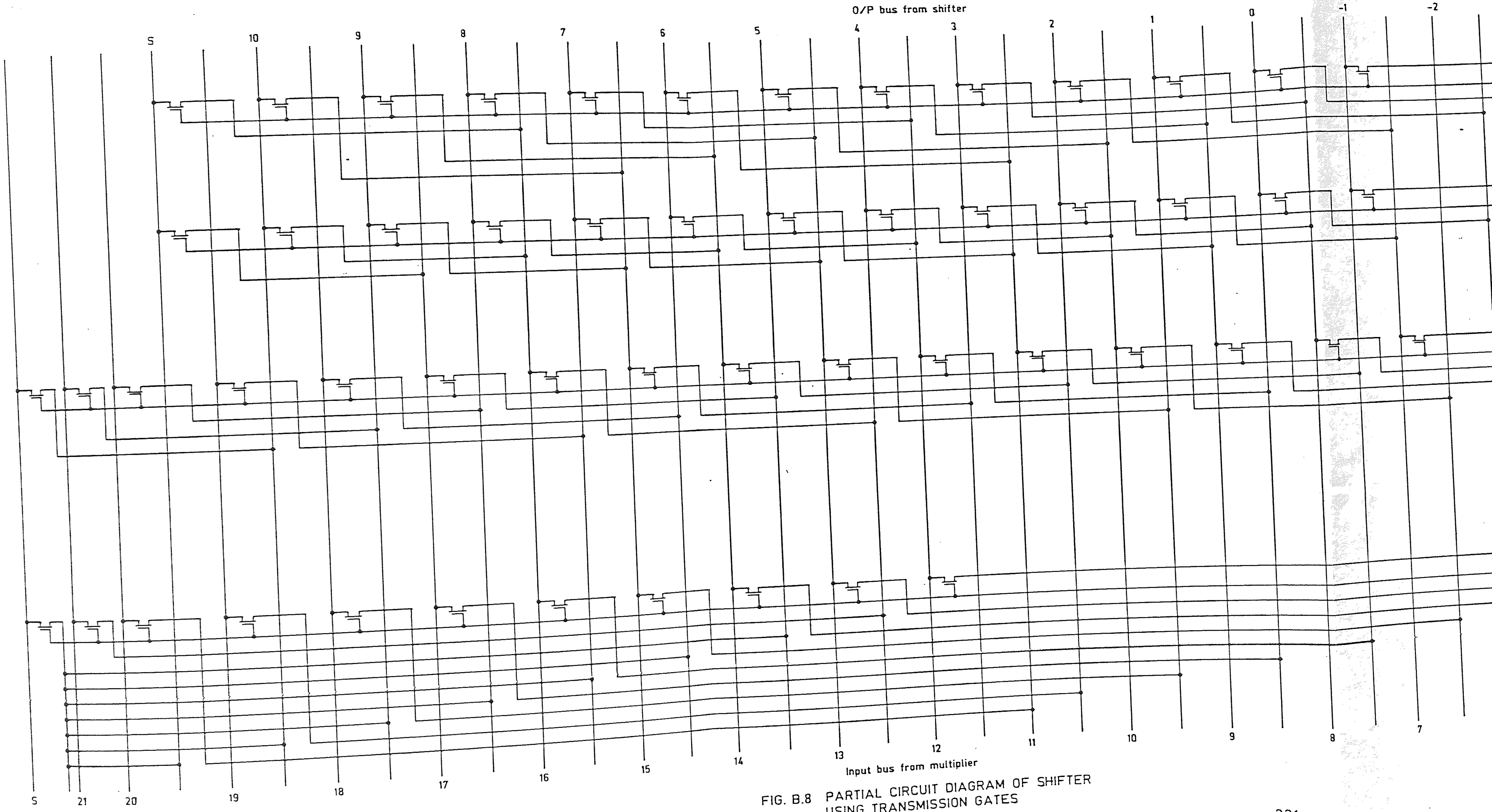


FIG. B.8 PARTIAL CIRCUIT DIAGRAM OF SHIFTER USING TRANSMISSION GATES

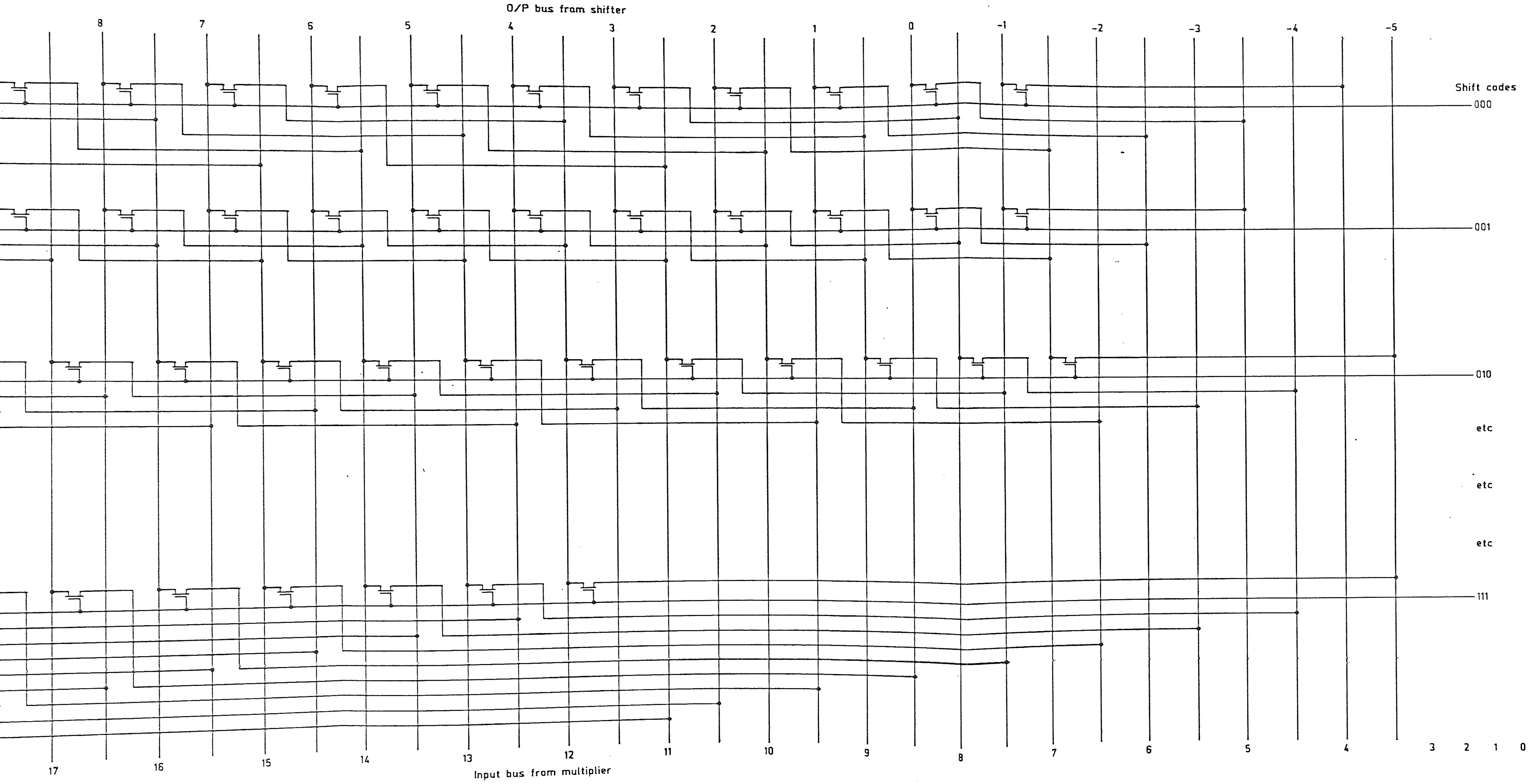


FIG. B.8 PARTIAL CIRCUIT DIAGRAM OF SHIFTER USING TRANSMISSION GATES

Zero after the shifter.

There are 4 conditions which require the output bus from the shifter to be set to zero:

1) overflow detected at the SSX SIPO-1 during BCLKM,

2) overflow detected on f(2μe) bus during BCLKM,

3) overflow detected on the Δh bus during NOT(BCLKM),

4) to prevent the filter from further adaptation during NOT(BCLKM), in response to the external command 'Recirculate' (applied to control line C5).

Figure B.9 shows the logic to perform the above zero condition.

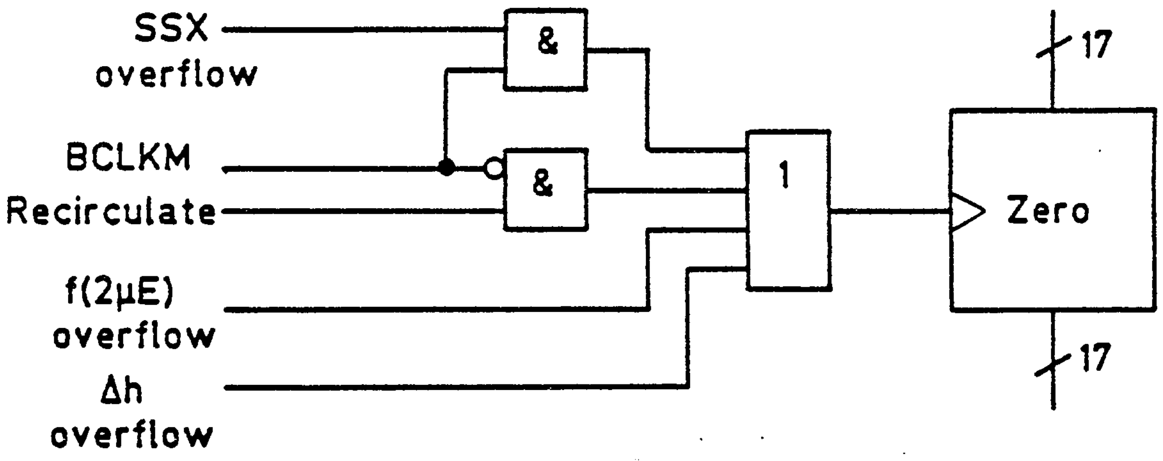


FIG. B.9 LOGIC TO ZERO SHIFTER OUTPUT

Latch L-1.

Referring to figure B.2, a 17-bit latch (L-1) follows the zero function, to store either f(2μe) (13-bit data-word) or Δh (17-bit data-word). Depending on the type of latch used, data must be valid by one of the edges of the latch pulse $\alpha_1$. The output of the multiplier is valid from the falling edge of $\alpha_3$. Hence, the the time allowed for propagation through the overflow detector, barrell shifter and zero function is set. Output from the latch L-1 feeds the 16 bit ADDER-3 (plus carry in) of h+Δh and a rounding 12-bit (plus carry in) ADDER-2, for f(2μe), prior to the B-latch of the multiplier.

Rounding Adder and Multiplexer MPX-3.

Section 3.8 described the necessity of removing the offset produced by truncating a two's complement number and the simple technique of adding the bit immediately below the LSB of the data word, to the data word.

ADDER-2 is used to round f(2$\mu$e). Referring to figure B.2, of the 13 MSBs from the output of latch L-1 the upper 12 go to the A-input of ADDER-2 and the LSB to the carry in. The B-input is tied to zero. One input code will produce an overflow, i.e. positive full-scale, which rounds to negative full-scale. Therefore, positive full-scale is detected and used to zero the carry-in of ADDER-2.

The resulting 12-bit word (f(2$\mu$e) is multiplexed with the 12-bit ERROR (adapt) and input to the B-latch of the multiplier during strobe MPX1+3 high.

B.2.5 H Update.

The 16-bit value of $h_{old}$ (ignoring the coefficient designation) is entered onto the V bus, from the memory device (see figure 4.1), on the rising edge of clock $\Omega_{41}$ (see figure B.1). Allowing for a settle time, the data must be stable by the rising edge of $\Delta_1$, when it is stored in the $H_{old}$ latch (L-2). The 17-bit value for $\Delta h$ is stored in L-1 at the same time. $h_{old}$ and $\Delta h$ are applied respectively to the A and B inputs of a 16 bit ADDER-3, the LSB of $\Delta h$ to the carry-in to remove the offset. Under normal operation the output from the adder, $h_{new}$, is checked for overflow, together with (half-full-scale).NOT(overflow), which are output from the filter on the C bus (multiplexed onto C7). The latter condition provides a limited amount information on the maximum amplitude of H stored in the filter. To optimise the numerical precision of the filter this condition should be set (logic 1). If it is logical 0, then, this can be detected externally and the value of k adjusted. Either $h_{new}$ or $h_{max}$ (if overflow is detected) is output onto the U bus, to be used by the multiplier-accumulator device for convolution, and stored in the memory device.

Prior to $H_{new}$ being applied to the U bus the value can be reset to zero, i.e. applying a ZEROH condition externally on the C bus (C5). Tri-state buffers interface between ADDER-3 output and the U bus, applying $h_{new}$ during clock $\Omega_{23}$ high. A data value must have settled on the U bus, at the input to the multiplier-accumulator and memory devices, by the rising edge of $\alpha_3$. Therefore, the total time available for addition, overflow detection and and bus settling is a half-process-cycle (240 ns at the 8 kHz sample rate).

A feature of the filter design is the provision to reconnect the system on a larger U and V bus. The object is to decrease the q-factor and hence improve adaptation for noisy reference paths. In simple terms, the convolution (on the multiplier-accumulator) is performed on the 16 MSBs, while the convergence device is directly connected to the 16 LSBs. Storage of the extra bits for the two impulse responses is on an additional memory device and the $H+\Delta H$ adder (ADDER-3) is extended over the total U and V buses by external fast adders. A circuit description is given later. The above is sufficient to understand the provision of the associated output pins. To use this feature a pin INTernal/EXTernal is held at logical 1, then the sign of $\Delta h$ and carry from ADDER-3 (which now forms a partial sum) are connected to output pins (via fast buffers) to form part of an extended adder. In an extended mode overflow presents a problem, since to use the full-scale facility ($h_{max}$) on the device requires information that an overflow has occured plus the correct sign of $h_{new}$ (to provide all 1s or 0s on the 16 LSBs), i.e. two input pins must be included. An economic constraint on the number of pins (i.e. 64) allowed only one pin for overflow. Consequently an overflow, if detected, signals the convergence device on a pin designated EXTOVF, which is then transmitted to the C bus for the host processor to act upon.

Figure B.10 shows a simple schematic diagram of the external adder and overflow detector, associated with the convergence device.

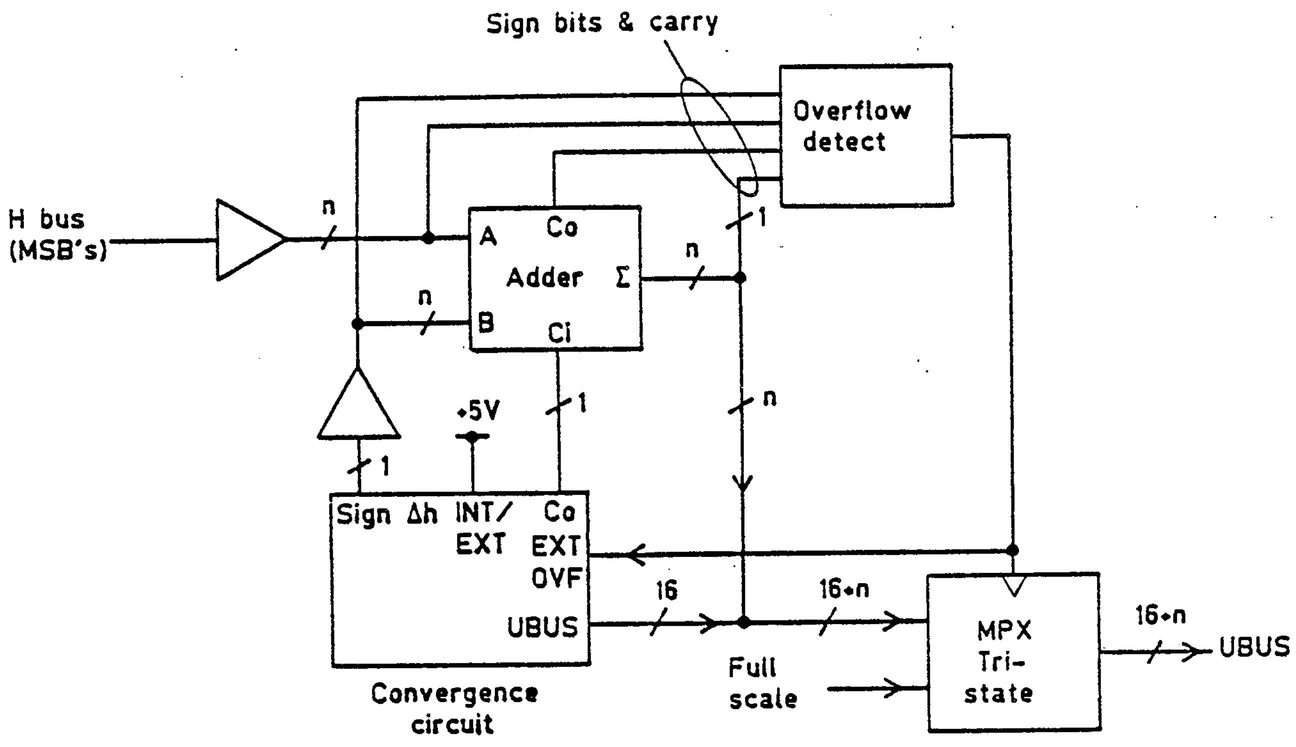


FIG. 8.10 EXTERNAL ADDITION AND OVERFLOW DETECTION FOR q-FACTOR REDUCTION

The time required to perform a 16 bit addition, in ADDER-3, is a function of the physical design, which, for worst-case parameters, has been estimated to be 160 ns. Estimating worst-case propagation delays for external processing using fast devices, based on a 4 bit extension, i.e.

| | |
|---|---|
| Sign extend buffer (5 outputs into 15 pf) | 4 ns |
| 4-bit add | 9 ns |
| overflow detect (1 INVERT plus 2 NANDs) | 13 ns |
| overflow correction (AND gate plus tri-state) | 13 ns |
| bus settling | 7 ns |
| Total delay | 46 ns |

shows that approximately 50 ns is required.

The total time for $h_{old} + \Delta h$ is a half-process-cycle, which is approximately 240 ns at the 8 kHz sample-rate. Thus 30 ns can be allocated to extending the q-factor reduction. NB in a prototype system an 8-bit extension was implemented using LSI devices.

$h_{old}$ latch - 16 bit.

Referring to figure B.2, impulse-response coefficients $h_a$ are written to the V bus (upper left corner) on the rising edge of $\alpha_3$ and stored in the $h_{old}$ latch (L-2) on $\alpha_1$. The TTL emulator used a transparent latch (type 7475), which opens on the rising edge and latches on the falling edge of its clock. Thus, enabling addition to begin from the rising edge of the clock.

16 bit adder (ADDER-3)

The A input is from the $h_{old}$ latch (L-2). The $\Delta h$ latch (L-1) provides the B input and its LSB to the carry-in port (for rounding the truncated $\Delta h$). TTL devices 74283 were used in the emulator, having a worst case propagation delay (carry in to sign out) of 75 ns, much faster than the LSI equivalent. Carry out and the sign of $\Delta h$ were brought to output pins via buffers to drive fast TTL inputs.

Overflow detection and correction.

Figure B.11 shows the overflow-detection logic, utilizing equation (B.1) (for minimum propagation delay) to determine an overflow out of ADDER-3, which is dependent on the state on pin INT/EXT. The half-full-scale (HFS) condition corresponds to detecting a difference between the sign and bit 9 (using the numbering scheme of figure 3.8)

## B.2.6 The Control Bus (CBUS).

Referring to figure B.2 (lower right), data is externally applied to the CBUS and stored in either Creg0 (bits C2 to C5) or Creg1 (bits C2 to C6) depending on the output value of the address decoder. Figure B.12 shows the logic to decode the 3 input-lines (AC/NEL, C0 and C1) during the CB strobe (begining in process-cycle 251). If the logic states on C1 and AC/NEL are equal, then, any changes to be made on the DAF will be incorporated in the next audio-cycle.

Table B.4 shows the codes required by the barrel-shifter to perform the scaling for K, Q and c. K is decoded from the CBUS by C0.C4 (from Creg0), and toggles the LSB of the B input of MPX-5 to generate either 000 or 001 during the pause. Q and c must be coded according to:

$$\text{shift code} = Q + c + 2 \qquad \ldots \text{(B.9)}$$

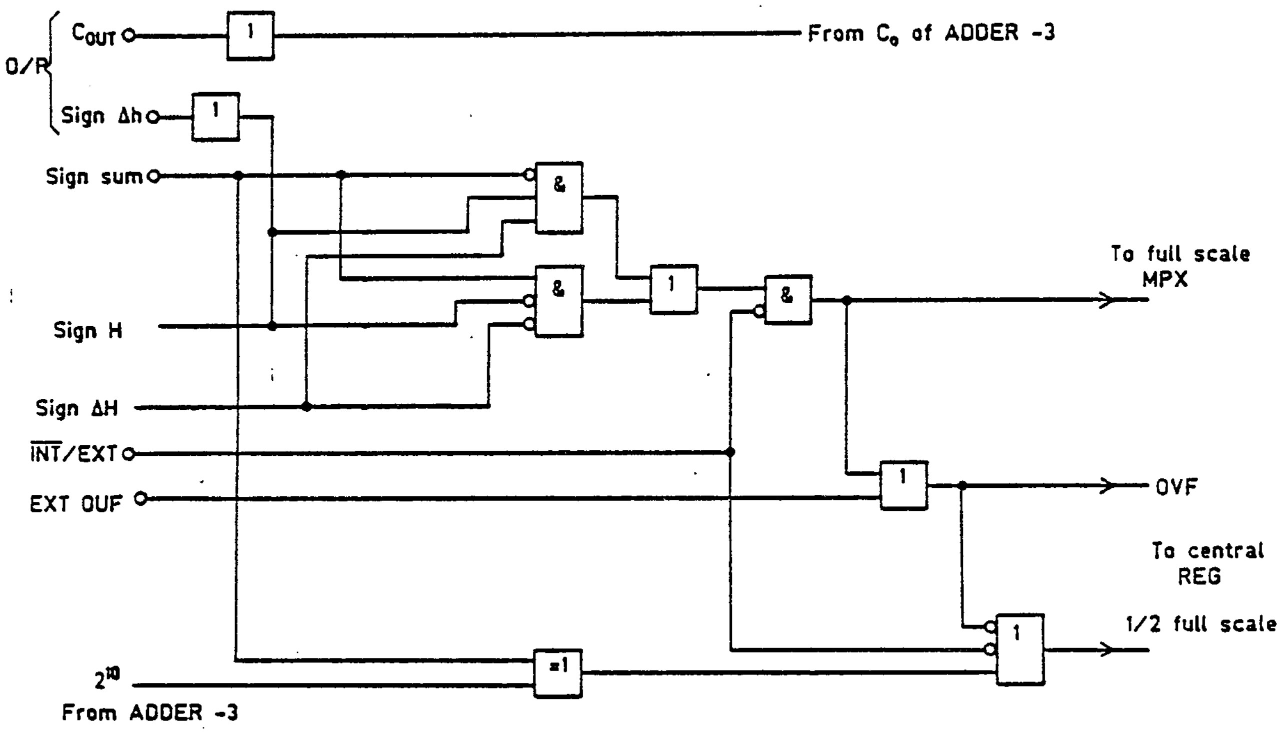


FIG. B.11 OVERFLOW AND HALF FULL SCALE DETECTION FOR h • Δh

where 2 is the displacement caused by utilizing the first two codes for K. A better system would have been to allocate shifter codes 110 and 111 to K. To obtain the displacement, 1 is added to the carry-in of ADDER-4, and, since a code is unused in the cascade word, cascade is recoded to cascade + 1. The coding becomes:

| Configuration | Cascade (i/p) | | Encoded word (o/p) | |
|---|---|---|---|---|
| | Cas1 | Cas0 | Cas1 | Cas0 |
| single | 0 | 0 | 0 | 1 |
| dual | 0 | 1 | 1 | 0 |
| quad | 1 | 0 | 1 | 1 |
| quad (also) | 1 | 1 | 1 | 1 |

for which simple Boolean expressions can be found, i.e.

$$Cas0_{o/p} = \overline{Cas0}_{i/p} + Cas1_{i/p}$$

$$Cas1_{o/p} = Cas0_{i/p} + Cas1_{i/p} \qquad \ldots \text{(B.10)}$$

Equation (B.9) is solved by summing the word on C2,C3 from Creg0 and the encoded C2,C3 word from Creg1 in ADDER-4. The sum provides the shift codes required in Table B.4, via MPX-5, during the burst period.

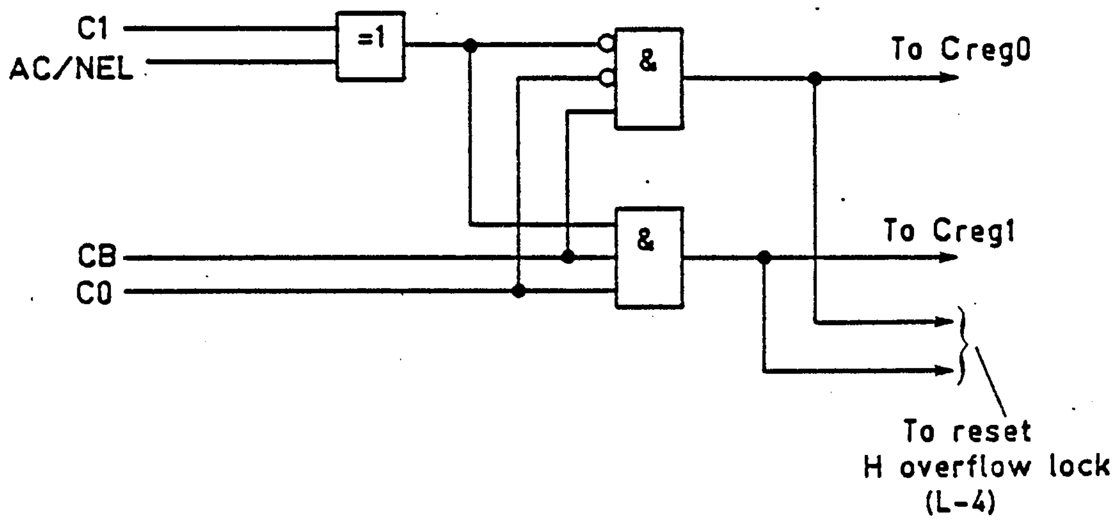


FIG. B.12 LOGIC FOR CBUS ADDRESS DECODER

The two output conditions multiplexed onto C7 (see table 4.1) are derived from the overflow detector associated with the H + ΔH sum in ADDER-3. Each value is valid during $\alpha_1$.BURST (to avoid H values in the garbage locations producing an output) and stored in the H overflow latch L-3. Transfer to the H overflow lock, L-4, takes place on strobe PL1, the previous contents being removed by a reset pulse, produced by the external controller writing to the CBUS (i.e. an output from the CBUS address decoder). The contents of the H overflow latch (L-3) are subsequently cleared by a reset pulse BCLKM.NOT(MPX1+3). The structure described allows an external controller to perform a WRITE then READ instruction (at some time after the rising edge of Δ) after an arbitary number of audio cycles and be able to detect (via MPX-11) if either condition has occurred since the previous WRITE then READ instruction.

## B.3 Multiplier and Triple Accumulator.

Figure B.13 shows the logic diagram for the device and figure B.14 the local strobe waveforms (excluding those covered in figure B.3).

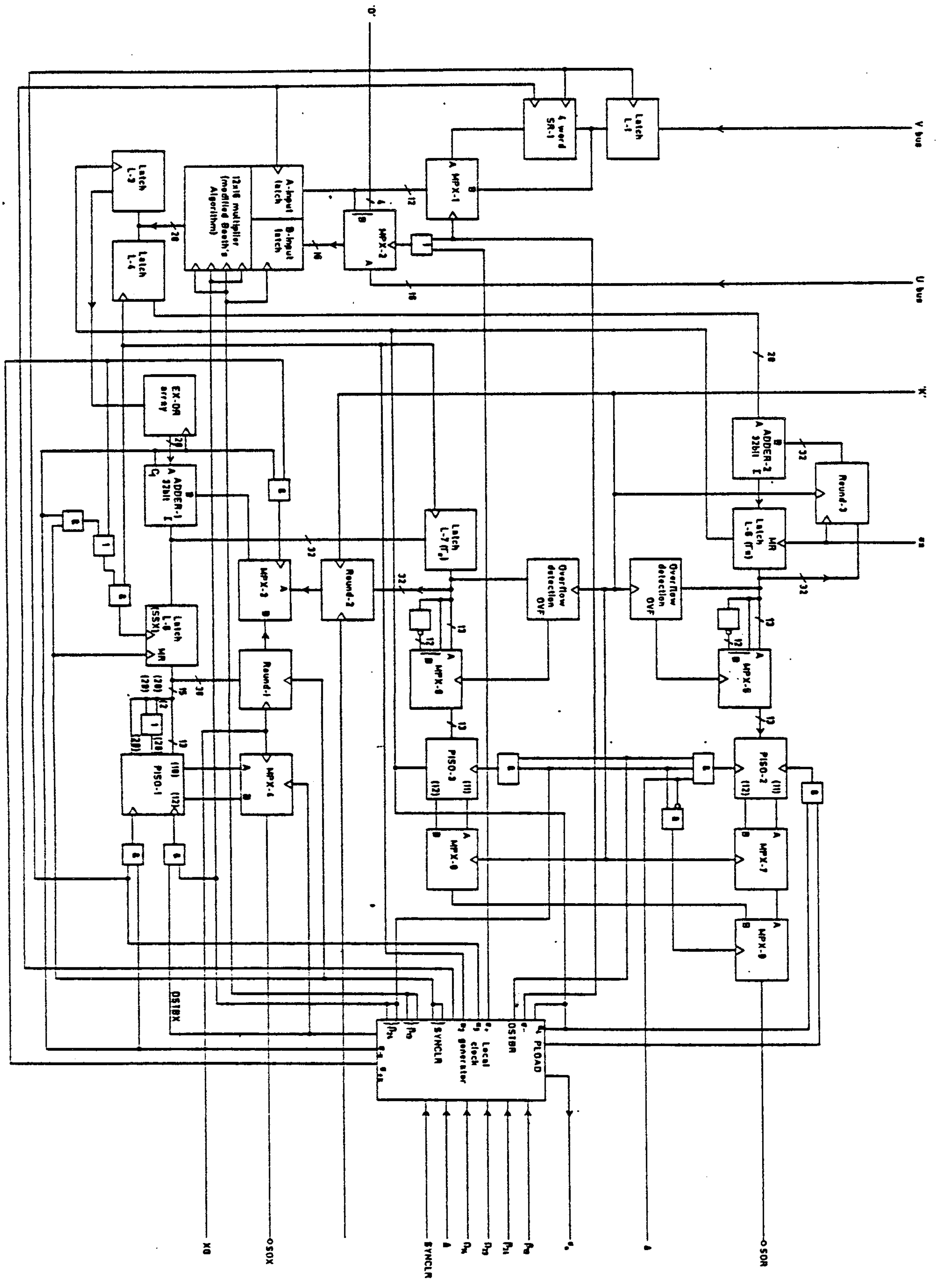


Fig. B.13 LOGIC OF MULTIPLIER AND TRIPLE ACCUMULATOR.

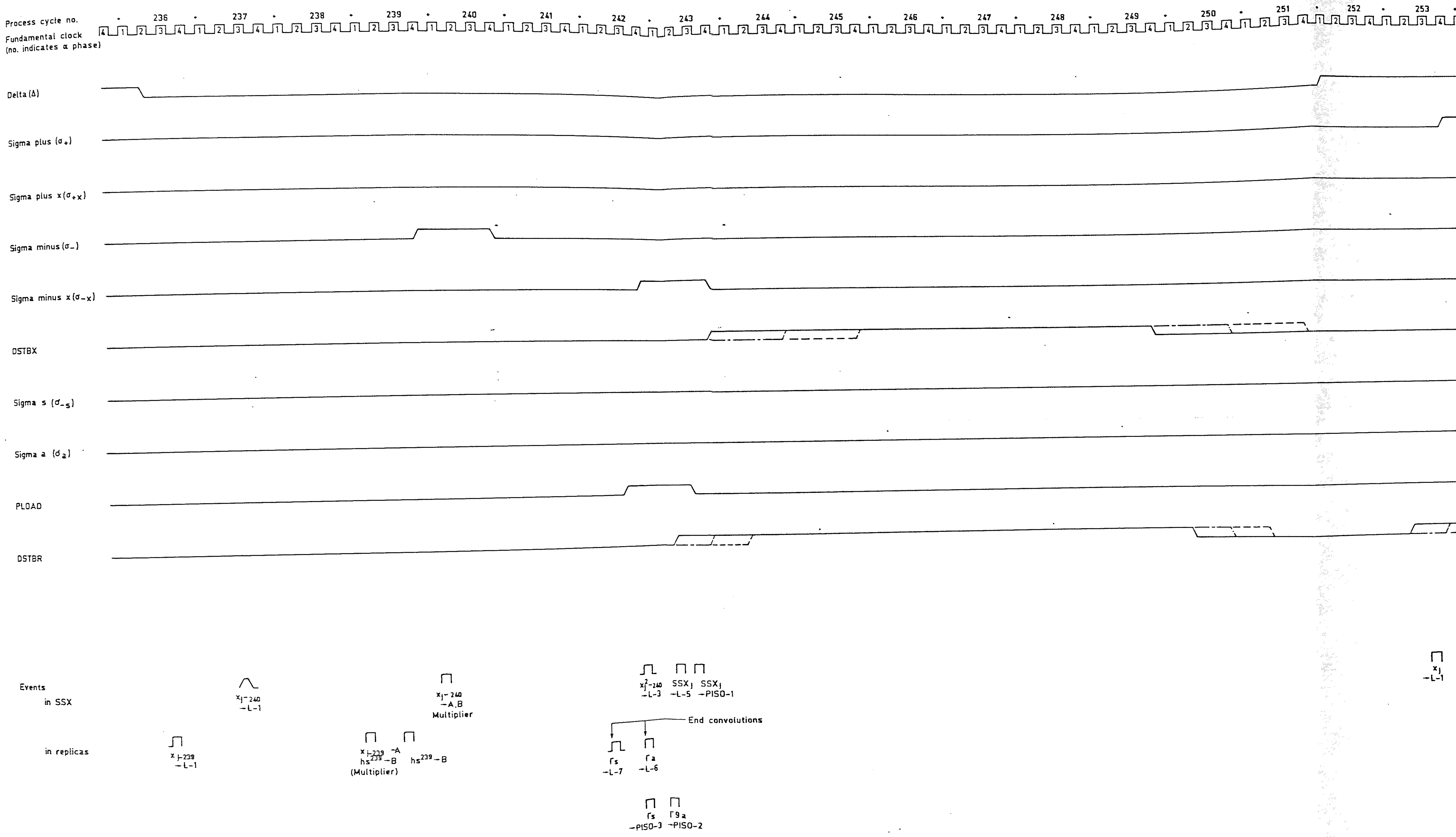


FIG. B.14 LOCAL STROBES FOR MULTIPLIER AND TRIPLE ACCUMULATOR

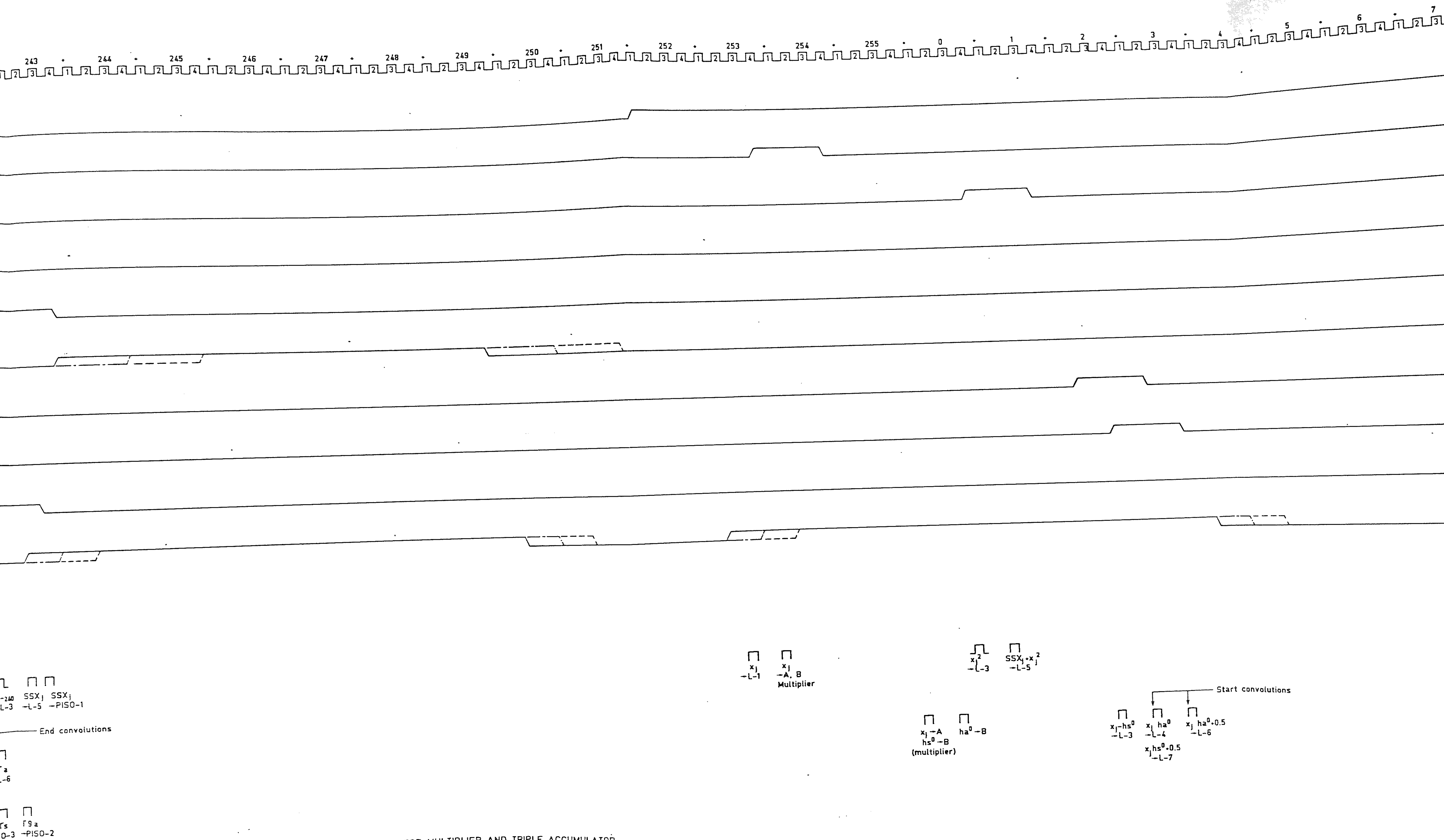


FIG. 8.14 LOCAL STROBES FOR MULTIPLIER AND TRIPLE ACCUMULATOR

Sum-of-squares of x.

The equation (3.30) to be solved is:

$$SSX_j = SSX_{j-1} + x_j^2 - x_{j-240}^2$$

From figure B.1, $x_j$ (the newest value of x, about to be used in the convolution) is present on the V bus during $\Omega_{23}$ in process-cycle 253. Referring to figure B.13, $x_j$ is stored in latch L-1 during $\beta_{13}.\Omega_{23}$. Strobe $\sigma_+$ (see figure B.14) switches MPX-1 to its B-input, bypassing the delay element SR-1, so that it is stored in the A-latch of the multiplier on $\alpha_1$ in process-cycle 254. $\sigma_+$ also switches MPX-2 to its B-input, thus, $x_j$ is stored in the multiplier's B-latch simultaneously with $x_j$ in the A-input latch. The multiplier requires 12 clock-edges to form the product. Therefore, the product $x_j^2$ is stored in latch L-3 on $\alpha_4$ in process-cycle 0 and presented to the A-input of ADDER-1. A strobe $\sigma_{+x}$, which is $\sigma_+$ delayed 3 process-cycles, switches multiplexer MPX-3 to its B-input. Latch L-5 is the SSX store, and its output is now connected to the B-input of ADDER-1. The partial sum output from the adder ($SSX_{j-1} + x_j^2$) is written into the SSX store (L-5) on $\alpha_2.\sigma_{+x}$ (in process-cycle 0).

Now consider the subtraction of $x_{j-240}^2$ from the partial sum. $x_{j-240}$ appears on the V bus during $\Omega_{23}$ in process-cycle 237 (figure B.1), and is stored in latch L-1 on $\alpha_3$. The data word passes through the normal 2 process-cycle delay of the shift-register SR-1, and is stored in the multiplier's A-input latch on $\alpha_1$ in process-cycle 240. It is applied simultaneously to the B-input latch via MPX-2, switched on strobe $\sigma_-$. The product, $x_{j-240}^2$, is stored in the multiplier output latch L-3 on $\alpha_4$ in process-cycle 242. The strobe $\sigma_{-x}$ ($\sigma_-$ delayed 3 process-cycles) inverts the data word out of the multiplier into the B-input of ADDER-1, while also setting the carry-in to logic 1. The result is the negation of $x_{j-240}^2$ into the adder. Multiplexer MPX-3 is switched to its B input and routes the partial sum stored in the SSX latch (L-5) to the B input of ADDER-1. The subtraction is completed and $SSX_j$ is stored in L-5 on

$a_2.\sigma_{-X}$ in process-cycle 243. $SSX_j$ is parallel loaded into PISO-1 on $a_3.\sigma_{-X}$, prior to the begining of the serial output mask DSTBX. Slight variations in clock edges were used in both real-time systems to suit the devices available.

Overflow detection can be simplified by considering 'don't care' conditions. As stated in Section B.2.4, it is necessary to trap overflows for the number range selected by pin XG = 1. The truth table for overflow, as a function of bit 28, is given in Table B.5.

| input | | | output |
|---|---|---|---|
| XG | bit 29 | bit 28 | bit 28 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | ? |
| 0 | 1 | 1 | ? |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

? indicates don't care

Table B.5 Truth table for SSX overflow.

Therefore, the overflow condition is given by:

$$\text{bit } 28_{out} = \text{bit } 29 + \text{bit } 28 \qquad \text{... (B.11)}$$

The choice of LSB to begin the serial output in DSTBX (bit 17 for XG = 1, bit 18 for XG = 0) is made in multiplexer MPX-4.

An important point to note is the filling of SSX prior to starting adaptation, since it is necessary to inhibit the subtraction of $x^2_{j-240}$ for 240 audio cycles. A simple NOT(CLEAR) presents some difficulties. Since it would occur asynchronously, each device in the filter (including cascaded filters) would have to synchronise individually. Propagation delays create the possibility of error. Therefore, the master-clock generator receives the NOT(CLEAR) signal and provides a synchronous (SYNCLR) signal. The structure of

SYNCLR, shown in figure B.15, provides sufficient edges so that it may be multipurpose. Falling edges reset latches and counters, while a logic 0 inhibits subtration of $x^2_{j-240}$. i.e. the falling edge of SYNCLR resets the SSX latch (L-5), which is then clocked on the composite strobe $\alpha_2.(\text{SYNCLR}.\sigma_{-x} + \sigma_{+x})$.

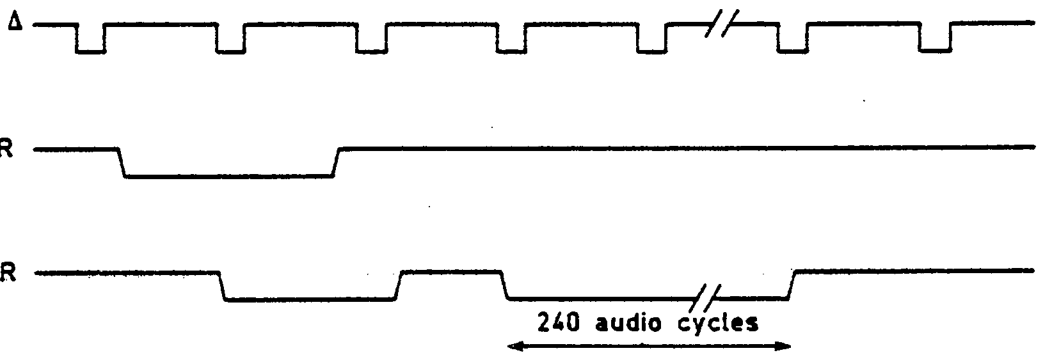


FIG. B.15 STRUCTURE OF SYNCLR WAVEFORM (NOT TO SCALE)

**Rounding.**

Both SSX and replicas are truncated two's complement numbers and therefore have an offset of -0.5 LSB (described in section 3.8). A convenient solution is to initialise to a 0.5 LSB instead of zero. For SSX, this requires the detection of the falling edge of SYNCLR and forcing the appropriate bit high (depending on the logic level on XG) during $\sigma_{+x}$, for one audio cycle. One method of implementing this function for ROUND-1 (figure B.13) is shown in figure B.16.

Replicas $r_a$ and $r_s$.

Referring to figure B.13 the new value $x_j$ is stored in latch L-1 on $\alpha_3$ in process-cycle 253. It is delayed 2 process-cycles in the shift register SR-1, then stored in the multiplier's A-latch on $\alpha_1$ in process-cycle 0. Simultaneously, $hs^0$ is stored in the multiplier's B-latch (see U bus in figure B.1). The product $x_j.hs^0$ is stored in latch L-3 on $\alpha_4$ in process-cycle 2. $ha^0_{in}$ follows $hs^0$ into the multiplier's B-latch on $\alpha_3$ in process-cycle 0, and the product $x_j.ha^0_{in}$ is stored in latch L-4 on $\alpha_2$ in process-cycle 3. The subscript 'in' will be assumed from this point on.

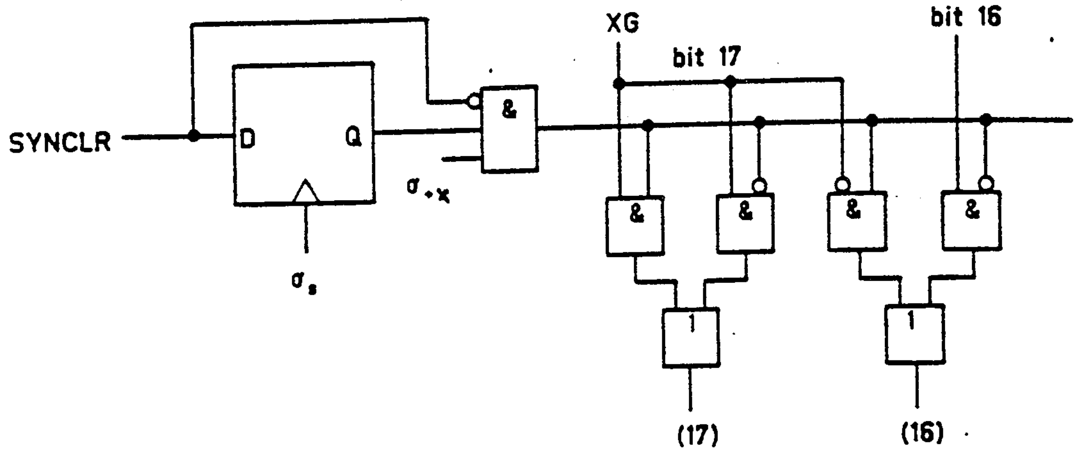


FIG. B.16 LOGIC FOR ROUNDING SSX

The first accumulation, for both replicas, requires a 0.5 LSB (dependent on k, set by C4 on the control bus) to be added to each of the two products above. Consider the stored replica ($r_s$), the output of L-3 ($x_j.hs^0$) is presented to the A input of ADDER-1. The B input of ADDER-1 is the output of the $r_s$ latch (L-7), via ROUND-2 and multiplexer MPX-3. L-7 was reset by the rising edge of strobe $\sigma_s$ in process-cycle 2, which can also be used, by ROUND-2, to provide the correct 0.5 LSB offset, as shown in figure B.17.

The sum $x_j.hs^0$ + 0.5 LSB is stored in L-7 on $\alpha_2$ in process-cycle 3. A similar circuit is provided for the adapt replica, using the strobe $\sigma_a$.

Accumulation continues for 240 h coefficients to compute each replica. The last x value is $x_{j-239}$ which appears on the V bus on $\alpha_3$ in process-cycle 236 and is stored in the multiplier A-input on $\alpha_1$ in process-cycle 239. Simultaneously $hs^{239}$ is stored in the B input. The product ($x_{j-239}.hs^{239}$) is stored in latch L-3 on $\alpha_4$ in process-cycle 241, then added to the partial stored-replica (in ADDER-1) and written into latch L-7 on $\alpha_2$ in process-cycle 242. Bits 10 to 22 of the final value of the stored replica ($r_s$) are parallel loaded into PISO-1 on $\alpha_4$ in process-cycle 242 (using strobe PLOAD). The bit selection is equivalent to dividing by g and truncating after the binary point. The adapt replica ($r_a$) follows a half-process-cycle later into PISO-2.

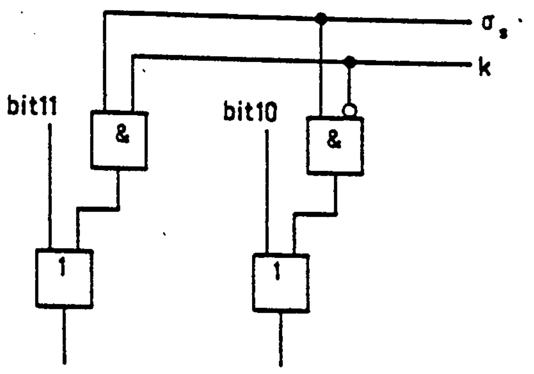


FIG. B.17 REPLICA ROUNDING

Overflow.

The MSBs not transmitted (dependent on the value of k) are compared with the sign bit in each replica store. If a difference is detected, full-scale is loaded into the appropriate PISO.

Finally the stored replica is output first, on DSTBR.NOT(Δ), via multiplexers MPX-8 (to select the LSB transmitted first, as a function of k) and MPX-9 which connects the correct replica to the output pin SOR. The adapt-replica follows on DSTBR.Δ.

Local Strobe Generator.

All local strobes were generated from a counter, reset on the rising edge of Δ, similar to the method used on the convergence circuit.

B.4 Circuit Description of the Memory.

Figure B.18 is a schematic representation of the memory circuit, showing complete buses. If the structure had been implemented in full for the LSI, the silicon area would have been large, consequently, producing a low yield of working devices. The LSI device was similar to figure B.18, except that the word size of the three vectors were halved. Thus, two LSI devices were required per filter. Three additional pins were needed: one defines which

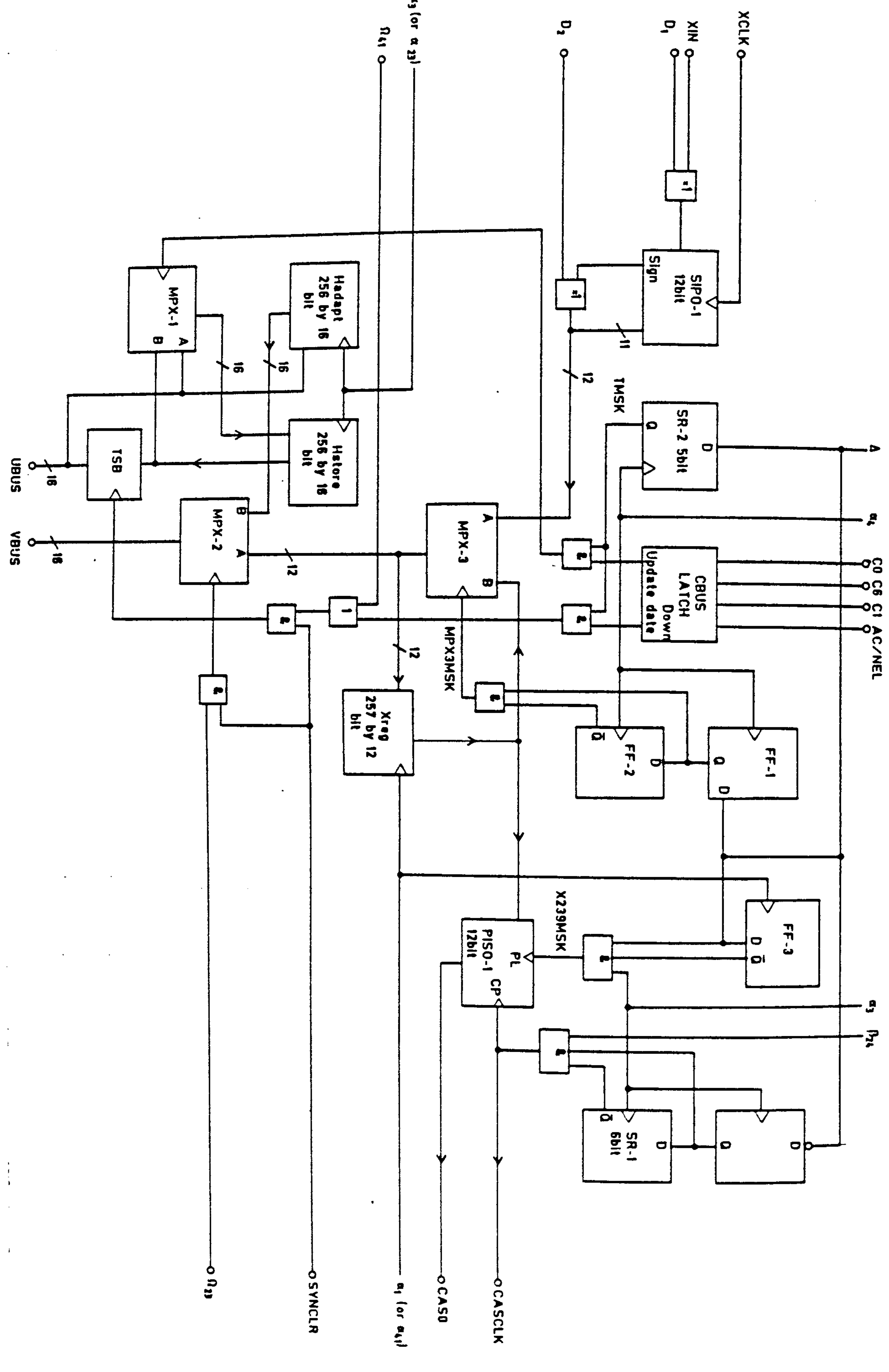


Fig. B.18 SCHEMATIC OF MEMORY CIRCUIT.

device contains the MSB (for number format changes); the other two extend the x-input SIPO and x-output PISO across both devices. The three pins are excluded from figure B.18.

Consider a signal $x_j$, derived serially from an a-to-d converter or alternative digital source, and clocked into SIPO-1 (via pin XIN, MSB first), using an external strobe XCLK. Pins $D_1$ and $D_2$ are used to convert number formats to two's complement, as shown in Table B.1. A strobe MPX3MSK (see figure B.19) switches multiplexer MPX-3 to its A input, i.e. connecting $x_j$ to its output. The output of MPX-3 is latched on $\alpha_1$ (or $\gamma_{41}$) by the X shift-register, overwriting the value of $x_{j-257}$. $x_j$ is also routed to the A input of multiplexer MPX-2, which is switched on strobe $\Omega_{23}$. Thus, $x_j$ appears on the V bus in process-cycle 253. The serial input must be complete before the rising edge of MPX3MSK and must not occur when the stobe is high. The component of X to be taken off the memory device, becoming the newest input word of the next filter section in cascade, is $x_{j-239}$, i.e. it will become $x_{j-240}$ in the next audio-cycle. The strobe X239MSK, used to latch $x_{j-239}$ into PISO-1, can be generated a number of ways. The method shown in figure B.18 uses the falling edge of $\Delta$ to the next $\alpha_1$, gated by FF-3. A similar circuit is used to generate CASMSK (FF-4 and SR-1), which is ANDed with $\beta_{24}$ to produce the cascade-output clock (CASOCLK).

NB the internal clock of the 12-bit a-to-d converter used in experimental measurements (Burr-Brown ADC80) produces 13 pulses. The first pulse signals the start of serial output, with data changing on the rising edge of succesive pulses, to be latched on the falling edge. Therefore, it is necessary to gate out the first pulse before application to the XCLK pin.

Both H registers are clocked on $\alpha_3$. $h_{old}$ is connected, via MPX-2, to the V bus during $\Omega_{23}$ low and the updated $h_{new}$ (from the convergence circuit) returns via the U bus to be stored in the top of the shift-register. The output from the Hstore register is connected to the U bus, via a tri-state buffer, which transmits on $\Omega_{41}$ high. It is also connected back to its input via the B input of multiplexer MPX-1. This multiplexer enables the transfer of the contents of Hadapt to Hstore. After detecting an UPDATE command from the CBUS (as described in Section B.2.6) a strobe TMASK.UPDATE switches MPX-1 to its A input and the first 240 coefficients of Hadapt, from the U bus, are written to both H registers. TMASK is derived by delaying $\Delta$ 5-process-cycles on $\alpha_4$.

To implement a DOWNDATE after the command has been detected on the CBUS, the tri-state buffer is held in transmit by strobe TMASK.DOWNDATE. Therefore, the input of the Hadapt register is connected to the output of Hstore. NB the tri-state buffer on the convergence device is held at high impedance during a downdate command, preventing any bus contention.

## B.5 Circuit Description of H Extension.

Figure B.20 shows a block diagram for an 8-bit H extension. H comprises 24 bits, numbered 11 to -12 (relative to the binary point). Consider the V bus (upper left), on which X (12 bits) and the 16 MSBs (bits 11 to -4) are time multiplexed. Convolution is carried out with this number range. The additional lower 8-bits of H (bits -5 to -12) are stored in the extra memory circuit (MEM). On $\alpha_1$, an external $H_{old}$ latch (L-1) stores the 8 MSBs (bits 11 to 4), while the 16 LSBs (bits 3 to -12) are stored in the $H_{old}$ latch (L-2) on the convergence circuit. An external multiplexer (MPX-1) selects either X or the 16 LSBs of H for the convergence circuit. The 16-bit $\Delta H$ is added to the 16 LSBs of H in ADDER-3, while the sign of $\Delta H$ is brought off the convergence circuit and connected to the B input of external ADDER-1. The 8 MSBs from latch L-1 form the A input to ADDER-1. The extended addition is completed by connecting the carry out from ADDER-3 to the carry in of ADDER-1. As explained in Section B.2.5, insuficient pins prevent the use of the overflow circuit on the convergence circuit, which is therfore disabled with a single pin

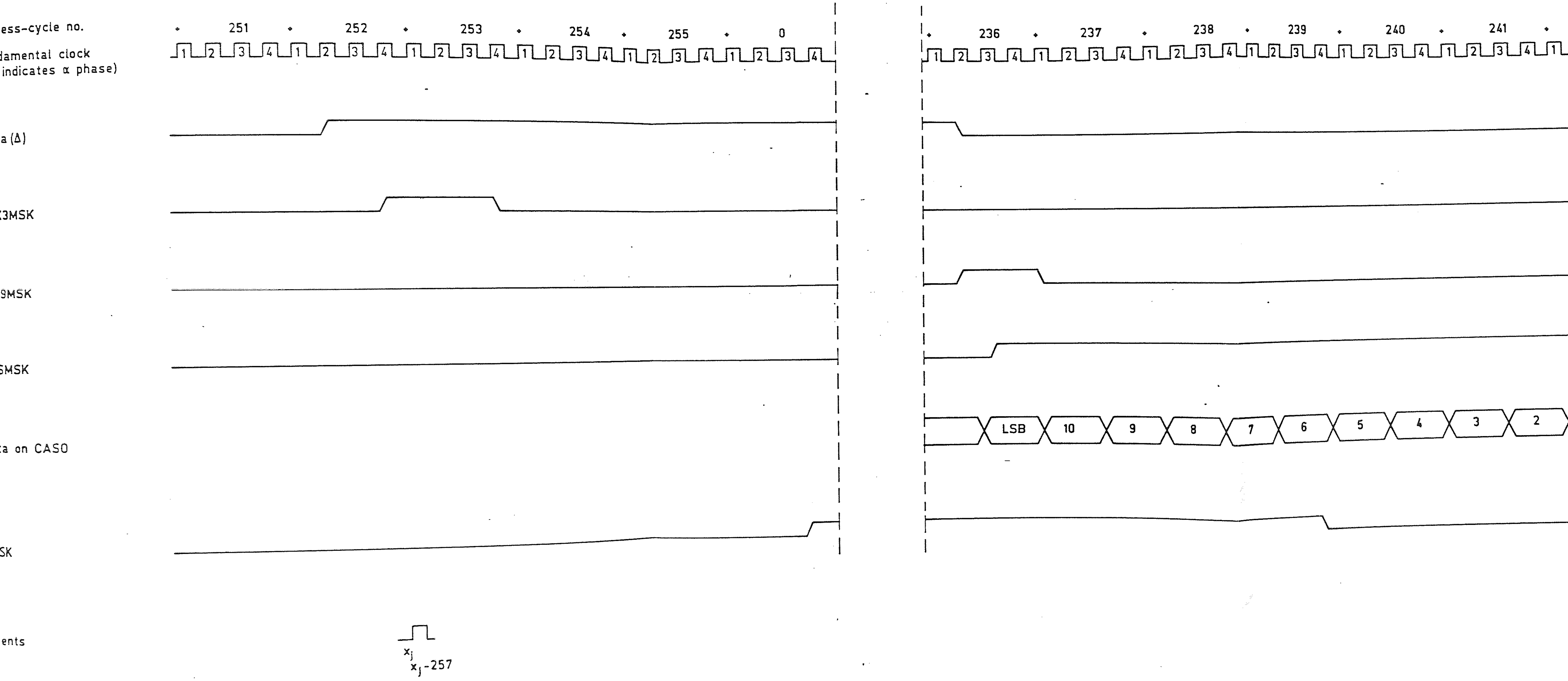


FIG. B.19 WAVEFORMS FOR MEMORY

-state
input
fer of
m the
-1 to
, are
cycles

CBUS,
efore,
NB the
ring a

prises
V bus
e time
itional
circuit
to 4),
on the
or the
the 16
ergence
3s from
eted by
plained
cuit on
gle pin

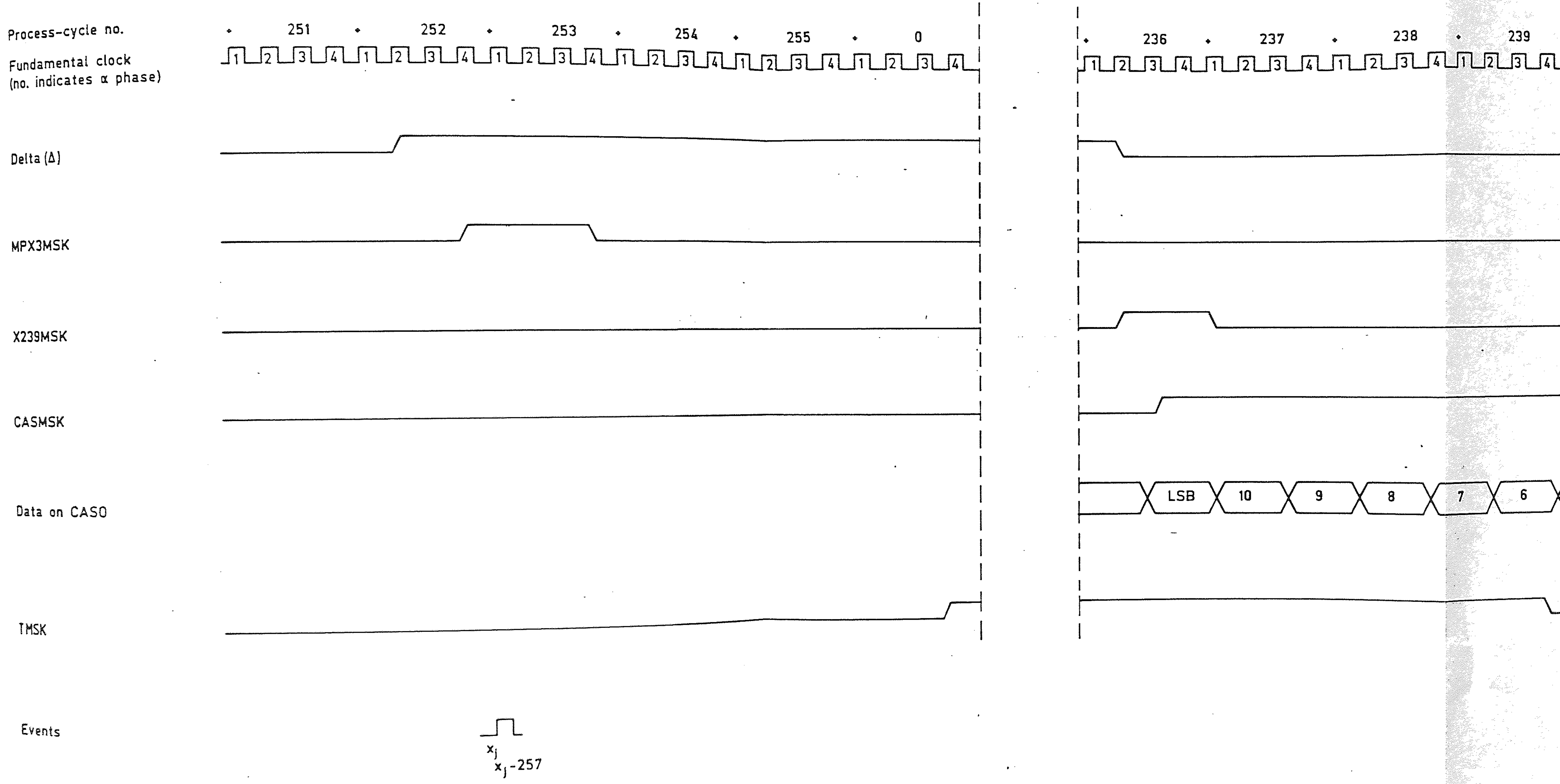


FIG. B.19 WAVEFORMS FOR MEMORY

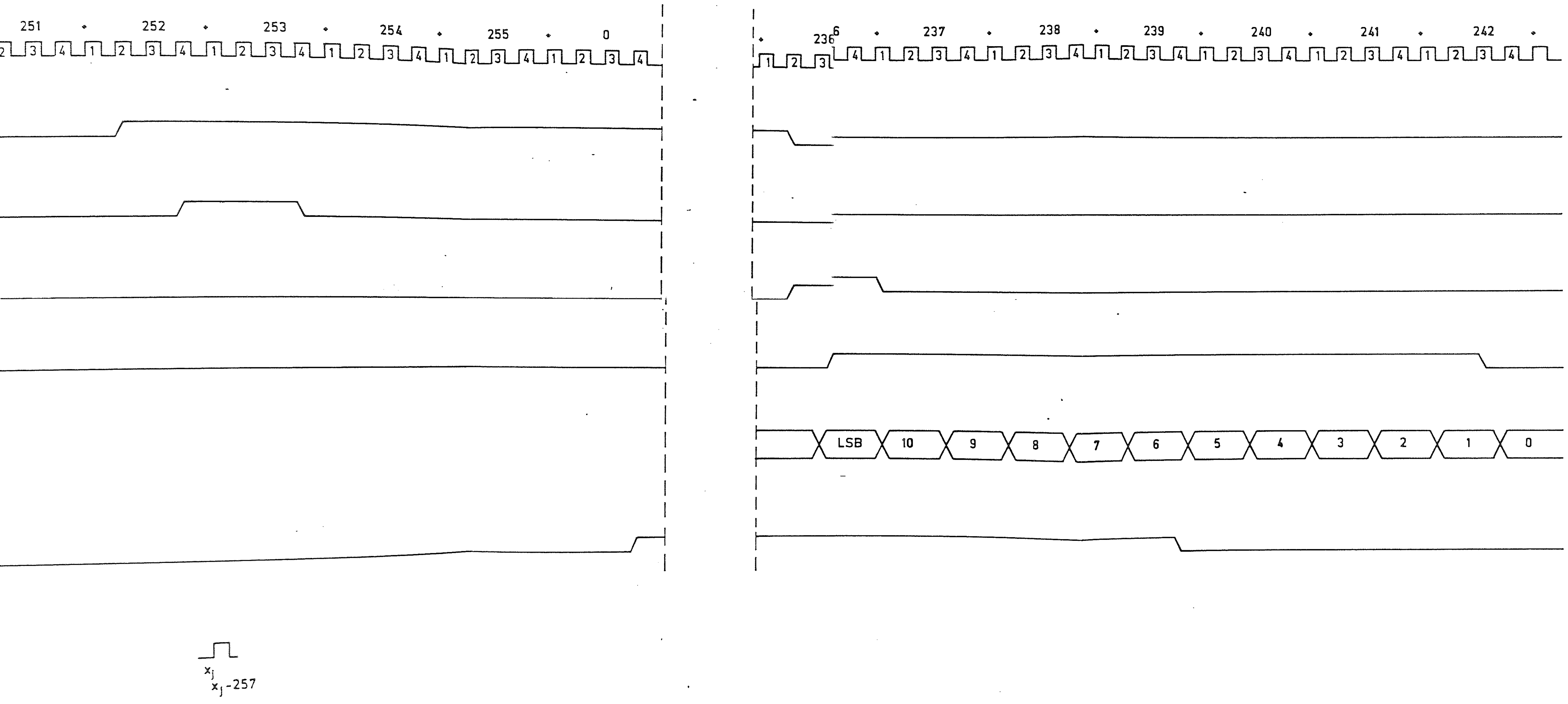


FIG. B.19 WAVEFORMS FOR MEMORY

This page has been left intentionally blank

(INT/NOT(EXT)). Overflow is detected externally and passed back to the convergence circuit, via pin EXTOVF, to be read by an external controller from the CBUS (C7). The data bits of the resulting new value of H are then distributed to the memory circuits. Bits 11 to -4 to the standard filter and bits -5 to -12 to the additional storage. It is also necessary to detect the ZERO H condition on the CBUS and zero the 8 MSBs of H using MPX-2.

NB the filter construction described in figure B.20 has a Q-shift range of 8 to 12, giving a 24 to 36 dB improvement in adaptation over a standard filter for poor echo-to-noise ratios.

## B.6 Circuit description of the H Monitor.

Referring to figure B.21, a counter (COUNT-1) is reset on SYNCLR, to define the coefficient $h^0$, and increments by one on the first occurence of $\alpha_4$ after $\Delta$. The value 239 is detected on the output of COUNT-1 and used to reset the counter to 0, thus counting 240 audio-cycles. The waveform out of DETECT-239 is one audio-cycle long, from the rising edge of $\alpha_4$ in process-cycle 252, as shown in figure B.22. The rising edge of DETECT-239 trips a monostable to provide a trigger pulse for an oscilloscope. The waveform DETECT-239 is delayed 3 process-cycles in shift-register SR-1, and flip-flop FF-2 derives a pulse one process-cycle wide (shown as PHI) from the rising edge of $\alpha_3$ in process-cycle 255. PHI is used to reset a second counter (COUNT-2) which counts process-cycles on $\alpha_4$. Two outputs from COUNT-2 are detected, DETECT-256, i.e. 257 process-cycles later, resets the counter to zero: DETECT-0 is the waveform (EPSILON) which masks the stored and adapt h coefficients. The 12 MSBs of the U bus are latched in L-1 on $\alpha_1$ and L-2 on $\alpha_3$ to store the values of $hs^0$ and $ha^0$ respectively. NB ADD-1 is used to round the truncated two's complement number. The latch outputs are converted to analogue signals and displayed on the oscilloscope. After 257 process-cycles EPSILON has slipped one place and masks coefficients $hs^1$ and $ha^1$. After 240 audio-cycles from PHI, EPSILON masks coefficients $hs^{239}$ and $ha^{239}$. In the next audio-cycle COUNT-1 is

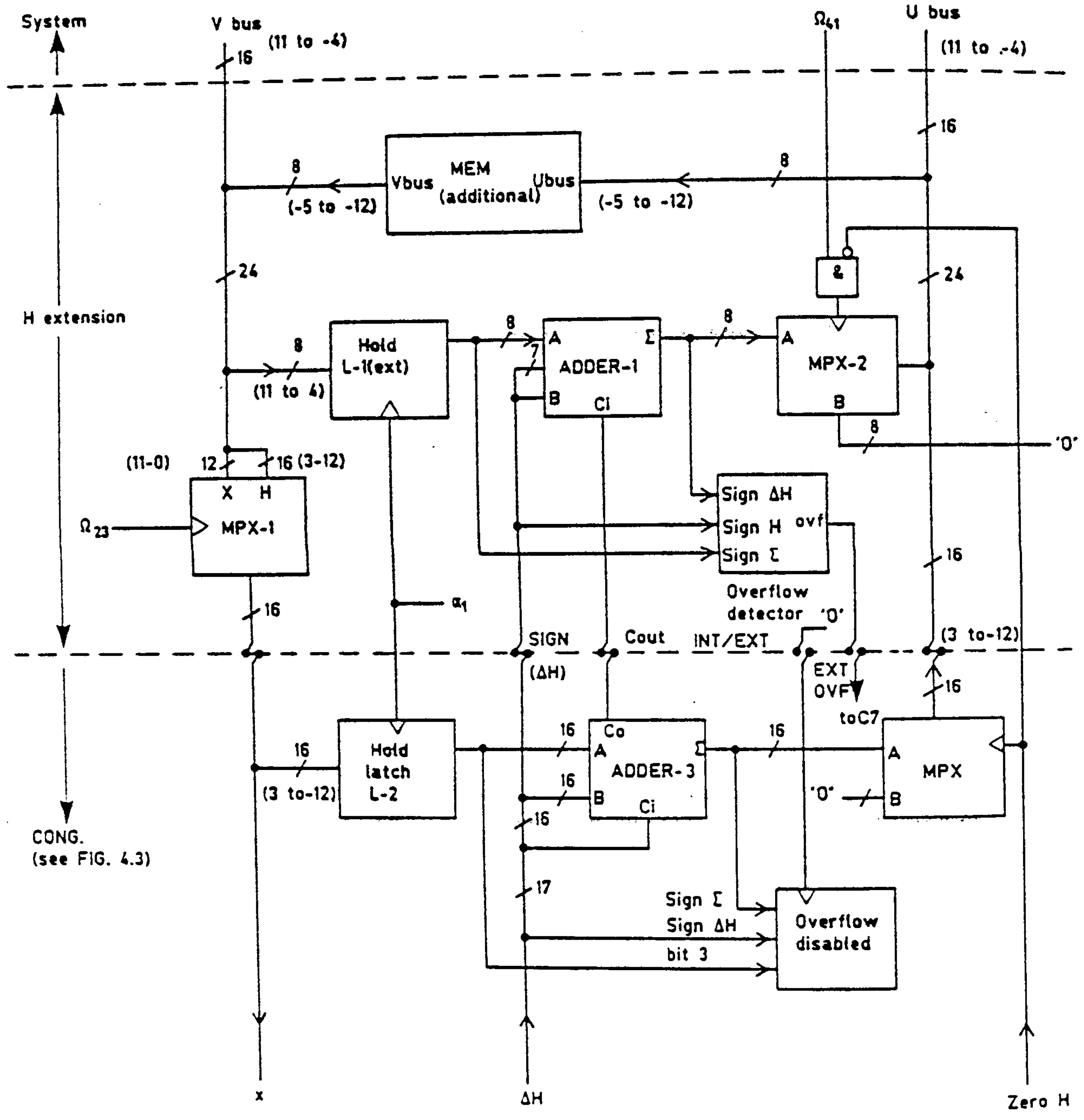


FIG. B.20 CIRCUIT CONFIGURATION FOR AN 8-BIT H EXTENSION

reset and the process is repeated. Figure 3.8 is a typical oscillogram using the H monitor.

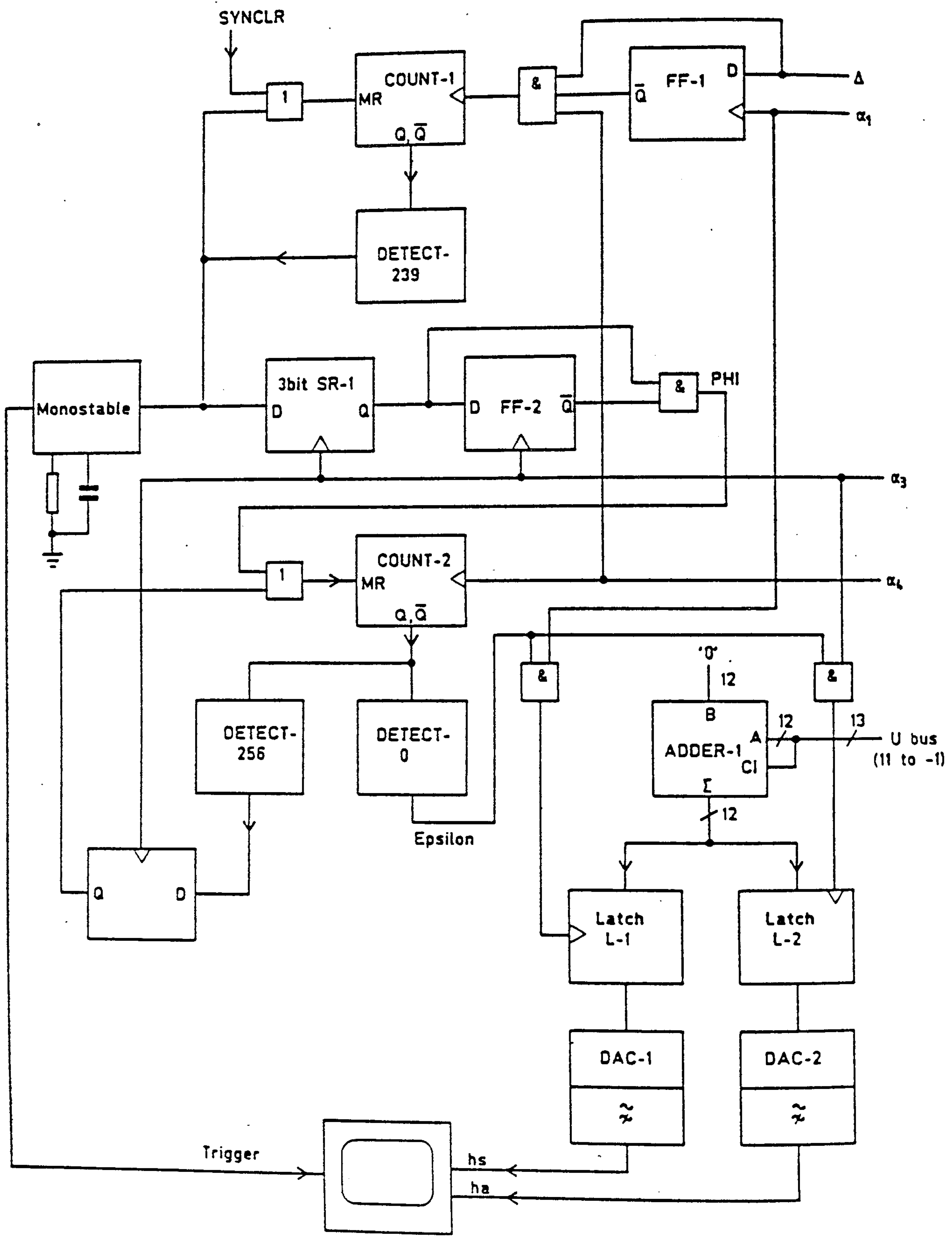


FIG. B.21 SCHEMATIC OF H MONITOR

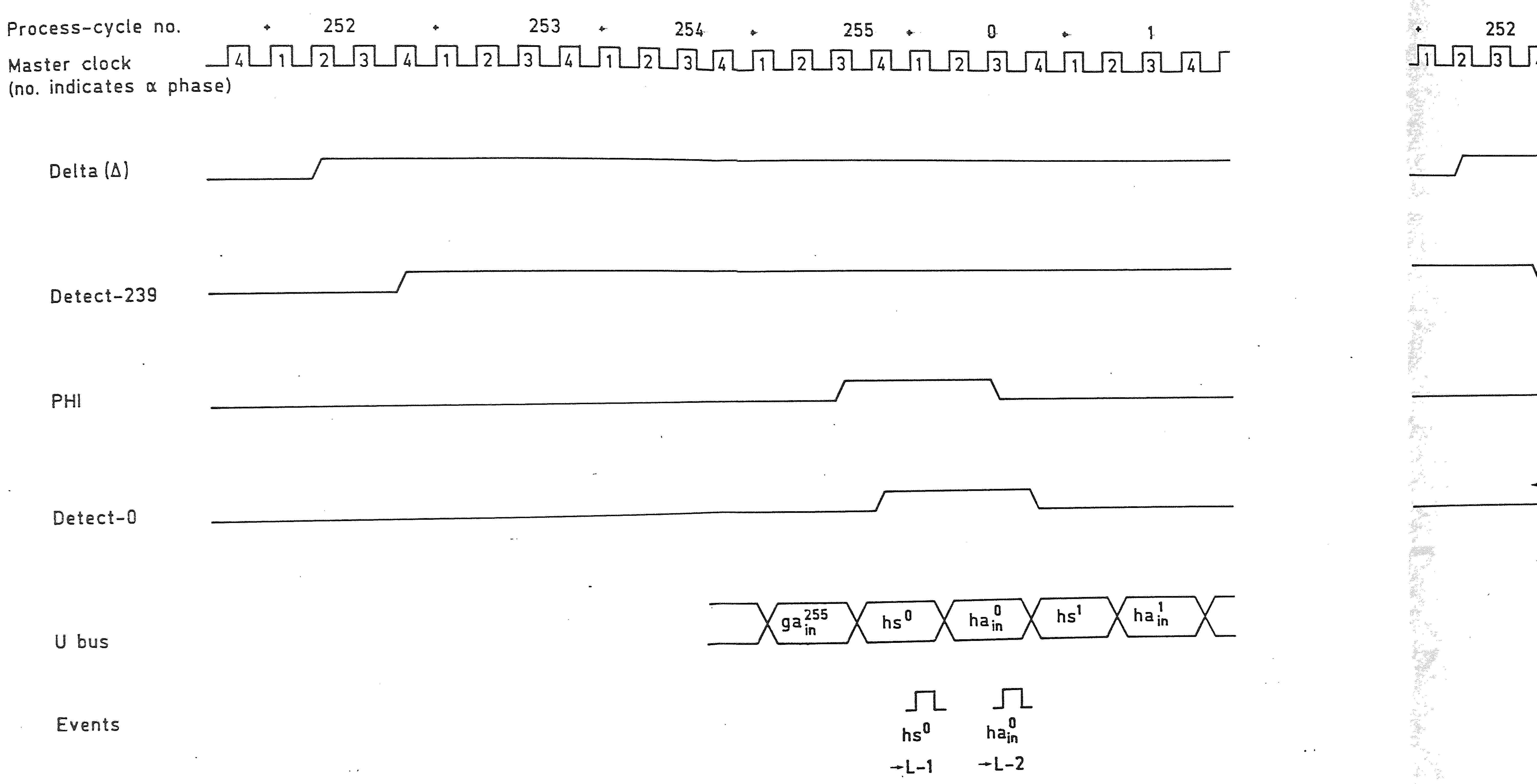


FIG. B.22 WAVEFORMS IN THE H MONIT

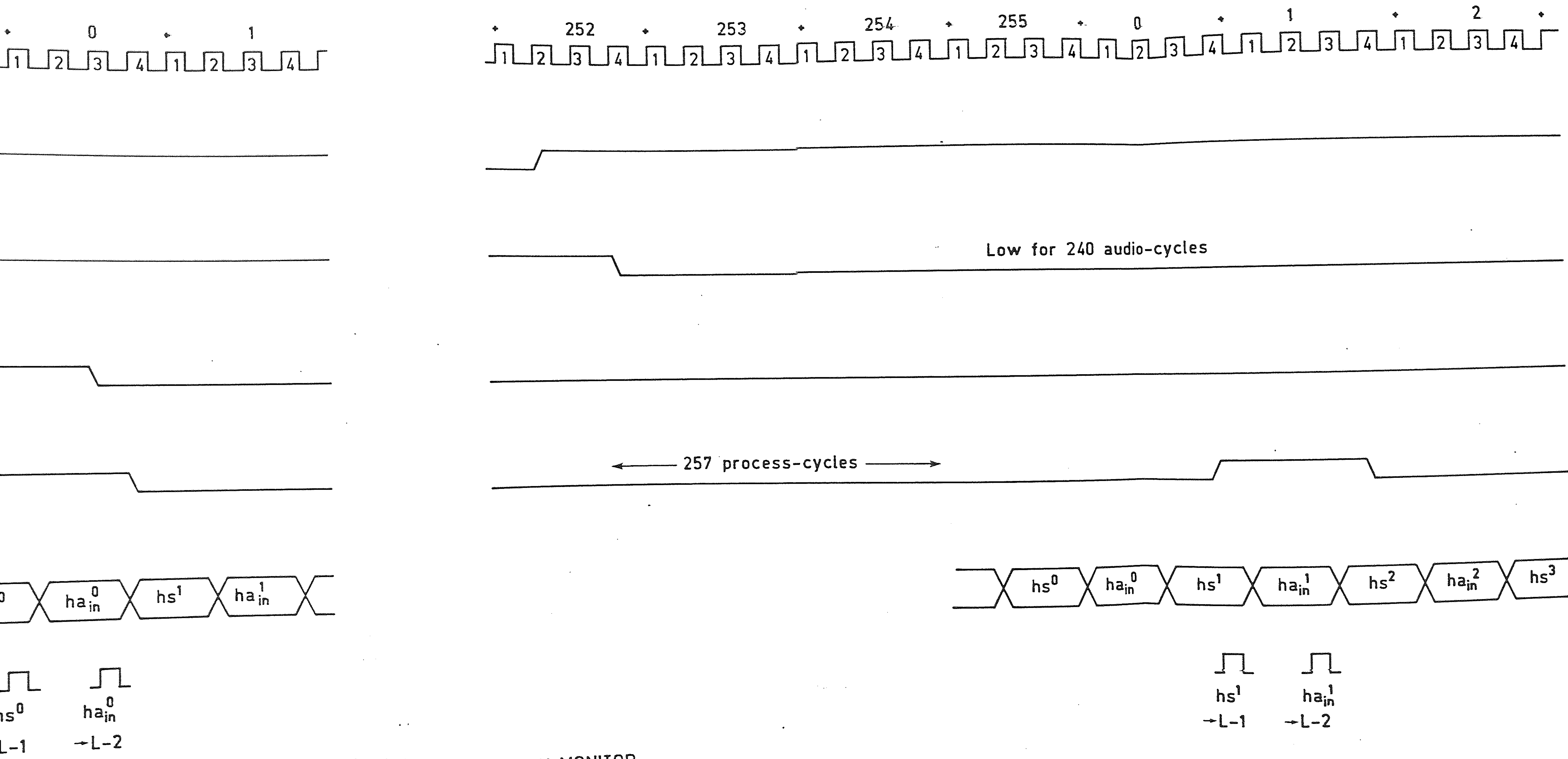


FIG. B.22 WAVEFORMS IN THE H MONITOR

## APPENDIX C.

Computer progam suite for performance calculation of an adaptive filter.

```
$DEBUG ON$
$HEAP DISPOSE ON$
PROGRAM x var MEASURE (inDut.outDut):
 (*************************************************************************)
 (* Prooram in HP Pascal. to run on a HP 200 series machine.              *)
 (* Measures the performance  of the svstem used in measurement           *)
 (* (see South's PhD submission). in terms of the coherent power          *)
 (* adaptation and total power adaptation for aiven inout parameters.     *)
 (* i.e. the program requests the level of inout noise in dBm iniected    *)
 (*into the echo path. the number of coefficients in the filter and the*)
 (* gain shift. corresponding to the value of k in theorv. The outout     *)
 (* is the cpa and tpa from the maximum inout sional level down to an     *)
 (* arbitarv value of -24 dBm.                                            *)
 (*************************************************************************)

 IMPORT          IODECLARATIONS.
                 GENERAL 2.
                 PLOT ROUTINES.
                 DGL LIB.
                 DGL TYPES.
                 MATH LIB:

 CONST
  form feed = CHR(12):  (* clears screen *)
  count max = 60: (* maximum number of measurements made *)
  offset = 0.0001: (* measure of closeness to zero *)

 TYPE
   variances = RECORD
   error. (*variance of total error*)
   psi.   (* component due to impulse response *)
   ad da.  (* quantisation noise in an ad da converter *)
   nt. (* due to finite precision of h in convolution*)
   x nt.  (* product of inout and delta h truncation noise *)
   na nt. (* component due a d and h  resolution *)
   misadiust.  (* i.e. iitter noise coherent with inout sia. *)
   replica trunc. (* quantisation noise from replica truncation *)
   x inout.       (* variance of inout sianal *)
   path noise.    (* var. of noise aenerated within path beina modelled*)
   t1 bus trunc. (* quantisation noise from truncation of 2*mu*error bus*)
   t2 bus trunc : REAL: (* ditto delta h bus *)
              END:

   iitter noise = RECORD  (* components of misadiustment as above *)
     external.            (* noise produced outside the filter    *)
     bus t1 component.    (* component due to 2*mu*error truncation*)
     bus t2 component  :REAL:   (* ditto delta h *)
                  END:

   svstem constants = RECORD (* variables within svstem architecture *)
     compensation.           (* max value compensation ranae on x inout*)
     aain.                   (* division factor from convolved replica*)
     t1 bus.                 (* ditto for 2*mu*error bus *)
     t2 bus :REAL:           (* ditto delta h bus         *)
                    END:
  a ranae =   -3..0:         (* sets run time limit on size of a factor*)
  c pwr ranae = 29..31:      (* ditto svstem compensation const *)
  coeffs ranae = 1..960:     (* ditto number of filter cefficients *)
  aain ranae = 11..12:       (* ditto aain shift numaber *)
  a select = 0..1:        (* operates a select in the filter *)
  counter = 0..count max:
```

```
  tables = RECORD           (* output table for calculated results *)
    x input :REAL:
    a cp .
    a sp .
    a tp : ARRAY[-3..0] OF REAL: END:
  jitters = RECORD           (* record of jitter noise components *)
    external.
    t1 bus.
    t2 bus : ARRAY[-3..0] OF REAL: END:

  noises measurements = RECORD
    x input dBm.
    psi.
    ad da.
    replica trunc.
    x nt.
    na nt.
    path noise.
    t1 bus trunc.
    t2 bus trunc.
    misadjust path noise : ARRAY[-3..0.counter] OF REAL
    END:


VAR
  x in dBm .          (* input level measured in dBm           *)
  path noise in.      (* noise generated in path model in dBm *)
  max x dBm.          (* system limit on max input mean level *)
  signal to noise.(* ratio in dB                               *)
  advantage.          (* component which reduces the jitter noise *)
  gamma.              (* vvariable used for brevity. gain on X length*)
  system 2 mu.        (* variable to hold the calculated 2u value *)
  convergence.        (* ratio of desired signal to misadjustment*)
  estimate of mean H.  (* this is a measureable number or an estimate
                          of the size of the rms of H as a signal *)
  value.
  index.
  x rms.
  bit resolution.  (* resolution of ad da conversion - nominal 1 bit*)
  h resolution (* from no of bits used in convolution*)
                   : REAL:
  run flag         : BOOLEAN:
  variance         : variances:
  jitter contrib : jitter noise:
  shift cntl       :system constants:
  a gain shift     : a select:  (* pwr of 2 shift to switch a gain *)
  jitter           : jitters:
  noise measurements : noises measurements:
  table            : ARRAY[1..count max] OF tables:
  x counter : counter:
  n.                      (* general purpose integer variable *)
  x max.
  h bits in convolution.
  total h bits (* bits in standard system *) : INTEGER:

  a shift range : a range:    (* hardware limitation on a shift as pwr 2 *)
  no of coeffs :coeffs range: (* total number of tap coeffs in filter *)
  c pwr 2 :c pwr range: (* const used for compensation as pwr of 2 *)
  gain pwr 2 :gain range:(* shift right for replica select from convolution *)
  delta h offset.    (* extension of a shift facility in bits *)
```

```
   a factor .          (* integration constant as a pwr of 2    *)
   two mu err shift.  (* variable to hold shift code i/p for nominal setting*)
   t1 pwr 2.            (* shift right no. for 2.mu.error bus        *)
   t2 pwr.2 : INTEGER:  (* shift right no. for delta H bus          *)

  device : TYPE DEVICE:
  text line : STRING[255]:
  position  : INTEGER:
  left 9 margin.   (* escape sequence for 2673A printer *)
  left 1 margin.
  expand text.   (* ESC. sequence for expanded text *)
  normal text : STRING[5]:

   init ok.
   plot ok  : BOOLEAN:
   x input level.
   fn of x : real array:
   x first.
   x deltav.
   v first.
   v deltav : REAL:
   device switch : CHAR:
   sub title : ARRAY[1..10] OF titles:
   main title: titles:
   count.
   pen.
   plot no : INTEGER:
   check ok  : CHAR:
   temp store : REAL:
   stability ratio : REAL:

   (****************************************************************)


FUNCTION two mu factor(svs const.no of coeffs.rms of input : REAL):REAL:

(*forms the 2*mu/a factor for the adaptive filter compensation range *)
(* svs const is the maximum power of 2 value calculated for the upper
level of input signal. NB compensation is set to 36dB in this routine *)

 VAR
   log2.  (* will hold the log of 2 *)
   temp store.
   result.
   indicie (* raised to the power of *)
          : REAL:
   switch : INTEGER:

BEGIN (* two mu factor *)

  IF rms of input > 0.0 THEN
   BEGIN
    log2 := LN(2.0):
   (* calc the index to a base 2 of the input signal vector length *)
   temp store := LN(no of coeffs * SQR(rms of input))/log2:

   indicie := svs const - temp store:

   (* provide non-linear truncation as hardware upper end *)
   IF indicie <= 0.0 THEN switch := 1:
```

```
  (*   truncation of indicie emulates precision in hardware *)
  indicie := TRUNC(indicie):

  result := exponentiate(2.0.indicie):

  (* provide non-linear truncation to match hardware lower end*)

  IF result > 1024.0 THEN switch := 2:
  result := ROUND(result):
  (* output value *)
  CASE switch OF
   1: two mu factor := 0.0:
   2: two mu factor := 2047.0:
  OTHERWISE
    two mu factor := result:
  END: (* end case *)
 END (* correct calculation for valid input level *)

 ELSE
  BEGIN
   IF rms of input = 0.0 THEN two mu factor := 2047.0
   ELSE
    BEGIN
      WRITELN('Negative value of input level in TWO MU FACTOR routine'):
      WRITELN('value substituted = 2047 program continues.!!!'):
      two mu factor := 2047.0:
    END:
 END: (* end zero or incorrect input level *)

END: (* end function two mu factor *)

(**********************************************************************)

FUNCTION convert dbm to variance(level in dBm: REAL) : REAL:
  (* takes in a level in dBm and converts to a variance value *)
  (* assuming an AD DA signal range of 10 volts corresponding *)
  (* to 2047 levels *)

  VAR
   index.
   result : REAL:

  BEGIN (* convert dBm to variance*)
     index := (level in dBm -2.22)/10.0:
     result := exponentiate(10.0.index) * SQR(204.7):
     convert dBm to variance := result:
  END: (* ends convert dBm to variance *)

(***********************************************************************)

(************************************************************************)

PROCEDURE initialise(no of coeffs : coeffs range:g factor : INTEGER:
            c pwr 2 : c pwr range:a gain shift : g select:two mu err shift :
            INTEGER: VAR t1 pwr 2.t2 pwr 2 : INTEGER:VAR max x dBm : REAL):
 TYPE
   cascade = 0..2:  (* cascade level found from No. of coeffs *)
   bit shift = INTEGER:
 VAR
   cas level : cascade:
```

```
nominal shift.  (* total shift for minimal svstem *)
t1 nominal.     (* 2*mu*err shift    ditto        *)
t2 nominal.     (* delta H shift     ditto        *)
gain nominal : bit shift:(* nominal settina for a shift *)
a shift advantage : REAL:

 (* sets up initial conditions for solution of formula *)
 (* usina all alobal variables *)
BEGIN   (* initialise *)

  (* the number of coeffs sets a limit on the compensation range *)
  (* which in turn sets the maximum allowable inout sianal level *)
  IF (no of coeffs  >= 1) AND (no of coeffs <= 240) THEN
           cas level := 0:
  IF (no of coeffs >= 241) AND (no of coeffs <= 480) THEN
           cas level := 1:
  IF (no of coeffs >= 481) AND (no of coeffs <= 960) THEN
           cas level := 2:
  (***************************************************************)
  (* the following parameters are calculated based on the chosen *)
  (* values of t1 and t2 for the desianed svstem and must be     *)
  (* altered for chanaes in the proportionalitv between the two  *)
  (* based on a svstem constant of c pwr 2 defined in program    *)
  (***************************************************************)

  c pwr 2 := c pwr 2 + cas level:(* automatic compenstion of C when
                                    cascadina. is done bv the filter*)
  gain nominal := 11: (*minimum useful value limited bv a->d range*)
  nominal shift := c pwr 2 - gain nominal:
                              (* sum of t1 and t2 nominal shifts *)
  t1 nominal := two mu err shift:  (* value in svstem *)
  t2 nominal := nominal shift - t1 nominal:

  gain pwr 2 := gain nominal + a gain shift:
  t1 pwr 2 := gain nominal + t1 nominal - gain pwr 2:
  t2 pwr 2 := t2 nominal - a factor + cas level:

(***************************************************************)

  max x dBm := 10.0 * log10(exponentiate(2.0.c pwr 2)/no of coeffs)
               - 20.0 * log10(204.7) + 2.22:

  WITH shift cntl DO
   BEGIN
    compensation := exponentiate(2.0.c pwr 2):
    gain := exponentiate(2.0.gain pwr 2):
    t1 bus := exponentiate(2.0.t1 pwr 2):
    t2 bus := exponentiate(2.0.t2 pwr 2)
   END:

  WITH variance DO
   BEGIN
    ad da := SQR(bit resolution)/12.0:
                          (* variance of lsb for ad-da converter *)
    psi := no of coeffs * SQR(estimate of mean H/shift cntl.gain)
             * variance.ad da:
    replica trunc := exponentiate(2.0.cas level)/(12.0):
    path noise := convert dBm to variance(path noise in):
    t1 bus trunc := 1.0/12.0: (* error is due to lsb = +/- 0.5 bit*)
    t2 bus trunc :=
             exponentiate(2.(2*(12-total h bits-delta h offset)))/12.0:
                  (* this uses a supplied N bit word for H in the system*)
     error := 0.0
    END;

  END; (* end intialise *)
(*************************************************************************)
(*************************************************************************)
BEGIN (* test main program *)
  PROMPT('Enter Crt or Printer for output ==>');
  READLN(device switch);
  IF (device switch = 'c') OR (device switch = 'C') THEN
   device := 1
   ELSE
   device := 701;
  left 9 margin := CHR(27)+CHR(38)+CHR(97)+CHR(57)+CHR(76);
  left 1 margin := CHR(27)+CHR(38)+CHR(97)+CHR(49)+CHR(76);
  expand text   := CHR(27)+CHR(38)+CHR(107)+CHR(49)+CHR(83);
  normal text   := CHR(27)+CHR(38)+CHR(107)+CHR(48)+CHR(83);
  WRITESTRING(device,left 1 margin);
  WRITE('Enter path noise in dBm ==> ');
  READLN(path noise in);
  WRITE('Enter the number of coefficients (range 1 to 960) ==> ');
  READLN(no of coeffs);
  WRITE('Enter a select 0 or 1 ==> ');
  READLN(a gain shift);
  WRITE('Estimate the rms value of ''h'' ==> ');
  READLN(estimate of mean h);
  WRITE('Enter the resolution of A-D, D-A conversion ==> ');
  READLN(bit resolution);
  run flag := TRUE;
  WHILE run flag = TRUE DO
  BEGIN
   WRITE('Enter the No. of bits of H used in the storage ==> ');
   READLN(total h bits);
   WRITE('Enter the No. of bits of H used in the convolution ==> ');
   READLN(h bits in convolution);
   IF h bits in convolution > total h bits THEN
    BEGIN
     WRITELN('More bits in convolution than store. Try again !!')
    END
     ELSE run flag := FALSE;
  END;  (* end WHILE - DO *)
  WRITE('Enter nominal shift for t1 bus ==> ');
  READLN(two mu err shift);
   delta h offset := 0;  (* initial value for no H extension *)
   WRITE('Enter No. of bits for H extension ==> ');
   READLN(delta h offset);
   c pwr 2 := 29; (* system constant *)
   h resolution := 12 - h bits in convolution; (* since we use a 12 bit
     system then the resolution in convolution is given as a pwr of 2 *)

   FOR a shift range := 0 DOWNTO -3 DO
    BEGIN
    a factor := - delta h offset + a shift range;
    initialise(no of coeffs,a factor,c pwr 2,a gain shift,two mu err shift,
                                  t1 pwr 2,t2 pwr 2,max x dBm);
```

```
WRITESTRING(device.form feed):                       (*advances paper*)
WRITESTRING(device.left 9 margin):   (* set left margin 9 cols *)
WRITESTRING(device.expand text):
WRITESTRINGLN(device.'Adaptation over X level variation.'):
WRITESTRINGLN(device.'for analogue input - Measurement.'):
WRITESTRING(device.normal text):
WRITESTRINGLN(device.''):
WRITESTRINGLN(device.
       'Nomenclature :- Coherent Power Adaptation - ''cp'''):
WRITESTRINGLN(device.
       '                Svstem Power Adaptation   - ''sp'''):
WRITESTRINGLN(device.
       '                Total Power Adaptation    - ''tp'''):
STRWRITE(text line.1.position.'a shift = '.a factor:3.
       ' gain shift = '.a gain shift:2.
       '  H extension = '.delta h offset:2.
       ' No. of coeff''s = '.no of coeffs:4):
SETSTRLEN(text line.position-1):
WRITESTRINGLN(device.text line):
WRITESTRINGLN(device.''):
STRWRITE(text line.1.position.
       'Echo path noise = '.path noise in:4:1.' dBm'.
       '       Estimate of rms of h = '.estimate of mean h:4:1):
SETSTRLEN(text line.position-1):
WRITESTRINGLN(device.text line):
STRWRITE(text line.1.position.
                        'No. of bits in h = '.total h bits:3.
 '       No. of bits used in convolution = '.h bits in convolution:3):
SETSTRLEN(text line.position-1):
WRITESTRINGLN(device.text line):
STRWRITE(text line.1.position.
                        'A-D resolution = '.bit resolution:4:1.' bits'.
 '       t1 shift = '.t1 pwr 2:3.'     t2 shift = '.t2 pwr 2:2):
SETSTRLEN(text line.position-1):
WRITESTRINGLN(device.text line):
WRITESTRINGLN(device.''):
WRITESTRINGLN(device.
 '  X dBm       Adaptation            Variance of Misadjustment   Stab.'):
WRITESTRINGLN(device.
 '           cp     sp     tp        ext. noise    t1 bus    t2 bus   ratio'):
WRITESTRINGLN(device.''):

x in dBm := TRUNC(max x dBm): (* initial value for x in *)
run flag := TRUE:
x counter := 0:
WHILE run flag DO
 BEGIN
 x counter := x counter + 1:
                        (* gives total no of measurements to count max*)

 x rms := SQRT(convert dBm to variance(x in dBm)):


 svstem 2 mu := - two mu factor(c pwr 2.no of coeffs.x rms):

 variance.x input := SQR(x rms):
 (* calc the gain function of X length - gamma *)
 variance.nt :=(exponentiate(2.(2*(12-h bits in convolution)))/12.0) :
 variance.x nt := no of coeffs * variance.x input *
                                  variance.nt/SQR(shift cntl.gain):
 variance.na nt := no of coeffs*variance.ad da*
```

```
                                   variance.nt/SQR(shift cntl.gain):
      gamma := no of coeffs*variance.x input*
                                    (1.0 + variance.ad da/variance.x input):
        (* gamma is slightly different than that quoted in the theory *)
        (* as it combines W.sigma(x)^2 ie the vector norm.             *)

            (* calc system advantage see theory *)
       advantage := 1.0 + (2.0 * shift cntl.gain * shift cntl.t1 bus *
       shift cntl.t2 bus)/(gamma * system 2 mu):
(*****************************************************************)
(* compute the jitter contributions of each component in the misadjustment *)
     jitter contrib.external:= - (variance.psi + variance.ad da
                           + variance.replica trunc + variance.path noise
                           + variance.x nt + variance.na nt )/advantage:
     jitter contrib.bus t1 component:= - variance.t1 bus trunc *
                        SQR(shift cntl.t1 bus)/SQR(system 2 mu)/advantage:
     jitter contrib.bus t2 component:=
       - no of coeffs * variance.t2 bus trunc * SQR(shift cntl.t1 bus)
       * SQR(shift cntl.t2 bus) / (SQR(system 2 mu) * gamma)/advantage:
       (*******************************************************************)
     (* compute the misadjustment *)
     variance.misadjust := jitter contrib.external
                             + jitter contrib.bus t1 component
                             + jitter contrib.bus t2 component:
     table[x counter].g cp[g shift range] :=
                              10.0 * log10(variance.x input/variance.misadjust):

   (*****************************************************************)
     stability ratio := jitter contrib.external/variance.misadjust:
(* the stability ratio must not exceed 1.0 or the present system
 becomes unstable *)
     IF stability ratio >= 1.0 THEN
     WRITELN('System is unstable reset g or reduce noise in reference path!'):
     (*compute the system variance of error *)
     variance.error := variance.misadjust + 2.0*variance.ad da + variance.psi +
     variance.na nt + variance.replica trunc + variance.x nt :
     (* calc the system performance with the external noise removed *)
     table[x counter].g sp[g shift range] := 10.0 *
                                          log10(variance.x input/variance.error):
(*****************************************************************)
(* calc the total variance of error which includes the a-d noise in o/p *)
     variance.error := variance.error + variance.path noise :
     table[x counter].g tp[g shift range] := 10.0 *
                                          log10(variance.x input/variance.error):

     WITH noise measurements DO
       BEGIN
        x input dBm[g shift range.x counter] := x in dBm:
        psi[g shift range.x counter] := variance.psi/variance.error * 100.0:
        ad da[g shift range.x counter] :=
                                     2.0*variance.ad da/variance.error * 100.0:
        path noise[g shift range.x counter] :=
                                      variance.path noise/variance.error * 100:
        replica trunc[g shift range.x counter] :=
                                 variance.replica trunc/variance.error * 100.0:
        x nt[g shift range.x counter] := variance.x nt/variance.error * 100:
        na nt[g shift range.x counter] := variance.na nt/variance.error * 100:
        t1 bus trunc[g shift range.x counter] :=
                         jitter contrib.bus t1 component/variance.error*100.0:
        t2 bus trunc[g shift range.x counter] :=
                         jitter contrib.bus t2 component/variance.error * 100:
```

```
        misadjust path noise[q shift range.x counter] :=
        END:                            jitter contrib.external/variance.error*100.0:

(************************************************************)
        STRWRITE(text line.1.position.
        ' '.x in dBm:5:1.
        '   '.table[x counter].q cp[q shift range]:4:1.
        '   '.table[x counter].q sp[q shift range]:4:1.
        '     '.table[x counter].q tp[q shift range]:4:1.
        '     '.jitter contrib.external:6.
        '   '.jitter contrib.bus t1 component:6.
        '   '.jitter contrib.bus t2 component:6.
        '  '.stability ratio:3:1):
        SETSTRLEN(text line.position-1):
        WRITESTRINGLN(device.text line):
        x input level[x counter] := x in dBm: (* array for plotting *)
        x in dBm := x in dBm - 1.0:
      IF x in dBm < -24.0 THEN
       BEGIN
        x max := x counter: (* find max number of measurements made *)
        run flag := FALSE:
       END:
      END: (*ends input sequence *)
    END: (* FOR q shift range DO *)

  (* output a table of the percentage error contributions from each
                                               of the contributing noises *)
  t2 pwr 2 := t2 pwr 2 - 3:          (* restore nominal t2 setting for Q=0*)

  FOR q shift range := 0 DOWNTO -3 DO
   BEGIN
    WRITESTRING(device.form feed):              (*advances paper*)
    WRITESTRING(device.expand text):
    WRITESTRINGLN(device.'Percentage composition of noise.'):
    WRITESTRINGLN(device.'for analogue input - measurement.'):
    WRITESTRING(device.normal text):
    WRITESTRINGLN(device.''):
    STRWRITE(text line.1.position.'q shift = '.q shift range:3.
          ' gain shift = '.a gain shift:2.
          '  H extension = '.delta h offset:2.
          ' No. of coeff''s = '.no of coeffs:4):
    SETSTRLEN(text line.position-1):
    WRITESTRINGLN(device.text line):
    WRITESTRINGLN(device.''):
    STRWRITE(text line.1.position.
          'Echo path noise = '.path noise in:4:1.' dBm'.
          '        Estimate of rms of h = '.estimate of mean h:4:1):
    SETSTRLEN(text line.position-1):
    WRITESTRINGLN(device.text line):
    WRITESTRINGLN(device.''):
    STRWRITE(text line.1.position.
                          'No. of bits in h = '.total h bits:3.
    '       No. of bits used in convolution = '.h bits in convolution:3):
    SETSTRLEN(text line.position-1):
    WRITESTRINGLN(device.text line):
    WRITESTRINGLN(device.''):
    STRWRITE(text line.1.position.
                          'A-D resolution = '.bit resolution:4:1.' bits'.
    '       t1 shift = '.t1 pwr 2:3.
                          '    t2 shift = '.t2 pwr 2-q shift range:2):
```

```
      SETSTRLEN(text line.position-1):
      WRITESTRINGLN(device.text line):
      WRITESTRINGLN(device.''):
      STRWRITE(text line.1.position.
     .'                    Percentage of total noise'):
      SETSTRLEN(text line.position-1):
      WRITESTRINGLN(device.text line):
      WRITESTRINGLN(device.''):
      WRITESTRINGLN(device.
      'x dBm.  Psi    x nt   na nt   ad-da   path relica    ext''l   t1      t2'):
      WRITESTRINGLN(device.''):
         FOR x counter := 1 TO x max DO
         BEGIN
          STRWRITE(text line.1.position.
          noise measurements.x input dBm[a shift range.x counter]:5:1.'   '.
          noise measurements.psi[a shift range.x counter]:4:1.'   '.
          noise measurements.x nt[a shift range.x counter]:4:1.'   '.
          noise measurements.na nt[a shift range.x counter]:4:1.'   '.
          noise measurements.ad da[a shift range.x counter]:4:1.'   '.
          noise measurements.path noise[a shift range.x counter]:4:1.'   '.
          noise measurements.replica trunc[a shift range.x counter]:4:1.'   '.
          noise measurements.misadjust path noise[a shift range.x counter]:4:1.'
          noise measurements.t1 bus trunc[a shift range.x counter]:4:1.'   '.
          noise measurements.t2 bus trunc[a shift range.x counter]:4:1):
          SETSTRLEN(text line.position-1):
          WRITESTRINGLN(device.text line):
         END:
     END:

  [illegible]
  (* plot results *)
  STRWRITE(sub title[1].1.position.'Path noise = '.path noise in:5:1.' dBm'):
  SETSTRLEN(sub title[1].position-1):
  STRWRITE(sub title[2].1.position.'No. of Coeff''s. = '.no of coeffs:1):
  SETSTRLEN(sub title[2].position-1):
  STRWRITE(sub title[3].1.position.'K factor = '.a gain shift:1):
  SETSTRLEN(sub title[3].position-1):
  STRWRITE(sub title[4].1.position.'t1 = '.t1 pwr 2:1):
  SETSTRLEN(sub title[4].position-1):
  STRWRITE(sub title[5].1.position.
                                   'Est. of mean h = '.estimate of mean h:4:1):
  SETSTRLEN(sub title[5].position-1):
  STRWRITE(sub title[6].1.position.'A-D resolution = '.bit resolution:3:1):
  SETSTRLEN(sub title[6].position-1):
  STRWRITE(sub title[7].1.position.'Conv bits = '.h bits in convolution:3):
  SETSTRLEN(sub title[7].position-1):
  STRWRITE(sub title[8].1.position.'H bits = '.total h bits+delta h offset:3):
  SETSTRLEN(sub title[8].position-1):

  plot no := 1:
  plot ok := TRUE:

   setup example (init ok):
    IF init ok THEN
     BEGIN
      WHILE plot ok DO
      BEGIN
       pen := 1:
       window(296.0.210.0): (* draw A4 format *)
       origin(20.0.20.0):
       x first := -24.0:
```

```
x deltav := 2.0:
axis(0.0.0.0.'Inout Level in dBm.'.19.220.0.0.x first.x deltav):
v first := 24:
v deltav := 2:
WRITE('Enter v first ==. '):
READLN(v first):
WRITE('Enter delta v ==> '):
READLN(v deltav):
axis(0.0.0.0.'Adaptation in dB''s.'.-19.160.90.0.v first.v deltav):
FOR a shift ranae := 0 DOWNTO -3 DO
 BEGIN
  FOR count := 1 TO x max DO
   CASE plot no OF
    1:fn of x[count] := table[count].a sp[a shift ranae]:
    2:fn of x[count] := table[count].a cp[a shift ranae]:
   END:
  plot(x inout level.x first.x deltav.fn of x.v first.v deltav.x max.1):
  STRWRITE(sub title[10].1.position.
                              'Q  = '.a shift ranae-delta h offset:2):
  SETSTRLEN(sub title[10].position-1):
  svmbol(240.70-(6*ABS(a shift ranae)).4.0.sub title[10].0.0.255):
  pen := pen + 1:
  SET COLOR(pen):
 END:
pen := 1:
SET COLOR(1):
 CASE plot no OF
 1: main title := 'Svstem Adaptation':
 2: main title := 'Coherent Adapation':
 END:
svmbol(30.170.6.0.main title.0.0.255):
svmbol(32.164.4.0.'(Measurement Setup)'.0.0.19):
FOR count := 1 TO 8 DO
svmbol(225.176 - (6*count).3.0.sub title[count].0.0.255):
WRITELN('Chanae paper on plotter and tvpe C'):
READLN(check ok):
IF (check ok = 'C') OR (check ok = 'c') THEN plot ok := TRUE
ELSE plot ok := FALSE:
plot no := plot no +1:
IF plot no > 2 THEN plot ok := FALSE:
END:  (* ends the plot routine *)
END:
GRAPHICS TERM:

END. (* end main test program *)
```

```
$DEBUG ON.LINES 20000.SYSPROG ON$
PROGRAM test int (INPUT.OUTPUT):
(* To calculate the quantisation noise for the SHL algorithm *)
(* using the companding shown in figure 5.20. and produces    *)
(* output which is shown in figure  5.21                       *)
  (**********************************************************)

 MODULE integrate:
 EXPORT

  CONST
   upper bound = 1025:

  TYPE
  vector = ARRAY[1..upper bound] OF REAL:

  PROCEDURE qsf (h : REAL:v vector:vector :VAR z vector
                :vector :vector length : INTEGER: VAR error : CHAR):

    IMPLEMENT

    PROCEDURE qsf (h : REAL:v vector:vector :VAR z vector
                :vector :vector length : INTEGER: VAR error : CHAR):
    (**********************************************************)
    (* Integration of an equidistantly tabulated function   *)
    (*  by Simpson's Rule                                   *)
    (**********************************************************)

  VAR
    aux.
    sum 1.
    sum 2.
    hh.
    f1.
    f2 : REAL:
    i  : INTEGER:


  BEGIN
    error := '1':
    IF vector length >= 4 THEN
     BEGIN
      error := '0':
      hh := h/3.0:
      f1 := v vector[1]:
      f2 := v vector[2]:
      sum 1 := 0.0:
      z vector[1] := 0.0:
      sum 2 := hh*0.125*(9.0*f1 + 19.0*f2 -5.0*v vector[3] + v vector[4]):
      z vector[2] := sum 2: (* compute z vector[2] by combination of
                               Simpson's Rule and Newtons 3/8 rule *)

       FOR i := 3 TO vector length DO
        BEGIN
         aux := f2 + f2:
         aux := aux + aux + f1:
         f1 := f2:
         f2 := v vector[i]:
         TRY
         aux := hh*(aux + f2):
         RECOVER
        IF ESCAPECODE = -7 THEN
         BEGIN
          aux := 0.0:
         END:
        sum 1 := sum 1 + aux: (* accumulate intreqral value *)
        aux := sum 1:
        z vector[i] := sum 1:
        sum 1 := sum 2:
        sum 2 := aux:
       END:
     END:
  END:   (* END asf *)
END:
   (***************************************************************)


MODULE dbm to variance:

IMPORT math lib:

EXPORT
  FUNCTION convert dbm to variance(level in dbm : REAL) : REAL:

  IMPLEMENT

  FUNCTION convert dbm to variance(level in dbm : REAL) : REAL:
   (****************************************************************)
   (* takes in a level in dBm and converts to a variance value *)
   (* assuming an AD-DA sional ranae of 10 volts. correspondina*)
   (* to 2047 levels.                                           *)
   (****************************************************************)


   VAR
    index.
    result : REAL:

    BEGIN
     (* convert dbm to variance *)
     index := (level in dbm -2.22)/10.0:
     result := exponentiate(10.0.index) * SQR(204.7):
     convert dbm to variance := result:
    END:
END:

MODULE pcm noise:

 IMPORT inteqrate.
        dbm to variance.
        MATH LIB:

 EXPORT
 PROCEDURE quant shl sn(input level : REAL: VAR q sional to noise : REAL):

 IMPLEMENT

 PROCEDURE quant shl sn(input level : REAL: VAR q sional to noise : REAL):
 (********************************************************************)
 (* calculates the quantisation sional-to-noise ratio for the SHL      *)
(* alaorithm. assuming Gaussian noise inout. See PhD for compandina     *)
(****************************************************************************)
CONST
    max = 1025:

  VAR

   z flaa : BOOLEAN:
   index     : 1..max:
   i.j.
   k.
   l bound.
   u bound.
   sector count          : INTEGER:
   error     : CHAR:
   v vector : vector:
   z vector : vector:
   v var.
   h var.
   std dev.
   x increment.
   inout variance.
   result.
   noise.
   delta i.   (* variable width of inout numbers *)
   sianal to noise.
   partial prob.
   normalisation   : REAL:
   sector probabilitv : ARRAY[1..11] OF REAL: (* there are 11 sectors
      in the auantisation produced bv SHL *)
   ei sauared : ARRAY[1..11] OF REAL:(* to hold the error sauared *)


FUNCTION erfc (x var.mean.std dev : REAL) : REAL:
(**************************************************************)
(* returns the value of the Gaussian variable                 *)
(**************************************************************)
VAR
 result.
 intermediate result : REAL:
BEGIN
 intermediate result :=(x var - mean)/std dev:
 intermediate result := intermediate result * intermediate result:
 TRY
 result := EXP(-intermediate result/2.0):
 RECOVER
 IF ESCAPECODE = -7 THEN result := 0.0
 ELSE ESCAPE(ESCAPECODE):
 erfc :=  result:
END:

BEGIN
   inout variance :=convert dbm to variance(inout level):
   std dev := SQRT(inout variance):
   (* construct the v vector *)
   (* calculate the orobabilitv of the first sector separatelv
     since it onlv has 3 values i.e. 0.1.2. which is less than
     the minimum for the asf procedure *)
   normalisation := std dev*2.506628275:
   partial prob := 0:
```

```
FOR i := 1 TO 11 DO sector probabilitv[i] := 0:
x increment := 0.5:
l bound := 0:
u bound := 2:
result := 0.0:
FOR i := 1 TO 5 DO
        v vector[i] := erfc(x increment*(i-1).0.std dev):
qsf(x increment.v vector.z vector.5.error):
sector probabilitv[1] := z vector[5]/normalisation:
  (*normalisation of the probabilitv  constant is the root of 2*pi *)
partial prob := sector probabilitv[1]:
ei squared[1] := 0: (* set first value to zero as no error occurs*)
x increment :=  1.0:
delta i := 1.0:
l bound := 2:  (* lower bound for sector range *)
u bound := 4: (*        ditto upper bound       *)
FOR sector count := 2 TO 11 DO
 (* integrate over each sector of quantiser to calc the probabilitv of *)
 (* signal occurence                                                    *)
 BEGIN
   z flag := FALSE:
   (* construct vector of probabilities*)
  FOR i := 1 to u bound - l bound DO v vector[i] := 0:
  i := l bound:
  REPEAT
  BEGIN
   i := i+1:
   v vector[i-l bound] := erfc(i-1.0.std dev)/normalisation:
   IF i = u bound THEN z flag := TRUE:
   IF v vector[i-l bound] < 1E-9 THEN z flag := TRUE:
   END:
   UNTIL z flag:
   FOR i:= 1 TO u bound-l bound DO
    BEGIN
     z vector[i] := v vector[i] :
     (* integrate probabilitv over the sector *)
     sector probabilitv[sector count] :=
           sector probabilitv[sector count] + z vector[i]:
    END:
   (* sum the noise contributions to each quantum *)
    k := 1:
    ei squared[sector count] := 0:
    FOR i := l bound+1 to u bound DO (* misses out the first zero value*)
     BEGIN (* calc the quantisation noise in each sector *)
      ei squared[sector count] := ei squared[sector count]
                                  + SQR(i-l bound)*z vector[k]:
     k := k+1: (* increment the z  vector *)
     END:
    partial prob:= partial prob + sector probabilitv[sector count]:
    delta i := 2.0 * delta i: (* increase the quantising bv 2.0 *)
    l bound := u bound:
    u bound := 2*u bound:
  END:
 noise := 0:
 FOR i := 2 TO 11 DO
 noise := noise + ei squared[i]:
(* times 2 to account for each half of the distribution *)
 noise := 2.0* noise:

signal to noise := 10*log10(input variance/noise):
q signal to noise := signal to noise:
   END:
END:

   IMPORT pcm noise:
   VAR
    input level.
    signal to noise : REAL:

BEGIN
 PROMPT('Enter the input level in dBm ==> '):
 READLN(input level):
 quant shl sn(input level.signal to noise):
 WRITELN('The quatistion signal-to-noise is '.signal to noise:5:2.' dB'):

END.
```

```
$DEBUG ON.LINES 2000.SYSPROG ON$
PROGRAM test norm (INPUT.OUTPUT):
(*To model the coherent noise component produced bv the SHL alaorithm
  bv aeneratina sianals from a random number aenerator. calculatina
  error term from the compandina shown in fiaure 5.20 and performina
  the convolution. the result is fiaure 5.21 *)
(*****************************************************************)

(* It is necessarv to aanerate a Gaussian random number with
   mean zero. and standard deviation of one *)
IMPORT
       RND.
       SYSGLOBALS.
       MATH LIB:
 VAR
  inout dbm.
  std dev.
  ranae.
  normal oo.
  machine oo.
  sum.
  variance.
  total noise.
  noise sianal.
  a noise sianal : REAL:
  forever.
  auit flaa : BOOLEAN:
  seed : INTEGER:
  i.
  i    : INTEGER:
  int outout.
  trunc val.
  sian : INTEGER:
  max oo : INTEGER:
  x inout : ARRAY[1..240] OF INTEGER:
  a noise : ARRAY[1..240] OF INTEGER:


PROCEDURE
    aaussian ( VAR resultant: REAL:VAR seed : INTEGER):

 (* Usina the direct method *)

 VAR
   x1.
   x2. (* normal deviates *)
   u1.
   u2. (*uniform deviates *)
   u3.
   v1.
   v2  (* also normal deviates when usina acceptance-rejection method *)
       : REAL:
   ranae : SHORTINT:
   accept : BOOLEAN:

   BEGIN
    ranae := 10000:
    accept := TRUE:
    WHILE accept DO
    BEGIN
     u1 := RAND(seed.ranae)/10000:
```

```
   IF u1 = 0.0 THEN u1 := RAND(seed.range)/10000:
   u2 := RAND(seed.range)/10000:
   x1 := SQR(u1):
   x2 := SQR(u2):
   IF SQR(2*u1-1)+SQR(2*u2-1) <= 1.0 THEN
   BEGIN
    u3 := RAND(seed.range)/10000:
    IF u3 = 0.0 THEN u3 := RAND(seed.range)/10000:
    accept := FALSE:
   END:
  END:
 x1 := SQR(u1):
 x2 := SQR(u2):
 TRY
 v1 := SQRT(-LN(u3))*(x1-x2)/(x1+x2):
 x2 := SQRT(-2*LN(u1))*SIN(6.283185308*v1):
 RECOVER
  IF ESCAPECODE = -16 THEN
   BEGIN
    WRITELN('u1 = '.u1.'  u3 = '.u3):
   END:
 resultant := x2:

END: (* end gaussian*)

   BEGIN
   PROMPT('Enter input level in dBm ==> '):
   READLN(input dbm):
   std dev := exponentiate(10.(input dbm-2.22)/20) * 204.7:
   max op := 2047:
   PROMPT('Enter a value for the seed ==> '):
   READLN(seed):
   sum := 0:
   variance := 0:
   FOR r := 1 TO 10 DO
    BEGIN
     total noise := 0:
     FOR i := 1 to 240 DO
     BEGIN
      gaussian(normal op.seed):
      machine op := std dev * normal op:
      int output := ROUND(machine op):
      IF ABS(int output) > max op THEN
       IF int output > 0 THEN int output := max op
       ELSE int output := -max op:
      x input[i] := int output:
     END:

    FOR i := 1 TO 240 DO
    BEGIN
     quit flag := TRUE:
     trunc val := 1024:
     WHILE quit flag DO
      BEGIN
       If x input[i] >= 0 THEN sign := 1
       ELSE sign := -1:
       IF ABS(x input[i]) >= trunc val THEN
        BEGIN
         q noise[i] := sign * (ABS(x input[i]) - trunc val):
         quit flag := FALSE:
        END:
```

```
    trunc val := trunc val DIV 2:
   END:
  total noise := x input[i]*q noise[i] + total noise:
 END:
  sum := sum + total noise:
  variance := variance + SQR(total noise):
WRITELN('Total noise = '.total noise.'  Sum = '.sum):
END:
 sum := sum/10:
variance := variance/10 - SQR(sum):
noise signal := sum/120.0/SQR(std dev):
q noise signal := 10*LOG10(1.0/noise signal):
WRITELN('Total noise = '.sum.'  std dev = '.SQRT(variance)):
WRITELN('Coherent quantisation noise = '.q noise signal):
END.
```

```
·MODULE pcm noise:

 IMPORT integrate.
        dbm to variance.
        MATH LIB:

 EXPORT
 PROCEDURE quantisation sn(input level : REAL: VAR q signal to noise : REAL):

 IMPLEMENT

 PROCEDURE quantisation sn(input level : REAL: VAR q signal to noise : REAL):
 (*************************************************************************)
 (* calculates the quantisation signal-to-noise ratio for an 8-bit pcm *)
 (* system. assuming Gaussian noise input and A-law companding        *)
 (***************************************************************************)

 CONST
     max = 1025:

   VAR

    index     : 1..max:
    i.
    l bound.
    u bound.
    sector count         : INTEGER:
    error     : CHAR:
    v vector : vector:
    z vector : vector:
    v var.
    h var.
    std dev.
    x increment.
    input variance.
    result.
    delta i.  (* variable width of input numbers *)
    signal to noise   : REAL:
    sector probability : ARRAY[1..7] OF REAL:

  FUNCTION erfc (x var.mean.std dev : REAL) : REAL:
  (*****************************************************************)
  (* returns the value of the Gaussian variable                   *)
  (*****************************************************************)
  VAR
   result.
   intermediate result : REAL:
  BEGIN
   intermediate result :=(x var - mean)/std dev:
   intermediate result := intermediate result * intermediate result:
   TRY
   result := EXP(-intermediate result/2.0):
   RECOVER
   IF ESCAPECODE = -7 THEN
    BEGIN
     result := 0.0:
    END
     ELSE ESCAPE(ESCAPECODE):
   erfc :=  result:
```

```
END:

BEGIN
   input variance :=convert dbm to variance(input level):
   std dev := SQRT(input variance):
   (* construct the v vector *)
   x increment :=  1.0:
   delta i := 1.0:
   l bound := 0:  (* lower bound for sector range *)
   u bound := 32: (*       ditto upper bound       *)
   result := 0.0:
   FOR sector count := 1 TO 7 DO
    (* integrate over each sector of pcm  to calc the probability of *)
    (* signal occurence                                               *)
    BEGIN
     FOR i := l bound+1 TO u bound+1 DO
      BEGIN
       v vector[i-l bound] := erfc(i-1.0.std dev):
      END:
                                              (* integrate over the sector *)
      qsf(x increment.v vector.z vector.u bound+1-l bound.error):
      sector probability[sector count] :=
                                   z vector[u bound+1-l bound]/(std dev*2.506628275):
                                              (* constant is the root of 2*pi *)
       result := result + SQR(delta i) * sector probability[sector count]:
       delta i := 2.0 * delta i: (* increase the quantising by 2.0 *)
       l bound := u bound:
       u bound := 2*u bound:
     END:
    result := result/6.0:
                         (* times 2 to account for each half of the distribution *)
   signal to noise := 10*log10(input variance/result):
   q signal to noise := signal to noise:

   END:
END:
```

```
MODULE correlatn noise;

IMPORT normal,
       MATH LIB;

 EXPORT

 PROCEDURE
         correlated pcm noise(input dBm : REAL;VAR q noise to signal : REAL);
 IMPLEMENT

 PROCEDURE
         correlated pcm noise(input dBm : REAL;VAR q noise to signal : REAL);

 VAR
  std dev,
  range,
  normal op,
  machine op,
  sum,
  variance,
  total noise,
  noise signal,
  q noise signal : REAL;
  forever,
  quit flag : BOOLEAN;
  seed : INTEGER;
  i,
  j,
  kk    : INTEGER;
  int output,
  trunc val,
  sign : INTEGER;
  max op : INTEGER;
  x input : ARRAY[1..240] OF INTEGER;
  q noise : ARRAY[1..240] OF INTEGER;
  pcm array : ARRAY[0..128] OF INTEGER;

    BEGIN
    FOR i := 0 TO 32 DO pcm array[i] := i;
    FOR i := 33 TO 48 DO pcm array[i] :=  2 + pcm array[i-1];
    FOR i := 49 TO 64 DO pcm array[i] := pcm array[i-1] +4;
    FOR i := 65 TO 80 DO pcm array[i] := pcm array[i-1] + 8;
    FOR i := 81 TO 96 DO pcm array[i] := pcm array[i-1] + 16;
    FOR i := 97 TO 112 DO pcm array[i] := pcm array[i-1] + 32;
    FOR i := 113 TO 128 DO pcm array[i] := pcm array[i-1] + 64;

    std dev := exponentiate(10,(input dbm-2.22)/20) * 204.7;
    max op := 2047;
    seed := ROUND((input dbm+3.14159)*std dev);
    sum := 0;
    variance := 0;
    FOR j := 1 TO 10 DO
     BEGIN
      total noise := 0;
      FOR i := 1 to 240 DO
      BEGIN
       gaussian(normal op,seed);
       machine op := std dev * normal op;
      int output := ROUND(machine op):
      IF ABS(int output) > max op THEN
       IF int output > 0 THEN int output := max op
       ELSE int output := -max op:
      x input[i] := int output:
     END:

    FOR i := 1 TO 240 DO
    BEGIN
     quit flag := TRUE:
     kk := 0:
       If x input[i] >= 0 THEN sign := 1
       ELSE sign := -1:
     WHILE quit flag DO
      BEGIN
       IF (ABS(x input[i]) < pcm array[kk+1])
                           AND (ABS(x input[i]) >= pcm array[kk]) THEN
        BEGIN
         q noise[i] := sign * (ABS(x input[i]) - pcm array[kk]):
         quit flag := FALSE:
        END:
       kk := kk+1:
      END:
     total noise := x input[i]*q noise[i] + total noise:
    END:
     sum := sum + total noise:
     variance := variance + SQR(total noise):
   END:
    sum := sum/10:
    variance := variance/10 - SQR(sum):
    noise signal := sum/120.0/SQR(std dev):
    q noise to signal := noise signal:
    END:

END:(*ends correlated noise *)
```

## Appendix D.

### D.1 I/O Map for PPIs.

| Address (Hex) | PPI No. | Port | Description. |
|---|---|---|---|
| FB | CPU USART | Cntl Reg. | Used for printer. |
| FA | | Data Reg. | |
| 63 | PPI-11 | Cntl Reg.1 | Set to 83H (fig. D.1a) |
| 62 | | Port C | LST interface (see below) |
| 61 | | Port B | Control-line A/D |
| 60 | | Port A | Spare |
| 67 | PPI-12 | Cntl Reg.2 | Set to 81H |
| 66 | | Port C | DAF interface (see below) |
| 65 | | Port B | RX variolosser D/A |
| 64 | | Port A | TX variolosser D/A |
| 73 | PPI-21 | Cntl Reg. 3 | Set to 92H |
| 72 | | Port C | Triggers for A/D and D/As |
| 71 | | Port B | Hybrid echo A/D |
| 70 | | Port A | Hybrid error A/D |
| 77 | PPI-22 | Cntl Reg.4 | Set to 92H |
| 76 | | Port C | Hex display (o/p) |
| 75 | | Port B | Acoustic echo A/D |
| 74 | | Port A | Acoustic error A/D |

Port bit assignments.

PPI-11 Port C

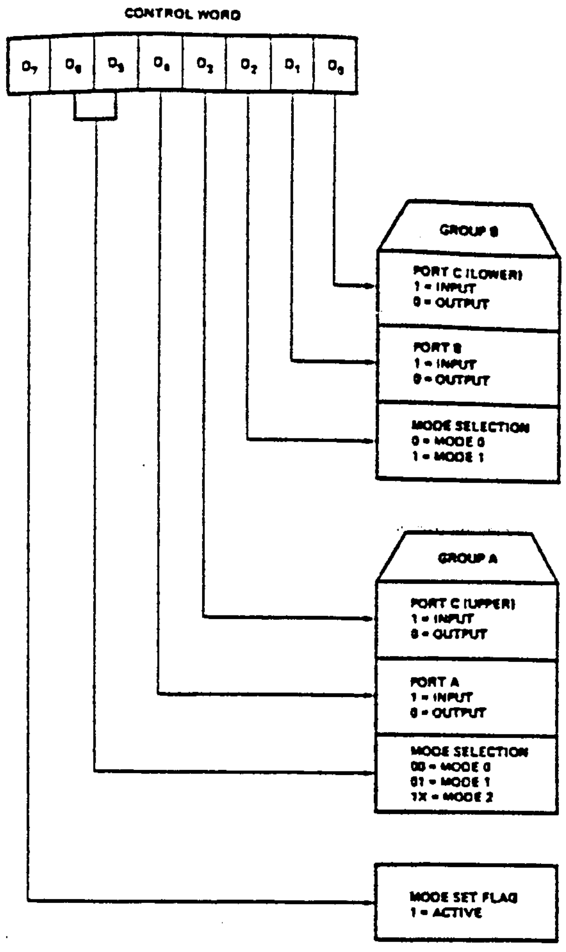
| Bit No. | Direction | Description |
|---|---|---|
| PC7 | ——> | |
| PC6 | ——> | Loop ON/OFF |
| PC5 | ——> | Noise generator ON/OFF |
| PC4 | ——> | Trigger A/D on control line |
| PC3 | <—— | Voice-switched mode 1 = ON |
| PC2 | <—— | Panic button |
| PC1 | <—— | LST ON/OFF |
| PC0 | <—— | Control-line A/Dconversion complete |

PPI-12 Port C

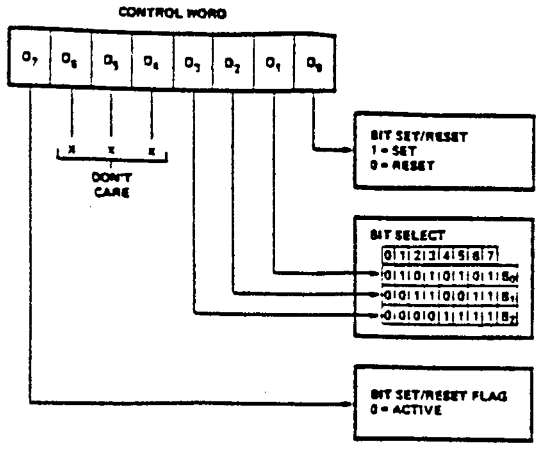
| Bit No. | Direction | Description |
|---|---|---|
| PC7 | ——> | Zero-H acoustic DAF |
| PC6 | ——> | Zero-H hybrid DAF |
| PC5 | ——> | Recirculate-H acoustic DAF |
| PC4 | ——> | Recirculate-H hybrid DAF |
| PC3 | <—— | Overflow acoustic DAF |
| PC2 | <—— | Overflow hybrid DAF |
| PC1 | <—— | Signal/Noise comparator |
| PC0 | <—— | TX-RX comparator |

## D.2 System Memory Map for MSSS 8080.

| | | | |
|---|---|---|---|
| 2BFF | 2000 | program (to 2892) on 3 by 1k EPROM (PROM card) | |
| 1FFF | 1400 | spare | |
| 13FF | 1300 | Stack area | 1k RAM (on CPU card) |
| 12FF | 1000 | Data area | 1k RAM (on CPU card) |
| 0FFF | 0C00 | Spare | 1k EPROM (on CPU card). |
| 0BFF | 0800 | Log lookup table | 1k EPROM (on CPU card). |
| 07FF | 0000 | SDK Monitor 2 by 1k EPROM (on CPU card) | |



Mode Definition Format



Bit Set/Reset Format

Figure D.1 Mode selection and bit set/reset format for PPIs

## D.3 Controller program.

### D.3.1

```
MAIN: DO:
    DECLARE LIT LITERALLY 'LITERALLY':
    DECLARE DCL LIT 'DECLARE':
    DCL PROC LIT 'PROCEDURE':
    DCL INIT LIT 'INITIAL':
    /******/
    DCL  VLOSS$CNTL     LIT '61H'. /*CONTROL-LINE INTERFACE ADDRESS*/
         CNTL$REG1      LIT '63H'.
         CNTL$REG2      LIT '67H'. /* SEE I/O MAP */
         PORTC$11$ADD   LIT '62H'.
         PORTC$12$ADD   LIT '66H'.
         HEX$DISPLAY    LIT '76H':    /* ADDRESS OF LED DISPLAY */
    DCL (PORTC11.
         PORTC12.
         COUNT INIT(0).
         DELAY INIT(0).
         N.
         H$COUNT.
         A$COUNT.
         NEW$AVE$H.
         OLD$AVE$H.
         NEW$AVE$A.
         OLD$AVE$A.
         DET$COUNT INIT(04H). /* DETERIMENTAL COUNTER */
         AVE$LNTH INIT(08H).
         DIVISOR INIT(02H).
         OAL. /*OVERAL ACOUSTIC LOSS*/
         OHL. /*OVERALL HYBRID LOSS */
         ONSWITCH.
         PANIC. /* PANIC BUTTON RESTARTS PROGRAM */
         SIG$NOISE. /* O/P FROM SIG-TO-NOISE COMPARATOR */
         TX$RX. /* O/P FROM TX-RX COMPARATOR */
         VOLUME INIT(0).      /*CONTROL LINE VALUE IN RECEIVE*/
         NOISE$GUARD INIT(0)./*CONTROL LINE VALUE IN TRANSMIT*/
         TX$COUNT.            /*TO MEASURE TIME IN TX AND IDLE*/
         MAX$TIME INIT(0FFH): /* 1 SEC AT 4 mS LOOP RATE*/) BYTE:
     DCL (STATE$COUNT.
          LAST$STATE.
          CHANGE$STATE /* TO MONITOR A CHANGE OF TX-RX STATE */) BYTE:
     DCL (VS$DEPTH.  /* DEPTH OF VOICE-SWITCHING IN dBS*/
          LAST$T$LOSS.
          LAST$R$LOSS) BYTE:
     DCL THRESH BYTE PUBLIC:
     DCL SHIFT BYTE PUBLIC:
     DCL TRUE LIT '0FFH'.
         FALSE LIT '00H':
     DCL (SWITCH.
          TEST.
          CHECK) BYTE:
     DCL LOG(1024) BYTE PUBLIC AT(0800H): /* 20LOG10 LOOKUP TABLE */
     DCL ANTILOG(61) ADDRESS PUBLIC
         (1.1.1.1.1.1.2.2.2.2.3.3.4.4.5.5.6.7.8.9.10.11.12.14.16.18.
          20.22.25.28.32.35.40.45.50.56.63.71.79.89.100.112.126.141.
          159.178.200.224.251.282.316.355.398.447.501.562.631.708.794.
          891.1000):
     DCL (H$SUM.
          A$SUM) ADDRESS: /*16 BIT DATA VALUE*/
    /****************/
    INITIALISER: PROC EXTERNAL:
                 END INITIALISER:
```

```
IDLE$LOOP: PROC EXTERNAL:
           END IDLE$LOOP:

ACOUSTIC$SETUP : PROC(OAL$PNTR) EXTERNAL:
   DCL   OAL$PNTR ADDRESS: /*ADDRESS FOR ADAPTAION OF ACOUSTIC DAF*/
   END ACOUSTIC$SETUP:

VOICE$SWITCH:  PROC EXTERNAL:
               END VOICE$SWITCH:

SWITCH$DEPTH: PROC(OHL$PNTR.OAL$PNTR.VS$DEPTH$PNTR) EXTERNAL:
              DCL (OAL$PNTR.
                   OHL$PNTR.
                   VS$DEPTH$PNTR) ADDRESS:
              END SWITCH$DEPTH:

VST$CNTL: PROC(VS$DEPTH$PNTR.L$T$LOSS$PNTR.L$R$LOSS$PNTR.VOLUME$PNTR.
               NOISE$G$PNTR.STATE$PNTR) EXTERNAL:
          DCL(VS$DEPTH$PNTR.
              L$T$LOSS$PNTR.
              L$R$LOSS$PNTR.
              VOLUME$PNTR.
              NOISE$G$PNTR.
              STATE$PNTR) ADDRESS:
          END VST$CNTL:

VSR$CNTL: PROC(VS$DEPTH$PNTR.L$T$LOSS$PNTR.L$R$LOSS$PNTR.VOLUME$PNTR.
               NOISE$G$PNTR.STATE$PNTR) EXTERNAL:
          DCL(VS$DEPTH$PNTR.
              L$T$LOSS$PNTR.
              L$R$LOSS$PNTR.
              VOLUME$PNTR.
              NOISE$G$PNTR.
              STATE$PNTR) ADDRESS:
          END VSR$CNTL:

HYBRID: PROC(OHL$PNTR) EXTERNAL:
        DCL (OHL$PNTR) ADDRESS:
        END HYBRID:

ACOUST: PROC(OAL$PNTR) EXTERNAL:
        DCL (OAL$PNTR) ADDRESS:
        END ACOUSTIC:
/****************/
START:
   THRESH = 50H: /* SETS MIN VALUE OF XIN BEFORE ADAPTAION IS MEASURED*/
   H$COUNT.A$COUNT = 00H:
   H$SUM.A$SUM = 0000H:
   NEW$AVE$H.OLD$AVE$H.NEW$AVE$A.OLD$AVE$A = 0000H:
   STATECOUNT = FALSE:   /* USED IN TX-RX TRANSITION COMPARISON */
   LAST$STATE = 01H:     /* SET FOR TRANSMIT I.E. 00H = RECEIVE */
   CHANGE$STATE = FALSE:          /* SET FOR NO CHANGE OF STATE */
   TX$COUNT = 00H:
   /**********/
   CALL INITIALISER:                              /* SET UP PPIs */
   /**********/
WAIT: CALL IDLE$LOOP:             /* WAIT OF ONSWITCH TO BE VALID */
   OUTPUT(CNTL$REG2) = 0EH:       /*ZERO H ON ACOUSTIC DAF */
   CALL TIME(0FFH):               /* HELD LOW FOR 125 MICROSECS  */
   OUTPUT(CNTL$REG2) = 0FH: /* BIT 7 HIGH RELEASES ZERO CONDITION */
   DO N = 0 TO 10:
```

```
   CALL TIME(0FFH):
   END:              /* GIVE TIME FOR HAND TO LEAVE ON SWITCH AT LST */
   CALL ACOUSTIC$SETUP:        /* THIS MAY OR MAY NOT BE TOLERABLE */
/************/
RESTART:
   ENABLE:  /* ENABLE INTERRUPTS */
   OUTPUT(CNTL$REG2) = 0DH:    /*RELEASE ZERO H ON HYBRID DAF*/
   OUTPUT(CNTL$REG2) = 09H:    /*SET HYBRID DAF TO CONVERGE   */
   OHL = 00H:                  /* INITIALISE OVERALL HYBRID LOSS*/
   OAL = 00H:                  /* AND ACOUSTIC LOSS            */
VS$MODE:                       /* VOICE-SWITCHED MODE */
   SWITCH = INPUT(PORTC11$ADD) AND 08H:  /* READ VS MODE SWITCH */
   IF SWITCH = 08H THEN CHECK = TRUE:
   ELSE CHECK = FALSE:
   /**********/
   DO WHILE CHECK =  TRUE:     /* RUN IN VS MODE WHEN SWITH ON */
     CALL VOICE$SWITCH:        /* O/P CONTROL LINE TO VARIOLOSSERS */
     PORTC11 = INPUT(PORTC11$ADD):  /* READ SWITCHES */
     ONSWITCH = PORTC11 AND 02H:    /* READ LST ON-OFF   SWITCH */
     IF ONSWITCH = 00H THEN GO TO START:
     SWITCH = PORTC11 AND 08H:      /* LOOK FOR VS MODE OFF */
     IF SWITCH <> 08H THEN CHECK = FALSE:
   END:                             /* END VOICE-SWITCHED MODE */
   /**********/
   OUTPUT(CNTL$REG1) = 09H:         /* SIGNAL A/D CONVERSION ON*/
   OUTPUT(CNTL$REG1) = 08H:         /* CONTROL LINE            */
   LAST$T$LOSS.LAST$R$LOSS = INPUT(VLOSS$CNTL): /*INITIALISE LAST VALUES*/
   /**********/
   DO WHILE TRUE:                      /* MAIN LOOP DO FOREVER */
   LOOP:                       /* CALC THE DEPTH OF VOICE SWITCHING */
     CALL SWITCH$DEPTH(.OLD$AVE$H.OLD$AVE$A.VS$DEPTH):
     PORTC11 = INPUT(PORTC11$ADD):            /* READ SWITCHES */
     SWITCH = PORTC11 AND 08H:                /* CHECK FOR VS MODE */
     IF SWITCH = 08H THEN GOTO VS$MODE:
     ONSWITCH = PORTC11 AND 02H:              /* CHECK LST STILL ON */
     IF ONSWITCH <> 02H THEN GO TO START:
     PANIC = PORTC11 AND 04H:                 /* CHECK FOR MANUAL PANIC */
     IF PANIC = 04H THEN
      DO:
        CALL ACOUSTIC$SETUP:
        GO TO RESTART:
      END:
     PORTC12 = INPUT(PORTC12$ADD):
     TX$RX = PORTC12 AND 01H:   /* EXAMINE STATE OF TX-RX COMPARATOR*/
     CHANGE$STATE = TX$RX XOR LAST$STATE:  /* LOOK FOR A TRANSITION */
     IF CHANGE$STATE = 01H THEN STATE$COUNT = 0FFH:
     LAST$STATE = TX$RX:
     IF TX$RX = 01H: THEN             /* IN TRANSMIT MODE */
      DO:
       OUTPUT(CNTL$REG2) = 0AH:       /* RECIRCULATE ACOUSTIC DAF*/
       OUTPUT(CNTL$REG2) = 09H:       /* HYBRID DAF TO CONVERGE  */
       OUTPUT(CNTL$REG1) = 09H:       /* SIGNAL A/D ON CONTROL LINE*/
       OUTPUT(CNTL$REG1) = 08H:
  /* O/P TO VARIOLOSSERS THE PREVIOUSLY CALC VALUE FOR VOICE SWITCHING*/
       CALL VST$CNTL(.VS$DEPTH..LAST$T$LOSS..LAST$R$LOSS.
                  .VOLUME..NOISE$GUARD..STATECOUNT):
       SIG$NOISE = PORTC12 AND 02H:   /* READ SIG/NOISE COMPARATOR */
       IF SIG$NOISE = 02H THEN
        DO:                           /* SIGNAL PRESENT */
         IF STATE$COUNT = FALSE THEN      /*ATTENUATION FOR LOCAL NOISE*/
          NOISE$GUARD = INPUT(VLOSS$CNTL):/*WHEN NO TRANSITION         */
        TX$COUNT = TX$COUNT + 1: /* MEASURE LENGTH OF TIME SPENT IN TX */
        IF TX$COUNT = MAX$TIME THEN
         DO:  /* ASSUME THAT ACOUSTIC PATH WILL LOSE ADAPTATION AND
                                RAMP DOWN ENHANCEMENT*/
         IF OLD$AVE$A > 0 THEN OLD$AVE$A = OLD$AVE$A -1:
         TX$COUNT = 0:                     /* ACOUSTIC ENHANCEMENT          */
         END:
        CALL HYBRID(.OHL):                 /*CALC NEW VALUE OF OHL */
        /* USES A NONLINEAR AVERAGING ROUTINE WHICH UPDATES OHL*/
        /* IMMEDIATELY IF THERE IS AN IMPROVEMENT, BUT REDUCES */
        /* THE ENHANCEMENT SLOWLY IF ADAPTATION FALLS.          */
        IF OHL > OLD$AVE$H THEN
          DO:                                /* CHANGE TO NEW IF IMPROVED */
          OLD$AVE$H = OHL:
          H$COUNT = 00H:                     /*SET DETRIMENTAL COUNTER */
          END:
        ELSE
        DO:
          H$COUNT = HCOUNT + 1:              /*COUNT NO OF FALLS OF OHL */
          IF H$COUNT = DET$COUNT THEN
            DO:
            OLD$AVE$H = OLD$AVE$H - 1:  /*RAMP DOWN ENHANCEMENT */
            H$COUNT = 00H:              /*RESET COUNTER */
            END:
      END:
      ELSE
       DO:                    /* IN IDLE, RECIRCULATE HYBRID DAF */
       OUTPUT(CNTL$REG2) = 08H:  /* RECIRCULATE HYBRID DAF */
       TX$COUNT = TX$COUNT + 1: /* MEASURE TIME SPENT IN IDLE   */
       IF TX$COUNT = MAX$TIME THEN
        DO:   /* ASSUME THAT ACOUSTIC PATH WILL LOSE ADAPTATION
                             AND RAMP DOWN ENHANCEMENT*/
        IF OLD$AVE$A > 0 THEN OLD$AVE$A = OLD$AVE$A -1:
        TX$COUNT = 0:                     /* ACOUSTIC ENHANCEMENT          */
        END:
       END:                            /* END IDLE STATE */
    END:                               /* END TRANSMIT STATE */
      ELSE
       DO:                             /* IN RECEIVE STATE  */
       OUTPUT(CNTL$REG2) = 08H:        /* HYBRID DAF TO RECIRCULATE*/
       OUTPUT(CNTL$REG2) = 0BH:        /* ACOUSTIC DAF TO CONVERGE */
       OUTPUT(CNTL$REG1) = 09H:        /* SIGNAL A/D ON CONTROL LINE*/
       OUTPUT(CNTL$REG1) = 08H:
     /* O/P TO VARIOLOSSERS PREVIOUSLY CALC LEVEL OF VOICE SWITCHING */
       CALL VSR$CNTL(.VS$DEPTH,.LAST$T$LOSS,.LAST$R$LOSS,
                     .VOLUME,.NOISE$GUARD,.STATE$COUNT):
       SIG$NOISE = PORTC12 AND 02H:  /* CHECK FOR SIGNAL PRESENT*/
       IF SIG$NOISE = 02H THEN
        DO:
        IF STATE$COUNT = FALSE THEN
        VOLUME = INPUT(VLOSS$CNTL):   /*VOLUME SETTING FOR  RX PATH */
        CALL  ACOUST(.OAL):           /*CALC NEW VALUE OF OAL */
        IF OAL > OLD$AVE$A THEN       /*REPEAT FAST ATTACK SLOW DECAY*/
         DO:                           /*AS FOR TRANSMIT STATE        */
         OLD$AVE$A = OAL:
         A$COUNT = 00H:
         END:
        ELSE
         DO:
         A$COUNT = A$COUNT + 1:
         IF A$COUNT = DET$COUNT THEN
```

```
          DO:
          OLD$AVE$A = OLD$AVE$A - 1:       /*RAMP DOWN ENHANCEMENT */
          A$COUNT = 0:
          END:
         END:
        END:
         ELSE
          OUTPUT(CNTL$REG2) = 0AH:        /*RECIRCULATE ACOUSTIC DAF*/
          /* CAN ONLY LAST 1.5 SECS NO NEED TO MEASURE TIME HERE */
      END:                                /* END  RECEIVE LOOP */
END:                                /* END DUPLEX LST PROGRAM */
HOWL: PROCEDURE INTURRUPT 7:
      /* RST7 JAMMED IN ADDRESS 38H ON DETECTION OF HOWL */
      GO TO RESTART:
      END HOWL:
GO TO  START:                       /* SHOULD PROG FAIL */
END MAIN:
```

D.3.2

```
START: DO:
    INITIALISER: PROCEDURE PUBLIC:
         /*ROUTINE TO SET UP I/P AND O/P PORTS TO LST */
      DECLARE LIT LITERALLY 'LITERALLY':
      DECLARE DCL LIT 'DECLARE':
      DCL (CNTL11.CNTL12.CNTL21.CNTL22) BYTE:
      DCL CNTL$REG1 LIT '63H'. /* SEE I/O MAP FOR PPIs */
          CNTL$REG2 LIT '67H'. /* APPENDIX D.1          */
          CNTL$REG3 LIT '73H'.
          CNTL$REG4 LIT '77H':
      /***********************************************/
      CNTL11 = 83H:   /* IO1. PPI1 IS MODE 0 PORT A O/P. C UPPER O/P */
                      /*                     PORT B I/P. C LOWER I/P */
      CNTL12 = 81H:   /* IO1. PPI2 IS MODE 0. ALL PORTS O/P          */
      CNTL21 = 92H:   /* IO2. PPI1 IS MODE 0. A I/P - READ XIN.      */
                      /* C UPPER O/P. B I/P - READ ECHO. C LOWER O/P */
      CNTL22 = 92H:   /* IO2. PPI2 IS MODE 0. A & B I/P.
                                             C IS O/P TO HEX DISPLAY */
      /************/
      OUTPUT(CNTL$REG1) = CNTL11:
      OUTPUT(CNTL$REG2) = CNTL12:
      OUTPUT(CNTL$REG3) = CNTL21:
      OUTPUT(CNTL$REG4) = CNTL22: /*SEND OUT ALL CONTROL WORDS TO PPIs*/
      /************/
      /*INITIALISE ALL OUTPUT PORTS TO ZERO */
      OUTPUT(60H) = 00H:  /*IO1. PPI1 PORT A*/
      OUTPUT(62H) = 60H:  /*CONTROLLER PORT C*/
      OUTPUT(64H) = 00H:  /*IO1. PPI2 PORT A */
      OUTPUT(65H) = 00H:  /*IO1. PPI2 PORT B */
      OUTPUT(66H) = 00H:  /*CONTROLLER PORT C*/
      /************/
    END INITIALISER:
END:
```

D.3.3

```
IDLE: DO:
  IDLE$LOOP: PROCEDURE PUBLIC:
  /***************************************************/
  /* ROUTINE TO SIT IN IDLE STATE MONITORING THE 'ON' */
  /* SWITCH. PUTS MAX LOSS IN VARIOLOSSERS AND RETURNS*/
  /* TO MAIN PROG WHEN 'ON' DETECTED                  */
  /***************************************************/
   DECLARE LIT LITERALLY 'LITERALLY':
   DECLARE DCL LIT 'DECLARE':
   DCL VLT$ADD LIT '64H':  /* TRANSMIT VARIOLOSSER */
   DCL VLR$ADD LIT '65H':  /* RECEIVE VARIOLOSSER ADDRESS*/
   DCL PORTC11 LIT '62H':  /* SWITCH PORT INCLUDING ON/OFF */
   DCL ONSWITCH BYTE:
   /**********/
   ONSWITCH = 00H:  /* INITIALISE SWITCH */
   OUTPUT(VLT$ADD) = 0FFH /* PUT MAX LOSS IN TX PATH */
   OUTPUT(VLR$ADD) = 00H /* DITTO RX PATH           */
    DO  WHILE ONSWITCH = 00H:
     ONSWITCH = INPUT(PORTC11) AND 02H: /* MASK LST ON/OFF  SWITCH*/
    END:  /* WAIT FOR ON CONDITION */
  END IDLE$LOOP:
END:
```

D.3.4

```
V$SWITCH: DO:
  VOICE$SWITCH: PROCEDURE PUBLIC:
  /***********************************************************/
  /*THIS PROCEDURE CAUSES THE LST TO BE RUN IN VOICE-SWITCHED */
  /*MODE. THE ANALOGUE CONTROL-LINE O/P IS PATCHED  DIRECTLY  */
  /*TO THE VARIOLOSSER INPUTS.                                */
  /************************************************************/
  DECLARE LIT LITERALLY 'LITERALLY':
  DECLARE DCL LIT 'DECLARE':
  DCL CNTL$REG1 LIT '63H'.
      CNTL$REG2 LIT '67H'.
      VLC$ADD   LIT '61H'. /*CONTROL-LINE ADDRESS */
      VLT$ADD   LIT '64H'. /*TX-PATH  VARIOLOSSER */
      VLR$ADD   LIT '65H': /*RX-PATH VARIOLOSSER  */
   DCL ADC BYTE:
  /************/
  /* READ STATE OF CONTROL-LINE FROM ANALOGUE PROCESSOR */
   OUTPUT(CNTL$REG1) = 09H: /* INITIALISE AND A/D CONVERSION */
   OUTPUT(CNTL$REG1) = 08H: /* STOP CONVERSION               */
   ADC = INPUT(VLC$ADD):    /* READ DIGITAL OUTPUT           */
   OUTPUT(VLT$ADD) = ADC:  /* O/P TO TX VARIOLOSSER          */
   OUTPUT(VLR$ADD) = ADC    /* O/P TO  RX VARIOLOSSER        */
  END VOICE$SWITCH:
END V$SWITCH:
```

D.3.5

```
ACSHYB: DO:  /* ACOUSTIC HYBRID */
  ACOUSTIC#SETUP: PROCEDURE(OAL$PNTR) PUBLIC:
  /*******************************************************/
  /* TO SETUP THE ACOUSTIC DAF WITH A NOISE SOURCE AND */
  /* ESTIMATE THE OVERALL ACOUSTIC LOSS AVAILABLE      */
  /*******************************************************/
  DECLARE LIT LITERALLY 'LITERALLY':
  DECLARE DCL LIT 'DECLARE':
  DCL CNTL$REG1 LIT '63H',
      CNTL$REG2 LIT '67H',
      CNTL$REG3 LIT '73H':     /*SEE I/O PORT MAP APPENDIX D.1*/
  DCL VSR$ADD LIT '65H',       /*RECEIVE VARIOLOSSER ADDRESS  */
      VST$ADD LIT '64H':       /*TRANSMIT VARIOLOSSER ADDRESS */
  DCL AC$ERR LIT '74H':        /* ACOUSTIC ERROR O/P A/D      */
  DCL TRUE LIT '0FFH',
      FALSE LIT '00H':
  DCL OAL$PNTR ADDRESS:        /*ADDRESS OF VALUE FOR OVERALL */
                               /*ACOUSTIC LOSS I.E ADAPTATION */
  DCL OAL BASED OAL$PNTR BYTE:
  DCL ERROR BYTE,
      N ADDRESS,
      QERROR ADDRESS:
  DCL COUNT BYTE:
      /*********************/
   COUNT = 25: /*TO EXTEND THE TIME FOR THE NOISE GEN TO 2 SECS*/
   QERROR = 0000H:
   /* SET VARIOLOSSERS TO MAXIMUM ATTENUATION */
   OUTPUT(VSR$ADD) = 00H:
   OUTPUT(VST$ADD) = 0FFH:
   RETRY:
     TEST = TRUE:
     OUTPUT(CNTL$REG2) = 0EH: /* ZERO THE IMPULSE RESPONSE */
     CALL TIME(0FFH):         /* HELD LOW FOR 125 MICROSECS */
     OUTPUT(CNTL$REG2) = 0FH: /* BIT 7 SET HIGH RELEASES ZERO */
                              /* H   CONDITION -RESETS PORT C */
     OUTPUT(CNTL$REG1) = 0AH: /* TURN ON THE NOISE GENERATOR  */
     OUTPUT(CNTL$REG2) = 0BH: /* BIT 5 SET HIGH TO CONVERGE */
      DO N = 0 TO COUNT:      /* FOR 2 SECONDS.             */
      CALL TIME(0FFH):
      END:
     /***********/
     DO WHILE TEST = TRUE:    /* MEASURE OVERALL ACOUSTIC LOSS */
     OUTPUT(CNTL$REG2) = 0AH: /* BIT 5 SET LOW TO RECIRCULATE H*/
     QERROR = 00H:            /* INITIALISE ERROR ACCUMULATOR  */
      DO N = 0 TO 3:      /* OBTAIN SMOOTHED ESTIMATE OF ERROR */
      OUTPUT(CNTL$REG3) = 09H: /*BIT 4 SET HIGH TO START A/D
                                 CONVERSION */
      OUTPUT(CNTL$REG3) = 08H: /*RESET BIT 4 LOW               */
      QERROR = QERROR + (0FFH - INPUT(AC$ERR)) /* A/D IN COB   */
      CALL TIME(0FFH):
      END:
     ERROR = LOW(SHR(QERROR,4)): /*SMOOTHED ESTIMATE           */
     IF ERROR < 20H THEN ERROR = 20H: /*AVOIDS ADAPTATION > 17.5 dB*/
     OAL = LOW(240/ERROR): /* 240 IS PRECALIBRATED VALUE OF    */
                           /* ACOUSTIC INPUT                   */
     IF OAL < 3 THEN TEST = TRUE:
     ELSE TEST = FALSE:  /* CONTINUE UNTIL 9.5 dBS ADAPTATION  */
    END:
   OUTPUT(CNTL$REG1) = 0BH:        /*TURN OFF NOISE GENERATOR */
  END  ACOUSTIC$SETUP:
  /* NB THIS ROUTINE CONTINUES PRODUCING NOISE UNTIL > 9.5 dB OF*/

  /* ADAPTATION HAS BEEN ACHIEVED. CAN BE FOOLED BY MOVEMENT IN */
  /* ACOUSTIC PATH TO CONTINUE INDEFINIATELY                    */
END AC$HYB:
```

**D.3.6**

```
ACTPH: DO:
  DECLARE THRESH BYTE EXTERNAL:
  ACOUST: PROCEDURE(OAL$PNTR) PUBLIC:
      /*************************************************/
      /* CALLED DURING A KNOWN TRANSMIT PERIOD AND      */
      /* MEASURES THE ACOUSTIC ADAPTATION AS A RATIO.   */
      /* FOR PESIMISTIC STABILITY. IGNORES LOCAL NOISE  */
      /*************************************************/
      DECLARE LIT LITERALLY 'LITERALLY':
      DECLARE DCL LIT 'DECLARE':
      DCL CNTL$REG1  LIT '73H'.
          AC$ERR     LIT '74H'.           /*A/D ON ERROR */
          ACHXIN     LIT '75H'.           /*A/D ON X INPUT*/
          HEXDISPLAY LIT '76H':
      DCL (AC$XIN.
           AC$ERROR) ADDRESS:
      DCL OAL$PNTR ADDESS:
      DCL OAL BASED OAL$PNTR BYTE:
      /****************************/
      OUTPUT(CNTL$REG3) = 09H:     /* SET PULSE FOR A/D */
      OUTPUT(CNTL$REG3) = 08H:
      AC$XIN = 0FFH - INPUT(ACHXIN):  /*A/D IN COB*/
      AC$ERROR = 0FFH - INPUT(AC$ERR):/* READ IN RMS */
      /* AVOID DIVISION BY SMALL NUMBERS */
      IF AC$ERROR < 02H THEN  AC$ERROR =  02H:
      IF AC$XIN > AC$ERROR THEN          /*PROBABLY IN RECEIVE */
       IF AC$XIN > THRESH THEN     /* AVOID LOW INPUT SIGNALS */
       OAL = LOW(AC$XIN/AC$ERROR):
  END ACOUST:
 END  ACPTH:
```

**D.3.7**

```
VS$DEPTH:   DO:
DECLARE LOG(1024) BYTE EXTERNAL: /*20LOG10 LOOKUP TABLE*/
/**********/
SWITCH$DEPTH: PROCEDURE(OHL$PNTR.OAL$PNTR.VS$DEPTH$PNTR) PUBLIC:
/***************************************************************/
/* TAKES IN THE OVERALL ACOUSTIC AND HYBRID LOSS MEASUREMENTS */
/* TO CALCULATE THE STABILITY FACTOR                          */
/***************************************************************/
DECLARE DCL LITERALLY 'DECLARE':
DCL (OHL$PNTR.
     OAL$PNTR.
     VS$DEPTH$PNTR) ADDRESS:
DCL (OAL.
     OHL.
     VS$DEPTH) BYTE:
DCL STABILITY ADDRESS:
/*******************/
STABILITY = OAL * OHL:
STABILITY = STABILITY/5:
               /* CORRESPONDS TO 14 dB MARGIN FOR WHITE NOISE */
            /* CAN BE INCREASED TO IMPROVE STABILITY ON SPEECH */
IF STABILITY =  0 THEN
 DO:           /* MEASUREMENT ERROR IE AN INPUT WAS ZERO */
 VS$DEPTH = 0:  /* NO ENHANCEMENT */
 RETURN:
 END:
IF STABILITY < 1024 THEN
   VS$DEPTH = LOG(STABILITY):                    /* 60 dB MARGIN */
ELSE VS$DEPTH = 60:                        /* RETURN VALUE IN dBS*/
END SWITCH$DEPTH:
END VS$DEPTH:
```

D.3.8

```
ELHYB: DO:
  DECLARE THRESH BYTE EXTERNAL:
  HYBRID: PROCEDURE(OHL$PNTR) PUBLIC:
      /*****************************************************/
      /* CALLED DURING A KNOWN RECEIVE PERIOD AND          */
      /* MEASURES THE HYBRID ADAPTATION AS A RATIO.        */
      /* FOR PESIMISTIC STABILITY. IGNORES LOCAL NOISE     */
      /*****************************************************/
      DECLARE LIT LITERALLY 'LITERALLY':
      DECLARE DCL LIT 'DECLARE':
      DCL CNTL$REG1  LIT '73H',
          HY$ERR     LIT '74H',           /*A/D ON ERROR */
          EHYXIN     LIT '75H',           /*A/D ON X INPUT*/
          HEXDISPLAY LIT '76H':
      DCL (HY$XIN,
           HY$ERROR) ADDRESS:
      DCL OHL$PNTR ADDESS:
      DCL OHL BASED OHL$PNTR BYTE:
      /*****************************/
      OUTPUT(CNTL$REG3) = 09H:      /* SET PULSE FOR A/D */
      OUTPUT(CNTL$REG3) = 08H:
      HY$XIN = 0FFH - INPUT(EHYXIN):      /*A/D IN COB*/
      HY$ERROR = 0FFH - INPUT(HY$ERR):    /*READ IN RMS*/
      /* AVOID DIVISION BY SMALL NUMBERS */
      IF HY$ERROR < 02H THEN  HY$ERROR = 02H:
      IF HY$XIN > HY$ERROR THEN           /*PROBABLY IN TRANSMIT*/
       IF HY$XIN > THRESH THEN      /* AVOID LOW INPUT SIGNALS */
       OHL = LOW(HY$XIN/HY$ERROR):
 END HYBRID:
END ELHYB:
```

D.3.9

```
VSTC: DO:
  /**/
  VST$CNTL: PROCEDURE(VS$DEPTH$PNTR.L$T$LOSS$PNTR.L$R$LOSS$PNTR.
                      VOLUME$PNTR.NOISE$PNTR.STATE$PNTR) PUBLIC:
  /*****************************************************/
  /*FOR TRANSMIT CONDITION ONLY                          */
  /*TAKES IN THE STABILITY & CALCS THE ALLOWABLE GAIN IN */
  /*RECEIVE PATH. DEPENDING ON VOLUME CONTROL            */
  /*AND THE NOISE-GUARD CIRCUIT.                         */
  /*ALSO HANDLES A TRANSITION  FROM RX TO TX             */
  /*****************************************************/
  DECLARE LIT LITERALLY 'LITERALLY':
  DECLARE DCL LIT 'DECLARE':
  DCL TRUE LIT '0FFH'.
      FALSE LIT '00H':
  DCL CNTL$REG1 LIT  '63H'.
      CNTL$REG2 LIT  '67H'.
      VLOSS$CNTL LIT '61H'.   /* O/P FROM CONTROL LINE */
      TX$VLOSS   LIT '64H'.   /* I/P TO TX VARIOLOSSER */
      RX$VLOSS   LIT '65H'.   /* I/P TO RX VARIOLOSSER */
      HEX$DISPLAY LIT '76H':
  DCL (VS$DEPTH$PNTR.
       L$T$LOSS$PNTR.
       L$R$LOSS$PNTR.
       VOLUME$PNTR.
       NOISE$PNTR.
       STATE$PNTR) ADDRESS:
  DCL (VS$DEPTH BASED VS$DEPTH$PNTR.
       LAST$T$LOSS BASED L$T$LOSS$PNTR.
       LAST$R$LOSS BASED L$R$LOSS$PNTR.
       VOLUME BASED VOLUME$PNTR. /* RANGE 120(MAX) TO 240(MIN)*/
       NOISE$GUARD BASED NOISE$PNTR. /*RANGE 0 TO  40*/
       STATE$COUNT BASED STATE$PNTR) BYTE:
  DCL (VR$GAIN.            /*GAIN IN RECEIVE LEG FROM VOLUME*/
       VR$ALLOWED.         /*ALLOWED GAIN FOR STABILITY */
        RX$LOSS.            /*VALUE TO O/P TO RX VARIOLOSSER*/
        RX$TRANSIT.         /*USED IN TRANSITION */
        TX$TRANSIT.
        STORE.
        TEMP$T) BYTE:
  DCL (TX$FLAG.
       RX$FLAG) BYTE:
   /****************************************/
   /* TO CALC THE ALLOWABLE GAIN IN RX */
   IF VOLUME > 120 THEN VR$GAIN = SHR((VOLUME - 120).2): /*IN dBS*/
   ELSE VR$GAIN = 0: /* ERROR OCCURRED SET TO LEAST DAMAGE*/
   IF VS$DEPTH > SHR(NOISE$GUARD.2) THEN
                VR$ALLOWED = VS$DEPTH - SHR(NOISE$GUARD.2):
   ELSE VR$ALLOWED = 0:  /* NO ENHANCEMENT */
   /*IF HAVE MORE STABILITY THAN REQUIRED BY VOLUME THEN LIMIT*/
   IF VR$ALLOWED > VR$GAIN THEN VR$ALLOWED = VR$GAIN:
   RX$LOSS = 0F0H - SHL(VR$ALLOWED.2): /* VALUE SUITABLE
                                  FOR O/P TO RX VARIOLOSSER */
   /**************/
   /* TRANSITION DETECTED FROM RX TO TX */
   IF STATE$COUNT = TRUE THEN
    DO:
    RX$TRANSIT = LAST$R$LOSS:
    TX$TRANSIT = LAST$T$LOSS:
    TX$FLAG.RX$FLAG = FALSE:
     DO WHILE STATE$COUNT = TRUE: /*RAMP VOICE SWITCH OVER */
    RX$TRANSIT = RX$TRANSIT - 4: /*IN ARBITARY STEPS      */
    TX$TRANSIT = TX$TRANSIT - 4:
    IF RX$TRANSIT >  RX$LOSS THEN OUTPUT(RX$VLOSS) = RX$TRANSIT:
    ELSE RX$FLAG = TRUE:  /* UNTIL ALLOWED GAIN LEVEL */
    IF TX$TRANSIT > NOISE$GUARD THEN OUTPUT(TX$VLOSS) = TX$TRANSIT:
    ELSE TX$FLAG = TRUE: /*UNTIL NOISE GUARD LIMIT */
    IF TX$FLAG = TRUE AND RX$FLAG = TRUE THEN  STATE$COUNT = FALSE:
    END: /* DO WHILE LOOP. IE RAMPING  COMPLETE */
   /* NOW WAIT IN TX UNTIL CONTROL LINE HAS SETTLED */
   /* TO PREVENT MIS-READING OF CONTROL-LINE CHANGE */
   TX$FLAG = TRUE:
   STORE = NOISE$GUARD + 16: /* 4 dB ABOVE NOISEGUARD */
    DO WHILE TX$FLAG = TRUE:
    OUTPUT(CNTL$REG1) = 09H: /* INITIALISE A/D ON CONTROL LINE*/
    OUTPUT(CNTL$REG1) = 08H:
    TEMP$T = INPUT(VLOSS$CNTL):
    IF STORE > TEMP$T THEN /*WAIT UNTIL CNTL LINE GOES BETWEEN */
    TX$FLAG = FALSE:        /*STORE AND NOISEGUARD*/
    CALL TIME(10H):
    END:  /* WAIT ON CONTROL LINE CHANGE */
   END: /* DETECTION OF A TRANSITION */
   /******************/
   /* OUTPUT THE VALUES TO EACH VARIOLOSSER */
   LAST$T$LOSS = NOISE$GUARD:
   LAST$R$LOSS = RX$LOSS:
   OUTPUT(RX$VLOSS) = LAST$R$LOSS:
   OUTPUT(TX$VLOSS) = LAST$T$LOSS:
   /***********/
   OUTPUT(HEXDISPLAY) = SHR((VOLUME - 120).2) - VR$ALLOWED:
   /* DISPLAYS DEPTH OF VS IN RX LEG */
  END VST$CNTL:
 END VSTC:
```

```
VSRC: DO:
  /**/
  VSR$CNTL: PROCEDURE(VS$DEPTH$PNTR.L$T$LOSS$PNTR.L$R$LOSS$PNTR.
                      VOLUME$PNTR.NOISE$PNTR.STATE$PNTR) PUBLIC:
  /*******************************************************************/
  /*FOR RECEIVE CONDITION ONLY                                       */
  /*TAKES IN THE STABILITY MARGIN & CALCS DEPTH OF VOICE             */
  /*SWITHC FOR THE TX LEG DEPENDING ON VOLUME CONTROL                */
  /*THE NOISE-GUARD CIRCUIT. AND SIDETONE REQUIREMENT                */
  /*ALSO HANDLES A TRANSITION  FROM TX TO RX                         */
  /*******************************************************************/
  DECLARE LIT LITERALLY 'LITERALLY':
  DECLARE DCL LIT 'DECLARE':
  DCL TRUE LIT 'OFFH'.
      FALSE LIT '00H':
  DCL CNTL$REG1 LIT  '63H'.
      CNTL$REG2 LIT  '67H'.
      VLOSS$CNTL LIT '61H'.  /* O/P FROM CONTROL LINE */
      TX$VLOSS   LIT '64H'.  /* I/P TO TX VARIOLOSSER */
      RX$VLOSS   LIT '65H'.  /* I/P TO RX VARIOLOSSER */
      HEX$DISPLAY LIT '76H':
  DCL (VS$DEPTH$PNTR.
       L$T$LOSS$PNTR.
       L$R$LOSS$PNTR.
       VOLUME$PNTR.
       NOISE$PNTR.
       STATE$PNTR) ADDRESS:
  DCL (VS$DEPTH BASED VS$DEPTH$PNTR.
       LAST$T$LOSS BASED L$T$LOSS$PNTR.
       LAST$R$LOSS BASED L$R$LOSS$PNTR.
       VOLUME BASED VOLUME$PNTR. /* RANGE 120(MAX) TO 240(MIN)*/
       NOISE$GUARD BASED NOISE$PNTR. /* RANGE 0 TO 40 */
       STATE$COUNT BASED STATE$PNTR) BYTE:
   DCL (VR$GAIN.             /*GAIN IN RECEIVE LEG FROM VOLUME*/
        VT$REQUIRED.         /*NEEDED LOSS FOR STABILITY */
        RX$LOSS.             /*VALUE TO O/P TO RX VARIOLOSSER*/
        RX$TRANSIT.          /*USED IN TRANSITION */
        TX$TRANSIT.
        STORE.
        TEMP$R) BYTE:
   DCL (TX$FLAG.
        RX$FLAG) BYTE:
   /*********************************************/
   /* TO CALC THE DEPTH OF VOICE SWITCH FOR TX PATH*/
   IF VOLUME > 120 THEN VR$GAIN = SHR((VOLUME - 120).2): /* IN dBS */
   ELSE VR$GAIN = 0: /* ERROR OCCURRED SET TO LEAST DAMAGE*/
   IF VS$GAIN > VS$DEPTH THEN VT$REQUIRED = VR$GAIN - VS$DEPTH:
   ELSE VT$REQUIRED = 0:  /* MAX ENHANCEMENT */
   /* NOW INSERT ATTENUATION ACCORDING TO NOISE GUARD
                                        AND SIDETONE LIMIT*/
   IF VT$REQUIRED < SHR(NOISE$GUARD.2) THEN
   VT$REQUIRED = NOISE$GUARD: /* INCREASE LOSS TO NOISEGUARD LIMIT*/
   IF VT$REQUIRED < 6 THEN  VT$REQUIRED =  6: /* MIN FOR SIDETONE*/
   TX$LOSS = SHL(VT$REQUIRED.2): /* VALUE SUITABLE
                                  FOR O/P TO TX VARIOLOSSER */
   /***************/
   /* TRANSITION DETECTED FROM TX TO RX */
   IF STATE$COUNT = TRUE THEN
    DO:
    RX$TRANSIT = LAST$R$LOSS:
    TX$TRANSIT = LAST$T$LOSS:
```

```
    TX$FLAG.RX$FLAG = FALSE:
     DO WHILE STATE$COUNT = TRUE: /*RAMP VOICE SWITCH OVER */
     RX$TRANSIT = RX$TRANSIT + 4: /*IN ARBITARY (1 dB) STEPS */
     TX$TRANSIT = TX$TRANSIT + 4:
     IF RX$TRANSIT < VOLUME THEN OUTPUT(RX$VLOSS) = RX$TRANSIT:
     ELSE RX$FLAG = TRUE:  /* UNTIL VOLUME LEVEL */
     IF TX$TRANSIT < TX$LOSS THEN OUTPUT(TX$VLOSS) = TX$TRANSIT:
     ELSE TX$FLAG = TRUE: /*UNTIL NOISEGUARD OR SIDETONE LIMIT */
     IF TX$FLAG = TRUE AND RX$FLAG = TRUE THEN  STATE$COUNT = FALSE:
     END: /* DO WHILE LOOP. IE RAMPING  COMPLETE */
    /* NOW WAIT IN RX UNTIL CONTROL LINE HAS SETTLED */
    /* TO PREVENT MIS-READING OF CONTROL-LINE CHANGE */
    RX$FLAG = TRUE:
    STORE = VOLUME - 16: /* 4 dB LESS THAN LAST VOLUME SETTING */
     DO WHILE RX$FLAG = TRUE:
     OUTPUT(CNTL$REG1) = 09H: /* INITIALISE A/D ON CONTROL LINE*/
     OUTPUT(CNTL$REG1) = 08H:
     TEMP$R = INPUT(VLOSS$CNTL):
     IF STORE < TEMP$R THEN /*FOLLOW UNTIL CNTL LINE GOES BETWEEN*/
     RX$FLAG = FALSE:        /* STORE AND VOLUME*/
     CALL TIME(10H):
     END:  /* WAIT ON CONTROL LINE CHANGE */
    END: /* DETECTION OF A TX TO RX TRANSITION */
    /*******************/
    /* OUTPUT THE VALUES TO EACH VARIOLOSSER */
    LAST$T$LOSS = TX$LOSS:
    LAST$R$LOSS = VOLUME:
    OUTPUT(TX$VLOSS) = LAST$T$LOSS:
    OUTPUT(RX$VLOSS) = LAST$R$LOSS:
/************/
  OUTPUT(HEXDISPLAY) = SHR((LAST$T$LOSS - NOISE$GUARD).2):
  /* DISPLAYS DEPTH OF VS IN TX LEG  IN dBS*/
 END VSR$CNTL:
END VSRC:
```

## REFERENCES.

[101] D.A. Berkley and O.M. Mracek Mitchell, 'Seeking the Ideal in Handsfree Telephony,' Bell Labs. Record, Vol. 52, pp 318-324, Nov. 1974.

[102] O.M.M. Mitchell and D.A. Berkley, 'Reduction of Long Reverberation Time by a Center-Clipping Process,' Journal of the Acoustic Society of America, Vol. 47, pp 84, 1970.

[103] J.L. Flanagan and R.C. Lummis, 'Signal processing to Reduce Multipath Distortion in Small Rooms,' The Journal of the Acoustic Society of America, Vol. 47, No. 6, pp 1475-1481, Feb. 1970.

[104] R. McK Carson, 'Investigation of Optimum Listening-Level for Loudspeaking Telephones,' British Telecomm Technology Executive, Memorandum No. R13/005/82.

[105] G. Cosier, 'Reference Positions Associated with the Artificial Voice Type 4219 and Microphone Type 4132 for LST Testing,' British Telecomm Technology Executive, R13.3 Section Memo. No. G165, March 1984.

[106] CCITT 'Sensitivities of Loudspeaker Telephones,' Recommendation P.34. Yellow Book, Vol. 5, Nov. 1980.

[107] G. Dalzell, 'A Preliminary Investigation of LST Case Acoustic Cross-Coupling,' British Telecomm Technology Executive, R13.3.3 Group Memo. No. 01/84.

[108] G. Cook, Private Communication. BTTE Network Planning Dept.

[109] D. Mc Millan and J. Doust, 'Two-way transmission: volume regulation,' Patent No. 415,173, March 20, 1933, No. 8342.

[110] Standard Telephones and Cables, 'Two-way transmission,' Patent No. 439,752, Aug. 17 1934, No. 23794.

[111] L.H. Paddle, 'Two-way transmission,' Patent No. 451,496, Jan. 2 1935, Nos. 134 and 413.

[112] Telephone Manufacturing Co. and L.H. Paddle, 'Two-way transmission,' Patent No. 459,779, June 11 1935, No. 16791.

[113] L.E. Ryall, 'A New Subscriber's Loudspeaking Telephone,' POEEJ Vol. 29, No. 1, pp 6-15, April 1936.

[114] E.J. Barnes, L.E. Ryall, J.F. Doust, and D. McMillan, 'General Considerations Affecting the Design of Subscriber's Loud Speaking Telephones,' Post Office Research Report No. 9570, August 1936.

[115] K.S. Stanbury, 'Improved Telephone Substation Apparatus,' New Zealand Patent Nos. 30452/48 and 30453/48, filed Nov. 23 1948.

[116] K.S. Stanbury, 'Improved Telephone Substation Apparatus,' UK Patent No. 655927, Aug, 8, 1951.

[117] A. Busala, 'Fundamental Considerations in the Design of a Voice-Switched Speakerphone,' BSTJ Vol. 39, No. 2, pp 265-294, March 1960.

[118] B. Copping and R.G. Fidler, 'Designing a Voice-Switched Loudspeaking Telephone - LST No. 4,' POEE Journal, Vol. 60, Part 1, pp 65-71, April 1967.

[119] G.J. Barnes, 'Voice Switching Parameters in Telephony,' Electrical Communication , Vol. 47, No. 3, pp 186-196, 1972.

[120] O. Larsson, 'Characteristics of Voice-Switched Loudspeaking Telephones,' Ericsson Review, No. 3/4, pp 148-156, 1975.

[121] W. Clarke and J. Gale, 'A New Look at Loudspeaking Telephones,' Telesis, pp 79-84, Fall 1973,

[122] R.F. Laurence, 'Improvements Relating to Loudspeaking Telephone Systems,' UK Patent No. 909191.

[123] L.M. Ericsson, 'An Amplifier Arrangement for use in Loudspeaking Telephones,' UK Patent No. 944087.

[124] Gylling and Co., 'Amplifier for Two-Way Speech Transmission. UK Patent No. 979825.

[125] R.A. Jones, 'Improvements in or Relating to the Detection of Alternating Current Signals,' UK Patent No. 1046485.

[126] GEC Ltd., 'Improvements in of Relating to Loudspeaking Telephone Instruments,' UK Patent No. 1406243.

[127] J.L.E. Thompson, W.E. Clarke, J. Gale, 'Loudspeaking Communication Terminal Apparatus and Method of Operation,' US Patent No. 3889059.

[128] A. Rickaby, 'The Loudspeaking Telephone,' ATE Journal, Vol. 9, No. 1, pp 28-37, January 1953.

[129] J.L. Connan and G. Pays, 'Poste telephonique a ecoute amplifiee sur haut-parleur,' Brevet francais, Vol. 79, No. 17, pp 620, 1979.

[130] G. Ferrieu and P Amstutz, 'A New Principle to Avoid Undesirable Oscillations in Electro Acoustic Loops. Application to the Design of a Hands Free Telephone Set Without Voice Switching. Int. Conf. on Acoustics, Speech and Signal Processing 82.

[131] S. Bernard, D Lajotte, F. Michelon and J Sourgens, 'A Loudspeaking Telephone,' FR Patent No. 7737650 filed 14th Dec. 1977, UK Patent No. 2011230, filed 6th Nov. 1978.

[132] E.J. Powter, 'Evolution of a Loudspeaking Telephone,' PO Research Memorandum No. 78/R13/7, Nov. 1978.

[133] C.R. South, C.E. Hoppit and A.V. Lewis, 'Adaptive filters to Improve a Loudspeaking Telephone,' Electronic Letters, Vol. 15, No. 21, pp 673-679, Oct. 1979.

[201] H.A. Lancaster, 'Echo Canceller Performance in the U.K. Network,' B.T.T.E Memo. R13/005/83.

[202] C.L. Monk, 'Transmission Characteristics of Connections in the Switched Telephone Network Measured in the 1967 Field Study,' P.O. Res. Report No. 598 March 1977.

[203] C.E. Hoppitt. 'Measurements of Echo Path Impulse Responses at Wood Street Exchange Using Pseudo Noise Test Signal,' P.O. Research Report No. 715 July 1978.

[204] E. Mayer and E. Neumann, 'Physical and Applied Acoustics An Introduction,' Academic Press 1972.

[205] S. White, 'An Adaptive Recursive Digital Filter,' Conf. Rec. 9th. Annual Asilomar Conf. on Circ. Sys. and Comp. Nov. 1975, pp. 21-25.

[206] S. Stearns and G. Elliot, 'On Adaptive Recursive Filtering,' Conf. Rec. 10th. Annual Asilomar Conf. on Circ. Sys. and Comp., 1977 pp. 5-11.

[207] P. Feintuch, 'An Adaptive Recursive LMS Filter, 'Proc. IEEE, Vol. 64, No. 11, pp 1624-6, Nov. 1976.

[208] C.R. Johnson and M.G Larimore, P.L. Fientuch and N.J. Bershad. 'Comments and Additions to an Adaptive Recursive LMS Filter,' Proc. IEEE, Vol. 65, No. 9, pp 1399-1402, Sept. 1977.

[209] B. Widrow and J.M. McCool, 'Comments on an Adaptive Recursive LMS Filter,' Proc. IEEE, Vol. 65, No. 9, pp 1402-1404, Sept. 1977.

[210] D. Parikh and N. Ahmed, 'On an Adaptive Algorithm for IIR Filters,' Proc. IEEE, Vol. 66, No. 5, pp 585-588, May 1978.

[211] D. Parikh and N. Ahmed, 'Sequential Regression Considerations of an Adaptive Filtering Example,' Proc. IEEE, Vol. 66, No. 12, pp 1660-1662, Dec. 1978.

[212] R.A. David 'IIR Adaptive Algorithms Based on Gradient Search Techniques,' Ph.D. Dissertation, Stanford University, August 1981.

[213] R.A. David and S.D. Stearns, 'Adaptive IIR Algorithms Based on Gradient Search,' Proc. 24th. Midwest Symposium on Circuits and Systems, June 1981.

[214] R.A. David and S.D. Stearns, 'A Comparison of IIR Adaptive Algorithms,' Int. Conf. on Acoustics Speech and Signal Procesing, pp 180-184, 1982.

[215] M.A. Soderstrand, 'Cost and Performance Comparisons of Several Implementations of Adaptive Recursive Filters,' Conf. Record, 13th. Annual Asilomar Conf. on Circ., Sys., and Comp., Nov. 1979.

[216] L.R. McMurray 'Stability Diagram for an Adaptive Recursive LMS Filter,' 11th. Asilomar, Circuits, Systems, Computers, pp 263-7, Nov. 1977.

[217] J.R. Treichler et al, 'Simple Adaptive IIR Filtering,' IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp 118-122, April 1978.

[218] V. M. Popov, 'Hyperstability of Control Systems,' Berlin: Springer-Verlag, 1973.

[219] B. Widrow and M.E. Hoff, 'Adaptive Switching Circuits,' Wescon Conv. Rec., pt. 4, pp 96-140, 1960.

[220] B. Widrow, P.E. Mantley, L.J. Griffiths and B. Goode, 'Adaptive Antenna Systems,' Proc. IEEE, Vol. 55, pp 2143-2159, Dec. 1967.

[221] N. Wiener, 'The Extrapolation, Interpolation and Smoothing of Stationary Time Series,' MIT Press, Cambridge, Mass., 1949.

[222] B. Widrow, 'Adaptive Filters' in 'Aspects of Network and System Theory,' Edited by R.E. Kalman and N. De Claris, Pub. Holt, Rinehart and Winston, ISBN 03-0772200-6.

[223] T.P. Daniell, 'Adaptive Estimation with Mutually Correlated Training Samples,' Rept. SEL-68-083 (TR No.6778-4), Stanford Electronics Laboratories, Stanford, Calif., June 1968.

[224] R.D. Gitlin, J.E. Mazo and M.G. Taylor, 'On the Design of Gradient Algorithms for Digitally Implemented Adaptive Filters,' IEEE Trans. on Circuit Theory, Vol. CT-20, No. 2, pp 125-136, March 1973.

[225] K.P. Perry, 'A Comparison of the Effects of Component Nonidealities on the Performance of Analog and Digital LMS Adaptive Noise Cancellers,' Proc. IEEE ICASSP 81, pp 1284-1287.

[226] J.L. Moschner, 'Adaptive Filter with Clipped Data Input,' Rept. SEL-70-053 (TR No. 6796-1), Stanford Eleotronics Laboratories, Stanford, Calif., June 1970.

[227] C.R. South, 'First Report On Signal Processing in a Loud-Speaking Telephone,' Submitted to Aston Univ. Dec 1980.

[228] J.E. Paul 'Adaptive Digital Techniques for Audio Noise Cancelation,' IEEE Circuits and Systems Magazine, Vol. 1, No. 4, pp 2-7, Dec. 1979.

[229] D.L. Duttweiler 'A 12 Channel Digital Echo Canceler,' IEEE Trans. Commun.,' Vol. COM 26, pp 647-53, May 1978.

[230] O.A. Horner 'Echo Canceler with Adaptive Transversal Filter Utilising Pseudo-logarithmic Coding,' Comasat Tech. Rev., Vol. 7, No. 2, pp 393-428, 1977.

[231] D.L. Duttweiler 'Adaptive Filter Performance with Nonlinearities in the Correlation Multiplier,' IEEE Trans. on Acoustics Speech and Signal Processing, Vol. ASSP 30, No. 4, pp 578-87, Aug. 1982.

[232] D.L. Duttweiler 'A Single Chip VLSI Echo Canceler,' BSTJ, Vol. 59, No. 2, pp 149-160, Feb. 1980.

[301] C.R. South and A.V. Lewis, 'Extension Facilities an Performance of an LSI Adaptive Filter,' Int. Conf. on Acoustics Speech and Signal Processing, pp 341-344, March 1984.

[302] C.R. South, 'An Adaptive Filter in LSI,' BTTJ, Vol 3, No. 1, pp 30-46, Jan. 1985.

[303] K. Ochia, T. Araseki and T. Ogihara, 'Echo Canceller with Two Echo Path Models,' IEEE Trans. on Communications, Vol. 25, No. 6, pp 589-595, June 1977.

[304] K. Cattermole, 'Principles of Pulse Code Modulation,' Illife Books Ltd., ISBN 592 02834 8

[305] Private correspondence, 'Analogue Devices Catalogue. IC Handbook 82.'

[306] Private correspondence, 'Feranti Data Converter Handbook 1980'

[307] D.L. Richards, 'Statistical Properties of Speech Signals,' Proc. IEE, Vol. 3, No. 5, pp 941-949, May 1964.

[308] M.G. Bulmer, 'Pinciples of Statistics,' Oliver and Boyd Ltd., 1967.

[309] B.V. Rao and T. Murali, 'A New Design for Digital Adaptive Fiters,' Int. J. Electronics, Vol. 55, No. 3, pp 473-477, 1983.

[310] O.L. MacSorley, 'High-speed Arithmetic in Binary Computers,' Proc. IRE, Vol. 49, pp 67-91, Jan. 1961.

[401] C.R. South and A.V. Lewis. 'Improvements in or Relating to Digital Filters', Patent Application No. 8406846, Filed 16th March 1984.

[501] CCITT Fascicle III.3 - Rec. G.711, G.712 pp 67-74.

[502] C.E. Hoppitt 'A Prototype Echo Canceller for use in a Loudspeaking Telephone.' Memmo R13/007/82.

[601] J.W. Cooley, P.A.W. Lewis and P.D Welch, 'The Finite Fourier Transform,' IEEE Trans. Audio Electroacoustics, Vol. AU-17, pp.77-85, June 1969.

[602] 'Handbook of Mathematical Functions,' Edited by M. Abramowitz and I.A. Stegun, Dover, 1965.

[603] 'Biometrika Tables for Statisticians,' Vol. 1, Edited by E.S. Pearsons and H.O. Hartley, Cambridge, 1966.

[604] CCITT, 'Measurement of Sidtone Reference Equivalent,' Orange Book, Vol. 5, Recommendation P.73, 1977.

[605] C.E. Hoppitt, 'Telephone Sidetone Rating which includes Human Sidetone effects,' Electronics Letters, Vol 12, No. 6, pp 137-139 March 1976.

[606] A.E. Coleman, 'Sidetone and its Effects on Customer Satisfaction,' British Telecomunications Engineering, Vol. 1, pp 10-13, April 1982.

[701] 'TMS32010 AND TMS320M10 High-Performance 16/32-Bit Microcomputers,' Texas Instruments Microcomputer Series, Preliminary Data Manual, June 1982.

[702] J. Lawrence, E Thompson, W E Clarke and J Gale, 'Loudspeaking Communication Terminal and Method of Operation,' Patent Spec. 1444828, Filed 12th March 1974.

[703] P.J. See, 'Subjective Response to Depth of Voice Switching,' BT Memorandum No. R13/010/83, 1983.

[704] D.J. Finney, 'Probit Analysis,' Cambridge University Press, 1980 ISBN 0 521 080421 X.